

libyui  
3.10.0

Generated by Doxygen 1.9.1



<b>1 Notes about Initialization</b>	<b>1</b>
1.1 entry points	1
1.2 happens on initialization	1
1.3 classes	1
<b>2 Hierarchical Index</b>	<b>3</b>
2.1 Class Hierarchy	3
<b>3 Class Index</b>	<b>9</b>
3.1 Class List	9
<b>4 File Index</b>	<b>17</b>
4.1 File List	17
<b>5 Class Documentation</b>	<b>21</b>
5.1 FSize Class Reference	21
5.1.1 Detailed Description	23
5.1.2 Constructor & Destructor Documentation	23
5.1.2.1 FSize() [1/3]	24
5.1.2.2 FSize() [2/3]	24
5.1.2.3 FSize() [3/3]	24
5.1.3 Member Function Documentation	25
5.1.3.1 form()	25
5.1.3.2 operator long long()	26
5.2 ImplPtr< _Impl > Class Template Reference	26
5.2.1 Detailed Description	27
5.3 OptimizeChanges Class Reference	28
5.3.1 Detailed Description	28
5.4 SortedTreeltem< PAYLOAD > Class Template Reference	28
5.4.1 Detailed Description	29
5.4.2 Constructor & Destructor Documentation	29
5.4.2.1 SortedTreeltem()	30
5.4.3 Member Function Documentation	30
5.4.3.1 insertChildSorted()	30
5.5 Treeltem< PAYLOAD > Class Template Reference	31
5.5.1 Detailed Description	32
5.5.2 Constructor & Destructor Documentation	32
5.5.2.1 Treeltem() [1/2]	32
5.5.2.2 Treeltem() [2/2]	33
5.5.2.3 ~Treeltem()	33
5.5.3 Member Function Documentation	34

---

5.5.3.1 addChild()	34
5.5.3.2 setValue()	34
5.6 YAlignment Class Reference	35
5.6.1 Detailed Description	37
5.6.2 Member Function Documentation	37
5.6.2.1 addChild()	37
5.6.2.2 minHeight()	38
5.6.2.3 minWidth()	38
5.6.2.4 preferredHeight()	38
5.6.2.5 preferredWidth()	39
5.6.2.6 setBackgroundPixmap()	39
5.6.2.7 setSize()	40
5.6.2.8 stretchable()	40
5.7 YAlignmentPrivate Struct Reference	41
5.7.1 Detailed Description	42
5.8 YApplication Class Reference	42
5.8.1 Detailed Description	44
5.8.2 Constructor & Destructor Documentation	45
5.8.2.1 YApplication()	45
5.8.3 Member Function Documentation	45
5.8.3.1 applicationIcon()	45
5.8.3.2 applicationTitle()	46
5.8.3.3 askForExistingDirectory()	46
5.8.3.4 askForExistingFile()	46
5.8.3.5 askForSaveFileName()	47
5.8.3.6 beep()	47
5.8.3.7 busyCursor()	47
5.8.3.8 defaultFunctionKey()	48
5.8.3.9 deviceUnits()	48
5.8.3.10 findWidget()	49
5.8.3.11 glyph()	49
5.8.3.12 iconBasePath()	50
5.8.3.13 initConsoleKeyboard()	50
5.8.3.14 language()	50
5.8.3.15 layoutUnits()	51
5.8.3.16 makeScreenShot()	51
5.8.3.17 normalCursor()	51
5.8.3.18 openContextMenu()	52
5.8.3.19 openUI()	52

5.8.3.20 redrawScreen()	52
5.8.3.21 runInTerminal()	52
5.8.3.22 setConsoleFont()	53
5.8.3.23 setDefaultFunctionKey()	53
5.8.3.24 setLanguage()	54
5.8.3.25 setProductName()	54
5.8.3.26 setReverseLayout()	55
5.9 YApplicationPrivate Struct Reference	55
5.9.1 Detailed Description	56
5.10 YBarGraph Class Reference	56
5.10.1 Detailed Description	57
5.10.2 Member Function Documentation	58
5.10.2.1 addSegment()	58
5.10.2.2 doUpdate()	58
5.10.2.3 getProperty()	59
5.10.2.4 propertySet()	59
5.10.2.5 segment()	60
5.10.2.6 setLabel()	60
5.10.2.7 setProperty()	61
5.10.2.8 setSegmentColor()	62
5.10.2.9 setTextColor()	62
5.10.2.10 setValue()	63
5.11 YBarGraphMultiUpdate Class Reference	63
5.11.1 Detailed Description	63
5.11.2 Constructor & Destructor Documentation	63
5.11.2.1 YBarGraphMultiUpdate()	64
5.11.2.2 ~YBarGraphMultiUpdate()	64
5.12 YBarGraphPrivate Struct Reference	65
5.12.1 Detailed Description	65
5.13 YBarGraphSegment Class Reference	65
5.13.1 Detailed Description	66
5.13.2 Constructor & Destructor Documentation	66
5.13.2.1 YBarGraphSegment()	67
5.13.3 Member Function Documentation	67
5.13.3.1 hasSegmentColor()	67
5.13.3.2 hasTextColor()	68
5.13.3.3 label()	68
5.13.3.4 setLabel()	68
5.14 YBothDim< T > Class Template Reference	69

---

5.14.1 Detailed Description . . . . .	69
5.15 YBuiltinCaller Class Reference . . . . .	70
5.15.1 Detailed Description . . . . .	70
5.15.2 Member Function Documentation . . . . .	70
5.15.2.1 call() . . . . .	70
5.16 YBusyIndicator Class Reference . . . . .	71
5.16.1 Detailed Description . . . . .	72
5.16.2 Member Function Documentation . . . . .	72
5.16.2.1 getProperty() . . . . .	72
5.16.2.2 propertySet() . . . . .	73
5.16.2.3 setAlive() . . . . .	74
5.16.2.4 setLabel() . . . . .	74
5.16.2.5 setProperty() . . . . .	75
5.16.2.6 setTimeout() . . . . .	76
5.17 YBusyIndicatorPrivate Struct Reference . . . . .	76
5.17.1 Detailed Description . . . . .	77
5.18 YButtonBox Class Reference . . . . .	77
5.18.1 Detailed Description . . . . .	79
5.18.2 Member Function Documentation . . . . .	80
5.18.2.1 buttonsByButtonOrder() . . . . .	80
5.18.2.2 doLayout() . . . . .	81
5.18.2.3 findButton() . . . . .	81
5.18.2.4 margins() . . . . .	82
5.18.2.5 moveChild() . . . . .	82
5.18.2.6 preferredHeight() . . . . .	83
5.18.2.7 preferredWidth() . . . . .	83
5.18.2.8 sanityCheck() . . . . .	84
5.18.2.9 setLayoutPolicy() . . . . .	84
5.18.2.10 setMargins() . . . . .	85
5.18.2.11 setSanityCheckRelaxed() . . . . .	86
5.18.2.12 setSize() . . . . .	86
5.18.2.13 stretchable() . . . . .	87
5.19 YButtonBoxLayoutPolicy Struct Reference . . . . .	87
5.19.1 Detailed Description . . . . .	87
5.20 YButtonBoxMargins Struct Reference . . . . .	88
5.20.1 Detailed Description . . . . .	88
5.21 YButtonBoxPrivate Struct Reference . . . . .	88
5.21.1 Detailed Description . . . . .	89
5.22 YCancelEvent Class Reference . . . . .	89

---

5.22.1 Detailed Description	90
5.22.2 Constructor & Destructor Documentation	90
5.22.2.1 ~YCancelEvent()	90
5.23 YCheckBox Class Reference	91
5.23.1 Detailed Description	92
5.23.2 Member Function Documentation	92
5.23.2.1 getProperty()	93
5.23.2.2 propertySet()	93
5.23.2.3 setLabel()	94
5.23.2.4 setProperty()	94
5.23.2.5 setShortcutString()	95
5.23.2.6 setUseBoldFont()	96
5.23.2.7 setValue()	96
5.23.2.8 shortcutString()	97
5.23.2.9 userInputProperty()	97
5.23.2.10 value()	97
5.24 YCheckBoxFrame Class Reference	98
5.24.1 Detailed Description	99
5.24.2 Member Function Documentation	100
5.24.2.1 getProperty()	100
5.24.2.2 handleChildrenEnablement()	101
5.24.2.3 propertySet()	101
5.24.2.4 setAutoEnable()	102
5.24.2.5 setInvertAutoEnable()	103
5.24.2.6 setLabel()	103
5.24.2.7 setProperty()	104
5.24.2.8 setShortcutString()	105
5.24.2.9 setValue()	105
5.24.2.10 shortcutString()	105
5.24.2.11 userInputProperty()	106
5.24.2.12 value()	106
5.25 YCheckBoxFramePrivate Struct Reference	106
5.25.1 Detailed Description	107
5.26 YCheckBoxPrivate Struct Reference	107
5.26.1 Detailed Description	108
5.27 YChildrenManager< T > Class Template Reference	108
5.27.1 Detailed Description	110
5.27.2 Constructor & Destructor Documentation	110
5.27.2.1 YChildrenManager()	110

5.27.3 Member Function Documentation	111
5.27.3.1 add()	111
5.27.3.2 clear()	111
5.27.3.3 container()	111
5.27.3.4 contains()	112
5.27.3.5 empty()	112
5.27.3.6 firstChild()	112
5.27.3.7 remove()	112
5.28 YChildrenRejector< T > Class Template Reference	113
5.28.1 Detailed Description	114
5.28.2 Member Function Documentation	114
5.28.2.1 add()	114
5.29 YCodeLocation Class Reference	115
5.29.1 Detailed Description	115
5.29.2 Constructor & Destructor Documentation	115
5.29.2.1 YCodeLocation()	116
5.30 YColor Class Reference	116
5.30.1 Detailed Description	116
5.31 YComboBox Class Reference	117
5.31.1 Detailed Description	119
5.31.2 Constructor & Destructor Documentation	119
5.31.2.1 YComboBox()	119
5.31.3 Member Function Documentation	119
5.31.3.1 editable()	119
5.31.3.2 getProperty()	120
5.31.3.3 inputMaxLength()	120
5.31.3.4 propertySet()	121
5.31.3.5 selectedItem()	121
5.31.3.6 selectedItems()	122
5.31.3.7 selectItem()	123
5.31.3.8 setInputMaxLength()	123
5.31.3.9 setProperty()	124
5.31.3.10 setText()	124
5.31.3.11 setValidChars()	125
5.31.3.12 setValue()	125
5.31.3.13 text()	126
5.31.3.14 userInputProperty()	126
5.31.3.15 validChars()	126
5.31.3.16 value()	127



5.32 YComboBoxPrivate Struct Reference	127
5.32.1 Detailed Description	128
5.33 YCommandLine Class Reference	128
5.33.1 Detailed Description	129
5.33.2 Constructor & Destructor Documentation	129
5.33.2.1 YCommandLine()	129
5.33.3 Member Function Documentation	129
5.33.3.1 arg()	129
5.33.3.2 argc()	130
5.33.3.3 argv()	130
5.33.3.4 find()	131
5.33.3.5 operator[]()	131
5.33.3.6 remove()	132
5.33.3.7 replace()	132
5.34 YCommandLinePrivate Struct Reference	133
5.34.1 Detailed Description	133
5.35 YContextMenu Class Reference	134
5.35.1 Detailed Description	135
5.35.2 Constructor & Destructor Documentation	136
5.35.2.1 YContextMenu()	136
5.35.3 Member Function Documentation	136
5.35.3.1 addItem()	136
5.35.3.2 addItems()	137
5.35.3.3 deleteAllItems()	137
5.35.3.4 findMenuitem() [1/2]	138
5.35.3.5 findMenuitem() [2/2]	138
5.35.3.6 getProperty()	139
5.35.3.7 itemAt()	139
5.35.3.8 propertySet()	140
5.35.3.9 rebuildMenuTree()	140
5.35.3.10 resolveShortcutConflicts()	141
5.35.3.11 setProperty()	141
5.36 YContextMenuPrivate Struct Reference	142
5.36.1 Detailed Description	142
5.37 YDateField Class Reference	142
5.37.1 Detailed Description	143
5.38 YDateFieldPrivate Struct Reference	144
5.38.1 Detailed Description	144
5.39 YDebugEvent Class Reference	144

5.39.1 Detailed Description	145
5.39.2 Constructor & Destructor Documentation	145
5.39.2.1 ~YDebugEvent()	145
5.40 YDescribedItem Class Reference	146
5.40.1 Detailed Description	147
5.40.2 Member Function Documentation	147
5.40.2.1 description()	147
5.40.2.2 enabled()	147
5.40.2.3 setEnabled()	148
5.41 YDialog Class Reference	148
5.41.1 Detailed Description	151
5.41.2 Constructor & Destructor Documentation	151
5.41.2.1 YDialog()	152
5.41.2.2 ~YDialog()	152
5.41.3 Member Function Documentation	152
5.41.3.1 activate()	153
5.41.3.2 addEventFilter()	153
5.41.3.3 checkShortcuts()	153
5.41.3.4 currentDialog()	154
5.41.3.5 defaultButton()	154
5.41.3.6 deleteTopmostDialog()	154
5.41.3.7 destroy()	155
5.41.3.8 filterInvalidEvents()	155
5.41.3.9 highlight()	156
5.41.3.10 open()	156
5.41.3.11 openInternal()	157
5.41.3.12 pollEvent()	157
5.41.3.13 pollEventInternal()	158
5.41.3.14 postponeShortcutCheck()	158
5.41.3.15 recalcLayout()	159
5.41.3.16 removeEventFilter()	159
5.41.3.17 requestMultiPassLayout()	159
5.41.3.18 setDefaultButton()	160
5.41.3.19 showHelpText()	160
5.41.3.20 showRelNotesText()	161
5.41.3.21 showText()	161
5.41.3.22 waitForEvent()	162
5.41.3.23 waitForEventInternal()	163
5.42 YDialogPrivate Struct Reference	163

---

5.42.1 Detailed Description	164
5.43 YDialogSpy Class Reference	164
5.43.1 Detailed Description	165
5.43.2 Constructor & Destructor Documentation	165
5.43.2.1 YDialogSpy()	165
5.43.3 Member Function Documentation	166
5.43.3.1 exec()	166
5.43.3.2 showDialogSpy()	166
5.44 YDialogSpyPrivate Class Reference	168
5.44.1 Detailed Description	169
5.44.2 Member Function Documentation	169
5.44.2.1 addWidget()	169
5.44.2.2 selectedWidget()	170
5.44.2.3 toggleProperties()	170
5.45 YDownloadProgress Class Reference	171
5.45.1 Detailed Description	172
5.45.2 Constructor & Destructor Documentation	172
5.45.2.1 YDownloadProgress()	173
5.45.3 Member Function Documentation	173
5.45.3.1 currentFileSize()	173
5.45.3.2 getProperty()	174
5.45.3.3 propertySet()	174
5.45.3.4 setExpectedSize()	175
5.45.3.5 setFilename()	175
5.45.3.6 setLabel()	176
5.45.3.7 setProperty()	177
5.46 YDownloadProgressPrivate Struct Reference	178
5.46.1 Detailed Description	178
5.47 YDumbTab Class Reference	179
5.47.1 Detailed Description	180
5.47.2 Member Function Documentation	180
5.47.2.1 activate()	181
5.47.2.2 addItem()	181
5.47.2.3 debugLabel()	182
5.47.2.4 getProperty()	182
5.47.2.5 propertySet()	183
5.47.2.6 setProperty()	184
5.47.2.7 setShortcutString()	185
5.47.2.8 shortcutChanged()	185

5.47.2.9 shortcutString()	185
5.47.2.10 stretchable()	186
5.48 YDumbTabPrivate Struct Reference	186
5.48.1 Detailed Description	186
5.49 YEmpty Class Reference	187
5.49.1 Detailed Description	188
5.49.2 Member Function Documentation	188
5.49.2.1 preferredHeight()	188
5.49.2.2 preferredWidth()	188
5.50 YEmptyPrivate Struct Reference	189
5.50.1 Detailed Description	189
5.51 YEnvVar Class Reference	189
5.51.1 Detailed Description	189
5.52 YEvent Class Reference	190
5.52.1 Detailed Description	191
5.52.2 Constructor & Destructor Documentation	191
5.52.2.1 ~YEvent()	191
5.52.3 Member Function Documentation	192
5.52.3.1 invalidate()	192
5.52.3.2 isValid()	192
5.52.3.3 item()	192
5.52.3.4 serial()	193
5.52.3.5 widget()	193
5.53 YEventFilter Class Reference	193
5.53.1 Detailed Description	194
5.53.2 Constructor & Destructor Documentation	194
5.53.2.1 YEventFilter()	195
5.53.2.2 ~YEventFilter()	195
5.53.3 Member Function Documentation	196
5.53.3.1 filter()	196
5.54 YEventFilterPrivate Struct Reference	197
5.54.1 Detailed Description	197
5.55 YExternalWidgetFactory Class Reference	198
5.55.1 Detailed Description	198
5.55.2 Constructor & Destructor Documentation	198
5.55.2.1 YExternalWidgetFactory()	198
5.56 YExternalWidgets Class Reference	199
5.56.1 Detailed Description	199
5.56.2 Constructor & Destructor Documentation	199

5.56.2.1 YExternalWidgets()	200
5.56.3 Member Function Documentation	200
5.56.3.1 createExternalWidgetFactory()	200
5.56.3.2 externalWidgetFactory()	201
5.56.3.3 externalWidgets()	201
5.57 YExternalWidgetsTerminator Class Reference	202
5.57.1 Detailed Description	202
5.57.2 Constructor & Destructor Documentation	202
5.57.2.1 ~YExternalWidgetsTerminator()	202
5.58 YFrame Class Reference	203
5.58.1 Detailed Description	204
5.58.2 Member Function Documentation	204
5.58.2.1 getProperty()	204
5.58.2.2 propertySet()	205
5.58.2.3 setLabel()	206
5.58.2.4 setProperty()	206
5.59 YFramePrivate Struct Reference	207
5.59.1 Detailed Description	208
5.60 YGraph Class Reference	208
5.60.1 Detailed Description	209
5.60.2 Constructor & Destructor Documentation	209
5.60.2.1 YGraph() [1/2]	210
5.60.2.2 YGraph() [2/2]	210
5.60.3 Member Function Documentation	211
5.60.3.1 activatedNode()	211
5.60.3.2 getProperty()	211
5.60.3.3 propertySet()	212
5.60.3.4 renderGraph() [1/2]	212
5.60.3.5 renderGraph() [2/2]	212
5.60.3.6 setFilename()	213
5.60.3.7 setGraph()	213
5.60.3.8 setLayoutAlgorithm()	214
5.60.3.9 setProperty()	214
5.61 YGraphPlugin Class Reference	215
5.61.1 Detailed Description	216
5.61.2 Constructor & Destructor Documentation	217
5.61.2.1 ~YGraphPlugin()	217
5.61.3 Member Function Documentation	217
5.61.3.1 createGraph()	217

5.62 YGraphPrivate Struct Reference	218
5.62.1 Detailed Description	218
5.63 YHelpButtonHandler Class Reference	219
5.63.1 Detailed Description	219
5.63.2 Member Function Documentation	220
5.63.2.1 filter()	220
5.64 YIconLoader Class Reference	221
5.64.1 Detailed Description	221
5.65 YImage Class Reference	221
5.65.1 Detailed Description	222
5.65.2 Constructor & Destructor Documentation	223
5.65.2.1 YImage()	223
5.65.3 Member Function Documentation	223
5.65.3.1 hasZeroSize()	223
5.65.3.2 setAutoScale()	223
5.65.3.3 setImage()	224
5.65.3.4 setZeroSize()	224
5.66 YImagePrivate Struct Reference	225
5.66.1 Detailed Description	225
5.67 YInputField Class Reference	226
5.67.1 Detailed Description	227
5.67.2 Constructor & Destructor Documentation	228
5.67.2.1 YInputField()	228
5.67.3 Member Function Documentation	228
5.67.3.1 getProperty()	228
5.67.3.2 inputMaxLength()	229
5.67.3.3 passwordMode()	229
5.67.3.4 propertySet()	230
5.67.3.5 saveUserInput()	230
5.67.3.6 setInputMaxLength()	231
5.67.3.7 setLabel()	231
5.67.3.8 setProperty()	232
5.67.3.9 setShortcutString()	233
5.67.3.10 setShrinkable()	233
5.67.3.11 setValidChars()	234
5.67.3.12 setValue()	234
5.67.3.13 shortcutString()	234
5.67.3.14 userInputProperty()	235
5.67.3.15 validChars()	235

5.67.3.16 value()	235
5.68 YInputFieldPrivate Struct Reference	236
5.68.1 Detailed Description	236
5.69 YIntField Class Reference	237
5.69.1 Detailed Description	238
5.69.2 Constructor & Destructor Documentation	239
5.69.2.1 YIntField()	239
5.69.3 Member Function Documentation	239
5.69.3.1 enforceRange()	239
5.69.3.2 getProperty()	240
5.69.3.3 propertySet()	241
5.69.3.4 setLabel()	241
5.69.3.5 setMaxValue()	242
5.69.3.6 setMinValue()	242
5.69.3.7 setProperty()	243
5.69.3.8 setShortcutString()	244
5.69.3.9 setValue()	244
5.69.3.10 setValueInternal()	245
5.69.3.11 shortcutString()	245
5.69.3.12 userInputProperty()	245
5.69.3.13 value()	246
5.70 YIntFieldPrivate Struct Reference	246
5.70.1 Detailed Description	247
5.71 YItem Class Reference	247
5.71.1 Detailed Description	248
5.71.2 Member Function Documentation	248
5.71.2.1 childrenBegin()	249
5.71.2.2 childrenEnd()	249
5.71.2.3 label()	249
5.71.2.4 parent()	250
5.71.2.5 setData()	250
5.71.2.6 setSelected()	250
5.71.2.7 setStatus()	250
5.71.2.8 status()	251
5.72 YItemCustomStatus Class Reference	251
5.72.1 Detailed Description	252
5.72.2 Member Function Documentation	252
5.72.2.1 nextStatus()	252
5.72.2.2 setNextStatus()	253

5.72.2.3 textIndicator()	253
5.73 YItemSelector Class Reference	253
5.73.1 Detailed Description	255
5.73.2 Constructor & Destructor Documentation	255
5.73.2.1 YItemSelector()	255
5.73.3 Member Function Documentation	256
5.73.3.1 activateItem()	256
5.73.3.2 customStatus()	256
5.73.3.3 cycleCustomStatus()	256
5.73.3.4 getProperty()	257
5.73.3.5 propertySet()	257
5.73.3.6 setItemStatus()	258
5.73.3.7 setProperty()	259
5.73.3.8 setVisibleItems()	259
5.73.3.9 updateCustomStatusIndicator()	260
5.73.3.10 userInputProperty()	260
5.73.3.11 usingCustomStatus()	260
5.73.3.12 validCustomStatusIndex()	261
5.73.3.13 visibleItems()	261
5.74 YItemSelectorPrivate Struct Reference	262
5.74.1 Detailed Description	262
5.75 YItemShortcut Class Reference	263
5.75.1 Detailed Description	264
5.75.2 Member Function Documentation	264
5.75.2.1 setShortcut()	264
5.76 YKeyEvent Class Reference	265
5.76.1 Detailed Description	266
5.76.2 Constructor & Destructor Documentation	267
5.76.2.1 YKeyEvent()	267
5.76.2.2 ~YKeyEvent()	267
5.76.3 Member Function Documentation	267
5.76.3.1 focusWidget()	267
5.77 YLabel Class Reference	268
5.77.1 Detailed Description	269
5.77.2 Constructor & Destructor Documentation	269
5.77.2.1 YLabel()	270
5.77.3 Member Function Documentation	270
5.77.3.1 debugLabel()	270
5.77.3.2 getProperty()	271



5.77.3.3 isHeading()	271
5.77.3.4 isOutputField()	272
5.77.3.5 layoutPass()	272
5.77.3.6 propertySet()	272
5.77.3.7 setAutoWrap()	273
5.77.3.8 setProperty()	274
5.77.3.9 setText()	275
5.77.3.10 setUseBoldFont()	275
5.77.3.11 widgetClass()	275
5.78 YLabelPrivate Struct Reference	276
5.78.1 Detailed Description	276
5.79 YLayoutBox Class Reference	277
5.79.1 Detailed Description	279
5.79.2 Constructor & Destructor Documentation	279
5.79.2.1 YLayoutBox()	279
5.79.3 Member Function Documentation	280
5.79.3.1 countLayoutStretchChildren()	280
5.79.3.2 countStretchableChildren()	280
5.79.3.3 doResize()	281
5.79.3.4 findDominatingChild()	281
5.79.3.5 isLayoutStretch()	282
5.79.3.6 moveChild()	283
5.79.3.7 preferredHeight()	283
5.79.3.8 preferredSize()	284
5.79.3.9 preferredWidth()	284
5.79.3.10 setSize()	285
5.79.3.11 stretchable()	286
5.80 YLayoutBoxPrivate Struct Reference	286
5.80.1 Detailed Description	287
5.81 YLogView Class Reference	287
5.81.1 Detailed Description	289
5.81.2 Constructor & Destructor Documentation	289
5.81.2.1 YLogView()	289
5.81.3 Member Function Documentation	289
5.81.3.1 displayLogText()	289
5.81.3.2 getProperty()	290
5.81.3.3 maxLines()	290
5.81.3.4 propertySet()	291
5.81.3.5 setLabel()	291

---

5.81.3.6	setMaxLines()	292
5.81.3.7	setProperty()	292
5.81.3.8	setShortcutString()	293
5.81.3.9	setVisibleLines()	294
5.81.3.10	shortcutString()	294
5.82	YLogViewPrivate Struct Reference	295
5.82.1	Detailed Description	295
5.83	YMacro Class Reference	296
5.83.1	Detailed Description	296
5.83.2	Member Function Documentation	297
5.83.2.1	setPlayer()	297
5.83.2.2	setRecorder()	297
5.84	YMacroPlayer Class Reference	298
5.84.1	Detailed Description	298
5.85	YMacroRecorder Class Reference	298
5.85.1	Detailed Description	299
5.85.2	Member Function Documentation	299
5.85.2.1	recordMakeScreenShot()	299
5.86	YMenuButton Class Reference	300
5.86.1	Detailed Description	302
5.86.2	Constructor & Destructor Documentation	302
5.86.2.1	YMenuButton()	302
5.86.3	Member Function Documentation	302
5.86.3.1	activateItem()	302
5.86.3.2	addItem()	303
5.86.3.3	addItems()	304
5.86.3.4	deleteAllItems()	304
5.86.3.5	findItem() [1/2]	305
5.86.3.6	findItem() [2/2]	305
5.86.3.7	findMenuitem() [1/2]	306
5.86.3.8	findMenuitem() [2/2]	307
5.86.3.9	getProperty()	307
5.86.3.10	itemAt()	308
5.86.3.11	propertySet()	309
5.86.3.12	rebuildMenuTree()	309
5.86.3.13	resolveShortcutConflicts()	310
5.86.3.14	setProperty()	310
5.87	YMenuButtonPrivate Struct Reference	311
5.87.1	Detailed Description	311

---

5.88 YMenuEvent Class Reference	312
5.88.1 Detailed Description	313
5.88.2 Constructor & Destructor Documentation	313
5.88.2.1 ~YMenuEvent()	313
5.88.3 Member Function Documentation	313
5.88.3.1 id()	314
5.88.3.2 item()	314
5.89 YMenuItem Class Reference	314
5.89.1 Detailed Description	315
5.89.2 Constructor & Destructor Documentation	315
5.89.2.1 YMenuItem()	316
5.89.2.2 ~YMenuItem()	316
5.90 YMultiLineEdit Class Reference	316
5.90.1 Detailed Description	318
5.90.2 Member Function Documentation	318
5.90.2.1 defaultVisibleLines()	318
5.90.2.2 getProperty()	318
5.90.2.3 inputMaxLength()	319
5.90.2.4 propertySet()	319
5.90.2.5 setDefaultVisibleLines()	320
5.90.2.6 setInputMaxLength()	320
5.90.2.7 setLabel()	321
5.90.2.8 setProperty()	321
5.90.2.9 setShortcutString()	322
5.90.2.10 setValue()	323
5.90.2.11 shortcutString()	323
5.90.2.12 userInputProperty()	323
5.90.2.13 value()	324
5.91 YMultiLineEditPrivate Struct Reference	324
5.91.1 Detailed Description	325
5.92 YMultiProgressMeter Class Reference	325
5.92.1 Detailed Description	326
5.92.2 Member Function Documentation	327
5.92.2.1 currentValue()	327
5.92.2.2 doUpdate()	327
5.92.2.3 getProperty()	328
5.92.2.4 propertySet()	328
5.92.2.5 setCurrentValue()	329
5.92.2.6 setProperty()	330

5.93 YMultiProgressMeterPrivate Struct Reference	331
5.93.1 Detailed Description	331
5.94 YMultiSelectionBox Class Reference	332
5.94.1 Detailed Description	333
5.94.2 Member Function Documentation	333
5.94.2.1 currentItem()	334
5.94.2.2 getProperty()	334
5.94.2.3 propertySet()	335
5.94.2.4 saveUserInput()	335
5.94.2.5 setCurrentItem()	336
5.94.2.6 setProperty()	336
5.94.2.7 setShrinkable()	337
5.94.2.8 userInputProperty()	338
5.95 YMultiSelectionBoxPrivate Struct Reference	338
5.95.1 Detailed Description	338
5.96 YOptionalWidgetFactory Class Reference	338
5.96.1 Detailed Description	340
5.96.2 Constructor & Destructor Documentation	340
5.96.2.1 YOptionalWidgetFactory()	340
5.97 YPackageSelector Class Reference	341
5.97.1 Detailed Description	342
5.97.2 Constructor & Destructor Documentation	342
5.97.2.1 YPackageSelector()	342
5.98 YPackageSelectorPlugin Class Reference	343
5.98.1 Detailed Description	344
5.98.2 Constructor & Destructor Documentation	344
5.98.2.1 ~YPackageSelectorPlugin()	344
5.98.3 Member Function Documentation	344
5.98.3.1 createPackageSelector()	344
5.99 YPartitionSplitter Class Reference	345
5.99.1 Detailed Description	346
5.99.2 Constructor & Destructor Documentation	347
5.99.2.1 YPartitionSplitter()	347
5.99.3 Member Function Documentation	348
5.99.3.1 getProperty()	348
5.99.3.2 propertySet()	348
5.99.3.3 setProperty()	349
5.99.3.4 setValue()	350
5.99.3.5 userInputProperty()	350

5.99.3.6 value()	351
5.100 YPartitionSplitterPrivate Struct Reference	351
5.100.1 Detailed Description	352
5.101 YPath Class Reference	352
5.101.1 Detailed Description	352
5.101.2 Constructor & Destructor Documentation	353
5.101.2.1 YPath()	353
5.102 YPerThreadLogInfo Struct Reference	354
5.102.1 Detailed Description	355
5.103 YPopupInternal Class Reference	355
5.103.1 Detailed Description	355
5.103.2 Member Function Documentation	355
5.103.2.1 editNewStringArray()	355
5.103.2.2 editStringArray()	356
5.103.2.3 message()	357
5.104 YProgressBar Class Reference	357
5.104.1 Detailed Description	359
5.104.2 Member Function Documentation	359
5.104.2.1 getProperty()	359
5.104.2.2 maxValue()	360
5.104.2.3 propertySet()	360
5.104.2.4 setLabel()	361
5.104.2.5 setProperty()	361
5.104.2.6 setValue()	362
5.105 YProgressBarPrivate Struct Reference	363
5.105.1 Detailed Description	363
5.106 YProperty Class Reference	363
5.106.1 Detailed Description	364
5.106.2 Constructor & Destructor Documentation	364
5.106.2.1 YProperty()	364
5.107 YPropertyEditor Class Reference	365
5.107.1 Detailed Description	365
5.107.2 Constructor & Destructor Documentation	365
5.107.2.1 YPropertyEditor()	365
5.107.3 Member Function Documentation	366
5.107.3.1 edit()	366
5.108 YPropertyEditorPriv Class Reference	366
5.108.1 Detailed Description	366
5.109 YPropertySet Class Reference	367

5.109.1 Detailed Description	367
5.109.2 Member Function Documentation	368
5.109.2.1 add()	368
5.109.2.2 check() [1/2]	368
5.109.2.3 check() [2/2]	369
5.109.2.4 contains() [1/2]	369
5.109.2.5 contains() [2/2]	370
5.110 YPropertyValue Class Reference	370
5.110.1 Detailed Description	371
5.110.2 Member Function Documentation	371
5.110.2.1 operator!=(())	371
5.110.2.2 operator==(())	372
5.110.2.3 stringVal()	372
5.110.2.4 type()	373
5.111 YPushButton Class Reference	373
5.111.1 Detailed Description	375
5.111.2 Member Function Documentation	375
5.111.2.1 activate()	375
5.111.2.2 getProperty()	375
5.111.2.3 isDefaultButton()	376
5.111.2.4 isHelpButton()	376
5.111.2.5 isRelNotesButton()	376
5.111.2.6 propertySet()	377
5.111.2.7 setDefaultButton()	377
5.111.2.8 setFunctionKey()	378
5.111.2.9 setHelpButton()	379
5.111.2.10 setIcon()	379
5.111.2.11 setLabel()	379
5.111.2.12 setProperty()	380
5.111.2.13 setRelNotesButton()	380
5.111.2.14 setRole()	381
5.111.2.15 setShortcutString()	382
5.111.2.16 shortcutString()	382
5.112 YPushButtonPrivate Struct Reference	383
5.112.1 Detailed Description	383
5.113 YRadioButton Class Reference	384
5.113.1 Detailed Description	385
5.113.2 Constructor & Destructor Documentation	386
5.113.2.1 YRadioButton()	386

5.113.3 Member Function Documentation	386
5.113.3.1 findRadioButtonGroup()	386
5.113.3.2 getProperty()	387
5.113.3.3 propertySet()	387
5.113.3.4 saveUserInput()	388
5.113.3.5 setLabel()	389
5.113.3.6 setProperty()	389
5.113.3.7 setShortcutString()	390
5.113.3.8 setUseBoldFont()	390
5.113.3.9 setValue()	390
5.113.3.10 shortcutString()	391
5.113.3.11 userInputProperty()	391
5.113.3.12 value()	391
5.113.3.13 widgetClass()	392
5.114 YRadioButtonGroup Class Reference	392
5.114.1 Detailed Description	394
5.114.2 Member Function Documentation	394
5.114.2.1 addRadioButton()	394
5.114.2.2 getProperty()	395
5.114.2.3 propertySet()	395
5.114.2.4 radioButtonsBegin()	396
5.114.2.5 removeRadioButton()	396
5.114.2.6 setProperty()	397
5.114.2.7 uncheckOtherButtons()	398
5.115 YRadioButtonGroupPrivate Struct Reference	399
5.115.1 Detailed Description	399
5.116 YRadioButtonPrivate Struct Reference	400
5.116.1 Detailed Description	400
5.117 YRelNotesButtonHandler Class Reference	401
5.117.1 Detailed Description	401
5.117.2 Member Function Documentation	402
5.117.2.1 filter()	402
5.118 YReplacePoint Class Reference	403
5.118.1 Detailed Description	404
5.118.2 Member Function Documentation	404
5.118.2.1 showChild()	404
5.119 YRichText Class Reference	405
5.119.1 Detailed Description	407
5.119.2 Constructor & Destructor Documentation	407

5.119.2.1 YRichText()	407
5.119.3 Member Function Documentation	407
5.119.3.1 autoScrollDown()	408
5.119.3.2 getProperty()	408
5.119.3.3 hScrollValue()	409
5.119.3.4 plainTextMode()	409
5.119.3.5 propertySet()	409
5.119.3.6 setAutoScrollDown()	410
5.119.3.7 setHScrollValue()	410
5.119.3.8 setPlainTextMode()	411
5.119.3.9 setProperty()	411
5.119.3.10 setShrinkable()	412
5.119.3.11 setValue()	413
5.119.3.12 setVScrollValue()	413
5.119.3.13 shrinkable()	413
5.119.3.14 vScrollValue()	414
5.120 YRichTextPrivate Struct Reference	414
5.120.1 Detailed Description	415
5.121 YRpmGroupsTree Class Reference	415
5.121.1 Detailed Description	416
5.122 YSelectionBox Class Reference	417
5.122.1 Detailed Description	418
5.122.2 Member Function Documentation	419
5.122.2.1 getProperty()	419
5.122.2.2 immediateMode()	419
5.122.2.3 propertySet()	420
5.122.2.4 setProperty()	420
5.122.2.5 setShrinkable()	421
5.122.2.6 userInputProperty()	422
5.123 YSelectionBoxPrivate Struct Reference	422
5.123.1 Detailed Description	422
5.124 YSelectionWidget Class Reference	423
5.124.1 Detailed Description	426
5.124.2 Constructor & Destructor Documentation	426
5.124.2.1 YSelectionWidget()	426
5.124.3 Member Function Documentation	427
5.124.3.1 addItem() [1/2]	427
5.124.3.2 addItem() [2/2]	427
5.124.3.3 addItem()	428



5.124.3.4 deleteAllItems()	429
5.124.3.5 deselectAllItems()	429
5.124.3.6 findItem() [1/2]	430
5.124.3.7 findItem() [2/2]	430
5.124.3.8 findSelectedItem()	431
5.124.3.9 iconFullPath() [1/2]	431
5.124.3.10 iconFullPath() [2/2]	432
5.124.3.11 itemsBegin()	432
5.124.3.12 itemCount()	432
5.124.3.13 selectedItems()	433
5.124.3.14 selectItem()	433
5.124.3.15 setEnforceInitialSelection()	434
5.124.3.16 setEnforceSingleSelection()	434
5.124.3.17 setIconBasePath()	435
5.124.3.18 setItemStatus()	435
5.124.3.19 setLabel()	435
5.124.3.20 setShortcutString()	436
5.124.3.21 shortcutString()	436
5.125 YSelectionWidgetPrivate Struct Reference	437
5.125.1 Detailed Description	437
5.126 YSettings Class Reference	438
5.126.1 Detailed Description	438
5.126.2 Member Function Documentation	439
5.126.2.1 loadedUI() [1/2]	439
5.126.2.2 loadedUI() [2/2]	439
5.126.2.3 setIconDir()	440
5.126.2.4 setLocaleDir()	440
5.126.2.5 setProgDir()	440
5.126.2.6 setThemeDir()	440
5.127 YShortcut Class Reference	441
5.127.1 Detailed Description	443
5.127.2 Member Function Documentation	443
5.127.2.1 clearShortcut()	443
5.127.2.2 conflict()	444
5.127.2.3 distinctShortcutChars()	444
5.127.2.4 findShortcut()	444
5.127.2.5 findShortcutPos()	445
5.127.2.6 isValid()	445
5.127.2.7 normalized()	445

5.127.2.8 preferred()	446
5.127.2.9 shortcut()	446
5.127.2.10 shortcutString()	447
5.128 YShortcutManager Class Reference	447
5.128.1 Detailed Description	449
5.128.2 Member Function Documentation	449
5.128.2.1 checkShortcuts()	449
5.128.2.2 conflictCount()	450
5.128.2.3 findShortestWidget()	450
5.128.2.4 findShortestWizardButton()	451
5.128.2.5 resolveAllConflicts()	451
5.128.2.6 resolveConflict()	452
5.129 YSimpleEventHandler Class Reference	452
5.129.1 Detailed Description	453
5.129.2 Constructor & Destructor Documentation	453
5.129.2.1 ~YSimpleEventHandler()	454
5.129.3 Member Function Documentation	454
5.129.3.1 blockEvents()	454
5.129.3.2 consumePendingEvent()	454
5.129.3.3 deleteEvent()	455
5.129.3.4 deletePendingEventsFor()	455
5.129.3.5 pendingEvent()	456
5.129.3.6 sendEvent()	456
5.129.3.7 unblockEvents()	457
5.130 YSimpleInputField Class Reference	457
5.130.1 Detailed Description	459
5.130.2 Member Function Documentation	459
5.130.2.1 getProperty()	459
5.130.2.2 propertySet()	460
5.130.2.3 setLabel()	460
5.130.2.4 setProperty()	461
5.130.2.5 setShortcutString()	462
5.130.2.6 setValue()	463
5.130.2.7 shortcutString()	463
5.130.2.8 userInputProperty()	463
5.130.2.9 value()	464
5.131 YSimpleInputFieldPrivate Struct Reference	464
5.131.1 Detailed Description	465
5.132 YSingleChildContainerWidget Class Reference	465

5.132.1 Detailed Description	466
5.132.2 Member Function Documentation	466
5.132.2.1 preferredHeight()	466
5.132.2.2 preferredWidth()	467
5.132.2.3 setSize()	467
5.132.2.4 stretchable()	468
5.133 YSingleChildManager< T > Class Template Reference	468
5.133.1 Detailed Description	469
5.133.2 Member Function Documentation	470
5.133.2.1 add()	470
5.134 YSlider Class Reference	471
5.134.1 Detailed Description	472
5.134.2 Constructor & Destructor Documentation	472
5.134.2.1 YSlider()	472
5.135 YSliderPrivate Struct Reference	473
5.135.1 Detailed Description	473
5.136 YSpacing Class Reference	473
5.136.1 Detailed Description	475
5.136.2 Constructor & Destructor Documentation	475
5.136.2.1 YSpacing()	475
5.136.3 Member Function Documentation	476
5.136.3.1 dimension()	476
5.136.3.2 preferredHeight()	476
5.136.3.3 preferredWidth()	476
5.136.3.4 size() [1/2]	477
5.136.3.5 size() [2/2]	477
5.137 YSpacingPrivate Struct Reference	477
5.137.1 Detailed Description	477
5.138 YSquash Class Reference	478
5.138.1 Detailed Description	479
5.138.2 Constructor & Destructor Documentation	479
5.138.2.1 YSquash()	479
5.138.3 Member Function Documentation	480
5.138.3.1 stretchable()	480
5.139 YSquashPrivate Struct Reference	480
5.139.1 Detailed Description	481
5.140 YStringTree Class Reference	481
5.140.1 Detailed Description	483
5.140.2 Constructor & Destructor Documentation	483

5.140.2.1 YStringTree()	483
5.140.3 Member Function Documentation	484
5.140.3.1 addBranch()	484
5.140.3.2 completePath()	485
5.140.3.3 origPath()	485
5.140.3.4 path()	486
5.140.3.5 root()	487
5.140.3.6 setTextdomain()	488
5.140.3.7 translate()	488
5.140.3.8 translatedPath()	488
5.141 YStringWidgetID Class Reference	489
5.141.1 Detailed Description	490
5.141.2 Member Function Documentation	490
5.141.2.1 isEqual()	490
5.141.2.2 toString()	491
5.142 YTable Class Reference	491
5.142.1 Detailed Description	493
5.142.2 Constructor & Destructor Documentation	493
5.142.2.1 YTable()	494
5.142.3 Member Function Documentation	494
5.142.3.1 cellChanged()	494
5.142.3.2 findItem()	495
5.142.3.3 getProperty()	495
5.142.3.4 hasColumn()	496
5.142.3.5 immediateMode()	496
5.142.3.6 keepSorting()	497
5.142.3.7 propertySet()	497
5.142.3.8 setKeepSorting()	498
5.142.3.9 setProperty()	498
5.142.3.10 setTableHeader()	499
5.142.3.11 userInputProperty()	500
5.143 YTableCell Class Reference	500
5.143.1 Detailed Description	501
5.143.2 Constructor & Destructor Documentation	501
5.143.2.1 ~YTableCell()	501
5.143.3 Member Function Documentation	502
5.143.3.1 column()	502
5.143.3.2 label()	502
5.143.3.3 reparent()	502

5.143.3.4 setIconName()	503
5.143.3.5 setLabel()	503
5.143.3.6 setSortKey()	503
5.144 YTableHeader Class Reference	504
5.144.1 Detailed Description	504
5.144.2 Member Function Documentation	504
5.144.2.1 hasColumn()	505
5.145 YTableHeaderPrivate Struct Reference	505
5.145.1 Detailed Description	506
5.146 YTableItem Class Reference	506
5.146.1 Detailed Description	507
5.146.2 Constructor & Destructor Documentation	508
5.146.2.1 YTableItem() [1/2]	508
5.146.2.2 YTableItem() [2/2]	508
5.146.2.3 ~YTableItem()	509
5.146.3 Member Function Documentation	509
5.146.3.1 addCell()	509
5.146.3.2 iconName()	510
5.146.3.3 label()	510
5.147 YTablePrivate Struct Reference	511
5.147.1 Detailed Description	511
5.148 YTimeField Class Reference	512
5.148.1 Detailed Description	513
5.149 YTimeFieldPrivate Struct Reference	513
5.149.1 Detailed Description	513
5.150 YTimeoutEvent Class Reference	514
5.150.1 Detailed Description	514
5.150.2 Constructor & Destructor Documentation	515
5.150.2.1 ~YTimeoutEvent()	515
5.151 YTimezoneSelector Class Reference	515
5.151.1 Detailed Description	516
5.151.2 Constructor & Destructor Documentation	517
5.151.2.1 YTimezoneSelector()	517
5.151.3 Member Function Documentation	517
5.151.3.1 getProperty()	517
5.151.3.2 propertySet()	518
5.151.3.3 setProperty()	519
5.152 YTimezoneSelectorPrivate Class Reference	520
5.152.1 Detailed Description	520

5.153 YTransText Class Reference	520
5.153.1 Detailed Description	521
5.153.2 Member Function Documentation	521
5.153.2.1 setOrig()	521
5.153.2.2 trans()	521
5.154 YTree Class Reference	522
5.154.1 Detailed Description	524
5.154.2 Member Function Documentation	524
5.154.2.1 activate()	524
5.154.2.2 addItems()	524
5.154.2.3 currentItem()	525
5.154.2.4 findItem() [1/2]	525
5.154.2.5 findItem() [2/2]	526
5.154.2.6 getProperty()	526
5.154.2.7 immediateMode()	527
5.154.2.8 propertySet()	527
5.154.2.9 rebuildTree()	528
5.154.2.10 setProperty()	528
5.154.2.11 userInputProperty()	529
5.155 YTreeWidgetItem Class Reference	530
5.155.1 Detailed Description	531
5.155.2 Constructor & Destructor Documentation	531
5.155.2.1 YTreeWidgetItem()	531
5.155.2.2 ~TreeWidgetItem()	532
5.155.3 Member Function Documentation	532
5.155.3.1 addChild()	533
5.155.3.2 childrenBegin()	533
5.155.3.3 childrenEnd()	533
5.155.3.4 hasChildren()	533
5.155.3.5 isOpen()	534
5.155.3.6 parent()	534
5.156 YTreePrivate Struct Reference	534
5.156.1 Detailed Description	535
5.157 YUI Class Reference	535
5.157.1 Detailed Description	537
5.157.2 Member Function Documentation	538
5.157.2.1 app()	538
5.157.2.2 blockEvents()	538
5.157.2.3 builtinCaller()	538

5.157.2.4 createApplication()	539
5.157.2.5 createOptionalWidgetFactory()	539
5.157.2.6 createWidgetFactory()	539
5.157.2.7 deleteNotify()	539
5.157.2.8 ensureUICreated()	540
5.157.2.9 eventsBlocked()	540
5.157.2.10 idleLoop()	540
5.157.2.11 optionalWidgetFactory()	541
5.157.2.12 runPkgSelection()	541
5.157.2.13 sendWidgetID()	541
5.157.2.14 setBuiltinCaller()	542
5.157.2.15 setButtonOrderFromEnvironment()	542
5.157.2.16 shutdownThreads()	543
5.157.2.17 topmostConstructorHasFinished()	543
5.157.2.18 uiThreadDestructor()	544
5.157.2.19 uiThreadMainLoop()	544
5.157.2.20 unblockEvents()	545
5.157.2.21 widgetFactory()	545
5.157.3 Member Data Documentation	546
5.157.3.1 _eventsBlocked	546
5.157.3.2 _terminate_ui_thread	546
5.157.3.3 pipe_from_ui	546
5.157.3.4 pipe_to_ui	547
5.158 YUIBadPropertyArgException Class Reference	547
5.158.1 Detailed Description	548
5.158.2 Member Function Documentation	548
5.158.2.1 dumpOn()	549
5.159 YUIButtonRoleMismatchException Class Reference	549
5.159.1 Detailed Description	550
5.160 YUICantLoadAnyUIException Class Reference	551
5.160.1 Detailed Description	552
5.161 YUIDialogStackingOrderException Class Reference	552
5.161.1 Detailed Description	553
5.162 YUIException Class Reference	553
5.162.1 Detailed Description	555
5.162.2 Constructor & Destructor Documentation	556
5.162.2.1 YUIException() [1/2]	556
5.162.2.2 YUIException() [2/2]	556
5.162.3 Member Function Documentation	556

5.162.3.1 log()	556
5.162.3.2 msg()	557
5.162.3.3 what()	557
5.163 YUIIndexOutOfRangeException Class Reference	558
5.163.1 Detailed Description	559
5.163.2 Constructor & Destructor Documentation	559
5.163.2.1 YUIIndexOutOfRangeException()	559
5.163.3 Member Function Documentation	560
5.163.3.1 dumpOn()	560
5.164 YUIInvalidChildException< YWidget > Class Template Reference	560
5.164.1 Detailed Description	562
5.164.2 Member Function Documentation	562
5.164.2.1 dumpOn()	562
5.165 YUIInvalidDimensionException Class Reference	563
5.165.1 Detailed Description	564
5.166 YUIInvalidWidgetException Class Reference	564
5.166.1 Detailed Description	565
5.167 YUILoader Class Reference	566
5.167.1 Detailed Description	566
5.167.2 Member Function Documentation	566
5.167.2.1 deleteUI()	566
5.167.2.2 loadExternalWidgets()	567
5.167.2.3 loadPlugin()	567
5.167.2.4 loadUI()	568
5.168 YUILog Class Reference	569
5.168.1 Detailed Description	570
5.168.2 Member Function Documentation	570
5.168.2.1 debug()	570
5.168.2.2 instance()	570
5.168.2.3 log()	571
5.168.2.4 logFileName()	571
5.168.2.5 loggerFunction()	572
5.168.2.6 setEnableDebugLoggingHooks()	572
5.168.2.7 setLogFileName()	573
5.168.2.8 setLoggerFunction()	573
5.169 YUILogBuffer Class Reference	574
5.169.1 Detailed Description	575
5.169.2 Member Function Documentation	575
5.169.2.1 overflow()	575



---

5.169.2.2 writeBuffer()	576
5.169.2.3 xsputn()	576
5.170 YUILogPrivate Struct Reference	577
5.170.1 Detailed Description	578
5.170.2 Member Function Documentation	578
5.170.2.1 findCurrentThread()	578
5.171 YUINoDialogException Class Reference	578
5.171.1 Detailed Description	579
5.172 YUINullPointerException Class Reference	579
5.172.1 Detailed Description	580
5.173 YUIOutOfMemoryException Class Reference	581
5.173.1 Detailed Description	582
5.174 YUIPlugin Class Reference	582
5.174.1 Detailed Description	583
5.174.2 Constructor & Destructor Documentation	583
5.174.2.1 ~YUIPlugin()	583
5.174.3 Member Function Documentation	583
5.174.3.1 locateSymbol()	584
5.174.3.2 unload()	584
5.175 YUIPluginException Class Reference	585
5.175.1 Detailed Description	586
5.176 YUIPropertyException Class Reference	586
5.176.1 Detailed Description	587
5.176.2 Member Function Documentation	587
5.176.2.1 dumpOn()	587
5.177 YUIPropertyTypeMismatchException Class Reference	588
5.177.1 Detailed Description	589
5.177.2 Member Function Documentation	589
5.177.2.1 dumpOn()	590
5.178 YUISetReadOnlyPropertyException Class Reference	590
5.178.1 Detailed Description	592
5.178.2 Member Function Documentation	592
5.178.2.1 dumpOn()	592
5.179 YUISyntaxErrorException Class Reference	593
5.179.1 Detailed Description	594
5.180 YUITooManyChildrenException< YWidget > Class Template Reference	594
5.180.1 Detailed Description	595
5.180.2 Member Function Documentation	596
5.180.2.1 dumpOn()	596

5.181 YUIUnknownPropertyException Class Reference	596
5.181.1 Detailed Description	598
5.181.2 Member Function Documentation	598
5.181.2.1 dumpOn()	598
5.182 YUIUnsupportedWidgetException Class Reference	599
5.182.1 Detailed Description	600
5.183 YUIWidgetNotFoundException Class Reference	600
5.183.1 Detailed Description	601
5.184 YWidget Class Reference	601
5.184.1 Detailed Description	606
5.184.2 Member Function Documentation	606
5.184.2.1 addChild()	606
5.184.2.2 begin()	607
5.184.2.3 debugLabel()	607
5.184.2.4 dumpDialogWidgetTree()	608
5.184.2.5 end()	608
5.184.2.6 findDialog()	609
5.184.2.7 findWidget()	609
5.184.2.8 firstChild()	610
5.184.2.9 functionKey()	610
5.184.2.10 getProperty()	611
5.184.2.11 isValid()	611
5.184.2.12 notify()	612
5.184.2.13 notifyContextMenu()	612
5.184.2.14 operator new()	612
5.184.2.15 preferredHeight()	612
5.184.2.16 preferredSize()	613
5.184.2.17 preferredWidth()	613
5.184.2.18 propertySet()	613
5.184.2.19 removeChild()	614
5.184.2.20 saveUserInput()	615
5.184.2.21 sendKeyEvents()	615
5.184.2.22 setBeingDestroyed()	616
5.184.2.23 setChildrenManager()	616
5.184.2.24 setDefaultStretchable()	616
5.184.2.25 setEnabled()	617
5.184.2.26 setFunctionKey()	617
5.184.2.27 setHelpText()	617
5.184.2.28 setId()	618

5.184.2.29 setKeyboardFocus()	618
5.184.2.30 setProperty()	618
5.184.2.31 setShortcutString()	619
5.184.2.32 setSize()	620
5.184.2.33 setWidgetRep()	620
5.184.2.34 shortcutString()	621
5.184.2.35 startMultipleChanges()	621
5.184.2.36 stretchable()	621
5.184.2.37 userInputProperty()	622
5.184.2.38 weight()	622
5.185 YWidgetEvent Class Reference	622
5.185.1 Detailed Description	623
5.185.2 Constructor & Destructor Documentation	623
5.185.2.1 ~YWidgetEvent()	624
5.185.3 Member Function Documentation	624
5.185.3.1 reason()	624
5.185.3.2 widget()	624
5.186 YWidgetFactory Class Reference	625
5.186.1 Detailed Description	626
5.186.2 Constructor & Destructor Documentation	627
5.186.2.1 YWidgetFactory()	627
5.187 YWidgetID Class Reference	627
5.187.1 Detailed Description	628
5.187.2 Constructor & Destructor Documentation	628
5.187.2.1 YWidgetID()	628
5.187.3 Member Function Documentation	628
5.187.3.1 toString()	628
5.188 YWidgetPrivate Struct Reference	629
5.188.1 Detailed Description	629
5.189 YWidgetTreeItem Class Reference	630
5.189.1 Detailed Description	631
5.190 YWizard Class Reference	631
5.190.1 Detailed Description	633
5.190.2 Constructor & Destructor Documentation	634
5.190.2.1 YWizard()	634
5.190.3 Member Function Documentation	635
5.190.3.1 addMenu()	635
5.190.3.2 addMenuEntry()	635
5.190.3.3 addStep()	635

5.190.3.4 addStepHeading()	636
5.190.3.5 addTreeItem()	636
5.190.3.6 backButton()	636
5.190.3.7 contentsReplacePoint()	636
5.190.3.8 deleteSteps()	636
5.190.3.9 getProperty()	637
5.190.3.10 propertySet()	637
5.190.3.11 retranslateInternalButtons()	638
5.190.3.12 setButtonLabel()	638
5.190.3.13 setCurrentStep()	639
5.190.3.14 setDialogIcon()	639
5.190.3.15 setDialogTitle()	639
5.191 YWizardPrivate Struct Reference	639
5.191.1 Detailed Description	639
<b>6 File Documentation</b>	<b>641</b>
6.1 /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YItem.h File Reference	641
6.2 YItem.h	642
6.3 /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YTableItem.h File Reference	645
6.4 YTableItem.h	646
6.5 /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YTypes.h File Reference	650
6.5.1 Detailed Description	652
6.6 YTypes.h	652
6.7 /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YWizard.h File Reference	653
6.7.1 Detailed Description	655
6.7.2 Enumeration Type Documentation	655
6.7.2.1 YWizardMode	655
6.8 YWizard.h	655
<b>Index</b>	<b>661</b>

# Chapter 1

## Notes about Initialization

### 1.1 entry points

- `YUI::widgetFactory()`
- `YUI::ui()`

### 1.2 happens on initialization

`YUI::ui()` calls `YUI::ensureUICreated()` calls `YUILoader::loadUI` calls `YUILoader::loadPlugin`. That instantiates `YUIPlugin(string plugin_name)` and calls its `YUIPlugin::locateSymbol "_Z8createUIb"` (mangled `createUI(bool)` ) to produce a `YUI * (* createUIFunction_t)(bool withThreads)` which is called. Its result is discarded, but `YUI::_ui` gets initialized by the `YUI::YUI` ctor called by the derived ctor.

`YUIPlugin` calls `dlopen`, but `dclose` is not called by default, and actually probably never.

### 1.3 classes

- `YCommandLine`



## Chapter 2

# Hierarchical Index

### 2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

std::exception	
YUIException . . . . .	553
YUIButtonRoleMismatchException . . . . .	549
YUICantLoadAnyUIException . . . . .	551
YUIDialogStackingOrderException . . . . .	552
YUIIndexOutOfRangeException . . . . .	558
YUIInvalidChildException< YWidget > . . . . .	560
YUIInvalidDimensionException . . . . .	563
YUIInvalidWidgetException . . . . .	564
YUINoDialogException . . . . .	578
YUINullPointerException . . . . .	579
YUIOutOfMemoryException . . . . .	581
YUIPluginException . . . . .	585
YUIPropertyException . . . . .	586
YUIBadPropertyArgException . . . . .	547
YUIPropertyTypeMismatchException . . . . .	588
YUISetReadOnlyPropertyException . . . . .	590
YUIUnknownPropertyException . . . . .	596
YUISyntaxErrorException . . . . .	593
YUITooManyChildrenException< YWidget > . . . . .	594
YUIUnsupportedWidgetException . . . . .	599
YUIWidgetNotFoundException . . . . .	600
boost::noncopyable	
ImplPtr< YTablePrivate > . . . . .	26
ImplPtr< YRadioButtonGroupPrivate > . . . . .	26
ImplPtr< YMultiLineEditPrivate > . . . . .	26
ImplPtr< YDateFieldPrivate > . . . . .	26
ImplPtr< YSliderPrivate > . . . . .	26
ImplPtr< YDumbTabPrivate > . . . . .	26
ImplPtr< YTableHeaderPrivate > . . . . .	26
ImplPtr< YImagePrivate > . . . . .	26

ImplPtr< YSelectionBoxPrivate > . . . . .	26
ImplPtr< YLabelPrivate > . . . . .	26
ImplPtr< YWizardPrivate > . . . . .	26
ImplPtr< YIntFieldPrivate > . . . . .	26
ImplPtr< YFramePrivate > . . . . .	26
ImplPtr< YContextMenuPrivate > . . . . .	26
ImplPtr< YBarGraphPrivate > . . . . .	26
ImplPtr< YGraphPrivate > . . . . .	26
ImplPtr< YDialogSpyPrivate > . . . . .	26
ImplPtr< YPushButtonPrivate > . . . . .	26
ImplPtr< YProgressBarPrivate > . . . . .	26
ImplPtr< YMultiProgressMeterPrivate > . . . . .	26
ImplPtr< YItemSelectorPrivate > . . . . .	26
ImplPtr< YCommandLinePrivate > . . . . .	26
ImplPtr< YTreePrivate > . . . . .	26
ImplPtr< YPropertyEditorPriv > . . . . .	26
ImplPtr< YMultiSelectionBoxPrivate > . . . . .	26
ImplPtr< YDownloadProgressPrivate > . . . . .	26
ImplPtr< YComboBoxPrivate > . . . . .	26
ImplPtr< YTimezoneSelectorPrivate > . . . . .	26
ImplPtr< YLogViewPrivate > . . . . .	26
ImplPtr< YUILogPrivate > . . . . .	26
ImplPtr< YInputFieldPrivate > . . . . .	26
ImplPtr< YCheckBoxPrivate > . . . . .	26
ImplPtr< YEmptyPrivate > . . . . .	26
ImplPtr< YSpacingPrivate > . . . . .	26
ImplPtr< YRichTextPrivate > . . . . .	26
ImplPtr< YRadioButtonPrivate > . . . . .	26
ImplPtr< YPartitionSplitterPrivate > . . . . .	26
ImplPtr< YMenuButtonPrivate > . . . . .	26
ImplPtr< YDialogPrivate > . . . . .	26
ImplPtr< YCheckBoxFramePrivate > . . . . .	26
ImplPtr< YBusyIndicatorPrivate > . . . . .	26
ImplPtr< YApplicationPrivate > . . . . .	26
ImplPtr< YWidgetPrivate > . . . . .	26
ImplPtr< YTimeFieldPrivate > . . . . .	26
ImplPtr< YSquashPrivate > . . . . .	26
ImplPtr< YSimpleInputFieldPrivate > . . . . .	26
ImplPtr< YButtonBoxPrivate > . . . . .	26
ImplPtr< YSelectionWidgetPrivate > . . . . .	26
ImplPtr< YLayoutBoxPrivate > . . . . .	26
ImplPtr< YEventFilterPrivate > . . . . .	26
ImplPtr< YAlignmentPrivate > . . . . .	26
ImplPtr< _Impl > . . . . .	26
OptimizeChanges . . . . .	28
boost::ordered_field_operators	
FSize . . . . .	21
std::streambuf	
YUILogBuffer . . . . .	574
TreelItem< PAYLOAD > . . . . .	31
SortedTreelItem< PAYLOAD > . . . . .	28
boost::unit_steppable	
FSize . . . . .	21
YAlignmentPrivate . . . . .	41



YApplication	42
YApplicationPrivate	55
YBarGraphMultiUpdate	63
YBarGraphPrivate	65
YBarGraphSegment	65
YBothDim< T >	69
YBothDim< bool >	69
YBothDim< int >	69
YBothDim< YAlignmentType >	69
YBuiltinCaller	70
YBusyIndicatorPrivate	76
YButtonBoxLayoutPolicy	87
YButtonBoxMargins	88
YButtonBoxPrivate	88
YCheckBoxFramePrivate	106
YCheckBoxPrivate	107
YChildrenManager< T >	108
YChildrenRejector< T >	113
YSingleChildManager< T >	468
YCodeLocation	115
YColor	116
YComboBoxPrivate	127
YCommandLine	128
YCommandLinePrivate	133
YContextMenuPrivate	142
YDateFieldPrivate	144
YDialogPrivate	163
YDialogSpy	164
YDialogSpyPrivate	168
YDownloadProgressPrivate	178
YDumbTabPrivate	186
YEmptyPrivate	189
YEnvVar	189
YEvent	190
YCancelEvent	89
YDebugEvent	144
YKeyEvent	265
YMenuEvent	312
YTimeoutEvent	514
YWidgetEvent	622
YEventFilter	193
YHelpButtonHandler	219
YRelNotesButtonHandler	401
YEventFilterPrivate	197
YExternalWidgetFactory	198
YExternalWidgets	199
YExternalWidgetsTerminator	202
YFramePrivate	207
YGraphPrivate	218
YIconLoader	221
YImagePrivate	225
YInputFieldPrivate	236
YIntFieldPrivate	246

YItem	247
YDescribedItem	146
YTableItem	506
YTreeItem	530
YMenuItem	314
YWidgetTreeItem	630
YItemCustomStatus	251
YItemSelectorPrivate	262
YLabelPrivate	276
YLayoutBoxPrivate	286
YLogViewPrivate	295
YMacro	296
YMacroPlayer	298
YMacroRecorder	298
YMenuButtonPrivate	311
YMultiLineEditPrivate	324
YMultiProgressMeterPrivate	331
YMultiSelectionBoxPrivate	338
YOptionalWidgetFactory	338
YPartitionSplitterPrivate	351
YPath	352
YPerThreadLogInfo	354
YPopupInternal	355
YProgressBarPrivate	363
YProperty	363
YPropertyEditor	365
YPropertyEditorPriv	366
YPropertySet	367
YPropertyValue	370
YPushButtonPrivate	383
YRadioButtonGroupPrivate	399
YRadioButtonPrivate	400
YRichTextPrivate	414
YSelectionBoxPrivate	422
YSelectionWidgetPrivate	437
YSettings	438
YShortcut	441
YItemShortcut	263
YShortcutManager	447
YSimpleEventHandler	452
YSimpleInputFieldPrivate	464
YSliderPrivate	473
YSpacingPrivate	477
YSquashPrivate	480
YStringTree	481
YRpmGroupsTree	415
YTableCell	500
YTableHeader	504
YTableHeaderPrivate	505
YTablePrivate	511
YTimeFieldPrivate	513
YTimezoneSelectorPrivate	520
YTransText	520

YTreePrivate . . . . .	534
YUI . . . . .	535
YUILoader . . . . .	566
YUILog . . . . .	569
YUILogPrivate . . . . .	577
YUIPlugin . . . . .	582
YGraphPlugin . . . . .	215
YPackageSelectorPlugin . . . . .	343
YWidget . . . . .	601
YBarGraph . . . . .	56
YBusyIndicator . . . . .	71
YButtonBox . . . . .	77
YCheckBox . . . . .	91
YDownloadProgress . . . . .	171
YEmpty . . . . .	187
YGraph . . . . .	208
YImage . . . . .	221
YInputField . . . . .	226
YIntField . . . . .	237
YSlider . . . . .	471
YLabel . . . . .	268
YLayoutBox . . . . .	277
YLogView . . . . .	287
YMultiLineEdit . . . . .	316
YMultiProgressMeter . . . . .	325
YPackageSelector . . . . .	341
YPartitionSplitter . . . . .	345
YProgressBar . . . . .	357
YPushButton . . . . .	373
YRadioButton . . . . .	384
YRichText . . . . .	405
YSelectionWidget . . . . .	423
YComboBox . . . . .	117
YContextMenu . . . . .	134
YDumbTab . . . . .	179
YItemSelector . . . . .	253
YMenuButton . . . . .	300
YMultiSelectionBox . . . . .	332
YSelectionBox . . . . .	417
YTable . . . . .	491
YTree . . . . .	522
YSimpleInputField . . . . .	457
YDateField . . . . .	142
YTimeField . . . . .	512
YSingleChildContainerWidget . . . . .	465
YAlignment . . . . .	35
YCheckBoxFrame . . . . .	98
YDialog . . . . .	148
YFrame . . . . .	203
YRadioButtonGroup . . . . .	392
YReplacePoint . . . . .	403
YSquash . . . . .	478
YSpacing . . . . .	473

YTimezoneSelector . . . . .	515
YWizard . . . . .	631
YWidgetFactory . . . . .	625
YWidgetID . . . . .	627
YStringWidgetID . . . . .	489
YWidgetPrivate . . . . .	629
YWizardPrivate . . . . .	639

## Chapter 3

# Class Index

### 3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">FSize</a>	Store and operate on (file/package/partition) sizes . . . . .	21
<a href="#">ImplPtr&lt; _Impl &gt;</a>	Helper template class for implementation pointers (pointers to a private class or structure that hold the member variables of a higher-level class that is part of a public API) . . . . .	26
<a href="#">OptimizeChanges</a>	Helper class that calls startMultipleChanges() in its constructor and cares about the necessary call to doneMultipleChanges() when it goes out of scope . . . . .	28
<a href="#">SortedTreeItem&lt; PAYLOAD &gt;</a>	Template class for tree items that maintain sort order . . . . .	28
<a href="#">TreeItem&lt; PAYLOAD &gt;</a>	Template class for tree items that can handle tree children in a generic way - firstChild(), next() and parent() . . . . .	31
<a href="#">YAlignment</a>	Implementation of all the alignment widgets: . . . . .	35
<a href="#">YAlignmentPrivate</a>	. . . . .	41
<a href="#">YApplication</a>	Class for application-wide values and functions . . . . .	42
<a href="#">YApplicationPrivate</a>	. . . . .	55
<a href="#">YBarGraph</a>	A graph showing partitioning of a whole . . . . .	56
<a href="#">YBarGraphMultiUpdate</a>	Helper class for multiple updates to a <a href="#">YBarGraph</a> widget: This will hold back display updates until this object goes out of scope . . . . .	63
<a href="#">YBarGraphPrivate</a>	. . . . .	65
<a href="#">YBarGraphSegment</a>	One segment of a <a href="#">YBarGraph</a> . . . . .	65
<a href="#">YBothDim&lt; T &gt;</a>	Template class for two-dimensional entities, such as . . . . .	69
<a href="#">YBuiltinCaller</a>	Abstract base class for transparently calling a built-in function . . . . .	70

<a href="#">YBusyIndicator</a>	
Indicates that something is in progress and has not frozen yet	71
<a href="#">YBusyIndicatorPrivate</a>	76
<a href="#">YButtonBox</a>	
Container widget for dialog buttons that abstracts the button order depending on the desktop environment	77
<a href="#">YButtonBoxLayoutPolicy</a>	
Helper class: Layout policy for <a href="#">YButtonBox</a> widgets	87
<a href="#">YButtonBoxMargins</a>	
Helper class: Margins for <a href="#">YButtonBox</a> widgets	88
<a href="#">YButtonBoxPrivate</a>	88
<a href="#">YCancelEvent</a>	
Event to be returned upon closing a dialog with the window manager close button (or Alt-F4)	89
<a href="#">YCheckBox</a>	
A tri-state check box	91
<a href="#">YCheckBoxFrame</a>	
A frame with a check-box, may auto-disable frame contents based on the check	98
<a href="#">YCheckBoxFramePrivate</a>	106
<a href="#">YCheckBoxPrivate</a>	107
<a href="#">YChildrenManager&lt; T &gt;</a>	
Abstract base template class for children management, such as child widgets	108
<a href="#">YChildrenRejector&lt; T &gt;</a>	
Children manager that rejects all children	113
<a href="#">YCodeLocation</a>	
Helper class for UI exceptions: Store <i>FILE</i> , <i>FUNCTION</i> and <i>LINE</i>	115
<a href="#">YColor</a>	
Helper class to define an RGB color	116
<a href="#">YComboBox</a>	
ComboBox (or "drop down box", "drop down selection"); may be editable	117
<a href="#">YComboBoxPrivate</a>	127
<a href="#">YCommandLine</a>	
Utility class to access /proc/<pid>/cmdline to retrieve argc and argv	128
<a href="#">YCommandLinePrivate</a>	133
<a href="#">YContextMenu</a>	
ContextMenu: Similar to PushButton, but with several actions: Upon clicking on a ContextMenu (or activating it with the keyboard), a pop-up menu opens where the user can activate an action	134
<a href="#">YContextMenuPrivate</a>	142
<a href="#">YDateField</a>	
Input field for entering a date	142
<a href="#">YDateFieldPrivate</a>	144
<a href="#">YDebugEvent</a>	
Event to be returned upon closing a dialog with the window manager close button (or Alt-F4)	144
<a href="#">YDescribedItem</a>	
Item class that has a (possibly multi-line) description text in addition to the normal label	146
<a href="#">YDialog</a>	
A window in the desktop environment	148
<a href="#">YDialogPrivate</a>	163
<a href="#">YDialogSpy</a>	
An interactive dialog debugger: Show the structure and content of a dialog and its widgets	164
<a href="#">YDialogSpyPrivate</a>	168
<a href="#">YDownloadProgress</a>	
DownloadProgress: A progress bar that monitors downloading a file by repeatedly polling its size up to its expected size	171
<a href="#">YDownloadProgressPrivate</a>	178

<a href="#">YDumbTab</a>	
DumbTab: A very simple tab widget that can display and switch between a number of tabs, but will only deliver the "user clicked on tab " event very much like a PushButton does . . . . .	179
<a href="#">YDumbTabPrivate</a> . . . . .	186
<a href="#">YEmpty</a>	
A widget with zero size, useful as a placeholder . . . . .	187
<a href="#">YEmptyPrivate</a> . . . . .	189
<a href="#">YEnvVar</a>	
Helper class to represent an environment variable and its value . . . . .	189
<a href="#">YEvent</a>	
Abstract base class for events to be returned upon UI::UserInput() and related functions . . . . .	190
<a href="#">YEventFilter</a>	
Abstract base class to filter events . . . . .	193
<a href="#">YEventFilterPrivate</a> . . . . .	197
<a href="#">YExternalWidgetFactory</a>	
Abstract widget factory for mandatory widgets . . . . .	198
<a href="#">YExternalWidgets</a>	
Abstract base class of a libYUI Widget Extension interface . . . . .	199
<a href="#">YExternalWidgetsTerminator</a>	
Helper class to make sure the EW is properly shut down . . . . .	202
<a href="#">YFrame</a>	
A labeled framed container . . . . .	203
<a href="#">YFramePrivate</a> . . . . .	207
<a href="#">YGraph</a>	
A graph with nodes and edges, rendered with Graphviz . . . . .	208
<a href="#">YGraphPlugin</a>	
Abstract base class for simplified access to UI plugins for graph widget . . . . .	215
<a href="#">YGraphPrivate</a> . . . . .	218
<a href="#">YHelpButtonHandler</a>	
Helper class: Event filter that handles "Help" buttons . . . . .	219
<a href="#">YIconLoader</a> . . . . .	221
<a href="#">YImage</a>	
A picture, possibly animated, loaded from a file . . . . .	221
<a href="#">YImagePrivate</a> . . . . .	225
<a href="#">YInputField</a>	
InputField: General purpose one line input field for entering text and other data . . . . .	226
<a href="#">YInputFieldPrivate</a> . . . . .	236
<a href="#">YIntField</a>	
IntField: Input field for integer values . . . . .	237
<a href="#">YIntFieldPrivate</a> . . . . .	246
<a href="#">YItem</a>	
Simple item class for SelectionBox, ComboBox, MultiSelectionBox etc . . . . .	247
<a href="#">YItemCustomStatus</a>	
Class describing a non-binary status for an item . . . . .	251
<a href="#">YItemSelector</a>	
Scrollable item selector widget with not only a label for each item, but also a (possible multi-line) description and an optional icon . . . . .	253
<a href="#">YItemSelectorPrivate</a> . . . . .	262
<a href="#">YItemShortcut</a>	
Special case for widgets that can have multiple shortcuts based on items (like <a href="#">YDumbTab</a> ) . . . . .	263
<a href="#">YKeyEvent</a> . . . . .	265
<a href="#">YLabel</a>	
Implementation of the Label, Heading and OutputField widgets . . . . .	268
<a href="#">YLabelPrivate</a> . . . . .	276

<a href="#">YLayoutBox</a>	
A vertical or horizontal stacking of widgets, implementing HBox and VBox	277
<a href="#">YLayoutBoxPrivate</a>	286
<a href="#">YLogView</a>	
LogView: A scrollable (output-only) text to display a growing log, very much like the "tail -f" shell command	287
<a href="#">YLogViewPrivate</a>	295
<a href="#">YMacro</a>	
Simple access to macro recording and playing	296
<a href="#">YMacroPlayer</a>	
Abstract base class for macro player	298
<a href="#">YMacroRecorder</a>	
Abstract base class for macro recorders	298
<a href="#">YMenuButton</a>	
MenuButton: Similar to PushButton, but with several actions: Upon clicking on a MenuButton (or activating it with the keyboard), a pop-up menu opens where the user can activate an action	300
<a href="#">YMenuButtonPrivate</a>	311
<a href="#">YMenuEvent</a>	
Event to be returned upon menu selection	312
<a href="#">YMenuItem</a>	
Item class for menu items	314
<a href="#">YMultiLineEdit</a>	
A multi-line plain-text area	316
<a href="#">YMultiLineEditPrivate</a>	324
<a href="#">YMultiProgressMeter</a>	
MultiProgressMeter: Progress bar with several segments that can indicate progress individually	325
<a href="#">YMultiProgressMeterPrivate</a>	331
<a href="#">YMultiSelectionBox</a>	
A variant of <a href="#">YSelectionBox</a> where more than one item can be selected	332
<a href="#">YMultiSelectionBoxPrivate</a>	338
<a href="#">YOptionalWidgetFactory</a>	
Abstract widget factory for optional ("special") widgets	338
<a href="#">YPackageSelector</a>	
A simple wrapper for an insanely complex UI for installing software	341
<a href="#">YPackageSelectorPlugin</a>	
Abstract base class for simplified access to UI plugins for package selection	343
<a href="#">YPartitionSplitter</a>	
PartitionSplitter: A (very custom) widget for easily splitting one existing partition into two	345
<a href="#">YPartitionSplitterPrivate</a>	351
<a href="#">YPath</a>	
Finds files (e.g.	352
<a href="#">YPerThreadLogInfo</a>	
Helper class: Per-thread logging information	354
<a href="#">YPopUpInternal</a>	355
<a href="#">YProgressBar</a>	
A progress bar, showing completion of <a href="#">value()</a> out of <a href="#">maxValue()</a> parts	357
<a href="#">YProgressBarPrivate</a>	363
<a href="#">YProperty</a>	
Class for widget properties	363
<a href="#">YPropertyEditor</a>	
An internal helper class for displaying the widget property editor in the spy dialog	365
<a href="#">YPropertyEditorPriv</a>	366
<a href="#">YPropertySet</a>	
A set of properties to check names and types against	367



<a href="#">YPropertyValue</a>	
Transport class for the value of simple properties	370
<a href="#">YPushButton</a>	
A push button; may have an icon, and a F-key shortcut	373
<a href="#">YPushButtonPrivate</a>	383
<a href="#">YRadioButton</a>	
RadioButton: Widget for one-out-of-many selection	384
<a href="#">YRadioButtonGroup</a>	
A group of <a href="#">YRadioButton</a> widgets	392
<a href="#">YRadioButtonGroupPrivate</a>	399
<a href="#">YRadioButtonPrivate</a>	400
<a href="#">YRelNotesButtonHandler</a>	
Helper class: Event filter that handles "ReleaseNotes" buttons	401
<a href="#">YReplacePoint</a>	
A placeholder that can have its contents exchanged, using <a href="#">ReplaceWidget</a>	403
<a href="#">YRichText</a>	
Text formatted with simple HTML-like tags, with "links" generating events	405
<a href="#">YRichTextPrivate</a>	414
<a href="#">YRpmGroupsTree</a>	
This was an efficient storage for RPM group tags	415
<a href="#">YSelectionBox</a>	
Selection box: List box that displays a (scrollable) list of items from which the user can select exactly one	417
<a href="#">YSelectionBoxPrivate</a>	422
<a href="#">YSelectionWidget</a>	
Base class for various kinds of multi-value widgets	423
<a href="#">YSelectionWidgetPrivate</a>	437
<a href="#">YSettings</a>	
Settings for libyui	438
<a href="#">YShortcut</a>	
Helper class for shortcut management: This class holds data about the shortcut for one single widget	441
<a href="#">YShortcutManager</a>	
Helper class to manage keyboard shortcuts within one dialog and resolve keyboard shortcut conflicts	447
<a href="#">YSimpleEventHandler</a>	
Simple event handler suitable for most UIs	452
<a href="#">YSimpleInputField</a>	
Abstract base class for simple input fields with a label above the field and a text value	457
<a href="#">YSimpleInputFieldPrivate</a>	464
<a href="#">YSingleChildContainerWidget</a>	
Container widget class that manages one child	465
<a href="#">YSingleChildManager&lt; T &gt;</a>	
Children manager that can handle one single child (rejecting any more)	468
<a href="#">YSlider</a>	
Slider: Input widget for an integer value between a minimum and a maximum value	471
<a href="#">YSliderPrivate</a>	473
<a href="#">YSpacing</a>	
HSpacing, VSpacing, HStretch, VStretch	473
<a href="#">YSpacingPrivate</a>	477
<a href="#">YSquash</a>	
HSquash, VSquash HVSquash: reduce child to its preferred size	478
<a href="#">YSquashPrivate</a>	480
<a href="#">YStringTree</a>	
Abstract base class for filter views with hierarchical filter criteria - e.g., RPM group tags, MIME types	481

<a href="#">YStringWidgetID</a>	
Simple widget ID class based on strings	489
<a href="#">YTable</a>	
Table: Selection list with multiple columns	491
<a href="#">YTableCell</a>	
One cell (one column in one row) of a <a href="#">YTableItem</a>	500
<a href="#">YTableHeader</a>	
Helper class for <a href="#">YTable</a> for table column properties:	504
<a href="#">YTableHeaderPrivate</a>	505
<a href="#">YTableItem</a>	
Item class for <a href="#">YTable</a> items	506
<a href="#">YTablePrivate</a>	511
<a href="#">YTimeField</a>	
Input field for entering a time in "hh:mm:ss" format	512
<a href="#">YTimeFieldPrivate</a>	513
<a href="#">YTimeoutEvent</a>	
Event to be returned upon timeout (i.e	514
<a href="#">YTimezoneSelector</a>	
A fancy widget with a world map	515
<a href="#">YTimezoneSelectorPrivate</a>	520
<a href="#">YTransText</a>	
Helper class for translated strings: Stores a message in the original (untranslated) version along with the translation into the current locale	520
<a href="#">YTree</a>	
Tree: List box that displays a (scrollable) list of hierarchical items from which the user can select exactly one	522
<a href="#">YTreeItem</a>	
Item class for tree items	530
<a href="#">YTreePrivate</a>	534
<a href="#">YUI</a>	
Abstract base class of a libYUI user interface	535
<a href="#">YUIBadPropertyArgException</a>	547
<a href="#">YUIButtonRoleMismatchException</a>	
Exception class for "wrong button roles in YButtonBox"	549
<a href="#">YUICantLoadAnyUIException</a>	
Exception class for UI plugin load failure	551
<a href="#">YUIDialogStackingOrderException</a>	552
<a href="#">YUIException</a>	
Base class for UI Exceptions	553
<a href="#">YUIIndexOutOfRangeException</a>	
Exception class for "index out of range"	558
<a href="#">YUIInvalidChildException&lt; YWidget &gt;</a>	
Exception class for "invalid child"	560
<a href="#">YUIInvalidDimensionException</a>	
Exception class for "value other than YD_HORIZ or YD_VERT used for dimension"	563
<a href="#">YUIInvalidWidgetException</a>	
Exception class for invalid widgets	564
<a href="#">YUILoader</a>	
Class to load one of the concrete UI plug-ins: Qt, NCurses, Gtk	566
<a href="#">YUILog</a>	
UI logging	569
<a href="#">YUILogBuffer</a>	
Stream buffer class that will use the <a href="#">YUILog</a> 's logger function	574
<a href="#">YUILogPrivate</a>	577

<a href="#">YUINoDialogException</a>	578
<a href="#">YUINullPointerException</a>	
Exception class for generic null pointer exceptions	579
<a href="#">YUIOutOfMemoryException</a>	
Exception class for "out of memory"	581
<a href="#">YUIPlugin</a>	
Wrapper class for dlopen() and related	582
<a href="#">YUIPluginException</a>	
Exception class for plugin load failure	585
<a href="#">YUIPropertyException</a>	
Abstract base class for widget property exceptions	586
<a href="#">YUIPropertyTypeMismatchException</a>	
Exception class for "property type mismatch": The application tried to set a property with a wrong type	588
<a href="#">YUISetReadOnlyPropertyException</a>	
Exception class for attempt to set a read-only property	590
<a href="#">YUISyntaxErrorException</a>	593
<a href="#">YUITooManyChildrenException&lt; YWidget &gt;</a>	
Exception class for "too many children": Attempt to add a child to a widget class that can't handle children ( <a href="#">YPushButton</a> etc.) or just one child ( <a href="#">YFrame</a> , <a href="#">YDialog</a> )	594
<a href="#">YUIUnknownPropertyException</a>	
Exception class for "unknown property name": The application tried to set (or query) a property that doesn't exist	596
<a href="#">YUIUnsupportedWidgetException</a>	
Exception class for "optional widget not supported"	599
<a href="#">YUIWidgetNotFoundException</a>	
Exception class for "No widget found with that ID"	600
<a href="#">YWidget</a>	
Abstract base class of all UI widgets	601
<a href="#">YWidgetEvent</a>	622
<a href="#">YWidgetFactory</a>	
Abstract widget factory for mandatory widgets	625
<a href="#">YWidgetID</a>	
Abstract base class for widget IDs	627
<a href="#">YWidgetPrivate</a>	629
<a href="#">YWidgetTreeItem</a>	
Custom tree item class to map tree items to widgets	630
<a href="#">YWizard</a>	
A wizard is a more complex frame typically used for multi-step workflows:	631
<a href="#">YWizardPrivate</a>	639



## Chapter 4

# File Index

### 4.1 File List

Here is a list of all documented files with brief descriptions:

/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/ <b>FSize.cc</b>	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/ <b>FSize.h</b>	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/ <b>ImplPtr.h</b>	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/ <b>Notes.h</b>	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/ <b>Treeltem.h</b>	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/ <b>YAlignment.cc</b>	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/ <b>YAlignment.h</b>	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/ <b>YApplication.cc</b>	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/ <b>YApplication.h</b>	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/ <b>YBarGraph.cc</b>	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/ <b>YBarGraph.h</b>	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/ <b>YBothDim.h</b>	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/ <b>YBuiltinCaller.h</b>	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/ <b>YBusyIndicator.cc</b>	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/ <b>YBusyIndicator.h</b>	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/ <b>YButtonBox.cc</b>	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/ <b>YButtonBox.h</b>	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/ <b>YCheckBox.cc</b>	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/ <b>YCheckBox.h</b>	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/ <b>YCheckBoxFrame.cc</b>	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/ <b>YCheckBoxFrame.h</b>	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/ <b>YChildrenManager.h</b>	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/ <b>YColor.h</b>	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/ <b>YComboBox.cc</b>	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/ <b>YComboBox.h</b>	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/ <b>YCommandLine.cc</b>	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/ <b>YCommandLine.h</b>	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/ <b>YContextMenu.cc</b>	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/ <b>YContextMenu.h</b>	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/ <b>YDateField.cc</b>	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/ <b>YDateField.h</b>	??

/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YDescribedItem.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YDialog.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YDialog.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YDialogHelpers.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YDialogSpy.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YDialogSpy.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YDownloadProgress.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YDownloadProgress.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YDumbTab.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YDumbTab.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YEmpty.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YEmpty.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YEnvVar.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YEnvVar.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YEvent.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YEvent.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YEventFilter.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YEventFilter.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YExternalWidgetFactory.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YExternalWidgets.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YExternalWidgets.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YFrame.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YFrame.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YGraph.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YGraph.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YGraphPlugin.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YIconLoader.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YIconLoader.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YImage.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YImage.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YInputField.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YInputField.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YIntField.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YIntField.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YItem.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YItem.h	641
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YItemCustomStatus.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YItemSelector.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YItemSelector.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YLabel.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YLabel.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YLayoutBox.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YLayoutBox.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YLogView.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YLogView.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YMacro.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YMacro.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YMacroPlayer.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YMacroRecorder.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YMenuButton.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YMenuButton.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YMenuItem.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YMultiLineEdit.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YMultiLineEdit.h	??

/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YMultiProgressMeter.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YMultiProgressMeter.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YMultiSelectionBox.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YMultiSelectionBox.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YOptionalWidgetFactory.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YOptionalWidgetFactory.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YPackageSelector.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YPackageSelector.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YPackageSelectorPlugin.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YPartitionSplitter.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YPartitionSplitter.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YPath.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YPath.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YPopupInternal.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YPopupInternal.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YProgressBar.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YProgressBar.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YProperty.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YProperty.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YPropertyEditor.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YPropertyEditor.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YPushButton.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YPushButton.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YRadioButton.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YRadioButton.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YRadioButtonGroup.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YRadioButtonGroup.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YReplacePoint.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YReplacePoint.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YRichText.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YRichText.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YRpmGroupsTree.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YRpmGroupsTree.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YSelectionBox.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YSelectionBox.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YSelectionWidget.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YSelectionWidget.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YSettings.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YSettings.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YShortcut.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YShortcut.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YShortcutManager.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YShortcutManager.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YSimpleEventHandler.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YSimpleEventHandler.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YSimpleInputField.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YSimpleInputField.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YSingleChildContainerWidget.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YSingleChildContainerWidget.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YSlider.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YSlider.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YSpacing.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YSpacing.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YSquash.cc	??

/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YSquash.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YStringTree.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YStringTree.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YTable.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YTable.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YTableHeader.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YTableHeader.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YTableItem.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YTableItem.h	645
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YTimeField.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YTimeField.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YTimezoneSelector.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YTimezoneSelector.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YTransText.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YTree.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YTree.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YTreeltem.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YTreeltem.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YTypes.h	
Author: Stefan Hundhammer <a href="mailto:sh@suse.de">sh@suse.de</a>	650
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YUI.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YUI.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YUIException.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YUIException.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YUILoader.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YUILoader.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YUILog.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YUILog.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YUIPlugin.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YUIPlugin.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YUISymbols.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YWidget.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YWidget.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YWidget_OptimizeChanges.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YWidgetFactory.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YWidgetFactory.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YWidgetID.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YWidgetID.h	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YWizard.cc	??
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YWizard.h	
Author: Stefan Hundhammer <a href="mailto:sh@suse.de">sh@suse.de</a>	653



## Chapter 5

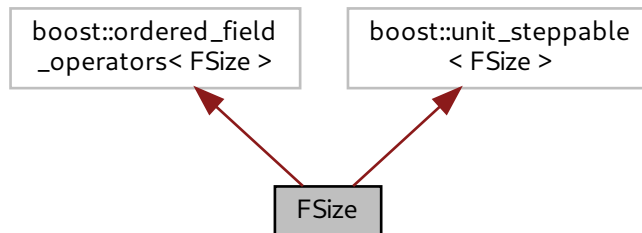
# Class Documentation

### 5.1 FSize Class Reference

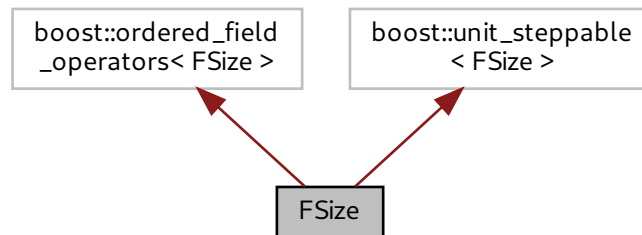
Store and operate on (file/package/partition) sizes.

```
#include <FSize.h>
```

Inheritance diagram for FSize:



Collaboration diagram for FSize:



## Public Types

- enum class [Unit](#) {  
**B**, **K**, **M**, **G**,  
**T**, **P**, **E**, **Z**,  
**Y**}

*The Units.*

## Public Member Functions

- [FSize](#) (const boost::multiprecision::cpp\_int &size\_r=0, const [Unit](#) unit\_r=Unit::B)  
*Construct from size in certain unit.*
- [FSize](#) (double size\_r)  
*Construct from size in Byte.*
- [FSize](#) (const std::string &sizeStr, const [Unit](#) unit\_r=Unit::B)  
*Construct from string containing a number in given unit.*
- [operator long long](#) () const  
*Conversions to native data types - only explicit as it might overflow If the value is out of range then the behavior depends on the boost version.*
- [operator int](#) () const
- [operator double](#) () const
- [operator boost::multiprecision::cpp\\_int](#) () const
- boost::multiprecision::cpp\_int [in\\_unit](#) (const [Unit](#) unit\_r) const
- [FSize operator-](#) () const
- [FSize & operator+=](#) (const [FSize](#) &rhs)
- [FSize & operator-=](#) (const [FSize](#) &rhs)
- [FSize & operator\\*=](#) (const [FSize](#) &rhs)
- [FSize & operator/=](#) (const [FSize](#) &rhs)
- bool [operator<](#) (const [FSize](#) &rhs) const
- bool [operator==](#) (const [FSize](#) &rhs) const
- [FSize & operator++](#) ()

- **FSize** & **operator--** ()
- **FSize** & **fillBlock** (**FSize** blocksize\_r=boost::multiprecision::cpp\_int(KB))  
*Adjust size to multiple of blocksize\_r*
- **FSize** **fullBlock** (**FSize** blocksize\_r=boost::multiprecision::cpp\_int(KB)) const  
*Return a new size adjusted to multiple of blocksize\_r*
- **Unit** **bestUnit** () const  
*Return the best unit for string representation.*
- std::string **form** (const **Unit** unit\_r, unsigned fw=0, unsigned prec=**bestPrec**, const bool showunit=true) const  
*Return string representation in given Unit.*
- std::string **form** (unsigned fw=0, unsigned prec=**bestPrec**, const bool showunit=true) const  
*Return string representation in bestUnit.*
- std::string **asString** () const  
*Default string representation (precision 1 and unit appended).*

## Static Public Member Functions

- static boost::multiprecision::cpp\_int **factor** (const **Unit** unit\_r)  
*Return ammount of bytes in Unit.*
- static const char \* **unit** (const **Unit** unit\_r)  
*String representation of Unit.*

## Static Public Attributes

- static const boost::multiprecision::cpp\_int **KB** = 1024
- static const boost::multiprecision::cpp\_int **MB** = FSize::KB \* 1024
- static const boost::multiprecision::cpp\_int **GB** = FSize::MB \* 1024
- static const boost::multiprecision::cpp\_int **TB** = FSize::GB \* 1024
- static const boost::multiprecision::cpp\_int **PB** = FSize::TB \* 1024
- static const boost::multiprecision::cpp\_int **EB** = FSize::PB \* 1024
- static const boost::multiprecision::cpp\_int **ZB** = FSize::EB \* 1024
- static const boost::multiprecision::cpp\_int **YB** = FSize::ZB \* 1024
- static const unsigned **bestPrec** = (unsigned)-1  
*Used as precision argument to **form()**, the 'best' precision according to Unist is chosen.*

### 5.1.1 Detailed Description

Store and operate on (file/package/partition) sizes.

Definition at line 45 of file **FSize.h**.

### 5.1.2 Constructor & Destructor Documentation

### 5.1.2.1 FSize() [1/3]

```
FSize::FSize (
    const boost::multiprecision::cpp_int & size_r = 0,
    const Unit unit_r = Unit::B ) [inline]
```

Construct from size in certain unit.

E.g. `FSize( 1, FSize::Unit::K )` makes 1024 Byte.

Definition at line 121 of file [FSize.h](#).

### 5.1.2.2 FSize() [2/3]

```
FSize::FSize (
    double size_r ) [inline]
```

Construct from size in Byte.

#### Parameters

<i>size</i> ↔ _r	the initial value
---------------------	-------------------

Definition at line 129 of file [FSize.h](#).

### 5.1.2.3 FSize() [3/3]

```
FSize::FSize (
    const std::string & sizeStr,
    const Unit unit_r = Unit::B )
```

Construct from string containing a number in given unit.

#### Parameters

<i>sizeStr</i>	input string - must contain <i>only</i> numbers
<i>unit</i> ↔ _r	optional unit, bytes by default

#### Exceptions

<i>std::runtime_error</i>	if the string contains any non numeric characters, even a trailing white space!
---------------------------	---

Definition at line 50 of file [FSize.cc](#).

### 5.1.3 Member Function Documentation

#### 5.1.3.1 form()

```
std::string FSize::form (
    const Unit unit_r,
    unsigned fw = 0,
    unsigned prec = bestPrec,
    const bool showunit = true ) const
```

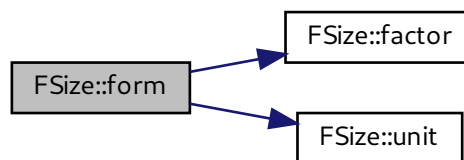
Return string representation in given Unit.

Parameter `fw` and `prec` denote field width and precision as in a `"%*.f"` printf format string. A value of `bestPrec` automatically picks an appropriate precision depending on the unit. If `showunit` is true, the string representation of Unit is *appended separated by a single blank*.

*If Unit is **Byte**, precision is set to zero.*

Definition at line 111 of file [FSize.cc](#).

Here is the call graph for this function:



### 5.1.3.2 operator long long()

```
FSize::operator long long ( ) const [inline], [explicit]
```

Conversions to native data types - only explicit as it might overflow If the value is out of range then the behavior depends on the boost version.

- 1.67 - the min/max values for the corresponding type are returned
- 1.66 - returns just the lower bits

Definition at line 148 of file [FSize.h](#).

The documentation for this class was generated from the following files:

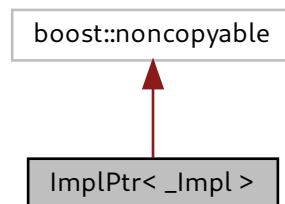
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/FSize.h
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/FSize.cc

## 5.2 ImplPtr< \_Impl > Class Template Reference

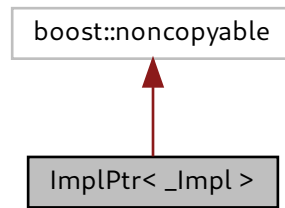
Helper template class for implementation pointers (pointers to a private class or structure that hold the member variables of a higher-level class that is part of a public API).

```
#include <ImplPtr.h>
```

Inheritance diagram for ImplPtr< \_Impl >:



Collaboration diagram for ImplPtr< \_Impl >:



## Public Types

- typedef `_Impl` **element\_type**

## Public Member Functions

- **ImplPtr** (`_Impl *impl_r=0`)
- void **reset** (`_Impl *impl_r=0`)
- void **swap** ([ImplPtr](#) rhs)
- **operator bool** () const
- const `_Impl` & **operator\*** () const
- const `_Impl` \* **operator->** () const
- const `_Impl` \* **get** () const
- `_Impl` & **operator\*** ()
- `_Impl` \* **operator->** ()
- `_Impl` \* **get** ()

### 5.2.1 Detailed Description

```
template<class _Impl>
class ImplPtr< _Impl >
```

Helper template class for implementation pointers (pointers to a private class or structure that hold the member variables of a higher-level class that is part of a public API).

This pointer class maintains constness of its parent class, i.e. if it is used in a const class the class this pointer points to will also be const.

This class automatically deletes the class it points to in its destructor.

Definition at line 42 of file [ImplPtr.h](#).

The documentation for this class was generated from the following file:

- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/ImplPtr.h`

## 5.3 OptimizeChanges Class Reference

Helper class that calls `startMultipleChanges()` in its constructor and cares about the necessary call to `doneMultipleChanges()` when it goes out of scope.

```
#include <YWidget_OptimizeChanges.h>
```

### Public Member Functions

- `OptimizeChanges` ([YWidget](#) &`w`)

#### 5.3.1 Detailed Description

Helper class that calls `startMultipleChanges()` in its constructor and cares about the necessary call to `doneMultipleChanges()` when it goes out of scope.

Definition at line 44 of file [YWidget\\_OptimizeChanges.h](#).

The documentation for this class was generated from the following file:

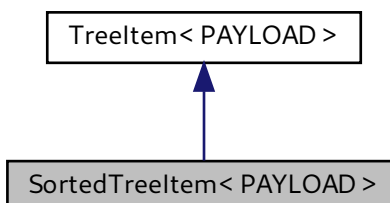
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YWidget_OptimizeChanges.h`

## 5.4 SortedTreeltem< PAYLOAD > Class Template Reference

Template class for tree items that maintain sort order.

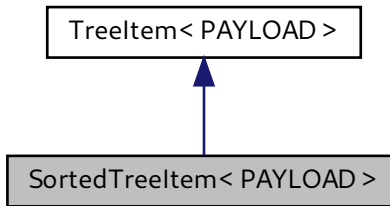
```
#include <TreeItem.h>
```

Inheritance diagram for `SortedTreeltem< PAYLOAD >`:





Collaboration diagram for SortedTreeltem< PAYLOAD >:



## Public Member Functions

- [SortedTreeltem](#) (PAYLOAD val, [SortedTreeltem](#)< PAYLOAD > \*parentItem=0)  
*Constructor.*
- virtual [~SortedTreeltem](#) ()  
*Destructor.*
- void [insertChildSorted](#) ([SortedTreeltem](#)< PAYLOAD > \*newChild)  
*Insert a child into the internal children list in ascending sort order.*
- [SortedTreeltem](#)< PAYLOAD > \* [parent](#) () const  
*Returns this item's parent or 0 if there is none.*
- [SortedTreeltem](#)< PAYLOAD > \* [next](#) () const  
*Returns this item's next sibling or 0 if there is none.*
- [SortedTreeltem](#)< PAYLOAD > \* [firstChild](#) () const  
*Returns this item's first child or 0 if there is none.*

## Additional Inherited Members

### 5.4.1 Detailed Description

```
template<class PAYLOAD>
class SortedTreeltem< PAYLOAD >
```

Template class for tree items that maintain sort order.

Class 'PAYLOAD' to provide operator<() in addition to what template '[Treeltem](#)' requires.

Definition at line [191](#) of file [Treeltem.h](#).

### 5.4.2 Constructor & Destructor Documentation

### 5.4.2.1 SortedTreeItem()

```
template<class PAYLOAD >
SortedTreeItem< PAYLOAD >::SortedTreeItem (
    PAYLOAD val,
    SortedTreeItem< PAYLOAD > * parentItem = 0 ) [inline]
```

Constructor.

Creates a new tree item with value "val" and inserts it in ascending sort order into the children list of "parent".

Definition at line 180 of file [TreeItem.h](#).

## 5.4.3 Member Function Documentation

### 5.4.3.1 insertChildSorted()

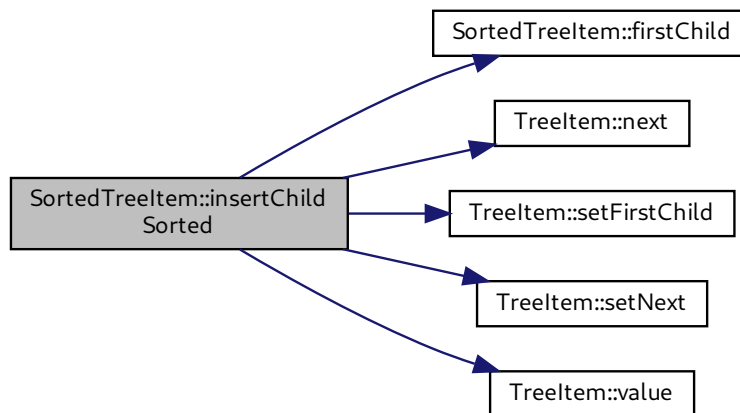
```
template<class PAYLOAD >
void SortedTreeItem< PAYLOAD >::insertChildSorted (
    SortedTreeItem< PAYLOAD > * newChild ) [inline]
```

Insert a child into the internal children list in ascending sort order.

Called from the new child's constructor, thus 'public'.

Definition at line 227 of file [TreeItem.h](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

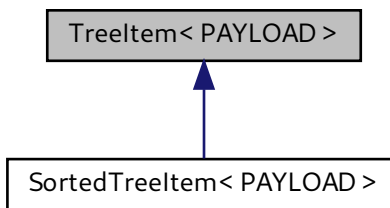
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/TreeItem.h](#)

## 5.5 TreelItem< PAYLOAD > Class Template Reference

Template class for tree items that can handle tree children in a generic way - [firstChild\(\)](#), [next\(\)](#) and [parent\(\)](#).

```
#include <TreeItem.h>
```

Inheritance diagram for TreelItem< PAYLOAD >:



### Public Member Functions

- [TreelItem](#) (const PAYLOAD &val, [TreelItem](#)< PAYLOAD > \*parent=0)  
*Constructor.*
- virtual [~TreelItem](#) ()  
*Destructor.*
- const PAYLOAD & [value](#) () const  
*Returns this item's value, the "payload".*
- void [setValue](#) (PAYLOAD newValue)  
*Set this item's value, the "payload".*
- [TreelItem](#)< PAYLOAD > \* [parent](#) () const  
*Returns this item's parent or 0 if there is none.*
- [TreelItem](#)< PAYLOAD > \* [next](#) () const  
*Returns this item's next sibling or 0 if there is none.*
- [TreelItem](#)< PAYLOAD > \* [firstChild](#) () const  
*Returns this item's first child or 0 if there is none.*
- void [setParent](#) ([TreelItem](#)< PAYLOAD > \*newParent)  
*Sets this item's parent.*
- void [setNext](#) ([TreelItem](#)< PAYLOAD > \*newNext)  
*Sets this item's next sibling.*
- void [setFirstChild](#) ([TreelItem](#)< PAYLOAD > \*newFirstChild)  
*Sets this item's first child.*
- void [addChild](#) ([TreelItem](#)< PAYLOAD > \*newChild)  
*Add a child to the internal children list - usually called from within the child's default constructor.*

## Protected Member Functions

- [TreeItem](#) (PAYLOAD val, bool autoAddChild, [TreeItem](#)< PAYLOAD > \*parent=0)

*Constructor to be called for derived classes: Decide whether or not to automatically insert this item into the parent's children list.*

## Protected Attributes

- PAYLOAD \_value
- [TreeItem](#)< PAYLOAD > \* \_parent
- [TreeItem](#)< PAYLOAD > \* \_next
- [TreeItem](#)< PAYLOAD > \* \_firstChild

### 5.5.1 Detailed Description

```
template<class PAYLOAD>
class TreeItem< PAYLOAD >
```

Template class for tree items that can handle tree children in a generic way - [firstChild\(\)](#), [next\(\)](#) and [parent\(\)](#).

Each item stores one value of type 'PAYLOAD'.

Class 'PAYLOAD' needs to provide operator=().

Definition at line 40 of file [TreeItem.h](#).

### 5.5.2 Constructor & Destructor Documentation

#### 5.5.2.1 [TreeItem\(\)](#) [1/2]

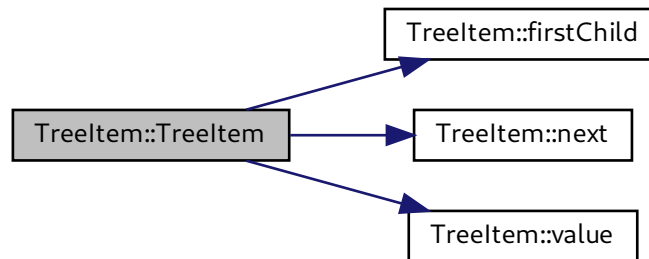
```
template<class PAYLOAD >
TreeItem< PAYLOAD >::TreeItem (
    const PAYLOAD & val,
    TreeItem< PAYLOAD > * parent = 0 ) [inline]
```

Constructor.

Creates a new tree item with value "val" and inserts it ( without maintaining any meaningful sort order! ) into the children list of "parent".

Definition at line 298 of file [TreeItem.h](#).

Here is the call graph for this function:



### 5.5.2.2 TreeItem() [2/2]

```

template<class PAYLOAD >
TreeItem< PAYLOAD >::TreeItem (
    PAYLOAD val,
    bool autoAddChild,
    TreeItem< PAYLOAD > * parent = 0 ) [inline], [protected]
  
```

Constructor to be called for derived classes: Decide whether or not to automatically insert this item into the parent's children list.

Useful for derived classes that want to maintain a specific sort order among children.

Definition at line [298](#) of file [TreeItem.h](#).

### 5.5.2.3 ~TreeItem()

```

template<class PAYLOAD >
virtual TreeItem< PAYLOAD >::~~TreeItem ( ) [inline], [virtual]
  
```

Destructor.

Takes care of children - they will be deleted along with this item.

Definition at line [88](#) of file [TreeItem.h](#).

### 5.5.3 Member Function Documentation

#### 5.5.3.1 addChild()

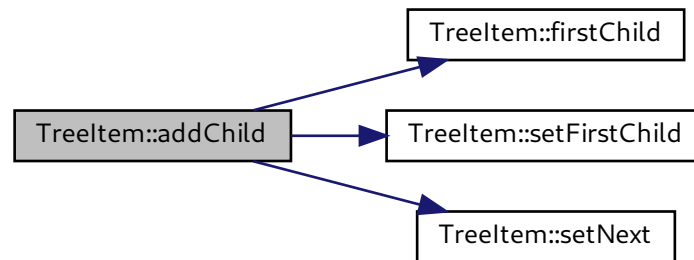
```
template<class PAYLOAD >
void TreeItem< PAYLOAD >::addChild (
    TreeItem< PAYLOAD > * newChild ) [inline]
```

Add a child to the internal children list - usually called from within the child's default constructor.

This default method does not maintain any meaningful sorting order - derived classes that require this might want to use the other constructor ( with 'autoAddChild' set to 'false' ) take care of child insertion themselves.

Definition at line 165 of file [TreeItem.h](#).

Here is the call graph for this function:



#### 5.5.3.2 setValue()

```
template<class PAYLOAD >
void TreeItem< PAYLOAD >::setValue (
    PAYLOAD newValue ) [inline]
```

Set this item's value, the "payload".

If the sort order among children of one level is important, overwrite this method and change the sort order according to the new value. The template class itself never calls this.

Definition at line 122 of file [TreeItem.h](#).

The documentation for this class was generated from the following file:

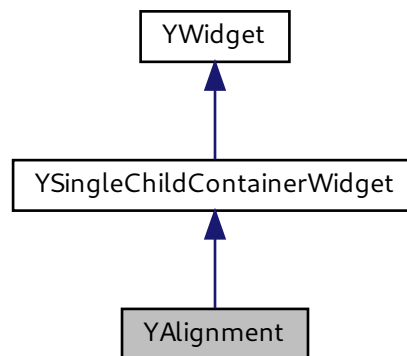
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/TreeItem.h`

## 5.6 YAlignment Class Reference

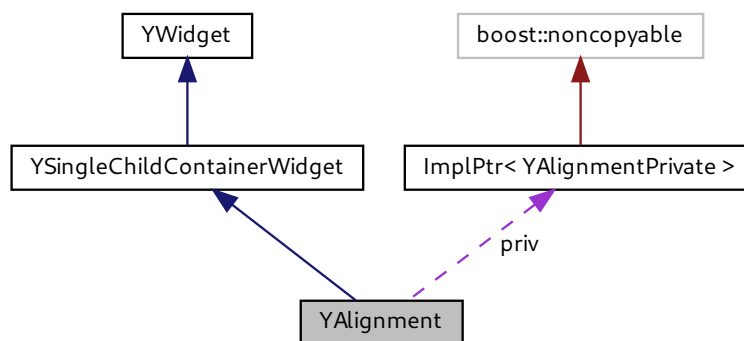
Implementation of all the alignment widgets:

```
#include <YAlignment.h>
```

Inheritance diagram for YAlignment:



Collaboration diagram for YAlignment:



### Public Member Functions

- virtual `~YAlignment()`

*Destructor.*

- virtual const char \* [widgetClass](#) () const  
*Return a descriptive name of this widget class for logging, debugging etc.*
- YAlignmentType [alignment](#) (YUIDimension dim) const  
*Return the alignment in the specified dimension.*
- int [leftMargin](#) () const  
*Return the left margin in pixels, the distance between the left edge of this alignment and the left edge of the child widget.*
- int [rightMargin](#) () const  
*Return the right margin in pixels, the distance between the right edge of this alignment and the right edge of the child widget.*
- int [topMargin](#) () const  
*Return the top margin in pixels, the distance between the top edge of this alignment and the top edge of the child widget.*
- int [bottomMargin](#) () const  
*Return the bottom margin in pixels, the distance between the bottom edge of this alignment and the bottom edge of the child widget.*
- int [totalMargins](#) (YUIDimension dim) const  
*Return the sum of all margins in the specified dimension.*
- void [setLeftMargin](#) (int margin)  
*Set the left margin in pixels.*
- void [setRightMargin](#) (int margin)  
*Set the right margin in pixels.*
- void [setTopMargin](#) (int margin)  
*Set the top margin in pixels.*
- void [setBottomMargin](#) (int margin)  
*Set the bottom margin in pixels.*
- int [minWidth](#) () const  
*Return the minimum width of this alignment or 0 if none is set.*
- int [minHeight](#) () const  
*Return the minimum height of this alignment or 0 if none is set.*
- void [setMinWidth](#) (int width)  
*Set the minimum width to return for [preferredWidth\(\)](#).*
- void [setMinHeight](#) (int height)  
*Set the minimum height to return for [preferredHeight\(\)](#).*
- virtual void [setBackgroundPixmap](#) (const std::string &pixmapFileName)  
*Set a background pixmap.*
- std::string [backgroundPixmap](#) () const  
*Return the name of the background pixmap or an empty string, if there is none.*
- virtual void [addChild](#) (YWidget \*child)  
*Add a child widget.*
- virtual void [moveChild](#) (YWidget \*child, int newx, int newy)=0  
*Move a child widget to a new position.*
- virtual bool [stretchable](#) (YUIDimension dim) const  
*Return this widget's stretchability.*
- virtual int [preferredWidth](#) ()  
*Preferred width of the widget.*
- virtual int [preferredHeight](#) ()  
*Preferred height of the widget.*
- virtual void [setSize](#) (int newWidth, int newHeight)  
*Set the current size and move the child widget according to its alignment.*



## Protected Member Functions

- [YAlignment](#) ([YWidget](#) \*parent, YAlignmentType horAlign, YAlignmentType vertAlign)  
*Constructor.*

## Protected Attributes

- [ImplPtr](#)< [YAlignmentPrivate](#) > **priv**

### 5.6.1 Detailed Description

Implementation of all the alignment widgets:

- Left, Right, HCenter,
- Top, Bottom, VCenter,
- HVCenter
- MinSize, MinWidth, MinHeight

Definition at line 41 of file [YAlignment.h](#).

### 5.6.2 Member Function Documentation

#### 5.6.2.1 addChild()

```
void YAlignment::addChild (
    YWidget * child ) [virtual]
```

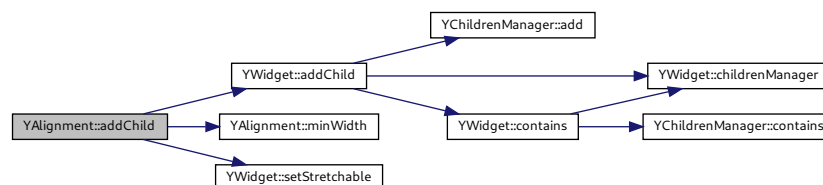
Add a child widget.

Reimplemented from [YSingleChildContainerWidget](#) to propagate stretchability down to the single child.

Reimplemented from [YWidget](#).

Definition at line 176 of file [YAlignment.cc](#).

Here is the call graph for this function:



### 5.6.2.2 minHeight()

```
int YAlignment::minHeight ( ) const
```

Return the minimum height of this alignment or 0 if none is set.

[preferredHeight\(\)](#) will never return less than this value.

Definition at line 152 of file [YAlignment.cc](#).

### 5.6.2.3 minWidth()

```
int YAlignment::minWidth ( ) const
```

Return the minimum width of this alignment or 0 if none is set.

[preferredWidth\(\)](#) will never return less than this value.

Definition at line 146 of file [YAlignment.cc](#).

### 5.6.2.4 preferredHeight()

```
int YAlignment::preferredHeight ( ) [virtual]
```

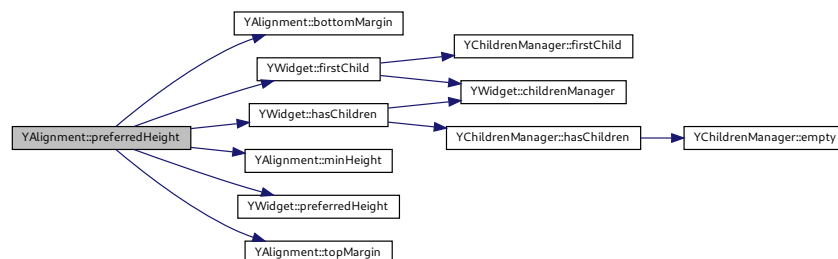
Preferred height of the widget.

Reimplemented from [YWidget](#).

Reimplemented from [YSingleChildContainerWidget](#).

Definition at line 206 of file [YAlignment.cc](#).

Here is the call graph for this function:



### 5.6.2.5 preferredWidth()

```
int YAlignment::preferredWidth ( ) [virtual]
```

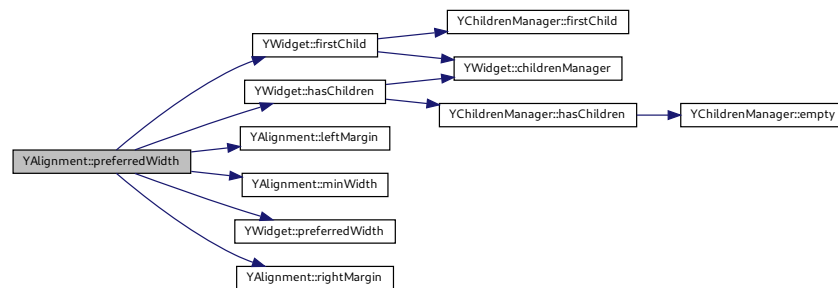
Preferred width of the widget.

Reimplemented from [YWidget](#).

Reimplemented from [YSingleChildContainerWidget](#).

Definition at line 194 of file [YAlignment.cc](#).

Here is the call graph for this function:



### 5.6.2.6 setBackgroundPixmap()

```
void YAlignment::setBackgroundPixmap (
    const std::string & pixmapFileName ) [virtual]
```

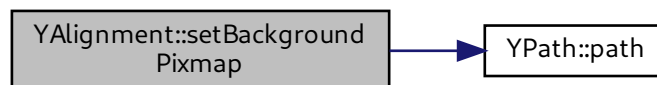
Set a background pixmap.

Derived classes may want to overwrite this.

This parent method should be called first in the overwritten method to ensure path expansion is done as specified (prepend the theme path ("`/usr/share/libyui/theme/`") if the path doesn't start with "/" or ".").

Definition at line 333 of file [YAlignment.cc](#).

Here is the call graph for this function:



### 5.6.2.7 setSize()

```
void YAlignment::setSize (
    int newWidth,
    int newHeight ) [virtual]
```

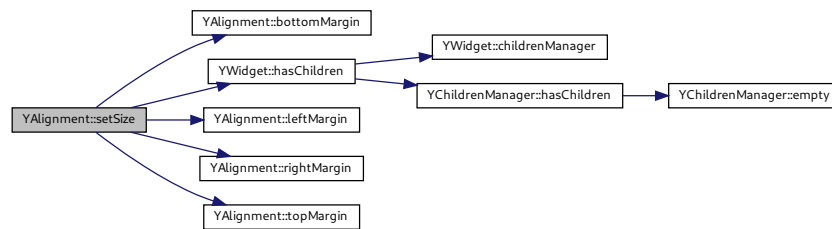
Set the current size and move the child widget according to its alignment.

Derived classes should reimplement this, but call this base class function in their own implementation.

Reimplemented from [YSingleChildContainerWidget](#).

Definition at line 218 of file [YAlignment.cc](#).

Here is the call graph for this function:



### 5.6.2.8 stretchable()

```
bool YAlignment::stretchable (
    YUIDimension dim ) const [virtual]
```

Return this widget's stretchability.

Reimplemented from [YWidget](#).

In an aligned dimension the widget is always stretchable. In an unchanged dimension the widget is stretchable if the child is stretchable.

Reimplemented from [YSingleChildContainerWidget](#).

Definition at line 185 of file [YAlignment.cc](#).

Here is the call graph for this function:

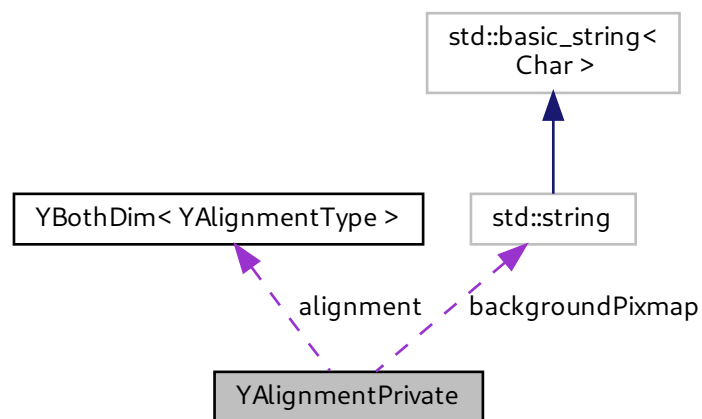


The documentation for this class was generated from the following files:

- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YAlignment.h`
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YAlignment.cc`

## 5.7 YAlignmentPrivate Struct Reference

Collaboration diagram for YAlignmentPrivate:



### Public Member Functions

- [YAlignmentPrivate](#) (YAlignmentType horAlign, YAlignmentType vertAlign)  
*Constructor.*

## Public Attributes

- int **leftMargin**
- int **rightMargin**
- int **topMargin**
- int **bottomMargin**
- int **minWidth**
- int **minHeight**
- string **backgroundPixmap**
- [YBothDim](#)< YAlignmentType > **alignment**

### 5.7.1 Detailed Description

Definition at line 37 of file [YAlignment.cc](#).

The documentation for this struct was generated from the following file:

- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YAlignment.cc

## 5.8 YApplication Class Reference

Class for application-wide values and functions.

```
#include <YApplication.h>
```

### Public Member Functions

- [YWidget](#) \* **findWidget** ([YWidgetID](#) \*id, bool doThrow=true) const  
*Find a widget in the topmost dialog by its ID.*
- virtual std::string **iconBasePath** () const  
*Get the base path for icons used by the UI.*
- virtual void **setIconBasePath** (const std::string &newIconBasePath)  
*Set the icon base path.*
- [YIconLoader](#) \* **iconLoader** ()
- int **defaultFunctionKey** (const std::string &label) const  
*Return the default function key number for a widget with the specified label or 0 if there is none.*
- void **setDefaultFunctionKey** (const std::string &label, int fkey)  
*Add a mapping from the specified label to the specified F-key number.*
- void **clearDefaultFunctionKeys** ()  
*Clear all previous label-to-function-key mappings.*
- virtual void **setLanguage** (const std::string &language, const std::string &encoding=std::string())  
*Set language and encoding for the locale environment (\$LANG).*
- std::string **language** (bool stripEncoding=false) const  
*Return the current language from the locale environment (\$LANG).*
- virtual std::string **glyph** (const std::string &glyphSymbolName)

*Return a string for a named glyph:*

- virtual std::string [askForExistingDirectory](#) (const std::string &startDir, const std::string &headline)=0

*Open a directory selection box and prompt the user for an existing directory.*

- virtual std::string [askForExistingFile](#) (const std::string &startWith, const std::string &filter, const std::string &headline)=0

*Open a file selection box and prompt the user for an existing file.*

- virtual std::string [askForSaveFileName](#) (const std::string &startWith, const std::string &filter, const std::string &headline)=0

*Open a file selection box and prompt the user for a file to save data to.*

- virtual bool [openContextMenu](#) (const [YItemCollection](#) &itemCollection)

*Open a context menu for a widget.*

- virtual void [setProductName](#) (const std::string &productName)

*Set the current product name ("openSUSE", "SLES", ...).*

- std::string [productName](#) () const

*Get the current product name ("openSUSE", "SLES", ...).*

- void [setReleaseNotes](#) (const std::map< std::string, std::string > &relNotes)

*Set release notes; map product => text.*

- std::map< std::string, std::string > [releaseNotes](#) () const

*Get the current release notes map.*

- void [setShowProductLogo](#) (bool show)

*Set whether the product logo (in top bar) should be shown.*

- bool [showProductLogo](#) () const

*Return true if product logo should be shown.*

- virtual int [deviceUnits](#) (YUIDimension dim, float [layoutUnits](#))

*Convert logical layout spacing units into device dependent units.*

- virtual float [layoutUnits](#) (YUIDimension dim, int [deviceUnits](#))

*Convert device dependent units into logical layout spacing units.*

- virtual void [setReverseLayout](#) (bool reverse)

*Set reverse layout for Arabic / Hebrew support.*

- bool [reverseLayout](#) () const

*Returns 'true' if widget geometry should be reversed for languages that have right-to-left writing direction (Arabic, Hebrew).*

- virtual void [busyCursor](#) ()

*Change the (mouse) cursor to indicate busy status.*

- virtual void [normalCursor](#) ()

*Change the (mouse) cursor back from busy status to normal.*

- virtual void [makeScreenShot](#) (const std::string &fileName)

*Make a screen shot and save it to the specified file.*

- virtual void [beep](#) ()

*Beep.*

- virtual void [redrawScreen](#) ()

*Redraw the screen.*

- virtual void [initConsoleKeyboard](#) ()

*Initialize the (text) console keyboard.*

- virtual void [setConsoleFont](#) (const std::string &console\_magic, const std::string &font, const std::string &screen←\_map, const std::string &unicode\_map, const std::string &language)

*Set the (text) console font according to the current encoding etc.*

- virtual int [runInTerminal](#) (const std::string &command)

*Run a shell command (typically an interactive program using NCurses) in a terminal (window).*

- virtual int **displayWidth** ()=0
- virtual int **displayHeight** ()=0
- virtual int **displayDepth** ()=0
- virtual long **displayColors** ()=0
- virtual int **defaultWidth** ()=0
- virtual int **defaultHeight** ()=0
- virtual bool **isTextMode** ()=0
- virtual bool **hasImageSupport** ()=0
- virtual bool **hasIconSupport** ()=0
- virtual bool **hasAnimationSupport** ()=0
- virtual bool **hasFullUtf8Support** ()=0
- virtual bool **richTextSupportsTable** ()=0
- virtual bool **leftHandedMouse** ()=0
- virtual bool **hasWizardDialogSupport** ()
- virtual void **setApplicationTitle** (const std::string &title)

*Set the application title.*

- virtual const std::string & **applicationTitle** () const

*Get the application title.*

- virtual void **setApplicationIcon** (const std::string &icon)

*Set the application icon.*

- virtual const std::string & **applicationIcon** () const

*Get the application icon.*

- virtual void **openUI** ()

*To mix TUI (NCurses) with stdio, enclose the UI parts within openUI/closeUI.*

- virtual void **closeUI** ()

## Protected Member Functions

- **YApplication** ()

*Constructor.*

- virtual **~YApplication** ()

*Destructor.*

## Friends

- class **YUI**

### 5.8.1 Detailed Description

Class for application-wide values and functions.

This is a singleton. Access and create it via the static functions in **YUI**.

Definition at line 45 of file **YApplication.h**.



## 5.8.2 Constructor & Destructor Documentation

### 5.8.2.1 YApplication()

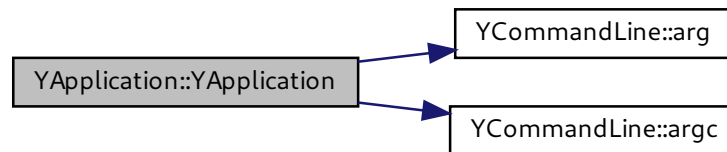
```
YApplication::YApplication ( ) [protected]
```

Constructor.

Use [YUI::app\(\)](#) to get the singleton for this class.

Definition at line 64 of file [YApplication.cc](#).

Here is the call graph for this function:



## 5.8.3 Member Function Documentation

### 5.8.3.1 applicationIcon()

```
const string & YApplication::applicationIcon ( ) const [virtual]
```

Get the application icon.

Default icon is an empty string

Definition at line 297 of file [YApplication.cc](#).

### 5.8.3.2 applicationTitle()

```
const string & YApplication::applicationTitle ( ) const [virtual]
```

Get the application title.

Default title is the running command (argv[0])

Definition at line 288 of file [YApplication.cc](#).

### 5.8.3.3 askForExistingDirectory()

```
virtual std::string YApplication::askForExistingDirectory (
    const std::string & startDir,
    const std::string & headline ) [pure virtual]
```

Open a directory selection box and prompt the user for an existing directory.

'startDir' is the initial directory that is displayed.

'headline' is an explanatory text for the directory selection box. Graphical UIs may omit that if no window manager is running.

Returns the selected directory name or an empty string if the user canceled the operation.

Derived classes are required to implement this.

### 5.8.3.4 askForExistingFile()

```
virtual std::string YApplication::askForExistingFile (
    const std::string & startWith,
    const std::string & filter,
    const std::string & headline ) [pure virtual]
```

Open a file selection box and prompt the user for an existing file.

'startWith' is the initial directory or file.

'filter' is one or more blank-separated file patterns, e.g. "\*.png \*.jpg"

'headline' is an explanatory text for the file selection box. Graphical UIs may omit that if no window manager is running.

Returns the selected file name or an empty string if the user canceled the operation.

Derived classes are required to implement this.

### 5.8.3.5 askForSaveFileName()

```
virtual std::string YApplication::askForSaveFileName (
    const std::string & startWith,
    const std::string & filter,
    const std::string & headline ) [pure virtual]
```

Open a file selection box and prompt the user for a file to save data to.

Automatically asks for confirmation if the user selects an existing file.

'startWith' is the initial directory or file.

'filter' is one or more blank-separated file patterns, e.g. "\*.png \*.jpg"

'headline' is an explanatory text for the file selection box. Graphical UIs may omit that if no window manager is running.

Returns the selected file name or an empty string if the user canceled the operation.

Derived classes are required to implement this.

### 5.8.3.6 beep()

```
virtual void YApplication::beep ( ) [inline], [virtual]
```

Beep.

This default implementation does nothing.

Definition at line [327](#) of file [YApplication.h](#).

### 5.8.3.7 busyCursor()

```
virtual void YApplication::busyCursor ( ) [inline], [virtual]
```

Change the (mouse) cursor to indicate busy status.

This default implementation does nothing.

Definition at line [309](#) of file [YApplication.h](#).

### 5.8.3.8 defaultFunctionKey()

```
int YApplication::defaultFunctionKey (
    const std::string & label ) const
```

Return the default function key number for a widget with the specified label or 0 if there is none.

Any keyboard shortcuts that may be contained in 'label' are stripped away before any comparison.

The basic idea behind this concept is to have an easy default mapping from buttons etc. with the same semantics to function keys:

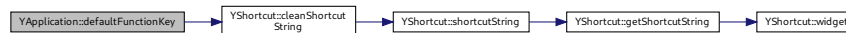
"OK" -> F10 "Accept" -> F10 "Yes" -> F10 "Next" -> F10

"Cancel" -> F9 "No" -> F9 ...

This function returns 10 for F10, F for F9 etc.; 0 means "no function key".

Definition at line 163 of file [YApplication.cc](#).

Here is the call graph for this function:



### 5.8.3.9 deviceUnits()

```
int YApplication::deviceUnits (
    YUIDimension dim,
    float layoutUnits ) [virtual]
```

Convert logical layout spacing units into device dependent units.

A default size dialog is assumed to be 80x25 layout spacing units.

Derived classes may want to reimplement this method.

Definition at line 262 of file [YApplication.cc](#).

Here is the call graph for this function:



### 5.8.3.10 findWidget()

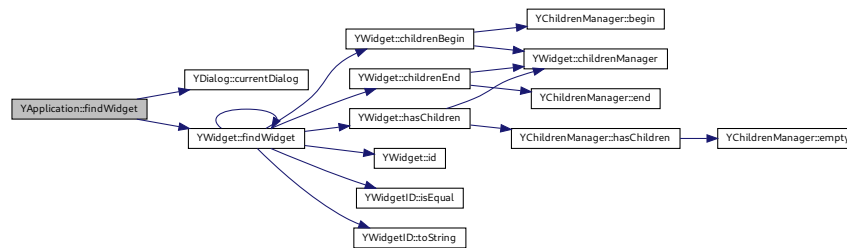
```
YWidget * YApplication::findWidget (
    YWidgetID * id,
    bool doThrow = true ) const
```

Find a widget in the topmost dialog by its ID.

If there is no widget with that ID (or no dialog at all), this function throws a [YUIWidgetNotFoundException](#) if 'doThrow' is 'true'. It returns 0 if 'doThrow' is 'false'.

Definition at line 82 of file [YApplication.cc](#).

Here is the call graph for this function:



### 5.8.3.11 glyph()

```
string YApplication::glyph (
    const std::string & glyphSymbolName ) [virtual]
```

Return a string for a named glyph:

YUIGlyph\_ArrowLeft YUIGlyph\_ArrowRight YUIGlyph\_ArrowUp YUIGlyph\_ArrowDown YUIGlyph\_CheckMark  
 YUIGlyph\_BulletArrowRight YUIGlyph\_BulletCircle YUIGlyph\_BulletSquare

Using this is discouraged in new applications. This method is available for backward compatibility.

This default implementation returns simple textual representations for each glyph symbol (e.g., "<->" for YUIGlyph\_ArrowLeft<->Right).

Derived classes are free to overwrite this. It does not make sense to call this base class method in a new implementation.

Definition at line 235 of file [YApplication.cc](#).

Here is the call graph for this function:



#### 5.8.3.12 iconBasePath()

```
string YApplication::iconBasePath ( ) const [virtual]
```

Get the base path for icons used by the UI.

Selection widgets like [YSelectionBox](#), [YComboBox](#), etc. or [YWizard](#) prepend this to icon specifications that don't use an absolute path.

Definition at line 94 of file [YApplication.cc](#).

#### 5.8.3.13 initConsoleKeyboard()

```
virtual void YApplication::initConsoleKeyboard ( ) [inline], [virtual]
```

Initialize the (text) console keyboard.

This default implementation does nothing.

Definition at line 344 of file [YApplication.h](#).

#### 5.8.3.14 language()

```
string YApplication::language (
    bool stripEncoding = false ) const
```

Return the current language from the locale environment (\$LANG).

If 'stripEncoding' is true, any encoding (".utf8" etc.) is removed.

Definition at line 211 of file [YApplication.cc](#).

### 5.8.3.15 layoutUnits()

```
float YApplication::layoutUnits (
    YUIDimension dim,
    int deviceUnits ) [virtual]
```

Convert device dependent units into logical layout spacing units.

A default size dialog is assumed to be 80x25 layout spacing units.

Derived classes may want to reimplement this method.

Definition at line 269 of file [YApplication.cc](#).

Here is the call graph for this function:



### 5.8.3.16 makeScreenShot()

```
virtual void YApplication::makeScreenShot (
    const std::string & fileName ) [inline], [virtual]
```

Make a screen shot and save it to the specified file.

This default implementation does nothing.

Definition at line 321 of file [YApplication.h](#).

### 5.8.3.17 normalCursor()

```
virtual void YApplication::normalCursor ( ) [inline], [virtual]
```

Change the (mouse) cursor back from busy status to normal.

This default implementation does nothing.

Definition at line 315 of file [YApplication.h](#).

#### 5.8.3.18 openContextMenu()

```
bool YApplication::openContextMenu (
    const YItemCollection & itemCollection ) [virtual]
```

Open a context menu for a widget.

'itemCollection' describes the menu structure

Returns true on success (otherwise false).

Derived classes are free to overwrite this.

Definition at line 253 of file [YApplication.cc](#).

#### 5.8.3.19 openUI()

```
virtual void YApplication::openUI ( ) [inline], [virtual]
```

To mix TUI (NCurses) with stdio, enclose the UI parts within openUI/closeUI.

This default implementation does nothing.

Definition at line 379 of file [YApplication.h](#).

#### 5.8.3.20 redrawScreen()

```
virtual void YApplication::redrawScreen ( ) [inline], [virtual]
```

Redraw the screen.

This default implementation does nothing.

Definition at line 338 of file [YApplication.h](#).

#### 5.8.3.21 runInTerminal()

```
int YApplication::runInTerminal (
    const std::string & command ) [virtual]
```

Run a shell command (typically an interactive program using NCurses) in a terminal (window).

This is useful for text UIs (e.g., NCurses) that need special preparation prior to running an NCurses-based application and special clean-up afterwards.

This default implementation logs an error and returns -1.

Definition at line 276 of file [YApplication.cc](#).



### 5.8.3.22 setConsoleFont()

```
virtual void YApplication::setConsoleFont (
    const std::string & console_magic,
    const std::string & font,
    const std::string & screen_map,
    const std::string & unicode_map,
    const std::string & language ) [inline], [virtual]
```

Set the (text) console font according to the current encoding etc.

See the `setFont(8)` command and the console `HowTo` for details.

This default implementation does nothing.

Definition at line 352 of file [YApplication.h](#).

### 5.8.3.23 setDefaultFunctionKey()

```
void YApplication::setDefaultFunctionKey (
    const std::string & label,
    int fkey )
```

Add a mapping from the specified label to the specified F-key number.

This is the counterpart to [defaultFunctionKey\(\)](#).

This only affects widgets that are created after this call.

Definition at line 176 of file [YApplication.cc](#).

Here is the call graph for this function:



#### 5.8.3.24 setLanguage()

```
void YApplication::setLanguage (
    const std::string & language,
    const std::string & encoding = std::string() ) [virtual]
```

Set language and encoding for the locale environment (\$LANG).

This affects UI-internal translations (e.g. for predefined dialogs like file selection), encoding and fonts.

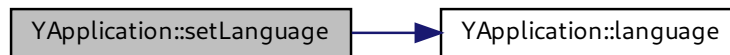
'language' is the ISO short code ("de\_DE", "en\_US", ...).

'encoding' an (optional) encoding ("utf8", ...) that will be appended if present.

Derived classes can overwrite this method, but they should call this base class method at the beginning of the new implementation.

Definition at line 193 of file [YApplication.cc](#).

Here is the call graph for this function:



#### 5.8.3.25 setProductName()

```
void YApplication::setProductName (
    const std::string & productName ) [virtual]
```

Set the current product name ("openSUSE", "SLES", ...).

This name will be expanded in help texts when the entity is used.

Derived classes can overwrite this method, but they should call this base class method in the new implementation.

Definition at line 113 of file [YApplication.cc](#).

Here is the call graph for this function:



### 5.8.3.26 setReverseLayout()

```
void YApplication::setReverseLayout (
    bool reverse ) [virtual]
```

Set reverse layout for Arabic / Hebrew support.

Derived classes can overwrite this method, but they should call this base class method in the new implementation.

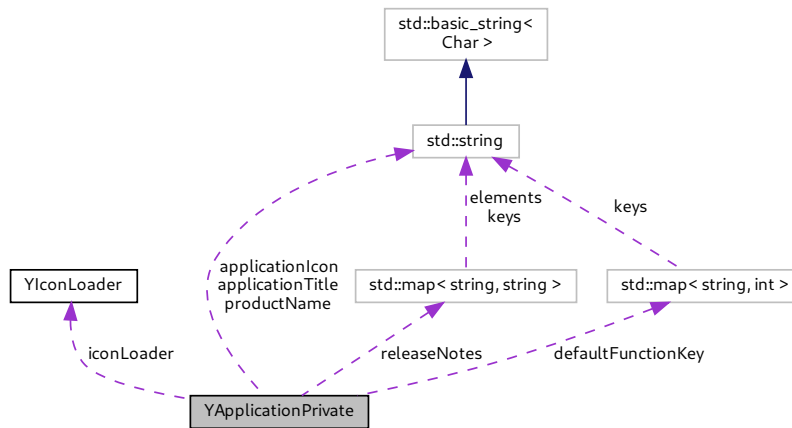
Definition at line 150 of file [YApplication.cc](#).

The documentation for this class was generated from the following files:

- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YApplication.h
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YApplication.cc

## 5.9 YApplicationPrivate Struct Reference

Collaboration diagram for YApplicationPrivate:



### Public Attributes

- string **productName**
- bool **reverseLayout**
- string **applicationTitle**
- string **applicationIcon**
- YFunctionKeyMap **defaultFunctionKey**
- [YIconLoader](#) \* **iconLoader**
- map< string, string > **releaseNotes**
- bool **showProductLogo**

### 5.9.1 Detailed Description

Definition at line 45 of file [YApplication.cc](#).

The documentation for this struct was generated from the following file:

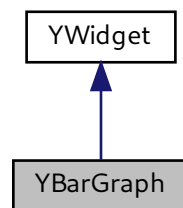
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YApplication.cc](#)

## 5.10 YBarGraph Class Reference

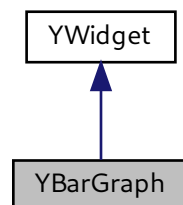
A graph showing partitioning of a whole.

```
#include <YBarGraph.h>
```

Inheritance diagram for YBarGraph:



Collaboration diagram for YBarGraph:



## Public Member Functions

- virtual [~YBarGraph](#) ()  
*Destructor.*
- virtual const char \* [widgetClass](#) () const  
*Return a descriptive name of this widget class for logging, debugging etc.*
- void [addSegment](#) (const [YBarGraphSegment](#) &[segment](#))  
*Add one segment.*
- void [deleteAllSegments](#) ()  
*Delete all segments.*
- int [segments](#) ()  
*Return the current number of segments.*
- const [YBarGraphSegment](#) & [segment](#) (int segmentIndex) const  
*Return the segment with the specified index (from 0 on).*
- void [setValue](#) (int segmentIndex, int newValue)  
*Set the value of the segment with the specified index (from 0 on).*
- void [setLabel](#) (int segmentIndex, const std::string &newLabel)  
*Set the label of the segment with the specified index (from 0 on).*
- void [setSegmentColor](#) (int segmentIndex, const [YColor](#) &color)  
*Set the background color of the segment with the specified index (from 0 on).*
- void [setTextColor](#) (int segmentIndex, const [YColor](#) &color)  
*Set the text color of the segment with the specified index (from 0 on).*
- virtual bool [setProperty](#) (const std::string &propertyName, const [YPropertyValue](#) &val)  
*Set a property.*
- virtual [YPropertyValue](#) [getProperty](#) (const std::string &propertyName)  
*Get a property.*
- virtual const [YPropertySet](#) & [propertySet](#) ()  
*Return this class's property set.*

## Protected Member Functions

- [YBarGraph](#) ([YWidget](#) \*parent)  
*Constructor.*
- virtual void [doUpdate](#) ()=0  
*Perform a display update after any change to any of the segments.*

## Friends

- class [YBarGraphMultiUpdate](#)

### 5.10.1 Detailed Description

A graph showing partitioning of a whole.

The whole is divided into [YBarGraphSegment](#) each of which has a relative size, a text color, a background color, and a label.

Definition at line 40 of file [YBarGraph.h](#).

## 5.10.2 Member Function Documentation

### 5.10.2.1 addSegment()

```
void YBarGraph::addSegment (
    const YBarGraphSegment & segment )
```

Add one segment.

If the segment's background and text colors are not explicitly specified, the [YBarGraph](#) widget will assign them from a list of (at least 5 different) color sets.

When adding multiple segments, use a [YBarGraphMultiUpdate](#) object for improved performance to hold back display updates until all segments are added.

Definition at line 97 of file [YBarGraph.cc](#).

Here is the call graph for this function:



### 5.10.2.2 doUpdate()

```
virtual void YBarGraph::doUpdate ( ) [protected], [pure virtual]
```

Perform a display update after any change to any of the segments.

Derived classes are required to implement this.

### 5.10.2.3 getProperty()

```
YPropertyValue YBarGraph::getProperty (
    const std::string & propertyName ) [virtual]
```

Get a property.

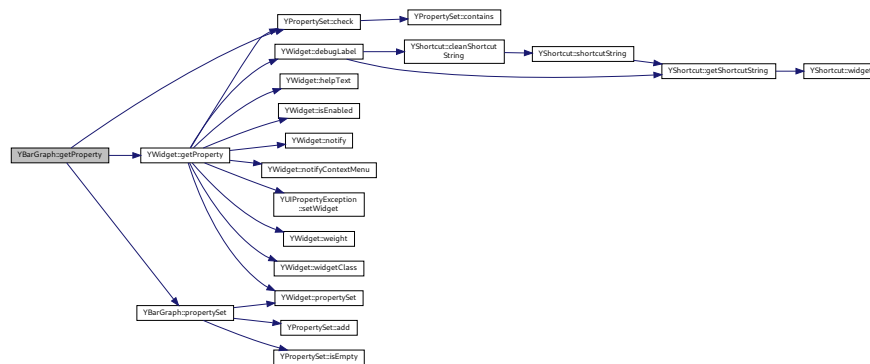
Reimplemented from [YWidget](#).

This method may throw YUIPropertyExceptions.

Reimplemented from [YWidget](#).

Definition at line 212 of file [YBarGraph.cc](#).

Here is the call graph for this function:



### 5.10.2.4 propertySet()

```
const YPropertySet & YBarGraph::propertySet ( ) [virtual]
```

Return this class's property set.

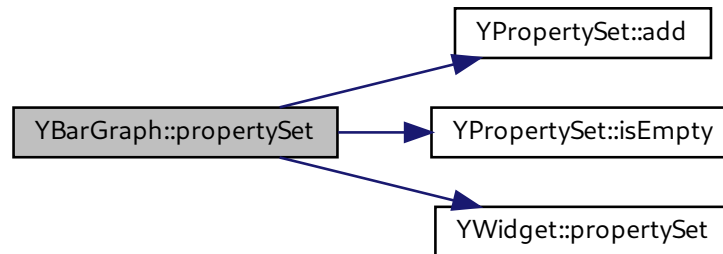
This also initializes the property upon the first call.

Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 175 of file [YBarGraph.cc](#).

Here is the call graph for this function:



#### 5.10.2.5 `segment()`

```
const YBarGraphSegment & YBarGraph::segment (
    int segmentIndex ) const
```

Return the segment with the specified index (from 0 on).

This will throw an exception if there are not this many segments.

Definition at line 113 of file [YBarGraph.cc](#).

#### 5.10.2.6 `setLabel()`

```
void YBarGraph::setLabel (
    int segmentIndex,
    const std::string & newLabel )
```

Set the label of the segment with the specified index (from 0 on).

Use %1 as a placeholder for the current value.

This will throw an exception if there are not this many segments.

Note: Use a [YBarGraphMultiUpdate](#) object for improved performance when doing multiple changes at the same time.

Definition at line 139 of file [YBarGraph.cc](#).



### 5.10.2.7 setProperty()

```
bool YBarGraph::setProperty (
    const std::string & propertyName,
    const YPropertyValue & val ) [virtual]
```

Set a property.

Reimplemented from [YWidget](#).

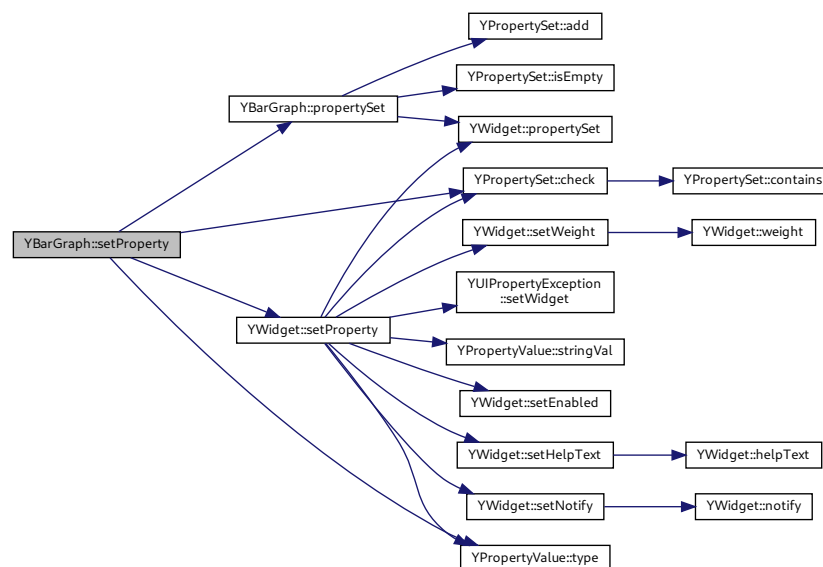
This function may throw YUIPropertyExceptions.

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from [YWidget](#).

Definition at line 196 of file [YBarGraph.cc](#).

Here is the call graph for this function:



### 5.10.2.8 setSegmentColor()

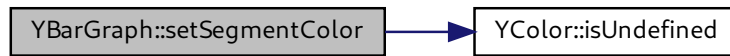
```
void YBarGraph::setSegmentColor (
    int segmentIndex,
    const YColor & color )
```

Set the background color of the segment with the specified index (from 0 on).

This will throw an exception if there are not this many segments or if the color is undefined.

Definition at line 149 of file [YBarGraph.cc](#).

Here is the call graph for this function:



### 5.10.2.9 setTextColor()

```
void YBarGraph::setTextColor (
    int segmentIndex,
    const YColor & color )
```

Set the text color of the segment with the specified index (from 0 on).

This will throw an exception if there are not this many segments or if the color is undefined.

Definition at line 162 of file [YBarGraph.cc](#).

Here is the call graph for this function:



#### 5.10.2.10 setValue()

```
void YBarGraph::setValue (
    int segmentIndex,
    int newValue )
```

Set the value of the segment with the specific index (from 0 on).

This will throw an exception if there are not this many segments.

Note: Use a [YBarGraphMultiUpdate](#) object for improved performance when doing multiple changes at the same time.

Definition at line 129 of file [YBarGraph.cc](#).

The documentation for this class was generated from the following files:

- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YBarGraph.h
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YBarGraph.cc

## 5.11 YBarGraphMultiUpdate Class Reference

Helper class for multiple updates to a [YBarGraph](#) widget: This will hold back display updates until this object goes out of scope.

```
#include <YBarGraph.h>
```

### Public Member Functions

- [YBarGraphMultiUpdate](#) ([YBarGraph](#) \*barGraph)  
*Constructor.*
- [~YBarGraphMultiUpdate](#) ()  
*Destructor.*

#### 5.11.1 Detailed Description

Helper class for multiple updates to a [YBarGraph](#) widget: This will hold back display updates until this object goes out of scope.

Definition at line 285 of file [YBarGraph.h](#).

#### 5.11.2 Constructor & Destructor Documentation

### 5.11.2.1 YBarGraphMultiUpdate()

```
YBarGraphMultiUpdate::YBarGraphMultiUpdate (
    YBarGraph * barGraph )
```

Constructor.

This will make the corresponding [YBarGraph](#) widget hold back any pending display updates (due to changed values, labels, or colors) until this object is destroyed (goes out of scope).

Create objects of this class on the stack (as local variables) and simply let them go out of scope.

Example:

```
{ YBarGraphMultiUpdate multiUpdate( myBarGraph ); myBarGraph->setValue( 0, 42 ); // No display update yet my↔
BarGraph->setValue( 1, 84 ); // No display update yet myBarGraph->setValue( 2, 21 ); // No display update yet

} // multiUpdate goes out of scope, will trigger display update now
```

Definition at line 227 of file [YBarGraph.cc](#).

### 5.11.2.2 ~YBarGraphMultiUpdate()

```
YBarGraphMultiUpdate::~YBarGraphMultiUpdate ( )
```

Destructor.

This will trigger display updates of the corresponding [YBarGraph](#) widget if any are necessary.

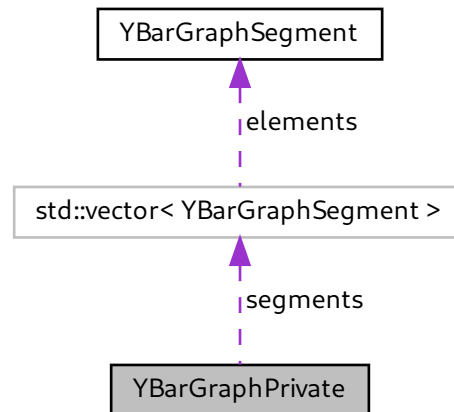
Definition at line 236 of file [YBarGraph.cc](#).

The documentation for this class was generated from the following files:

- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YBarGraph.h
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YBarGraph.cc

## 5.12 YBarGraphPrivate Struct Reference

Collaboration diagram for YBarGraphPrivate:



### Public Attributes

- `std::vector< YBarGraphSegment > segments`
- `bool updatesPending`
- `bool postponeUpdates`

#### 5.12.1 Detailed Description

Definition at line [53](#) of file [YBarGraph.cc](#).

The documentation for this struct was generated from the following file:

- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YBarGraph.cc`

## 5.13 YBarGraphSegment Class Reference

One segment of a [YBarGraph](#).

```
#include <YBarGraph.h>
```

## Public Member Functions

- [YBarGraphSegment](#) (int [value](#)=0, const std::string &[label](#)=std::string(), const [YColor](#) &[segmentColor](#)=[YColor](#)(), const [YColor](#) &[textColor](#)=[YColor](#)())  
*Constructor.*
- int [value](#) () const  
*Return the current value of this segment.*
- void [setValue](#) (int [newValue](#))  
*Set the value of this segment.*
- std::string [label](#) () const  
*Return the current text label of this segment.*
- void [setLabel](#) (const std::string &[newLabel](#))  
*Set the text label of this segment.*
- [YColor](#) [segmentColor](#) () const  
*Return the segment background color.*
- bool [hasSegmentColor](#) () const  
*Return 'true' if this segment's background color is defined, i.e.*
- void [setSegmentColor](#) (const [YColor](#) &[color](#))  
*Set this segment's background color.*
- [YColor](#) [textColor](#) () const  
*Return this segment's text color.*
- bool [hasTextColor](#) () const  
*Return 'true' if this segment's text color is defined, i.e.*
- void [setTextColor](#) (const [YColor](#) &[color](#))  
*Set this segment's text color.*

### 5.13.1 Detailed Description

One segment of a [YBarGraph](#).

It has a relative size, a label, label color and background color.

Definition at line [186](#) of file [YBarGraph.h](#).

### 5.13.2 Constructor & Destructor Documentation

### 5.13.2.1 YBarGraphSegment()

```
YBarGraphSegment::YBarGraphSegment (
    int value = 0,
    const std::string & label = std::string(),
    const YColor & segmentColor = YColor(),
    const YColor & textColor = YColor() ) [inline]
```

Constructor.

'value' is the initial value of this segment.

'label' is the label text in the segment. Use %1 as a placeholder for the current value.

'segmentColor' is the background color of this segment.

'textColor' is the color for the label text.

The [YBarGraph](#) widget will automatically assign some default colors (one of at least 5 different ones) if none are specified.

Definition at line [204](#) of file [YBarGraph.h](#).

## 5.13.3 Member Function Documentation

### 5.13.3.1 hasSegmentColor()

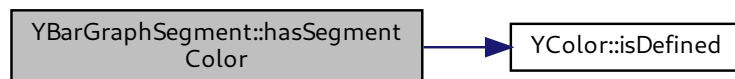
```
bool YBarGraphSegment::hasSegmentColor ( ) const [inline]
```

Return 'true' if this segment's background color is defined, i.e.

it has a real RGB value and was not just created with the default constructor.

Definition at line [246](#) of file [YBarGraph.h](#).

Here is the call graph for this function:



### 5.13.3.2 hasTextColor()

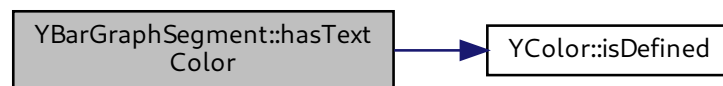
```
bool YBarGraphSegment::hasTextColor ( ) const [inline]
```

Return 'true' if this segment's text color is defined, i.e.

it has a real RGB value and was not just created with the default constructor.

Definition at line 263 of file [YBarGraph.h](#).

Here is the call graph for this function:



### 5.13.3.3 label()

```
std::string YBarGraphSegment::label ( ) const [inline]
```

Return the current text label of this segment.

Any %1 placeholder will be returned as %1 (not expanded).

Definition at line 228 of file [YBarGraph.h](#).

### 5.13.3.4 setLabel()

```
void YBarGraphSegment::setLabel (
    const std::string & newLabel ) [inline]
```

Set the text label of this segment.

Use %1 as a placeholder for the current value.

Definition at line 234 of file [YBarGraph.h](#).

The documentation for this class was generated from the following file:

- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YBarGraph.h`



## 5.14 YBothDim< T > Class Template Reference

Template class for two-dimensional entities, such as.

```
#include <YBothDim.h>
```

### Public Member Functions

- [YBothDim](#) (T hor, T vert)  
*Constructor with explicit initialization for both values.*
- [YBothDim](#) ()  
*Default constructor (calls T default constructor for both values)*
- T & [operator\[\]](#) (YUIDimension dim)  
*operator[] for alternative access via myVar[ YD\_HORIZ ] Please note that this returns a non-const reference, so this can be used as an lvalue (e.g., in assignments)*
- const T & [operator\[\]](#) (YUIDimension dim) const  
*Same as above for const objects.*

### Public Attributes

- T **vert**
- T **hor**

#### 5.14.1 Detailed Description

```
template<typename T>  
class YBothDim< T >
```

Template class for two-dimensional entities, such as.

- width, height
- x\_pos, y\_pos
- hStretchable, vStretchable

Precondition: type T needs to have a default constructor (which all simple types like int, long, bool have).

Definition at line 41 of file [YBothDim.h](#).

The documentation for this class was generated from the following file:

- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YBothDim.h

## 5.15 YBuiltinCaller Class Reference

Abstract base class for transparently calling a built-in function.

```
#include <YBuiltinCaller.h>
```

### Public Member Functions

- virtual void [call](#) ()=0  
*Call the built-in.*

#### 5.15.1 Detailed Description

Abstract base class for transparently calling a built-in function.

Derived classes will want to add some methods to store the function to be called, arguments to that function and its result and to retrieve the result when needed.

See YCPBuiltinCaller.h for an implementation.

Definition at line [37](#) of file [YBuiltinCaller.h](#).

#### 5.15.2 Member Function Documentation

##### 5.15.2.1 [call\(\)](#)

```
virtual void YBuiltinCaller::call ( ) [pure virtual]
```

Call the built-in.

This will be called in the UI thread with appropriate syncing between the threads.

Derived classes might want to store the result of the call in a member variable in this class so it can later be queried.

Derived classes are required to implement this method.

The documentation for this class was generated from the following file:

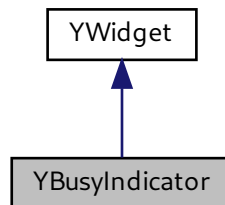
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YBuiltinCaller.h`

## 5.16 YBusyIndicator Class Reference

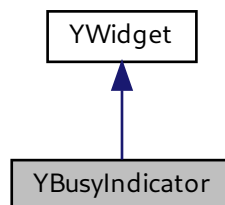
Indicates that something is in progress and has not frozen yet.

```
#include <YBusyIndicator.h>
```

Inheritance diagram for YBusyIndicator:



Collaboration diagram for YBusyIndicator:



### Public Member Functions

- virtual `~YBusyIndicator` ()  
*Destructor.*
- virtual const char \* `widgetClass` () const  
*Returns a descriptive name of this widget class for logging, debugging etc.*
- std::string `label` ()  
*Get the label (the caption above the progress bar).*
- virtual void `setLabel` (const std::string &`label`)  
*Set the label (the caption above the progress bar).*

- int [timeout](#) () const  
*Return the current timeout in milliseconds.*
- virtual void [setTimeout](#) (int newTimeout)  
*Set the timeout in milliseconds after that the widget shows 'stalled' when no new tick is received.*
- bool [alive](#) () const  
*Return whether busy indicator is alive or in stalled stated.*
- virtual void [setAlive](#) (bool newAlive)  
*Send a keep alive message to prevent BusyIndicator from changing to 'stalled' state.*
- virtual bool [setProperty](#) (const std::string &propertyName, const [YPropertyValue](#) &val)  
*Set a property.*
- virtual [YPropertyValue](#) [getProperty](#) (const std::string &propertyName)  
*Get a property.*
- virtual const [YPropertySet](#) & [propertySet](#) ()  
*Return this class's property set.*

## Protected Member Functions

- [YBusyIndicator](#) ([YWidget](#) \*parent, const std::string &label, int timeout=1000, bool alive=true)  
*Constructor.*

### 5.16.1 Detailed Description

Indicates that something is in progress and has not frozen yet.

It has a label and an "indeterminate" progress bar which will be "ticking" until a timeout occurs or until it receives an "alive" message.

Definition at line 38 of file [YBusyIndicator.h](#).

### 5.16.2 Member Function Documentation

#### 5.16.2.1 [getProperty\(\)](#)

```
YPropertyValue YBusyIndicator::getProperty (  
    const std::string & propertyName ) [virtual]
```

Get a property.

Reimplemented from [YWidget](#).

This method may throw [YUIPropertyExceptions](#).

Reimplemented from [YWidget](#).



### 5.16.2.3 `setAlive()`

```
void YBusyIndicator::setAlive (
    bool newAlive ) [virtual]
```

Send a keep alive message to prevent BusyIndicator from changing to 'stalled' state.

Derived classes should reimplement this, but they should call this base class method at the end of the overloaded function.

Definition at line 101 of file [YBusyIndicator.cc](#).

Here is the call graph for this function:



### 5.16.2.4 `setLabel()`

```
void YBusyIndicator::setLabel (
    const std::string & label ) [virtual]
```

Set the label (the caption above the progress bar).

Derived classes are free to reimplement this, but they should call this base class method at the end of the overloaded function.

Definition at line 80 of file [YBusyIndicator.cc](#).

Here is the call graph for this function:



## 5.16.2.5 setProperty()

```
bool YBusyIndicator::setProperty (
    const std::string & propertyName,
    const YPropertyValue & val ) [virtual]
```

Set a property.

Reimplemented from [YWidget](#).

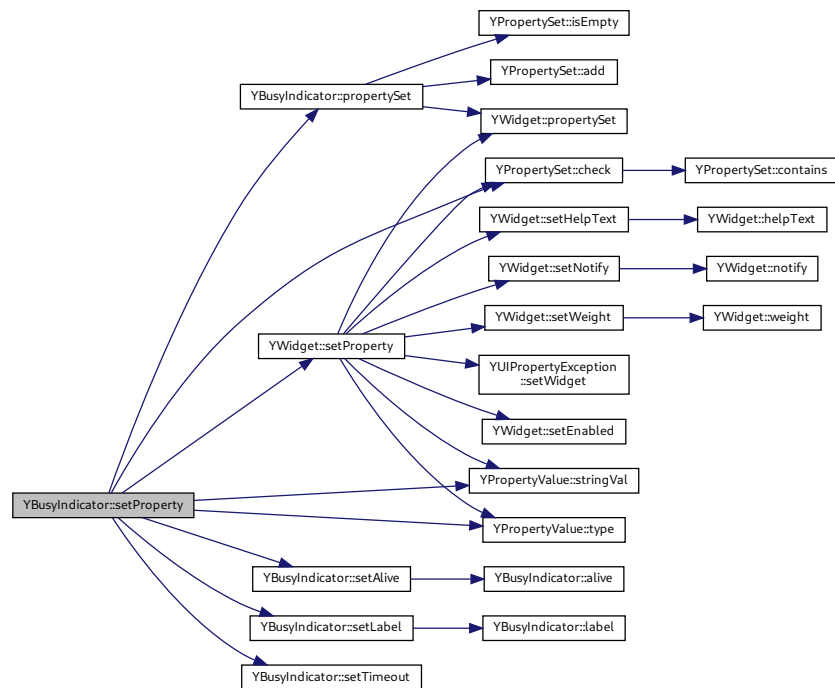
This function may throw YUIPropertyExceptions.

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from [YWidget](#).

Definition at line 134 of file [YBusyIndicator.cc](#).

Here is the call graph for this function:



### 5.16.2.6 setTimeout()

```
void YBusyIndicator::setTimeout (
    int newTimeout ) [virtual]
```

Set the timeout in milliseconds after that the widget shows 'stalled' when no new tick is received.

Derived classes should reimplement this, but they should call this base class method at the end of the overloaded function.

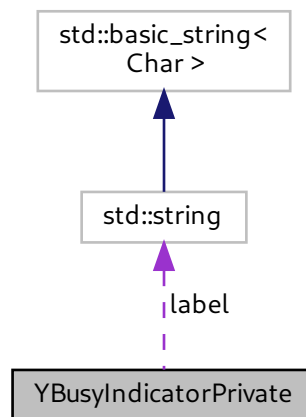
Definition at line 92 of file [YBusyIndicator.cc](#).

The documentation for this class was generated from the following files:

- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YBusyIndicator.h
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YBusyIndicator.cc

## 5.17 YBusyIndicatorPrivate Struct Reference

Collaboration diagram for YBusyIndicatorPrivate:



### Public Member Functions

- **YBusyIndicatorPrivate** (const string &label, int timeout, bool alive)



## Public Attributes

- string **label**
- int **timeout**
- bool **alive**

### 5.17.1 Detailed Description

Definition at line 35 of file [YBusyIndicator.cc](#).

The documentation for this struct was generated from the following file:

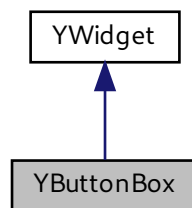
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YBusyIndicator.cc](#)

## 5.18 YButtonBox Class Reference

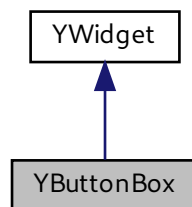
Container widget for dialog buttons that abstracts the button order depending on the desktop environment.

```
#include <YButtonBox.h>
```

Inheritance diagram for YButtonBox:



Collaboration diagram for YButtonBox:



## Public Member Functions

- virtual `~YButtonBox ()`  
*Destructor.*
- virtual const char \* `widgetClass () const`  
*Return a descriptive name of this widget class for logging, debugging etc.*
- virtual void `setMargins (const YButtonBoxMargins &margins)`  
*Set the margins for this YButtonBox.*
- `YButtonBoxMargins margins () const`  
*Return the margins of this YButtonBox.*
- virtual void `doLayout (int width, int height)`  
*Lay out the button box and its children (its buttons).*
- `YPushButton * findButton (YButtonRole role)`  
*Search the child widgets for the first button with the specified role.*
- void `sanityCheck ()`  
*Sanity check: Check if all child widgets have the correct widget class and if the button roles are assigned correctly.*
- void `setSanityCheckRelaxed (bool relax=true)`  
*Relax the sanity check done in `sanityCheck()`: Do not enforce that there has to be a YOKButton and a YCancelButton if there is more than one button.*
- bool `sanityCheckRelaxed () const`  
*Return 'true' if sanity checks are currently relaxed, 'false' if not.*
- virtual int `preferredWidth ()`  
*Preferred width of the widget.*
- virtual int `preferredHeight ()`  
*Preferred height of the widget.*
- virtual void `setSize (int newWidth, int newHeight)`  
*Sets the size of the YButtonBox.*
- virtual bool `stretchable (YUIDimension dimension) const`  
*Returns the stretchability of the YButtonBox.*

## Static Public Member Functions

- static void `setLayoutPolicy (const YButtonBoxLayoutPolicy &layoutPolicy)`  
*Set the button policy for all future YButtonBox widgets: Button order, alignment if there is any excess space, whether or not to give all buttons the same size.*
- static `YButtonBoxLayoutPolicy layoutPolicy ()`  
*Return the layout policy.*
- static `YButtonBoxLayoutPolicy kdeLayoutPolicy ()`  
*Predefined layout policy for KDE-like behaviour.*
- static `YButtonBoxLayoutPolicy gnomeLayoutPolicy ()`  
*Predefined layout policy for GNOME-like behaviour.*
- static void `setDefaultMargins (const YButtonBoxMargins &margins)`  
*Set the default margins for all future YButtonBox widgets.*
- static `YButtonBoxMargins defaultMargins ()`  
*Return the default margins for all future YButtonBox widgets.*

## Protected Member Functions

- [YGroupBox](#) ([YWidget](#) \*parent)  
*Constructor.*
- virtual std::vector< [YPushButton](#) \* > [buttonsByButtonOrder](#) ()  
*Return the button children sorted (left to right) by the current button order (from the layout policy).*
- int [maxChildSize](#) (YUIDimension dim) const  
*Return the (preferred) size of the biggest child widget in the specified dimension.*
- int [totalChildrenWidth](#) () const  
*Return the sum of the (preferred) widths of all child widgets.*
- virtual void [moveChild](#) ([YWidget](#) \*child, int newX, int newY)=0  
*Move a child to a new position.*
- int [preferredWidth](#) (bool equalSizeButtons)  
*Calculate the preferred width with or without trying to enforce buttons of equal size.*

## Friends

- class **YGroupBoxPrivate**

### 5.18.1 Detailed Description

Container widget for dialog buttons that abstracts the button order depending on the desktop environment.

KDE and Windows arrange dialog buttons like this:

```
[OK] [Apply] [Cancel] [Custom1] [Custom2] ... [Help]

[Continue] [Cancel]

[Yes] [No]
```

GNOME and MacOS arrange them like this:

```
[Help] [Custom1] [Custom2] ... [Apply] [Cancel] [OK]

[Cancel] [Continue]

[No] [Yes]
```

This class provides the abstraction to use whatever layout is more appropriate in the current environment. The application creates the buttons as child widgets of a [YGroupBox](#) (rather than a [YHBox](#)) and leaves the button order to the [YGroupBox](#).

Each of the standard buttons ([OK], [Apply], [Cancel], [Help]) needs to have a button role properly assigned.

If set up properly (see [YApplication::setDefaultFunctionKey\(\)](#)), known button labels will be assigned an appropriate role:

```

[OK]                                F10
[Continue] -> [OK]    F10
[Yes]      -> [OK]    F10
[Accept]   -> [OK]    F10
[Next]     -> [OK]    F10

[Cancel]                                F9
[No]      -> [Cancel]  F9

[Help]                                F1

```

Buttons with nonstandard labels that act in such a role need to be explicitly assigned that role:

```

[Print ] [Cancel] [Help]
[Delete] [Cancel] [Help]

```

Those [Print] or [Delete] buttons act as [OK] buttons (the "yes, do it" action of that dialog). Call `YPushButton::setButtonRole( YOkButton )` explicitly for them.

[YButtonBox](#) widgets only accept [YPushButton](#) child widgets. Otherwise an exception is thrown.

If there is more than one button, one of the child buttons needs to have the [OK] role, and another needs to have the [Cancel] role. Otherwise an exception is thrown.

Definition at line [148](#) of file [YButtonBox.h](#).

## 5.18.2 Member Function Documentation

### 5.18.2.1 buttonsByButtonOrder()

```
vector< YPushButton * > YButtonBox::buttonsByButtonOrder ( ) [protected], [virtual]
```

Return the button children sorted (left to right) by the current button order (from the layout policy).

This default implementation handles KDE and Gnome button orders. It is used in the default [doLayout\(\)](#) method.

This may throw exceptions if there are non-button children or if there are multiple buttons with any of the standard button roles (except `YCustomButton`, of course).

Definition at line [411](#) of file [YButtonBox.cc](#).

### 5.18.2.2 doLayout()

```
void YGroupBox::doLayout (
    int width,
    int height ) [virtual]
```

Lay out the button box and its children (its buttons).

This is where the button order is implemented.

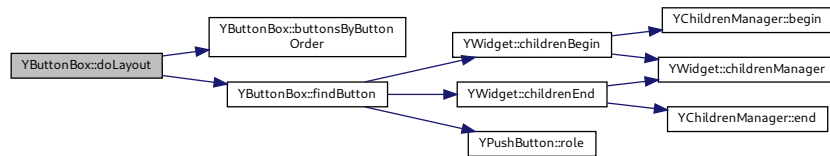
This method is called by the default [setSize\(\)](#) method. It uses [YGroupBox::layoutPolicy\(\)](#) and the specified margins (defaultMargins unless changed with [setMargins\(\)](#) ). It moves the buttons to their position with [moveChild\(\)](#).

This all should work reasonably in all cases (Qt-UI with KDE button order, Gtk-UI with Gnome button order, NCurses-UI with KDE button order).

Still, derived classes can reimplement this. It does not make very much sense to call this default method in a new implementation.

Definition at line 162 of file [YGroupBox.cc](#).

Here is the call graph for this function:



### 5.18.2.3 findButton()

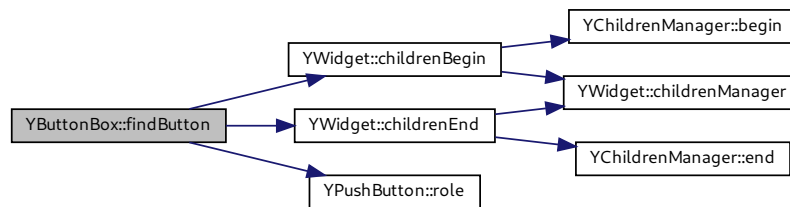
```
YPushButton * YGroupBox::findButton (
    YButtonRole role )
```

Search the child widgets for the first button with the specified role.

Return the button or 0 if there is no button with that role.

Definition at line 576 of file [YGroupBox.cc](#).

Here is the call graph for this function:



#### 5.18.2.4 margins()

```
YButtonBoxMargins YButtonBox::margins ( ) const
```

Return the margins of this [YButtonBox](#).

Notice that those are only the desired margins; if there is not enough space, margins and spacings will be reduced before buttons are cut off.

Definition at line [147](#) of file [YButtonBox.cc](#).

#### 5.18.2.5 moveChild()

```
virtual void YButtonBox::moveChild (
    YWidget * child,
    int newX,
    int newY ) [protected], [pure virtual]
```

Move a child to a new position.

This is used in [doLayout\(\)](#).

Derived classes are required to implement this.

### 5.18.2.6 preferredHeight()

```
int YButtonBox::preferredHeight ( ) [virtual]
```

Preferred height of the widget.

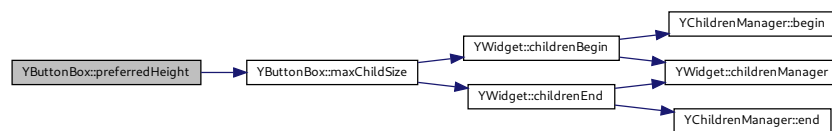
Reimplemented from [YWidget](#). This default method returns the height of the highest child plus the top and bottom margins.

Derived classes can reimplement this method. It does not make very much sense to call this base class method in a new implementation.

Implements [YWidget](#).

Definition at line 518 of file [YButtonBox.cc](#).

Here is the call graph for this function:



### 5.18.2.7 preferredWidth()

```
int YButtonBox::preferredWidth ( ) [virtual]
```

Preferred width of the widget.

Reimplemented from [YWidget](#). This default method returns the sum of the widths of all child widgets plus the left and right margins plus the spacings.

Derived classes can reimplement this method. It does not make very much sense to call this base class method in a new implementation.

Implements [YWidget](#).

Definition at line 511 of file [YButtonBox.cc](#).

### 5.18.2.8 sanityCheck()

```
void YButtonBox::sanityCheck ( )
```

Sanity check: Check if all child widgets have the correct widget class and if the button roles are assigned correctly.

A [YButtonBox](#) with more than one button is required to have one [YOKButton](#) and only [YCancelButton](#). Neither button role may be assigned more than once.

This method may throw exceptions:

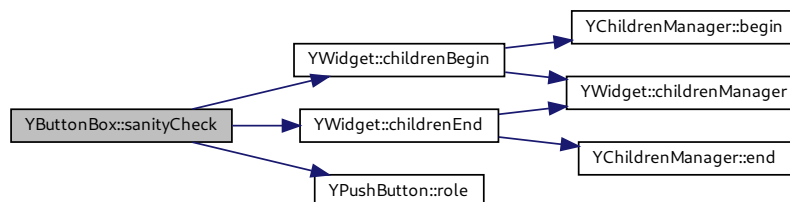
- [YUIButtonRoleMismatchException](#)
- [YUIInvalidChildException](#) (wrong widget class)

This cannot be done as child widgets are inserted since this is done from the child widgets' constructors, so virtual methods or `dynamic_cast` don't work at that point.

This is called in the default [setSize\(\)](#) method (just before [doLayout\(\)](#)), so any of the above errors are caught only at that time. Applications are free to call this before that time to make error handling more transparent.

Definition at line 607 of file [YButtonBox.cc](#).

Here is the call graph for this function:



### 5.18.2.9 setLayoutPolicy()

```
void YButtonBox::setLayoutPolicy (
    const YButtonBoxLayoutPolicy & layoutPolicy ) [static]
```

Set the button policy for all future [YButtonBox](#) widgets: Button order, alignment if there is any excess space, whether or not to give all buttons the same size.

You might want to use one of the predefined static methods: [YButtonBox::kdeLayoutPolicy\(\)](#), [YButtonBox::gnomeLayoutPolicy\(\)](#).

The default [doLayout\(\)](#) method uses those values.



Notice that there is intentionally no way to set this differently for each individual [YButtonBox](#): This would defeat the purpose of consistency (with the desktop environment this application is running in), which is the reason for having this widget class.

Definition at line 83 of file [YButtonBox.cc](#).

Here is the call graph for this function:



#### 5.18.2.10 setMargins()

```
void YButtonBox::setMargins (
    const YButtonBoxMargins & margins ) [virtual]
```

Set the margins for this [YButtonBox](#).

Derived classes are free to reimplement this, but they should call this base class method in the new method.

Definition at line 140 of file [YButtonBox.cc](#).

Here is the call graph for this function:



### 5.18.2.11 setSanityCheckRelaxed()

```
void YButtonBox::setSanityCheckRelaxed (
    bool relax = true )
```

Relax the sanity check done in [sanityCheck\(\)](#): Do not enforce that there has to be a YOKButton and a YCancelButton if there is more than one button.

In very rare cases, it might be necessary to have a less stringent sanity check: There are some very few legitimate cases for having a [YButtonBox](#) with multiple buttons, yet without a YCancelButton.

Examples:

```
...message...
<Countdown>
[OK] [Stop]

...message...
[OK] [Details]
```

In those cases, it makes sense to relax the sanity check.

Definition at line [593](#) of file [YButtonBox.cc](#).

### 5.18.2.12 setSize()

```
void YButtonBox::setSize (
    int newWidth,
    int newHeight ) [virtual]
```

Sets the size of the [YButtonBox](#).

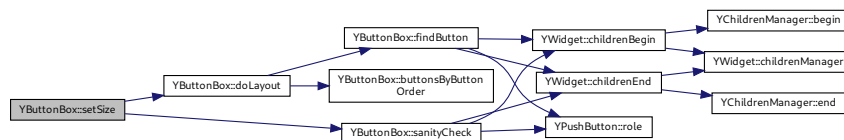
Derived classes can reimplement this, but this base class method should be called in the reimplemented function.

Reimplemented from [YWidget](#).

Implements [YWidget](#).

Definition at line [154](#) of file [YButtonBox.cc](#).

Here is the call graph for this function:



### 5.18.2.13 stretchable()

```
bool YButtonBox::stretchable (
    YUIDimension dimension ) const [virtual]
```

Returns the stretchability of the [YButtonBox](#).

[YButtonBox](#) widgets are horizontally stretchable by default. How any excess space is used is specified in the layout policy.

Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 561 of file [YButtonBox.cc](#).

The documentation for this class was generated from the following files:

- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YButtonBox.h](#)
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YButtonBox.cc](#)

## 5.19 YButtonBoxLayoutPolicy Struct Reference

Helper class: Layout policy for [YButtonBox](#) widgets.

```
#include <YButtonBox.h>
```

### Public Attributes

- YButtonOrder **buttonOrder**
- bool **equalSizeButtons**
- bool **addExcessSpaceToHelpButtonExtraMargin**
- YAlignmentType **alignment** [YUIAllDimensions]

### 5.19.1 Detailed Description

Helper class: Layout policy for [YButtonBox](#) widgets.

This is used in the default [YButtonBox::doLayout\(\)](#) method.

Definition at line 41 of file [YButtonBox.h](#).

The documentation for this struct was generated from the following file:

- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YButtonBox.h](#)

## 5.20 YButtonBoxMargins Struct Reference

Helper class: Margins for [YButtonBox](#) widgets.

```
#include <YButtonBox.h>
```

### Public Attributes

- int **left**
- int **right**
- int **top**
- int **bottom**
- int **spacing**
- int **helpButtonExtraSpacing**

### 5.20.1 Detailed Description

Helper class: Margins for [YButtonBox](#) widgets.

All sizes are in UI-dependent units, i.e. in pixels.

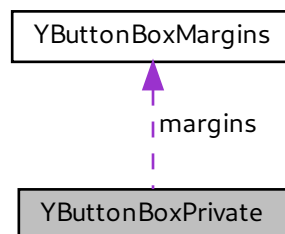
Definition at line 65 of file [YButtonBox.h](#).

The documentation for this struct was generated from the following file:

- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YButtonBox.h

## 5.21 YButtonBoxPrivate Struct Reference

Collaboration diagram for YButtonBoxPrivate:



## Public Member Functions

- [YButtonBoxPrivate](#) ()  
*Constructor.*

## Public Attributes

- bool **sanityCheckRelaxed**
- [YButtonBoxMargins](#) **margins**

### 5.21.1 Detailed Description

Definition at line 46 of file [YButtonBox.cc](#).

The documentation for this struct was generated from the following file:

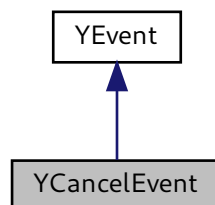
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YButtonBox.cc

## 5.22 YCancelEvent Class Reference

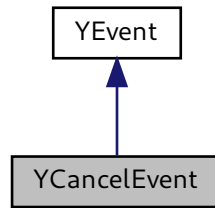
Event to be returned upon closing a dialog with the window manager close button (or Alt-F4)

```
#include <YEvent.h>
```

Inheritance diagram for YCancelEvent:



Collaboration diagram for YCancelEvent:



## Protected Member Functions

- virtual [~YCancelEvent\(\)](#)

*Protected destructor - events can only be deleted via [YDialog::deleteEvent\(\)](#).*

## Additional Inherited Members

### 5.22.1 Detailed Description

Event to be returned upon closing a dialog with the window manager close button (or Alt-F4)

Definition at line 305 of file [YEvent.h](#).

### 5.22.2 Constructor & Destructor Documentation

#### 5.22.2.1 ~YCancelEvent()

```
virtual YCancelEvent::~YCancelEvent ( ) [inline], [protected], [virtual]
```

Protected destructor - events can only be deleted via [YDialog::deleteEvent\(\)](#).

The associated dialog will take care of this event and delete it when appropriate.

Definition at line 318 of file [YEvent.h](#).

The documentation for this class was generated from the following file:

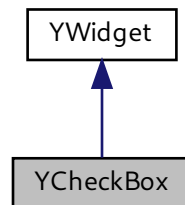
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YEvent.h](#)

## 5.23 YCheckBox Class Reference

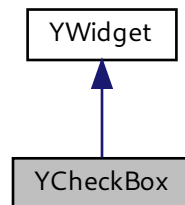
A tri-state check box.

```
#include <YCheckBox.h>
```

Inheritance diagram for YCheckBox:



Collaboration diagram for YCheckBox:



### Public Member Functions

- virtual `~YCheckBox()`  
*Destructor.*
- virtual const char \* `widgetClass()` const  
*Returns a descriptive name of this widget class for logging, debugging etc.*
- virtual YCheckBoxState `value()`=0  
*Get the current value:*
- virtual void `setValue(YCheckBoxState state)`=0  
*Set the CheckBox value (on/off/don't care).*

- bool `isChecked` ()  
*Simplified access to `value()`: Return 'true' if the CheckBox is checked.*
- void `setChecked` (bool checked=true)  
*Simplified access to `setValue()`: Check or uncheck the CheckBox.*
- bool `dontCare` ()  
*Simplified access to tri-state ("don't care").*
- void `setDontCare` ()  
*Simplified access to setting tri-state ("don't care").*
- std::string `label` () const  
*Get the label (the text on the CheckBox).*
- virtual void `setLabel` (const std::string &label)  
*Set the label (the text on the CheckBox).*
- bool `useBoldFont` () const  
*Returns 'true' if a bold font should be used.*
- virtual void `setUseBoldFont` (bool bold=true)  
*Indicate whether or not a bold font should be used.*
- virtual bool `setProperty` (const std::string &propertyName, const `YPropertyValue` &val)  
*Set a property.*
- virtual `YPropertyValue` `getProperty` (const std::string &propertyName)  
*Get a property.*
- virtual const `YPropertySet` & `propertySet` ()  
*Return this class's property set.*
- virtual std::string `shortcutString` () const  
*Get the string of this widget that holds the keyboard shortcut.*
- virtual void `setShortcutString` (const std::string &str)  
*Set the string of this widget that holds the keyboard shortcut.*
- const char \* `userInputProperty` ()  
*The name of the widget property that will return user input.*

## Protected Member Functions

- `YCheckBox` (`YWidget` \*parent, const std::string &label)  
*Constructor.*

### 5.23.1 Detailed Description

A tri-state check box.

It can be toggled between ON and OFF by the user and additionally set to a DONT-CARE value programmatically.

Definition at line 46 of file `YCheckBox.h`.

### 5.23.2 Member Function Documentation



### 5.23.2.1 getProperty()

```
YPropertyValue YCheckBox::getProperty (
    const std::string & propertyName ) [virtual]
```

Get a property.

Reimplemented from [YWidget](#).

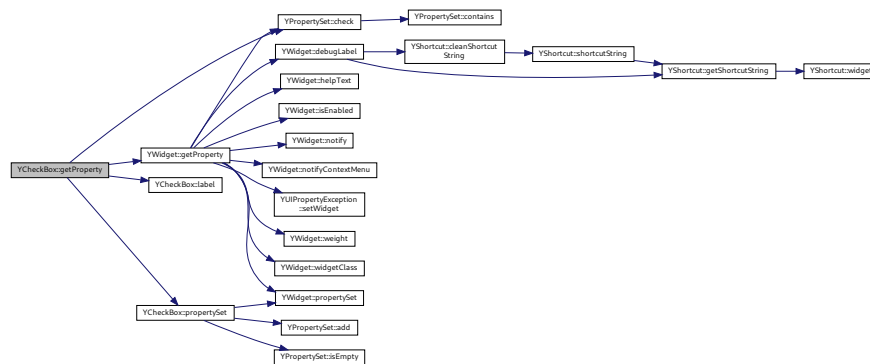
This method may throw exceptions, for example

- if there is no property with that name

Reimplemented from [YWidget](#).

Definition at line 123 of file [YCheckBox.cc](#).

Here is the call graph for this function:



### 5.23.2.2 propertySet()

```
const YPropertySet & YCheckBox::propertySet ( ) [virtual]
```

Return this class's property set.

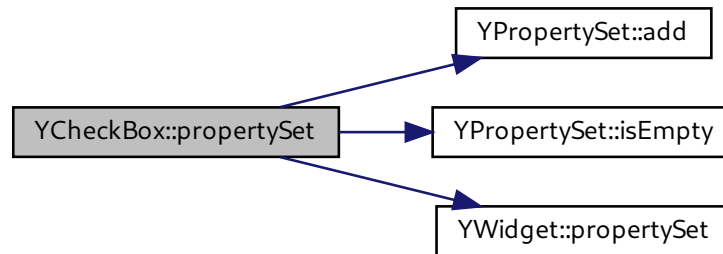
This also initializes the property set upon the first call.

Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 86 of file [YCheckBox.cc](#).

Here is the call graph for this function:



### 5.23.2.3 setLabel()

```
void YCheckBox::setLabel (
    const std::string & label ) [virtual]
```

Set the label (the text on the CheckBox).

Derived classes are free to reimplement this, but they should call this base class method at the end of the overloaded function.

Definition at line 61 of file [YCheckBox.cc](#).

### 5.23.2.4 setProperty()

```
bool YCheckBox::setProperty (
    const std::string & propertyName,
    const YPropertyValue & val ) [virtual]
```

Set a property.

Reimplemented from [YWidget](#).

This method may throw exceptions, for example

- if there is no property with that name
- if the expected type and the type mismatch

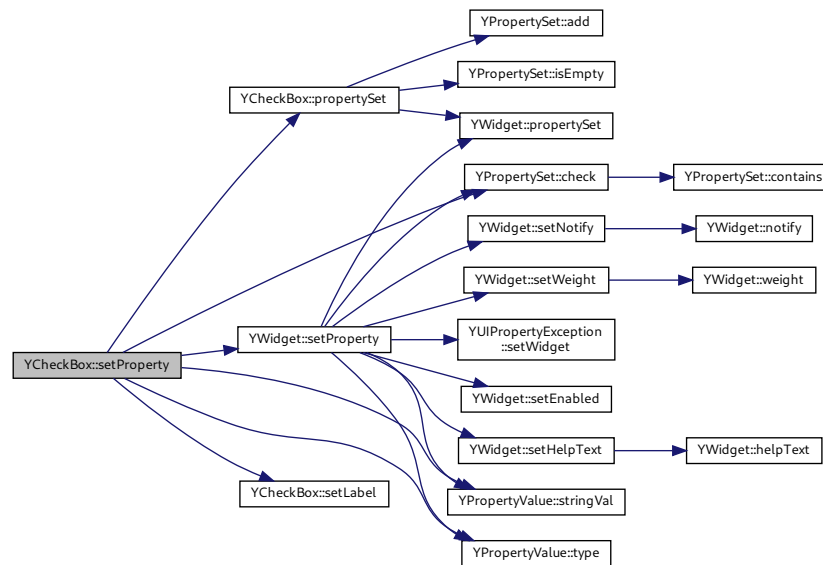
- if the value is out of range

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from [YWidget](#).

Definition at line 107 of file [YCheckBox.cc](#).

Here is the call graph for this function:



#### 5.23.2.5 setShortcutString()

```
virtual void YCheckBox::setShortcutString (
    const std::string & str ) [inline], [virtual]
```

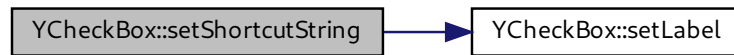
Set the string of this widget that holds the keyboard shortcut.

Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 190 of file [YCheckBox.h](#).

Here is the call graph for this function:



#### 5.23.2.6 `setUseBoldFont()`

```
void YCheckBox::setUseBoldFont (
    bool bold = true ) [virtual]
```

Indicate whether or not a bold font should be used.

Derived classes are free to reimplement this, but they should call this base class method at the end of the overloaded function.

Definition at line [79](#) of file [YCheckBox.cc](#).

#### 5.23.2.7 `setValue()`

```
virtual void YCheckBox::setValue (
    YCheckBoxState state ) [pure virtual]
```

Set the CheckBox value (on/off/don't care).

Derived classes are required to implement this.

### 5.23.2.8 shortcutString()

```
virtual std::string YCheckBox::shortcutString ( ) const [inline], [virtual]
```

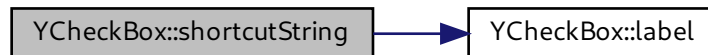
Get the string of this widget that holds the keyboard shortcut.

Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 183 of file [YCheckBox.h](#).

Here is the call graph for this function:



### 5.23.2.9 userInputProperty()

```
const char* YCheckBox::userInputProperty ( ) [inline], [virtual]
```

The name of the widget property that will return user input.

Inherited from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 197 of file [YCheckBox.h](#).

### 5.23.2.10 value()

```
virtual YCheckBoxState YCheckBox::value ( ) [pure virtual]
```

Get the current value:

`YCheckBox_on` CheckBox is checked `YCheckBox_off` CheckBox is unchecked

`YCheckBox_dont_care` tri-state: CheckBox is greyed out, neither checked nor unchecked

The user cannot set `YCheckBox_dont_care` directly. This status is always only set from the outside, usually because a setting cannot be clearly determined. For example, a checkbox

[ ] Read only

would be set to "don't care" (by the application, not directly by the user) when it is to display the read-only state of a group of files where some are read-only and some are writeable.

Derived classes are required to implement this function. (Intentionally not const)

The documentation for this class was generated from the following files:

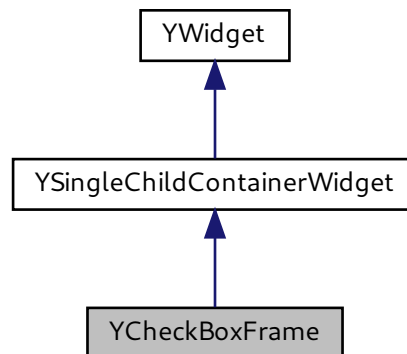
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YCheckBox.h`
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YCheckBox.cc`

## 5.24 YCheckBoxFrame Class Reference

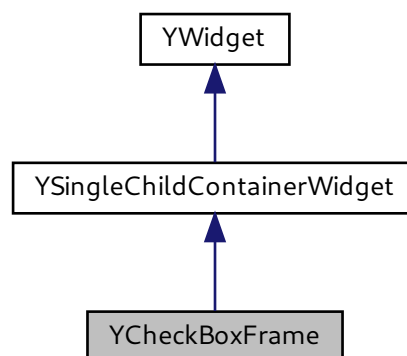
A frame with a check-box, may auto-disable frame contents based on the check.

```
#include <YCheckBoxFrame.h>
```

Inheritance diagram for YCheckBoxFrame:



Collaboration diagram for YCheckBoxFrame:



## Public Member Functions

- [YCheckBoxFrame](#) ([YWidget](#) \*[parent](#), const std::string &[label](#), bool checked)  
*Constructor.*
- virtual [~YCheckBoxFrame](#) ()  
*Destructor.*
- virtual const char \* [widgetClass](#) () const  
*Returns a descriptive name of this widget class for logging, debugging etc.*
- std::string [label](#) () const  
*Return the label text on the CheckBoxFrame.*
- virtual void [setLabel](#) (const std::string &[label](#))  
*Change the label text on the CheckBoxFrame.*
- virtual void [setValue](#) (bool isChecked)=0  
*Check or uncheck the CheckBoxFrame's check box.*
- virtual bool [value](#) ()=0  
*Get the status of the CheckBoxFrame's check box.*
- bool [autoEnable](#) () const  
*Handle children enabling/disabling automatically based on the CheckBoxFrame's check box?*
- virtual void [setAutoEnable](#) (bool [autoEnable](#))  
*Change autoEnabled flag.*
- bool [invertAutoEnable](#) () const  
*Invert the meaning of the CheckBoxFrame's check box, i.e., disable child widgets when checked?*
- virtual void [setInvertAutoEnable](#) (bool [invertAutoEnable](#))  
*Change invertAutonEnable flag.*
- void [handleChildrenEnablement](#) (bool isChecked)  
*Handle enabling/disabling of child widgets based on 'isChecked' (the current status of the check box) and [autoEnable\(\)](#) and [invertAutoEnable\(\)](#).*
- virtual std::string [shortcutString](#) () const  
*Get the string of this widget that holds the keyboard shortcut.*
- virtual void [setShortcutString](#) (const std::string &str)  
*Set the string of this widget that holds the keyboard shortcut.*
- const char \* [userInputProperty](#) ()  
*The name of the widget property that will return user input.*
- virtual bool [setProperty](#) (const std::string &propertyName, const [YPropertyValue](#) &val)  
*Set a property.*
- virtual [YPropertyValue](#) [getProperty](#) (const std::string &propertyName)  
*Get a property.*
- virtual const [YPropertySet](#) & [propertySet](#) ()  
*Return this class's property set.*

## Additional Inherited Members

### 5.24.1 Detailed Description

A frame with a check-box, may auto-disable frame contents based on the check.

See [setAutoEnable](#), [invertAutoEnable](#).

Definition at line 39 of file [YCheckBoxFrame.h](#).

## 5.24.2 Member Function Documentation

### 5.24.2.1 getProperty()

```
YPropertyValue YCheckBoxFrame::getProperty (
    const std::string & propertyName ) [virtual]
```

Get a property.

Reimplemented from [YWidget](#).

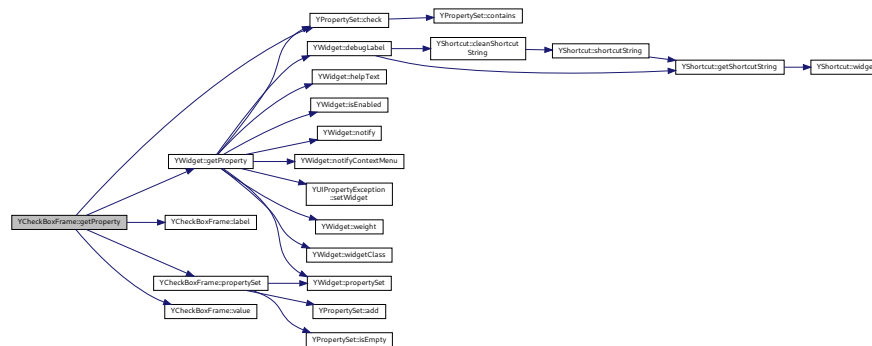
This method may throw exceptions, for example

- if there is no property with that name

Reimplemented from [YWidget](#).

Definition at line 156 of file [YCheckBoxFrame.cc](#).

Here is the call graph for this function:





### 5.24.2.2 handleChildrenEnablement()

```
void YCheckBoxFrame::handleChildrenEnablement (
    bool isChecked )
```

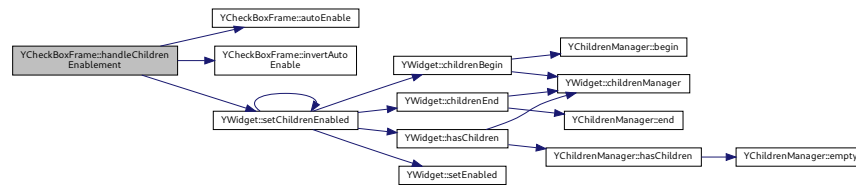
Handle enabling/disabling of child widgets based on 'isChecked' (the current status of the check box) and [autoEnable\(\)](#) and [invertAutoEnable\(\)](#).

Derived classes should call this when the check box status changes rather than try to handle it on their level.

This method also needs to be called after new child widgets are added to establish the initial enabled or disabled state of the child widgets.

Definition at line 105 of file [YCheckBoxFrame.cc](#).

Here is the call graph for this function:



### 5.24.2.3 propertySet()

```
const YPropertySet & YCheckBoxFrame::propertySet ( ) [virtual]
```

Return this class's property set.

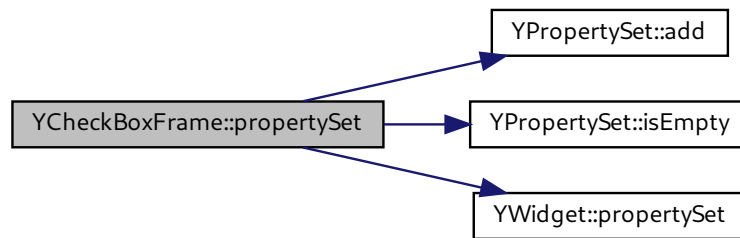
This also initializes the property set upon the first call.

Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 119 of file [YCheckBoxFrame.cc](#).

Here is the call graph for this function:



#### 5.24.2.4 setAutoEnable()

```
void YCheckBoxFrame::setAutoEnable (
    bool autoEnable ) [virtual]
```

Change `autoEnabled` flag.

Derived classes are free to overload this, but they should call this base class function in the overloaded function.

Definition at line 85 of file [YCheckBoxFrame.cc](#).

Here is the call graph for this function:



#### 5.24.2.5 setInvertAutoEnable()

```
void YCheckBoxFrame::setInvertAutoEnable (
    bool invertAutoEnable ) [virtual]
```

Change invertAutonEnable flag.

Derived classes are free to overload this, but they should call this base class function in the overloaded function.

Definition at line 98 of file [YCheckBoxFrame.cc](#).

Here is the call graph for this function:



#### 5.24.2.6 setLabel()

```
void YCheckBoxFrame::setLabel (
    const std::string & label ) [virtual]
```

Change the label text on the CheckBoxFrame.

Derived classes should overload this, but call this base class function in the overloaded function.

Definition at line 73 of file [YCheckBoxFrame.cc](#).

Here is the call graph for this function:



### 5.24.2.7 setProperty()

```
bool YCheckBoxFrame::setProperty (
    const std::string & propertyName,
    const YPropertyValue & val ) [virtual]
```

Set a property.

Reimplemented from [YWidget](#).

This method may throw exceptions, for example

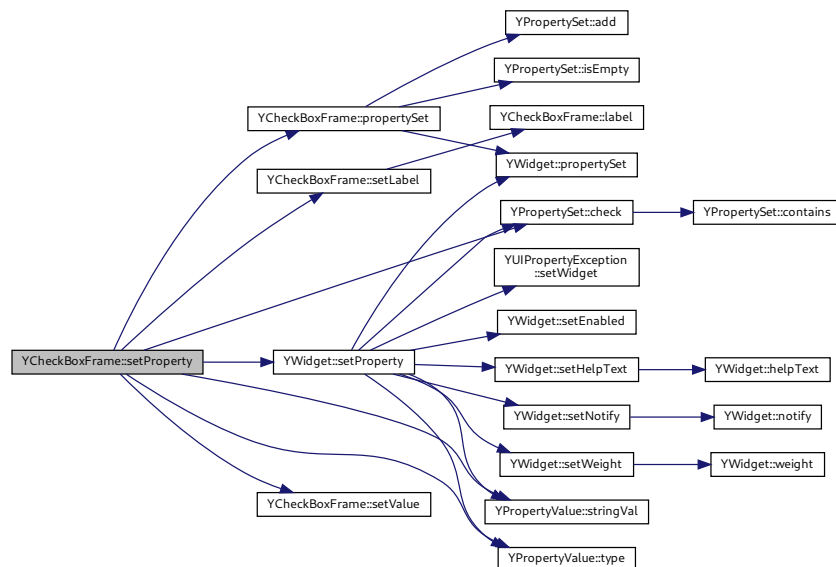
- if there is no property with that name
- if the expected type and the type mismatch
- if the value is out of range

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from [YWidget](#).

Definition at line 140 of file [YCheckBoxFrame.cc](#).

Here is the call graph for this function:



#### 5.24.2.8 setShortcutString()

```
virtual void YCheckBoxFrame::setShortcutString (
    const std::string & str ) [inline], [virtual]
```

Set the string of this widget that holds the keyboard shortcut.

Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 140 of file [YCheckBoxFrame.h](#).

Here is the call graph for this function:



#### 5.24.2.9 setValue()

```
virtual void YCheckBoxFrame::setValue (
    bool isChecked ) [pure virtual]
```

Check or uncheck the CheckBoxFrame's check box.

Derived classes are required to implement this.

#### 5.24.2.10 shortcutString()

```
virtual std::string YCheckBoxFrame::shortcutString ( ) const [inline], [virtual]
```

Get the string of this widget that holds the keyboard shortcut.

Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 133 of file [YCheckBoxFrame.h](#).

Here is the call graph for this function:



#### 5.24.2.11 userInputProperty()

```
const char* YCheckBoxFrame::userInputProperty ( ) [inline], [virtual]
```

The name of the widget property that will return user input.

Inherited from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 147 of file [YCheckBoxFrame.h](#).

#### 5.24.2.12 value()

```
virtual bool YCheckBoxFrame::value ( ) [pure virtual]
```

Get the status of the CheckBoxFrame's check box.

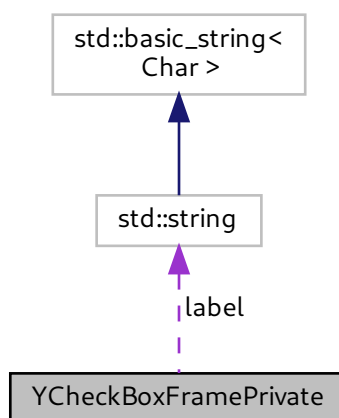
Derived classes are required to implement this.

The documentation for this class was generated from the following files:

- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YCheckBoxFrame.h](#)
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YCheckBoxFrame.cc](#)

## 5.25 YCheckBoxFramePrivate Struct Reference

Collaboration diagram for YCheckBoxFramePrivate:



## Public Member Functions

- **YCheckBoxFramePrivate** (const string &label)

## Public Attributes

- string **label**
- bool **autoEnable**
- bool **invertAutoEnable**

### 5.25.1 Detailed Description

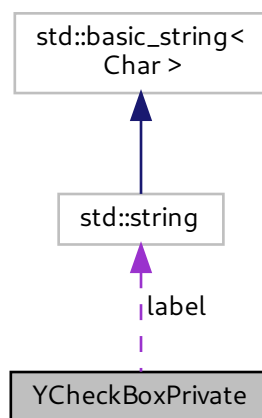
Definition at line 35 of file [YCheckBoxFrame.cc](#).

The documentation for this struct was generated from the following file:

- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YCheckBoxFrame.cc

## 5.26 YCheckBoxPrivate Struct Reference

Collaboration diagram for YCheckBoxPrivate:



## Public Member Functions

- **YCheckBoxPrivate** (const string &label)

## Public Attributes

- string **label**
- bool **useBoldFont**

### 5.26.1 Detailed Description

Definition at line 35 of file [YCheckBox.cc](#).

The documentation for this struct was generated from the following file:

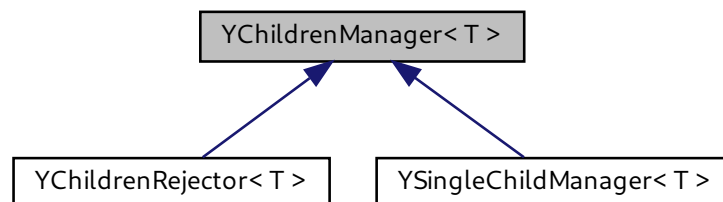
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YCheckBox.cc

## 5.27 YChildrenManager< T > Class Template Reference

Abstract base template class for children management, such as child widgets.

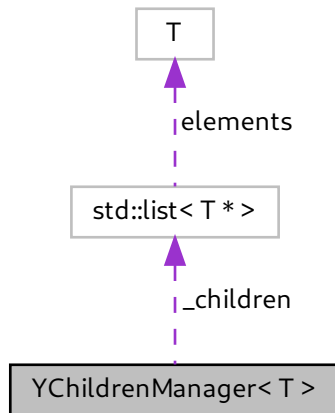
```
#include <YChildrenManager.h>
```

Inheritance diagram for YChildrenManager< T >:





Collaboration diagram for YChildrenManager< T >:



## Public Types

- typedef `std::list< T * >` **ChildrenList**

## Public Member Functions

- `YChildrenManager` (`T *containerParent`)  
*Constructor.*
- virtual `~YChildrenManager` ()  
*Destructor.*
- bool `hasChildren` () const  
*Check if there are any children.*
- bool `empty` () const  
*Check if the children list is empty, i.e.*
- int `count` () const  
*Returns the number of children.*
- `ChildrenList::iterator` `begin` ()  
*Return an iterator that points to the first child.*
- `ChildrenList::iterator` `end` ()  
*Return an iterator that points after the last child.*
- `ChildrenList::const_iterator` `begin` () const  
*Return an iterator that points to the first child.*
- `ChildrenList::const_iterator` `end` () const  
*Return an iterator that points after the last child.*
- `ChildrenList::const_reverse_iterator` `rbegin` () const

- Return a reverse iterator that points to the last child.*
- `ChildrenList::const_reverse_iterator rend () const`  
*Return a reverse iterator that points before the first child.*
- `T * firstChild ()`  
*Returns the first child or 0 if there is none.*
- `T * lastChild ()`  
*Returns the last child or 0 if there is none.*
- `virtual void add (T *child)`  
*Add a new child.*
- `virtual void remove (T *child)`  
*Remove a child.*
- `virtual void clear ()`  
*Remove all children.*
- `bool contains (T *child) const`  
*Check if the children list contains the specified child.*
- `T * container () const`  
*Returns the associated container, i.e.*

## Protected Attributes

- `T * _container`
- `ChildrenList _children`

## 5.27.1 Detailed Description

```
template<class T>
class YChildrenManager< T >
```

Abstract base template class for children management, such as child widgets.

Definition at line 37 of file [YChildrenManager.h](#).

## 5.27.2 Constructor & Destructor Documentation

### 5.27.2.1 YChildrenManager()

```
template<class T >
YChildrenManager< T >::YChildrenManager (
    T * containerParent ) [inline]
```

Constructor.

'containerParent' is the class whose children are managed.

Definition at line 46 of file [YChildrenManager.h](#).

### 5.27.3 Member Function Documentation

#### 5.27.3.1 add()

```
template<class T >
virtual void YChildrenManager< T >::add (
    T * child ) [inline], [virtual]
```

Add a new child.

This may throw exceptions if more children are added than the class whose children are handled (the associated widget) can handle.

Reimplemented in [YChildrenRejector< T >](#), and [YSingleChildManager< T >](#).

Definition at line [128](#) of file [YChildrenManager.h](#).

#### 5.27.3.2 clear()

```
template<class T >
virtual void YChildrenManager< T >::clear ( ) [inline], [virtual]
```

Remove all children.

This only removes the children from the children manager's list; it does not delete them.

Definition at line [142](#) of file [YChildrenManager.h](#).

#### 5.27.3.3 container()

```
template<class T >
T* YChildrenManager< T >::container ( ) const [inline]
```

Returns the associated container, i.e.

the object whose children are handled here.

Definition at line [160](#) of file [YChildrenManager.h](#).

#### 5.27.3.4 contains()

```
template<class T >
bool YChildrenManager< T >::contains (
    T * child ) const    [inline]
```

Check if the children list contains the specified child.

Returns 'true' if the children list contains the child, 'false' otherwise.

Definition at line 150 of file [YChildrenManager.h](#).

#### 5.27.3.5 empty()

```
template<class T >
bool YChildrenManager< T >::empty ( ) const    [inline]
```

Check if the children list is empty, i.e.

if there are no children.

Definition at line 66 of file [YChildrenManager.h](#).

#### 5.27.3.6 firstChild()

```
template<class T >
T* YChildrenManager< T >::firstChild ( )    [inline]
```

Returns the first child or 0 if there is none.

Useful mostly for children managers that handle only one child.

Definition at line 113 of file [YChildrenManager.h](#).

#### 5.27.3.7 remove()

```
template<class T >
virtual void YChildrenManager< T >::remove (
    T * child )    [inline], [virtual]
```

Remove a child.

This only removes the child from the children manager's list; it does not delete it.

Definition at line 135 of file [YChildrenManager.h](#).

The documentation for this class was generated from the following file:

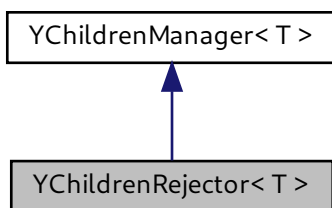
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YChildrenManager.h](#)

## 5.28 YChildrenRejector< T > Class Template Reference

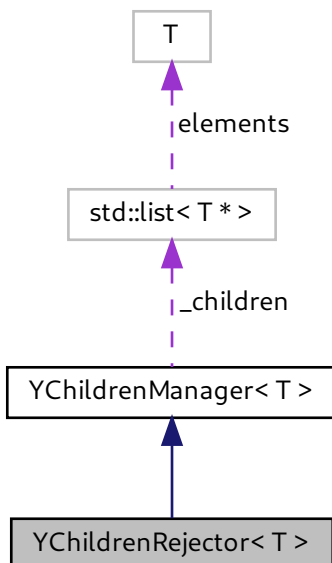
Children manager that rejects all children.

```
#include <YChildrenManager.h>
```

Inheritance diagram for YChildrenRejector< T >:



Collaboration diagram for YChildrenRejector< T >:



## Public Member Functions

- [YChildrenRejector](#) (T \*containerParent)  
*Constructor.*
- virtual void [add](#) (T \*child)  
*Add a new child.*

## Additional Inherited Members

### 5.28.1 Detailed Description

```
template<class T>
class YChildrenRejector< T >
```

Children manager that rejects all children.

Useful for widget classes that can't handle children such as [YPushButton](#), [YSelectionBox](#) etc.

Definition at line [214](#) of file [YChildrenManager.h](#).

### 5.28.2 Member Function Documentation

#### 5.28.2.1 add()

```
template<class T >
virtual void YChildrenRejector< T >::add (
    T * child ) [inline], [virtual]
```

Add a new child.

Reimplemented from [YChildrenManager](#).

Since this class is designed to reject children, this always throws a [YUITooManyChildrenException](#).

Reimplemented from [YChildrenManager< T >](#).

Definition at line [232](#) of file [YChildrenManager.h](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YChildrenManager.h](#)

## 5.29 YCodeLocation Class Reference

Helper class for UI exceptions: Store *FILE*, *FUNCTION* and *LINE*.

```
#include <YUIException.h>
```

### Public Member Functions

- [YCodeLocation](#) (const std::string &file\_r, const std::string &func\_r, int line\_r)  
*Constructor.*
- [YCodeLocation](#) ()  
*Default constructor.*
- std::string [file](#) () const  
*Returns the source file name where the exception occurred.*
- std::string [func](#) () const  
*Returns the name of the function where the exception occurred.*
- int [line](#) () const  
*Returns the source line number where the exception occurred.*
- std::string [asString](#) () const  
*Returns the location in normalized string format.*

### Friends

- std::ostream & [operator<<](#) (std::ostream &str, const [YCodeLocation](#) &obj)  
*Stream output.*

#### 5.29.1 Detailed Description

Helper class for UI exceptions: Store *FILE*, *FUNCTION* and *LINE*.

Construct this using the YUI\_EXCEPTION\_CODE\_LOCATION macro.

Definition at line 229 of file [YUIException.h](#).

#### 5.29.2 Constructor & Destructor Documentation

### 5.29.2.1 YCodeLocation()

```
YCodeLocation::YCodeLocation (
    const std::string & file_r,
    const std::string & func_r,
    int line_r ) [inline]
```

Constructor.

Commonly called using the YUI\_EXCEPTION\_CODE\_LOCATION macro.

Definition at line 236 of file [YUIException.h](#).

The documentation for this class was generated from the following files:

- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YUIException.h
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YUIException.cc

## 5.30 YColor Class Reference

Helper class to define an RGB color.

```
#include <YColor.h>
```

### Public Member Functions

- [YColor](#) (uchar [red](#), uchar [green](#), uchar [blue](#))  
*Constructor.*
- [YColor](#) ()  
*Default constructor: Create "undefined" color.*
- uchar [red](#) () const  
*Return the red component (0: none, 255: bright red).*
- uchar [green](#) () const  
*Return the green component (0: none, 255: bright green).*
- uchar [blue](#) () const  
*Return the blue component (0: none, 255: bright blue).*
- bool [isUndefined](#) () const  
*Return 'true' if this color is undefined.*
- bool [isDefined](#) () const  
*Return 'true' if this color is defined.*

### 5.30.1 Detailed Description

Helper class to define an RGB color.

Definition at line 34 of file [YColor.h](#).

The documentation for this class was generated from the following file:

- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YColor.h

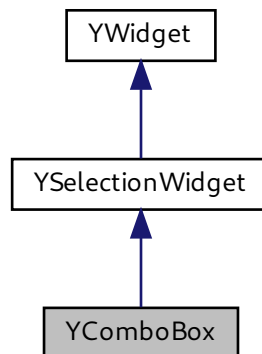


## 5.31 YComboBox Class Reference

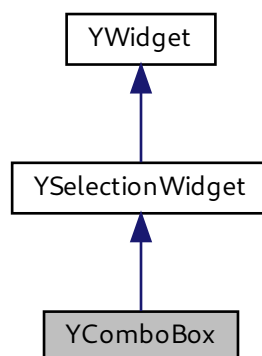
ComboBox (or "drop down box", "drop down selection"); may be editable.

```
#include <YComboBox.h>
```

Inheritance diagram for YComboBox:



Collaboration diagram for YComboBox:



## Public Member Functions

- virtual `~YComboBox ()`  
*Destructor.*
- virtual const char \* `widgetClass () const`  
*Returns a descriptive name of this widget class for logging, debugging etc.*
- bool `editable () const`  
*Return 'true' if this ComboBox is editable, i.e.*
- std::string `value ()`  
*Return the value of this ComboBox:*
- void `setValue (const std::string &newText)`  
*Set the value of this ComboBox by string: Try to find a list item with that label and select it.*
- virtual `YItem * selectedItem ()`  
*Return the (first) selected item or 0 if none is selected or if this ComboBox is editable and the user entered something that does not match any of the ComboBox's list items (in that case, use `value()` instead).*
- virtual `YItemCollection selectedItems ()`  
*Return all selected items.*
- virtual void `selectItem (YItem *item, bool selected=true)`  
*Select or deselect an item.*
- std::string `validChars ()`  
*Get the valid input characters.*
- virtual void `setValidChars (const std::string &validChars)`  
*Set the valid input characters.*
- int `inputMaxLength () const`  
*The maximum input length, i.e., the maximum number of characters the user can enter.*
- virtual void `setInputMaxLength (int numberOfChars)`  
*Set the maximum input length, i.e., the maximum number of characters the user can enter.*
- virtual bool `setProperty (const std::string &propertyName, const YPropertyValue &val)`  
*Set a property.*
- virtual `YPropertyValue getProperty (const std::string &propertyName)`  
*Get a property.*
- virtual const `YPropertySet &propertySet ()`  
*Return this class's property set.*
- const char \* `userInputProperty ()`  
*The name of the widget property that will return user input.*

## Protected Member Functions

- `YComboBox (YWidget *parent, const std::string &label, bool editable)`  
*Constructor.*
- virtual std::string `text ()=0`  
*Return this ComboBox's current value as text.*
- virtual void `setText (const std::string &newText)=0`  
*Set this ComboBox's current value as text.*

### 5.31.1 Detailed Description

ComboBox (or "drop down box", "drop down selection"); may be editable.

A widget with a drop-down list of predefined values to select from. Optionally, this widget can be created in "editable" mode which means that the user can freely enter any text.

In non-editable mode, a ComboBox works very much like a SelectionBox that uses fewer screen space. In that mode, it is recommended to use [selectedItem\(\)](#) to retrieve its current value and [selectItem\(\)](#) to set it.

In editable mode, a ComboBox is more like an InputField with a list to pick predefined values from (for less typing). In that mode, it is recommended to use [value\(\)](#) and [setValue\(\)](#).

In either mode, it might be dangerous to use the iterators the ([itemsBegin\(\)](#), [itemsEnd\(\)](#)) the base class ([YSelectionWidget](#)) provides to find the currently selected item: The items' "selected" flag may or may not be up to date. [YComboBox::selectedItem\(\)](#) makes sure they are up to date.

Definition at line 53 of file [YComboBox.h](#).

### 5.31.2 Constructor & Destructor Documentation

#### 5.31.2.1 YComboBox()

```
YComboBox::YComboBox (
    YWidget * parent,
    const std::string & label,
    bool editable ) [protected]
```

Constructor.

'editable' means the user can freely enter any value without being restricted to the items of the ComboBox's list.

Definition at line 51 of file [YComboBox.cc](#).

### 5.31.3 Member Function Documentation

#### 5.31.3.1 editable()

```
bool YComboBox::editable ( ) const
```

Return 'true' if this ComboBox is editable, i.e.

if the user can freely enter any value without being restricted to the items of the ComboBox's list.

Notice that this can only be set in the constructor.

Definition at line 66 of file [YComboBox.cc](#).

### 5.31.3.2 getProperty()

```
YPropertyValue YComboBox::getProperty (
    const std::string & propertyName ) [virtual]
```

Get a property.

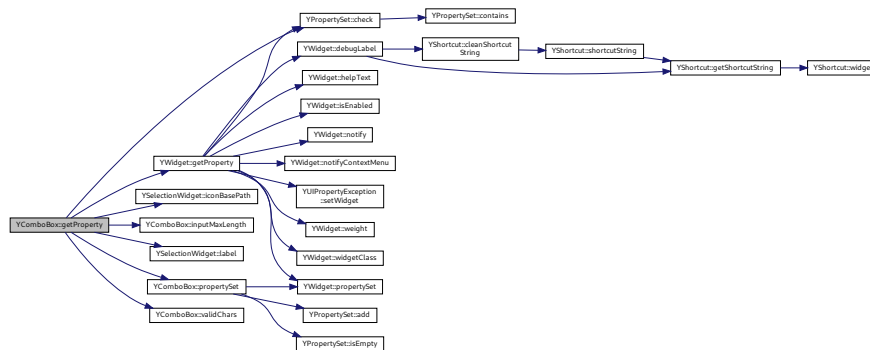
Reimplemented from [YWidget](#).

This method may throw [YUIPropertyExceptions](#).

Reimplemented from [YWidget](#).

Definition at line 222 of file [YComboBox.cc](#).

Here is the call graph for this function:



### 5.31.3.3 inputMaxLength()

```
int YComboBox::inputMaxLength ( ) const
```

The maximum input length, i.e., the maximum number of characters the user can enter.

-1 means no limit.

This is only meaningful for if the ComboBox is editable.

Definition at line 84 of file [YComboBox.cc](#).

#### 5.31.3.4 propertySet()

```
const YPropertySet & YComboBox::propertySet ( ) [virtual]
```

Return this class's property set.

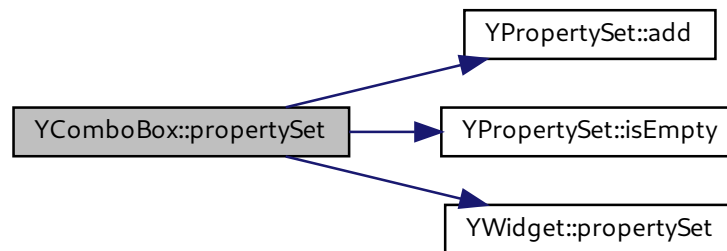
This also initializes the property upon the first call.

Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 174 of file [YComboBox.cc](#).

Here is the call graph for this function:



#### 5.31.3.5 selectedItem()

```
YItem * YComboBox::selectedItem ( ) [virtual]
```

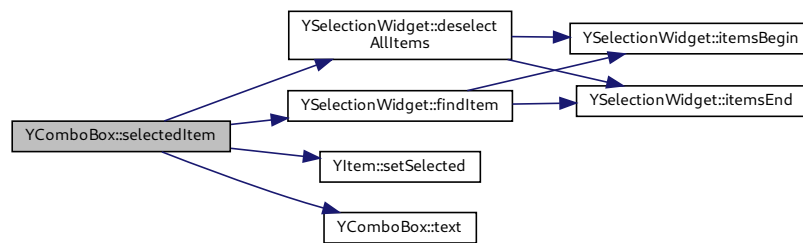
Return the (first) selected item or 0 if none is selected or if this ComboBox is editable and the user entered something that does not match any of the ComboBox's list items (in that case, use [value\(\)](#) instead).

Reimplemented from [YSelectionWidget](#) for better reliability: This will compare an editable ComboBox's user input against the text labels of all items and try to return an item if there is any match.

Reimplemented from [YSelectionWidget](#).

Definition at line 138 of file [YComboBox.cc](#).

Here is the call graph for this function:



### 5.31.3.6 selectedItems()

```
YItemCollection YComboBox::selectedItems ( ) [virtual]
```

Return all selected items.

This is not particularly useful for ComboBoxes since there can be no more than one selected item anyway; \* better use [selectedItem\(\)](#) or [value\(\)](#) instead.

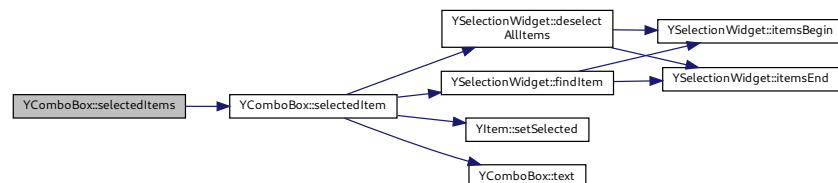
This function does not transfer ownership of those items to the caller, so don't try to delete them!

Reimplemented from [YSelectionWidget](#) for better reliability.

Reimplemented from [YSelectionWidget](#).

Definition at line 159 of file [YComboBox.cc](#).

Here is the call graph for this function:



### 5.31.3.7 selectItem()

```
void YComboBox::selectItem (
    YItem * item,
    bool selected = true ) [virtual]
```

Select or deselect an item.

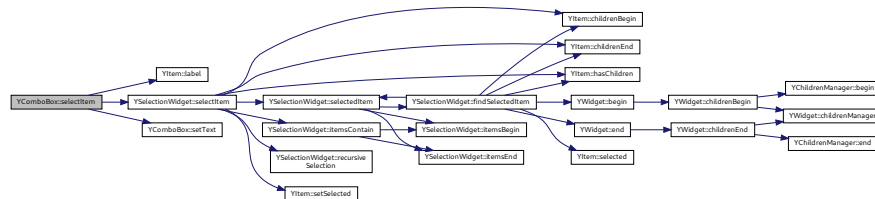
See also [setValue\(\)](#).

Reimplemented from [YSelectionWidget](#).

Reimplemented from [YSelectionWidget](#).

Definition at line 124 of file [YComboBox.cc](#).

Here is the call graph for this function:



### 5.31.3.8 setInputMaxLength()

```
void YComboBox::setInputMaxLength (
    int numberOfChars ) [virtual]
```

Set the maximum input length, i.e., the maximum number of characters the user can enter.

-1 means no limit.

This is only meaningful for if the ComboBox is editable.

Derived classes are free to reimplement this, but they should call this base class method at the end of the overloaded function.

Definition at line 90 of file [YComboBox.cc](#).

### 5.31.3.9 setProperty()

```
bool YComboBox::setProperty (
    const std::string & propertyName,
    const YPropertyValue & val ) [virtual]
```

Set a property.

Reimplemented from [YWidget](#).

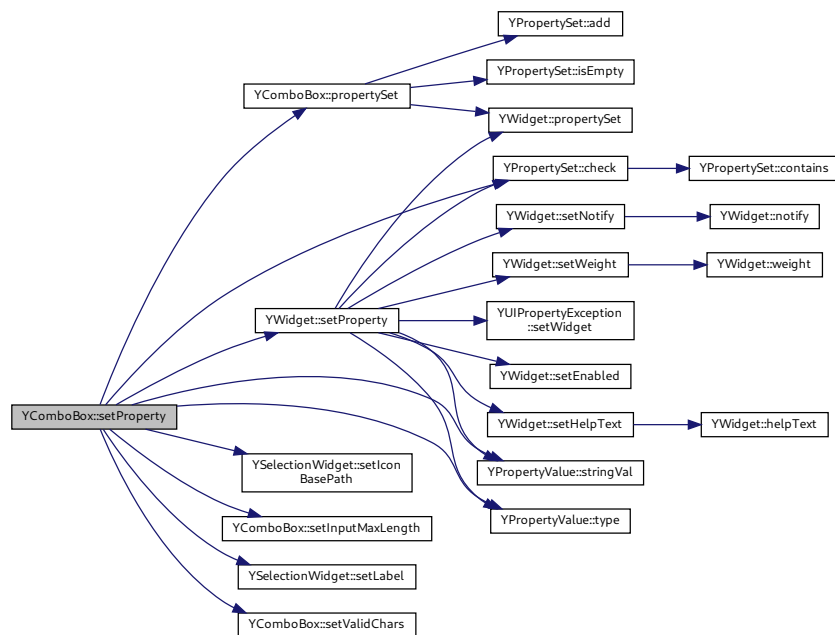
This function may throw [YUIPropertyExceptions](#).

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from [YWidget](#).

Definition at line 202 of file [YComboBox.cc](#).

Here is the call graph for this function:



### 5.31.3.10 setText()

```
virtual void YComboBox::setText (
    const std::string & newText ) [protected], [pure virtual]
```

Set this ComboBox's current value as text.

Called internally whenever the content is to change programmatically. Don't call [setValue\(\)](#) or [selectItem\(\)](#) from here.

Derived classes are required to implement this function.



**5.31.3.11 setValidChars()**

```
void YComboBox::setValidChars (
    const std::string & validChars ) [virtual]
```

Set the valid input characters.

No input validation is performed (i.e., the user can enter anything) if this is empty.

This is only meaningful for if the ComboBox is editable.

Derived classes are free to reimplement this, but they should call this base class method at the end of the overloaded function.

Definition at line 78 of file [YComboBox.cc](#).

**5.31.3.12 setValue()**

```
void YComboBox::setValue (
    const std::string & newText )
```

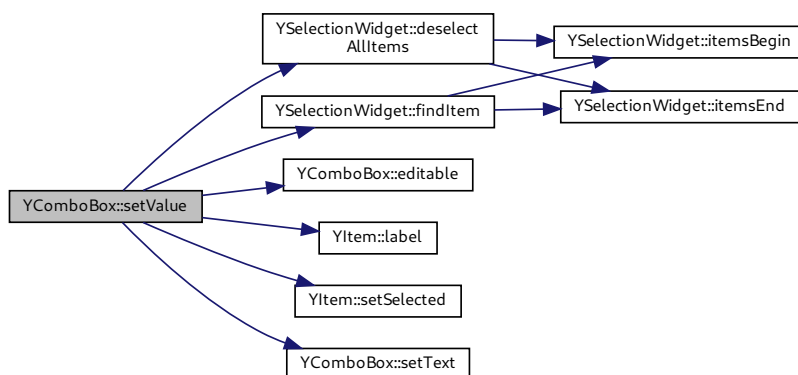
Set the value of this ComboBox by string: Try to find a list item with that label and select it.

If there is no matching list item, editable ComboBoxes will set their input field to that text. Non-editable ComboBoxes will throw an exception.

See also [selectItem\(\)](#).

Definition at line 102 of file [YComboBox.cc](#).

Here is the call graph for this function:



#### 5.31.3.13 text()

```
virtual std::string YComboBox::text ( ) [protected], [pure virtual]
```

Return this ComboBox's current value as text.

Called internally from [value\(\)](#), [selectedItem\(\)](#) and related.

Derived classes are required to implement this function.

#### 5.31.3.14 userInputProperty()

```
const char* YComboBox::userInputProperty ( ) [inline], [virtual]
```

The name of the widget property that will return user input.

Inherited from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line [211](#) of file [YComboBox.h](#).

#### 5.31.3.15 validChars()

```
string YComboBox::validChars ( )
```

Get the valid input characters.

No input validation is performed (i.e., the user can enter anything) if this is empty.

This is only meaningful for if the ComboBox is editable.

Definition at line [72](#) of file [YComboBox.cc](#).

### 5.31.3.16 value()

```
string YComboBox::value ( )
```

Return the value of this ComboBox:

The text of a list item if the user (or the application) selected a list item or the content of the ComboBox's input field if the ComboBox is editable and the user (or the application) entered text there.

See also [YComboBox::selectedItem\(\)](#).

Definition at line 96 of file [YComboBox.cc](#).

Here is the call graph for this function:

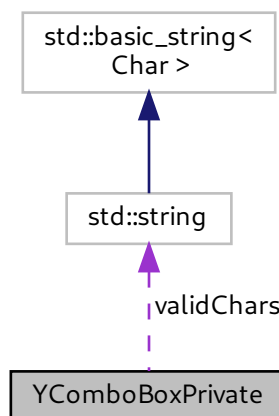


The documentation for this class was generated from the following files:

- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YComboBox.h
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YComboBox.cc

## 5.32 YComboBoxPrivate Struct Reference

Collaboration diagram for YComboBoxPrivate:



## Public Member Functions

- **YComboBoxPrivate** (bool editable)

## Public Attributes

- bool **editable**
- string **validChars**
- int **inputMaxLength**

### 5.32.1 Detailed Description

Definition at line 36 of file [YComboBox.cc](#).

The documentation for this struct was generated from the following file:

- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YComboBox.cc

## 5.33 YCommandLine Class Reference

Utility class to access /proc/<pid>/cmdline to retrieve argc and argv.

```
#include <YCommandLine.h>
```

## Public Member Functions

- [YCommandLine](#) ()  
*Constructor.*
- [~YCommandLine](#) ()  
*Destructor.*
- int [argc](#) () const  
*Return the number of arguments in the command line.*
- char \*\* [argv](#) () const  
*Return the arguments in a C compatible fashion: An array of pointers to characters.*
- int [size](#) () const  
*Alias for [argc\(\)](#) for those who like a more C++ -like syntax.*
- std::string [arg](#) (int index) const  
*Return command line argument no.*
- std::string [operator\[\]](#) (int index) const  
*Return command line argument no.*
- void [add](#) (const std::string &[arg](#))  
*Add a command line argument (at the end of the existing ones).*
- void [remove](#) (int index)  
*Remove command line argument no.*
- void [replace](#) (int index, const std::string &[arg](#))  
*Replace command line argument no.*
- int [find](#) (const std::string &argName) const  
*Find a command line argument 'argName' ("display" etc.).*

### 5.33.1 Detailed Description

Utility class to access `/proc/<pid>/cmdline` to retrieve `argc` and `argv`.

Definition at line 37 of file [YCommandLine.h](#).

### 5.33.2 Constructor & Destructor Documentation

#### 5.33.2.1 YCommandLine()

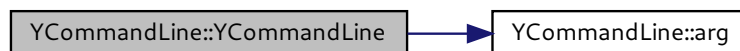
```
YCommandLine::YCommandLine ( )
```

Constructor.

This will read `/proc/<pid>/cmdline` of this process.

Definition at line 50 of file [YCommandLine.cc](#).

Here is the call graph for this function:



### 5.33.3 Member Function Documentation

#### 5.33.3.1 arg()

```
string YCommandLine::arg (
    int index ) const
```

Return command line argument no.

'index' (from 0 on).

This might throw an [YUIIndexOutOfRangeException](#).

Definition at line 111 of file [YCommandLine.cc](#).

### 5.33.3.2 argc()

```
int YCommandLine::argc ( ) const
```

Return the number of arguments in the command line.

Remember that the command itself (the binary of the process) is included, so a value of 1 (not 0!) means "no additional arguments".

Definition at line 80 of file [YCommandLine.cc](#).

### 5.33.3.3 argv()

```
char ** YCommandLine::argv ( ) const
```

Return the arguments in a C compatible fashion: An array of pointers to characters.

The data are copied with `strdup()`, so they are valid beyond the life time of this object (but OTOH should be released with `free()` at some point).

Definition at line 87 of file [YCommandLine.cc](#).

Here is the call graph for this function:



#### 5.33.3.4 find()

```
int YCommandLine::find (
    const std::string & argName ) const
```

Find a command line argument 'argName' ("-display" etc.).

Notice that leading minus signs must be specified in 'argName'. Since argv[0] is the program name, the search starts from argv[1].

Return the position of 'argName' (from 0 on) or -1 if not found.

Definition at line 138 of file [YCommandLine.cc](#).

Here is the call graph for this function:



#### 5.33.3.5 operator[]()

```
std::string YCommandLine::operator[] (
    int index ) const [inline]
```

Return command line argument no.

'index' (from 0 on) as operator[]:

```
for ( int i=0; i < cmdLine.argc(); i++ ) cout << cmdLine[i] << std::endl;
```

This might throw an [YUIIndexOutOfRangeException](#).

Definition at line 85 of file [YCommandLine.h](#).

Here is the call graph for this function:



#### 5.33.3.6 remove()

```
void YCommandLine::remove (
    int index )
```

Remove command line argument no.

'index' (from 0 on).

This might throw an [YUIIndexOutOfRangeException](#).

Definition at line [120](#) of file [YCommandLine.cc](#).

#### 5.33.3.7 replace()

```
void YCommandLine::replace (
    int index,
    const std::string & arg )
```

Replace command line argument no.

'index' (from 0 on) with 'arg'.

This might throw an [YUIIndexOutOfRangeException](#).

Definition at line [129](#) of file [YCommandLine.cc](#).

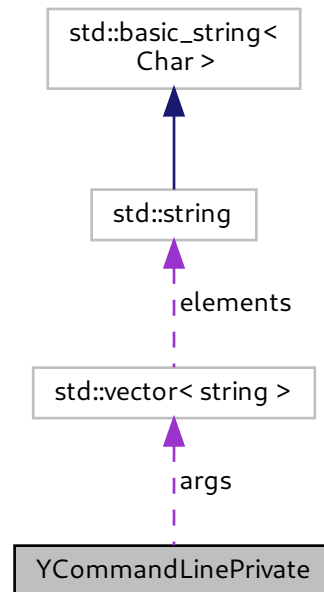
The documentation for this class was generated from the following files:

- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YCommandLine.h](#)
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YCommandLine.cc](#)



## 5.34 YCommandLinePrivate Struct Reference

Collaboration diagram for YCommandLinePrivate:



### Public Attributes

- `std::vector< string > args`

#### 5.34.1 Detailed Description

Definition at line 41 of file [YCommandLine.cc](#).

The documentation for this struct was generated from the following file:

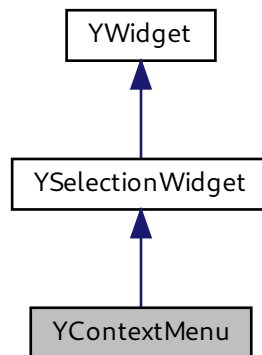
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YCommandLine.cc`

## 5.35 YContextMenu Class Reference

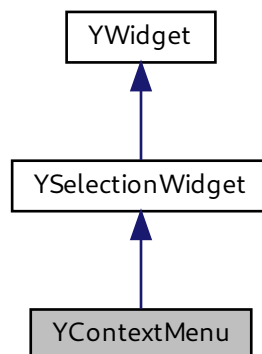
ContextMenu: Similar to PushButton, but with several actions: Upon clicking on a ContextMenu (or activating it with the keyboard), a pop-up menu opens where the user can activate an action.

```
#include <YContextMenu.h>
```

Inheritance diagram for YContextMenu:



Collaboration diagram for YContextMenu:



## Public Member Functions

- virtual [~YContextMenu](#) ()  
*Destructor.*
- virtual const char \* [widgetClass](#) () const  
*Returns a descriptive name of this widget class for logging, debugging etc.*
- virtual void [rebuildMenuTree](#) ()=0  
*Rebuild the displayed menu tree from the internally stored YMenuItems.*
- virtual void [addItems](#) (const [YItemCollection](#) &itemCollection)  
*Add multiple items.*
- virtual void [addItem](#) ([YItem](#) \*item\_disown)  
*Add one item.*
- virtual void [deleteAllItems](#) ()  
*Delete all items.*
- void [resolveShortcutConflicts](#) ()  
*Resolve keyboard shortcut conflicts: Change shortcuts of menu items if there are duplicates in the respective menu level.*
- virtual bool [setProperty](#) (const std::string &propertyName, const [YPropertyValue](#) &val)  
*Set a property.*
- virtual [YPropertyValue](#) [getProperty](#) (const std::string &propertyName)  
*Get a property.*
- virtual const [YPropertySet](#) & [propertySet](#) ()  
*Return this class's property set.*

## Protected Member Functions

- [YContextMenu](#) ()  
*Constructor.*
- [YMenuItem](#) \* [findMenuItem](#) (int index)  
*Recursively find the first menu item with the specified index.*
- [YMenuItem](#) \* [findMenuItem](#) (int index, [YItemConstIterator](#) begin, [YItemConstIterator](#) end)  
*Recursively find the first menu item with the specified index from iterator 'begin' to iterator 'end'.*
- [YMenuItem](#) \* [itemAt](#) (int index)  
*Alias for [findMenuItem\(\)](#).*

### 5.35.1 Detailed Description

ContextMenu: Similar to PushButton, but with several actions: Upon clicking on a ContextMenu (or activating it with the keyboard), a pop-up menu opens where the user can activate an action.

Menu items in that pop-up menu can have submenus (that will pop up in separate pop-up menus).

Internally, this widget is more similar to the Tree widget. The difference is that it does not keep a "selected" status, but triggers an action right away, just like a PushButton. Like PushButton, ContextMenu sends an event right away when the user selects an item (clicks on a menu item or activates it with the keyboard). Items that have a submenu never send an event, they simply open their submenu when activated.

Note: unlike other widgets, this one is not created via [YWidgetFactory](#) or [YOptionalWidgetFactory](#) but with [YApplication::openContextMenu\(\)](#)

Definition at line 51 of file [YContextMenu.h](#).

## 5.35.2 Constructor & Destructor Documentation

### 5.35.2.1 YContextMenu()

```
YContextMenu::YContextMenu ( ) [protected]
```

Constructor.

'label' is the user-visible text on the button (not above it like all other SelectionWidgets).

Definition at line 48 of file [YContextMenu.cc](#).

## 5.35.3 Member Function Documentation

### 5.35.3.1 addItem()

```
void YContextMenu::addItem (
    YItem * item_disown ) [virtual]
```

Add one item.

This widget assumes ownership of the item object and will delete it in its destructor.

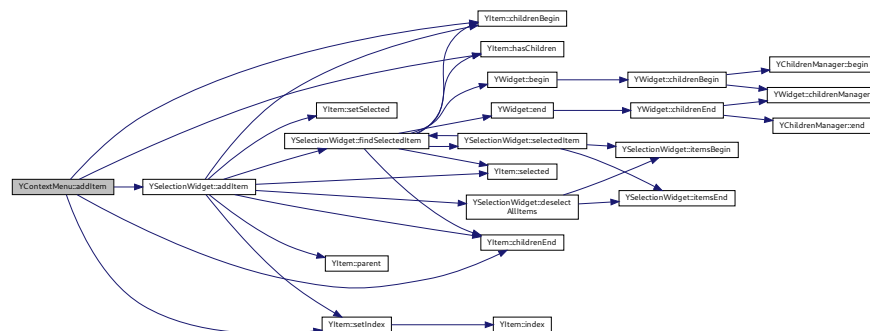
This reimplement will an index to the item that is unique for all items in this ContextMenu. That index can be used later with [findMenuItem\(\)](#) to find the item by that index.

Reimplemented from [YSelectionWidget](#).

Reimplemented from [YSelectionWidget](#).

Definition at line 73 of file [YContextMenu.cc](#).

Here is the call graph for this function:



### 5.35.3.2 addItem()

```
void YContextMenu::addItem (
    const YItemCollection & itemCollection ) [virtual]
```

Add multiple items.

For some UIs, this can be more efficient than calling [addItem\(\)](#) multiple times. This function also automatically calls [resolveShortcutConflicts\(\)](#) and [rebuildMenuTree\(\)](#) at the end.

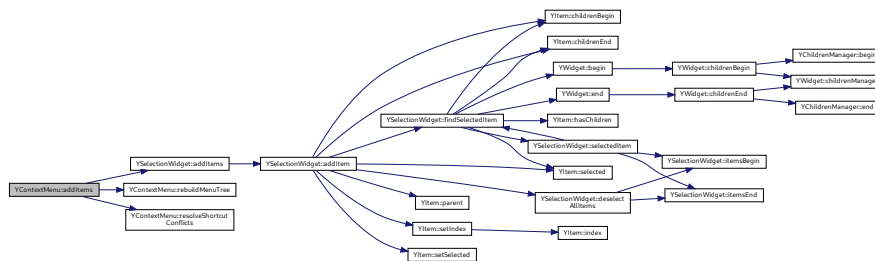
Derived classes can overwrite this function, but they should call this base class function at the end of the new implementation.

Reimplemented from [YSelectionWidget](#).

Reimplemented from [YSelectionWidget](#).

Definition at line 64 of file [YContextMenu.cc](#).

Here is the call graph for this function:



### 5.35.3.3 deleteAllItems()

```
void YContextMenu::deleteAllItems ( ) [virtual]
```

Delete all items.

Reimplemented from [YSelectionWidget](#).

Reimplemented from [YSelectionWidget](#).

Definition at line 99 of file [YContextMenu.cc](#).

Here is the call graph for this function:



### 5.35.3.4 findMenuItem() [1/2]

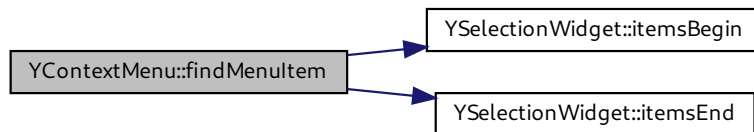
```
YMenuItem * YContextMenu::findMenuItem (
    int index ) [protected]
```

Recursively find the first menu item with the specified index.

Returns 0 if there is no such item.

Definition at line 107 of file [YContextMenu.cc](#).

Here is the call graph for this function:



### 5.35.3.5 findMenuItem() [2/2]

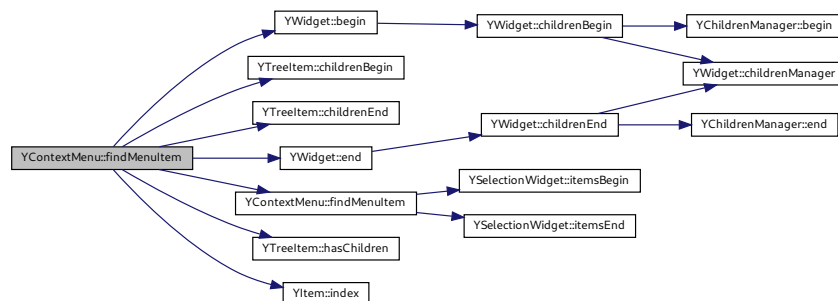
```
YMenuItem * YContextMenu::findMenuItem (
    int index,
    YItemConstIterator begin,
    YItemConstIterator end ) [protected]
```

Recursively find the first menu item with the specified index from iterator 'begin' to iterator 'end'.

Returns 0 if there is no such item.

Definition at line 114 of file [YContextMenu.cc](#).

Here is the call graph for this function:



### 5.35.3.6 getProperty()

```
YPropertyValue YContextMenu::getProperty (
    const std::string & propertyName ) [virtual]
```

Get a property.

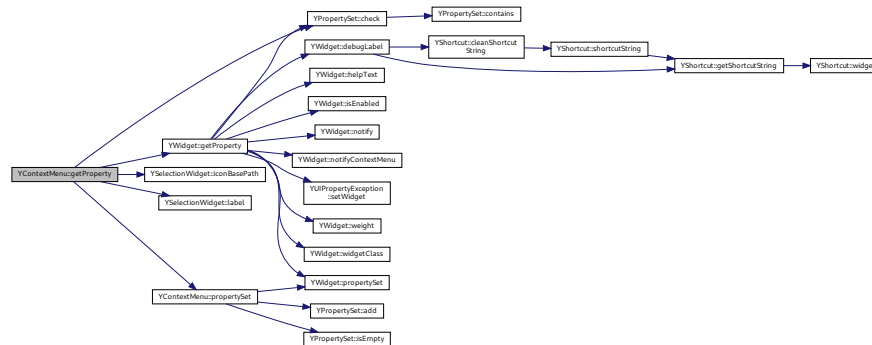
Reimplemented from [YWidget](#).

This method may throw [YUIPropertyExceptions](#).

Reimplemented from [YWidget](#).

Definition at line 194 of file [YContextMenu.cc](#).

Here is the call graph for this function:



### 5.35.3.7 itemAt()

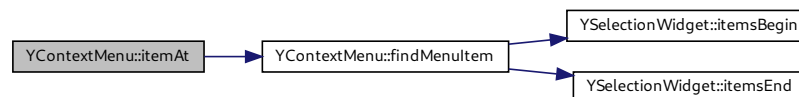
```
YMenuItem* YContextMenu::itemAt (
    int index ) [inline], [protected]
```

Alias for [findMenuItem\(\)](#).

Reimplemented to ensure consistent behaviour with [YSelectionWidget::itemAt\(\)](#).

Definition at line 173 of file [YContextMenu.h](#).

Here is the call graph for this function:



### 5.35.3.8 propertySet()

```
const YPropertySet & YContextMenu::propertySet ( ) [virtual]
```

Return this class's property set.

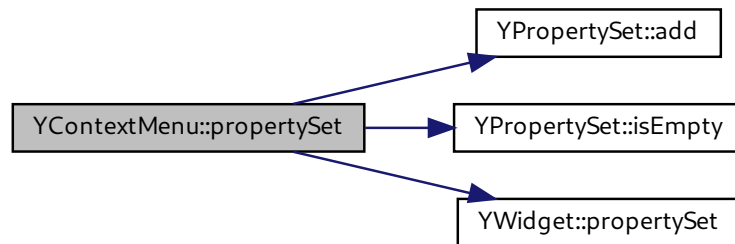
This also initializes the property upon the first call.

Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 155 of file [YContextMenu.cc](#).

Here is the call graph for this function:



### 5.35.3.9 rebuildMenuTree()

```
virtual void YContextMenu::rebuildMenuTree ( ) [pure virtual]
```

Rebuild the displayed menu tree from the internally stored YMenuItems.

The application should call this (once) after all items have been added with [addItem\(\)](#). [YContextMenu::addItem\(\)](#) calls this automatically.

Derived classes are required to implement this.



**5.35.3.10 resolveShortcutConflicts()**

```
void YContextMenu::resolveShortcutConflicts ( )
```

Resolve keyboard shortcut conflicts: Change shortcuts of menu items if there are duplicates in the respective menu level.

This has to be called after all items are added, but before [rebuildMenuTree\(\)](#) (see above). [YContextMenu::addItem\(\)](#) calls this automatically.

Definition at line 140 of file [YContextMenu.cc](#).

**5.35.3.11 setProperty()**

```
bool YContextMenu::setProperty (
    const std::string & propertyName,
    const YPropertyValue & val ) [virtual]
```

Set a property.

Reimplemented from [YWidget](#).

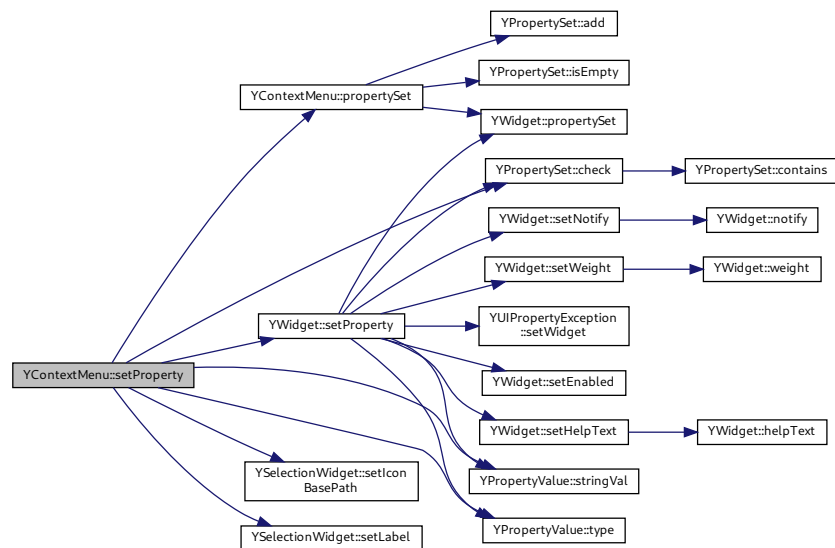
This function may throw YUIPropertyExceptions.

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from [YWidget](#).

Definition at line 177 of file [YContextMenu.cc](#).

Here is the call graph for this function:



The documentation for this class was generated from the following files:

- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YContextMenu.h`
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YContextMenu.cc`

## 5.36 YContextMenuPrivate Struct Reference

### Public Attributes

- int `nextSerialNo`

### 5.36.1 Detailed Description

Definition at line 36 of file [YContextMenu.cc](#).

The documentation for this struct was generated from the following file:

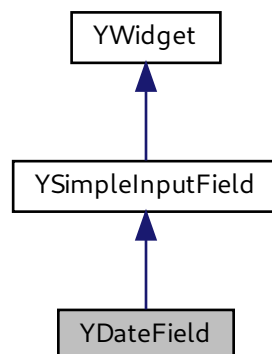
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YContextMenu.cc`

## 5.37 YDateField Class Reference

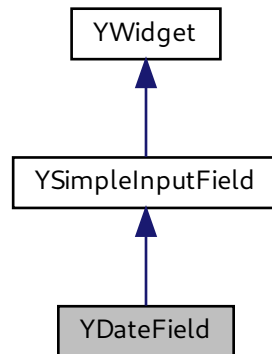
Input field for entering a date.

```
#include <YDateField.h>
```

Inheritance diagram for YDateField:



Collaboration diagram for YDateField:



## Public Member Functions

- virtual [~YDateField](#) ()  
*Destructor.*
- virtual const char \* [widgetClass](#) () const  
*Returns a descriptive name of this widget class for logging, debugging etc.*

## Protected Member Functions

- [YDateField](#) (YWidget \*parent, const std::string &label)  
*Constructor.*

### 5.37.1 Detailed Description

Input field for entering a date.

Derived classes are required to implement: [value\(\)](#) [setValue\(\)](#)

For both methods the date is formatted as "YYYY-MM-DD". See [YSimpleInputField.h](#) for more details.

Definition at line [42](#) of file [YDateField.h](#).

The documentation for this class was generated from the following files:

- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YDateField.h
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YDateField.cc

## 5.38 YDateFieldPrivate Struct Reference

### Public Attributes

- bool **dummy**

### 5.38.1 Detailed Description

Definition at line 34 of file [YDateField.cc](#).

The documentation for this struct was generated from the following file:

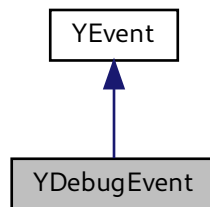
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YDateField.cc

## 5.39 YDebugEvent Class Reference

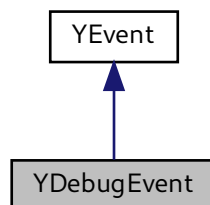
Event to be returned upon closing a dialog with the window manager close button (or Alt-F4)

```
#include <YEvent.h>
```

Inheritance diagram for YDebugEvent:



Collaboration diagram for YDebugEvent:



## Protected Member Functions

- virtual [~YDebugEvent](#) ()

*Protected destructor - events can only be deleted via [YDialog::deleteEvent\(\)](#).*

## Additional Inherited Members

### 5.39.1 Detailed Description

Event to be returned upon closing a dialog with the window manager close button (or Alt-F4)

Definition at line [326](#) of file [YEvent.h](#).

### 5.39.2 Constructor & Destructor Documentation

#### 5.39.2.1 ~YDebugEvent()

```
virtual YDebugEvent::~~YDebugEvent ( ) [inline], [protected], [virtual]
```

Protected destructor - events can only be deleted via [YDialog::deleteEvent\(\)](#).

The associated dialog will take care of this event and delete it when appropriate.

Definition at line [338](#) of file [YEvent.h](#).

The documentation for this class was generated from the following file:

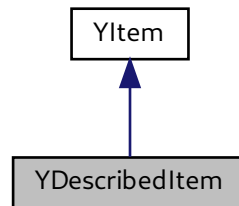
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YEvent.h](#)

## 5.40 YDescribedItem Class Reference

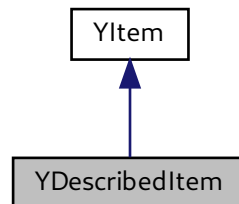
Item class that has a (possibly multi-line) description text in addition to the normal label.

```
#include <YDescribedItem.h>
```

Inheritance diagram for YDescribedItem:



Collaboration diagram for YDescribedItem:



### Public Member Functions

- [YDescribedItem](#) (const std::string &[label](#), const std::string &[description](#)="", bool [selected](#)=false)  
*Constructor with the label, the description and optionally the selected state.*
- [YDescribedItem](#) (const std::string &[label](#), const std::string &[description](#), const std::string &[iconName](#), bool [selected](#)=false)  
*Constructor with the label, the description, the icon name and optionally the selected state.*
- virtual [~YDescribedItem](#) ()  
*Destructor.*
- std::string [description](#) () const

*Return this item's description text.*

- void [setDescription](#) (const std::string &desc)

*Set this item's description text.*

- bool [enabled](#) () const

*Return 'true' if this item is enabled (which is the default).*

- void [setEnabled](#) (bool value)

*Set this item to enabled or disabled.*

### 5.40.1 Detailed Description

Item class that has a (possibly multi-line) description text in addition to the normal label.

Definition at line 35 of file [YDescribedItem.h](#).

### 5.40.2 Member Function Documentation

#### 5.40.2.1 description()

```
std::string YDescribedItem::description ( ) const [inline]
```

Return this item's description text.

This is the (typically longer) subtext that the user sees in a dialog, so this will usually be a translated text.

Definition at line 74 of file [YDescribedItem.h](#).

#### 5.40.2.2 enabled()

```
bool YDescribedItem::enabled ( ) const [inline]
```

Return 'true' if this item is enabled (which is the default).

Items are only ever disabled if the application explicitly sets them to disabled.

Definition at line 85 of file [YDescribedItem.h](#).

### 5.40.2.3 `setEnabled()`

```
void YDescribedItem::setEnabled (
    bool value ) [inline]
```

Set this item to enabled or disabled.

Notice that this only stores that status internally. To have any effect on an associated widget, use the widget's method to enable or disable an item (which will usually call this method internally at some point).

Definition at line 94 of file [YDescribedItem.h](#).

The documentation for this class was generated from the following file:

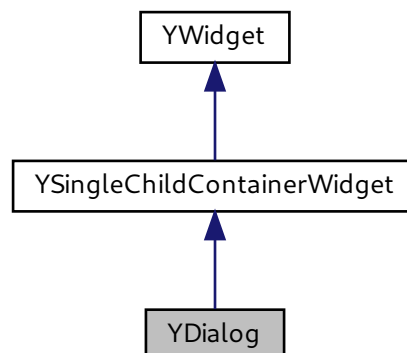
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YDescribedItem.h`

## 5.41 YDialog Class Reference

A window in the desktop environment.

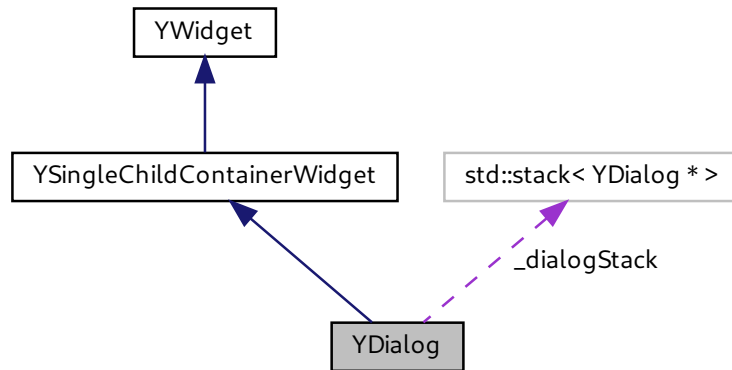
```
#include <YDialog.h>
```

Inheritance diagram for YDialog:





Collaboration diagram for YDialog:



## Public Member Functions

- virtual const char \* **widgetClass** () const  
*Return a descriptive name of this widget class for logging, debugging etc.*
- void **open** ()  
*Open a newly created dialog: Finalize it and make it visible on the screen.*
- bool **isOpen** () const  
*Return 'true' if **open()** has already been called for this dialog.*
- YEvent \* **waitForEvent** (int timeout\_millsec=0)  
*Wait for a user event.*
- YEvent \* **pollEvent** ()  
*Check if a user event is pending.*
- bool **isTopmostDialog** () const  
*Return 'true' if this dialog is the topmost dialog.*
- void **requestMultiPassLayout** ()  
*Request multiple passes of the layout engine.*
- int **layoutPass** () const  
*Return the number of the current layout pass: 0: No layout going on right now 1: First pass 2: Second pass of a multi-pass layout.*
- bool **destroy** (bool doThrow=true)  
*Close and delete this dialog (and all its children) if it is the topmost dialog.*
- void **setInitialSize** ()  
*Set the initial dialog size, depending on dialogType: YMainDialog dialogs get the UI's "default main window" size, YPopupDialog dialogs use their content's preferred size.*
- void **recalcLayout** ()  
*Recalculate the layout of the dialog and of all its children after children have been added or removed or if any of them changed its preferred width of height.*
- YDialogType **dialogType** () const

- Return this dialog's type (YMainDialog / YPopupDialog / YWizardDialog).*

  - bool `isMainDialog` ()

*Return 'true' if this dialog is a dialog of main dialog size: YMainDialog or YWizardDialog.*
- YDialogColorMode `colorMode` () const

*Return this dialog's color mode.*
- void `checkShortcuts` (bool force=false)

*Checks the keyboard shortcuts of widgets in this dialog unless shortcut checks are postponed or 'force' is 'true'.*
- void `postponeShortcutCheck` ()

*From now on, postpone keyboard shortcut checks - i.e.*
- bool `shortcutCheckPostponed` () const

*Return whether or not shortcut checking is currently postponed.*
- YPushButton \* `defaultButton` () const

*Return this dialog's default button: The button that is activated when the user hits [Return] anywhere in this dialog.*
- void `deleteEvent` (YEvent \*event)

*Delete an event.*
- void `addEventFilter` (YEventFilter \*eventFilter)

*Add an event filter.*
- void `removeEventFilter` (YEventFilter \*eventFilter)

*Remove an event filter.*
- virtual void `highlight` (YWidget \*child)

*Highlight a child widget of this dialog.*
- virtual void `setDefaultButton` (YPushButton \*defaultButton)

*Set this dialog's default button (the button that is activated when the user hits [Return] anywhere in this dialog).*
- virtual void `activate` ()=0

*Activate this dialog: Make sure that it is shown as the topmost dialog of this application and that it can receive input.*

## Static Public Member Functions

- static bool `deleteTopmostDialog` (bool doThrow=true)

*Delete the topmost dialog.*
- static void `deleteAllDialogs` ()

*Delete all open dialogs.*
- static void `deleteTo` (YDialog \*dialog)

*Delete all dialogs from the topmost to the one specified.*
- static int `openDialogsCount` ()

*Returns the number of currently open dialogs (from 1 on), i.e., the depth of the dialog stack.*
- static YDialog \* `currentDialog` (bool doThrow=true)

*Return the current (topmost) dialog.*
- static YDialog \* `topmostDialog` (bool doThrow=true)

*Alias for `currentDialog()`.*
- static void `showText` (const std::string &text, bool richText=false)

*Show the specified text in a pop-up dialog with a local event loop.*
- static bool `showHelpText` (YWidget \*widget)

*Show the help text for the specified widget.*
- static bool `showRelNotesText` ()

*Show the release notes.*

## Protected Member Functions

- [YDialog](#) ([YDialogType](#) dialogType, YDialogColorMode colorMode=YDialogNormalColor)  
*Constructor.*
- virtual [~YDialog](#) ()  
*Destructor.*
- virtual void [openInternal](#) ()=0  
*Internal [open\(\)](#) method.*
- void [doLayout](#) ()  
*Calculate the layout and set the size of the dialog and all widgets.*
- virtual [YEvent](#) \* [waitForEventInternal](#) (int timeout\_millisec)=0  
*Wait for a user event.*
- virtual [YEvent](#) \* [pollEventInternal](#) ()=0  
*Check if a user event is pending.*
- [YEvent](#) \* [filterInvalidEvents](#) ([YEvent](#) \*event)  
*Filter out invalid events: Return 0 if the event does not belong to this dialog or the unchanged event if it does.*
- [YEvent](#) \* [callEventFilters](#) ([YEvent](#) \*event)  
*Call the installed event filters.*
- void [deleteEventFilters](#) ()  
*Delete all (remaining) event filters.*

## Static Protected Attributes

- static std::stack< [YDialog](#) \* > [\\_dialogStack](#)  
*Stack holding all currently existing dialogs.*

### 5.41.1 Detailed Description

A window in the desktop environment.

A YPopupDialog always has a dedicated window but YMainDialog may be stacked in a single window.

Definition at line 47 of file [YDialog.h](#).

### 5.41.2 Constructor & Destructor Documentation

### 5.41.2.1 YDialog()

```
YDialog::YDialog (
    YDialogType dialogType,
    YDialogColorMode colorMode = YDialogNormalColor ) [protected]
```

Constructor.

'dialogType' is one of YMainDialog or YPopupDialog.

'colorMode' can be set to YDialogWarnColor to use very bright "warning" colors or YDialogInfoColor to use more prominent, yet not quite as bright as "warning" colors. Use both only very rarely.

Definition at line 136 of file [YDialog.cc](#).

### 5.41.2.2 ~YDialog()

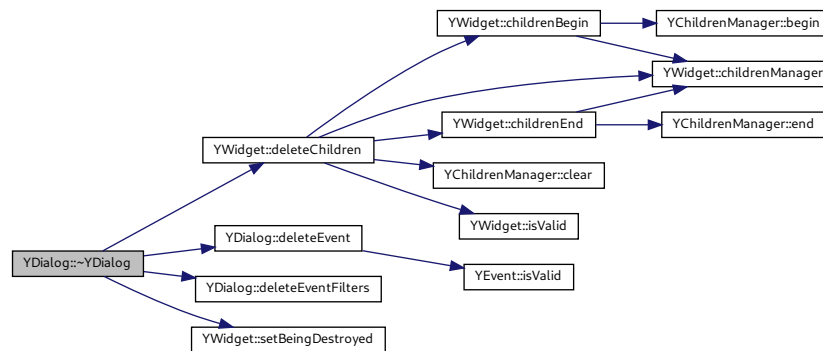
```
YDialog::~YDialog ( ) [protected], [virtual]
```

Destructor.

Don't delete a dialog directly, use [YDialog::deleteTopmostDialog\(\)](#) or [YDialog::destroy\(\)](#).

Definition at line 153 of file [YDialog.cc](#).

Here is the call graph for this function:



## 5.41.3 Member Function Documentation



#### 5.41.3.4 `currentDialog()`

```
YDialog * YDialog::currentDialog (
    bool doThrow = true ) [static]
```

Return the current (topmost) dialog.

If there is none, throw a [YUINoDialogException](#) if 'doThrow' is 'true' and return 0 if 'doThrow' is false.

Definition at line 539 of file [YDialog.cc](#).

#### 5.41.3.5 `defaultButton()`

```
YPushButton * YDialog::defaultButton ( ) const
```

Return this dialog's default button: The button that is activated when the user hits [Return] anywhere in this dialog.

Note that this is not the same as the button that currently has the keyboard focus.

This might return 0 if there is no default button.

Definition at line 323 of file [YDialog.cc](#).

#### 5.41.3.6 `deleteTopmostDialog()`

```
bool YDialog::deleteTopmostDialog (
    bool doThrow = true ) [static]
```

Delete the topmost dialog.

Will throw a [YUINoDialogException](#) if there is no dialog and 'doThrow' is 'true'.

This is equivalent to [YDialog::currentDialog\(\)->destroy\(\)](#).

Returns 'true' if there is another open dialog after deleting, 'false' if there is none.

Definition at line 553 of file [YDialog.cc](#).

### 5.41.3.7 destroy()

```
bool YDialog::destroy (
    bool doThrow = true )
```

Close and delete this dialog (and all its children) if it is the topmost dialog.

If this is not the topmost dialog, this will throw an exception if 'doThrow' is true (default).

Remember that all pointers to the dialog and its children will be invalid after this operation.

This is intentionally not named close() since close() would not imply that the dialog and its children are deleted.

Returns 'true' upon success, 'false' upon failure.

Definition at line [238](#) of file [YDialog.cc](#).

Here is the call graph for this function:



### 5.41.3.8 filterInvalidEvents()

```
YEvent * YDialog::filterInvalidEvents (
    YEvent * event ) [protected]
```

Filter out invalid events: Return 0 if the event does not belong to this dialog or the unchanged event if it does.

Silently discard events from widgets that have become invalid.

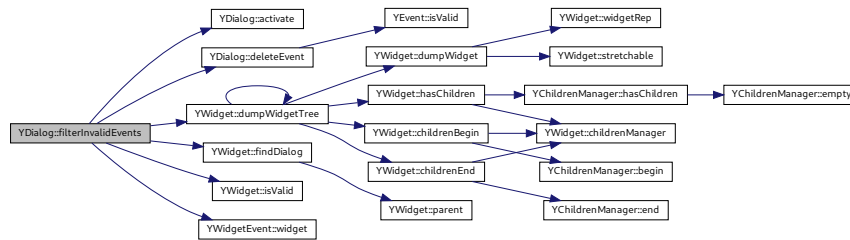
This may legitimately happen if some widget triggered an event yet nobody cared for that event (i.e. called `UserInput()` or `PollInput()`) and the widget has been destroyed meanwhile.

Silently discard events from all but the current (topmost) dialog.

This may happen even here even though the specific UI should have taken care about that: Events may still be in the queue. They might have been valid (i.e. belonged to the topmost dialog) when they arrived, but maybe simply nobody has evaluated them.

Definition at line [452](#) of file [YDialog.cc](#).

Here is the call graph for this function:



#### 5.41.3.9 highlight()

```
virtual void YDialog::highlight (
    YWidget * child ) [inline], [virtual]
```

Highlight a child widget of this dialog.

This is meant for debugging: [YDialogSpy](#) and similar uses.

No more than one widget can be highlighted at any one time in the same dialog. Highlighting another widget un-highlights a previously highlighted widget. 0 means 'unhighlight the last highlighted widget, but don't highlight any other'.

This default implementation does nothing.

Definition at line 334 of file [YDialog.h](#).

#### 5.41.3.10 open()

```
void YDialog::open ( )
```

Open a newly created dialog: Finalize it and make it visible on the screen.

Applications should call this once after all children are created. If the application doesn't do this, it will be done automatically upon the next call of [YDialog::waitForEvent\(\)](#) (or related). This is OK if [YDialog::waitForEvent\(\)](#) is called immediately after creating the dialog anyway. If it is not, the application might appear sluggish to the user.

Derived classes are free to reimplement this, but they should call this base class method in the new implementation.

Definition at line 189 of file [YDialog.cc](#).



```
virtual void YDialog::openInternal ( ) [protected], [pure virtual]
```

This is called (exactly once during the life time of the dialog) in `open()`.

### 5.41.3.12 pollEvent()

Check if a user event is pending.

If `open()` has not been called for this dialog until now, it is called now.

If this dialog is not the topmost dialog, an exception is thrown.

Generated by Doxygen



### 5.41.3.15 recalcLayout()

```
void YDialog::recalcLayout ( )
```

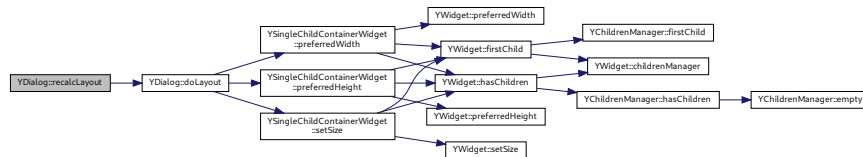
Recalculate the layout of the dialog and of all its children after children have been added or removed or if any of them changed its preferred width or height.

This is a very expensive operation. Call it only when really necessary. [YDialog::open\(\)](#) includes a call to [YDialog::setInitialSize\(\)](#) which does the same.

The basic idea behind this function is to call it when the dialog changed after it (and its children hierarchy) was initially created.

Definition at line 356 of file [YDialog.cc](#).

Here is the call graph for this function:



### 5.41.3.16 removeEventFilter()

```
void YDialog::removeEventFilter (
    YEventFilter * eventFilter )
```

Remove an event filter.

Notice that applications never need to call this function: [YEventFilter](#) does it automatically in its destructor.

Definition at line 630 of file [YDialog.cc](#).

### 5.41.3.17 requestMultiPassLayout()

```
void YDialog::requestMultiPassLayout ( )
```

Request multiple passes of the layout engine.

This is intended for widgets that don't know their preferred width and height immediately because one depends on the other, for example labels with auto-wrapping: They know (roughly) the area they will need, but they can calculate their preferred height only when their width is known.

Once set, this option cannot be unset; it remains set for the life time of the dialog.

Definition at line 662 of file [YDialog.cc](#).



```
bool YDialog::showRelNotesText ( ) [static]
```

If there are release notes, they are displayed in a pop-up dialog with a local event loop. This returns 'true' on success (there were relnotes) and 'false' on failure (no relnotes).

Definition at line 134 of file YDialogHelpers.cc.

[illegible]

```
void YDialog::showText (
    const std::string & text,
    bool richText = false ) [static]
```

This is useful for help texts. 'richText' indicates if **YRichText** formatting should be applied.

Definition at line 56 of file YDialogHelpers.cc.

[illegible]

### 5.41.3.22 waitForEvent()

```
YEvent * YDialog::waitForEvent (
    int timeout_millisec = 0 )
```

Wait for a user event.

In most cases, this means waiting until the user has clicked on a button in this dialog. If any widget has its 'notify' flag set (`opt (notify)` in YCP, `setNotify( true )` in C++), an action on such a widget will also make [waitForEvent\(\)](#) return.

If the specified timeout elapses without any user event, a [YTimeoutEvent](#) will be returned. 0 means no timeout (wait forever).

If [open\(\)](#) has not been called for this dialog until now, it is called now.

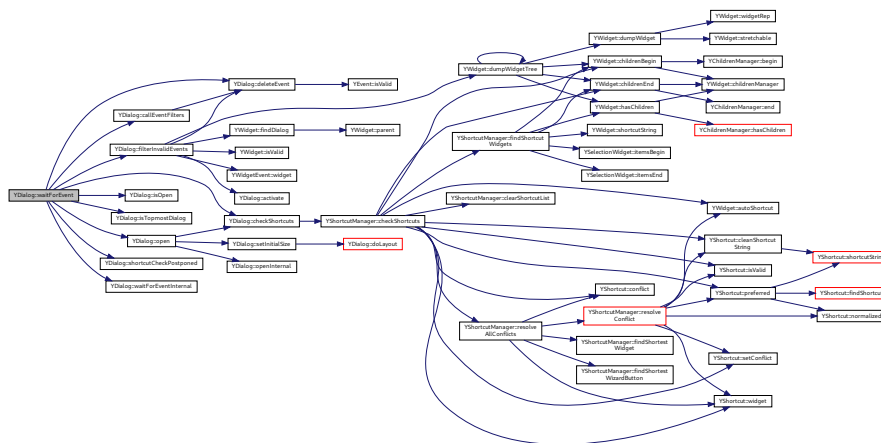
The dialog retains ownership of the event and will delete it upon the next call to [waitForEvent\(\)](#) or [pollEvent\(\)](#) or when the dialog is deleted. This also means that the return value of this function can safely be ignored without fear of memory leaks.

Applications can create [YEventFilters](#) to act upon some events before they are delivered to the application. Each event filter of this dialog is called (in undefined order) in [waitForEvent\(\)](#). An event filter can consume an event (in which case [waitForEvent\(\)](#) will return to its internal event loop), pass it through unchanged, or even replace it with a new event. Refer to the [YEventFilter](#) documentation for more details.

If this dialog is not the toplevel dialog, an exception is thrown.

Definition at line 387 of file [YDialog.cc](#).

Here is the call graph for this function:



### 5.41.3.23 waitForEventInternal()

```
virtual YEvent* YDialog::waitForEventInternal (
    int timeout_millisec ) [protected], [pure virtual]
```

Wait for a user event.

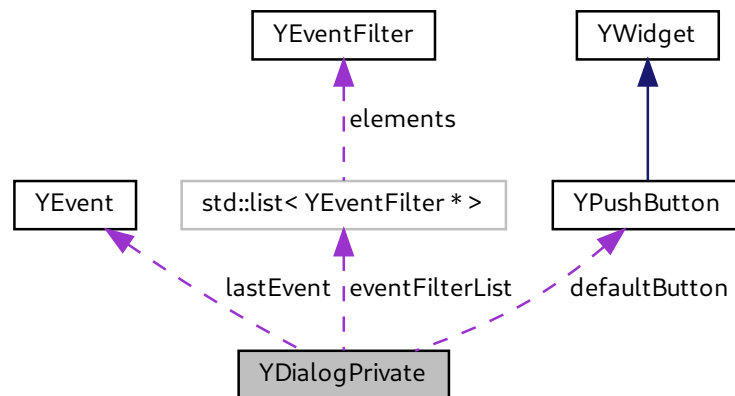
Derived classes are required to implement this.

The documentation for this class was generated from the following files:

- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YDialog.h
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YDialog.cc
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YDialogHelpers.cc
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YUI.cc

## 5.42 YDialogPrivate Struct Reference

Collaboration diagram for YDialogPrivate:



### Public Member Functions

- **YDialogPrivate** ([YDialogType](#) dialogType, [YDialogColorMode](#) colorMode)

## Public Attributes

- [YDialogType](#) **dialogType**
- YDialogColorMode **colorMode**
- bool **shortcutCheckPostponed**
- [YPushButton](#) \* **defaultButton**
- bool **isOpen**
- bool **multiPassLayout**
- int **layoutPass**
- [YEvent](#) \* **lastEvent**
- YEventFilterList **eventFilterList**

### 5.42.1 Detailed Description

Definition at line 45 of file [YDialog.cc](#).

The documentation for this struct was generated from the following file:

- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YDialog.cc

## 5.43 YDialogSpy Class Reference

An interactive dialog debugger: Show the structure and content of a dialog and its widgets.

```
#include <YDialogSpy.h>
```

## Public Member Functions

- void [showProperties](#) ()  
*Show the "Properties" sub-window.*
- void [hideProperties](#) ()  
*Hide the "Properties" sub-window.*
- bool [propertiesShown](#) () const  
*Return 'true' if the "Properties" sub-window is currently shown, 'false' if not.*

## Static Public Member Functions

- static void [showDialogSpy](#) (YDialog \*dialog=0)  
*Show a [YDialogSpy](#) for the specified dialog.*

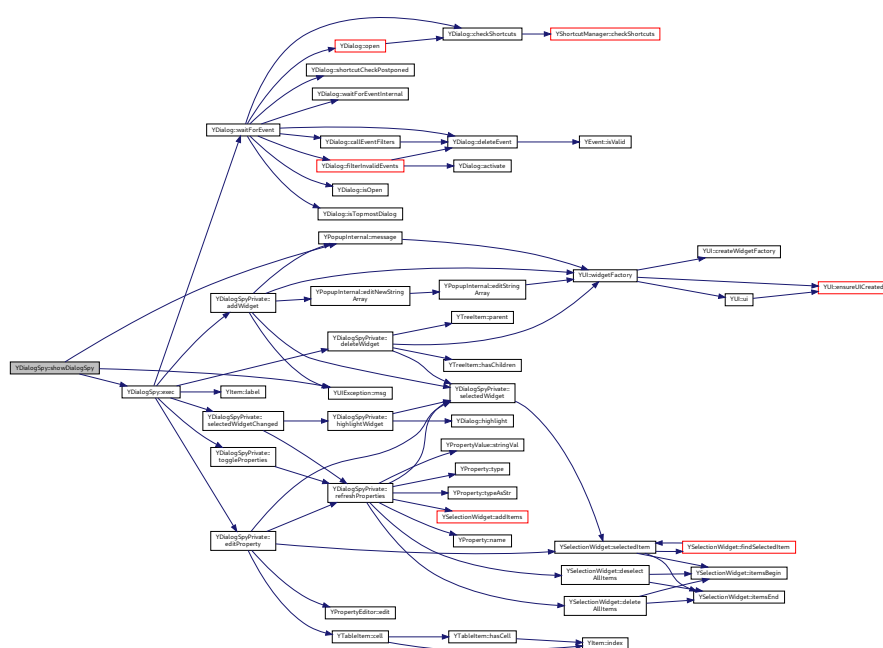






<i>dialog</i>	UI dialog to examine
---------------	----------------------

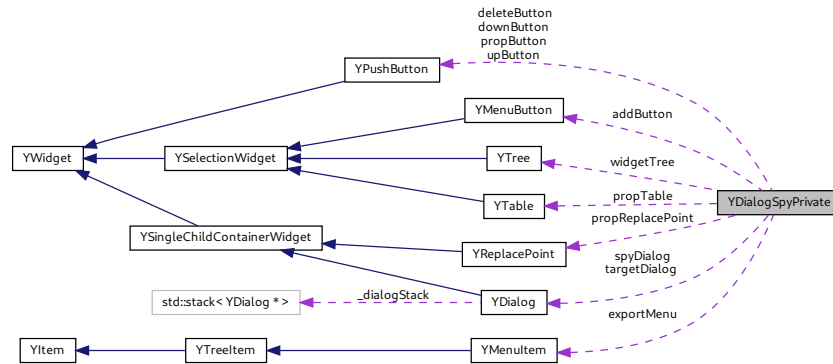
Here is the call graph for this function:



- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YDialogSpy.h
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YDialogSpy.cc

## 5.44 YDialogSpyPrivate Class Reference

Collaboration diagram for YDialogSpyPrivate:



### Public Member Functions

- `~YDialogSpyPrivate ()`  
Destructor - switch off widget highlighting at the end.
- `YWidget * selectedWidget ()`  
The currently selected widget.
- `void selectedWidgetChanged ()`  
The selected item has been changed, refresh the UI.
- `void refreshProperties ()`  
Refresh the displayed properties.
- `bool toggleProperties ()`  
Hide or show the properties dialog.
- `void highlightWidget (bool enable=true)`  
Highlight the currently selected widget in the spy dialog.
- `void deleteWidget ()`  
Delete the currently selected widget.
- `void addWidget (const string &type)`  
Generic handler for adding widgets.
- `void editProperty ()`  
Run the property editor for the current widget.
- `void moveSelectedUp ()`
- `void moveSelectedDown ()`

- `YDialog` \* **targetDialog**
- `YDialog` \* **spyDialog**
- `YTree` \* **widgetTree**
- `YPushButton` \* **propButton**
- `YMenuButton` \* **addButton**
- `YPushButton` \* **deleteButton**
- `YPushButton` \* **upButton**
- `YPushButton` \* **downButton**
- `YReplacePoint` \* **propReplacePoint**
- `YTable` \* **propTable**
- `YMenuItem` \* **exportMenu**

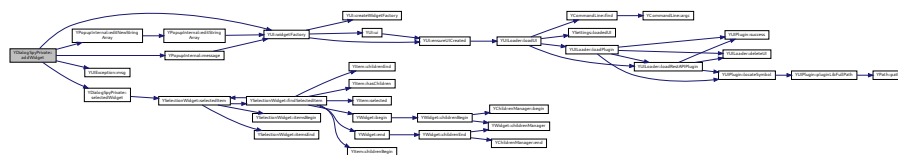
Definition at line 134 of file YDialogSpy.cc.

### 5.44.2.1 addWidget()

Generic handler for adding widgets.

<i>type</i>	Type of the widget to add
-------------	---------------------------

Here is the call graph for this function:



### 5.44.2.2 selectedWidget()

```
YWidget * YDialogSpyPrivate::selectedWidget ( )
```

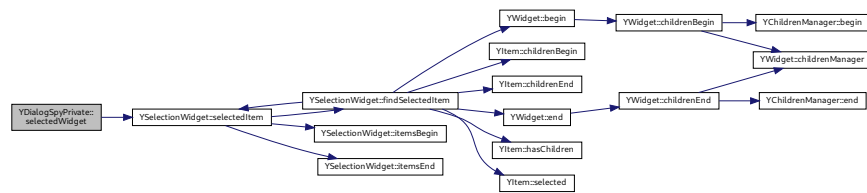
The currently selected widget.

#### Returns

The currently selected widget (or nullptr if nothing is selected)

Definition at line 570 of file [YDialogSpy.cc](#).

Here is the call graph for this function:



### 5.44.2.3 toggleProperties()

```
bool YDialogSpyPrivate::toggleProperties ( )
```

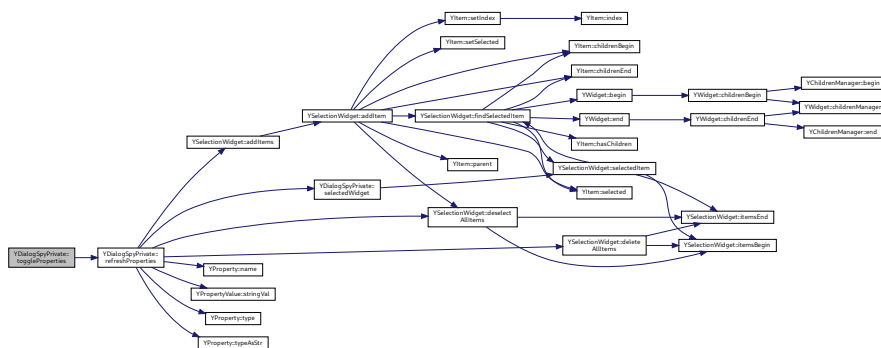
Hide or show the properties dialog.

#### Returns

true if the dialog is now displayed

Definition at line 408 of file [YDialogSpy.cc](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

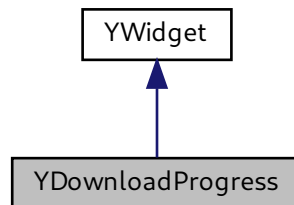
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YDialogSpy.cc`

## 5.45 YDownloadProgress Class Reference

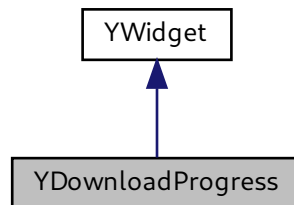
DownloadProgress: A progress bar that monitors downloading a file by repeatedly polling its size up to its expected size.

```
#include <YDownloadProgress.h>
```

Inheritance diagram for YDownloadProgress:



Collaboration diagram for YDownloadProgress:



### Public Member Functions

- virtual `~YDownloadProgress()`  
*Destructor.*
- virtual const char \* `widgetClass()` const  
*Returns a descriptive name of this widget class for logging, debugging etc.*
- std::string `label()` const  
*Get the label (the text above the progress bar).*
- virtual void `setLabel(const std::string &label)`  
*Set the label (the text above the progress bar).*

- `std::string filename () const`  
*Return the name of the file that is being monitored.*
- `virtual void setFilename (const std::string &filename)`  
*Set the name of a new file to monitor.*
- `YFileSize_t expectedSize () const`  
*Return the expected file size.*
- `virtual void setExpectedSize (YFileSize_t newSize)`  
*Set the expected file size.*
- `virtual YFileSize_t currentFileSize () const`  
*Return the current size of the file that is being downloaded or 0 if this file doesn't exist (yet).*
- `int currentPercent () const`  
*Return the percentage (0..100) of the file being downloaded so far.*
- `int value () const`  
*Alias for `currentPercent()`.*
- `virtual bool setProperty (const std::string &propertyName, const YPropertyValue &val)`  
*Set a property.*
- `virtual YPropertyValue getProperty (const std::string &propertyName)`  
*Get a property.*
- `virtual const YPropertySet & propertySet ()`  
*Return this class's property set.*

## Protected Member Functions

- `YDownloadProgress (YWidget *parent, const std::string &label, const std::string &filename, YFileSize_t expectedSize)`  
*Constructor.*

### 5.45.1 Detailed Description

DownloadProgress: A progress bar that monitors downloading a file by repeatedly polling its size up to its expected size.

Definition at line 37 of file `YDownloadProgress.h`.

### 5.45.2 Constructor & Destructor Documentation



### 5.45.2.1 YDownloadProgress()

```
YDownloadProgress::YDownloadProgress (
    YWidget * parent,
    const std::string & label,
    const std::string & filename,
    YFileSize_t expectedSize ) [protected]
```

Constructor.

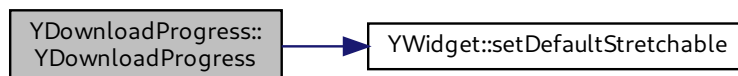
'label' is the label above the progress bar.

'filename' is the name (with path) of the file being monitored.

'expectedSize' is the expected size of the file in bytes.

Definition at line 53 of file [YDownloadProgress.cc](#).

Here is the call graph for this function:



## 5.45.3 Member Function Documentation

### 5.45.3.1 currentFileSize()

```
YFileSize_t YDownloadProgress::currentFileSize ( ) const [virtual]
```

Return the current size of the file that is being downloaded or 0 if this file doesn't exist (yet).

This default implementation returns the 'st\_size' field of a stat() system call on the file. This should be useful for most implementations.

Definition at line 131 of file [YDownloadProgress.cc](#).

### 5.45.3.2 getProperty()

```
YPropertyValue YDownloadProgress::getProperty (
    const std::string & propertyName ) [virtual]
```

Get a property.

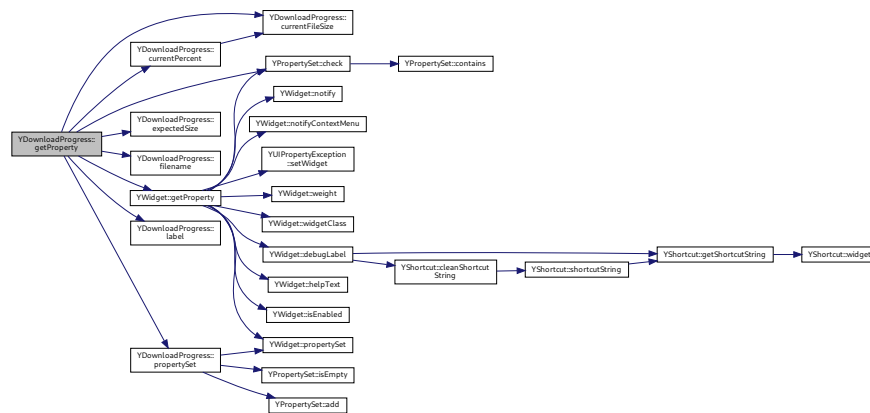
Reimplemented from [YWidget](#).

This method may throw [YUIPropertyExceptions](#).

Reimplemented from [YWidget](#).

Definition at line 186 of file [YDownloadProgress.cc](#).

Here is the call graph for this function:



### 5.45.3.3 propertySet()

```
const YPropertySet & YDownloadProgress::propertySet ( ) [virtual]
```

Return this class's property set.

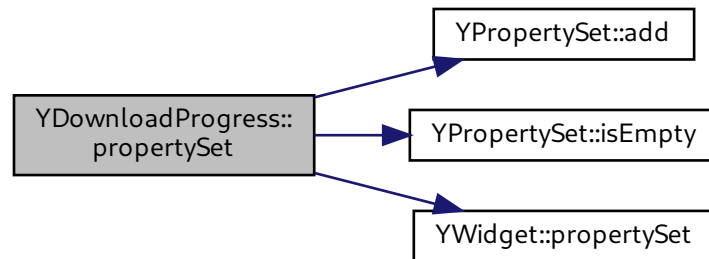
This also initializes the property upon the first call.

Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 143 of file [YDownloadProgress.cc](#).

Here is the call graph for this function:



#### 5.45.3.4 setExpectedSize()

```
void YDownloadProgress::setExpectedSize (
    YFileSize_t newSize ) [virtual]
```

Set the expected file size.

Derived classes are free to reimplement this, but they should call this base class method at the end of the overloaded function.

Definition at line 109 of file [YDownloadProgress.cc](#).

#### 5.45.3.5 setFilename()

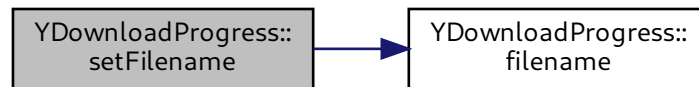
```
void YDownloadProgress::setFilename (
    const std::string & filename ) [virtual]
```

Set the name of a new file to monitor.

Derived classes are free to reimplement this, but they should call this base class method at the end of the overloaded function.

Definition at line 95 of file [YDownloadProgress.cc](#).

Here is the call graph for this function:



#### 5.45.3.6 `setLabel()`

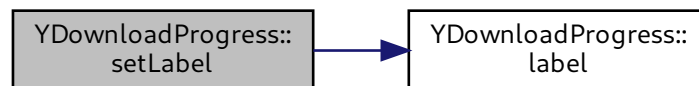
```
void YDownloadProgress::setLabel (
    const std::string & label ) [virtual]
```

Set the label (the text above the progress bar).

Derived classes are free to reimplement this, but they should call this base class method at the end of the overloaded function.

Definition at line 81 of file [YDownloadProgress.cc](#).

Here is the call graph for this function:



### 5.45.3.7 setProperty()

```
bool YDownloadProgress::setProperty (
    const std::string & propertyName,
    const YPropertyValue & val ) [virtual]
```

Set a property.

Reimplemented from [YWidget](#).

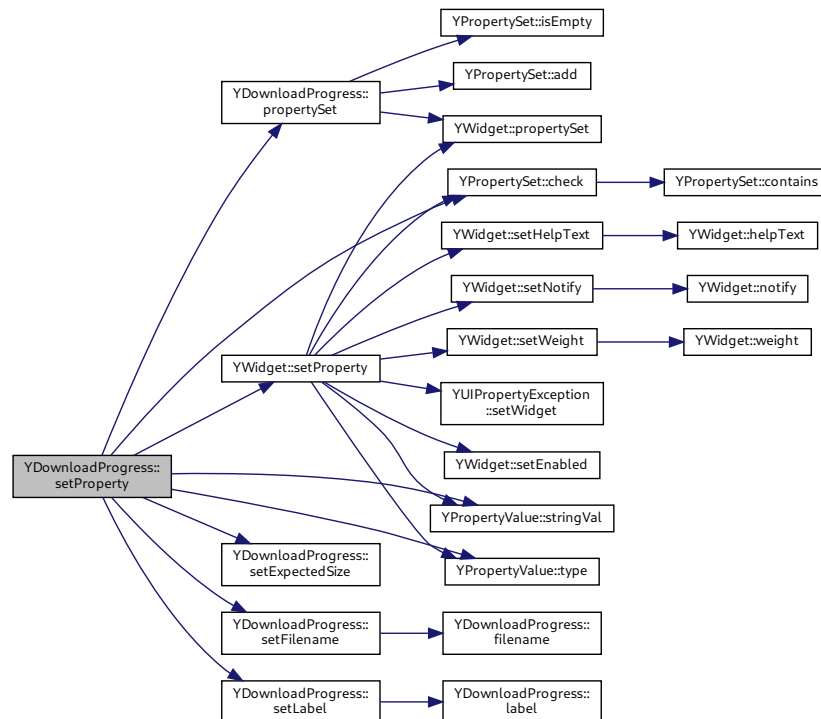
This function may throw YUIPropertyExceptions.

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from [YWidget](#).

Definition at line 169 of file [YDownloadProgress.cc](#).

Here is the call graph for this function:

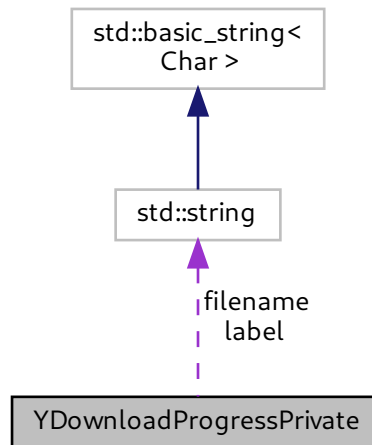


The documentation for this class was generated from the following files:

- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YDownloadProgress.h`
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YDownloadProgress.cc`

## 5.46 YDownloadProgressPrivate Struct Reference

Collaboration diagram for YDownloadProgressPrivate:



### Public Member Functions

- **YDownloadProgressPrivate** (const string &label, const string &filename, YFileSize\_t expectedSize)

### Public Attributes

- string **label**
- string **filename**
- YFileSize\_t **expectedSize**

#### 5.46.1 Detailed Description

Definition at line 37 of file [YDownloadProgress.cc](#).

The documentation for this struct was generated from the following file:

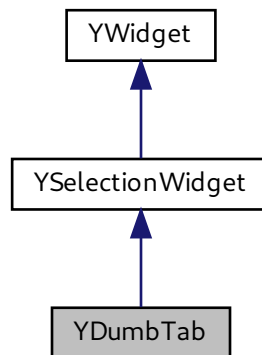
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YDownloadProgress.cc

## 5.47 YDumbTab Class Reference

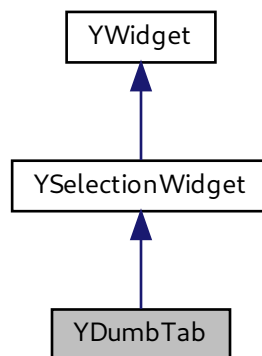
DumbTab: A very simple tab widget that can display and switch between a number of tabs, but will only deliver the "user clicked on tab " event very much like a PushButton does.

```
#include <YDumbTab.h>
```

Inheritance diagram for YDumbTab:



Collaboration diagram for YDumbTab:



## Public Member Functions

- virtual [~YDumbTab](#) ()  
*Destructor.*
- virtual const char \* [widgetClass](#) () const  
*Returns a descriptive name of this widget class for logging, debugging etc.*
- virtual void [addItem](#) (YItem \*item)  
*Add an item (a tab page).*
- virtual void [shortcutChanged](#) ()  
*Notification that any shortcut of any item was changed by the shortcut conflict manager.*
- virtual bool [setProperty](#) (const std::string &propertyName, const YPropertyValue &val)  
*Set a property.*
- virtual YPropertyValue [getProperty](#) (const std::string &propertyName)  
*Get a property.*
- virtual const YPropertySet & [propertySet](#) ()  
*Return this class's property set.*
- virtual std::string [shortcutString](#) () const  
*Get the string of this widget that holds the keyboard shortcut.*
- virtual void [setShortcutString](#) (const std::string &str)  
*Set the string of this widget that holds the keyboard shortcut.*
- virtual bool [stretchable](#) (YUIDimension dim) const  
*Returns 'true' if this widget is stretchable in the specified dimension.*
- virtual std::string [debugLabel](#) () const  
*Descriptive label for debugging.*
- virtual void [activate](#) ()=0  
*Activate selected tab.*

## Protected Member Functions

- [YDumbTab](#) (YWidget \*parent)  
*Constructor.*

### 5.47.1 Detailed Description

DumbTab: A very simple tab widget that can display and switch between a number of tabs, but will only deliver the "user clicked on tab " event very much like a PushButton does.

Actually exchanging the content of the tab is left to the application.

DumbTab accepts a single child widget.

Definition at line 40 of file [YDumbTab.h](#).

### 5.47.2 Member Function Documentation



### 5.47.2.1 activate()

```
virtual void YDumbTab::activate ( ) [pure virtual]
```

Activate selected tab.

Can be used in tests to simulate user input.

Derived classes are required to implement this.

### 5.47.2.2 addItem()

```
void YDumbTab::addItem (
    YItem * item ) [virtual]
```

Add an item (a tab page).

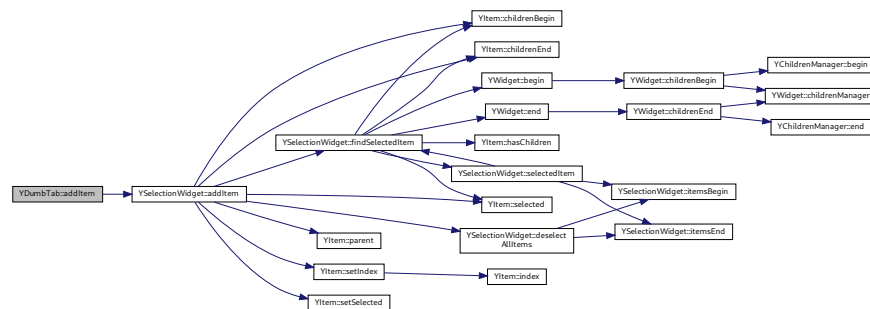
Reimplemented from [YSelectionWidget](#).

Derived classes can overwrite this function, but they should call this base class function in the new implementation.

Reimplemented from [YSelectionWidget](#).

Definition at line 68 of file [YDumbTab.cc](#).

Here is the call graph for this function:



### 5.47.2.3 debugLabel()

```
string YDumbTab::debugLabel ( ) const [virtual]
```

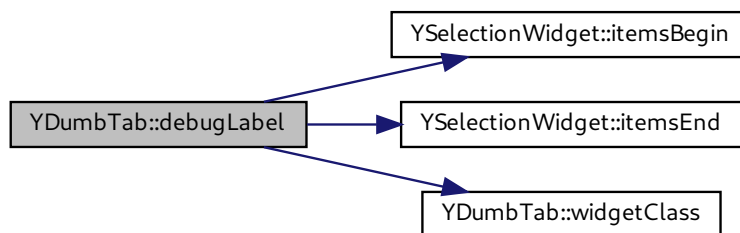
Descriptive label for debugging.

Derived from this widget's only child (if there is one).

Reimplemented from [YWidget](#).

Definition at line 85 of file [YDumbTab.cc](#).

Here is the call graph for this function:



### 5.47.2.4 getProperty()

```
YPropertyValue YDumbTab::getProperty (
    const std::string & propertyName ) [virtual]
```

Get a property.

Reimplemented from [YWidget](#).

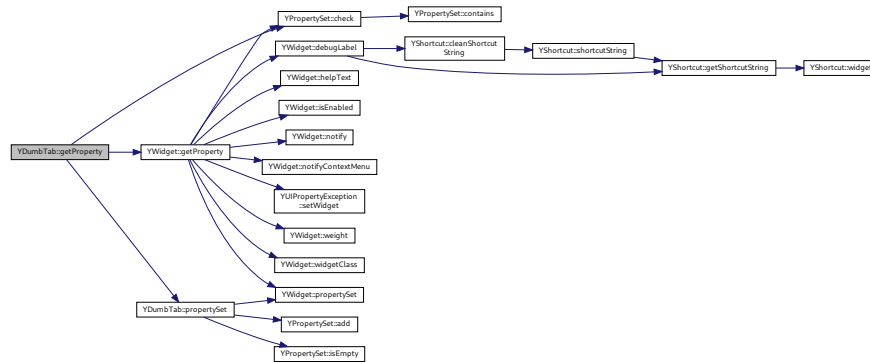
This method may throw exceptions, for example

- if there is no property with that name

Reimplemented from [YWidget](#).

Definition at line 141 of file [YDumbTab.cc](#).

Here is the call graph for this function:



### 5.47.2.5 propertySet()

```
const YPropertySet & YDumbTab::propertySet ( ) [virtual]
```

Return this class's property set.

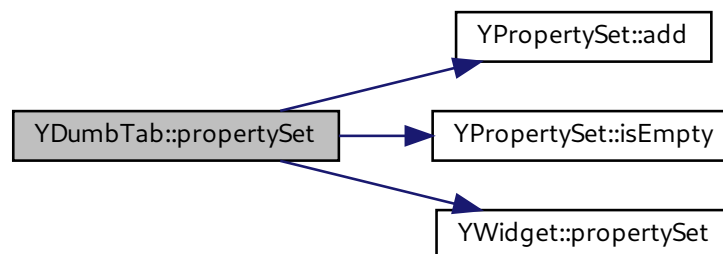
This also initializes the property set upon the first call.

Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 102 of file [YDumbTab.cc](#).

Here is the call graph for this function:



### 5.47.2.6 setProperty()

```
bool YDumbTab::setProperty (
    const std::string & propertyName,
    const YPropertyValue & val ) [virtual]
```

Set a property.

Reimplemented from [YWidget](#).

This method may throw exceptions, for example

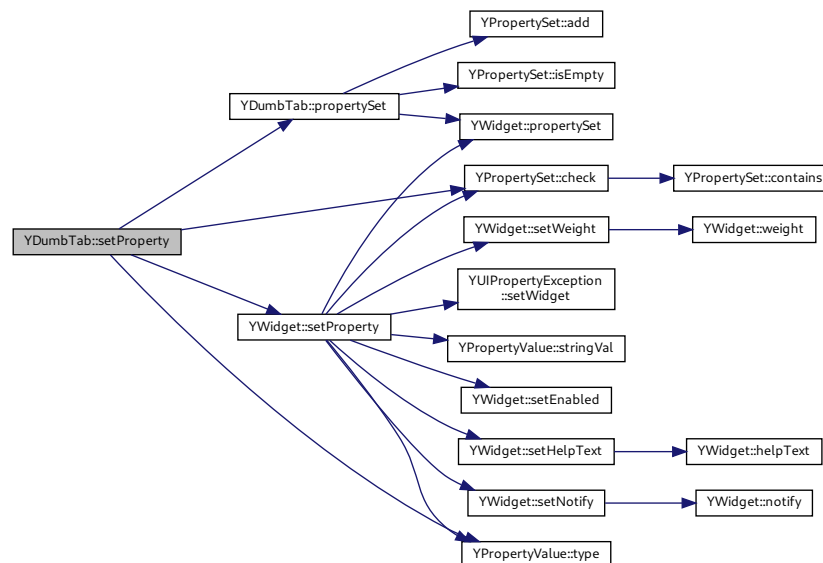
- if there is no property with that name
- if the expected type and the type mismatch
- if the value is out of range

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from [YWidget](#).

Definition at line 124 of file [YDumbTab.cc](#).

Here is the call graph for this function:



### 5.47.2.7 setShortcutString()

```
virtual void YDumbTab::setShortcutString (
    const std::string & str ) [inline], [virtual]
```

Set the string of this widget that holds the keyboard shortcut.

Since [YDumbTab](#) doesn't have a shortcut for the widget itself (only for the tab pages, i.e. the items), this will simply trigger a [shortcutChanged\(\)](#) notification.

Reimplemented from [YSelectionWidget](#).

Reimplemented from [YSelectionWidget](#).

Definition at line 130 of file [YDumbTab.h](#).

Here is the call graph for this function:



### 5.47.2.8 shortcutChanged()

```
virtual void YDumbTab::shortcutChanged ( ) [inline], [virtual]
```

Notification that any shortcut of any item was changed by the shortcut conflict manager.

Derived classes should reimplement this.

Definition at line 76 of file [YDumbTab.h](#).

### 5.47.2.9 shortcutString()

```
virtual std::string YDumbTab::shortcutString ( ) const [inline], [virtual]
```

Get the string of this widget that holds the keyboard shortcut.

Notice that since [YDumbTab](#) has one shortcut for each tab page (for each item), this is not meaningful for this widget class.

Check [YItemShortcut](#) in [YShortcut.{cc,h}](#) for more details.

Reimplemented from [YSelectionWidget](#).

Reimplemented from [YSelectionWidget](#).

Definition at line 120 of file [YDumbTab.h](#).

### 5.47.2.10 stretchable()

```
bool YDumbTab::stretchable (
    YUIDimension dim ) const [virtual]
```

Returns 'true' if this widget is stretchable in the specified dimension.

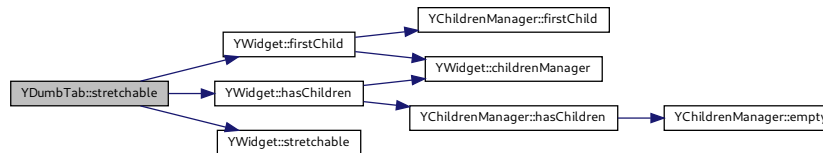
In this case, the stretchability of the single child is returned.

Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 75 of file [YDumbTab.cc](#).

Here is the call graph for this function:



The documentation for this class was generated from the following files:

- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YDumbTab.h`
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YDumbTab.cc`

## 5.48 YDumbTabPrivate Struct Reference

### Public Attributes

- `bool dummy`

### 5.48.1 Detailed Description

Definition at line 36 of file [YDumbTab.cc](#).

The documentation for this struct was generated from the following file:

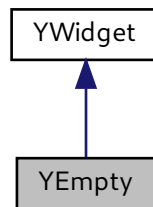
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YDumbTab.cc`

## 5.49 YEmpty Class Reference

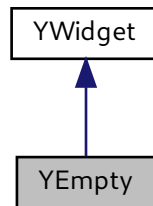
A widget with zero size, useful as a placeholder.

```
#include <YEmpty.h>
```

Inheritance diagram for YEmpty:



Collaboration diagram for YEmpty:



### Public Member Functions

- virtual `~YEmpty()`  
*Destructor.*
- virtual const char \* `widgetClass()` const  
*Returns a descriptive name of this widget class for logging, debugging etc.*
- virtual int `preferredWidth()`  
*Preferred width of the widget.*
- virtual int `preferredHeight()`  
*Preferred height of the widget.*

## Protected Member Functions

- [YEmpty](#) ([YWidget](#) \*parent)  
*Constructor.*

### 5.49.1 Detailed Description

A widget with zero size, useful as a placeholder.

Definition at line 37 of file [YEmpty.h](#).

### 5.49.2 Member Function Documentation

#### 5.49.2.1 preferredHeight()

```
int YEmpty::preferredHeight ( ) [virtual]
```

Preferred height of the widget.

Reimplemented from [YWidget](#).

Implements [YWidget](#).

Definition at line 56 of file [YEmpty.cc](#).

#### 5.49.2.2 preferredWidth()

```
int YEmpty::preferredWidth ( ) [virtual]
```

Preferred width of the widget.

Reimplemented from [YWidget](#).

Implements [YWidget](#).

Definition at line 50 of file [YEmpty.cc](#).

The documentation for this class was generated from the following files:

- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YEmpty.h
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YEmpty.cc



## 5.50 YEmptyPrivate Struct Reference

### Public Attributes

- bool **dummy**

### 5.50.1 Detailed Description

Definition at line 28 of file [YEmpty.cc](#).

The documentation for this struct was generated from the following file:

- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YEmpty.cc

## 5.51 YEnvVar Class Reference

Helper class to represent an environment variable and its value.

```
#include <YEnvVar.h>
```

### Public Member Functions

- [YEnvVar](#) (const std::string &name=std::string())  
*Constructor: Retrieve the environment variable 'name' and store the value (unless 'name' is empty).*
- std::string [name](#) () const  
*Return the name of the environment variable.*
- bool [isSet](#) () const  
*Return 'true' if the environment variable is set.*
- std::string [value](#) () const  
*Return the value of the environment variable.*
- bool [isEqual](#) (const std::string &str, bool caseSensitive=false) const  
*Return 'true' if the environment variable is set and the value is 'str'.*
- bool [operator==](#) (const std::string &str) const  
*Case-insensitive comparison (shortcut for [isEqual\(\)](#)): Return 'true' if the environment variable is set and the value is 'str'.*
- bool [contains](#) (const std::string &str, bool caseSensitive=false) const  
*Return 'true' if the environment variable is set and the value contains 'str'.*

### 5.51.1 Detailed Description

Helper class to represent an environment variable and its value.

Definition at line 36 of file [YEnvVar.h](#).

The documentation for this class was generated from the following files:

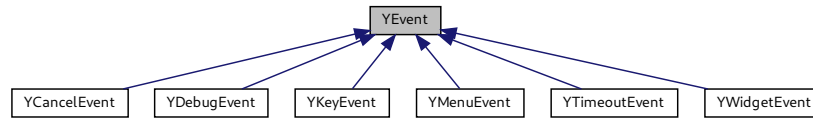
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YEnvVar.h
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YEnvVar.cc

## 5.52 YEvent Class Reference

Abstract base class for events to be returned upon UI::UserInput() and related functions.

```
#include <YEvent.h>
```

Inheritance diagram for YEvent:



### Public Types

- enum **EventType** {  
**NoEvent** = 0 , **UnknownEvent** , **WidgetEvent** , **MenuEvent** ,  
**KeyEvent** , **CancelEvent** , **TimeoutEvent** , **DebugEvent** ,  
**InvalidEvent** = 0x4242 }
- enum **EventReason** {  
**UnknownReason** = 0 , **Activated** , **SelectionChanged** , **ValueChanged** ,  
**ContextMenuActivated** }

### Public Member Functions

- **YEvent** (EventType **eventType**=UnknownEvent)  
*Constructor.*
- EventType **eventType** () const  
*Returns the event type.*
- unsigned long **serial** () const  
*Returns the unique serial no.*
- virtual **YWidget** \* **widget** () const  
*Returns the widget that caused this event or 0 if there is none.*
- virtual **YItem** \* **item** () const  
*Return the YItem that corresponds to this event or 0 if there is none.*
- **YDialog** \* **dialog** () const  
*Return the dialog this event belongs to or 0 if no dialog was set yet.*
- bool **isValid** () const  
*Check if this event is valid.*

### Static Public Member Functions

- static const char \* **toString** (EventType **eventType**)  
*Returns the character representation of an event type.*
- static const char \* **toString** (EventReason **reason**)  
*Returns the character representation of an event reason.*

## Protected Member Functions

- void [setDialog](#) ([YDialog](#) \*dia)  
*Set the dialog this event belongs to.*
- virtual [~YEvent](#) ()  
*Protected destructor - events can only be deleted via [YDialog::deleteEvent\(\)](#).*
- void [invalidate](#) ()  
*Mark this event as invalid.*

## Friends

- void [YDialog::deleteEvent](#) ([YEvent](#) \*event)
- void [YSimpleEventHandler::deleteEvent](#) ([YEvent](#) \*event)

### 5.52.1 Detailed Description

Abstract base class for events to be returned upon `UI::UserInput()` and related functions.

Definition at line 43 of file [YEvent.h](#).

### 5.52.2 Constructor & Destructor Documentation

#### 5.52.2.1 [~YEvent\(\)](#)

```
YEvent::~YEvent ( ) [protected], [virtual]
```

Protected destructor - events can only be deleted via [YDialog::deleteEvent\(\)](#).

The associated dialog will take care of this event and delete it when appropriate.

This desctructor is virtual to force a polymorph object so `dynamic_cast<>` can be used.

Definition at line 48 of file [YEvent.cc](#).

Here is the call graph for this function:



### 5.52.3 Member Function Documentation

#### 5.52.3.1 invalidate()

```
void YEvent::invalidate ( ) [protected]
```

Mark this event as invalid.

This cannot be undone.

Definition at line 62 of file [YEvent.cc](#).

#### 5.52.3.2 isValid()

```
bool YEvent::isValid ( ) const
```

Check if this event is valid.

Events become invalid in the destructor.

Definition at line 55 of file [YEvent.cc](#).

#### 5.52.3.3 item()

```
virtual YItem* YEvent::item ( ) const [inline], [virtual]
```

Return the [YItem](#) that corresponds to this event or 0 if there is none.

This default implementation always returns 0. Subclasses that actually return items should overwrite this method.

Reimplemented in [YMenuEvent](#).

Definition at line 101 of file [YEvent.h](#).

#### 5.52.3.4 serial()

```
unsigned long YEvent::serial ( ) const [inline]
```

Returns the unique serial no.

of this event. This is mainly useful for debugging.

Definition at line 85 of file [YEvent.h](#).

#### 5.52.3.5 widget()

```
virtual YWidget* YEvent::widget ( ) const [inline], [virtual]
```

Returns the widget that caused this event or 0 if there is none.

This default implementation always returns 0. Subclasses that actually return widgets should overwrite this method.

Reimplemented in [YWidgetEvent](#).

Definition at line 93 of file [YEvent.h](#).

The documentation for this class was generated from the following files:

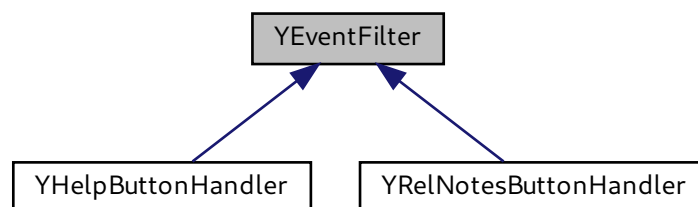
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YEvent.h](#)
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YEvent.cc](#)

## 5.53 YEventFilter Class Reference

Abstract base class to filter events.

```
#include <YEventFilter.h>
```

Inheritance diagram for YEventFilter:



## Public Member Functions

- virtual [~YEventFilter](#) ()  
*Destructor.*
- virtual [YEvent](#) \* [filter](#) ([YEvent](#) \*event)=0  
*The heart of the matter: The event filter function.*
- [YDialog](#) \* [dialog](#) () const  
*Return the dialog this event filter belongs to.*

## Protected Member Functions

- [YEventFilter](#) ([YDialog](#) \*dialog=0)  
*Constructor.*

### 5.53.1 Detailed Description

Abstract base class to filter events.

This class can be used to examine events just before they are delivered to the application. This is most useful for higher-level widgets or for libraries that need to react to certain events and either consume them, have them delivered unchanged to the application, or exchange an event with another one.

A [YEventFilter](#) belongs to one specific dialog. Each dialog can have any number of event filters. Each of those event filters is called (its [YEventFilter::filter\(\)](#) method) for each event inside [YDialog::waitForEvent\(\)](#). The order in which event filters are called is undefined.

[YEventFilter](#) objects should be created with 'new' (on the heap). Since an [YEventFilter](#) registers itself with its dialog, the dialog will delete it in its destructor if it still exists after all child widgets are deleted.

Thus, it is safe to store a pointer to an [YEventFilter](#) until the corresponding dialog is deleted. After that, the pointer becomes invalid.

See [YHelpButtonHandler](#) in [YDialog.cc](#) for an example.

Definition at line 62 of file [YEventFilter.h](#).

### 5.53.2 Constructor & Destructor Documentation

### 5.53.2.1 YEventFilter()

```
YEventFilter::YEventFilter (
    YDialog * dialog = 0 ) [protected]
```

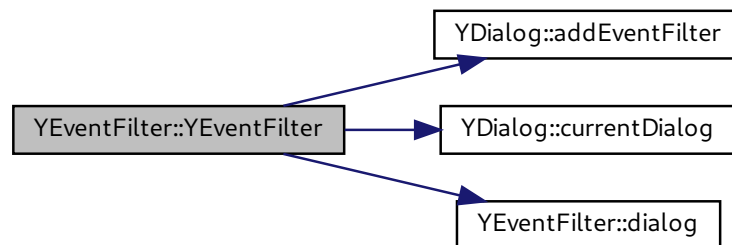
Constructor.

This registers the event filter with the specified dialog. The dialog assumes ownership of this object and will delete it in its destructor (unless this object is destroyed before that time).

If 'dialog' is 0, [YDialog::currentDialog\(\)](#) is used (which can throw a [YUINoDialogException](#) if there is no dialog).

Definition at line 44 of file [YEventFilter.cc](#).

Here is the call graph for this function:



### 5.53.2.2 ~YEventFilter()

```
YEventFilter::~YEventFilter ( ) [virtual]
```

Destructor.

This will unregister this object with its dialog.

Definition at line 56 of file [YEventFilter.cc](#).

Here is the call graph for this function:



### 5.53.3 Member Function Documentation

#### 5.53.3.1 filter()

```
virtual YEvent* YEventFilter::filter (
    YEvent * event ) [pure virtual]
```

The heart of the matter: The event filter function.

Derived classes are required to implement this.

This method can inspect the event it receives. Hint: `event->widget()` is typically the most interesting information.

This method can react on individual events and

- consume the event (i.e., return 0)
- pass the event through unchanged (simply return the event)
- create a new event (typically based on data in the received event).

If 0 or a new event (another value than 'event') is returned, the old event is deleted. If a value different from 'event' or 0 is returned, that value is assumed to be a pointer to a newly created event. The dialog will assume ownership of that event and delete it when appropriate.

Note: Never delete 'event' in this method! Return 0 or a new event instead; the caller will take care of deleting the old event.

Implemented in [YRelNotesButtonHandler](#), and [YHelpButtonHandler](#).

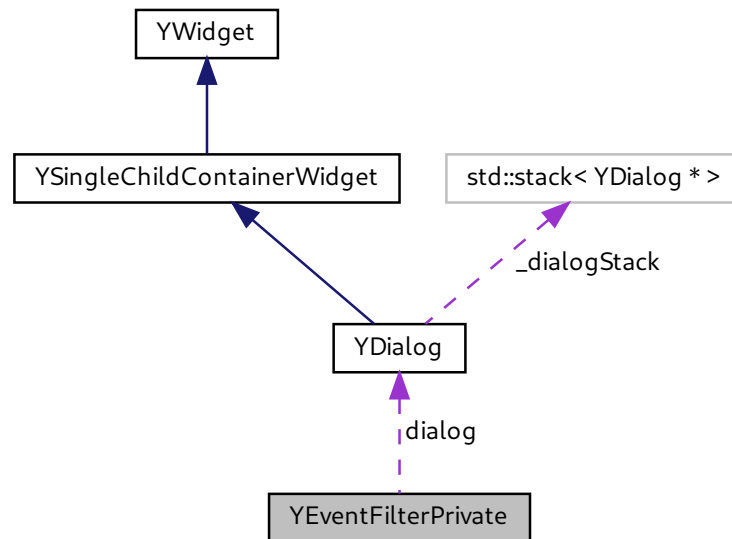
The documentation for this class was generated from the following files:

- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YEventFilter.h`
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YEventFilter.cc`



## 5.54 YEventFilterPrivate Struct Reference

Collaboration diagram for YEventFilterPrivate:



### Public Member Functions

- **YEventFilterPrivate** ([YDialog](#) \*dialog)

### Public Attributes

- [YDialog](#) \* dialog

#### 5.54.1 Detailed Description

Definition at line 32 of file [YEventFilter.cc](#).

The documentation for this struct was generated from the following file:

- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YEventFilter.cc

## 5.55 YExternalWidgetFactory Class Reference

Abstract widget factory for mandatory widgets.

```
#include <YExternalWidgetFactory.h>
```

### Protected Member Functions

- [YExternalWidgetFactory](#) ()  
*Constructor.*
- virtual [~YExternalWidgetFactory](#) ()  
*Destructor.*

### Friends

- class **YUI**
- class **YExternalWidgets**

### 5.55.1 Detailed Description

Abstract widget factory for mandatory widgets.

Use [YOptionalWidgetFactory](#) for optional ("special") widgets. [YExternalWidgetFactory](#) is used for external widgets, e.g. user defined plugin.

Refer to the respective widget's documentation (in the header file) for documentation about the function parameters.

Definition at line 30 of file [YExternalWidgetFactory.h](#).

### 5.55.2 Constructor & Destructor Documentation

#### 5.55.2.1 YExternalWidgetFactory()

```
YExternalWidgetFactory::YExternalWidgetFactory ( ) [inline], [protected]
```

Constructor.

Use `YExternalWidgets::widgetExtensionFactory()` to get the singleton for this class.

Definition at line 42 of file [YExternalWidgetFactory.h](#).

The documentation for this class was generated from the following file:

- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YExternalWidgetFactory.h`

## 5.56 YExternalWidgets Class Reference

Abstract base class of a libYUI Widget Extension interface.

```
#include <YExternalWidgets.h>
```

### Public Member Functions

- virtual [~YExternalWidgets](#) ()  
*Destructor.*
- [YExternalWidgetFactory](#) \* [externalWidgetFactory](#) ()  
*Return the external widget factory that provides all the createXY() methods for user defined widgets.*

### Static Public Member Functions

- static [YExternalWidgets](#) \* [externalWidgets](#) (const std::string &name)  
*Access the global YUI external widgets.*
- static [YExternalWidgetFactory](#) \* [externalWidgetFactory](#) (const std::string &name)

### Protected Member Functions

- [YExternalWidgets](#) (const std::string &name)  
*Constructor.*
- virtual [YExternalWidgetFactory](#) \* [createExternalWidgetFactory](#) ()=0  
*Create the external widgets factory that provides all the createXY() methods for.*

### Friends

- class [YExternalWidgetsTerminator](#)

#### 5.56.1 Detailed Description

Abstract base class of a libYUI Widget Extension interface.

Definition at line 29 of file [YExternalWidgets.h](#).

#### 5.56.2 Constructor & Destructor Documentation



### 5.56.3.2 externalWidgetFactory()

```
YExternalWidgetFactory * YExternalWidgets::externalWidgetFactory ( )
```

Return the external widget factory that provides all the createXY() methods for user defined widgets.

This will create the factory upon the first call and return a pointer to the one and only (singleton) factory upon each subsequent call. This may throw exceptions if the factory cannot be created.

It is up to user extend `YExternalWidgetFactory` to add `createXY()` methods in his/her implementation. So once `YExternalWidgetFactory` is extended with all the `createXY()` methods, three sub-plugins must be defined one for each supported graphical environment, e.g. Gtk, ncurses and QT, following the libyui implementation rules.

For instance an external widgets plugin called yui-foo that needs Gtk, ncurses and QT specialization will require also yui-foo-gtk, yui-foo-ncurses and yui-foo-qt plugin implementation.

Definition at line 83 of file YExternalWidgets.cc.

Here is the call graph for this function:



### 5.56.3.3 externalWidgets()

```
YExternalWidgets * YExternalWidgets::externalWidgets (
    const std::string & name ) [static]
```

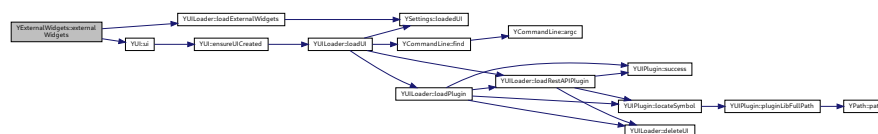
Access the global **YUI** external widgets.

'name' is the plugin name

if plugin 'name' has not been explicitly loaded by `YUILoader::loadExternalWidgets` externalWidgets try loading it (exactly as `YUI::ui` does) with default function symbol to be executed (see `YUILoader::loadExternalWidgets` for explanation)

Definition at line 60 of file YExternalWidgets.cc.

Here is the call graph for this function:



The documentation for this class was generated from the following files:

- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YExternalWidgets.h
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YExternalWidgets.cc

## 5.57 YExternalWidgetsTerminator Class Reference

Helper class to make sure the EW is properly shut down.

### Public Member Functions

- [~YExternalWidgetsTerminator\(\)](#)  
*Destructor.*

### 5.57.1 Detailed Description

Helper class to make sure the EW is properly shut down.

Definition at line 100 of file [YExternalWidgets.cc](#).

### 5.57.2 Constructor & Destructor Documentation

#### 5.57.2.1 ~YExternalWidgetsTerminator()

```
YExternalWidgetsTerminator::~YExternalWidgetsTerminator ( )
```

Destructor.

If there still is a EW, it will be deleted. If there is none, this will do nothing.

Definition at line 115 of file [YExternalWidgets.cc](#).

The documentation for this class was generated from the following file:

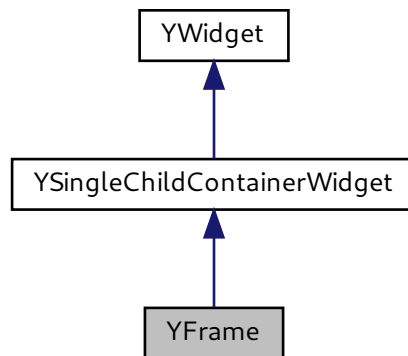
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YExternalWidgets.cc](#)

## 5.58 YFrame Class Reference

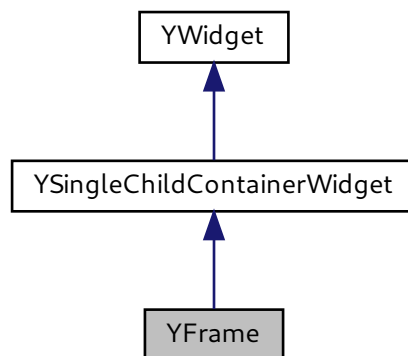
A labeled framed container.

```
#include <YFrame.h>
```

Inheritance diagram for YFrame:



Collaboration diagram for YFrame:



## Public Member Functions

- virtual [~YFrame](#) ()  
*Destructor.*
- virtual const char \* [widgetClass](#) () const  
*Returns a descriptive name of this widget class for logging, debugging etc.*
- virtual void [setLabel](#) (const std::string &newLabel)  
*Change the frame label.*
- std::string [label](#) () const  
*Get the current frame label.*
- virtual bool [setProperty](#) (const std::string &propertyName, const [YPropertyValue](#) &val)  
*Set a property.*
- virtual [YPropertyValue](#) [getProperty](#) (const std::string &propertyName)  
*Get a property.*
- virtual const [YPropertySet](#) & [propertySet](#) ()  
*Return this class's property set.*

## Protected Member Functions

- [YFrame](#) ([YWidget](#) \*parent, const std::string &label)  
*Constructor.*

### 5.58.1 Detailed Description

A labeled framed container.

Definition at line 38 of file [YFrame.h](#).

### 5.58.2 Member Function Documentation

#### 5.58.2.1 [getProperty\(\)](#)

```
YPropertyValue YFrame::getProperty (  
    const std::string & propertyName ) [virtual]
```

Get a property.

Reimplemented from [YWidget](#).

This method may throw exceptions, for example

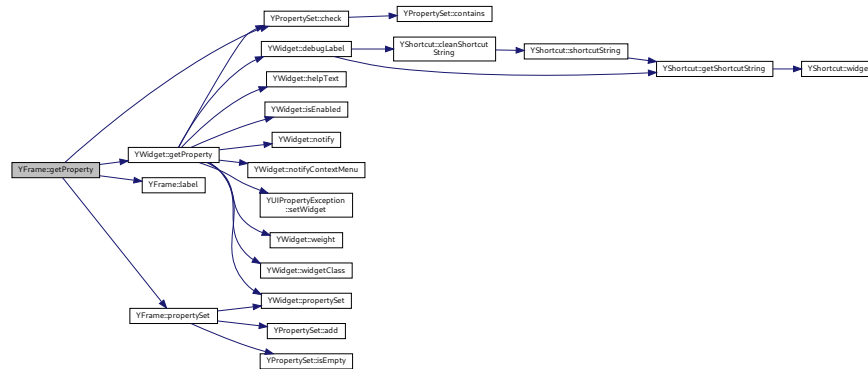
- if there is no property with that name



Reimplemented from [YWidget](#).

Definition at line 109 of file [YFrame.cc](#).

Here is the call graph for this function:



### 5.58.2.2 propertySet()

```
const YPropertySet & YFrame::propertySet ( ) [virtual]
```

Return this class's property set.

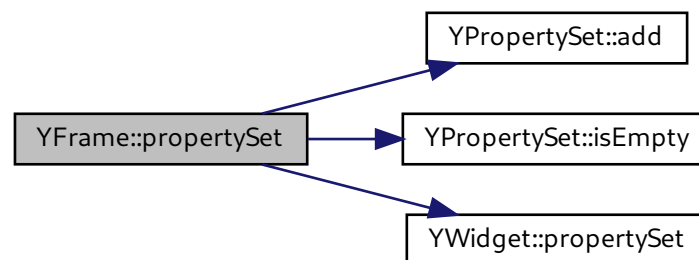
This also initializes the property set upon the first call.

Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 75 of file [YFrame.cc](#).

Here is the call graph for this function:



### 5.58.2.3 setLabel()

```
void YFrame::setLabel (
    const std::string & newLabel ) [virtual]
```

Change the frame label.

Derived classes should overwrite this, but call this base class function in the overwritten function.

Definition at line 62 of file [YFrame.cc](#).

Here is the call graph for this function:



### 5.58.2.4 setProperty()

```
bool YFrame::setProperty (
    const std::string & propertyName,
    const YPropertyValue & val ) [virtual]
```

Set a property.

Reimplemented from [YWidget](#).

This method may throw exceptions, for example

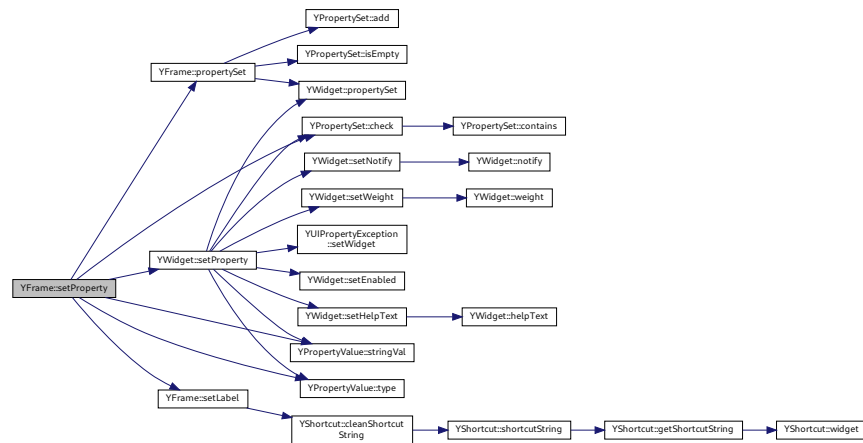
- if there is no property with that name
- if the expected type and the type mismatch
- if the value is out of range

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from [YWidget](#).

Definition at line 94 of file [YFrame.cc](#).

Here is the call graph for this function:

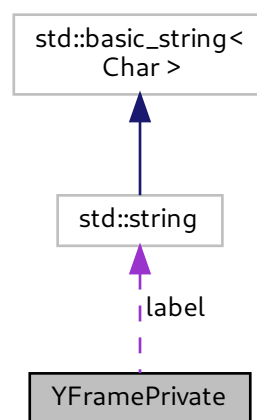


The documentation for this class was generated from the following files:

- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YFrame.h
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YFrame.cc

## 5.59 YFramePrivate Struct Reference

Collaboration diagram for YFramePrivate:



## Public Member Functions

- **YFramePrivate** (const string &frameLabel)

## Public Attributes

- string **label**

### 5.59.1 Detailed Description

Definition at line 36 of file [YFrame.cc](#).

The documentation for this struct was generated from the following file:

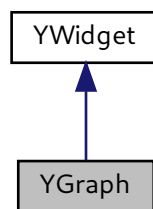
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YFrame.cc

## 5.60 YGraph Class Reference

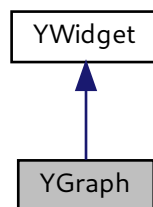
A graph with nodes and edges, rendered with Graphviz.

```
#include <YGraph.h>
```

Inheritance diagram for YGraph:



Collaboration diagram for YGraph:



## Public Member Functions

- virtual [~YGraph](#) ()  
*Destructor.*
- virtual const char \* [widgetClass](#) () const  
*Returns a descriptive name of this widget class for logging, debugging etc.*
- virtual bool [setProperty](#) (const std::string &propertyName, const [YPropertyValue](#) &val)  
*Set a property.*
- virtual [YPropertyValue](#) [getProperty](#) (const std::string &propertyName)  
*Get a property.*
- virtual const [YPropertySet](#) & [propertySet](#) ()  
*Return this class's property set.*
- std::string [filename](#) () const  
*Return the filename that describes the graph.*
- virtual void [setFilename](#) (const std::string &filename)  
*Set the filename that describes the graph and render the graph.*
- std::string [layoutAlgorithm](#) () const  
*Return the layout-algorithm used for the graph.*
- virtual void [setLayoutAlgorithm](#) (const std::string &filename)  
*Set the layout-algorithm used for the graph.*
- virtual void [setGraph](#) (void \*graph)  
*Render the graph.*
- virtual std::string [activatedNode](#) () const  
*Return name of activated node.*

## Protected Member Functions

- [YGraph](#) ([YWidget](#) \*parent, const std::string &filename, const std::string &layoutAlgorithm)  
*Constructor.*
- [YGraph](#) ([YWidget](#) \*parent, void \*graph)  
*Constructor.*
- virtual void [renderGraph](#) (const std::string &filename, const std::string &layoutAlgorithm)=0  
*Render the graph from the filename.*
- virtual void [renderGraph](#) (void \*graph)=0  
*Render the graph.*

### 5.60.1 Detailed Description

A graph with nodes and edges, rendered with Graphviz.

Definition at line 45 of file [YGraph.h](#).

### 5.60.2 Constructor & Destructor Documentation

### 5.60.2.1 YGraph() [1/2]

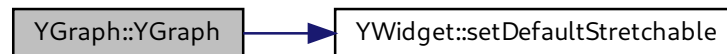
```
YGraph::YGraph (
    YWidget * parent,
    const std::string & filename,
    const std::string & layoutAlgorithm ) [protected]
```

Constructor.

Loads a graph in DOT format from filename and uses the layout algorithm layoutAlgorithm to layout and then render the graph. The layout algorithm can be any string accepted by the function gvLayout from graphviz, e.g. "dot" or "neato".

Definition at line 46 of file [YGraph.cc](#).

Here is the call graph for this function:



### 5.60.2.2 YGraph() [2/2]

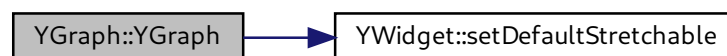
```
YGraph::YGraph (
    YWidget * parent,
    void * graph ) [protected]
```

Constructor.

Renders the graph. The graph must already contain layout information.

Definition at line 55 of file [YGraph.cc](#).

Here is the call graph for this function:



### 5.60.3 Member Function Documentation

#### 5.60.3.1 activatedNode()

```
string YGraph::activatedNode ( ) const [virtual]
```

Return name of activated node.

Activation can happen due to e.g. single right mouse click (context menu) or double left mouse click.

Definition at line 108 of file [YGraph.cc](#).

#### 5.60.3.2 getProperty()

```
YPropertyValue YGraph::getProperty (
    const std::string & propertyName ) [virtual]
```

Get a property.

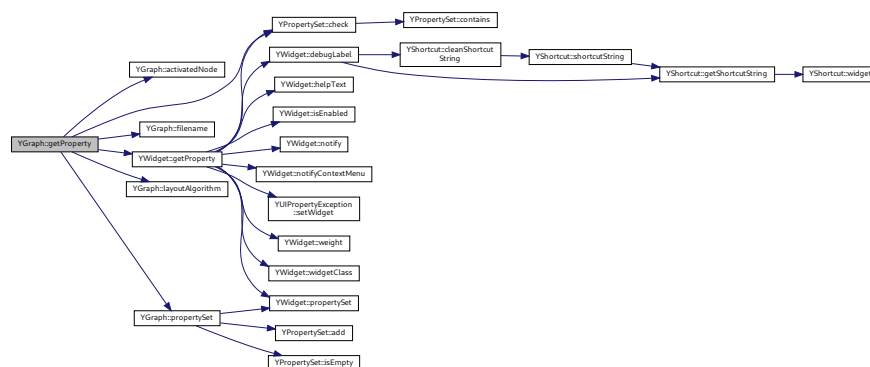
Reimplemented from [YWidget](#).

This method may throw YUIPropertyExceptions.

Reimplemented from [YWidget](#).

Definition at line 153 of file [YGraph.cc](#).

Here is the call graph for this function:



### 5.60.3.3 `propertySet()`

```
const YPropertySet & YGraph::propertySet ( ) [virtual]
```

Return this class's property set.

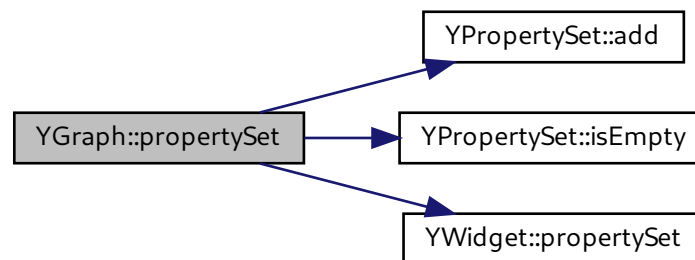
This also initializes the property upon the first call.

Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 115 of file [YGraph.cc](#).

Here is the call graph for this function:



### 5.60.3.4 `renderGraph()` [1/2]

```
virtual void YGraph::renderGraph (
    const std::string & filename,
    const std::string & layoutAlgorithm ) [protected], [pure virtual]
```

Render the graph from the filename.

Derived classes are required to implement this.

### 5.60.3.5 `renderGraph()` [2/2]

```
virtual void YGraph::renderGraph (
    void * graph ) [protected], [pure virtual]
```

Render the graph.

Derived classes are required to implement this.



### 5.60.3.6 setFilename()

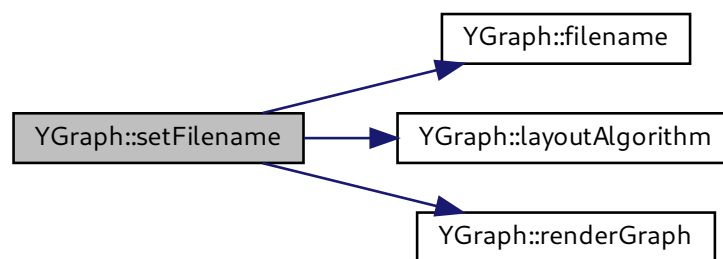
```
void YGraph::setFilename (
    const std::string & filename ) [virtual]
```

Set the filename that describes the graph and render the graph.

Derived classes can reimplement this, but they should call this base class method in the new implementation. Most derived classes only need to implement [renderGraph\(\)](#).

Definition at line 78 of file [YGraph.cc](#).

Here is the call graph for this function:



### 5.60.3.7 setGraph()

```
void YGraph::setGraph (
    void * graph ) [virtual]
```

Render the graph.

Derived classes can reimplement this, but they should call this base class method in the new implementation. Most derived classes only need to implement [renderGraph\(\)](#).

Definition at line 93 of file [YGraph.cc](#).

Here is the call graph for this function:



### 5.60.3.8 setLayoutAlgorithm()

```
void YGraph::setLayoutAlgorithm (
    const std::string & filename ) [virtual]
```

Set the layout-algorithm used for the graph.

Derived classes can reimplement this, but they should call this base class method in the new implementation.

Definition at line 101 of file [YGraph.cc](#).

Here is the call graph for this function:



### 5.60.3.9 setProperty()

```
bool YGraph::setProperty (
    const std::string & propertyName,
    const YPropertyValue & val ) [virtual]
```

Set a property.

Reimplemented from [YWidget](#).

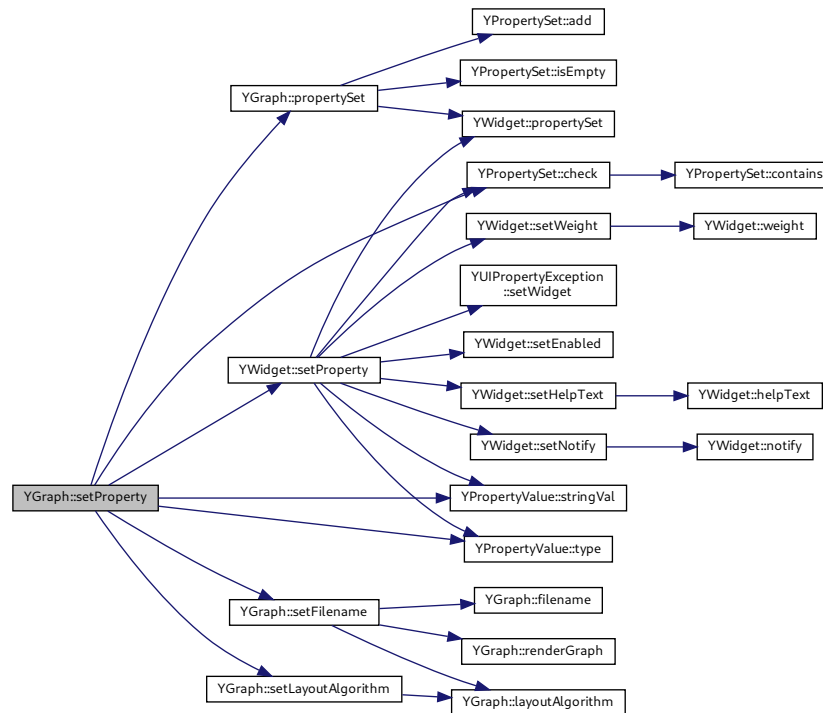
This function may throw `YUIPropertyExceptions`.

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from [YWidget](#).

Definition at line 137 of file [YGraph.cc](#).

Here is the call graph for this function:



The documentation for this class was generated from the following files:

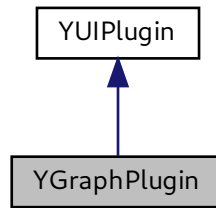
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YGraph.h
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YGraph.cc

## 5.61 YGraphPlugin Class Reference

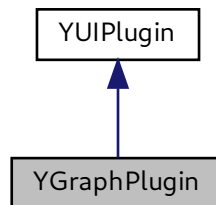
Abstract base class for simplified access to UI plugins for graph widget.

```
#include <YGraphPlugin.h>
```

Inheritance diagram for YGraphPlugin:



Collaboration diagram for YGraphPlugin:



## Public Member Functions

- virtual [YGraph](#) \* [createGraph](#) ([YWidget](#) \*parent, const std::string &filename, const std::string &layoutAlgorithm)=0  
*Create a graph widget.*

## Protected Member Functions

- [YGraphPlugin](#) (const char \*pluginLibBaseName)  
*Constructor: Load the specified plugin library from the standard UI plugin directory (/usr/lib/yui/).*
- virtual [~YGraphPlugin](#) ()  
*Destructor.*

### 5.61.1 Detailed Description

Abstract base class for simplified access to UI plugins for graph widget.

Definition at line 37 of file [YGraphPlugin.h](#).

## 5.61.2 Constructor & Destructor Documentation

### 5.61.2.1 ~YGraphPlugin()

```
virtual YGraphPlugin::~YGraphPlugin ( ) [inline], [protected], [virtual]
```

Destructor.

Calls `dlclose()` which will unload the plugin library if it is no longer used, i.e. if the reference count `dlopen()` uses reaches 0.

Definition at line 51 of file [YGraphPlugin.h](#).

## 5.61.3 Member Function Documentation

### 5.61.3.1 createGraph()

```
virtual YGraph* YGraphPlugin::createGraph (
    YWidget * parent,
    const std::string & filename,
    const std::string & layoutAlgorithm ) [pure virtual]
```

Create a graph widget.

Derived classes need to implement this.

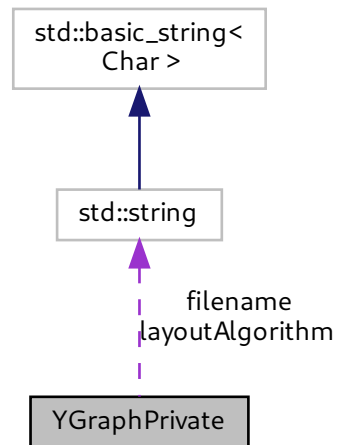
This might return 0 if the plugin lib could not be loaded or if the appropriate symbol could not be located in the plugin lib.

The documentation for this class was generated from the following file:

- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YGraphPlugin.h`

## 5.62 YGraphPrivate Struct Reference

Collaboration diagram for YGraphPrivate:



### Public Member Functions

- **YGraphPrivate** (string filename, string layoutAlgorithm)

### Public Attributes

- string **filename**
- string **layoutAlgorithm**

#### 5.62.1 Detailed Description

Definition at line 34 of file [YGraph.cc](#).

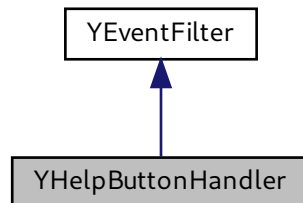
The documentation for this struct was generated from the following file:

- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YGraph.cc`

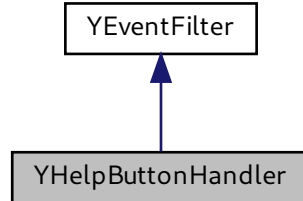
## 5.63 YHelpButtonHandler Class Reference

Helper class: Event filter that handles "Help" buttons.

Inheritance diagram for YHelpButtonHandler:



Collaboration diagram for YHelpButtonHandler:



### Public Member Functions

- **YHelpButtonHandler** ([YDialog](#) \*dialog)
- [YEvent](#) \* filter ([YEvent](#) \*event)

*The heart of the matter: The event filter function.*

### Additional Inherited Members

#### 5.63.1 Detailed Description

Helper class: Event filter that handles "Help" buttons.

Definition at line 74 of file [YDialog.cc](#).





## 5.64 YIconLoader Class Reference

### Public Member Functions

- `std::string findIcon` (`std::string name`)
- `void setIconBasePath` (`std::string path`)
- `std::string iconBasePath` () `const`
- `void addIconSearchPath` (`std::string path`)

### 5.64.1 Detailed Description

Definition at line 32 of file [YIconLoader.h](#).

The documentation for this class was generated from the following files:

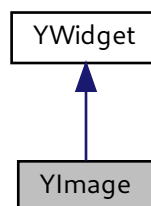
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YIconLoader.h`
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YIconLoader.cc`

## 5.65 YImage Class Reference

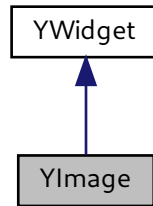
A picture, possibly animated, loaded from a file.

```
#include <YImage.h>
```

Inheritance diagram for YImage:



Collaboration diagram for YImage:



## Public Member Functions

- [YImage](#) ([YWidget](#) \*parent, const std::string &imageFileName, bool animated=false)  
*Constructor.*
- virtual [~YImage](#) ()  
*Destructor.*
- virtual const char \* [widgetClass](#) () const  
*Returns a descriptive name of this widget class for logging, debugging etc.*
- std::string [imageFileName](#) () const  
*Return the file name of this widget's image.*
- bool [animated](#) () const  
*Returns 'true' if the current image is an animated image format (e.g., MNG).*
- virtual void [setImage](#) (const std::string &imageFileName, bool animated=false)  
*Set and display a new image (or movie if animated is 'true').*
- void [setMovie](#) (const std::string &movieFileName)  
*Set and display a movie (an animated image).*
- bool [hasZeroSize](#) (YUIDimension dim) const  
*Return 'true' if the image widget should be stretchable with a default width of 0 in the specified dimension.*
- void [setZeroSize](#) (YUIDimension dim, bool zeroSize=true)  
*Make the image widget stretchable with a default size of 0 in the specified dimension.*
- bool [autoScale](#) () const  
*Return 'true' if the image should be scaled to fit into the available space.*
- virtual void [setAutoScale](#) (bool autoScale=true)  
*Make the image fit into the available space.*

## Additional Inherited Members

### 5.65.1 Detailed Description

A picture, possibly animated, loaded from a file.

Definition at line 37 of file [YImage.h](#).

## 5.65.2 Constructor & Destructor Documentation

### 5.65.2.1 YImage()

```
YImage::YImage (
    YWidget * parent,
    const std::string & imageFileName,
    bool animated = false )
```

Constructor.

'animated' indicates if 'imageFileName' is an animated image format (e.g., MNG).

Definition at line 56 of file [YImage.cc](#).

## 5.65.3 Member Function Documentation

### 5.65.3.1 hasZeroSize()

```
bool YImage::hasZeroSize (
    YUIDimension dim ) const
```

Return 'true' if the image widget should be stretchable with a default width of 0 in the specified dimension.

This is useful if the widget width is determined by outside constraints, like the width of a neighbouring widget.

Definition at line 91 of file [YImage.cc](#).

### 5.65.3.2 setAutoScale()

```
void YImage::setAutoScale (
    bool autoScale = true ) [virtual]
```

Make the image fit into the available space.

Derived classes should overwrite this, but call this base class function in the new function.

Definition at line 110 of file [YImage.cc](#).

Here is the call graph for this function:



### 5.65.3.3 setImage()

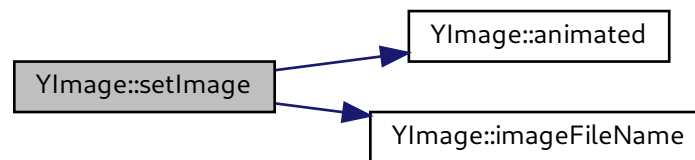
```
void YImage::setImage (
    const std::string & imageFileName,
    bool animated = false ) [virtual]
```

Set and display a new image (or movie if *animated* is 'true').

Derived classes should overwrite this, but call this base class function in the new function.

Definition at line 84 of file [YImage.cc](#).

Here is the call graph for this function:



### 5.65.3.4 setZeroSize()

```
void YImage::setZeroSize (
    YUIDimension dim,
    bool zeroSize = true )
```

Make the image widget stretchable with a default size of 0 in the specified dimension.

This is useful if the widget width is determined by outside constraints, like the width of a neighbouring widget.

This function is intentionally not virtual because it is only relevant during the next geometry update, in which case the derived class has to check this value anyway.

Definition at line 97 of file [YImage.cc](#).

Here is the call graph for this function:

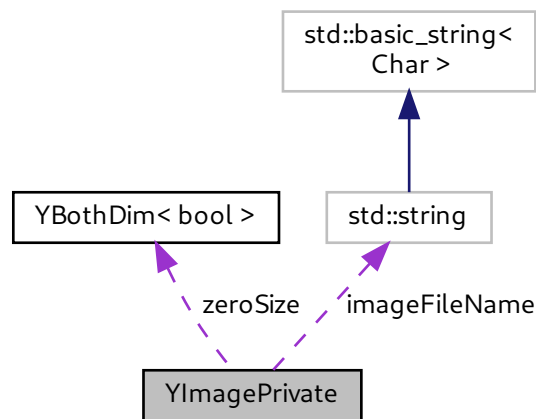


The documentation for this class was generated from the following files:

- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YImage.h`
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YImage.cc`

## 5.66 YImagePrivate Struct Reference

Collaboration diagram for YImagePrivate:



### Public Member Functions

- [YImagePrivate](#) (const string &imageFileName, bool animated)  
*Constructor.*

### Public Attributes

- string **imageFileName**
- bool **animated**
- [YBothDim< bool >](#) **zeroSize**
- bool **autoScale**

#### 5.66.1 Detailed Description

Definition at line 32 of file [YImage.cc](#).

The documentation for this struct was generated from the following file:

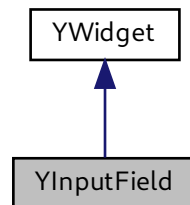
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YImage.cc`

## 5.67 YInputField Class Reference

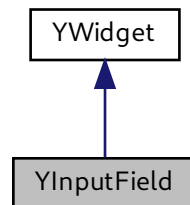
InputField: General purpose one line input field for entering text and other data.

```
#include <YInputField.h>
```

Inheritance diagram for YInputField:



Collaboration diagram for YInputField:



### Public Member Functions

- virtual `~YInputField()`  
*Destructor.*
- virtual const char \* `widgetClass()` const  
*Return a descriptive name of this widget class for logging, debugging etc.*
- virtual std::string `value()`=0  
*Get the current value (the text entered by the user or set from the outside) of this input field.*
- virtual void `setValue` (const std::string &text)=0  
*Set the current value (the text entered by the user or set from the outside) of this input field.*

- `std::string label () const`  
*Get the label (the caption above the input field).*
- `virtual void setLabel (const std::string &label)`  
*Set the label (the caption above the input field).*
- `bool passwordMode () const`  
*Returns 'true' if this input field is in password mode, i.e.*
- `std::string validChars ()`  
*Get the valid input characters.*
- `virtual void setValidChars (const std::string &validChars)`  
*Set the valid input characters.*
- `int inputMaxLength () const`  
*The maximum input length, i.e., the maximum number of characters the user can enter.*
- `virtual void setInputMaxLength (int numberOfChars)`  
*Set the maximum input length, i.e., the maximum number of characters the user can enter.*
- `bool shrinkable () const`  
*Return 'true' if this InputField should be very small.*
- `virtual void setShrinkable (bool shrinkable=true)`  
*Make this InputField very small.*
- `virtual bool setProperty (const std::string &propertyName, const YPropertyValue &val)`  
*Set a property.*
- `virtual YPropertyValue getProperty (const std::string &propertyName)`  
*Get a property.*
- `virtual const YPropertySet & propertySet ()`  
*Return this class's property set.*
- `virtual std::string shortcutString () const`  
*Get the string of this widget that holds the keyboard shortcut.*
- `virtual void setShortcutString (const std::string &str)`  
*Set the string of this widget that holds the keyboard shortcut.*
- `const char * userInputProperty ()`  
*The name of the widget property that will return user input.*
- `virtual void saveUserInput (YMacroRecorder *macroRecorder)`  
*Save the widget's user input to a macro recorder.*

## Protected Member Functions

- `YInputField (YWidget *parent, const std::string &label, bool passwordMode=false)`  
*Constructor.*

### 5.67.1 Detailed Description

InputField: General purpose one line input field for entering text and other data.

Can be used for entering passwords with a "\*" echoed for every character typed.

Like most widgets, the InputField has a label (a caption) above the input field itself. The label can and should get a keyboard shortcut (specified with '&') that will make the input field receive the keyboard focus with a special key combination ("&Name" -> Alt-N or Ctrl-N will make the keyboard focus jump to the corresponding input field).

Definition at line 46 of file [YInputField.h](#).

## 5.67.2 Constructor & Destructor Documentation

### 5.67.2.1 YInputField()

```
YInputField::YInputField (
    YWidget * parent,
    const std::string & label,
    bool passwordMode = false ) [protected]
```

Constructor.

Create an input field with 'label' as the caption. If 'passwordMode' is set, the input will be not be echoed as clear text.

Definition at line 54 of file [YInputField.cc](#).

Here is the call graph for this function:



## 5.67.3 Member Function Documentation

### 5.67.3.1 getProperty()

```
YPropertyValue YInputField::getProperty (
    const std::string & propertyName ) [virtual]
```

Get a property.

Reimplemented from [YWidget](#).

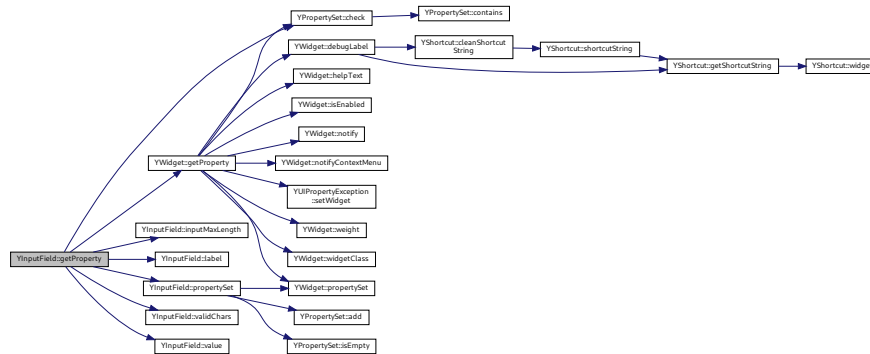
This method may throw YUIPropertyExceptions.

Reimplemented from [YWidget](#).

Definition at line 169 of file [YInputField.cc](#).



Here is the call graph for this function:



### 5.67.3.2 inputMaxLength()

```
int YInputField::inputMaxLength ( ) const
```

The maximum input length, i.e., the maximum number of characters the user can enter.

-1 means no limit.

Definition at line 114 of file [YInputField.cc](#).

### 5.67.3.3 passwordMode()

```
bool YInputField::passwordMode ( ) const
```

Returns 'true' if this input field is in password mode, i.e.

if there should be no on-screen echo or only a '\*' for each character typed.

Notice that this can only be set in the constructor.

Definition at line 83 of file [YInputField.cc](#).

#### 5.67.3.4 `propertySet()`

```
const YPropertySet & YInputField::propertySet ( ) [virtual]
```

Return this class's property set.

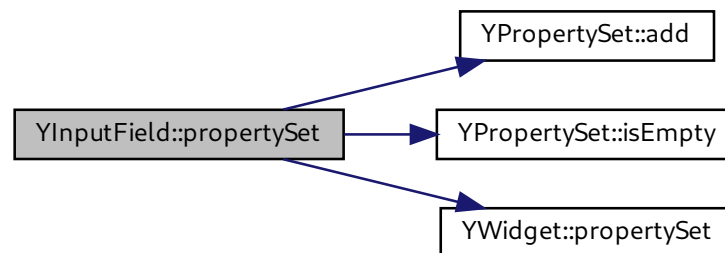
This also initializes the property upon the first call.

Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 127 of file [YInputField.cc](#).

Here is the call graph for this function:



#### 5.67.3.5 `saveUserInput()`

```
void YInputField::saveUserInput (
    YMacroRecorder * macroRecorder ) [virtual]
```

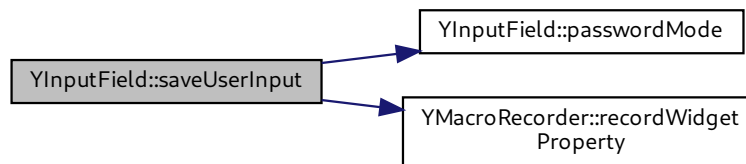
Save the widget's user input to a macro recorder.

Reimplemented from [YWidget](#) to avoid recording passwords.

Reimplemented from [YWidget](#).

Definition at line 185 of file [YInputField.cc](#).

Here is the call graph for this function:



### 5.67.3.6 setInputMaxLength()

```
void YInputField::setInputMaxLength (
    int numberOfChars ) [virtual]
```

Set the maximum input length, i.e., the maximum number of characters the user can enter.

-1 means no limit.

Derived classes are free to reimplement this, but they should call this base class method at the end of the overloaded function.

Definition at line 120 of file [YInputField.cc](#).

### 5.67.3.7 setLabel()

```
void YInputField::setLabel (
    const std::string & label ) [virtual]
```

Set the label (the caption above the input field).

Derived classes are free to reimplement this, but they should call this base class method at the end of the overloaded function.

Definition at line 77 of file [YInputField.cc](#).

Here is the call graph for this function:



### 5.67.3.8 setProperty()

```
bool YInputField::setProperty (
    const std::string & propertyName,
    const YPropertyValue & val ) [virtual]
```

Set a property.

Reimplemented from [YWidget](#).

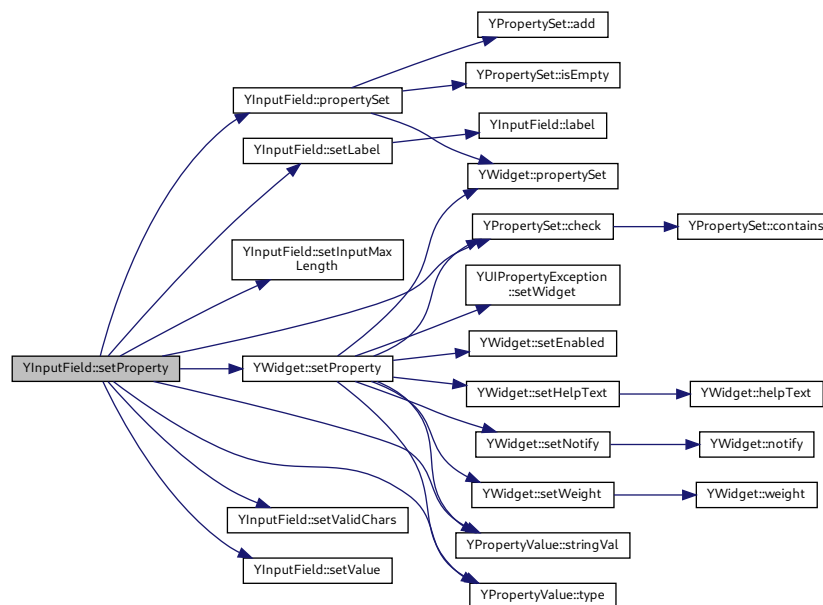
This function may throw YUIPropertyExceptions.

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from [YWidget](#).

Definition at line 151 of file [YInputField.cc](#).

Here is the call graph for this function:



### 5.67.3.9 setShortcutString()

```
virtual void YInputField::setShortcutString (
    const std::string & str ) [inline], [virtual]
```

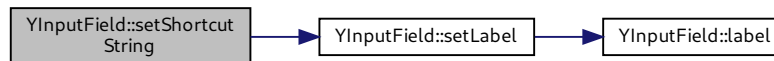
Set the string of this widget that holds the keyboard shortcut.

Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 193 of file [YInputField.h](#).

Here is the call graph for this function:



### 5.67.3.10 setShrinkable()

```
void YInputField::setShrinkable (
    bool shrinkable = true ) [virtual]
```

Make this InputField very small.

This will take effect only upon the next geometry management run.

Derived classes can overwrite this, but should call this base class function in the new function.

Definition at line 95 of file [YInputField.cc](#).

Here is the call graph for this function:



#### 5.67.3.11 `setValidChars()`

```
void YInputField::setValidChars (
    const std::string & validChars ) [virtual]
```

Set the valid input characters.

No input validation is performed (i.e., the user can enter anything) if this is empty.

Derived classes are free to reimplement this, but they should call this base class method at the end of the overloaded function.

Definition at line 108 of file [YInputField.cc](#).

#### 5.67.3.12 `setValue()`

```
virtual void YInputField::setValue (
    const std::string & text ) [pure virtual]
```

Set the current value (the text entered by the user or set from the outside) of this input field.

Derived classes are required to implement this.

#### 5.67.3.13 `shortcutString()`

```
virtual std::string YInputField::shortcutString ( ) const [inline], [virtual]
```

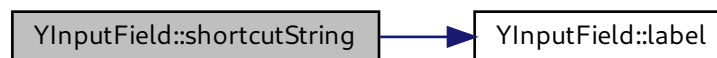
Get the string of this widget that holds the keyboard shortcut.

Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 186 of file [YInputField.h](#).

Here is the call graph for this function:



#### 5.67.3.14 userInputProperty()

```
const char* YInputField::userInputProperty ( ) [inline], [virtual]
```

The name of the widget property that will return user input.

Inherited from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 200 of file [YInputField.h](#).

#### 5.67.3.15 validChars()

```
string YInputField::validChars ( )
```

Get the valid input characters.

No input validation is performed (i.e., the user can enter anything) if this is empty.

Definition at line 102 of file [YInputField.cc](#).

#### 5.67.3.16 value()

```
virtual std::string YInputField::value ( ) [pure virtual]
```

Get the current value (the text entered by the user or set from the outside) of this input field.

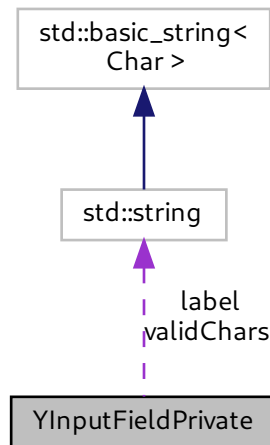
Derived classes are required to implement this.

The documentation for this class was generated from the following files:

- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YInputField.h](#)
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YInputField.cc](#)

## 5.68 YInputFieldPrivate Struct Reference

Collaboration diagram for YInputFieldPrivate:



### Public Member Functions

- **YInputFieldPrivate** (string label, bool passwordMode)

### Public Attributes

- string **label**
- bool **passwordMode**
- bool **shrinkable**
- string **validChars**
- int **inputMaxLength**

### 5.68.1 Detailed Description

Definition at line 36 of file [YInputField.cc](#).

The documentation for this struct was generated from the following file:

- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YInputField.cc`

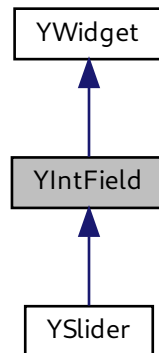


## 5.69 YIntField Class Reference

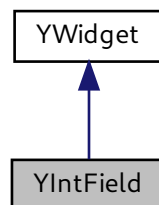
IntField: Input field for integer values.

```
#include <YIntField.h>
```

Inheritance diagram for YIntField:



Collaboration diagram for YIntField:



### Public Member Functions

- virtual `~YIntField ()`  
*Destructor.*
- virtual const char \* `widgetClass ()` const  
*Return a descriptive name of this widget class for logging, debugging etc.*

- virtual int [value](#) ()=0  
*Get the current value (the number entered by the user or set from the outside) of this IntField.*
- void [setValue](#) (int val)  
*Set the current value (the number entered by the user or set from the outside) of this IntField.*
- int [minValue](#) () const  
*Return the minimum value.*
- void [setMinValue](#) (int val)  
*Set a new minimum value.*
- int [maxValue](#) () const  
*Return the maximum value.*
- void [setMaxValue](#) (int val)  
*Set a new maximum value.*
- std::string [label](#) () const  
*Get the label (the caption above the input field).*
- virtual void [setLabel](#) (const std::string &label)  
*Set the label (the caption above the input field).*
- virtual bool [setProperty](#) (const std::string &propertyName, const [YPropertyValue](#) &val)  
*Set a property.*
- virtual [YPropertyValue](#) [getProperty](#) (const std::string &propertyName)  
*Get a property.*
- virtual const [YPropertySet](#) & [propertySet](#) ()  
*Return this class's property set.*
- virtual std::string [shortcutString](#) () const  
*Get the string of this widget that holds the keyboard shortcut.*
- virtual void [setShortcutString](#) (const std::string &str)  
*Set the string of this widget that holds the keyboard shortcut.*
- const char \* [userInputProperty](#) ()  
*The name of the widget property that will return user input.*

## Protected Member Functions

- [YIntField](#) ([YWidget](#) \*parent, const std::string &label, int [minValue](#), int [maxValue](#))  
*Constructor.*
- virtual void [setValueInternal](#) (int val)=0  
*Set the current value (the number entered by the user or set from the outside) of this IntField.*
- int [enforceRange](#) (int val) const  
*Enforce 'val' to be between minValue and maxValue.*

### 5.69.1 Detailed Description

IntField: Input field for integer values.

Enforces input range between a specified minimum and maximum value.

Definition at line 38 of file [YIntField.h](#).

## 5.69.2 Constructor & Destructor Documentation

### 5.69.2.1 YIntField()

```
YIntField::YIntField (
    YWidget * parent,
    const std::string & label,
    int minValue,
    int maxValue ) [protected]
```

Constructor.

Create an IntField with 'label' as the caption, and the specified minimum and maximum values.

Note that YWidgetFactory::createIntField() also has an 'initialValue' parameter that is not used here (because the current value is not stored in this base class, but in the derived class).

Definition at line 52 of file [YIntField.cc](#).

Here is the call graph for this function:



## 5.69.3 Member Function Documentation

### 5.69.3.1 enforceRange()

```
int YIntField::enforceRange (
    int val ) const [protected]
```

Enforce 'val' to be between minValue and maxValue.

Return a value that is in range. This does not change the internally stored value of this IntField in any way.

Definition at line 73 of file [YIntField.cc](#).

Here is the call graph for this function:



### 5.69.3.2 `getProperty()`

```
YPropertyValue YIntField::getProperty (
    const std::string & propertyName ) [virtual]
```

Get a property.

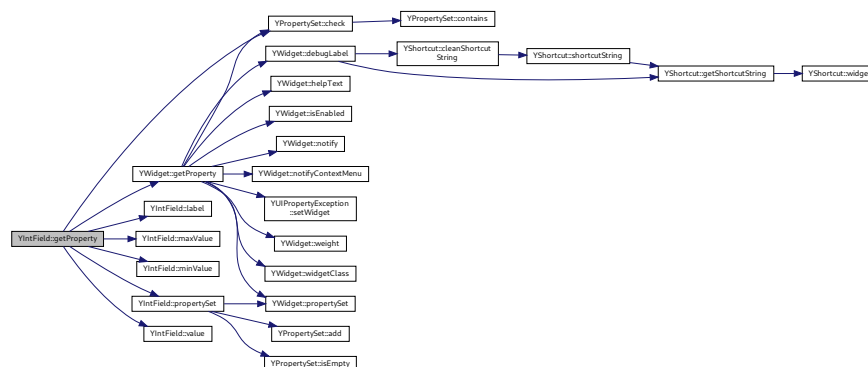
Reimplemented from [YWidget](#).

This method may throw `YUIPropertyExceptions`.

Reimplemented from [YWidget](#).

Definition at line 183 of file [YIntField.cc](#).

Here is the call graph for this function:



### 5.69.3.3 propertySet()

```
const YPropertySet & YIntField::propertySet ( ) [virtual]
```

Return this class's property set.

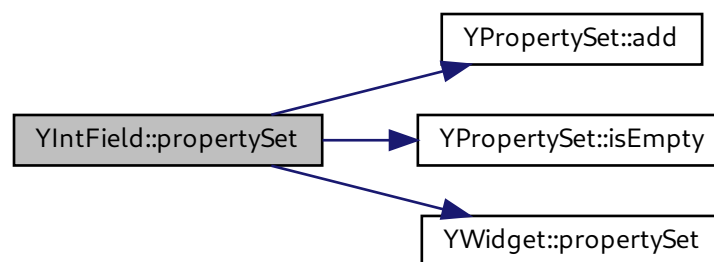
This also initializes the property upon the first call.

Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 141 of file [YIntField.cc](#).

Here is the call graph for this function:



### 5.69.3.4 setLabel()

```
void YIntField::setLabel (
    const std::string & label ) [virtual]
```

Set the label (the caption above the input field).

Derived classes are free to reimplement this, but they should call this base class method at the end of the overloaded function.

Definition at line 133 of file [YIntField.cc](#).

Here is the call graph for this function:



### 5.69.3.5 setMaxValue()

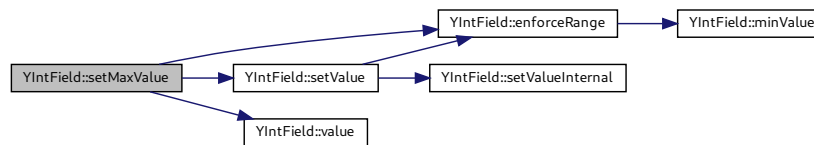
```
void YIntField::setMaxValue (
    int val )
```

Set a new maximum value.

If the current value is greater than that, it will be set to the new maximum.

Definition at line 113 of file [YIntField.cc](#).

Here is the call graph for this function:



### 5.69.3.6 setMinValue()

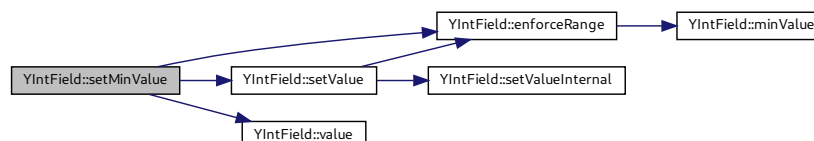
```
void YIntField::setMinValue (
    int val )
```

Set a new minimum value.

If the current value is less than that, it will be set to the new minimum.

Definition at line 93 of file [YIntField.cc](#).

Here is the call graph for this function:



### 5.69.3.7 setProperty()

```
bool YIntField::setProperty (
    const std::string & propertyName,
    const YPropertyValue & val ) [virtual]
```

Set a property.

Reimplemented from [YWidget](#).

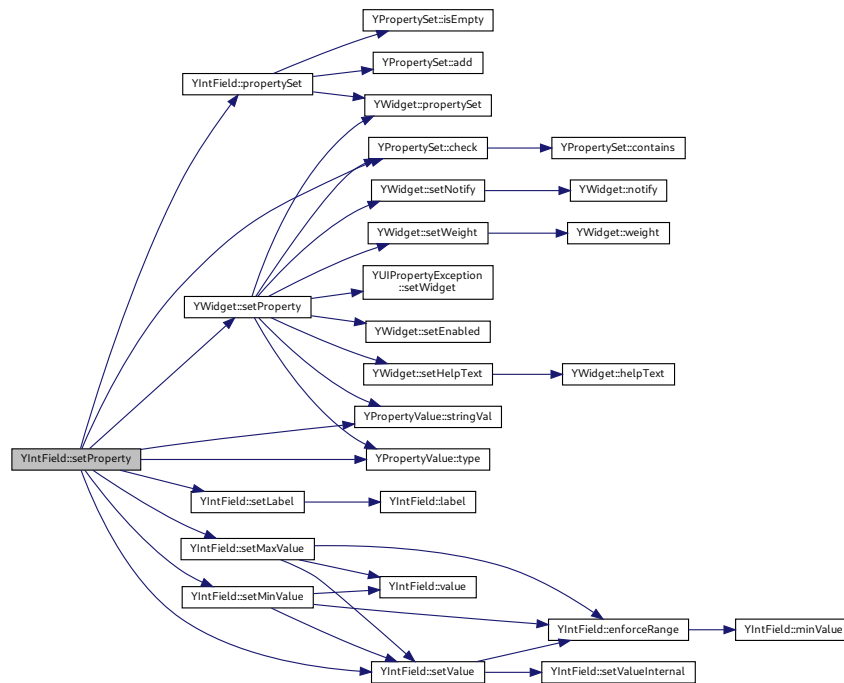
This function may throw [YUIPropertyExceptions](#).

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from [YWidget](#).

Definition at line 165 of file [YIntField.cc](#).

Here is the call graph for this function:



### 5.69.3.8 setShortcutString()

```
virtual void YIntField::setShortcutString (
    const std::string & str ) [inline], [virtual]
```

Set the string of this widget that holds the keyboard shortcut.

Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 179 of file [YIntField.h](#).

Here is the call graph for this function:



### 5.69.3.9 setValue()

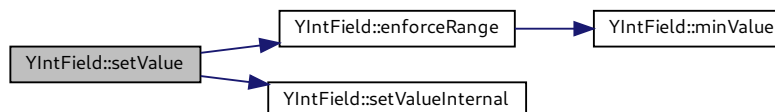
```
void YIntField::setValue (
    int val ) [inline]
```

Set the current value (the number entered by the user or set from the outside) of this IntField.

This method enforces 'val' to be between `minValue` and `maxValue`.

Definition at line 81 of file [YIntField.h](#).

Here is the call graph for this function:





#### 5.69.3.10 setValueInternal()

```
virtual void YIntField::setValueInternal (
    int val ) [protected], [pure virtual]
```

Set the current value (the number entered by the user or set from the outside) of this IntField.

'val' is guaranteed to be between minValue and maxValue; no further checks are required.

Derived classes are required to implement this method.

#### 5.69.3.11 shortcutString()

```
virtual std::string YIntField::shortcutString ( ) const [inline], [virtual]
```

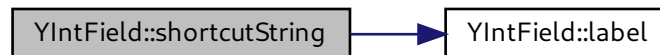
Get the string of this widget that holds the keyboard shortcut.

Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 172 of file [YIntField.h](#).

Here is the call graph for this function:



#### 5.69.3.12 userInputProperty()

```
const char* YIntField::userInputProperty ( ) [inline], [virtual]
```

The name of the widget property that will return user input.

Inherited from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 186 of file [YIntField.h](#).

### 5.69.3.13 value()

```
virtual int YIntField::value ( ) [pure virtual]
```

Get the current value (the number entered by the user or set from the outside) of this IntField.

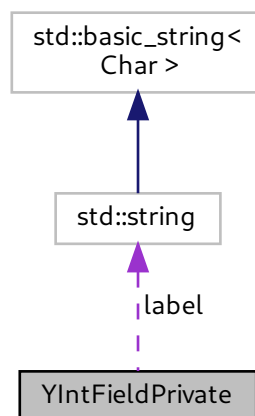
Derived classes are required to implement this.

The documentation for this class was generated from the following files:

- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YIntField.h
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YIntField.cc

## 5.70 YIntFieldPrivate Struct Reference

Collaboration diagram for YIntFieldPrivate:



### Public Member Functions

- **YIntFieldPrivate** (const string &label, int minValue, int maxValue)

### Public Attributes

- string **label**
- int **minValue**
- int **maxValue**

### 5.70.1 Detailed Description

Definition at line 34 of file [YIntField.cc](#).

The documentation for this struct was generated from the following file:

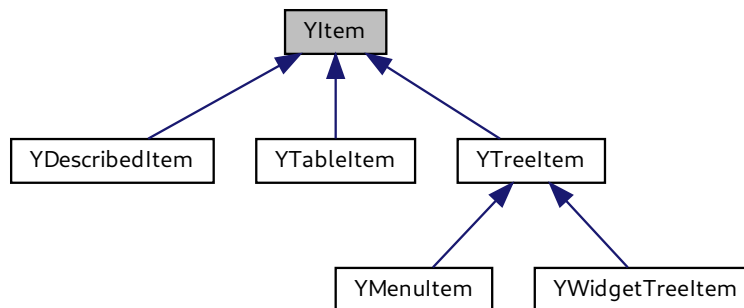
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YIntField.cc](#)

## 5.71 YItem Class Reference

Simple item class for SelectionBox, ComboBox, MultiSelectionBox etc.

```
#include <YItem.h>
```

Inheritance diagram for YItem:



### Public Member Functions

- [YItem](#) (const std::string &label, bool selected=false)  
*Constructor with just the label and optionally the selected state.*
- [YItem](#) (const std::string &label, const std::string &iconName, bool selected=false)  
*Constructor with label and icon name and optionally the selected state.*
- virtual [~YItem](#) ()  
*Destructor.*
- std::string [label](#) () const  
*Return this item's label.*
- void [setLabel](#) (const std::string &newLabel)  
*Set this item's label.*
- std::string [iconName](#) () const  
*Return this item's icon name.*

- bool [hasIconName](#) () const  
*Return 'true' if this item has an icon name.*
- void [setIconName](#) (const std::string &newIconName)  
*Set this item's icon name.*
- bool [selected](#) () const  
*Return 'true' if this item is currently selected.*
- void [setSelected](#) (bool sel=true)  
*Select or unselect this item.*
- int [status](#) () const  
*Return the status of this item.*
- void [setStatus](#) (int newStatus)  
*Set the status of this item.*
- void [setIndex](#) (int [index](#))  
*Set this item's index.*
- int [index](#) () const  
*Return the index of this item (as set with [setIndex\(\)](#) ).*
- void [setData](#) (void \*newData)  
*Set the opaque data pointer for application use.*
- void \* [data](#) () const  
*Return the opaque data pointer.*
- virtual bool [hasChildren](#) () const  
*Return 'true' if this item has any child items.*
- virtual [YItemIterator](#) [childrenBegin](#) ()  
*Return an iterator that points to the first child item of this item.*
- virtual [YItemConstIterator](#) [childrenBegin](#) () const
- virtual [YItemIterator](#) [childrenEnd](#) ()  
*Return an iterator that points after the last child item of this item.*
- virtual [YItemConstIterator](#) [childrenEnd](#) () const
- virtual [YItem](#) \* [parent](#) () const  
*Returns this item's parent item or 0 if it is a toplevel item.*

### 5.71.1 Detailed Description

Simple item class for SelectionBox, ComboBox, MultiSelectionBox etc.

items. This class provides stubs for children management.

Definition at line 49 of file [YItem.h](#).

### 5.71.2 Member Function Documentation

### 5.71.2.1 childrenBegin()

```
virtual YItemIterator YItem::childrenBegin ( ) [inline], [virtual]
```

Return an iterator that points to the first child item of this item.

This default implementation returns the 'end' iterator of the class-static always empty `_noChildren YItemCollection`. It is safe to use this iterator in classic iterator loops:

```
for ( YItemIterator it = myItem->childrenBegin(); it != myItem->childrenEnd(); ++it ) { ... }
```

The loop body will only ever be executed if this item is a derived class that actually manages child items.

Reimplemented in [YTreeItem](#).

Definition at line 186 of file [YItem.h](#).

### 5.71.2.2 childrenEnd()

```
virtual YItemIterator YItem::childrenEnd ( ) [inline], [virtual]
```

Return an iterator that points after the last child item of this item.

This default implementation returns the 'end' iterator of the class-static always empty `_noChildren YItemCollection`.

Reimplemented in [YTreeItem](#).

Definition at line 195 of file [YItem.h](#).

### 5.71.2.3 label()

```
std::string YItem::label ( ) const [inline]
```

Return this item's label.

This is what the user sees in a dialog, so this will usually be a translated text.

Definition at line 82 of file [YItem.h](#).

#### 5.71.2.4 parent()

```
virtual YItem* YItem::parent ( ) const [inline], [virtual]
```

Returns this item's parent item or 0 if it is a toplevel item.

This default implementation always returns 0. Derived classes that handle children should reimplement this.

Reimplemented in [YTreeItem](#), and [YMenuItem](#).

Definition at line 203 of file [YItem.h](#).

#### 5.71.2.5 setData()

```
void YItem::setData (
    void * newData ) [inline]
```

Set the opaque data pointer for application use.

Applications can use this to store the pointer to a counterpart of this tree item. It is the application's responsibility to watch for dangling pointers and possibly deleting the data. All this class ever does with this pointer is to store it.

Definition at line 148 of file [YItem.h](#).

#### 5.71.2.6 setSelected()

```
void YItem::setSelected (
    bool sel = true ) [inline]
```

Select or unselect this item.

This does not have any effect on any other item; if it is desired that only one item is selected at any time, the caller has to take care of that.

Definition at line 114 of file [YItem.h](#).

#### 5.71.2.7 setStatus()

```
void YItem::setStatus (
    int newStatus ) [inline]
```

Set the status of this item.

Most widgets only use 0 for "not selected" or nonzero for "selected". Some widgets may make use of other values as well.

Definition at line 128 of file [YItem.h](#).

### 5.71.2.8 status()

```
int YItem::status ( ) const [inline]
```

Return the status of this item.

This is a bit more generalized than 'selected'. Values other than 0 or 1 can mean different things to the application or to the specific widget.

Definition at line 121 of file [YItem.h](#).

The documentation for this class was generated from the following files:

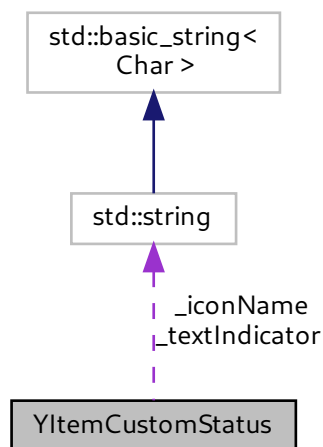
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YItem.h](#)
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YItem.cc](#)

## 5.72 YItemCustomStatus Class Reference

Class describing a non-binary status for an item.

```
#include <YItemCustomStatus.h>
```

Collaboration diagram for YItemCustomStatus:



## Public Member Functions

- [YItemCustomStatus](#) (const std::string &[iconName](#), const std::string &[textIndicator](#), int [nextStatus](#)== -1)  
*Constructor.*
- const std::string & [iconName](#) () const  
*The name of an icon to use in the widget in a graphical UI if an item has this status.*
- const std::string & [textIndicator](#) () const  
*A text representation of this status in a text-based UI if an item has this status, for example "[ ]", "[x]" or "[ ]", "[ +]", "[a+]".*
- int [nextStatus](#) () const  
*This returns the next status to cycle through if the user clicks on the status or cycles through status values with the corresponding shortcut key.*
- void [setNextStatus](#) (int value)  
*Set the next status.*
- bool [hasNextStatus](#) () const  
*Return 'true' if a next status to cycle to is defined for this status, 'false' if not.*

## Protected Attributes

- std::string [\\_iconName](#)
- std::string [\\_textIndicator](#)
- int [\\_nextStatus](#)

### 5.72.1 Detailed Description

Class describing a non-binary status for an item.

This is an extension of normal boolean item states: Rather than just "on" or "off" like for check boxes or radio buttons, a status of this kind can have any nonnegative integer number. The number is implicitly the index of this status in the corresponding vector.

For symmetry with boolean states, a value of 0 is defined to have the same semantics as "off" / "unselected", 1 has the semantics of "on" / "fully selected", and all other values are purely application-defined.

Definition at line 44 of file [YItemCustomStatus.h](#).

### 5.72.2 Member Function Documentation

#### 5.72.2.1 nextStatus()

```
int YItemCustomStatus::nextStatus ( ) const [inline]
```

This returns the next status to cycle through if the user clicks on the status or cycles through status values with the corresponding shortcut key.

If no such value was specified, this returns -1. The application can then still choose to set a different status depending on other application data.

Definition at line 82 of file [YItemCustomStatus.h](#).



### 5.72.2.2 setNextStatus()

```
void YItemCustomStatus::setNextStatus (
    int value ) [inline]
```

Set the next status.

This should only be done once when the status transition map is evaluated.

Definition at line 88 of file [YItemCustomStatus.h](#).

### 5.72.2.3 textIndicator()

```
const std::string& YItemCustomStatus::textIndicator ( ) const [inline]
```

A text representation of this status in a text-based UI if an item has this status, for example "[ ]", "[x]" or "[ ]", "[ +]", "[a+]".

It is recommended to use the same character length for all states so all items line up properly, even if they have different states.

Definition at line 71 of file [YItemCustomStatus.h](#).

The documentation for this class was generated from the following file:

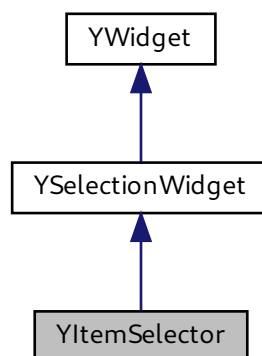
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YItemCustomStatus.h](#)

## 5.73 YItemSelector Class Reference

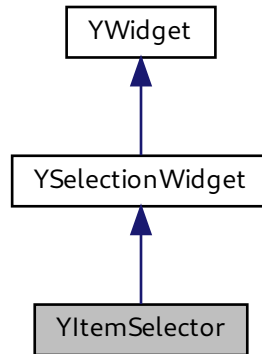
Scrollable item selector widget with not only a label for each item, but also a (possible multi-line) description and an optional icon.

```
#include <YItemSelector.h>
```

Inheritance diagram for YItemSelector:



Collaboration diagram for YItemSelector:



## Public Member Functions

- [YItemSelector](#) ([YWidget](#) \*parent, bool [enforceSingleSelection](#)=true)  
*Standard constructor.*
- [YItemSelector](#) ([YWidget](#) \*parent, const [YItemCustomStatusVector](#) &customStates)  
*Constructor for custom item status values.*
- virtual [~YItemSelector](#) ()  
*Destructor.*
- virtual const char \* [widgetClass](#) () const  
*Return a descriptive name of this widget class for logging, debugging etc.*
- int [visibleItems](#) () const  
*Return the number of visible items (i.e.*
- virtual void [setVisibleItems](#) (int newVal)  
*Set the number of visible items.*
- virtual void [setItemStatus](#) ([YItem](#) \*item, int status)  
*Set the status of an item.*
- bool [usingCustomStatus](#) () const  
*Return 'true' if this widget uses custom status values, 'false' if not (i.e.*
- int [customStatusCount](#) () const  
*Return the number of custom status values or 0 if no custom status values are used.*
- const [YItemCustomStatus](#) & [customStatus](#) (int index)  
*Return the custom status with the specified index (counting from 0).*
- bool [validCustomStatusIndex](#) (int index) const  
*Return 'true' if a custom status index is within the valid range, i.e.*
- int [cycleCustomStatus](#) (int oldStatus)  
*Cycle through the custom status values according to the custom status table, i.e.*
- virtual bool [setProperty](#) (const std::string &propertyName, const [YPropertyValue](#) &val)

*Set a property.*

- virtual [YPropertyValue](#) [getProperty](#) (const std::string &propertyName)

*Get a property.*

- virtual const [YPropertySet](#) & [propertySet](#) ()

*Return this class's property set.*

- const char \* [userInputProperty](#) ()

*The name of the widget property that will return user input.*

- virtual void [activateItem](#) ([YItem](#) \*item)=0

*Activate selected item.*

## Protected Member Functions

- virtual void [updateCustomStatusIndicator](#) ([YItem](#) \*item)

*Update the status indicator (status icon or text indicator) if this widget is using custom status values.*

### 5.73.1 Detailed Description

Scrollable item selector widget with not only a label for each item, but also a (possible multi-line) description and an optional icon.

This widget supports both 1-of-n or n-of-m selection, i.e. it can act as a (more verbose and more screen space consuming) replacement for [YSelectionBox](#) or [YMultiSelectionBox](#).

Definition at line 43 of file [YItemSelector.h](#).

### 5.73.2 Constructor & Destructor Documentation

#### 5.73.2.1 YItemSelector()

```
YItemSelector::YItemSelector (
    YWidget * parent,
    const YItemCustomStatusVector & customStates )
```

Constructor for custom item status values.

This makes it possible to set a wider variety of values than just 0 or

1. The semantics behind the individual status values is purely application defined; the specified customStates description only provides an icon (for graphical UIs) or a text representation (for text-based UIs) and an optional "next" status value that can be used to let the user cycle through different status values. The numeric value of each status is implicitly its index in the vector.

Notice that this constructor is the only way to set custom status value descriptions; they cannot be changed anymore after initializing the widget. This is by design so that any derived widgets in concrete UIs do not have to bother with possibly recreating any subwidgets if this should change; this guarantees that it does not change, neither the fact that there are custom status values nor their number or indicator icons or texts.

This constructor implicitly sets 'enforceSingleSelection' to 'false'.

In this mode, the widget sends [YMenuEvents](#) (which include the item that the user changed) if the notify option is set. For anything beyond the simple status transitions that are defined here in 'customStates', it is highly recommended to set that notify option and to handle those [YMenuEvents](#) on the application level.

Definition at line 59 of file [YItemSelector.cc](#).

### 5.73.3 Member Function Documentation

#### 5.73.3.1 activateItem()

```
virtual void YItemSelector::activateItem (
    YItem * item ) [pure virtual]
```

Activate selected item.

Can be used in tests to simulate user input.

Derived classes are required to implement this.

#### 5.73.3.2 customStatus()

```
const YItemCustomStatus & YItemSelector::customStatus (
    int index )
```

Return the custom status with the specified index (counting from 0).

Notice that this may throw a `std::out_of_range` exception if the index is invalid.

Definition at line 149 of file [YItemSelector.cc](#).

#### 5.73.3.3 cycleCustomStatus()

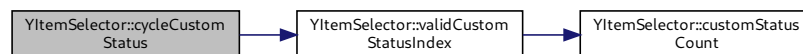
```
int YItemSelector::cycleCustomStatus (
    int oldStatus )
```

Cycle through the custom status values according to the custom status table, i.e.

return the 'nextStatus' field of table index 'oldStatus'. This may be -1 if no next status was specified there or if 'oldStatus' is out of range of that table.

Definition at line 196 of file [YItemSelector.cc](#).

Here is the call graph for this function:



### 5.73.3.4 getProperty()

```
YPropertyValue YItemSelector::getProperty (
    const std::string & propertyName ) [virtual]
```

Get a property.

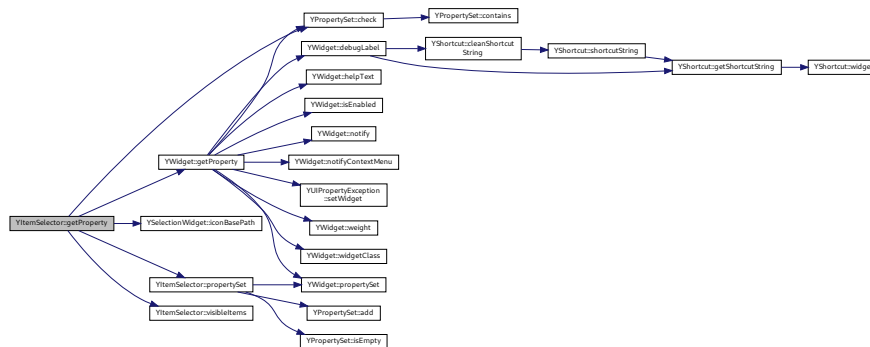
Reimplemented from [YWidget](#).

This method may throw [YUIPropertyExceptions](#).

Reimplemented from [YWidget](#).

Definition at line 260 of file [YItemSelector.cc](#).

Here is the call graph for this function:



### 5.73.3.5 propertySet()

```
const YPropertySet & YItemSelector::propertySet ( ) [virtual]
```

Return this class's property set.

This also initializes the property upon the first call.

Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 209 of file [YItemSelector.cc](#).



### 5.73.3.7 setProperty()

```
bool YItemSelector::setProperty (
    const std::string & propertyName,
    const YPropertyValue & val ) [virtual]
```

Set a property.

Reimplemented from [YWidget](#).

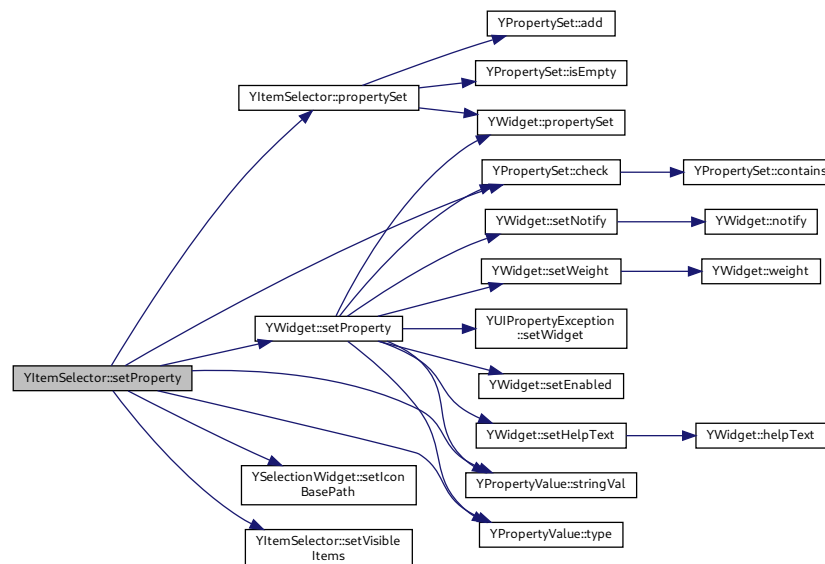
This function may throw YUIPropertyExceptions.

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from [YWidget](#).

Definition at line 239 of file [YItemSelector.cc](#).

Here is the call graph for this function:



### 5.73.3.8 setVisibleItems()

```
void YItemSelector::setVisibleItems (
    int newVal ) [virtual]
```

Set the number of visible items.

When changing this, make sure to recalculate the layout (`YDialog::recalc()`) so the change has any effect.

Derived classes are free to reimplement this, but they should call this base class method in the overloaded function.

Definition at line 106 of file [YItemSelector.cc](#).

#### 5.73.3.9 updateCustomStatusIndicator()

```
virtual void YItemSelector::updateCustomStatusIndicator (
    YItem * item ) [inline], [protected], [virtual]
```

Update the status indicator (status icon or text indicator) if this widget is using custom status values.

Derived classes should overwrite this.

Definition at line 204 of file [YItemSelector.h](#).

#### 5.73.3.10 userInputProperty()

```
const char* YItemSelector::userInputProperty ( ) [inline], [virtual]
```

The name of the widget property that will return user input.

Inherited from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 187 of file [YItemSelector.h](#).

#### 5.73.3.11 usingCustomStatus()

```
bool YItemSelector::usingCustomStatus ( ) const
```

Return 'true' if this widget uses custom status values, 'false' if not (i.e.

only 0 or 1).

Definition at line 136 of file [YItemSelector.cc](#).



### 5.73.3.12 validCustomStatusIndex()

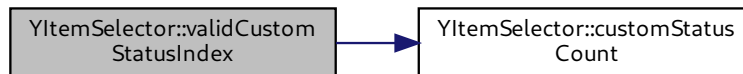
```
bool YItemSelector::validCustomStatusIndex (
    int index ) const
```

Return 'true' if a custom status index is within the valid range, i.e.

0..[customStatusCount\(\)](#)-1, 'false' otherwise.

Definition at line [161](#) of file [YItemSelector.cc](#).

Here is the call graph for this function:



### 5.73.3.13 visibleItems()

```
int YItemSelector::visibleItems ( ) const
```

Return the number of visible items (i.e.

items that are visible without scrolling). This is used to calculate the preferred height. If the widget gets more or less screen space than desired due to layout constraints, this number is not updated; this is purely the desired value for initializing layout negotiations.

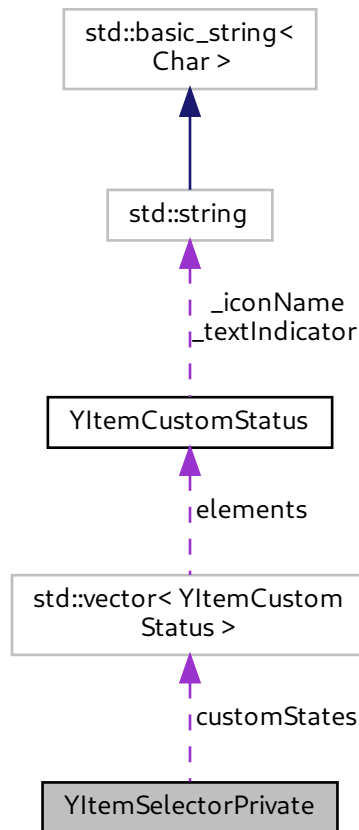
Definition at line [100](#) of file [YItemSelector.cc](#).

The documentation for this class was generated from the following files:

- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YItemSelector.h`
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YItemSelector.cc`

## 5.74 YItemSelectorPrivate Struct Reference

Collaboration diagram for YItemSelectorPrivate:



### Public Attributes

- int **visibleItems**
- YItemCustomStatusVector **customStates**

#### 5.74.1 Detailed Description

Definition at line 35 of file [YItemSelector.cc](#).

The documentation for this struct was generated from the following file:

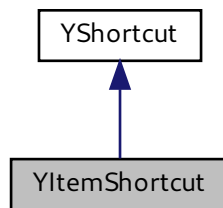
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YItemSelector.cc`

## 5.75 YItemShortcut Class Reference

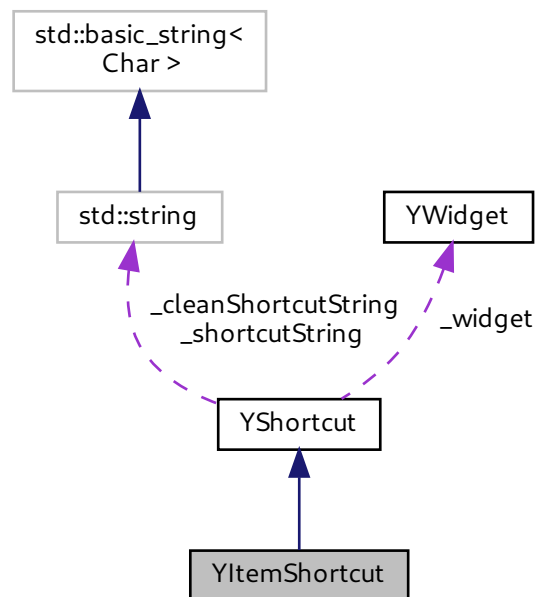
Special case for widgets that can have multiple shortcuts based on items (like [YDumbTab](#))

```
#include <YShortcut.h>
```

Inheritance diagram for YItemShortcut:



Collaboration diagram for YItemShortcut:



## Public Member Functions

- [YItemShortcut](#) ([YWidget](#) \*widget, [YItem](#) \*item)  
*Constructor.*
- virtual [~YItemShortcut](#) ()  
*Destructor.*
- [YItem](#) \* [item](#) () const  
*Return the associated item.*
- virtual void [setShortcut](#) (char newShortcut)  
*Set (override) the shortcut character.*

## Protected Member Functions

- virtual std::string [getShortcutString](#) ()  
*Obtain the the shortcut property of this shortcut's widget - the string that contains "&" to designate a shortcut.*

## Additional Inherited Members

### 5.75.1 Detailed Description

Special case for widgets that can have multiple shortcuts based on items (like [YDumbTab](#))

Definition at line [225](#) of file [YShortcut.h](#).

### 5.75.2 Member Function Documentation

#### 5.75.2.1 setShortcut()

```
void YItemShortcut::setShortcut (  
    char newShortcut ) [virtual]
```

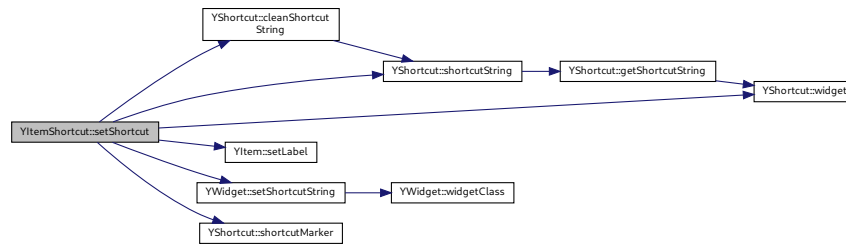
Set (override) the shortcut character.

In this subclass, it will change the internally stored item.

Reimplemented from [YShortcut](#).

Definition at line [322](#) of file [YShortcut.cc](#).

Here is the call graph for this function:

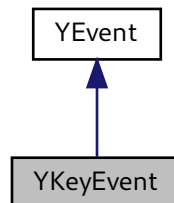


The documentation for this class was generated from the following files:

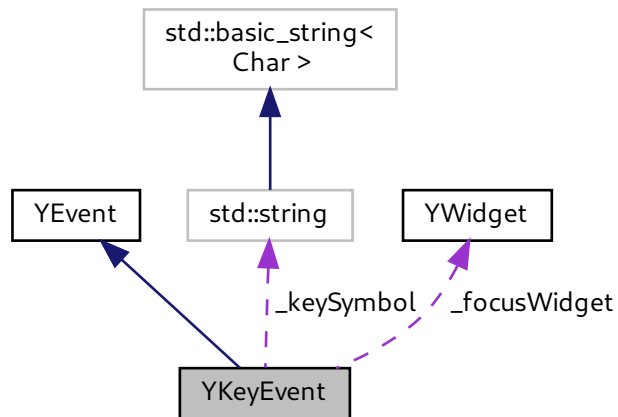
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YShortcut.h`
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YShortcut.cc`

## 5.76 YKeyEvent Class Reference

Inheritance diagram for YKeyEvent:



Collaboration diagram for YKeyEvent:



## Public Member Functions

- `YKeyEvent` (const std::string &keySymbol, YWidget \*focusWidget=0)  
*Constructor.*
- std::string keySymbol () const  
*Returns the key symbol - a text describing the key, such as "CursorLeft", "F1", "a", "A", etc.*
- YWidget \* focusWidget () const  
*Returns the widget that currently has the keyboard focus.*

## Protected Member Functions

- virtual ~YKeyEvent ()  
*Protected destructor - events can only be deleted via `YDialog::deleteEvent()`.*

## Protected Attributes

- std::string \_keySymbol
- YWidget \* \_focusWidget

## Additional Inherited Members

### 5.76.1 Detailed Description

Definition at line 206 of file `YEvent.h`.

## 5.76.2 Constructor & Destructor Documentation

### 5.76.2.1 YKeyEvent()

```
YKeyEvent::YKeyEvent (
    const std::string & keySymbol,
    YWidget * focusWidget = 0 )
```

Constructor.

Create a key event with a specified key symbol (a text describing the key, such as "CursorLeft", "F1", etc.) and optionally the widget that currently has the keyboard focus.

Definition at line 125 of file [YEvent.cc](#).

### 5.76.2.2 ~YKeyEvent()

```
virtual YKeyEvent::~~YKeyEvent ( ) [inline], [protected], [virtual]
```

Protected destructor - events can only be deleted via [YDialog::deleteEvent\(\)](#).

The associated dialog will take care of this event and delete it when appropriate.

Definition at line 241 of file [YEvent.h](#).

## 5.76.3 Member Function Documentation

### 5.76.3.1 focusWidget()

```
YWidget* YKeyEvent::focusWidget ( ) const [inline]
```

Returns the widget that currently has the keyboard focus.

This might be 0 if no widget has the focus or if the creator of this event could not obtain that information.

Definition at line 232 of file [YEvent.h](#).

The documentation for this class was generated from the following files:

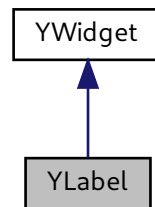
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YEvent.h
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YEvent.cc

## 5.77 YLabel Class Reference

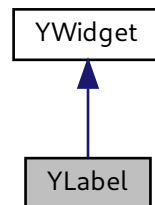
Implementation of the Label, Heading and OutputField widgets.

```
#include <YLabel.h>
```

Inheritance diagram for YLabel:



Collaboration diagram for YLabel:



### Public Member Functions

- `YLabel (YWidget *parent, const std::string &text, bool isHeading=false, bool isOutputField=false)`  
*Constructor.*
- `virtual ~YLabel ()`  
*Destructor.*
- `virtual const char * widgetClass () const`  
*Returns a descriptive name of this widget class for logging, debugging etc.*
- `std::string text () const`  
*Return the text the widget displays.*



- `std::string value () const`  
*Aliases for `text()`.*
- `std::string label () const`
- `virtual void setText (const std::string &newText)`  
*Set the text the widget displays.*
- `void setValue (const std::string &newValue)`  
*Aliases for `setText()`.*
- `void setLabel (const std::string &newLabel)`
- `bool isHeading () const`  
*Return 'true' if this is a Heading widget, i.e., it should display its text in a bold and/or larger font.*
- `bool isOutputField () const`  
*Return 'true' if this is an OutputField widget, i.e., it should display its text similar to an InputField the user can't change.*
- `bool useBoldFont () const`  
*Return 'true' if a bold font should be used.*
- `virtual void setUseBoldFont (bool bold=true)`  
*Switch bold font on or off.*
- `bool autoWrap () const`  
*Return 'true' if automatic word wrapping is enabled.*
- `virtual void setAutoWrap (bool autoWrap=true)`  
*Enable or disable automatic word wrapping.*
- `virtual bool setProperty (const std::string &propertyName, const YPropertyValue &val)`  
*Set a property.*
- `virtual YPropertyValue getProperty (const std::string &propertyName)`  
*Get a property.*
- `virtual const YPropertySet & propertySet ()`  
*Return this class's property set.*
- `virtual std::string debugLabel () const`  
*Returns a descriptive label of this widget instance for debugging.*

## Protected Member Functions

- `int layoutPass ()`  
*Convenience method for the parent dialog's `layoutPass()`: Return the number of the current layout pass.*

### 5.77.1 Detailed Description

Implementation of the Label, Heading and OutputField widgets.

Definition at line 38 of file [YLabel.h](#).

### 5.77.2 Constructor & Destructor Documentation

### 5.77.2.1 YLabel()

```
YLabel::YLabel (
    YWidget * parent,
    const std::string & text,
    bool isHeading = false,
    bool isOutputField = false )
```

Constructor.

'isHeading' indicates if this should be displayed as a Heading widget, i.e. with a bold and/or larger font. This cannot be changed after creating the widget.

'isOutputField' indicates if this should be displayed as an OutputField widget, i.e. similar to an InputField the user can't change. This cannot be changed after creating the widget.

Definition at line 61 of file [YLabel.cc](#).

## 5.77.3 Member Function Documentation

### 5.77.3.1 debugLabel()

```
string YLabel::debugLabel ( ) const [virtual]
```

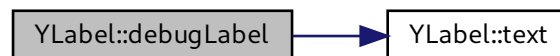
Returns a descriptive label of this widget instance for debugging.

Reimplemented from [YWidget](#) since a [YLabel](#) doesn't have a shortcut property.

Reimplemented from [YWidget](#).

Definition at line 200 of file [YLabel.cc](#).

Here is the call graph for this function:



### 5.77.3.2 getProperty()

```
YPropertyValue YLabel::getProperty (
    const std::string & propertyName ) [virtual]
```

Get a property.

Reimplemented from [YWidget](#).

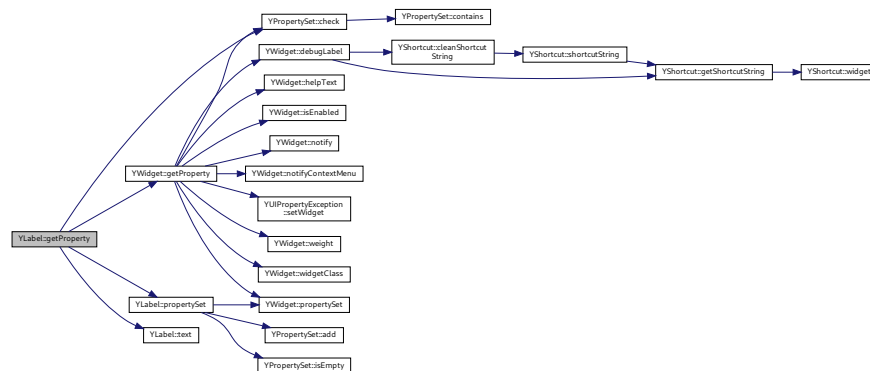
This method may throw exceptions, for example

- if there is no property with that name

Reimplemented from [YWidget](#).

Definition at line 186 of file [YLabel.cc](#).

Here is the call graph for this function:



### 5.77.3.3 isHeading()

```
bool YLabel::isHeading ( ) const
```

Return 'true' if this is a Heading widget, i.e., it should display its text in a bold and/or larger font.

This cannot be changed after creating the widget.

Definition at line 90 of file [YLabel.cc](#).

### 5.77.3.4 isOutputField()

```
bool YLabel::isOutputField ( ) const
```

Return 'true' if this is an OutputField widget, i.e., it should display its text similar to an InputField the user can't change.

This cannot be changed after creating the widget.

Definition at line 96 of file [YLabel.cc](#).

### 5.77.3.5 layoutPass()

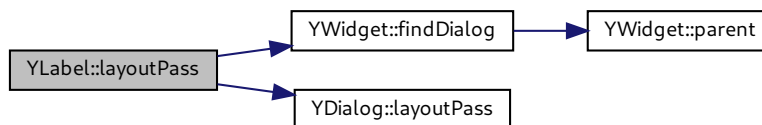
```
int YLabel::layoutPass ( ) [protected]
```

Convenience method for the parent dialog's [layoutPass\(\)](#): Return the number of the current layout pass.

0: No layout going on right now 1: First pass 2: Second pass of a multi-pass layout

Definition at line 137 of file [YLabel.cc](#).

Here is the call graph for this function:



### 5.77.3.6 propertySet()

```
const YPropertySet & YLabel::propertySet ( ) [virtual]
```

Return this class's property set.

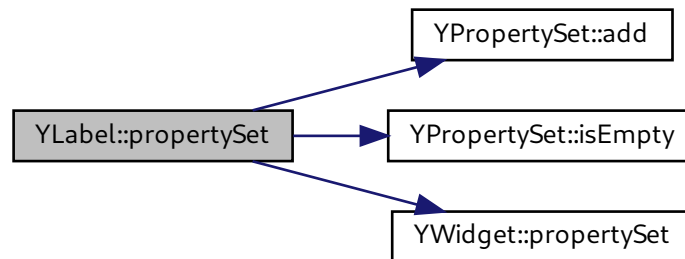
This also initializes the property set upon the first call.

Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 146 of file [YLabel.cc](#).

Here is the call graph for this function:



### 5.77.3.7 setAutoWrap()

```
void YLabel::setAutoWrap (
    bool autoWrap = true ) [virtual]
```

Enable or disable automatic word wrapping.

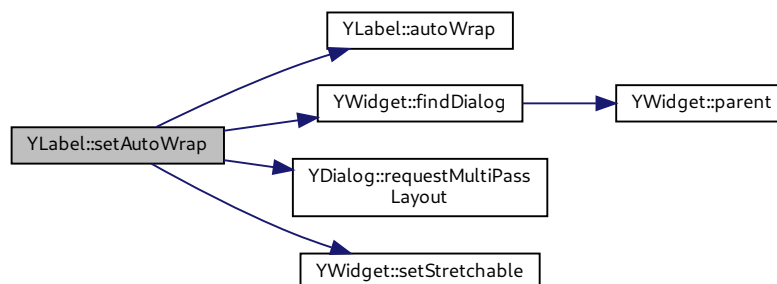
This has implications for geometry management: An auto-wrapping label does not have any reasonable preferred size; it needs to be put into a parent widget (like a `MinSize`) that enforces a reasonable width. The height can be then be calculated from that width.

Changing this setting takes only effect after the next layout geometry calculation.

Derived classes should overwrite this, but call this base class function in the overwritten function.

Definition at line 120 of file [YLabel.cc](#).

Here is the call graph for this function:



### 5.77.3.8 setProperty()

```
bool YLabel::setProperty (
    const std::string & propertyName,
    const YPropertyValue & val ) [virtual]
```

Set a property.

Reimplemented from [YWidget](#).

This method may throw exceptions, for example

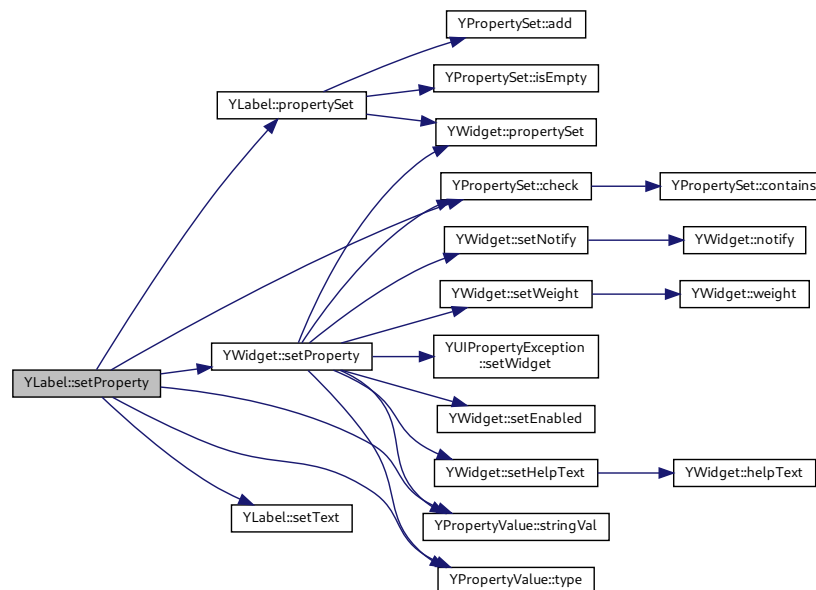
- if there is no property with that name
- if the expected type and the type mismatch
- if the value is out of range

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from [YWidget](#).

Definition at line 169 of file [YLabel.cc](#).

Here is the call graph for this function:



### 5.77.3.9 setText()

```
void YLabel::setText (
    const std::string & newText ) [virtual]
```

Set the text the widget displays.

Derived classes should overwrite this, but call this base class function in the overwritten function.

Definition at line 84 of file [YLabel.cc](#).

### 5.77.3.10 setUseBoldFont()

```
void YLabel::setUseBoldFont (
    bool bold = true ) [virtual]
```

Switch bold font on or off.

Derived classes should overwrite this, but call this base class function in the overwritten function.

Definition at line 108 of file [YLabel.cc](#).

### 5.77.3.11 widgetClass()

```
const char * YLabel::widgetClass ( ) const [virtual]
```

Returns a descriptive name of this widget class for logging, debugging etc.

Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

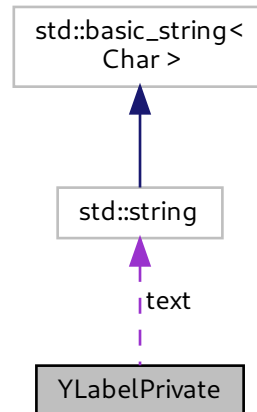
Definition at line 222 of file [YLabel.cc](#).

The documentation for this class was generated from the following files:

- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YLabel.h](#)
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YLabel.cc](#)

## 5.78 YLabelPrivate Struct Reference

Collaboration diagram for YLabelPrivate:



### Public Member Functions

- [YLabelPrivate](#) (const string &text, bool isHeading, bool isOutputField)  
*Constructor.*

### Public Attributes

- string **text**
- bool **isHeading**
- bool **isOutputField**
- bool **useBoldFont**
- bool **autoWrap**

### 5.78.1 Detailed Description

Definition at line 38 of file [YLabel.cc](#).

The documentation for this struct was generated from the following file:

- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YLabel.cc

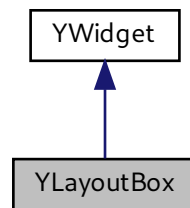


## 5.79 YLayoutBox Class Reference

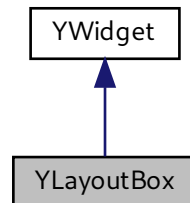
A vertical or horizontal stacking of widgets, implementing HBox and VBox.

```
#include <YLayoutBox.h>
```

Inheritance diagram for YLayoutBox:



Collaboration diagram for YLayoutBox:



### Public Types

- typedef std::vector< int > **sizeVector**
- typedef std::vector< int > **posVector**

## Public Member Functions

- virtual [~YLayoutBox](#) ()  
*Destructor.*
- virtual const char \* [widgetClass](#) () const  
*Returns a descriptive name of this widget class for logging, debugging etc.*
- YUIDimension [primary](#) () const  
*Return the primary dimension, i.e., the dimension this LayoutBox lays out its children in: YD\_VERT for a VBox, YD\_HORIZ for a HBox.*
- YUIDimension [secondary](#) () const  
*Return the secondary dimension.*
- bool [debugLayout](#) () const  
*Returns 'true' if layout debugging (verbose logging during layout) is on.*
- void [setDebugLayout](#) (bool deb=true)  
*Enable or disable layout debugging.*
- virtual int [preferredSize](#) (YUIDimension dim)  
*Preferred size of the widget in the specified dimension.*
- virtual int [preferredWidth](#) ()  
*Preferred width of the widget.*
- virtual int [preferredHeight](#) ()  
*Preferred height of the widget.*
- virtual void [setSize](#) (int newWidth, int newHeight)  
*Sets the size of the layout box.*
- virtual bool [stretchable](#) (YUIDimension dimension) const  
*Returns the stretchability of the layout box: The layout box is stretchable if one of the children is stretchable in this dimension or if one of the child widgets has a layout weight in this dimension.*
- virtual void [moveChild](#) (YWidget \*child, int newX, int newY)=0  
*Move a child to a new position.*

## Static Public Member Functions

- static bool [isLayoutStretch](#) (YWidget \*child, YUIDimension dimension)  
*Check if this is a layout stretch widget in the specified dimension, i.e.*

## Protected Member Functions

- [YLayoutBox](#) (YWidget \*parent, YUIDimension dim)  
*Constructor.*
- int [childrenTotalWeight](#) (YUIDimension dimension)  
*Add up all the children's weights.*
- int [childrenMaxPreferredSize](#) (YUIDimension dimension)  
*Return the maximum preferred size of all children in the specified dimension.*
- int [totalNonWeightedChildrenPreferredSize](#) (YUIDimension dimension)  
*Add up all the non-weighted children's preferred sizes in the specified dimension.*
- int [countNonWeightedChildren](#) (YUIDimension dimension)  
*Count the number of non-weighted children.*

- int [countStretchableChildren](#) (YUIDimension dimension)  
*Count the number of stretchable ( non-weighted ) children.*
- int [countLayoutStretchChildren](#) (YUIDimension dimension)  
*Count the number of "rubber bands", i.e.*
- [YWidget](#) \* [findDominatingChild](#) ()  
*Determine the number of the "dominating child" - the child widget that determines the overall size with respect to its weight.*
- void [calcPrimaryGeometry](#) (int newSize, sizeVector &childSize, posVector &childPos)  
*Calculate the sizes and positions of all children in the primary dimension and store them in "childSize" and "childPos".*
- void [calcSecondaryGeometry](#) (int newSize, sizeVector &childSize, posVector &childPos)  
*Calculate the sizes and positions of all children in the secondary dimension and store them in "childSize" and "childPos".*
- void [doResize](#) (sizeVector &width, sizeVector &height, posVector &x\_pos, posVector &y\_pos)  
*Actually perform resizing and moving the child widgets to the appropriate position.*

### 5.79.1 Detailed Description

A vertical or horizontal stacking of widgets, implementing HBox and VBox.

Definition at line 37 of file [YLayoutBox.h](#).

### 5.79.2 Constructor & Destructor Documentation

#### 5.79.2.1 YLayoutBox()

```
YLayoutBox::YLayoutBox (
    YWidget * parent,
    YUIDimension dim ) [protected]
```

Constructor.

Creates a VBox for dim == YD\_VERT or a HBox for YD\_HORIZ.

Definition at line 62 of file [YLayoutBox.cc](#).

Here is the call graph for this function:



### 5.79.3 Member Function Documentation

#### 5.79.3.1 countLayoutStretchChildren()

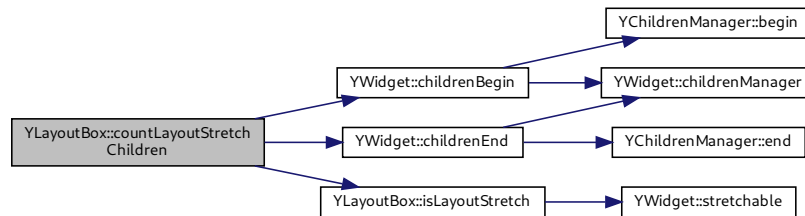
```
int YLayoutBox::countLayoutStretchChildren (
    YUIDimension dimension ) [protected]
```

Count the number of "rubber bands", i.e.

the number of stretchable layout spacings ( e.g. {H|V}Weight, {H|V}Spacing ). Only those without a weight are counted.

Definition at line 311 of file [YLayoutBox.cc](#).

Here is the call graph for this function:



#### 5.79.3.2 countStretchableChildren()

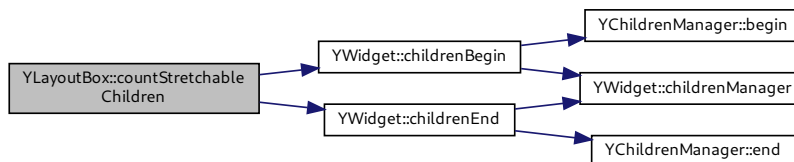
```
int YLayoutBox::countStretchableChildren (
    YUIDimension dimension ) [protected]
```

Count the number of stretchable ( non-weighted ) children.

Note: Weighted children are *always* considered stretchable.

Definition at line 293 of file [YLayoutBox.cc](#).

Here is the call graph for this function:



### 5.79.3.3 doResize()

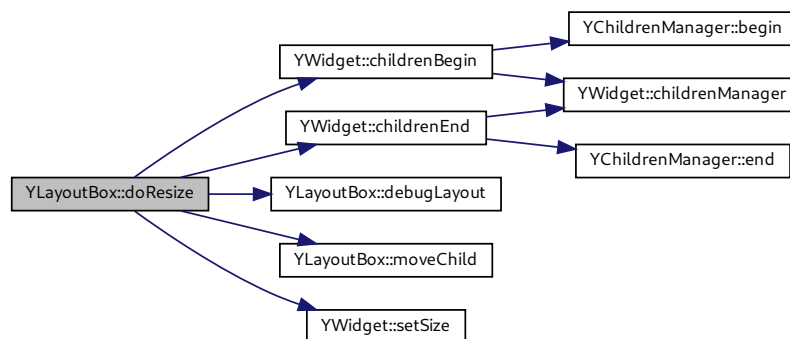
```
void YLayoutBox::doResize (
    sizeVector & width,
    sizeVector & height,
    posVector & x_pos,
    posVector & y_pos ) [protected]
```

Actually perform resizing and moving the child widgets to the appropriate position.

The vectors passed are the sizes previously calculated by [calcPrimaryGeometry\(\)](#) and [calcSecondaryGeometry\(\)](#).

Definition at line 742 of file [YLayoutBox.cc](#).

Here is the call graph for this function:



### 5.79.3.4 findDominatingChild()

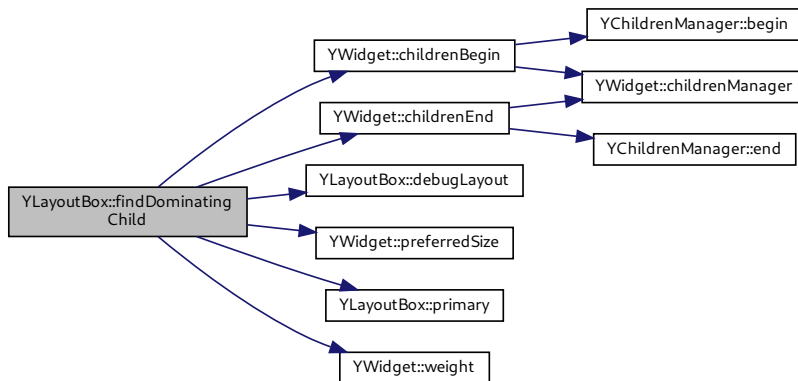
```
YWidget * YLayoutBox::findDominatingChild ( ) [protected]
```

Determine the number of the "dominating child" - the child widget that determines the overall size with respect to its weight.

Return 0 if there is no dominating child, i.e. none of the children has a weight specified.

Definition at line 181 of file [YLayoutBox.cc](#).

Here is the call graph for this function:



### 5.79.3.5 isLayoutStretch()

```

bool YLayoutBox::isLayoutStretch (
    YWidget * child,
    YUIDimension dimension ) [static]
  
```

Check if this is a layout stretch widget in the specified dimension, i.e.

an empty widget that is stretchable.

Definition at line 329 of file [YLayoutBox.cc](#).

Here is the call graph for this function:



### 5.79.3.6 moveChild()

```
virtual void YLayoutBox::moveChild (
    YWidget * child,
    int newX,
    int newY ) [pure virtual]
```

Move a child to a new position.

Derived classes are required to implement this.

### 5.79.3.7 preferredHeight()

```
int YLayoutBox::preferredHeight ( ) [virtual]
```

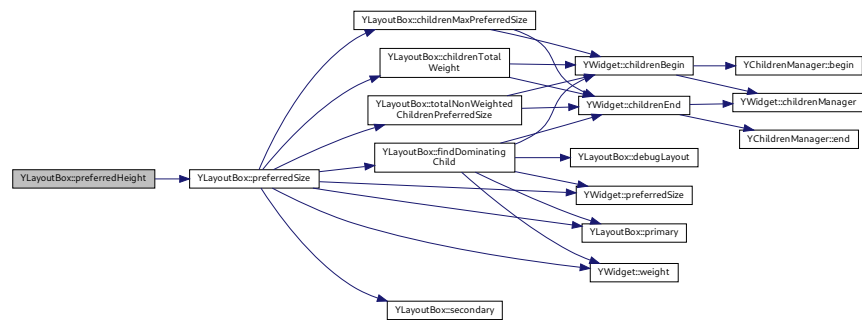
Preferred height of the widget.

Reimplemented from [YWidget](#).

Implements [YWidget](#).

Definition at line 161 of file [YLayoutBox.cc](#).

Here is the call graph for this function:



### 5.79.3.8 preferredSize()

```
int YLayoutBox::preferredSize (
    YUIDimension dim ) [virtual]
```

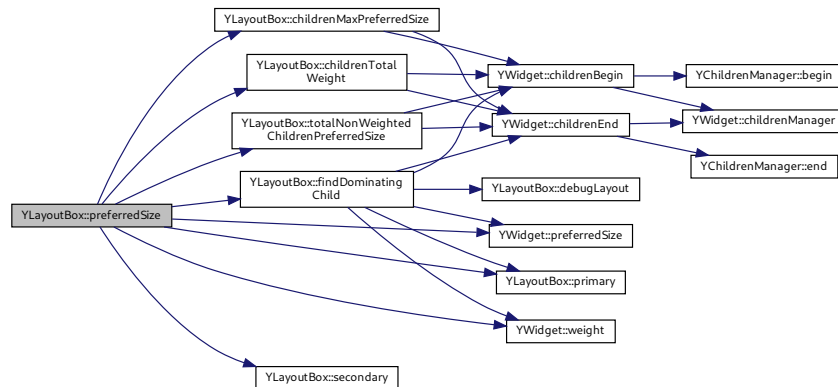
Preferred size of the widget in the specified dimension.

Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 107 of file [YLayoutBox.cc](#).

Here is the call graph for this function:



### 5.79.3.9 preferredWidth()

```
int YLayoutBox::preferredWidth ( ) [virtual]
```

Preferred width of the widget.

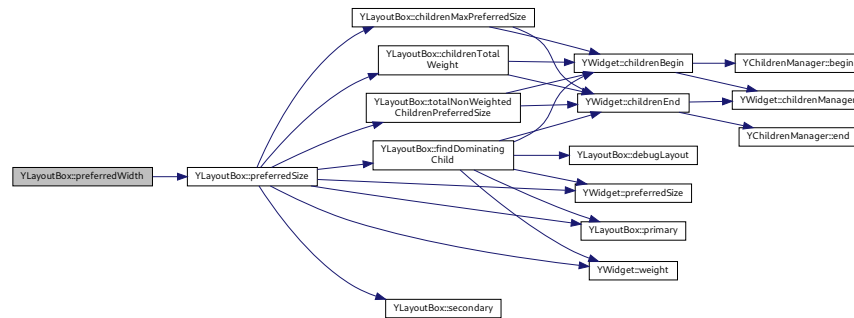
Reimplemented from [YWidget](#).

Implements [YWidget](#).

Definition at line 155 of file [YLayoutBox.cc](#).



Here is the call graph for this function:



#### 5.79.3.10 setSize()

```
void YLayoutBox::setSize (
    int newWidth,
    int newHeight ) [virtual]
```

Sets the size of the layout box.

This is where the layout policy is implemented.

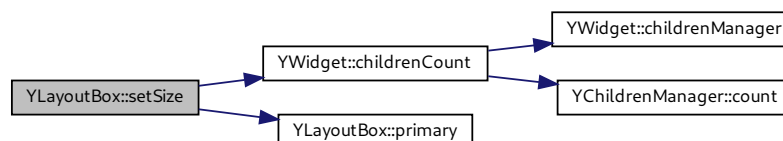
Derived classes can reimplement this, but this base class method should be called in the reimplemented function.

Reimplemented from [YWidget](#).

Implements [YWidget](#).

Definition at line 361 of file [YLayoutBox.cc](#).

Here is the call graph for this function:



### 5.79.3.11 stretchable()

```
bool YLayoutBox::stretchable (
    YUIDimension dimension ) const [virtual]
```

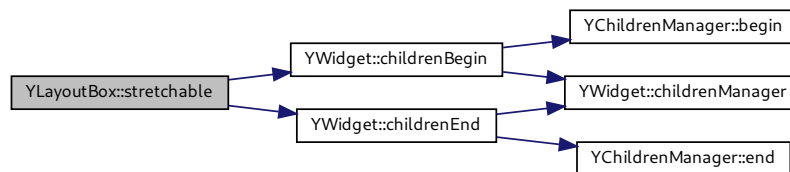
Returns the stretchability of the layout box: The layout box is stretchable if one of the children is stretchable in this dimension or if one of the child widgets has a layout weight in this dimension.

Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 345 of file [YLayoutBox.cc](#).

Here is the call graph for this function:



The documentation for this class was generated from the following files:

- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YLayoutBox.h`
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YLayoutBox.cc`

## 5.80 YLayoutBoxPrivate Struct Reference

### Public Member Functions

- [YLayoutBoxPrivate](#) (YUIDimension prim)  
*Constructor.*

### Public Attributes

- YUIDimension **primary**
- YUIDimension **secondary**
- bool **debugLayout**

### 5.80.1 Detailed Description

Definition at line 39 of file [YLayoutBox.cc](#).

The documentation for this struct was generated from the following file:

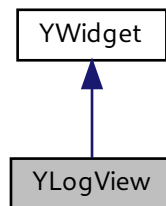
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YLayoutBox.cc](#)

## 5.81 YLogView Class Reference

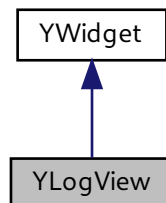
**LogView**: A scrollable (output-only) text to display a growing log, very much like the "tail -f" shell command.

```
#include <YLogView.h>
```

Inheritance diagram for YLogView:



Collaboration diagram for YLogView:



## Public Member Functions

- virtual [~YLogView](#) ()  
*Destructor.*
- virtual const char \* [widgetClass](#) () const  
*Returns a descriptive name of this widget class for logging, debugging etc.*
- std::string [label](#) () const  
*Return the label (the caption above the log text).*
- virtual void [setLabel](#) (const std::string &label)  
*Set the label (the caption above the log text).*
- int [visibleLines](#) () const  
*Return the number of visible lines.*
- void [setVisibleLines](#) (int newVisibleLines)  
*Set the number of visible lines.*
- int [maxLines](#) () const  
*Return the maximum number of lines to store.*
- void [setMaxLines](#) (int newMaxLines)  
*Set the maximum number of lines to store.*
- std::string [logText](#) () const  
*Return the entire log text as one large string of concatenated lines delimited with newlines.*
- void [setLogText](#) (const std::string &text)  
*Set (replace) the entire log text and trigger a display update.*
- std::string [lastLine](#) () const  
*Return the last log line.*
- void [appendLines](#) (const std::string &text)  
*Append one or more lines to the log text and trigger a display update.*
- void [clearText](#) ()  
*Clear the log text and trigger a display update.*
- int [lines](#) () const  
*Return the current number of lines.*
- virtual bool [setProperty](#) (const std::string &propertyName, const [YPropertyValue](#) &val)  
*Set a property.*
- virtual [YPropertyValue](#) [getProperty](#) (const std::string &propertyName)  
*Get a property.*
- virtual const [YPropertySet](#) & [propertySet](#) ()  
*Return this class's property set.*
- virtual std::string [shortcutString](#) () const  
*Get the string of this widget that holds the keyboard shortcut.*
- virtual void [setShortcutString](#) (const std::string &str)  
*Set the string of this widget that holds the keyboard shortcut.*

## Protected Member Functions

- [YLogView](#) ([YWidget](#) \*parent, const std::string &label, int visibleLines, int maxLines)  
*Constructor.*
- virtual void [displayLogText](#) (const std::string &text)=0  
*Display the part of the log text that should be displayed.*

### 5.81.1 Detailed Description

LogView: A scrollable (output-only) text to display a growing log, very much like the "tail -f" shell command.

Definition at line 37 of file [YLogView.h](#).

### 5.81.2 Constructor & Destructor Documentation

#### 5.81.2.1 YLogView()

```
YLogView::YLogView (
    YWidget * parent,
    const std::string & label,
    int visibleLines,
    int maxLines ) [protected]
```

Constructor.

'label' is the caption above the log. 'visibleLines' indicates how many lines should be visible by default (unless changed by other layout constraints), 'maxLines' specifies how many lines (always the last ones) to keep in the log. 0 for 'maxLines' means "keep all lines".

Definition at line 59 of file [YLogView.cc](#).

Here is the call graph for this function:



### 5.81.3 Member Function Documentation

#### 5.81.3.1 displayLogText()

```
virtual void YLogView::displayLogText (
    const std::string & text ) [protected], [pure virtual]
```

Display the part of the log text that should be displayed.

'text' contains the last '[visibleLines\(\)](#)' lines. This is called whenever the log text changes. Note that the text might also be empty, in which case the displayed log text should be cleared.

Derived classes are required to implement this.

### 5.81.3.2 getProperty()

```
YPropertyValue YLogView::getProperty (
    const std::string & propertyName ) [virtual]
```

Get a property.

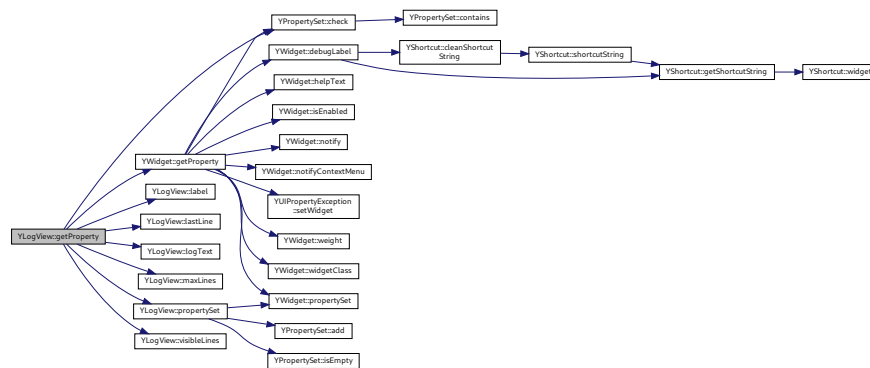
Reimplemented from [YWidget](#).

This method may throw [YUIPropertyExceptions](#).

Reimplemented from [YWidget](#).

Definition at line 286 of file [YLogView.cc](#).

Here is the call graph for this function:



### 5.81.3.3 maxLines()

```
int YLogView::maxLines ( ) const
```

Return the maximum number of lines to store.

The last [maxLines\(\)](#) lines of the log text will be kept.

Definition at line 105 of file [YLogView.cc](#).

#### 5.81.3.4 propertySet()

```
const YPropertySet & YLogView::propertySet ( ) [virtual]
```

Return this class's property set.

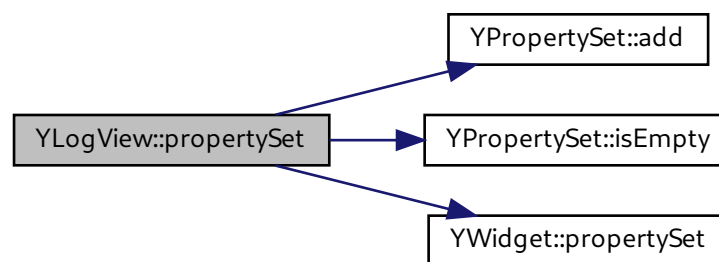
This also initializes the property upon the first call.

Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 241 of file [YLogView.cc](#).

Here is the call graph for this function:



#### 5.81.3.5 setLabel()

```
void YLogView::setLabel (
    const std::string & label ) [virtual]
```

Set the label (the caption above the log text).

Derived classes are free to reimplement this, but they should call this base class method at the end of the overloaded function.

Definition at line 84 of file [YLogView.cc](#).

Here is the call graph for this function:



### 5.81.3.6 setMaxLines()

```
void YLogView::setMaxLines (
    int newMaxLines )
```

Set the maximum number of lines to store.

"0" means "keep all lines" (beware of memory overflow!).

If the new value is lower than the old value, any (now) excess lines before the last 'newMaxLines' lines of the log text is cut off and a display update is triggered.

This method is intentionally not virtual since a display update is triggered when appropriate.

Definition at line 112 of file [YLogView.cc](#).

### 5.81.3.7 setProperty()

```
bool YLogView::setProperty (
    const std::string & propertyName,
    const YPropertyValue & val ) [virtual]
```

Set a property.

Reimplemented from [YWidget](#).

This function may throw YUIPropertyExceptions.

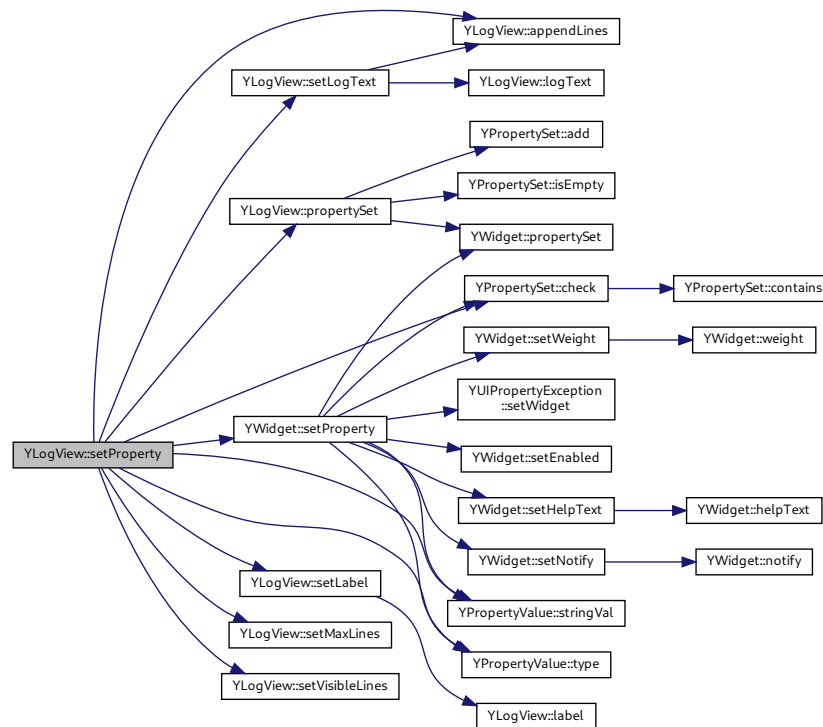
This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from [YWidget](#).

Definition at line 267 of file [YLogView.cc](#).



Here is the call graph for this function:



### 5.81.3.8 setShortcutString()

```
virtual void YLogView::setShortcutString (
    const std::string & str ) [inline], [virtual]
```

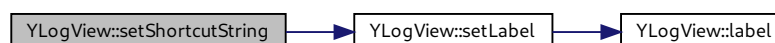
Set the string of this widget that holds the keyboard shortcut.

Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 185 of file [YLogView.h](#).

Here is the call graph for this function:



### 5.81.3.9 setVisibleLines()

```
void YLogView::setVisibleLines (
    int newVisibleLines )
```

Set the number of visible lines.

Changing this has only effect upon the next geometry call, so applications calling this function might want to trigger a re-layout afterwards.

This method is intentionally not virtual: [visibleLines\(\)](#) should be queried in the [preferredHeight\(\)](#) implementation.

Definition at line 98 of file [YLogView.cc](#).

### 5.81.3.10 shortcutString()

```
virtual std::string YLogView::shortcutString ( ) const [inline], [virtual]
```

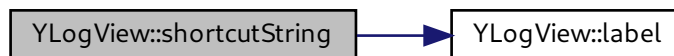
Get the string of this widget that holds the keyboard shortcut.

Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 178 of file [YLogView.h](#).

Here is the call graph for this function:

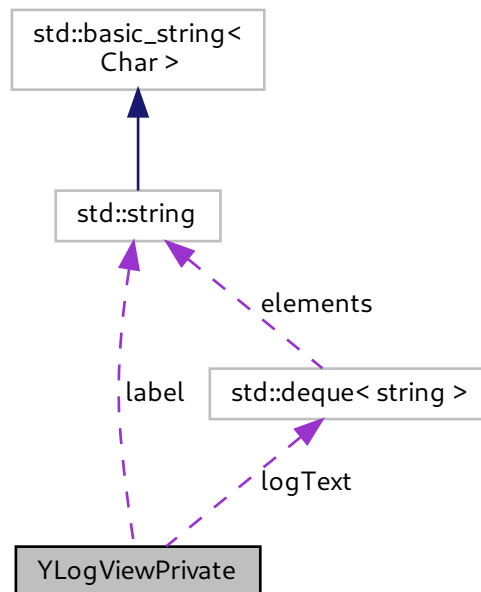


The documentation for this class was generated from the following files:

- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YLogView.h](#)
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YLogView.cc](#)

## 5.82 YLogViewPrivate Struct Reference

Collaboration diagram for YLogViewPrivate:



### Public Member Functions

- **YLogViewPrivate** (const string &label, int visibleLines, int maxLines)

### Public Attributes

- string **label**
- int **visibleLines**
- int **maxLines**
- StringDeque **logText**

#### 5.82.1 Detailed Description

Definition at line 41 of file [YLogView.cc](#).

The documentation for this struct was generated from the following file:

- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YLogView.cc

## 5.83 YMacro Class Reference

Simple access to macro recording and playing.

```
#include <YMacro.h>
```

### Static Public Member Functions

- static void [setRecorder](#) ([YMacroRecorder](#) \*recorder)  
*Set a macro recorder.*
- static void [setPlayer](#) ([YMacroPlayer](#) \*player)  
*Set a macro player.*
- static void [record](#) (const std::string &macroFile)  
*Record a macro to the specified macro file.*
- static void [endRecording](#) ()  
*End macro recording.*
- static bool [recording](#) ()  
*Return 'true' if a macro is currently being recorded.*
- static void [play](#) (const std::string &macroFile)  
*Play a macro from the specified macro file.*
- static void [playNextBlock](#) ()  
*Play the next block from the current macro, if there is one playing.*
- static bool [playing](#) ()  
*Return 'true' if a macro is currently being played.*
- static [YMacroRecorder](#) \* [recorder](#) ()  
*Return the current macro recorder or 0 if there is none.*
- static [YMacroPlayer](#) \* [player](#) ()  
*Return the current macro player or 0 if there is none.*
- static void [deleteRecorder](#) ()  
*Delete the current macro recorder if there is one.*
- static void [deletePlayer](#) ()  
*Delete the current macro player if there is one.*

### 5.83.1 Detailed Description

Simple access to macro recording and playing.

This class stores an instance of a macro recorder and a macro player. Since both [YMacroRecorder](#) and [YMacroPlayer](#) are abstract base classes, derived classes from either of them have to be instantiated and set ([setRecorder\(\)](#), [setPlayer\(\)](#)) from the outside for anything to happen. Until that point, none of the macro operations here do anything (but also don't throw any error or exception).

Definition at line 44 of file [YMacro.h](#).

## 5.83.2 Member Function Documentation

### 5.83.2.1 setPlayer()

```
void YMacro::setPlayer (
    YMacroPlayer * player ) [static]
```

Set a macro player.

This needs to be done from the outside since [YMacroRecorder](#) is an abstract base class, i.e., it needs to be derived to be instantiated.

Definition at line 46 of file [YMacro.cc](#).

Here is the call graph for this function:



### 5.83.2.2 setRecorder()

```
void YMacro::setRecorder (
    YMacroRecorder * recorder ) [static]
```

Set a macro recorder.

This needs to be done from the outside since [YMacroRecorder](#) is an abstract base class, i.e., it needs to be derived to be instantiated.

Definition at line 37 of file [YMacro.cc](#).

Here is the call graph for this function:



The documentation for this class was generated from the following files:

- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YMacro.h`
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YMacro.cc`

## 5.84 YMacroPlayer Class Reference

Abstract base class for macro player.

```
#include <YMacroPlayer.h>
```

### Public Member Functions

- virtual [~YMacroPlayer](#) ()  
*Destructor.*
- virtual void [play](#) (const std::string &macroFile)=0  
*Play a macro from the specified macro file.*
- virtual void [playNextBlock](#) ()=0  
*Play the next block from the current macro, if there is one playing.*
- virtual bool [playing](#) () const =0  
*Return 'true' if a macro is currently being played.*

### Protected Member Functions

- [YMacroPlayer](#) ()  
*Constructor.*

### Friends

- class **YMacro**

#### 5.84.1 Detailed Description

Abstract base class for macro player.

Applications should not use this directly, but the static methods in [YMacro](#).

Definition at line 35 of file [YMacroPlayer.h](#).

The documentation for this class was generated from the following file:

- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YMacroPlayer.h

## 5.85 YMacroRecorder Class Reference

Abstract base class for macro recorders.

```
#include <YMacroRecorder.h>
```

## Public Member Functions

- virtual `~YMacroRecorder()`  
*Destructor.*
- virtual void `record` (const std::string &macroFileName)=0  
*Start recording a macro to the specified file.*
- virtual void `endRecording` ()=0  
*End recording and close the current macro file (if there is any).*
- virtual bool `recording` () const =0  
*Return 'true' if a macro is currently being recorded.*
- virtual void `recordWidgetProperty` (YWidget \*widget, const char \*propertyName)=0  
*Record one widget property.*
- virtual void `recordMakeScreenShot` (bool enabled=false, const std::string &filename=std::string())=0  
*Record a "UI::MakeScreenShot()" statement.*

## Protected Member Functions

- `YMacroRecorder()`  
*Constructor.*

## Friends

- class `YMacro`

### 5.85.1 Detailed Description

Abstract base class for macro recorders.

Applications should not use this directly, but the static methods in `YMacro`.

Definition at line 38 of file `YMacroRecorder.h`.

### 5.85.2 Member Function Documentation

#### 5.85.2.1 recordMakeScreenShot()

```
virtual void YMacroRecorder::recordMakeScreenShot (
    bool enabled = false,
    const std::string & filename = std::string() ) [pure virtual]
```

Record a "UI::MakeScreenShot()" statement.

If 'enabled' is 'false', this statement will be commented out. If no file name is given, a default file name (with auto-increment) will be used.

The documentation for this class was generated from the following file:

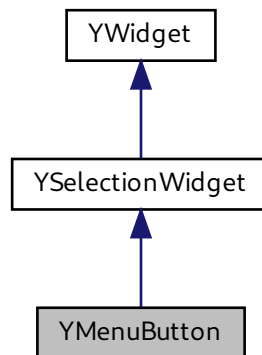
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YMacroRecorder.h`

## 5.86 YMenuButton Class Reference

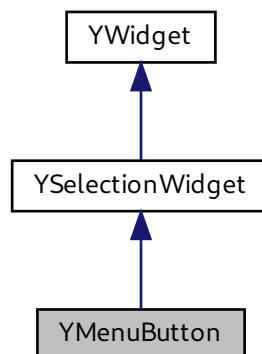
MenuButton: Similar to PushButton, but with several actions: Upon clicking on a MenuButton (or activating it with the keyboard), a pop-up menu opens where the user can activate an action.

```
#include <YMenuButton.h>
```

Inheritance diagram for YMenuButton:



Collaboration diagram for YMenuButton:





## Public Member Functions

- virtual `~YMenuButton ()`  
*Destructor.*
- virtual const char \* `widgetClass ()` const  
*Returns a descriptive name of this widget class for logging, debugging etc.*
- virtual void `rebuildMenuTree ()`=0  
*Rebuild the displayed menu tree from the internally stored YMenuItems.*
- virtual void `addItems` (const `YItemCollection` &itemCollection)  
*Add multiple items.*
- virtual void `addItem` (`YItem` \*item\_disown)  
*Add one item.*
- virtual void `deleteAllItems ()`  
*Delete all items.*
- void `resolveShortcutConflicts ()`  
*Resolve keyboard shortcut conflicts: Change shortcuts of menu items if there are duplicates in the respective menu level.*
- virtual bool `setProperty` (const std::string &propertyName, const `YPropertyValue` &val)  
*Set a property.*
- virtual `YPropertyValue` `getProperty` (const std::string &propertyName)  
*Get a property.*
- virtual const `YPropertySet` & `propertySet ()`  
*Return this class's property set.*
- `YMenuItem` \* `findItem` (std::vector< std::string > &path) const  
*Return item in the tree which matches path of labels or nullptr in case no item with such label was found and is a leaf, as other nodes do not trigger actions except showing children items.*
- virtual void `activateItem` (`YMenuItem` \*item)=0  
*Activate the item selected in the tree.*
- `YMenuItem` \* `findMenuItem` (int index)  
*Recursively find the first menu item with the specified index.*

## Protected Member Functions

- `YMenuButton` (`YWidget` \*parent, const std::string &label)  
*Constructor.*
- `YMenuItem` \* `findMenuItem` (int index, `YItemConstIterator` begin, `YItemConstIterator` end)  
*Recursively find the first menu item with the specified index from iterator 'begin' to iterator 'end'.*
- `YMenuItem` \* `findItem` (std::vector< std::string >::iterator path\_begin, std::vector< std::string >::iterator path\_end, `YItemConstIterator` begin, `YItemConstIterator` end) const  
*Recursively looks for the first item in the tree of the menu items using depth first search.*
- `YMenuItem` \* `itemAt` (int index)  
*Alias for `findMenuItem()`.*

### 5.86.1 Detailed Description

MenuButton: Similar to PushButton, but with several actions: Upon clicking on a MenuButton (or activating it with the keyboard), a pop-up menu opens where the user can activate an action.

Menu items in that pop-up menu can have submenus (that will pop up in separate pop-up menus).

Internally, this widget is more similar to the Tree widget. The difference is that it does not keep a "selected" status, but triggers an action right away, just like a PushButton. Like PushButton, MenuButton sends an event right away when the user selects an item (clicks on a menu item or activates it with the keyboard). Items that have a submenu never send an event, they simply open their submenu when activated.

Definition at line 48 of file [YMenuButton.h](#).

### 5.86.2 Constructor & Destructor Documentation

#### 5.86.2.1 YMenuButton()

```
YMenuButton::YMenuButton (
    YWidget * parent,
    const std::string & label ) [protected]
```

Constructor.

'label' is the user-visible text on the button (not above it like all other SelectionWidgets).

Definition at line 49 of file [YMenuButton.cc](#).

### 5.86.3 Member Function Documentation

#### 5.86.3.1 activateItem()

```
virtual void YMenuButton::activateItem (
    YMenuItem * item ) [pure virtual]
```

Activate the item selected in the tree.

Can be used in tests to simulate user input.

Derived classes are required to implement this.

### 5.86.3.2 addItem()

```
void YMenuButton::addItem (
    YItem * item_disown ) [virtual]
```

Add one item.

This widget assumes ownership of the item object and will delete it in its destructor.

This reimplement will an index to the item that is unique for all items in this MenuButton. That index can be used later with [findMenuItem\(\)](#) to find the item by that index.

#### Note

please do not forget to call after adding all elements [resolveShortcutConflicts](#) and [rebuildMenuTree](#) in this order. It is important to call it after all submenus are added otherwise it won't have proper shortcuts and won't be rendered.

#### See also

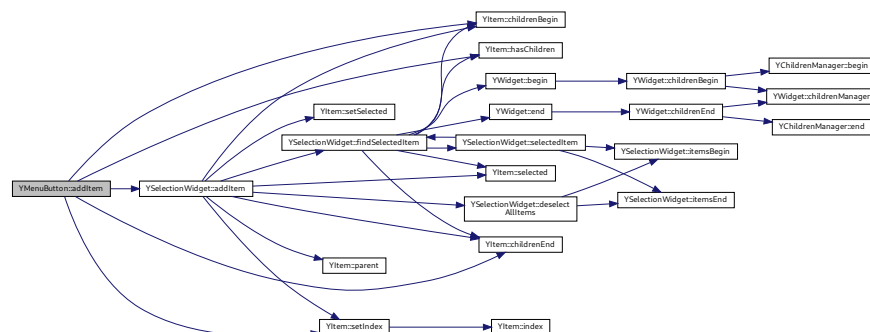
[examples/MenuButton.cc](#).

Reimplemented from [YSelectionWidget](#).

Reimplemented from [YSelectionWidget](#).

Definition at line 74 of file [YMenuButton.cc](#).

Here is the call graph for this function:





### 5.86.3.5 findItem() [1/2]

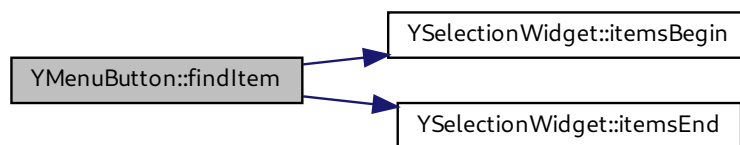
```
YMenuItem * YMenuButton::findItem (
    std::vector< std::string > & path ) const
```

Return item in the tree which matches path of labels or nullptr in case no item with such label was found and is a leaf, as other nodes do not trigger actions except showing children items.

Accepts vector of strings which denote path to the node.

Definition at line 270 of file [YMenuButton.cc](#).

Here is the call graph for this function:



### 5.86.3.6 findItem() [2/2]

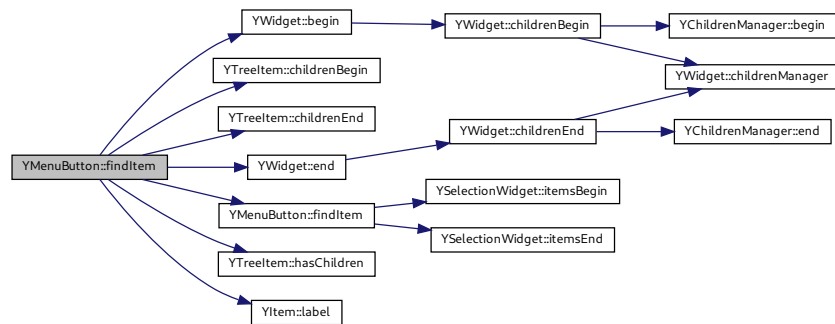
```
YMenuItem * YMenuButton::findItem (
    std::vector< std::string >::iterator path_begin,
    std::vector< std::string >::iterator path_end,
    YItemConstIterator begin,
    YItemConstIterator end ) const [protected]
```

Recursively looks for the first item in the tree of the menu items using depth first search.

Return nullptr if item which matches full path is not found. Path is a vector of strings, where next element is a child item, so in case one needs to select File->Export->As PDF, for instance, Vector will look like [ "File", "Export", "As PDF" ].

Definition at line 276 of file [YMenuButton.cc](#).

Here is the call graph for this function:



#### 5.86.3.7 findMenuItem() [1/2]

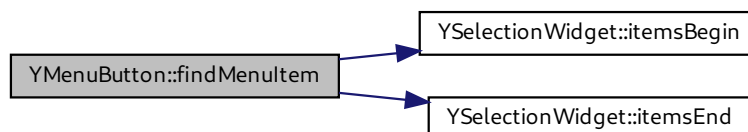
```
YMenuItem * YMenuItem::findMenuItem (
    int index )
```

Recursively find the first menu item with the specified index.

Returns 0 if there is no such item.

Definition at line 108 of file [YMenuItem.cc](#).

Here is the call graph for this function:



### 5.86.3.8 findMenuItem() [2/2]

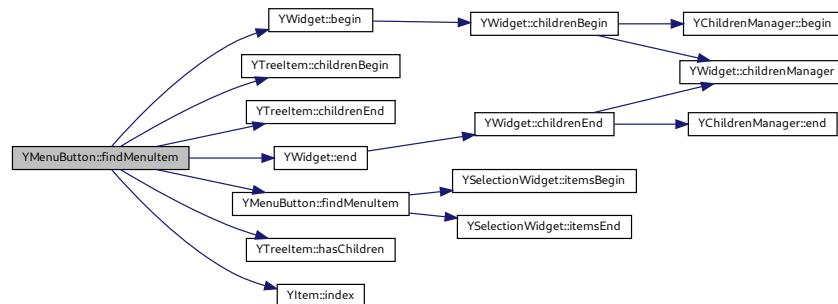
```
YMenuItem * YMenuButton::findMenuItem (
    int index,
    YItemConstIterator begin,
    YItemConstIterator end ) [protected]
```

Recursively find the first menu item with the specified index from iterator 'begin' to iterator 'end'.

Returns 0 if there is no such item.

Definition at line 115 of file [YMenuButton.cc](#).

Here is the call graph for this function:



### 5.86.3.9 getProperty()

```
YPropertyValue YMenuButton::getProperty (
    const std::string & propertyName ) [virtual]
```

Get a property.

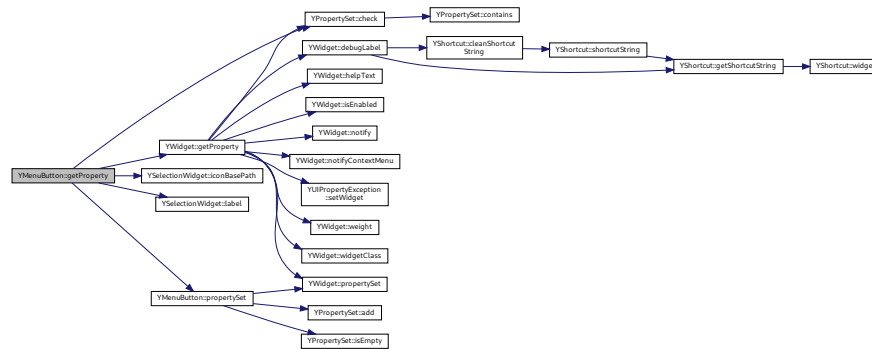
Reimplemented from [YWidget](#).

This method may throw `YUIPropertyExceptions`.

Reimplemented from [YWidget](#).

Definition at line 255 of file [YMenuButton.cc](#).

Here is the call graph for this function:



### 5.86.3.10 itemAt()

```
YMenuItem* YMenuBar::itemAt (
    int index ) [inline], [protected]
```

Alias for [findMenuItem\(\)](#).

Reimplemented to ensure consistent behaviour with [YSelectionWidget::itemAt\(\)](#).

Definition at line 204 of file [YMenuBar.h](#).

Here is the call graph for this function:





### 5.86.3.11 propertySet()

```
const YPropertySet & YMenuButton::propertySet ( ) [virtual]
```

Return this class's property set.

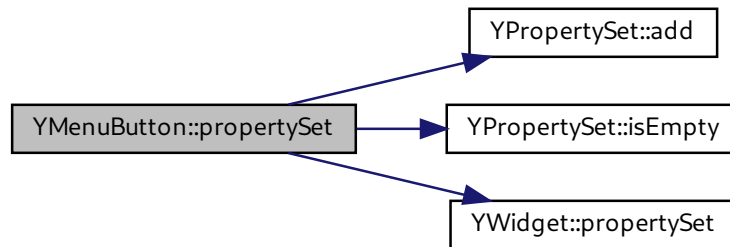
This also initializes the property upon the first call.

Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 216 of file [YMenuButton.cc](#).

Here is the call graph for this function:



### 5.86.3.12 rebuildMenuTree()

```
virtual void YMenuButton::rebuildMenuTree ( ) [pure virtual]
```

Rebuild the displayed menu tree from the internally stored YMenuItems.

The application should call this (once) after all items have been added with [addItem\(\)](#). [YMenuButton::addItem\(\)](#) calls this automatically.

Derived classes are required to implement this.

### 5.86.3.13 resolveShortcutConflicts()

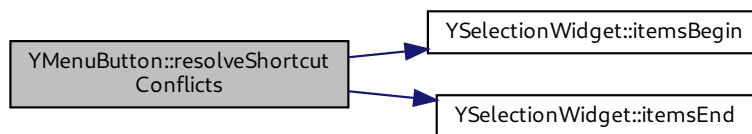
```
void YMenuButton::resolveShortcutConflicts ( )
```

Resolve keyboard shortcut conflicts: Change shortcuts of menu items if there are duplicates in the respective menu level.

This has to be called after all items are added, but before [rebuildMenuTree\(\)](#) (see above). [YMenuButton::addItem\(\)](#) calls this automatically.

Definition at line [209](#) of file [YMenuButton.cc](#).

Here is the call graph for this function:



### 5.86.3.14 setProperty()

```
bool YMenuButton::setProperty (
    const std::string & propertyName,
    const YPropertyValue & val ) [virtual]
```

Set a property.

Reimplemented from [YWidget](#).

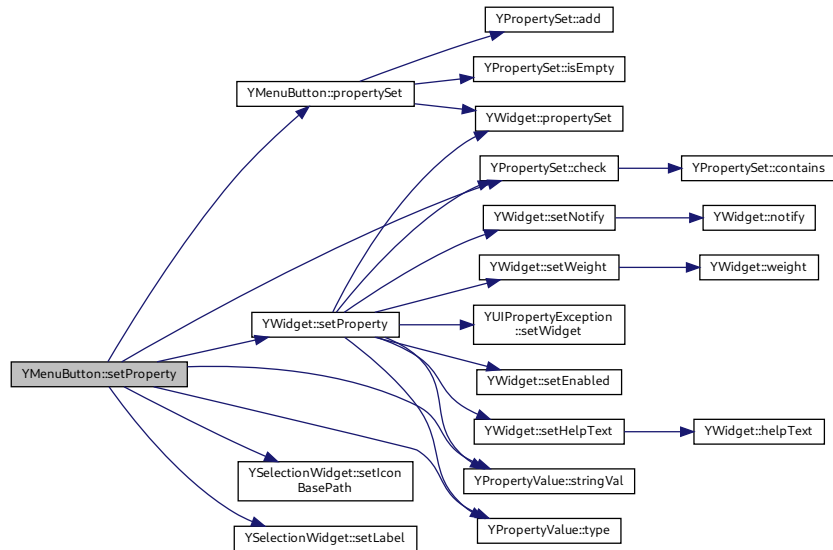
This function may throw `YUIPropertyExceptions`.

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from [YWidget](#).

Definition at line [238](#) of file [YMenuButton.cc](#).

Here is the call graph for this function:



The documentation for this class was generated from the following files:

- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YMenuBar.h`
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YMenuBar.cc`

## 5.87 YMenuBarPrivate Struct Reference

### Public Attributes

- `int nextSerialNo`

### 5.87.1 Detailed Description

Definition at line 37 of file [YMenuBar.cc](#).

The documentation for this struct was generated from the following file:

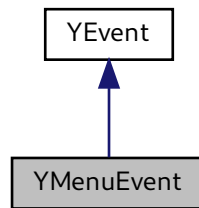
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YMenuBar.cc`

## 5.88 YMenuEvent Class Reference

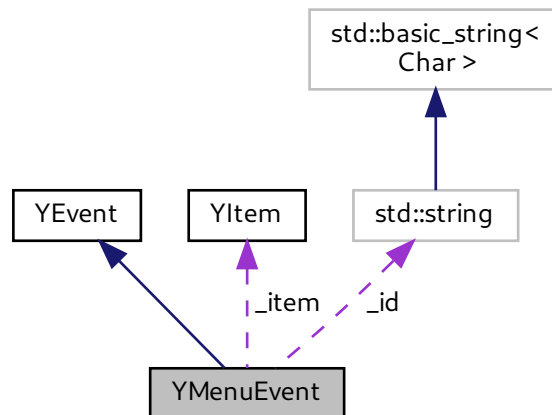
Event to be returned upon menu selection.

```
#include <YEvent.h>
```

Inheritance diagram for YMenuEvent:



Collaboration diagram for YMenuEvent:



### Public Member Functions

- **YMenuEvent** ([YItem](#) \*item)
- **YMenuEvent** (const char \*id)

- **YMenuEvent** (const std::string &id)
- virtual [YItem](#) \* [item](#) () const  
*Return the [YItem](#) that corresponds to this event or 0 if the event was constructed with a string ID.*
- std::string [id](#) () const  
*Return the string ID of this event.*

## Protected Member Functions

- virtual [~YMenuEvent](#) ()  
*Protected destructor - events can only be deleted via [YDialog::deleteEvent\(\)](#).*

## Protected Attributes

- [YItem](#) \* [\\_item](#)
- std::string [\\_id](#)

## Additional Inherited Members

### 5.88.1 Detailed Description

Event to be returned upon menu selection.

Definition at line 256 of file [YEvent.h](#).

### 5.88.2 Constructor & Destructor Documentation

#### 5.88.2.1 [~YMenuEvent\(\)](#)

```
virtual YMenuEvent::~YMenuEvent ( ) [inline], [protected], [virtual]
```

Protected destructor - events can only be deleted via [YDialog::deleteEvent\(\)](#).

The associated dialog will take care of this event and delete it when appropriate.

Definition at line 289 of file [YEvent.h](#).

### 5.88.3 Member Function Documentation

### 5.88.3.1 id()

```
std::string YMenuEvent::id ( ) const [inline]
```

Return the string ID of this event.

This will be an empty string if the event was constructed with a [YItem](#).

Definition at line 280 of file [YEvent.h](#).

### 5.88.3.2 item()

```
virtual YItem* YMenuEvent::item ( ) const [inline], [virtual]
```

Return the [YItem](#) that corresponds to this event or 0 if the event was constructed with a string ID.

Reimplemented from [YEvent](#).

Reimplemented from [YEvent](#).

Definition at line 274 of file [YEvent.h](#).

The documentation for this class was generated from the following file:

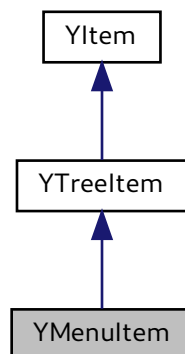
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YEvent.h](#)

## 5.89 YMenuItem Class Reference

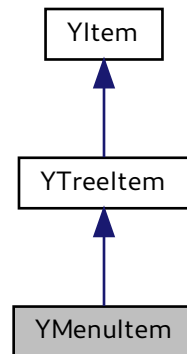
Item class for menu items.

```
#include <YMenuItem.h>
```

Inheritance diagram for YMenuItem:



Collaboration diagram for YMenuItem:



## Public Member Functions

- [YMenuItem](#) (const std::string &label)  
*Constructors for toplevel items.*
- **YMenuItem** (const std::string &label, const std::string &iconName)
- [YMenuItem](#) ([YMenuItem](#) \*parent, const std::string &label)  
*Constructors for items that have a parent item.*
- **YMenuItem** ([YMenuItem](#) \*parent, const std::string &label, const std::string &iconName)
- virtual [~YMenuItem](#) ()  
*Destructor.*
- [YMenuItem](#) \* [parent](#) () const  
*Returns this item's parent item or 0 if it is a toplevel item.*

### 5.89.1 Detailed Description

Item class for menu items.

Definition at line 35 of file [YMenuItem.h](#).

### 5.89.2 Constructor & Destructor Documentation

### 5.89.2.1 YMenuItem()

```
YMenuItem::YMenuItem (
    YMenuItem * parent,
    const std::string & label ) [inline]
```

Constructors for items that have a parent item.

They will automatically register this item with the parent item. The parent assumes ownership of this item and will delete it in its (the parent's) destructor.

Definition at line 57 of file [YMenuItem.h](#).

### 5.89.2.2 ~YMenuItem()

```
virtual YMenuItem::~YMenuItem ( ) [inline], [virtual]
```

Destructor.

This will delete all children.

Definition at line 73 of file [YMenuItem.h](#).

The documentation for this class was generated from the following file:

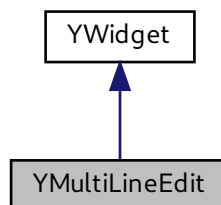
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YMenuItem.h

## 5.90 YMultiLineEdit Class Reference

A multi-line plain-text area.

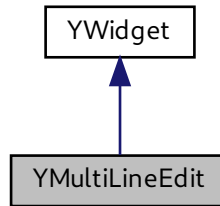
```
#include <YMultiLineEdit.h>
```

Inheritance diagram for YMultiLineEdit:





Collaboration diagram for YMultiLineEdit:



## Public Member Functions

- virtual `~YMultiLineEdit ()`  
*Destructor.*
- virtual const char \* `widgetClass () const`  
*Returns a descriptive name of this widget class for logging, debugging etc.*
- virtual std::string `value ()=0`  
*Get the current value (the text entered by the user or set from the outside) of this MultiLineEdit.*
- virtual void `setValue (const std::string &text)=0`  
*Set the current value (the text entered by the user or set from the outside) of this MultiLineEdit.*
- std::string `label () const`  
*Get the label (the caption above the MultiLineEdit).*
- virtual void `setLabel (const std::string &label)`  
*Set the label (the caption above the MultiLineEdit).*
- int `inputMaxLength () const`  
*The maximum input length, i.e., the maximum number of characters the user can enter.*
- virtual void `setInputMaxLength (int numberOfChars)`  
*Set the maximum input length, i.e., the maximum number of characters the user can enter.*
- int `defaultVisibleLines () const`  
*Return the number of input lines that are visible by default.*
- virtual void `setDefaultVisibleLines (int newVisibleLines)`  
*Set the number of input lines that are visible by default.*
- virtual bool `setProperty (const std::string &propertyName, const YPropertyValue &val)`  
*Set a property.*
- virtual YPropertyValue `getProperty (const std::string &propertyName)`  
*Get a property.*
- virtual const YPropertySet & `propertySet ()`  
*Return this class's property set.*
- virtual std::string `shortcutString () const`  
*Get the string of this widget that holds the keyboard shortcut.*
- virtual void `setShortcutString (const std::string &str)`  
*Set the string of this widget that holds the keyboard shortcut.*
- const char \* `userInputProperty ()`  
*The name of the widget property that will return user input.*

## Protected Member Functions

- [YMultiLineEdit](#) ([YWidget](#) \*parent, const std::string &label)  
*Constructor.*

### 5.90.1 Detailed Description

A multi-line plain-text area.

Definition at line 35 of file [YMultiLineEdit.h](#).

### 5.90.2 Member Function Documentation

#### 5.90.2.1 defaultVisibleLines()

```
int YMultiLineEdit::defaultVisibleLines ( ) const
```

Return the number of input lines that are visible by default.

This is what the widget would like to get (which will be reflected by [preferredHeight\(\)](#) ), not what it currently actually has due to layout constraints.

Definition at line 95 of file [YMultiLineEdit.cc](#).

#### 5.90.2.2 getProperty()

```
YPropertyValue YMultiLineEdit::getProperty (
    const std::string & propertyName ) [virtual]
```

Get a property.

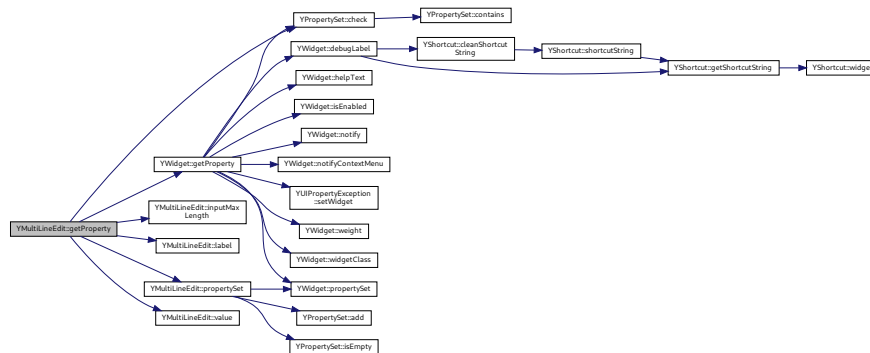
Reimplemented from [YWidget](#).

This method may throw YUIPropertyExceptions.

Reimplemented from [YWidget](#).

Definition at line 147 of file [YMultiLineEdit.cc](#).

Here is the call graph for this function:



### 5.90.2.3 inputMaxLength()

```
int YMultiLineEdit::inputMaxLength ( ) const
```

The maximum input length, i.e., the maximum number of characters the user can enter.

-1 means no limit.

Definition at line 83 of file [YMultiLineEdit.cc](#).

### 5.90.2.4 propertySet()

```
const YPropertySet & YMultiLineEdit::propertySet ( ) [virtual]
```

Return this class's property set.

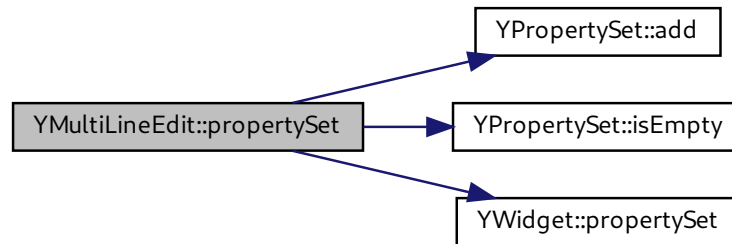
This also initializes the property upon the first call.

Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 108 of file [YMultiLineEdit.cc](#).

Here is the call graph for this function:



#### 5.90.2.5 `setDefaultVisibleLines()`

```
void YMultiLineEdit::setDefaultVisibleLines (
    int newVisibleLines ) [virtual]
```

Set the number of input lines that are visible by default.

This is what the widget would like to get (which will be reflected by [preferredHeight\(\)](#) ), not what it currently actually has due to layout constraints.

Notice that since a `MultiLineEdit` is stretchable in both dimensions, it might get more or less screen space, depending on the layout. This value is only meaningful if there are no other layout constraints.

Changing this value will not trigger a re-layout.

Derived classes can overwrite this function (but should call this base class function in the new function implementation), but it will normally be sufficient to query [defaultVisibleLines\(\)](#) in [preferredHeight\(\)](#).

Definition at line 101 of file [YMultiLineEdit.cc](#).

#### 5.90.2.6 `setInputMaxLength()`

```
void YMultiLineEdit::setInputMaxLength (
    int numberOfChars ) [virtual]
```

Set the maximum input length, i.e., the maximum number of characters the user can enter.

-1 means no limit.

Derived classes are free to reimplement this, but they should call this base class method at the end of the overloaded function.

Definition at line 89 of file [YMultiLineEdit.cc](#).

### 5.90.2.7 setLabel()

```
void YMultiLineEdit::setLabel (
    const std::string & label ) [virtual]
```

Set the label (the caption above the MultiLineEdit).

Derived classes are free to reimplement this, but they should call this base class method at the end of the overloaded function.

Definition at line 77 of file [YMultiLineEdit.cc](#).

Here is the call graph for this function:



### 5.90.2.8 setProperty()

```
bool YMultiLineEdit::setProperty (
    const std::string & propertyName,
    const YPropertyValue & val ) [virtual]
```

Set a property.

Reimplemented from [YWidget](#).

This function may throw `YUIPropertyExceptions`.

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from [YWidget](#).

Definition at line 130 of file [YMultiLineEdit.cc](#).



#### 5.90.2.10 setValue()

```
virtual void YMultiLineEdit::setValue (
    const std::string & text ) [pure virtual]
```

Set the current value (the text entered by the user or set from the outside) of this MultiLineEdit.

Derived classes are required to implement this.

#### 5.90.2.11 shortcutString()

```
virtual std::string YMultiLineEdit::shortcutString ( ) const [inline], [virtual]
```

Get the string of this widget that holds the keyboard shortcut.

Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 161 of file [YMultiLineEdit.h](#).

Here is the call graph for this function:



#### 5.90.2.12 userInputProperty()

```
const char* YMultiLineEdit::userInputProperty ( ) [inline], [virtual]
```

The name of the widget property that will return user input.

Inherited from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 175 of file [YMultiLineEdit.h](#).

### 5.90.2.13 value()

```
virtual std::string YMultiLineEdit::value ( ) [pure virtual]
```

Get the current value (the text entered by the user or set from the outside) of this MultiLineEdit.

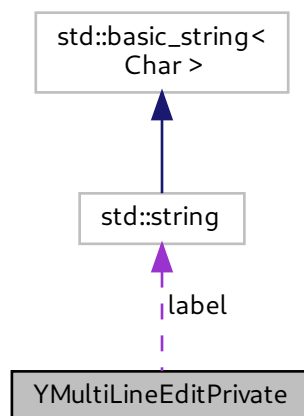
Derived classes are required to implement this.

The documentation for this class was generated from the following files:

- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YMultiLineEdit.h
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YMultiLineEdit.cc

## 5.91 YMultiLineEditPrivate Struct Reference

Collaboration diagram for YMultiLineEditPrivate:



### Public Member Functions

- **YMultiLineEditPrivate** (const string &label)

### Public Attributes

- string **label**
- int **inputMaxLength**
- int **defaultVisibleLines**



### 5.91.1 Detailed Description

Definition at line 38 of file [YMultiLineEdit.cc](#).

The documentation for this struct was generated from the following file:

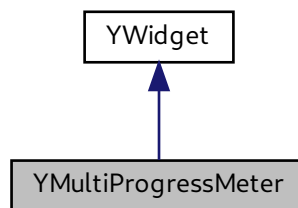
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YMultiLineEdit.cc](#)

## 5.92 YMultiProgressMeter Class Reference

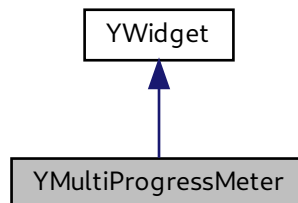
MultiProgressMeter: Progress bar with several segments that can indicate progress individually.

```
#include <YMultiProgressMeter.h>
```

Inheritance diagram for YMultiProgressMeter:



Collaboration diagram for YMultiProgressMeter:



## Public Member Functions

- virtual `~YMultiProgressMeter ()`  
*Destructor.*
- virtual const char \* `widgetClass () const`  
*Return a descriptive name of this widget class for logging, debugging etc.*
- YUIDimension `dimension () const`  
*Return the orientation of the MultiProgressBar.*
- bool `horizontal () const`  
*Return 'true' if the orientation is horizontal.*
- bool `vertical () const`  
*Return 'true' if the orientation is vertical.*
- int `segments () const`  
*Return the number of segments.*
- float `maxValue (int segment) const`  
*Return the maximum value for the specified segment (counting from 0).*
- float `currentValue (int segment) const`  
*Return the current value for the specified segment (counting from 0).*
- void `setCurrentValue (int segment, float value)`  
*Set the current value for the specified segment.*
- void `setCurrentValues (const std::vector< float > &values)`  
*Set all current values and call `doUpdate()`.*
- virtual bool `setProperty (const std::string &propertyName, const YPropertyValue &val)`  
*Set a property.*
- virtual `YPropertyValue getProperty (const std::string &propertyName)`  
*Get a property.*
- virtual const `YPropertySet & propertySet ()`  
*Return this class's property set.*
- virtual void `doUpdate ()=0`  
*Notification that values have been updated and the widget needs to be redisplayed.*

## Protected Member Functions

- `YMultiProgressMeter (YWidget *parent, YUIDimension dim, const std::vector< float > &maxValues)`  
*Constructor.*

### 5.92.1 Detailed Description

MultiProgressMeter: Progress bar with several segments that can indicate progress individually.

This is useful to display progress of several activities that might not necessarily all be done in sequence.

A common example is installing packages from several CDs: Each CD would get a separate segment. Each segment's size would be proportional to the amount of data to be installed from that CD. This visualizes at the same time (a) how many CDs are involved (b) how much in proportion is to be expected from each CD (c) whether or not a specific CD is finished.

Visual example (horizontal MultiProgressMeter):

```
[=====...] [===] [.....] [.]
```

This corresponds to 4 CDs:

CD #1: A lot of packages are to be installed from this CD, and a fair amount of those are already installed, but some are still missing. CD #2: Some packages were installed from this, but this CD is finished. CD #3: Quite some packages are to be installed from this CD. CD #4: Very few packages are to be installed from this CD.

As can be seen from this simple example, this widget can visualize a lot of complex information at the same time in a very natural way.

This is an optional widget, i.e. not all UIs support it.

Definition at line 64 of file [YMultiProgressMeter.h](#).

## 5.92.2 Member Function Documentation

### 5.92.2.1 currentValue()

```
float YMultiProgressMeter::currentValue (
    int segment ) const
```

Return the current value for the specified segment (counting from 0).

If no value has been set yet, -1 is returned.

Definition at line 109 of file [YMultiProgressMeter.cc](#).

### 5.92.2.2 doUpdate()

```
virtual void YMultiProgressMeter::doUpdate ( ) [pure virtual]
```

Notification that values have been updated and the widget needs to be redisplayed.

Derived classes need to reimplement this.

### 5.92.2.3 getProperty()

```
YPropertyValue YMultiProgressMeter::getProperty (
    const std::string & propertyName ) [virtual]
```

Get a property.

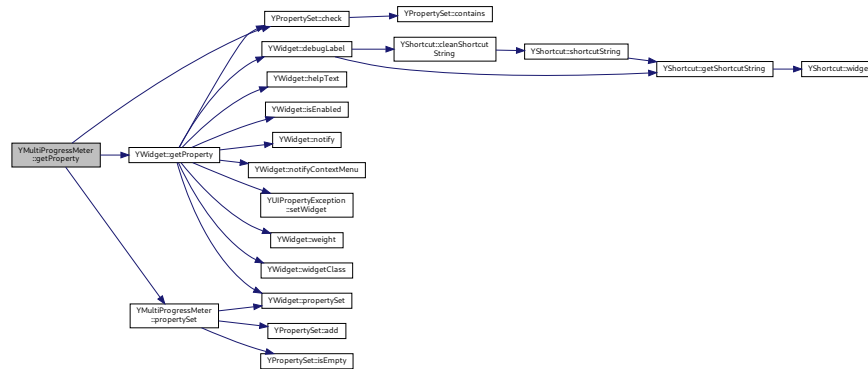
Reimplemented from [YWidget](#).

This method may throw [YUIPropertyExceptions](#).

Reimplemented from [YWidget](#).

Definition at line 176 of file [YMultiProgressMeter.cc](#).

Here is the call graph for this function:



### 5.92.2.4 propertySet()

```
const YPropertySet & YMultiProgressMeter::propertySet ( ) [virtual]
```

Return this class's property set.

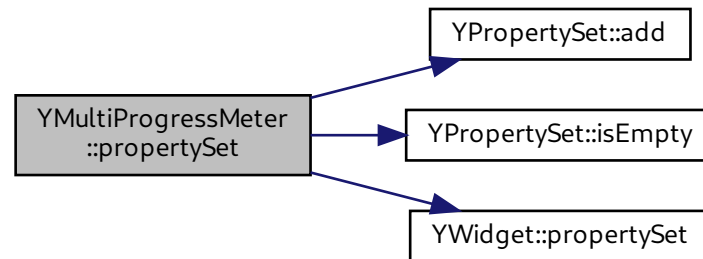
This also initializes the property upon the first call.

Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 143 of file [YMultiProgressMeter.cc](#).

Here is the call graph for this function:



#### 5.92.2.5 setCurrentValue()

```
void YMultiProgressMeter::setCurrentValue (
    int segment,
    float value )
```

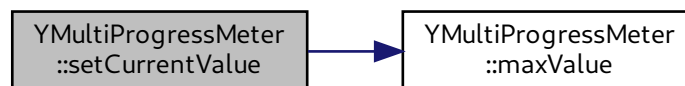
Set the current value for the specified segment.

This must be in the range `0..maxValue( segment )`.

Remember to call [doUpdate\(\)](#) after all changed values are set!

Definition at line 117 of file [YMultiProgressMeter.cc](#).

Here is the call graph for this function:



### 5.92.2.6 `setProperty()`

```
bool YMultiProgressMeter::setProperty (
    const std::string & propertyName,
    const YPropertyValue & val ) [virtual]
```

Set a property.

Reimplemented from [YWidget](#).

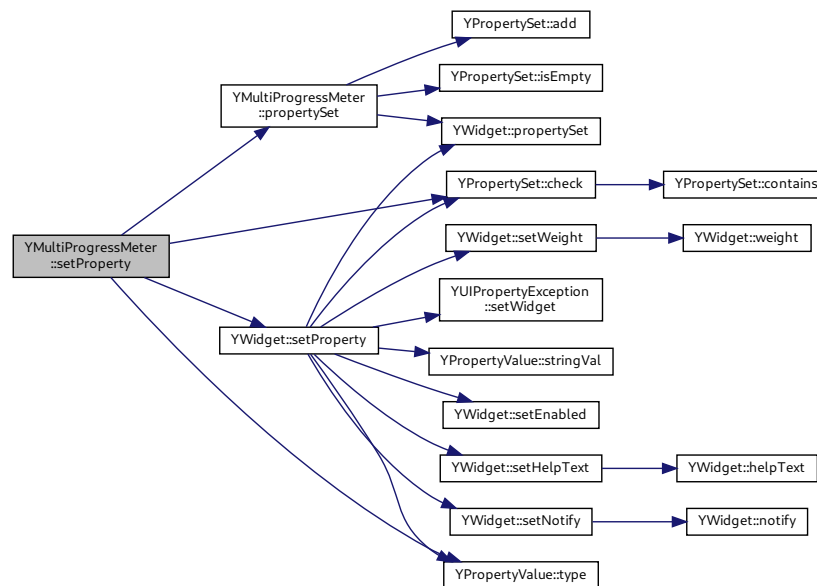
This function may throw [YUIPropertyExceptions](#).

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from [YWidget](#).

Definition at line 161 of file [YMultiProgressMeter.cc](#).

Here is the call graph for this function:

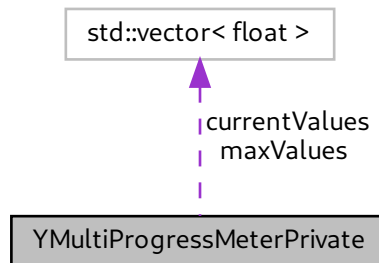


The documentation for this class was generated from the following files:

- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YMultiProgressMeter.h`
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YMultiProgressMeter.cc`

## 5.93 YMultiProgressMeterPrivate Struct Reference

Collaboration diagram for YMultiProgressMeterPrivate:



### Public Member Functions

- **YMultiProgressMeterPrivate** (YUIDimension dim, const vector< float > &maxValues)

### Public Attributes

- YUIDimension **dim**
- vector< float > **maxValues**
- vector< float > **currentValues**

#### 5.93.1 Detailed Description

Definition at line 36 of file [YMultiProgressMeter.cc](#).

The documentation for this struct was generated from the following file:

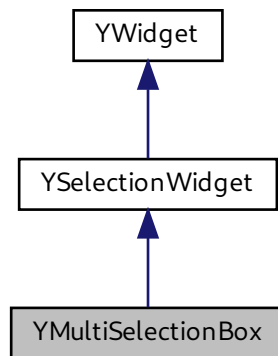
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YMultiProgressMeter.cc

## 5.94 YMultiSelectionBox Class Reference

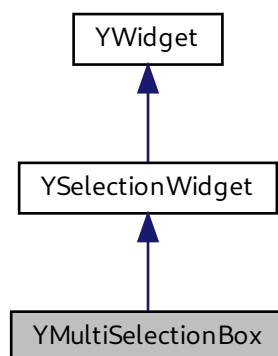
A variant of [YSelectionBox](#) where more than one item can be selected.

```
#include <YMultiSelectionBox.h>
```

Inheritance diagram for YMultiSelectionBox:



Collaboration diagram for YMultiSelectionBox:





## Public Member Functions

- virtual `~YMultiSelectionBox ()`  
*Destructor.*
- virtual const char \* `widgetClass ()` const  
*Returns a descriptive name of this widget class for logging, debugging etc.*
- bool `shrinkable ()` const  
*Return 'true' if this MultiSelectionBox should be very small.*
- virtual void `setShrinkable (bool shrinkable=true)`  
*Make this MultiSelectionBox very small.*
- virtual bool `setProperty (const std::string &propertyName, const YPropertyValue &val)`  
*Set a property.*
- virtual `YPropertyValue getProperty (const std::string &propertyName)`  
*Get a property.*
- virtual const `YPropertySet & propertySet ()`  
*Return this class's property set.*
- const char \* `userInputProperty ()`  
*The name of the widget property that will return user input.*
- virtual `YItem * currentItem ()=0`  
*Return the the item that currently has the keyboard focus or 0 if no item currently has the keyboard focus.*
- virtual void `setCurrentItem (YItem *item)=0`  
*Set the keyboard focus to the specified item.*
- virtual void `saveUserInput (YMacroRecorder *macroRecorder)`  
*Save the widget's user input to a macro recorder.*

## Protected Member Functions

- `YMultiSelectionBox (YWidget *parent, const std::string &label)`  
*Constructor.*

### 5.94.1 Detailed Description

A variant of `YSelectionBox` where more than one item can be selected.

Definition at line 36 of file `YMultiSelectionBox.h`.

### 5.94.2 Member Function Documentation

### 5.94.2.1 currentItem()

```
virtual YItem* YMultiSelectionBox::currentItem ( ) [pure virtual]
```

Return the the item that currently has the keyboard focus or 0 if no item currently has the keyboard focus.

Notice that for a MultiSelectionBox the current item is not necessarily selected, i.e., its check box may or may not be checked.

Derived classes are required to implement this function.

### 5.94.2.2 getProperty()

```
YPropertyValue YMultiSelectionBox::getProperty (
    const std::string & propertyName ) [virtual]
```

Get a property.

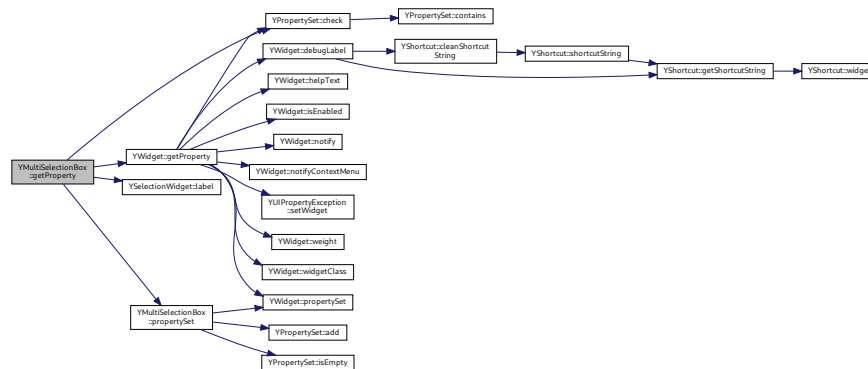
Reimplemented from [YWidget](#).

This method may throw YUIPropertyExceptions.

Reimplemented from [YWidget](#).

Definition at line 124 of file [YMultiSelectionBox.cc](#).

Here is the call graph for this function:



### 5.94.2.3 propertySet()

```
const YPropertySet & YMultiSelectionBox::propertySet ( ) [virtual]
```

Return this class's property set.

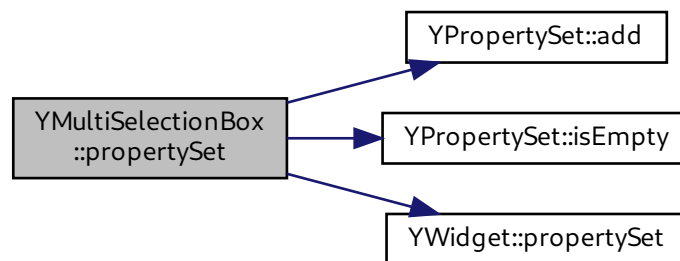
This also initializes the property upon the first call.

Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 80 of file [YMultiSelectionBox.cc](#).

Here is the call graph for this function:



### 5.94.2.4 saveUserInput()

```
void YMultiSelectionBox::saveUserInput (
    YMacroRecorder * macroRecorder ) [virtual]
```

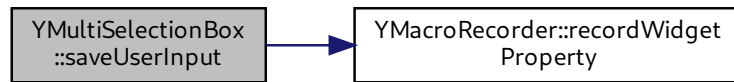
Save the widget's user input to a macro recorder.

Reimplemented from [YWidget](#) because two properties need to be recorded.

Reimplemented from [YWidget](#).

Definition at line 141 of file [YMultiSelectionBox.cc](#).

Here is the call graph for this function:



#### 5.94.2.5 setCurrentItem()

```
virtual void YMultiSelectionBox::setCurrentItem (
    YItem * item ) [pure virtual]
```

Set the keyboard focus to the specified item.

0 means clear the keyboard focus.

Notice that for a MultiSelectionBox the current item is not necessarily selected, i.e., its check box may or may not be checked. Use [selectItem\(\)](#) for that.

Also notice that [selectItem\(\)](#) does not make that newly selected item the current item.

Derived classes are required to implement this function.

#### 5.94.2.6 setProperty()

```
bool YMultiSelectionBox::setProperty (
    const std::string & propertyName,
    const YPropertyValue & val ) [virtual]
```

Set a property.

Reimplemented from [YWidget](#).

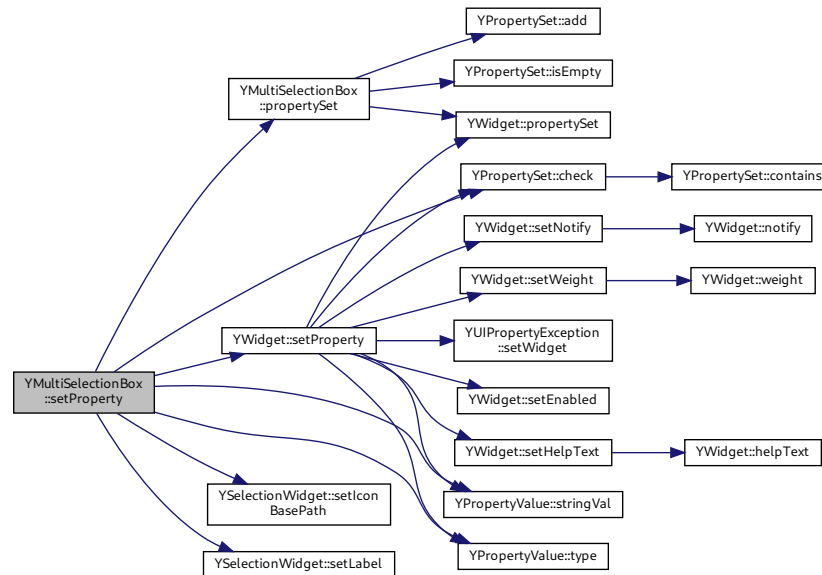
This function may throw YUIPropertyExceptions.

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from [YWidget](#).

Definition at line [105](#) of file [YMultiSelectionBox.cc](#).

Here is the call graph for this function:



#### 5.94.2.7 setShrinkable()

```
void YMultiSelectionBox::setShrinkable (
    bool shrinkable = true ) [virtual]
```

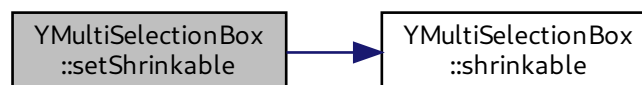
Make this MultiSelectionBox very small.

This will take effect only upon the next geometry management run.

Derived classes can overwrite this, but should call this base class function in the new function.

Definition at line 73 of file [YMultiSelectionBox.cc](#).

Here is the call graph for this function:



#### 5.94.2.8 userInputProperty()

```
const char* YMultiSelectionBox::userInputProperty ( ) [inline], [virtual]
```

The name of the widget property that will return user input.

Inherited from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 103 of file [YMultiSelectionBox.h](#).

The documentation for this class was generated from the following files:

- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YMultiSelectionBox.h](#)
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YMultiSelectionBox.cc](#)

## 5.95 YMultiSelectionBoxPrivate Struct Reference

### Public Attributes

- bool **shrinkable**

#### 5.95.1 Detailed Description

Definition at line 37 of file [YMultiSelectionBox.cc](#).

The documentation for this struct was generated from the following file:

- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YMultiSelectionBox.cc](#)

## 5.96 YOptionalWidgetFactory Class Reference

Abstract widget factory for optional ("special") widgets.

```
#include <YOptionalWidgetFactory.h>
```

## Public Member Functions

- virtual bool **hasWizard** ()
- virtual **YWizard** \* **createWizard** (**YWidget** \*parent, const std::string &backButtonLabel, const std::string &abort←  
ButtonLabel, const std::string &nextButtonLabel, **YWizardMode** wizardMode=**YWizardMode\_Standard**)
- virtual bool **hasDumbTab** ()
- virtual **YDumbTab** \* **createDumbTab** (**YWidget** \*parent)
- virtual bool **hasSlider** ()
- virtual **YSlider** \* **createSlider** (**YWidget** \*parent, const std::string &label, int minVal, int maxVal, int initialVal)
- virtual bool **hasDateField** ()
- virtual **YDateField** \* **createDateField** (**YWidget** \*parent, const std::string &label)
- virtual bool **hasTimeField** ()
- virtual **YTimeField** \* **createTimeField** (**YWidget** \*parent, const std::string &label)
- virtual bool **hasBarGraph** ()
- virtual **YBarGraph** \* **createBarGraph** (**YWidget** \*parent)
- virtual bool **hasPatternSelector** ()
- virtual **YWidget** \* **createPatternSelector** (**YWidget** \*parent, long modeFlags=0)
- virtual bool **hasSimplePatchSelector** ()
- virtual **YWidget** \* **createSimplePatchSelector** (**YWidget** \*parent, long modeFlags=0)
- virtual bool **hasMultiProgressMeter** ()
- **YMultiProgressMeter** \* **createHMultiProgressMeter** (**YWidget** \*parent, const std::vector< float > &maxValues)
- **YMultiProgressMeter** \* **createVMultiProgressMeter** (**YWidget** \*parent, const std::vector< float > &maxValues)
- virtual **YMultiProgressMeter** \* **createMultiProgressMeter** (**YWidget** \*parent, YUIDimension dim, const std←  
::vector< float > &maxValues)
- virtual bool **hasPartitionSplitter** ()
- virtual **YPartitionSplitter** \* **createPartitionSplitter** (**YWidget** \*parent, int usedSize, int totalFreeSize, int new←  
PartSize, int minNewPartSize, int minFreeSize, const std::string &usedLabel, const std::string &freeLabel, const  
std::string &newPartLabel, const std::string &freeFieldLabel, const std::string &newPartFieldLabel)
- virtual bool **hasDownloadProgress** ()
- virtual **YDownloadProgress** \* **createDownloadProgress** (**YWidget** \*parent, const std::string &label, const std←  
::string &filename, YFileSize\_t expectedFileSize)
- bool **hasDummySpecialWidget** ()
- **YWidget** \* **createDummySpecialWidget** (**YWidget** \*parent)
- virtual bool **hasTimezoneSelector** ()
- virtual **YTimezoneSelector** \* **createTimezoneSelector** (**YWidget** \*parent, const std::string &timezoneMap, const  
std::map< std::string, std::string > &timezones)
- virtual bool **hasGraph** ()
- virtual **YGraph** \* **createGraph** (**YWidget** \*parent, const std::string &filename, const std::string &layoutAlgorithm)
- virtual **YGraph** \* **createGraph** (**YWidget** \*parent, void \*graph)
- virtual bool **hasContextMenu** ()

## Protected Member Functions

- **YOptionalWidgetFactory** ()  
*Constructor.*
- virtual **~YOptionalWidgetFactory** ()  
*Destructor.*

## Friends

- class **YUI**

### 5.96.1 Detailed Description

Abstract widget factory for optional ("special") widgets.

Remember to always check with the corresponding "has..()" method if the current UI actually provides the requested widget. Otherwise the "create..." method will throw an exception.

Definition at line 56 of file [YOptionalWidgetFactory.h](#).

### 5.96.2 Constructor & Destructor Documentation

#### 5.96.2.1 YOptionalWidgetFactory()

```
YOptionalWidgetFactory::YOptionalWidgetFactory ( ) [protected]
```

Constructor.

Use [YUI::optionalWidgetFactory\(\)](#) to get the singleton for this class.

Definition at line 44 of file [YOptionalWidgetFactory.cc](#).

The documentation for this class was generated from the following files:

- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YOptionalWidgetFactory.h
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YOptionalWidgetFactory.cc

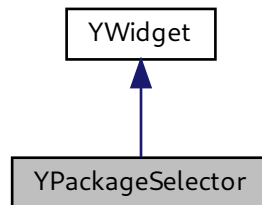


## 5.97 YPackageSelector Class Reference

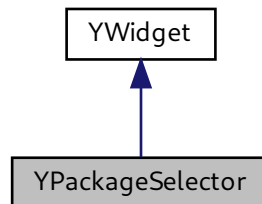
A simple wrapper for an insanely complex UI for installing software.

```
#include <YPackageSelector.h>
```

Inheritance diagram for YPackageSelector:



Collaboration diagram for YPackageSelector:



### Public Member Functions

- virtual const char \* [widgetClass](#) () const  
*Returns a descriptive name of this widget class for logging, debugging etc.*
- bool [testMode](#) () const  
*Check for the various modes.*
- bool **onlineUpdateMode** () const
- bool **updateMode** () const
- bool **searchMode** () const
- bool **summaryMode** () const
- bool **repoMode** () const
- bool **repoMgrEnabled** () const
- bool **confirmUnsupported** () const
- bool **onlineSearchEnabled** () const

## Protected Member Functions

- [YPackageSelector](#) ([YWidget](#) \*parent, long modeFlags=0)  
*Constructor.*

## Protected Attributes

- long `_modeFlags`

### 5.97.1 Detailed Description

A simple wrapper for an insanely complex UI for installing software.

Definition at line 43 of file [YPackageSelector.h](#).

### 5.97.2 Constructor & Destructor Documentation

#### 5.97.2.1 YPackageSelector()

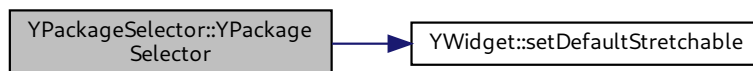
```
YPackageSelector::YPackageSelector (
    YWidget * parent,
    long modeFlags = 0 ) [protected]
```

Constructor.

'modeFlags' are flags determining which modes to use, ORed together: YPkg\_OnlineUpdateMode | YPkg\_TestMode

Definition at line 32 of file [YPackageSelector.cc](#).

Here is the call graph for this function:



The documentation for this class was generated from the following files:

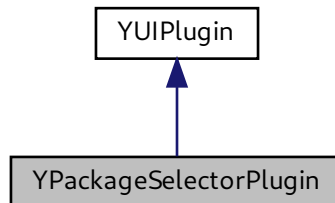
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YPackageSelector.h
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YPackageSelector.cc

## 5.98 YPackageSelectorPlugin Class Reference

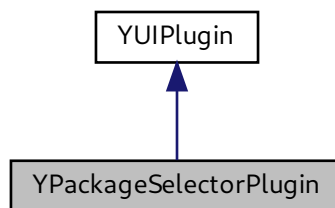
Abstract base class for simplified access to UI plugins for package selection.

```
#include <YPackageSelectorPlugin.h>
```

Inheritance diagram for YPackageSelectorPlugin:



Collaboration diagram for YPackageSelectorPlugin:



### Public Member Functions

- virtual [YPackageSelector](#) \* [createPackageSelector](#) ([YWidget](#) \*parent, long modeFlags=0)=0  
*Create a package selector.*

### Protected Member Functions

- [YPackageSelectorPlugin](#) (const char \*pluginLibBaseName)  
*Constructor: Load the specified plugin library from the standard UI plugin directory (/usr/lib/yui).*
- virtual [~YPackageSelectorPlugin](#) ()  
*Destructor.*

### 5.98.1 Detailed Description

Abstract base class for simplified access to UI plugins for package selection.

Definition at line 38 of file [YPackageSelectorPlugin.h](#).

### 5.98.2 Constructor & Destructor Documentation

#### 5.98.2.1 ~YPackageSelectorPlugin()

```
virtual YPackageSelectorPlugin::~YPackageSelectorPlugin ( ) [inline], [protected], [virtual]
```

Destructor.

Calls `dlclose()` which will unload the plugin library if it is no longer used, i.e. if the reference count `dlopen()` uses reaches 0.

Definition at line 52 of file [YPackageSelectorPlugin.h](#).

### 5.98.3 Member Function Documentation

#### 5.98.3.1 createPackageSelector()

```
virtual YPackageSelector* YPackageSelectorPlugin::createPackageSelector (
    YWidget * parent,
    long modeFlags = 0 ) [pure virtual]
```

Create a package selector.

Derived classes need to implement this.

This might return 0 if the plugin lib could not be loaded or if the appropriate symbol could not be located in the plugin lib.

The documentation for this class was generated from the following file:

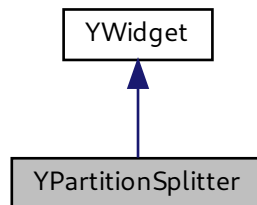
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YPackageSelectorPlugin.h`

## 5.99 YPartitionSplitter Class Reference

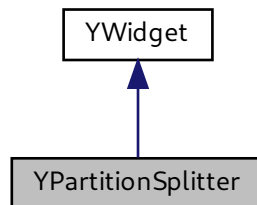
PartitionSplitter: A (very custom) widget for easily splitting one existing partition into two.

```
#include <YPartitionSplitter.h>
```

Inheritance diagram for YPartitionSplitter:



Collaboration diagram for YPartitionSplitter:



### Public Member Functions

- virtual `~YPartitionSplitter` ()  
*Destructor.*
- virtual const char \* `widgetClass` () const  
*Returns a descriptive name of this widget class for logging, debugging etc.*
- virtual int `value` ()=0  
*The value of this PartitionSplitter: The size of the new partition.*
- virtual void `setValue` (int newValue)=0  
*Set the value (the size of the new partition).*

- int **usedSize** () const
- int **totalFreeSize** () const
- int **minFreeSize** () const
- int **maxFreeSize** () const
- int **freeSize** ()
- int **newPartSize** ()
- int **minNewPartSize** () const
- int **maxNewPartSize** () const
- std::string **usedLabel** () const
- std::string **freeLabel** () const
- std::string **newPartLabel** () const
- std::string **freeFieldLabel** () const
- std::string **newPartFieldLabel** () const
- virtual bool **setProperty** (const std::string &propertyName, const [YPropertyValue](#) &val)  
*Set a property.*
- virtual [YPropertyValue](#) **getProperty** (const std::string &propertyName)  
*Get a property.*
- virtual const [YPropertySet](#) & **propertySet** ()  
*Return this class's property set.*
- const char \* **userInputProperty** ()  
*The name of the widget property that will return user input.*

## Protected Member Functions

- [YPartitionSplitter](#) ([YWidget](#) \*parent, int usedSize, int totalFreeSize, int newPartSize, int minNewPartSize, int minFreeSize, const std::string &usedLabel, const std::string &freeLabel, const std::string &newPartLabel, const std::string &freeFieldLabel, const std::string &newPartFieldLabel)  
*Constructor.*

### 5.99.1 Detailed Description

PartitionSplitter: A (very custom) widget for easily splitting one existing partition into two.

Layout:

```
+-----+-----+-----+
| Old Partition | Old Partition | New Partition |
| used         | free         |               |
+-----+-----+-----+

Old Partition free                               New Partition
[ 123 ] =====O===== [ 123 ]
```

At the top, there is a BarGraph that dynamically displays the sizes in graphical form. Below are an IntField to the left and an IntField to the right, each with its respective label. Between the two IntFields there is a Slider.

The user can enter a value in either IntField or drag the slider. The other sub-widgets (including the BarGraph) will automatically be adjusted. Visually (in the BarGraph), the border between "old partition free" and "new partition" will move left and right. The border between "old partition used" and "old partition free" is static.

There are built-in (configurable) limits for the minimum sizes of "old partition free" and "new partition".

Definition at line 63 of file [YPartitionSplitter.h](#).

## 5.99.2 Constructor & Destructor Documentation

### 5.99.2.1 YPartitionSplitter()

```
YPartitionSplitter::YPartitionSplitter (
    YWidget * parent,
    int usedSize,
    int totalFreeSize,
    int newPartSize,
    int minNewPartSize,
    int minFreeSize,
    const std::string & usedLabel,
    const std::string & freeLabel,
    const std::string & newPartLabel,
    const std::string & freeFieldLabel,
    const std::string & newPartFieldLabel ) [protected]
```

Constructor.

usedSize: Used size of the old partition (constant)

totalFreeSize: Total free size of the old partition before the split: OldPartitionFree + NewPartition

newPartSize': Initial size of the new partition

minNewPartSize: Mimumum size of the new partition

minFreeSize: Minimum free size of the old partition

usedLabel: BarGraph label for the used part of the old partition

freeLabel: BarGraph label for the free part of the old partition

newPartLabel: BarGraph label for the new partition

freeFieldLabel: IntField label for the free part of the old partition

newPartFieldLabel: IntField label for the size of the new partition

Definition at line 71 of file [YPartitionSplitter.cc](#).

Here is the call graph for this function:



### 5.99.3 Member Function Documentation

#### 5.99.3.1 getProperty()

```
YPropertyValue YPartitionSplitter::getProperty (
    const std::string & propertyName ) [virtual]
```

Get a property.

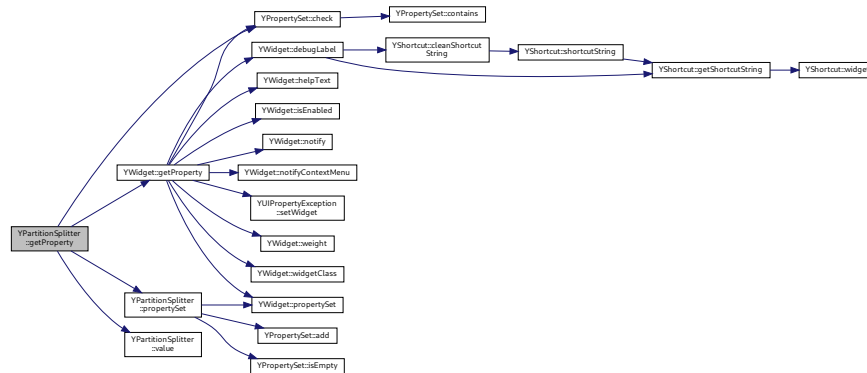
Reimplemented from [YWidget](#).

This method may throw YUIPropertyExceptions.

Reimplemented from [YWidget](#).

Definition at line 195 of file [YPartitionSplitter.cc](#).

Here is the call graph for this function:



#### 5.99.3.2 propertySet()

```
const YPropertySet & YPartitionSplitter::propertySet ( ) [virtual]
```

Return this class's property set.

This also initializes the property upon the first call.

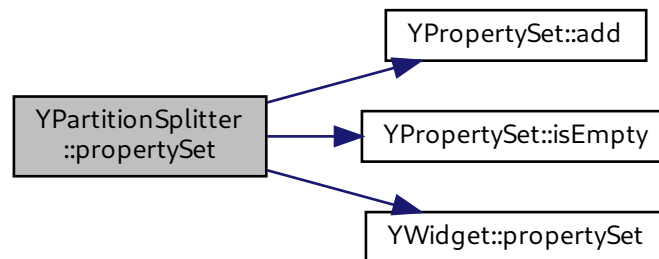
Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).



Definition at line 162 of file [YPartitionSplitter.cc](#).

Here is the call graph for this function:



### 5.99.3.3 `setProperty()`

```
bool YPartitionSplitter::setProperty (  
    const std::string & propertyName,  
    const YPropertyValue & val ) [virtual]
```

Set a property.

Reimplemented from [YWidget](#).

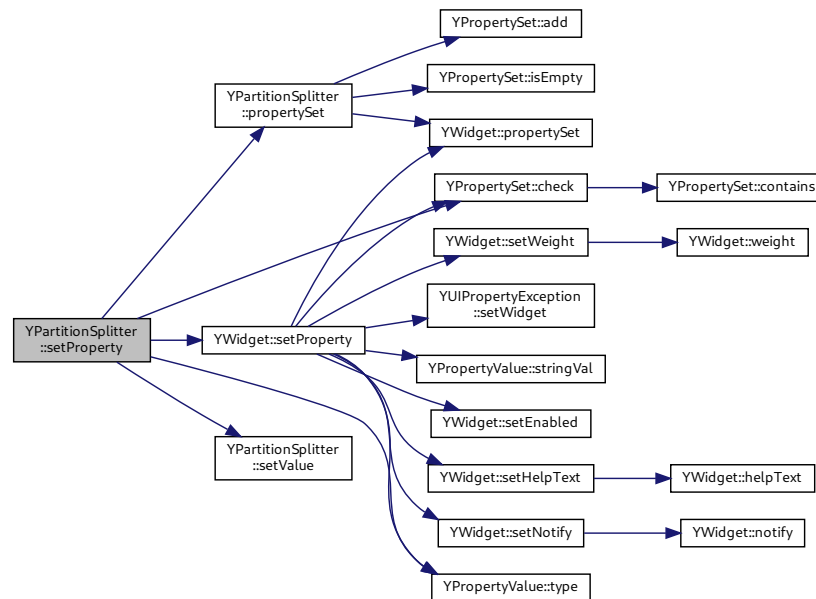
This function may throw `YUIPropertyExceptions`.

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from [YWidget](#).

Definition at line 180 of file [YPartitionSplitter.cc](#).

Here is the call graph for this function:



#### 5.99.3.4 setValue()

```
virtual void YPartitionSplitter::setValue (
    int newValue ) [pure virtual]
```

Set the value (the size of the new partition).

Derived classes are required to implement this.

#### 5.99.3.5 userInputProperty()

```
const char* YPartitionSplitter::userInputProperty ( ) [inline], [virtual]
```

The name of the widget property that will return user input.

Inherited from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 181 of file [YPartitionSplitter.h](#).

## 5.99.3.6 value()

```
virtual int YPartitionSplitter::value ( ) [pure virtual]
```

The value of this PartitionSplitter: The size of the new partition.

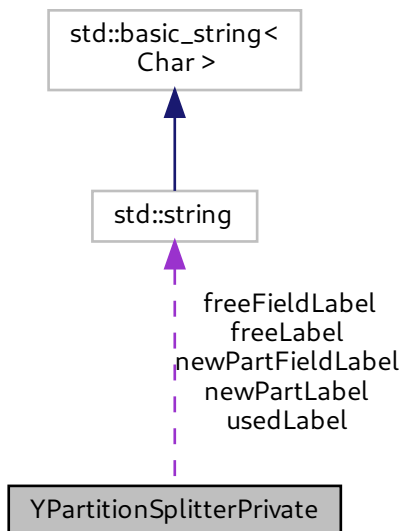
Derived classes are required to implement this.

The documentation for this class was generated from the following files:

- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YPartitionSplitter.h
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YPartitionSplitter.cc

## 5.100 YPartitionSplitterPrivate Struct Reference

Collaboration diagram for YPartitionSplitterPrivate:



## Public Member Functions

- **YPartitionSplitterPrivate** (int usedSize, int totalFreeSize, int minNewPartSize, int minFreeSize, const string &usedLabel, const string &freeLabel, const string &newPartLabel, const string &freeFieldLabel, const string &newPartFieldLabel)

## Public Attributes

- int **usedSize**
- int **totalFreeSize**
- int **minNewPartSize**
- int **minFreeSize**
- string **usedLabel**
- string **freeLabel**
- string **newPartLabel**
- string **freeFieldLabel**
- string **newPartFieldLabel**

### 5.100.1 Detailed Description

Definition at line 35 of file [YPartitionSplitter.cc](#).

The documentation for this struct was generated from the following file:

- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YPartitionSplitter.cc

## 5.101 YPath Class Reference

Finds files (e.g.

```
#include <YPath.h>
```

## Public Member Functions

- [YPath](#) (const std::string &directory, const std::string &filename)  
*Constructor.*
- [~YPath](#) ()  
*Destructor.*
- std::string [path](#) ()  
*Returns the full path of the file if found; if not found just the filename given in constructor.*
- std::string [dir](#) ()  
*Returns the directory where the file is found; if not found just the subdir part (if there's any) of the filename given in constructor.*

### 5.101.1 Detailed Description

Finds files (e.g.

plugins or theme pixmaps) recursively inside a directory.

Definition at line 43 of file [YPath.h](#).

## 5.101.2 Constructor & Destructor Documentation

### 5.101.2.1 YPath()

```
YPath::YPath (
    const std::string & directory,
    const std::string & filename )
```

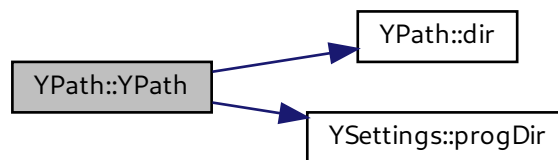
Constructor.

to be called with the directory where to look inside and filename which to lookup.

YSettings::progSubDir will be preferred by the lookup.

Definition at line 51 of file [YPath.cc](#).

Here is the call graph for this function:



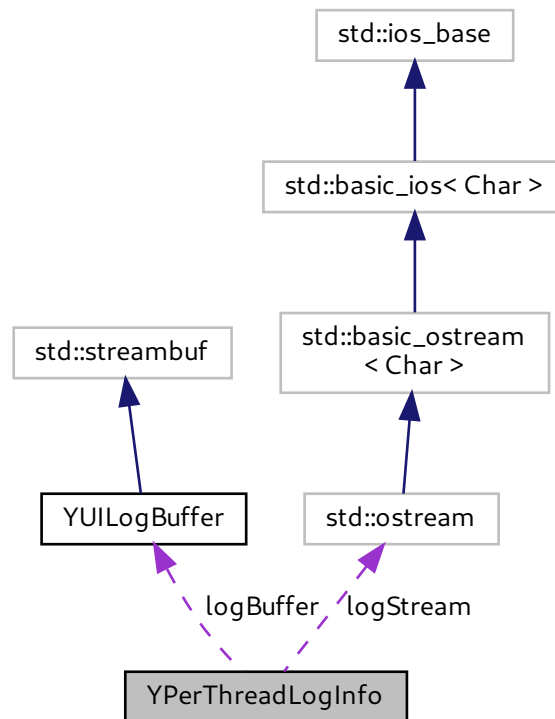
The documentation for this class was generated from the following files:

- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YPath.h
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YPath.cc

## 5.102 YPerThreadLogInfo Struct Reference

Helper class: Per-thread logging information.

Collaboration diagram for YPerThreadLogInfo:



### Public Member Functions

- [YPerThreadLogInfo](#) ()  
*Constructor.*
- [~YPerThreadLogInfo](#) ()  
*Destructor.*
- `bool` [isThread](#) (pthread\_t otherThreadHandle)  
*Check if this per-thread logging information belongs to the specified thread.*

### Public Attributes

- pthread\_t **threadHandle**
- [YUILogBuffer](#) **logBuffer**
- ostream **logStream**

### 5.102.1 Detailed Description

Helper class: Per-thread logging information.

Multiple threads can easily clobber each others' half-done logging. A naive approach to prevent this would be to lock a mutex when a thread starts logging and unlock it when it's done logging. But that "when it's done" condition might never come true. endl or a newline in the output stream would be one indication, but there is no way to make sure there always is such a delimiter. If it is forgotten and that thread (that still has the mutex locked) runs into a waiting condition itself (e.g., UI thread synchronization with pipes), there would be a deadlock.

So this much safer approach was chosen: Give each thread its own logging infrastructure, i.e., its own log stream and its own log buffer.

Sure, in bad cases the logger function might still be executed in parallel and thus clobber a line or two of log output. But that's merely bad output formatting, not writing another thread's data structures without control - which can easily happen if multiple threads are working on the same output buffer, i.e. manipulate the same string.

Definition at line 212 of file [YUILog.cc](#).

The documentation for this struct was generated from the following file:

- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YUILog.cc`

## 5.103 YPopupInternal Class Reference

### Public Types

- `typedef std::vector< std::string > StringArray`

### Static Public Member Functions

- static void [message](#) (const std::string &label)  
*Display a simple popup dialog with OK button.*
- static bool [editStringArray](#) (StringArray &array, const std::string &label)  
*Display a popup dialog with several input fields.*
- static StringArray [editNewStringArray](#) (const std::string &label)  
*Display a popup dialog with 3 initially empty input fields.*

### 5.103.1 Detailed Description

Definition at line 21 of file [YPopupInternal.h](#).

### 5.103.2 Member Function Documentation

#### 5.103.2.1 editNewStringArray()

```
YPopupInternal::StringArray YPopupInternal::editNewStringArray (
    const std::string & label ) [static]
```

Display a popup dialog with 3 initially empty input fields.





### 5.103.2.3 message()

```
void YPopupInternal::message (
    const std::string & label ) [static]
```

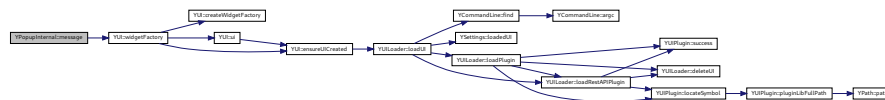
Display a simple popup dialog with OK button.

### Parameters

<i>label</i>	the message to display
--------------	------------------------

Definition at line 36 of file YPopupInternal.cc.

Here is the call graph for this function:



The documentation for this class was generated from the following files:

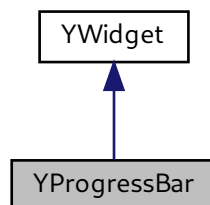
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YPopupInternal.h
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YPopupInternal.cc

## 5.104 YProgressBar Class Reference

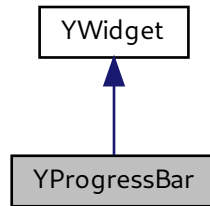
A progress bar, showing completion of `value()` out of `maxValue()` parts.

```
#include <YProgressBar.h>
```

Inheritance diagram for YProgressBar:



Collaboration diagram for YProgressBar:



## Public Member Functions

- virtual `~YProgressBar()`  
*Destructor.*
- virtual const char \* `widgetClass()` const  
*Returns a descriptive name of this widget class for logging, debugging etc.*
- std::string `label()`  
*Get the label (the caption above the progress bar).*
- virtual void `setLabel(const std::string &label)`  
*Set the label (the caption above the progress bar).*
- int `maxValue()` const  
*Return the maximum progress value.*
- int `value()` const  
*Return the current progress value.*
- virtual void `setValue(int newValue)`  
*Set the current progress value (  $\leq$  `maxValue()` ).*
- virtual bool `setProperty(const std::string &propertyName, const YPropertyValue &val)`  
*Set a property.*
- virtual `YPropertyValue getProperty(const std::string &propertyName)`  
*Get a property.*
- virtual const `YPropertySet &propertySet()`  
*Return this class's property set.*

## Protected Member Functions

- `YProgressBar(YWidget *parent, const std::string &label, int maxValue=100)`  
*Constructor.*

### 5.104.1 Detailed Description

A progress bar, showing completion of `value()` out of `maxValue()` parts.

Definition at line 36 of file `YProgressBar.h`.

### 5.104.2 Member Function Documentation

#### 5.104.2.1 `getProperty()`

```
YPropertyValue YProgressBar::getProperty (
    const std::string & propertyName ) [virtual]
```

Get a property.

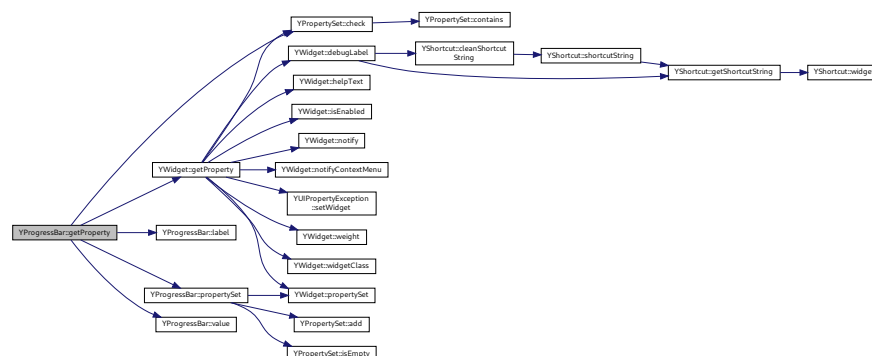
Reimplemented from `YWidget`.

This method may throw `YUIPropertyExceptions`.

Reimplemented from `YWidget`.

Definition at line 147 of file `YProgressBar.cc`.

Here is the call graph for this function:



### 5.104.2.2 `maxValue()`

```
int YProgressBar::maxValue ( ) const
```

Return the maximum progress value.

Notice that this value can only be set in the constructor.

Definition at line 86 of file [YProgressBar.cc](#).

### 5.104.2.3 `propertySet()`

```
const YPropertySet & YProgressBar::propertySet ( ) [virtual]
```

Return this class's property set.

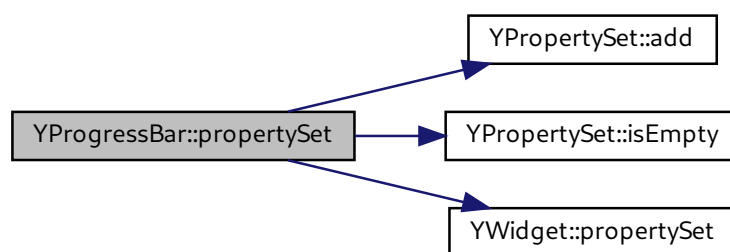
This also initializes the property upon the first call.

Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 111 of file [YProgressBar.cc](#).

Here is the call graph for this function:



#### 5.104.2.4 setLabel()

```
void YProgressBar::setLabel (
    const std::string & label ) [virtual]
```

Set the label (the caption above the progress bar).

Derived classes are free to reimplement this, but they should call this base class method at the end of the overloaded function.

Definition at line 80 of file [YProgressBar.cc](#).

Here is the call graph for this function:



#### 5.104.2.5 setProperty()

```
bool YProgressBar::setProperty (
    const std::string & propertyName,
    const YPropertyValue & val ) [virtual]
```

Set a property.

Reimplemented from [YWidget](#).

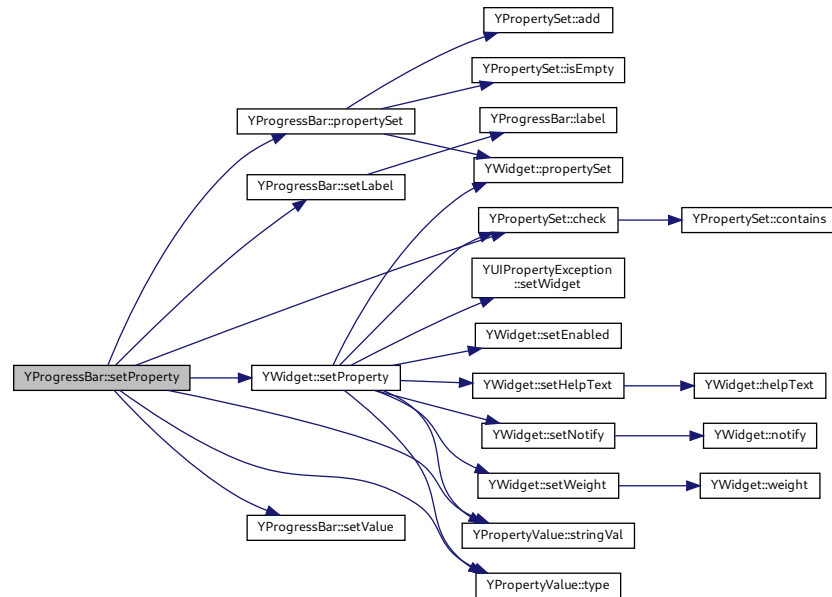
This function may throw `YUIPropertyExceptions`.

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from [YWidget](#).

Definition at line 131 of file [YProgressBar.cc](#).

Here is the call graph for this function:



#### 5.104.2.6 setValue()

```
void YProgressBar::setValue (
    int newValue ) [virtual]
```

Set the current progress value ( `<= maxValue()` ).

Derived classes should reimplement this, but they should call this base class method at the end of the overloaded function.

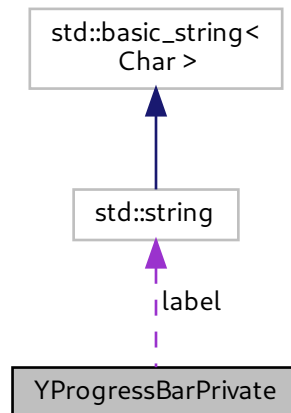
Definition at line 98 of file [YProgressBar.cc](#).

The documentation for this class was generated from the following files:

- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YProgressBar.h`
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YProgressBar.cc`

## 5.105 YProgressBarPrivate Struct Reference

Collaboration diagram for YProgressBarPrivate:



### Public Member Functions

- **YProgressBarPrivate** (const string &label, int maxValue)

### Public Attributes

- string **label**
- int **maxValue**
- int **value**

#### 5.105.1 Detailed Description

Definition at line 35 of file [YProgressBar.cc](#).

The documentation for this struct was generated from the following file:

- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YProgressBar.cc

## 5.106 YProperty Class Reference

Class for widget properties.

```
#include <YProperty.h>
```

## Public Member Functions

- [YProperty](#) (const std::string &[name](#), YPropertyType [type](#), bool [isReadOnly](#)=false)  
*Constructor: Create a property with the specified name and type.*
- std::string [name](#) () const  
*Returns the name of this property.*
- YPropertyType [type](#) () const  
*Returns the type of this property.*
- bool [isReadOnly](#) () const  
*Returns 'true' if this property cannot be changed, only retrieved.*
- std::string [typeAsStr](#) () const  
*Returns the type of this property as string.*

## Static Public Member Functions

- static std::string [typeAsStr](#) (YPropertyType [type](#))  
*Returns a string description of a property type.*

### 5.106.1 Detailed Description

Class for widget properties.

Definition at line 51 of file [YProperty.h](#).

### 5.106.2 Constructor & Destructor Documentation

#### 5.106.2.1 YProperty()

```
YProperty::YProperty (  
    const std::string & name,  
    YPropertyType type,  
    bool isReadOnly = false ) [inline]
```

Constructor: Create a property with the specified name and type.

'isReadOnly' is for properties that cannot be set, only retrieved.

Definition at line 58 of file [YProperty.h](#).

The documentation for this class was generated from the following files:

- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YProperty.h
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YProperty.cc



## 5.107 YPropertyEditor Class Reference

An internal helper class for displaying the widget property editor in the spy dialog.

```
#include <YPropertyEditor.h>
```

### Public Member Functions

- [YPropertyEditor](#) ([YWidget](#) \*widget)  
*Constructor.*
- bool [edit](#) (const std::string &property)  
*Display a popup for editing a widget property.*

### 5.107.1 Detailed Description

An internal helper class for displaying the widget property editor in the spy dialog.

See also

[YDialogSpy](#)

Definition at line 32 of file [YPropertyEditor.h](#).

### 5.107.2 Constructor & Destructor Documentation

#### 5.107.2.1 YPropertyEditor()

```
YPropertyEditor::YPropertyEditor (  
    YWidget * widget )
```

Constructor.

Parameters

<i>widget</i>	the target widget
---------------	-------------------

Definition at line 301 of file [YPropertyEditor.cc](#).

### 5.107.3 Member Function Documentation

#### 5.107.3.1 `edit()`

```
bool YPropertyEditor::edit (
    const std::string & property )
```

Display a popup for editing a widget property.

##### Parameters

<i>property</i>	name of the property to edit
-----------------	------------------------------

##### Returns

true if the property has been changed, false otherwise

Definition at line [295](#) of file [YPropertyEditor.cc](#).

The documentation for this class was generated from the following files:

- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YPropertyEditor.h](#)
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YPropertyEditor.cc](#)

## 5.108 YPropertyEditorPriv Class Reference

### Public Member Functions

- **YPropertyEditorPriv** ([YWidget](#) \*widget)
- bool **edit** (const string &property)

#### 5.108.1 Detailed Description

Definition at line [40](#) of file [YPropertyEditor.cc](#).

The documentation for this class was generated from the following file:

- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YPropertyEditor.cc](#)

## 5.109 YPropertySet Class Reference

A set of properties to check names and types against.

```
#include <YProperty.h>
```

### Public Types

- typedef std::vector< [YProperty](#) >::const\_iterator **const\_iterator**

### Public Member Functions

- [YPropertySet](#) ()  
*Constructor.*
- void [check](#) (const std::string &propertyName) const  
*Check if a property 'propertyName' exists in this property set.*
- void [check](#) (const std::string &propertyName, YPropertyType type) const  
*Check if a property 'propertyName' exists in this property set.*
- void [check](#) (const [YProperty](#) &prop) const  
*Same as above, overloaded for convenience.*
- bool [contains](#) (const std::string &propertyName) const throw ()  
*Check if a property 'propertyName' exists in this property set.*
- bool [contains](#) (const std::string &propertyName, YPropertyType type) const  
*Check if a property 'propertyName' exists in this property set.*
- bool [contains](#) (const [YProperty](#) &prop) const  
*Same as above, overloaded for convenience.*
- bool [isEmpty](#) () const  
*Returns 'true' if this property set does not contain anything.*
- int [size](#) () const  
*Returns the number of properties in this set.*
- void [add](#) (const [YProperty](#) &prop)  
*Add a property to this property set.*
- void [add](#) (const [YPropertySet](#) &otherSet)  
*Adds all properties of another property set.*
- const\_iterator [propertiesBegin](#) () const  
*Returns an iterator that points to the first property in this set.*
- const\_iterator [propertiesEnd](#) () const  
*Returns an iterator that points after the last property in this set.*

### 5.109.1 Detailed Description

A set of properties to check names and types against.

Definition at line [197](#) of file [YProperty.h](#).

## 5.109.2 Member Function Documentation

### 5.109.2.1 add()

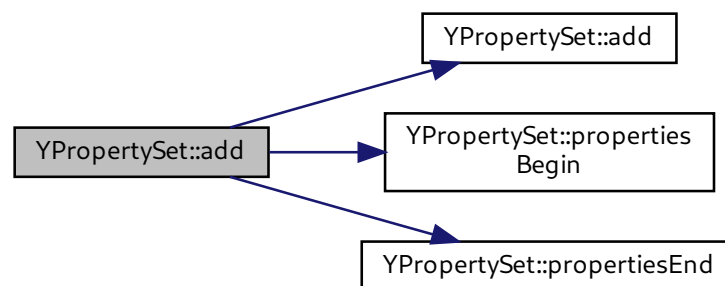
```
void YPropertySet::add (
    const YPropertySet & otherSet )
```

Adds all properties of another property set.

If that other set contains duplicates (properties that are already in this set), those others will never be found with lookup().

Definition at line 153 of file [YProperty.cc](#).

Here is the call graph for this function:



### 5.109.2.2 check() [1/2]

```
void YPropertySet::check (
    const std::string & propertyName ) const
```

Check if a property 'propertyName' exists in this property set.

Throw a [YUIUnknownPropertyException](#) if it does not exist. Use [YPropertySet::contains\(\)](#) for a check that simply returns 'false' if it does not exist.

Definition at line 88 of file [YProperty.cc](#).

Here is the call graph for this function:



### 5.109.2.3 check() [2/2]

```
void YPropertySet::check (
    const std::string & propertyName,
    YPropertyType type ) const
```

Check if a property 'propertyName' exists in this property set.

Throw a [YUIUnknownPropertyException](#) if it does not exist.

If there is a property with that name, check also the expected type against 'type'. If the types don't match, throw a [YUIPropertyTypeMismatchException](#). If the property is read-only, throw a [YUISetReadOnlyPropertyException](#).

Definition at line 96 of file [YProperty.cc](#).

Here is the call graph for this function:



### 5.109.2.4 contains() [1/2]

```
bool YPropertySet::contains (
    const std::string & propertyName ) const throw ( )
```

Check if a property 'propertyName' exists in this property set.

Returns 'true' if it exists, 'false' if not.

Use [YPropertySet::check\(\)](#) for a check that throws exceptions if there is no such property.

Definition at line 107 of file [YProperty.cc](#).

### 5.109.2.5 contains() [2/2]

```
bool YPropertySet::contains (
    const std::string & propertyName,
    YPropertyType type ) const
```

Check if a property 'propertyName' exists in this property set.

Returns 'true' if it exists, 'false' if not.

If there is a property with that name, check also the expected type against 'type'. If the types don't match, throw a [YUIPropertyTypeMismatchException](#).

If the property is read-only, throw a [YUISetReadOnlyPropertyException](#).

Use [YPropertySet::check\(\)](#) for a check that throws exceptions if there is no such property.

Definition at line 122 of file [YProperty.cc](#).

The documentation for this class was generated from the following files:

- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YProperty.h
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YProperty.cc

## 5.110 YPropertyValue Class Reference

Transport class for the value of simple properties.

```
#include <YProperty.h>
```

### Public Member Functions

- [YPropertyValue](#) (const std::string &str)  
*Constructor for string properties.*
- [YPropertyValue](#) (const char \*str)  
*Constructor for const char \* (string) properties.*
- [YPropertyValue](#) (bool b)  
*Constructor for bool properties.*
- [YPropertyValue](#) (YInteger num)  
*Constructor for numerical (YCP integer) properties.*
- [YPropertyValue](#) (int num)  
*Constructor for numerical (YCP integer) properties.*
- **YPropertyValue** (YPropertyType type)
- [YPropertyValue](#) ()  
*Default constructor.*
- [~YPropertyValue](#) ()  
*Destructor.*

- bool `operator==` (const [YPropertyValue](#) &other) const  
*Equality operator, can compare with another [YPropertyValue](#).*
- bool `operator!=` (const [YPropertyValue](#) &other) const  
*Inequality operator.*
- YPropertyType `type` () const  
*Returns the type of this property value.*
- std::string `typeAsStr` () const  
*Returns the type of this property value as string.*
- std::string `stringVal` () const  
*Methods to get the value of this property.*
- bool `boolVal` () const
- YInteger `integerVal` () const

### 5.110.1 Detailed Description

Transport class for the value of simple properties.

More complex properties (lists of items, tree descriptions, ...) have to be handled specifically someplace else, but most properties are of simple types and can be treated in similar ways.

Definition at line [104](#) of file [YProperty.h](#).

### 5.110.2 Member Function Documentation

#### 5.110.2.1 `operator"!=()`

```
bool YPropertyValue::operator!= (
    const YPropertyValue & other ) const
```

Inequality operator.

Exceptions

<a href="#">YUIException</a>	for incompatible property types
------------------------------	---------------------------------

See also

`operator==`

Definition at line [76](#) of file [YProperty.cc](#).

### 5.110.2.2 operator==()

```
bool YPropertyValue::operator== (
    const YPropertyValue & other ) const
```

Equality operator, can compare with another [YPropertyValue](#).

#### Exceptions

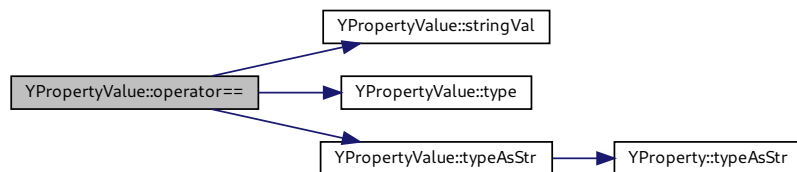
<a href="#">YUIException</a>	for incompatible property types
------------------------------	---------------------------------

#### Returns

true if the value is the same

Definition at line 54 of file [YProperty.cc](#).

Here is the call graph for this function:



### 5.110.2.3 stringVal()

```
std::string YPropertyValue::stringVal ( ) const [inline]
```

Methods to get the value of this property.

Check with [type\(\)](#) which one to use.

Definition at line 180 of file [YProperty.h](#).



#### 5.110.2.4 type()

```
YPropertyType YPropertyValue::type ( ) const [inline]
```

Returns the type of this property value.

Use this to determine which xyVal() method to use.

Definition at line 169 of file [YProperty.h](#).

The documentation for this class was generated from the following files:

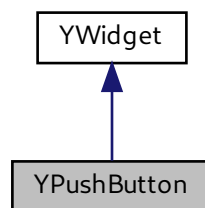
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YProperty.h](#)
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YProperty.cc](#)

## 5.111 YPushButton Class Reference

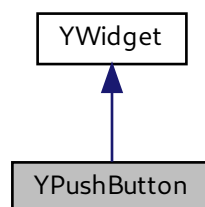
A push button; may have an icon, and a F-key shortcut.

```
#include <YPushButton.h>
```

Inheritance diagram for YPushButton:



Collaboration diagram for YPushButton:



## Public Member Functions

- virtual [~YPushButton](#) ()  
*Destructor.*
- virtual const char \* [widgetClass](#) () const  
*Return a descriptive name of this widget class for logging, debugging etc.*
- std::string [label](#) () const  
*Get the label (the text on the button).*
- virtual void [setLabel](#) (const std::string &label)  
*Set the label (the text on the button).*
- virtual void [setIcon](#) (const std::string &iconName)  
*Set this button's icon from an icon file in the UI's default icon directory.*
- bool [isDefaultButton](#) () const  
*Returns 'true' if this is the dialog's default button, i.e.*
- virtual void [setDefaultButton](#) (bool def=true)  
*Make this button the default button.*
- virtual void [setRole](#) (YButtonRole role)  
*Set a predefined role for this button.*
- YButtonRole [role](#) () const  
*Return the role of this button.*
- virtual void [setFunctionKey](#) (int fkey\_no)  
*Assign a function key to this widget (1 for F1, 2 for F2, etc.*
- bool [isHelpButton](#) () const  
*Returns 'true' if this is a "Help" button.*
- virtual void [setHelpButton](#) (bool helpButton=true)  
*Make this button a help button.*
- bool [isRelNotesButton](#) () const  
*Returns 'true' if this is a "Release Notes" button.*
- virtual void [setRelNotesButton](#) (bool relNotesButton=true)  
*Make this button a release notes button.*
- virtual bool [setProperty](#) (const std::string &propertyName, const [YPropertyValue](#) &val)  
*Set a property.*
- virtual [YPropertyValue](#) [getProperty](#) (const std::string &propertyName)  
*Get a property.*
- virtual const [YPropertySet](#) & [propertySet](#) ()  
*Return this class's property set.*
- virtual std::string [shortcutString](#) () const  
*Get the string of this widget that holds the keyboard shortcut.*
- virtual void [setShortcutString](#) (const std::string &str)  
*Set the string of this widget that holds the keyboard shortcut.*
- virtual void [activate](#) ()=0  
*Activate the button.*

## Protected Member Functions

- [YPushButton](#) ([YWidget](#) \*parent, const std::string &label)  
*Constructor.*

### 5.111.1 Detailed Description

A push button; may have an icon, and a F-key shortcut.

Definition at line 37 of file [YPushButton.h](#).

### 5.111.2 Member Function Documentation

#### 5.111.2.1 activate()

```
virtual void YPushButton::activate ( ) [pure virtual]
```

Activate the button.

Can be used in tests to simulate user input.

Derived classes are required to implement this.

#### 5.111.2.2 getProperty()

```
YPropertyValue YPushButton::getProperty (
    const std::string & propertyName ) [virtual]
```

Get a property.

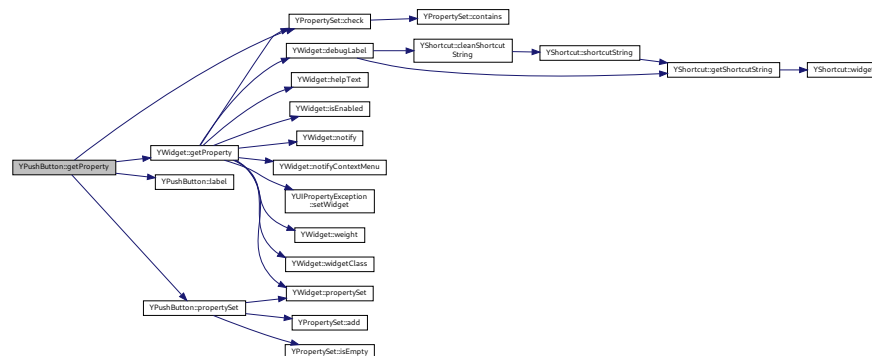
Reimplemented from [YWidget](#).

This method may throw [YUIPropertyExceptions](#).

Reimplemented from [YWidget](#).

Definition at line 245 of file [YPushButton.cc](#).

Here is the call graph for this function:



#### 5.111.2.3 isDefaultButton()

```
bool YPushButton::isDefaultButton ( ) const
```

Returns 'true' if this is the dialog's default button, i.e.

the one button that gets activated if the user hits the [Return] key anywhere in the dialog.

Definition at line 92 of file [YPushButton.cc](#).

#### 5.111.2.4 isHelpButton()

```
bool YPushButton::isHelpButton ( ) const
```

Returns 'true' if this is a "Help" button.

When activated, a help button will traverse up its widget hierarchy and search for the topmost widget with a [helpText\(\)](#) set and display that help text in a pop-up dialog (with a local event loop).

NOTE that this is only done during [YDialog::waitForEvent\(\)](#) (i.e. in YCP UI::WaitForEvent(), UI::UserInput(), UI::← TimeoutUserInput() ) and not during [YDialog::pollEvent\(\)](#) (i.e. YCP UI::PollInput()) since displaying the help text will block the application until the user closes the help text.

Definition at line 127 of file [YPushButton.cc](#).

#### 5.111.2.5 isRelNotesButton()

```
bool YPushButton::isRelNotesButton ( ) const
```

Returns 'true' if this is a "Release Notes" button.

NOTE that this is only done during [YDialog::waitForEvent\(\)](#) (i.e. in YCP UI::WaitForEvent(), UI::UserInput(), UI::← TimeoutUserInput() ) and not during [YDialog::pollEvent\(\)](#) (i.e. YCP UI::PollInput()) since displaying the release notes will block the application until the user closes the text.

Definition at line 139 of file [YPushButton.cc](#).

### 5.111.2.6 propertySet()

```
const YPropertySet & YPushButton::propertySet ( ) [virtual]
```

Return this class's property set.

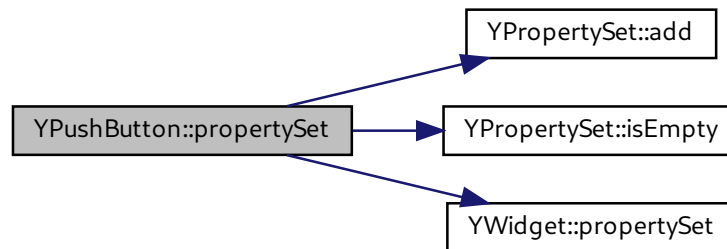
This also initializes the property upon the first call.

Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line [212](#) of file [YPushButton.cc](#).

Here is the call graph for this function:



### 5.111.2.7 setDefaultButton()

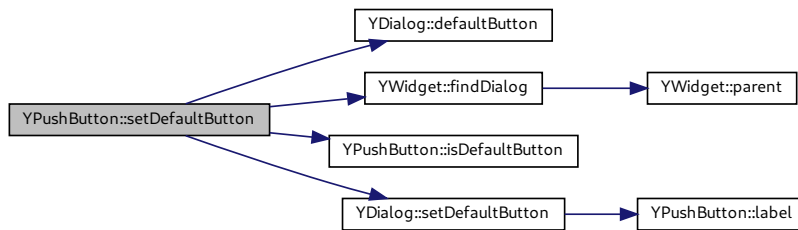
```
void YPushButton::setDefaultButton (
    bool def = true ) [virtual]
```

Make this button the default button.

Derived classes should reimplement this, but call this base class function in the overwritten function.

Definition at line [98](#) of file [YPushButton.cc](#).

Here is the call graph for this function:



#### 5.111.2.8 setFunctionKey()

```
void YPushButton::setFunctionKey (
    int fkey_no ) [virtual]
```

Assign a function key to this widget (1 for F1, 2 for F2, etc.

; 0 for none)

Reimplemented from [YWidget](#) to map function keys to button roles.

Derived classes may want to overwrite this function, but they should call this base class function in the new function.

Reimplemented from [YWidget](#).

Definition at line [186](#) of file [YPushButton.cc](#).

Here is the call graph for this function:



### 5.111.2.9 setHelpButton()

```
void YPushButton::setHelpButton (
    bool helpButton = true ) [virtual]
```

Make this button a help button.

Derived classes are free to reimplement this, but they should call this base class method in the overloaded function.

Definition at line 133 of file [YPushButton.cc](#).

### 5.111.2.10 setIcon()

```
virtual void YPushButton::setIcon (
    const std::string & iconName ) [inline], [virtual]
```

Set this button's icon from an icon file in the UI's default icon directory.

Clear the icon if the name is empty.

This default implementation does nothing. UIs that can handle icons can choose to overwrite this method.

Definition at line 77 of file [YPushButton.h](#).

### 5.111.2.11 setLabel()

```
void YPushButton::setLabel (
    const std::string & label ) [virtual]
```

Set the label (the text on the button).

Derived classes are free to reimplement this, but they should call this base class method at the end of the overloaded function.

Definition at line 80 of file [YPushButton.cc](#).

Here is the call graph for this function:



### 5.111.2.12 setProperty()

```
bool YPushButton::setProperty (
    const std::string & propertyName,
    const YPropertyValue & val ) [virtual]
```

Set a property.

Reimplemented from [YWidget](#).

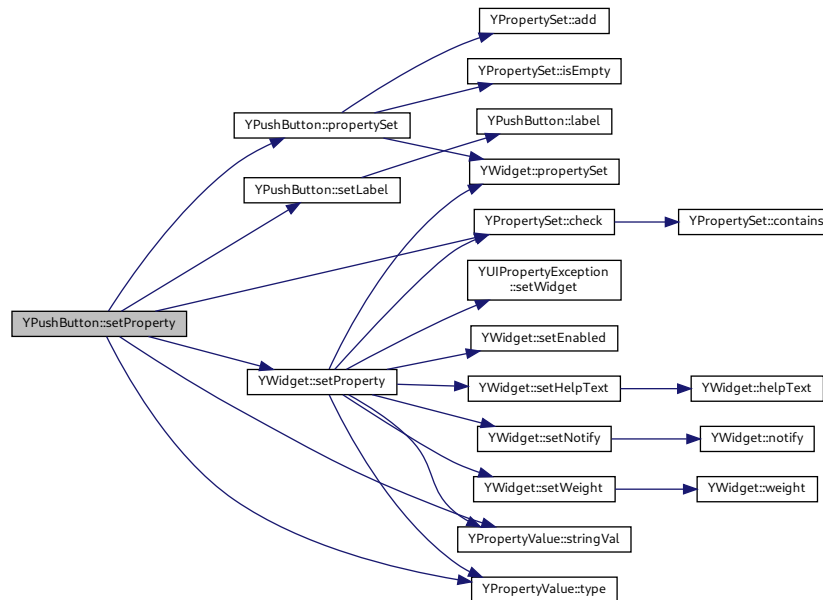
This function may throw YUIPropertyExceptions.

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from [YWidget](#).

Definition at line 230 of file [YPushButton.cc](#).

Here is the call graph for this function:



### 5.111.2.13 setRelNotesButton()

```
void YPushButton::setRelNotesButton (
    bool relNotesButton = true ) [virtual]
```

Make this button a release notes button.

Derived classes are free to reimplement this, but they should call this base class method in the overloaded function.

Definition at line 145 of file [YPushButton.cc](#).



#### 5.111.2.14 setRole()

```
void YPushButton::setRole (
    YButtonRole role ) [virtual]
```

Set a predefined role for this button.

This is important when the button is a child of a [YButtonBox](#) so the layout can be arranged according to the conventions of the current UI or desktop environment.

See [YButtonBox.h](#) for more details. YButtonRole is defined in [YTypes.h](#)

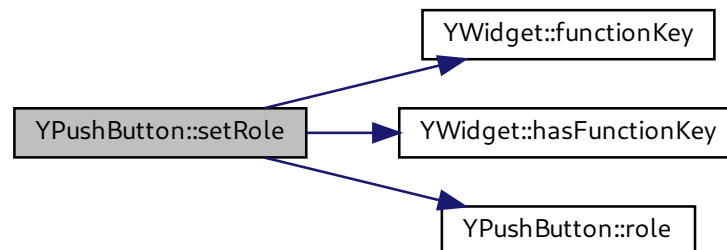
The default is YCustomButton, i.e., no predefined role. [setFunctionKey\(\)](#) uses some heuristics to map function keys to buttons:

```
F10 -> YOkButton
F9  -> YCancelButton
F1  -> YHelpButton
```

Derived classes are free to reimplement this, but they should call this base class function in the overwritten function.

Definition at line 154 of file [YPushButton.cc](#).

Here is the call graph for this function:



### 5.111.2.15 setShortcutString()

```
virtual void YPushButton::setShortcutString (
    const std::string & str ) [inline], [virtual]
```

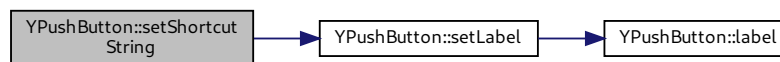
Set the string of this widget that holds the keyboard shortcut.

Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 215 of file [YPushButton.h](#).

Here is the call graph for this function:



### 5.111.2.16 shortcutString()

```
virtual std::string YPushButton::shortcutString ( ) const [inline], [virtual]
```

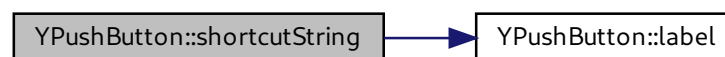
Get the string of this widget that holds the keyboard shortcut.

Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 208 of file [YPushButton.h](#).

Here is the call graph for this function:

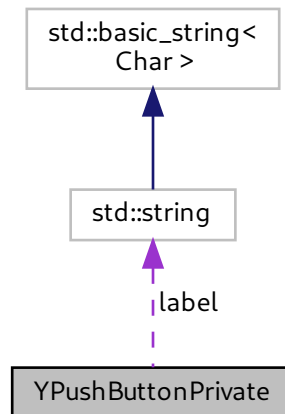


The documentation for this class was generated from the following files:

- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YPushButton.h`
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YPushButton.cc`

## 5.112 YPushButtonPrivate Struct Reference

Collaboration diagram for YPushButtonPrivate:



### Public Member Functions

- **YPushButtonPrivate** (const string &label)

### Public Attributes

- string **label**
- bool **isDefaultButton**
- bool **setDefaultButtonRecursive**
- bool **isHelpButton**
- bool **isRelNotesButton**
- YButtonRole **role**

### 5.112.1 Detailed Description

Definition at line 38 of file [YPushButton.cc](#).

The documentation for this struct was generated from the following file:

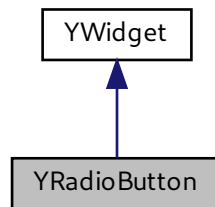
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YPushButton.cc`

## 5.113 YRadioButton Class Reference

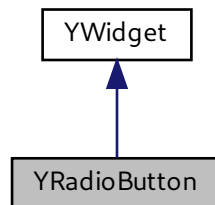
RadioButton: Widget for one-out-of-many selection.

```
#include <YRadioButton.h>
```

Inheritance diagram for YRadioButton:



Collaboration diagram for YRadioButton:



### Public Member Functions

- virtual `~YRadioButton()`  
*Destructor: Removes the button from the radio button group.*
- virtual const char \* `widgetClass()` const  
*Returns a descriptive name of this widget class for logging, debugging etc.*
- virtual bool `value()`=0  
*Get the current on/off value: 'true' if checked, 'false' if unchecked.*
- virtual void `setValue` (bool checked)=0  
*Set the radio button value (on/off).*

- `std::string label () const`  
*Get the label (the text on the RadioButton).*
- `virtual void setLabel (const std::string &label)`  
*Set the label (the text on the RadioButton).*
- `bool useBoldFont () const`  
*Returns 'true' if a bold font should be used.*
- `virtual void setUseBoldFont (bool bold=true)`  
*Indicate whether or not a bold font should be used.*
- `YRadioButtonGroup * buttonGroup ()`  
*Get a pointer to the radio button group this button belongs to.*
- `virtual bool setProperty (const std::string &propertyName, const YPropertyValue &val)`  
*Set a property.*
- `virtual YPropertyValue getProperty (const std::string &propertyName)`  
*Get a property.*
- `virtual const YPropertySet & propertySet ()`  
*Return this class's property set.*
- `virtual std::string shortcutString () const`  
*Get the string of this widget that holds the keyboard shortcut.*
- `virtual void setShortcutString (const std::string &str)`  
*Set the string of this widget that holds the keyboard shortcut.*
- `const char * userInputProperty ()`  
*The name of the widget property that will return user input.*

## Protected Member Functions

- `YRadioButton (YWidget *parent, const std::string &label)`  
*Constructor.*
- `YRadioButtonGroup * findRadioButtonGroup () const`  
*Traverse the widget hierarchy upwards to find the corresponding YRadioButtonGroup, i.e.*
- `virtual void saveUserInput (YMacroRecorder *macroRecorder)`  
*Save the widget's user input to a macro recorder.*

### 5.113.1 Detailed Description

RadioButton: Widget for one-out-of-many selection.

Only one RadioButton in a RadioBox (in a RadioButtonGroup) can be set to "on" at the same time. Setting any RadioButton of a RadioButtonGroup to "on" automatically sets all others in the same RadioButtonGroup to "off".

RadioButtons customarily have a distinct visual appearance from CheckBoxes:

```
( ) RadioButton 1
(*) RadioButton 2
( ) RadioButton 3

[ ] CheckBox 1
[*] CheckBox 2
[*] CheckBox 3
```

Definition at line 51 of file [YRadioButton.h](#).

## 5.113.2 Constructor & Destructor Documentation

### 5.113.2.1 YRadioButton()

```
YRadioButton::YRadioButton (
    YWidget * parent,
    const std::string & label ) [protected]
```

Constructor.

Creates a new RadioButton with user-visible text 'label'. 'label' can and should contain a keyboard shortcut (designated with '&').

The caller has to take care to add this RadioButton to its RadioButtonGroup:

```
if ( radioButton->buttonGroup() ) radioButton->buttonGroup()->addRadioButton( radioButton );
```

This can't be done in the constructor because it would involve calling a virtual function, which doesn't work yet within the constructor.

Definition at line 61 of file [YRadioButton.cc](#).

## 5.113.3 Member Function Documentation

### 5.113.3.1 findRadioButtonGroup()

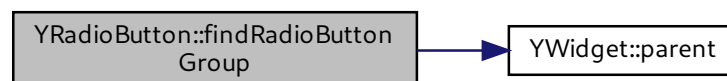
```
YRadioButtonGroup * YRadioButton::findRadioButtonGroup ( ) const [protected]
```

Traverse the widget hierarchy upwards to find the corresponding [YRadioButtonGroup](#), i.e.

the class that controls the radio box behaviour (i.e. that makes sure that no more than one RadioButton is set to "on" at the same time).

Definition at line 176 of file [YRadioButton.cc](#).

Here is the call graph for this function:



### 5.113.3.2 getProperty()

```
YPropertyValue YRadioButton::getProperty (
    const std::string & propertyName ) [virtual]
```

Get a property.

Reimplemented from [YWidget](#).

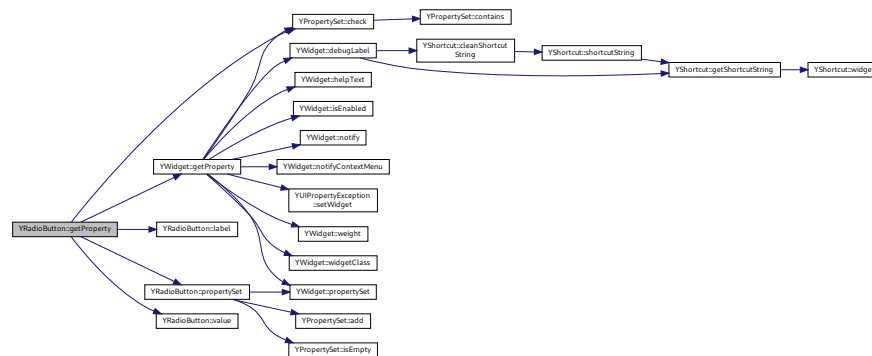
This method may throw exceptions, for example

- if there is no property with that name

Reimplemented from [YWidget](#).

Definition at line 150 of file [YRadioButton.cc](#).

Here is the call graph for this function:



### 5.113.3.3 propertySet()

```
const YPropertySet & YRadioButton::propertySet ( ) [virtual]
```

Return this class's property set.

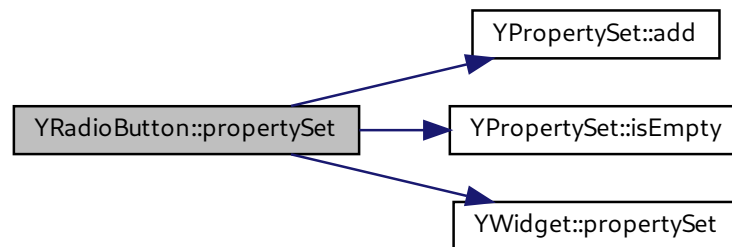
This also initializes the property set upon the first call.

Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 113 of file [YRadioButton.cc](#).

Here is the call graph for this function:



#### 5.113.3.4 `saveUserInput()`

```
void YRadioButton::saveUserInput (
    YMacroRecorder * macroRecorder ) [protected], [virtual]
```

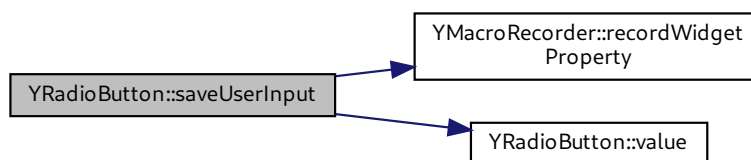
Save the widget's user input to a macro recorder.

Reimplemented from [YWidget](#) because only radio buttons that are on (no more than one per radio box) are recorded.

Reimplemented from [YWidget](#).

Definition at line 195 of file [YRadioButton.cc](#).

Here is the call graph for this function:





### 5.113.3.5 setLabel()

```
void YRadioButton::setLabel (
    const std::string & label ) [virtual]
```

Set the label (the text on the RadioButton).

Derived classes are free to reimplement this, but they should call this base class method at the end of the overloaded function.

Definition at line 88 of file [YRadioButton.cc](#).

### 5.113.3.6 setProperty()

```
bool YRadioButton::setProperty (
    const std::string & propertyName,
    const YPropertyValue & val ) [virtual]
```

Set a property.

Reimplemented from [YWidget](#).

This method may throw exceptions, for example

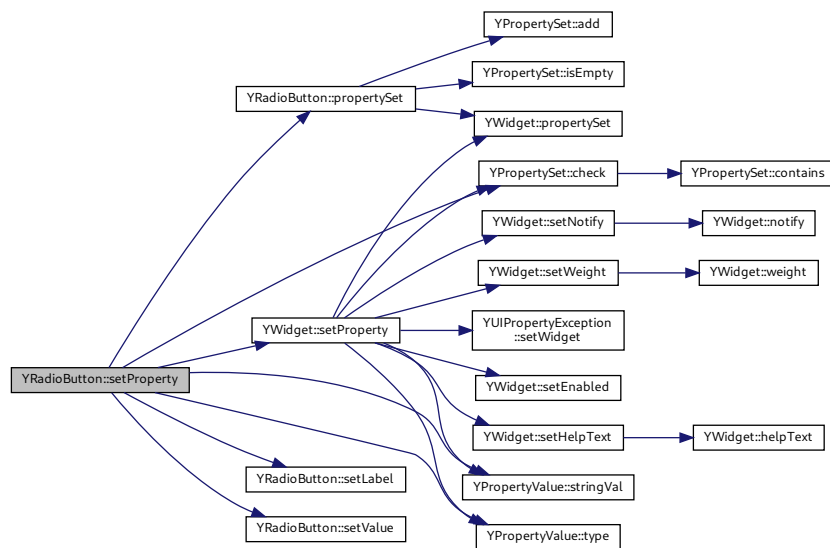
- if there is no property with that name
- if the expected type and the type mismatch
- if the value is out of range

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from [YWidget](#).

Definition at line 134 of file [YRadioButton.cc](#).

Here is the call graph for this function:



### 5.113.3.7 setShortcutString()

```
virtual void YRadioButton::setShortcutString (
    const std::string & str ) [inline], [virtual]
```

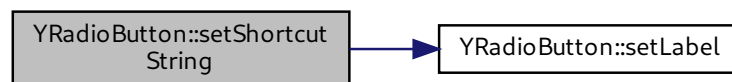
Set the string of this widget that holds the keyboard shortcut.

Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 177 of file [YRadioButton.h](#).

Here is the call graph for this function:



### 5.113.3.8 setUseBoldFont()

```
void YRadioButton::setUseBoldFont (
    bool bold = true ) [virtual]
```

Indicate whether or not a bold font should be used.

Derived classes are free to reimplement this, but they should call this base class method at the end of the overloaded function.

Definition at line 106 of file [YRadioButton.cc](#).

### 5.113.3.9 setValue()

```
virtual void YRadioButton::setValue (
    bool checked ) [pure virtual]
```

Set the radio button value (on/off).

Derived classes are required to implement this.

#### 5.113.3.10 shortcutString()

```
virtual std::string YRadioButton::shortcutString ( ) const [inline], [virtual]
```

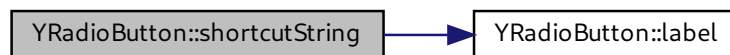
Get the string of this widget that holds the keyboard shortcut.

Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 170 of file [YRadioButton.h](#).

Here is the call graph for this function:



#### 5.113.3.11 userInputProperty()

```
const char* YRadioButton::userInputProperty ( ) [inline], [virtual]
```

The name of the widget property that will return user input.

Inherited from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 184 of file [YRadioButton.h](#).

#### 5.113.3.12 value()

```
virtual bool YRadioButton::value ( ) [pure virtual]
```

Get the current on/off value: 'true' if checked, 'false' if unchecked.

Derived classes are required to implement this.

### 5.113.3.13 widgetClass()

```
virtual const char* YRadioButton::widgetClass ( ) const [inline], [virtual]
```

Returns a descriptive name of this widget class for logging, debugging etc.

Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 84 of file [YRadioButton.h](#).

The documentation for this class was generated from the following files:

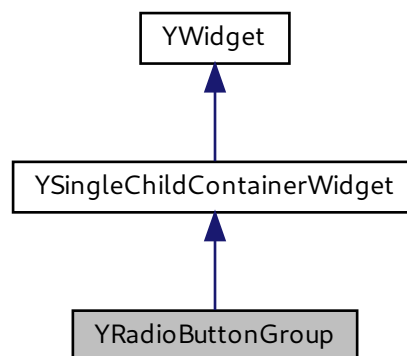
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YRadioButton.h](#)
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YRadioButton.cc](#)

## 5.114 YRadioButtonGroup Class Reference

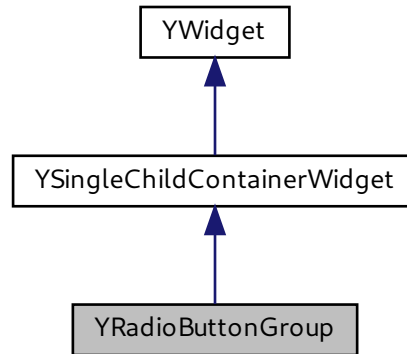
A group of [YRadioButton](#) widgets.

```
#include <YRadioButtonGroup.h>
```

Inheritance diagram for YRadioButtonGroup:



Collaboration diagram for YRadioButtonGroup:



## Public Member Functions

- virtual [~YRadioButtonGroup](#) ()  
*Destructor.*
- virtual const char \* [widgetClass](#) () const  
*Returns a descriptive name of this widget class for logging, debugging etc.*
- [YRadioButton](#) \* [currentButton](#) () const  
*Find the currently selected button.*
- [YRadioButton](#) \* [value](#) () const  
*The same as [currentButton\(\)](#) above for convenience.*
- virtual void [addRadioButton](#) ([YRadioButton](#) \*radioButton)  
*Add a RadioButton to this button group.*
- virtual void [removeRadioButton](#) ([YRadioButton](#) \*radioButton)  
*Remove a RadioButton from this button group.*
- void [uncheckOtherButtons](#) ([YRadioButton](#) \*radioButton)  
*Unchecks all radio buttons except one.*
- virtual bool [setProperty](#) (const std::string &propertyName, const [YPropertyValue](#) &val)  
*Set a property.*
- virtual [YPropertyValue](#) [getProperty](#) (const std::string &propertyName)  
*Get a property.*
- virtual const [YPropertySet](#) & [propertySet](#) ()  
*Return this class's property set.*

## Protected Member Functions

- [YRadioButtonGroup](#) ([YWidget](#) \*parent)  
*Constructor.*
- [YRadioButtonListConstIterator](#) [radioButtonsBegin](#) () const  
*Return an iterator that points to the first RadioButton of this button group.*
- [YRadioButtonListConstIterator](#) [radioButtonsEnd](#) () const  
*Return an iterator that points behind the last RadioButton of this button group.*
- [int](#) [radioButtonsCount](#) () const  
*Return the number of RadioButtons in this button group.*

### 5.114.1 Detailed Description

A group of [YRadioButton](#) widgets.

Definition at line 41 of file [YRadioButtonGroup.h](#).

### 5.114.2 Member Function Documentation

#### 5.114.2.1 addRadioButton()

```
void YRadioButtonGroup::addRadioButton (  
    YRadioButton * radioButton ) [virtual]
```

Add a RadioButton to this button group.

RadioButtons are required to call this in their constructor.

Derived classes are free to overload this, but they should call this base class function in the overloaded function.

Definition at line 83 of file [YRadioButtonGroup.cc](#).

### 5.114.2.2 getProperty()

```
YPropertyValue YRadioButtonGroup::getProperty (
    const std::string & propertyName ) [virtual]
```

Get a property.

Reimplemented from [YWidget](#).

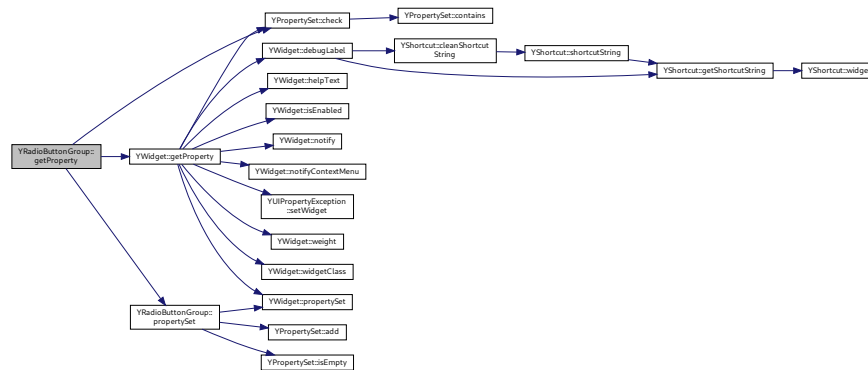
This method may throw exceptions, for example

- if there is no property with that name

Reimplemented from [YWidget](#).

Definition at line 161 of file [YRadioButtonGroup.cc](#).

Here is the call graph for this function:



### 5.114.2.3 propertySet()

```
const YPropertySet & YRadioButtonGroup::propertySet ( ) [virtual]
```

Return this class's property set.

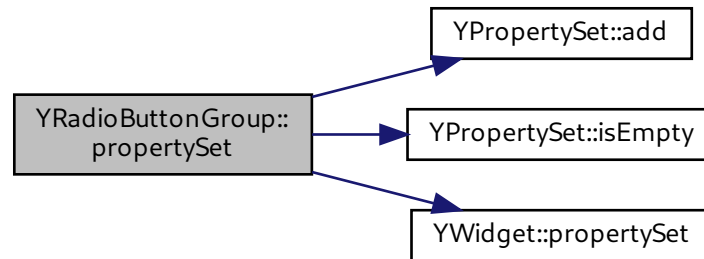
This also initializes the property set upon the first call.

Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 125 of file [YRadioButtonGroup.cc](#).

Here is the call graph for this function:



#### 5.114.2.4 `radioButtonsBegin()`

```
YRadioButtonListConstIterator YRadioButtonGroup::radioButtonsBegin ( ) const [protected]
```

Return an iterator that points to the first `RadioButton` of this button group.

Note that `RadioButtons` in this group may be direct or indirect children of the group, so don't confuse this with `YWidget::widgetsBegin()`.

Definition at line 62 of file [YRadioButtonGroup.cc](#).

#### 5.114.2.5 `removeRadioButton()`

```
void YRadioButtonGroup::removeRadioButton (
    YRadioButton * radioButton ) [virtual]
```

Remove a `RadioButton` from this button group.

`RadioButtons` are required to call this in their destructor, but only if the button group is not also in the process of being destroyed (otherwise there may be race conditions with child widgets already destroyed):

```
if ( ! buttonGroup()->beingDestroyed )
    buttonGroup()->removeRadioButton( this );
```

Definition at line 90 of file [YRadioButtonGroup.cc](#).



## 5.114.2.6 setProperty()

```
bool YRadioButtonGroup::setProperty (
    const std::string & propertyName,
    const YPropertyValue & val ) [virtual]
```

Set a property.

Reimplemented from [YWidget](#).

This method may throw exceptions, for example

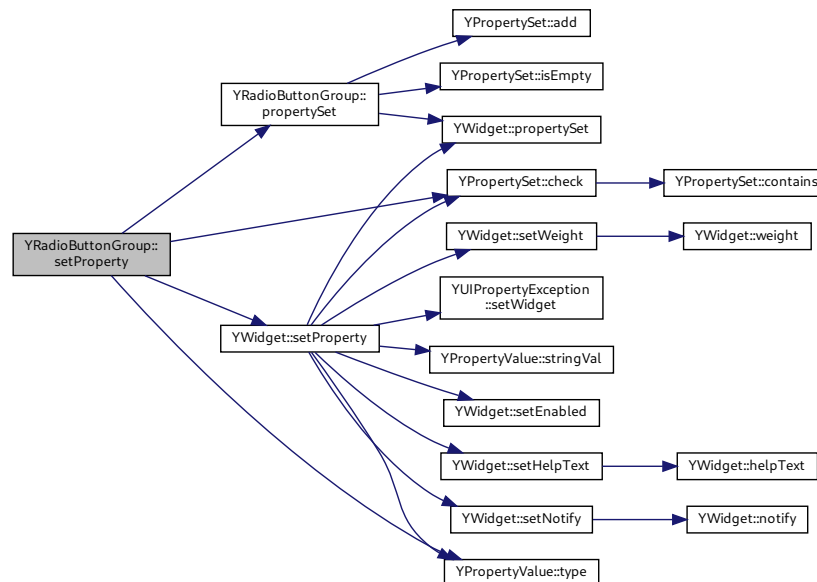
- if there is no property with that name
- if the expected type and the type mismatch
- if the value is out of range

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from [YWidget](#).

Definition at line 145 of file [YRadioButtonGroup.cc](#).

Here is the call graph for this function:



### 5.114.2.7 `uncheckOtherButtons()`

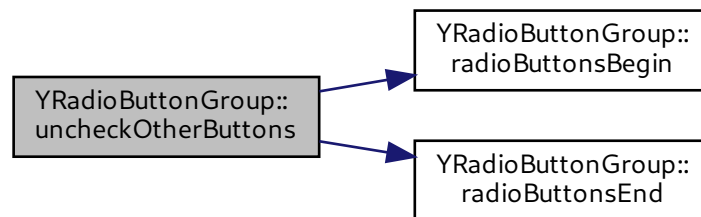
```
void YRadioButtonGroup::uncheckOtherButtons (
    YRadioButton * radioButton )
```

Unchecks all radio buttons except one.

This method can be used by a concrete UI (the Qt UI or the NCurses UI) in the implementation of `YRadioButton::setValue()`.

Definition at line 97 of file `YRadioButtonGroup.cc`.

Here is the call graph for this function:

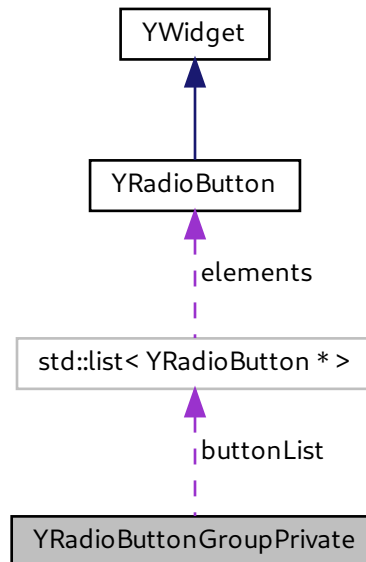


The documentation for this class was generated from the following files:

- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YRadioButtonGroup.h`
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YRadioButtonGroup.cc`

## 5.115 YRadioButtonGroupPrivate Struct Reference

Collaboration diagram for YRadioButtonGroupPrivate:



### Public Attributes

- YRadioButtonList **buttonList**

### 5.115.1 Detailed Description

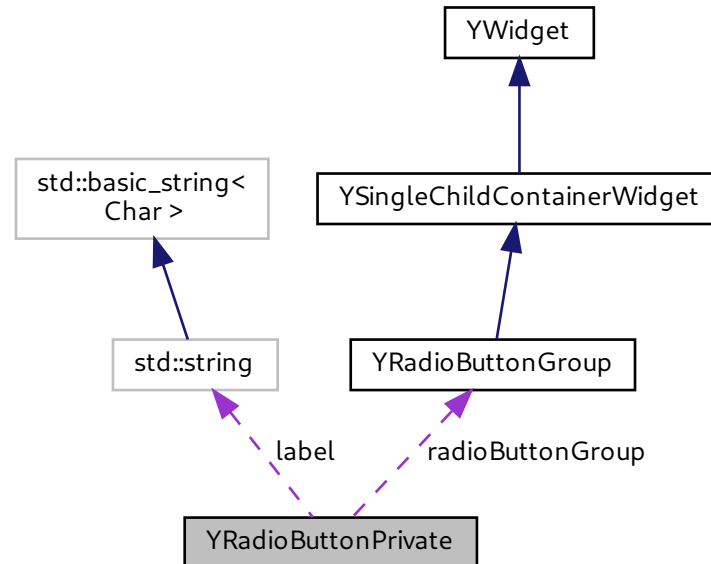
Definition at line 36 of file [YRadioButtonGroup.cc](#).

The documentation for this struct was generated from the following file:

- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YRadioButtonGroup.cc

## 5.116 YRadioButtonPrivate Struct Reference

Collaboration diagram for YRadioButtonPrivate:



### Public Member Functions

- [YRadioButtonPrivate](#) (const string &label)  
*Constructor.*

### Public Attributes

- string **label**
- [YRadioButtonGroup](#) \* **radioButtonGroup**
- bool **useBoldFont**

### 5.116.1 Detailed Description

Definition at line 40 of file [YRadioButton.cc](#).

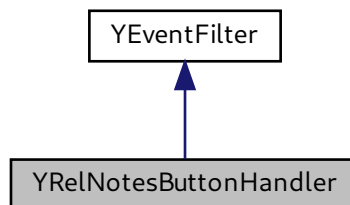
The documentation for this struct was generated from the following file:

- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YRadioButton.cc

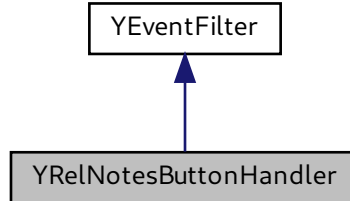
## 5.117 YRelNotesButtonHandler Class Reference

Helper class: Event filter that handles "ReleaseNotes" buttons.

Inheritance diagram for YRelNotesButtonHandler:



Collaboration diagram for YRelNotesButtonHandler:



### Public Member Functions

- **YRelNotesButtonHandler** ([YDialog](#) \*dialog)
- [YEvent](#) \* filter ([YEvent](#) \*event)

*The heart of the matter: The event filter function.*

### Additional Inherited Members

#### 5.117.1 Detailed Description

Helper class: Event filter that handles "ReleaseNotes" buttons.

Definition at line 105 of file [YDialog.cc](#).

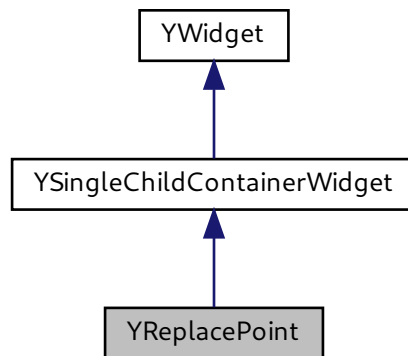


## 5.118 YReplacePoint Class Reference

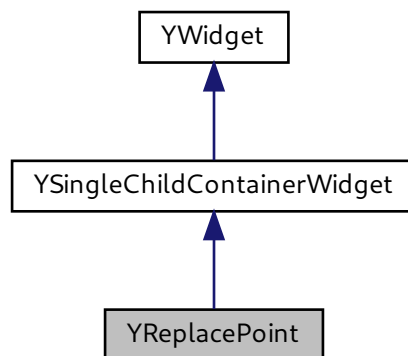
A placeholder that can have its contents exchanged, using ReplaceWidget.

```
#include <YReplacePoint.h>
```

Inheritance diagram for YReplacePoint:



Collaboration diagram for YReplacePoint:



## Public Member Functions

- virtual void [showChild](#) ()  
*Show a newly added child.*
- virtual const char \* [widgetClass](#) () const  
*Returns a descriptive name of this widget class for logging, debugging etc.*

## Protected Member Functions

- [YReplacePoint](#) (YWidget \*parent)  
*Constructor.*

### 5.118.1 Detailed Description

A placeholder that can have its contents exchanged, using [ReplaceWidget](#).

Definition at line [33](#) of file [YReplacePoint.h](#).

### 5.118.2 Member Function Documentation

#### 5.118.2.1 showChild()

```
void YReplacePoint::showChild ( ) [virtual]
```

Show a newly added child.

The application using the [ReplacePoint](#) is required to call this after the new child is created.

This cannot be done in the child widget's constructor (e.g., by overwriting [YWidget::addChild\(\)](#)) since at that point [YWidget::widgetRep\(\)](#) may or may not be initialized yet.

This default implementation does nothing. Derived classes should reimplement this to make new child widgets visible.

Definition at line [35](#) of file [YReplacePoint.cc](#).

The documentation for this class was generated from the following files:

- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YReplacePoint.h](#)
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YReplacePoint.cc](#)

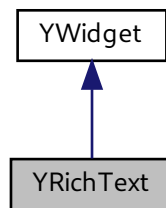


## 5.119 YRichText Class Reference

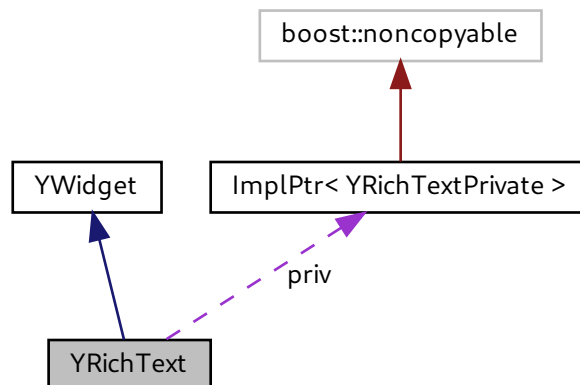
Text formatted with simple HTML-like tags, with "links" generating events.

```
#include <YRichText.h>
```

Inheritance diagram for YRichText:



Collaboration diagram for YRichText:



### Public Member Functions

- [YRichText](#) ([YWidget](#) \*[parent](#), const std::string &[text](#), bool [plainTextMode](#)=false)  
*Constructor.*
- virtual [~YRichText](#) ()

*Destructor.*

- virtual const char \* **widgetClass** () const  
*Returns a descriptive name of this widget class for logging, debugging etc.*
- virtual void **setValue** (const std::string &newValue)  
*Change the text content of the RichText widget.*
- std::string **value** () const  
*Return the text content of the RichText widget.*
- void **setText** (const std::string &newText)  
*Alias for setValue().*
- std::string **text** () const  
*Alias for value().*
- bool **plainTextMode** () const  
*Return 'true' if this RichText widget is in "plain text" mode, i.e.*
- virtual void **setPlainTextMode** (bool on=true)  
*Set this RichText widget's "plain text" mode on or off.*
- bool **autoScrollDown** () const  
*Return 'true' if this RichText widget should automatically scroll down when the text content is changed.*
- virtual void **setAutoScrollDown** (bool on=true)  
*Set this RichText widget's "auto scroll down" mode on or off.*
- bool **shrinkable** () const  
*Returns 'true' if this widget is "shrinkable", i.e.*
- void **setShrinkable** (bool shrinkable=true)  
*Make this widget shrinkable, i.e.*
- virtual bool **setProperty** (const std::string &propertyName, const YPropertyValue &val)  
*Set a property.*
- virtual YPropertyValue **getProperty** (const std::string &propertyName)  
*Get a property.*
- virtual const YPropertySet & **propertySet** ()  
*Return this class's property set.*
- virtual std::string **vScrollValue** () const  
*Get the position value of the vertical scrollbar.*
- virtual void **setVScrollValue** (const std::string &newValue)  
*Set the position value of the vertical scrollbar.*
- virtual std::string **hScrollValue** () const  
*Get the position value of the horizontal scrollbar.*
- virtual void **setHScrollValue** (const std::string &newValue)  
*Set the position value of the horizontal scrollbar.*
- virtual void **activateLink** (const std::string &url)=0  
*Derived classes should implement this, method is used to trigger event like user has pressed the link in the RichText.*

## Protected Attributes

- **ImplPtr**< YRichTextPrivate > **priv**

## Additional Inherited Members

### 5.119.1 Detailed Description

Text formatted with simple HTML-like tags, with "links" generating events.

Definition at line 40 of file [YRichText.h](#).

### 5.119.2 Constructor & Destructor Documentation

#### 5.119.2.1 YRichText()

```
YRichText::YRichText (
    YWidget * parent,
    const std::string & text,
    bool plainTextMode = false )
```

Constructor.

'plainTextMode' indicates that the text should be treated as plain text, i.e. any HTML-like tags in the text should not be interpreted in any way.

Definition at line 57 of file [YRichText.cc](#).

Here is the call graph for this function:



### 5.119.3 Member Function Documentation

### 5.119.3.1 autoScrollDown()

```
bool YRichText::autoScrollDown ( ) const
```

Return 'true' if this RichText widget should automatically scroll down when the text content is changed.

This is useful for progress displays and log files.

Definition at line 98 of file [YRichText.cc](#).

### 5.119.3.2 getProperty()

```
YPropertyValue YRichText::getProperty (
    const std::string & propertyName ) [virtual]
```

Get a property.

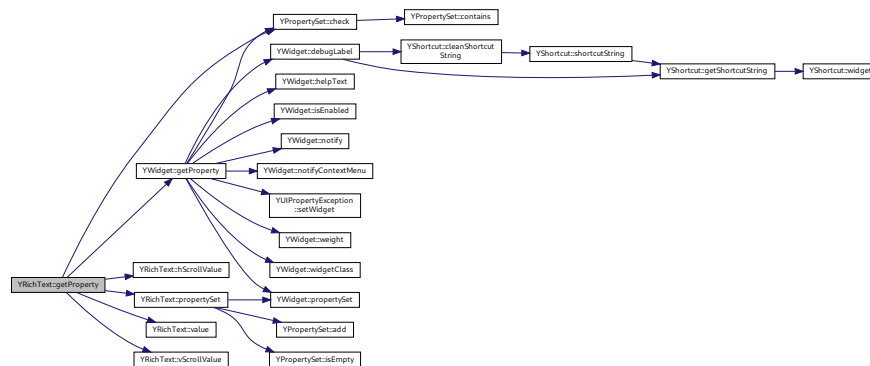
Reimplemented from [YWidget](#).

This method may throw YUIPropertyExceptions.

Reimplemented from [YWidget](#).

Definition at line 165 of file [YRichText.cc](#).

Here is the call graph for this function:



### 5.119.3.3 hScrollValue()

```
std::string YRichText::hScrollValue ( ) const [virtual]
```

Get the position value of the horizontal scrollbar.

Might not be available in all frontends.

Definition at line 191 of file [YRichText.cc](#).

### 5.119.3.4 plainTextMode()

```
bool YRichText::plainTextMode ( ) const
```

Return 'true' if this RichText widget is in "plain text" mode, i.e.

does not try to interpret RichText/HTML tags.

Definition at line 86 of file [YRichText.cc](#).

### 5.119.3.5 propertySet()

```
const YPropertySet & YRichText::propertySet ( ) [virtual]
```

Return this class's property set.

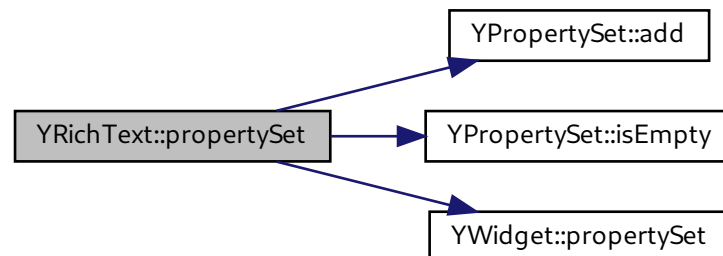
This also initializes the property upon the first call.

Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 123 of file [YRichText.cc](#).

Here is the call graph for this function:



### 5.119.3.6 setAutoScrollDown()

```
void YRichText::setAutoScrollDown (
    bool on = true ) [virtual]
```

Set this RichText widget's "auto scroll down" mode on or off.

Derived classes may want to reimplement this, but they should call this base class function in the new function.

Definition at line 104 of file [YRichText.cc](#).

Here is the call graph for this function:



### 5.119.3.7 setHScrollValue()

```
void YRichText::setHScrollValue (
    const std::string & newValue ) [virtual]
```

Set the position value of the horizontal scrollbar.

Only values previously obtained using [hScrollValue\(\)](#) and some special values are allowed.

The special values are:

"minimum" Moves the scrollbar to the start.

"maximum" Moves the scrollbar to the end.

The meaning of start and end can depend on the text direction (LTR or RTL).

Might not be available in all frontends.

Definition at line 197 of file [YRichText.cc](#).

### 5.119.3.8 setPlainTextMode()

```
void YRichText::setPlainTextMode (
    bool on = true ) [virtual]
```

Set this RichText widget's "plain text" mode on or off.

Derived classes may want to reimplement this, but they should call this base class function in the new function.

Definition at line 92 of file [YRichText.cc](#).

Here is the call graph for this function:



### 5.119.3.9 setProperty()

```
bool YRichText::setProperty (
    const std::string & propertyName,
    const YPropertyValue & val ) [virtual]
```

Set a property.

Reimplemented from [YWidget](#).

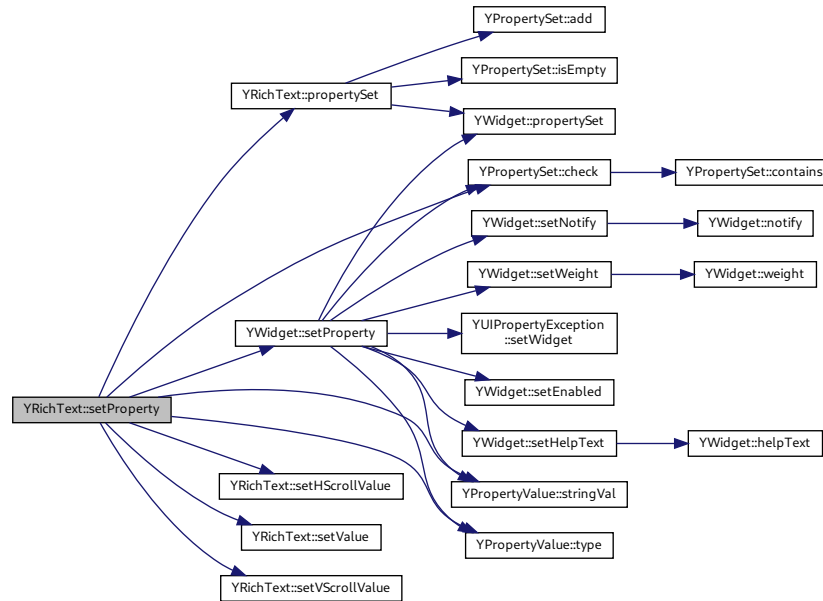
This function may throw `YUIPropertyExceptions`.

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from [YWidget](#).

Definition at line 147 of file [YRichText.cc](#).

Here is the call graph for this function:



#### 5.119.3.10 setShrinkable()

```
void YRichText::setShrinkable (
    bool shrinkable = true )
```

Make this widget shrinkable, i.e.

very small in layouts.

This method is intentionally not virtual because it doesn't have any immediate effect; it is only needed in [preferredWidth\(\)](#) / [preferredHeight\(\)](#).

Definition at line 116 of file [YRichText.cc](#).

Here is the call graph for this function:





#### 5.119.3.11 setValue()

```
void YRichText::setValue (
    const std::string & newValue ) [virtual]
```

Change the text content of the RichText widget.

Derived classes should overwrite this function, but call this base class function in the new function.

Definition at line 74 of file [YRichText.cc](#).

#### 5.119.3.12 setVScrollValue()

```
void YRichText::setVScrollValue (
    const std::string & newValue ) [virtual]
```

Set the position value of the vertical scrollbar.

Only values previously obtained using [vScrollValue\(\)](#) and some special values are allowed.

The special values are:

"minimum" Moves the scrollbar to the start.

"maximum" Moves the scrollbar to the end.

Might not be available in all frontends.

Definition at line 186 of file [YRichText.cc](#).

#### 5.119.3.13 shrinkable()

```
bool YRichText::shrinkable ( ) const
```

Returns 'true' if this widget is "shrinkable", i.e.

it should be very small by default.

Definition at line 110 of file [YRichText.cc](#).

#### 5.119.3.14 vScrollValue()

```
std::string YRichText::vScrollValue ( ) const [virtual]
```

Get the position value of the vertical scrollbar.

Might not be available in all frontends.

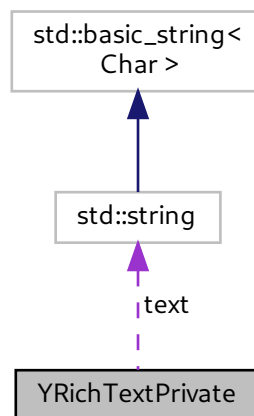
Definition at line 180 of file [YRichText.cc](#).

The documentation for this class was generated from the following files:

- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YRichText.h
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YRichText.cc

## 5.120 YRichTextPrivate Struct Reference

Collaboration diagram for YRichTextPrivate:



### Public Member Functions

- [YRichTextPrivate](#) (const string &text, bool plainTextMode)  
*Constructor.*

## Public Attributes

- string **text**
- bool **plainTextMode**
- bool **autoScrollDown**
- bool **shrinkable**

### 5.120.1 Detailed Description

Definition at line 36 of file [YRichText.cc](#).

The documentation for this struct was generated from the following file:

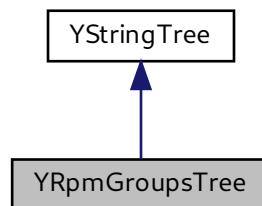
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YRichText.cc

## 5.121 YRpmGroupsTree Class Reference

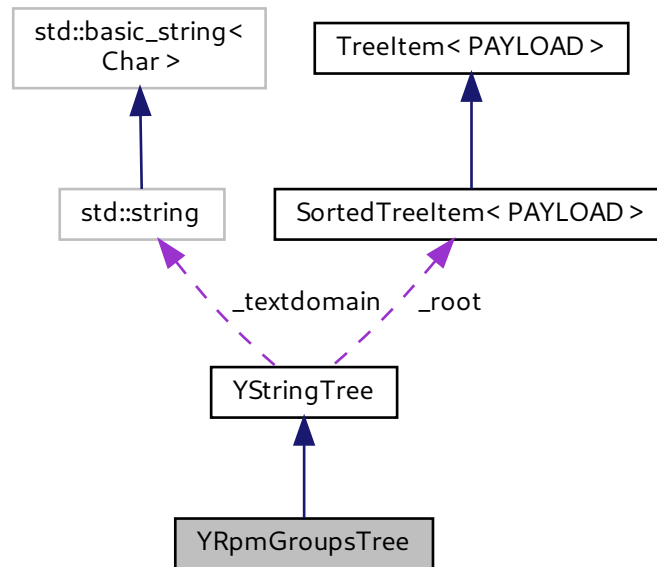
This was an efficient storage for RPM group tags.

```
#include <YRpmGroupsTree.h>
```

Inheritance diagram for YRpmGroupsTree:



Collaboration diagram for YRpmGroupsTree:



## Public Member Functions

- [YStringTreeItem](#) \* **addRpmGroup** (const std::string &rpmGroup)
- std::string **rpmGroup** (const [YStringTreeItem](#) \*node)
- std::string **translatedRpmGroup** (const [YStringTreeItem](#) \*node)
- void **addFallbackRpmGroups** ()

## Additional Inherited Members

### 5.121.1 Detailed Description

This was an efficient storage for RPM group tags.

THIS CLASS IS OBSOLETE. DO NOT USE THIS ANYMORE.

Definition at line 41 of file [YRpmGroupsTree.h](#).

The documentation for this class was generated from the following files:

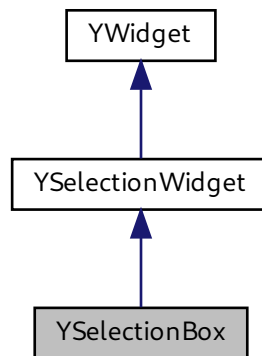
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YRpmGroupsTree.h
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YRpmGroupsTree.cc

## 5.122 YSelectionBox Class Reference

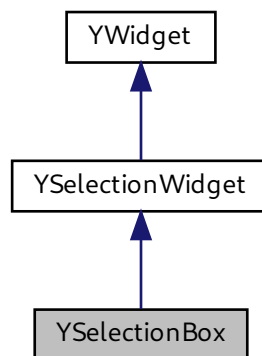
Selection box: List box that displays a (scrollable) list of items from which the user can select exactly one.

```
#include <YSelectionBox.h>
```

Inheritance diagram for YSelectionBox:



Collaboration diagram for YSelectionBox:



## Public Member Functions

- virtual [~YSelectionBox](#) ()  
*Destructor.*
- virtual const char \* [widgetClass](#) () const  
*Returns a descriptive name of this widget class for logging, debugging etc.*
- bool [shrinkable](#) () const  
*Return 'true' if this SelectionBox should be very small.*
- virtual void [setShrinkable](#) (bool [shrinkable](#)=true)  
*Make this SelectionBox very small.*
- bool [immediateMode](#) () const  
*Deliver even more events than with [notify\(\)](#) set.*
- void [setImmediateMode](#) (bool on=true)  
*Set [immediateMode\(\)](#) on or off.*
- virtual bool [setProperty](#) (const std::string &propertyName, const [YPropertyValue](#) &val)  
*Set a property.*
- virtual [YPropertyValue](#) [getProperty](#) (const std::string &propertyName)  
*Get a property.*
- virtual const [YPropertySet](#) & [propertySet](#) ()  
*Return this class's property set.*
- const char \* [userInputProperty](#) ()  
*The name of the widget property that will return user input.*

## Protected Member Functions

- [YSelectionBox](#) ([YWidget](#) \*parent, const std::string &label)  
*Constructor.*

### 5.122.1 Detailed Description

Selection box: List box that displays a (scrollable) list of items from which the user can select exactly one.

Each item has a label text and an optional icon (\*).

This widget displays a number of items at once (as screen space permits). If there is little screen space, you might consider using a [ComboBox](#) instead which (in non-editable mode which is the default) displays just one item (the selected item) right away and the others in a pop-up dialog upon mouse click or keypress.

The selection box also has a caption label that is displayed above the list. The hotkey displayed in that caption label will move the keyboard focus into the list.

If multiple columns are needed, use the [YTable](#) widget instead. For tree-like structures, use the [YTree](#) widget. Use [YMultiSelectionBox](#) if more than one item can be selected.

(\*) Not all UIs (in particular not text-based UIs) support displaying icons, so an icon should never be an exclusive means to display any kind of information.

Definition at line 56 of file [YSelectionBox.h](#).

## 5.122.2 Member Function Documentation

### 5.122.2.1 getProperty()

```
YPropertyValue YSelectionBox::getProperty (
    const std::string & propertyName ) [virtual]
```

Get a property.

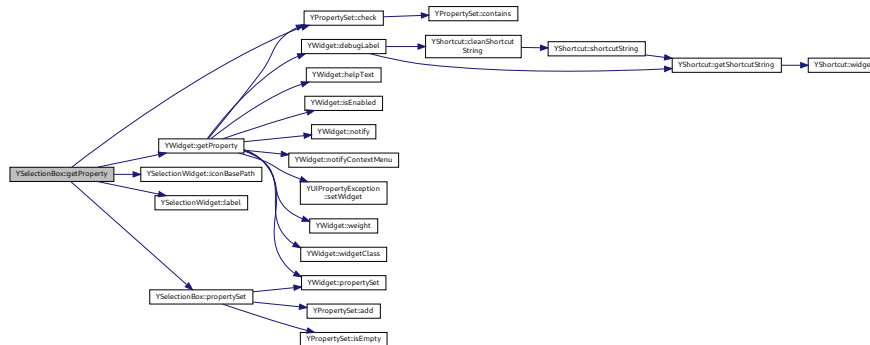
Reimplemented from [YWidget](#).

This method may throw [YUIPropertyExceptions](#).

Reimplemented from [YWidget](#).

Definition at line 141 of file [YSelectionBox.cc](#).

Here is the call graph for this function:



### 5.122.2.2 immediateMode()

```
bool YSelectionBox::immediateMode ( ) const
```

Deliver even more events than with [notify\(\)](#) set.

For [YSelectionBox](#), this is relevant mostly for the Ncurses UI:

In graphical UIs like the Qt UI, the user can use the mouse to select an item in a selection box. With [notify\(\)](#) set, this will send an event right away (i.e., it will make [UserInput](#) and related return, while normally it would only return when the user clicks a [PushButton](#)).

In the Ncurses UI, there is no mouse, so the user has to use the cursor keys to move to the item he wants to select. In [immediateMode\(\)](#), every cursor key press will make the selection box send an event. Without [immediateMode\(\)](#), the [NCSelectionBox](#) will wait until the user hits the [Return] key until an event is sent. Depending on what the application does upon each selection box event, [immediateMode\(\)](#) might make the application less responsive.

Definition at line 80 of file [YSelectionBox.cc](#).

### 5.122.2.3 `propertySet()`

```
const YPropertySet & YSelectionBox::propertySet ( ) [virtual]
```

Return this class's property set.

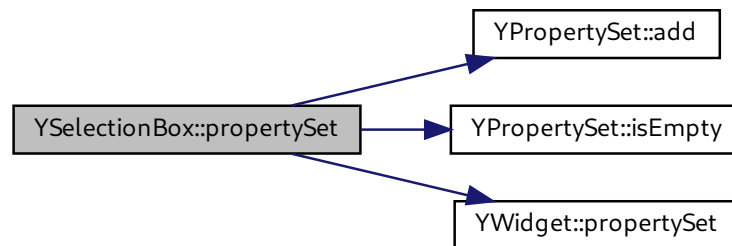
This also initializes the property upon the first call.

Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 96 of file [YSelectionBox.cc](#).

Here is the call graph for this function:



### 5.122.2.4 `setProperty()`

```
bool YSelectionBox::setProperty (
    const std::string & propertyName,
    const YPropertyValue & val ) [virtual]
```

Set a property.

Reimplemented from [YWidget](#).

This function may throw `YUIPropertyExceptions`.

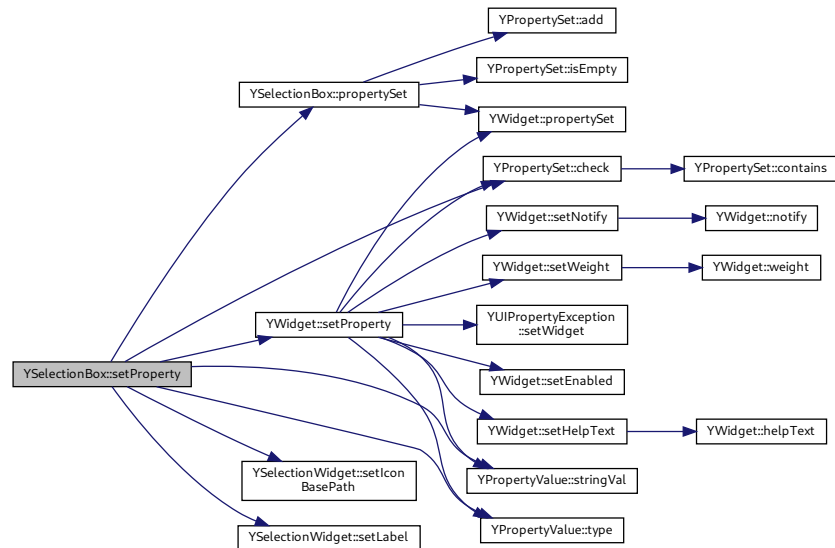
This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from [YWidget](#).



Definition at line 122 of file [YSelectionBox.cc](#).

Here is the call graph for this function:



#### 5.122.2.5 setShrinkable()

```
void YSelectionBox::setShrinkable (
    bool shrinkable = true ) [virtual]
```

Make this SelectionBox very small.

This will take effect only upon the next geometry management run.

Derived classes can overwrite this, but should call this base class function in the new function.

Definition at line 74 of file [YSelectionBox.cc](#).

Here is the call graph for this function:



### 5.122.2.6 userInputProperty()

```
const char* YSelectionBox::userInputProperty ( ) [inline], [virtual]
```

The name of the widget property that will return user input.

Inherited from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 149 of file [YSelectionBox.h](#).

The documentation for this class was generated from the following files:

- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YSelectionBox.h](#)
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YSelectionBox.cc](#)

## 5.123 YSelectionBoxPrivate Struct Reference

### Public Attributes

- bool **shrinkable**
- bool **immediateMode**

### 5.123.1 Detailed Description

Definition at line 36 of file [YSelectionBox.cc](#).

The documentation for this struct was generated from the following file:

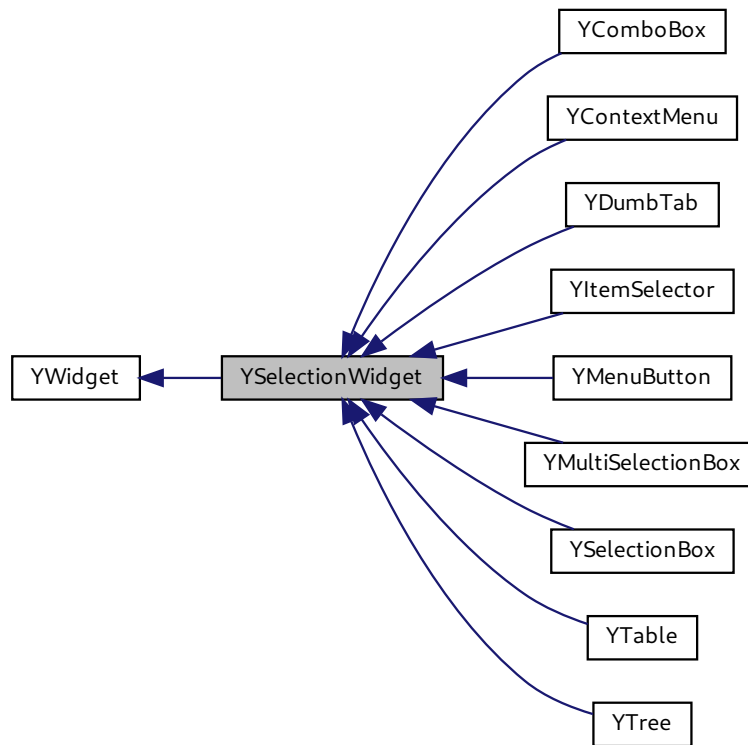
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YSelectionBox.cc](#)

## 5.124 YSelectionWidget Class Reference

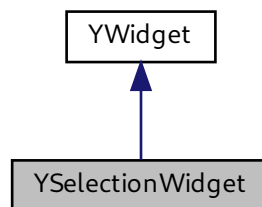
Base class for various kinds of multi-value widgets.

```
#include <YSelectionWidget.h>
```

Inheritance diagram for YSelectionWidget:



Collaboration diagram for YSelectionWidget:



## Public Member Functions

- virtual `~YSelectionWidget ()`  
*Destructor.*
- virtual const char \* `widgetClass () const`  
*Returns a descriptive name of this widget class for logging, debugging etc.*
- std::string `label () const`  
*Return this widget's label (the caption above the item list).*
- virtual void `setLabel (const std::string &newLabel)`  
*Change this widget's label (the caption above the item list).*
- virtual void `addItem (YItem *item_disown)`  
*Add one item.*
- void `addItem (const std::string &itemLabel, bool selected=false)`  
*Overloaded for convenience: Add an item by string.*
- void `addItem (const std::string &itemLabel, const std::string &iconName, bool selected=false)`  
*Overloaded for convenience: Add an item with a text and an icon.*
- virtual void `addItems (const YItemCollection &itemCollection)`  
*Add multiple items.*
- virtual void `deleteAllItems ()`  
*Delete all items.*
- void `setItems (const YItemCollection &itemCollection)`  
*Delete all items and add new items.*
- YItemIterator `itemsBegin ()`  
*Return an iterator that points to the first item.*
- YItemConstIterator `itemsBegin () const`
- YItemIterator `itemsEnd ()`  
*Return an iterator that points behind the last item.*
- YItemConstIterator `itemsEnd () const`
- bool `hasItems () const`  
*Return 'true' if this widget has any items.*
- int `itemsCount () const`  
*Return the number of items.*
- YItem \* `firstItem () const`  
*Return the first item or 0 if there is none.*
- virtual YItem \* `selectedItem ()`  
*Return the (first) selected item or 0 if none is selected.*
- virtual YItemCollection `selectedItems ()`  
*Return all selected items.*
- bool `hasSelectedItem ()`  
*Return 'true' if any item is selected.*
- virtual void `selectItem (YItem *item, bool selected=true)`  
*Select or deselect an item.*
- virtual void `setItemStatus (YItem *item, int status)`  
*Set the status of an item.*
- virtual void `deselectAllItems ()`  
*Deselect all items.*
- void `setIconBasePath (const std::string &basePath)`

- Set this widget's base path where to look up icons.*

  - `std::string iconBasePath () const`

*Return this widget's base path where to look up icons as set with `setIconBasePath()`.*
- `std::string iconFullPath (const std::string &iconName) const`

*Return the full path + file name for the specified icon name.*
- `std::string iconFullPath (YItem *item) const`

*Return the full path + file name for the icon of the specified item.*
- `bool itemsContain (YItem *item) const`

*Return 'true' if this widget's items contain the specified item.*
- `YItem * findItem (const std::string &itemLabel) const`

*Find the (first) item with the specified label.*
- `virtual std::string shortcutString () const`

*Get the string of this widget that holds the keyboard shortcut.*
- `virtual void setShortcutString (const std::string &str)`

*Set the string of this widget that holds the keyboard shortcut.*
- `void dumpItems () const`

*Dump all items and their selection state to the log.*
- `bool enforceSingleSelection () const`

*Return 'true' if this base class should enforce single selection.*

## Protected Member Functions

- `YSelectionWidget (YWidget *parent, const std::string &label, bool enforceSingleSelection, bool recursiveSelection=false)`

*Constructor.*
- `void setEnforceSingleSelection (bool on)`

*Set single selection mode on or off.*
- `void setEnforceInitialSelection (bool on)`

*In single selection mode, enforce selecting an initial item ('true' by default).*
- `bool enforceInitialSelection () const`

*Return 'true' if this class enforces an initial selection.*
- `bool recursiveSelection () const`

*Return 'true' if this base class should select children recursively.*
- `YItem * findSelectedItem (YItemConstIterator begin, YItemConstIterator end)`

*Recursively try to find the first selected item between iterators 'begin' and 'end'.*
- `void findSelectedItems (YItemCollection &selectedItems, YItemConstIterator begin, YItemConstIterator end)`

*Recursively find all selected items between iterators 'begin' and 'end' and add each of them to the 'selectedItems' YItemCollection.*
- `void deselectAllItems (YItemIterator begin, YItemIterator end)`

*Recursively deselect all items between iterators 'begin' and 'end'.*
- `YItem * findItem (const std::string &wantedItemLabel, YItemConstIterator begin, YItemConstIterator end) const`

*Recursively try to find an item with label 'wantedItemLabel' between iterators 'begin' and 'end'.*
- `bool itemsContain (YItem *wantedItem, YItemConstIterator begin, YItemConstIterator end) const`

*Recursively check if 'wantedItem' is between iterators 'begin' and 'end'.*
- `YItem * itemAt (int index) const`

*Return the item at index 'index' (from 0) or 0 if there is no such item.*

### 5.124.1 Detailed Description

Base class for various kinds of multi-value widgets.

- [YSelectionBox](#), [YMultiSelectionBox](#), [YComboBox](#)
- [YContextMenu](#), [YMenuButton](#)
- [YTable](#)
- [YTree](#)
- [YDumbTab](#)

Definition at line 42 of file [YSelectionWidget.h](#).

### 5.124.2 Constructor & Destructor Documentation

#### 5.124.2.1 YSelectionWidget()

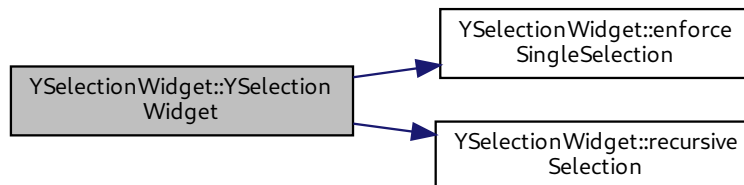
```
YSelectionWidget::YSelectionWidget (  
    YWidget * parent,  
    const std::string & label,  
    bool enforceSingleSelection,  
    bool recursiveSelection = false )    [protected]
```

Constructor.

'enforceSingleSelection' indicates if this base class should enforce single selection when items are added or when items are selected from the application. Note that single selection can also mean that no item is selected.

Definition at line 59 of file [YSelectionWidget.cc](#).

Here is the call graph for this function:



### 5.124.3 Member Function Documentation

#### 5.124.3.1 addItem() [1/2]

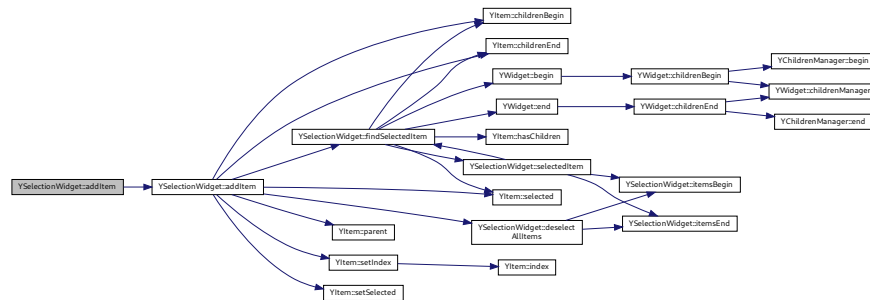
```
void YSelectionWidget::addItem (
    const std::string & itemLabel,
    const std::string & iconName,
    bool selected = false )
```

Overloaded for convenience: Add an item with a text and an icon.

Note that not all UIs can display icons.

Definition at line 248 of file [YSelectionWidget.cc](#).

Here is the call graph for this function:



#### 5.124.3.2 addItem() [2/2]

```
void YSelectionWidget::addItem (
    YItem * item_disown ) [virtual]
```

Add one item.

This widget assumes ownership of the item object and will delete it in its destructor.

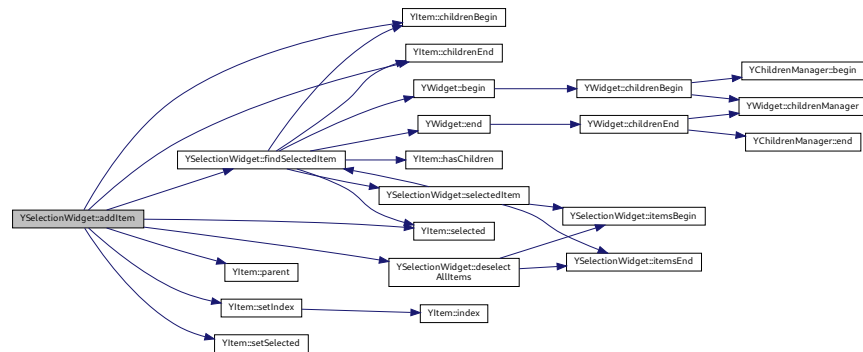
NOTE: For tree items, call this only for the toplevel items; all non-toplevel items are already owned by their respective parent items. Adding them to the parent widget will clash with this ownership.

Derived classes can overwrite this function, but they should call this base class function in the new implementation.

Reimplemented in [YMenuButton](#), [YContextMenu](#), and [YDumbTab](#).

Definition at line 186 of file [YSelectionWidget.cc](#).

Here is the call graph for this function:



### 5.124.3.3 addItem()

```
void YSelectionWidget::addItem (
    const YItemCollection & itemCollection ) [virtual]
```

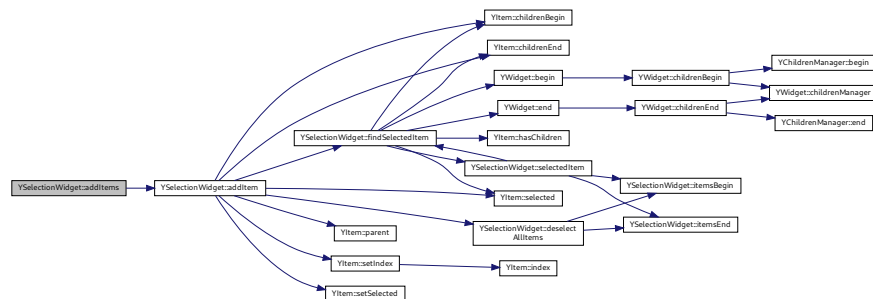
Add multiple items.

For some UIs, this can be more efficient than calling [addItem\(\)](#) multiple times.

Reimplemented in [YTree](#), [YMenuButton](#), and [YContextMenu](#).

Definition at line 264 of file [YSelectionWidget.cc](#).

Here is the call graph for this function:





#### 5.124.3.4 deleteAllItems()

```
void YSelectionWidget::deleteAllItems ( ) [virtual]
```

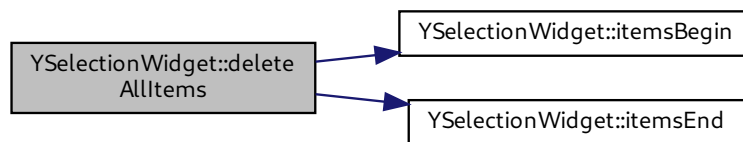
Delete all items.

Derived classes can overwrite this function, but they should call this base class function in the new implementation.

Reimplemented in [YMenuButton](#), and [YContextMenu](#).

Definition at line 80 of file [YSelectionWidget.cc](#).

Here is the call graph for this function:



#### 5.124.3.5 deselectAllItems()

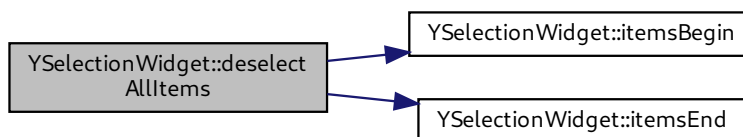
```
void YSelectionWidget::deselectAllItems ( ) [virtual]
```

Deselect all items.

Derived classes can overwrite this function, but they should call this base class function in the new implementation.

Definition at line 484 of file [YSelectionWidget.cc](#).

Here is the call graph for this function:



### 5.124.3.6 findItem() [1/2]

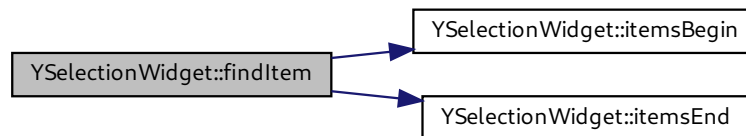
```
YItem * YSelectionWidget::findItem (
    const std::string & itemLabel ) const
```

Find the (first) item with the specified label.

Return 0 if there is no item with that label.

Definition at line 506 of file [YSelectionWidget.cc](#).

Here is the call graph for this function:



### 5.124.3.7 findItem() [2/2]

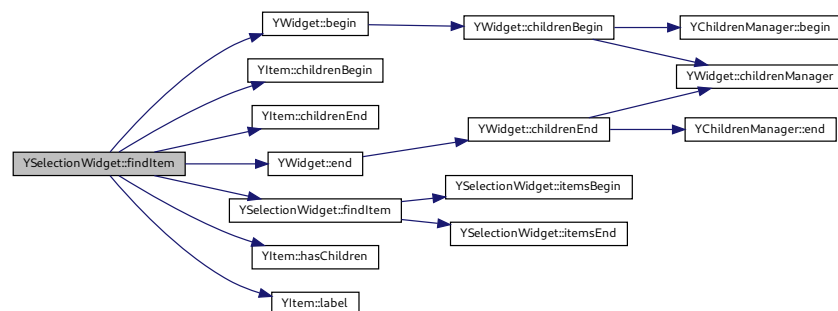
```
YItem * YSelectionWidget::findItem (
    const std::string & wantedItemLabel,
    YItemConstIterator begin,
    YItemConstIterator end ) const [protected]
```

Recursively try to find an item with label 'wantedItemLabel' between iterators 'begin' and 'end'.

Return that item or 0 if there is none.

Definition at line 513 of file [YSelectionWidget.cc](#).

Here is the call graph for this function:



### 5.124.3.8 findSelectedItem()

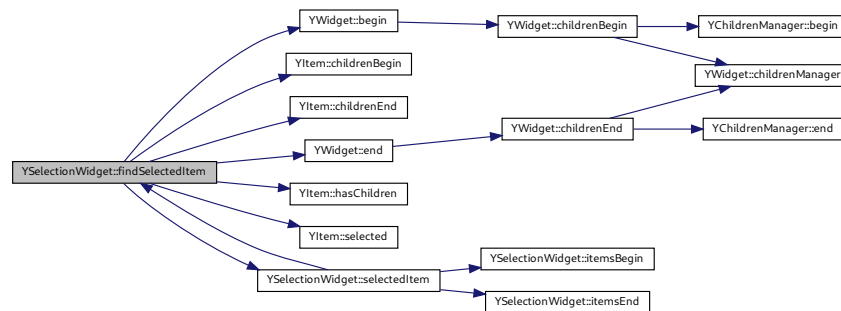
```
YItem * YSelectionWidget::findSelectedItem (
    YItemConstIterator begin,
    YItemConstIterator end ) [protected]
```

Recursively try to find the first selected item between iterators 'begin' and 'end'.

Return that item or 0 if there is none.

Definition at line 349 of file [YSelectionWidget.cc](#).

Here is the call graph for this function:



### 5.124.3.9 iconFullPath() [1/2]

```
string YSelectionWidget::iconFullPath (
    const std::string & iconName ) const
```

Return the full path + file name for the specified icon name.

If `iconBasePath` is non-empty, it is prepended to the icon name. Otherwise, `YUI::yApp()->iconLoader()` and its icon search paths is used find the icon in one of them

If 'iconName' is empty, this will return an empty string.

Definition at line 155 of file [YSelectionWidget.cc](#).

**5.124.3.10 iconFullPath() [2/2]**

```
string YSelectionWidget::iconFullPath (
    YItem * item ) const
```

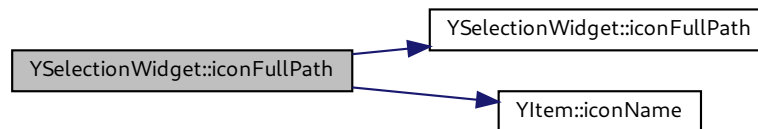
Return the full path + file name for the icon of the specified item.

If iconBasePath is non-empty, it is prepended to the item's iconName. Otherwise, YUI::yApp()->iconLoader() and its icon search paths is used find the icon in one of them

If 'item' does not have an iconName specified, this will return an empty string.

Definition at line 177 of file [YSelectionWidget.cc](#).

Here is the call graph for this function:

**5.124.3.11 itemsBegin()**

```
YItemIterator YSelectionWidget::itemsBegin ( )
```

Return an iterator that points to the first item.

For YSelectionWidgets that can have tree structures, this iterator will iterate over the toplevel items.

Important: Don't use this iterator to iterate over all items and check their "selected" state; that information might not always be up to date. Use the dedicated functions for that.

Definition at line 283 of file [YSelectionWidget.cc](#).

**5.124.3.12 itemCount()**

```
int YSelectionWidget::itemCount ( ) const
```

Return the number of items.

For YSelectionWidgets that can have tree structures, this returns the number of toplevel items.

Definition at line 315 of file [YSelectionWidget.cc](#).

**5.124.3.13 selectedItems()**

```
YItemCollection YSelectionWidget::selectedItems ( ) [virtual]
```

Return all selected items.

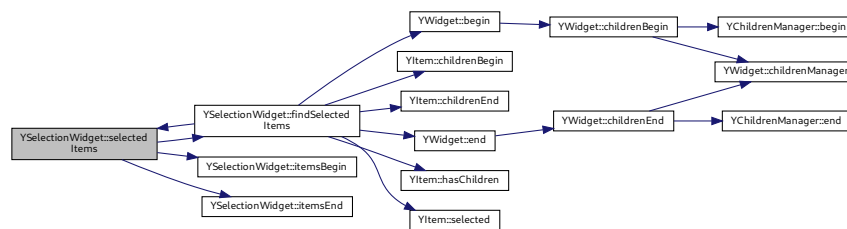
This is mostly useful for derived classes that allow selecting multiple items.

This function does not transfer ownership of those items to the caller, so don't try to delete them!

Reimplemented in [YComboBox](#).

Definition at line 377 of file [YSelectionWidget.cc](#).

Here is the call graph for this function:

**5.124.3.14 selectItem()**

```
void YSelectionWidget::selectItem (
    YItem * item,
    bool selected = true ) [virtual]
```

Select or deselect an item.

Notice that this is different from [YItem::setSelected\(\)](#) because unlike the latter function, this function informs the parent widget of the selection change.

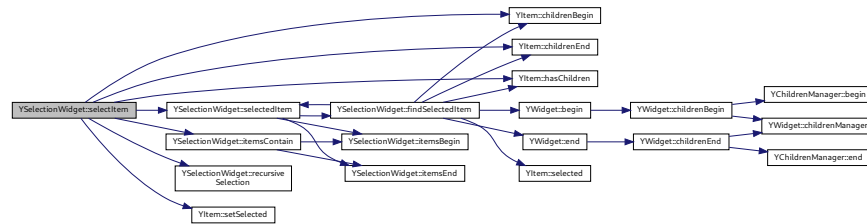
If only one item can be selected at any time (single selection), the derived class will make sure to deselect any previous selection, if applicable.

Derived classes should overwrite this function, but they should call this base class function at the new function's start (this will also check if the item really belongs to this widget and throw an exception if not).

Reimplemented in [YComboBox](#).

Definition at line 414 of file [YSelectionWidget.cc](#).

Here is the call graph for this function:



#### 5.124.3.15 setEnforceInitialSelection()

```
void YSelectionWidget::setEnforceInitialSelection (
    bool on ) [protected]
```

In single selection mode, enforce selecting an initial item ('true' by default).

This is ignored in multi selection mode.

Definition at line 117 of file [YSelectionWidget.cc](#).

#### 5.124.3.16 setEnforceSingleSelection()

```
void YSelectionWidget::setEnforceSingleSelection (
    bool on ) [protected]
```

Set single selection mode on or off.

In single selection mode, only one item can be selected at any time.

If set, this base class enforces this when items are added or when items are selected from the application. Note that single selection can also mean that no item is selected.

Definition at line 137 of file [YSelectionWidget.cc](#).

Here is the call graph for this function:



**5.124.3.17 setIconBasePath()**

```
void YSelectionWidget::setIconBasePath (
    const std::string & basePath )
```

Set this widget's base path where to look up icons.

If this is a relative path, `YUI::qApp()->iconBasePath()` is prepended.

Definition at line 143 of file [YSelectionWidget.cc](#).

**5.124.3.18 setItemStatus()**

```
void YSelectionWidget::setItemStatus (
    YItem * item,
    int status ) [virtual]
```

Set the status of an item.

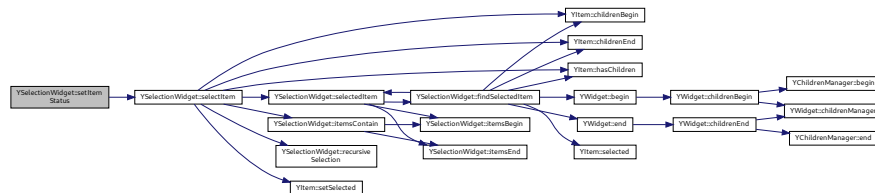
This is similar to [selectItem\(\)](#), but with numeric values.

This default implementation just calls [selectItem\(\)](#) with 'status' converted to boolean. Derived classes can choose to make more detailed use of the numeric value.

Reimplemented in [YItemSelector](#).

Definition at line 444 of file [YSelectionWidget.cc](#).

Here is the call graph for this function:

**5.124.3.19 setLabel()**

```
void YSelectionWidget::setLabel (
    const std::string & newLabel ) [virtual]
```

Change this widget's label (the caption above the item list).

Derived classes should overwrite this function, but they should call this base class function in the new implementation.

Definition at line 105 of file [YSelectionWidget.cc](#).

#### 5.124.3.20 `setShortcutString()`

```
virtual void YSelectionWidget::setShortcutString (
    const std::string & str ) [inline], [virtual]
```

Set the string of this widget that holds the keyboard shortcut.

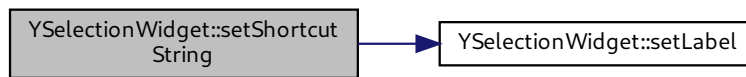
Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Reimplemented in [YDumbTab](#).

Definition at line 279 of file [YSelectionWidget.h](#).

Here is the call graph for this function:



#### 5.124.3.21 `shortcutString()`

```
virtual std::string YSelectionWidget::shortcutString ( ) const [inline], [virtual]
```

Get the string of this widget that holds the keyboard shortcut.

Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Reimplemented in [YDumbTab](#).

Definition at line 272 of file [YSelectionWidget.h](#).

Here is the call graph for this function:



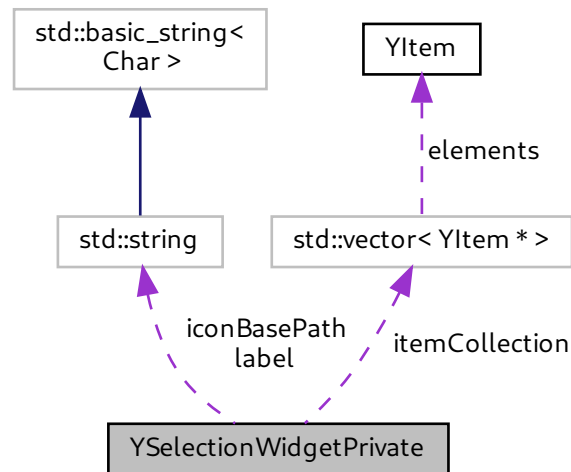
The documentation for this class was generated from the following files:

- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YSelectionWidget.h`
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YSelectionWidget.cc`



## 5.125 YSelectionWidgetPrivate Struct Reference

Collaboration diagram for YSelectionWidgetPrivate:



### Public Member Functions

- **YSelectionWidgetPrivate** (const string &label, bool enforceSingleSelection, bool recursiveSelection)

### Public Attributes

- string **label**
- bool **enforceSingleSelection**
- bool **enforceInitialSelection**
- bool **recursiveSelection**
- string **iconBasePath**
- [YItemCollection](#) **itemCollection**

### 5.125.1 Detailed Description

Definition at line 37 of file [YSelectionWidget.cc](#).

The documentation for this struct was generated from the following file:

- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YSelectionWidget.cc

## 5.126 YSettings Class Reference

Settings for libyui.

```
#include <YSettings.h>
```

### Static Public Member Functions

- static void [setProgDir](#) (std::string directory)  
*This can be used to set a subdir beneath PLUGINDIR or THEMEDIR, where your program stores a custom plugin or theme.*
- static std::string [progDir](#) ()  
*Returns the value of your program's subdir.*
- static void [setIconDir](#) (std::string directory)  
*This can be used to set a subdir ICONDIR, where your program stores a custom icons.*
- static std::string [iconDir](#) ()  
*Returns the value of your program's icons subdir.*
- static void [setThemeDir](#) (std::string directory)  
*This can be used to set a subdir THEMEDIR, where your program stores a custom icons.*
- static std::string [themeDir](#) ()  
*Returns the value of your program's theme subdir.*
- static void [setLocaleDir](#) (std::string directory)  
*This can be used to set a subdir LOCALEDIR, where your program stores translations.*
- static std::string [localeDir](#) ()  
*Returns the value of your program's locale subdir.*
- static void [loadedUI](#) (std::string ui)  
*This can be used to set the loaded UI-backend.*
- static std::string [loadedUI](#) ()  
*Returns the value of the loaded UI-backend.*

### Static Protected Member Functions

- static void [loadedUI](#) (std::string ui, bool force)  
*This can be used to set the loaded UI-backend.*

#### 5.126.1 Detailed Description

Settings for libyui.

This singleton-object hold some presets for libyui.

Definition at line 49 of file [YSettings.h](#).

## 5.126.2 Member Function Documentation

### 5.126.2.1 loadedUI() [1/2]

```
void YSettings::loadedUI (
    std::string ui ) [static]
```

This can be used to set the loaded UI-backend.

Once this is set, it can't be altered. If you do so although an exception will be thrown.

Definition at line 196 of file [YSettings.cc](#).

Here is the call graph for this function:



### 5.126.2.2 loadedUI() [2/2]

```
void YSettings::loadedUI (
    std::string ui,
    bool force ) [static], [protected]
```

This can be used to set the loaded UI-backend.

Once this is set, it can't be altered, except if you force it. If you do so without force an exception will be thrown.

Definition at line 179 of file [YSettings.cc](#).

### 5.126.2.3 setIconDir()

```
void YSettings::setIconDir (
    std::string directory ) [static]
```

This can be used to set a subdir ICONDIR, where your program stores a custom icons.

Once this is set, it can't be altered. If you do so although an exception will be thrown.

Definition at line 82 of file [YSettings.cc](#).

### 5.126.2.4 setLocaleDir()

```
void YSettings::setLocaleDir (
    std::string directory ) [static]
```

This can be used to set a subdir LOCALEDIR, where your program stores translations.

Once this is set, it can't be altered. If you do so although an exception will be thrown.

Definition at line 147 of file [YSettings.cc](#).

### 5.126.2.5 setProgDir()

```
void YSettings::setProgDir (
    std::string directory ) [static]
```

This can be used to set a subdir beneath PLUGINDIR or THEMEDIR, where your program stores a custom plugin or theme.

Once this is set, it can't be altered. If you do so although an exception will be thrown.

Definition at line 58 of file [YSettings.cc](#).

### 5.126.2.6 setThemeDir()

```
void YSettings::setThemeDir (
    std::string directory ) [static]
```

This can be used to set a subdir THEMEDIR, where your program stores a custom icons.

Once this is set, it can't be altered. If you do so although an exception will be thrown.

Definition at line 113 of file [YSettings.cc](#).

The documentation for this class was generated from the following files:

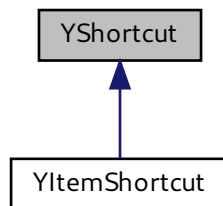
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YSettings.h](#)
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YSettings.cc](#)

## 5.127 YShortcut Class Reference

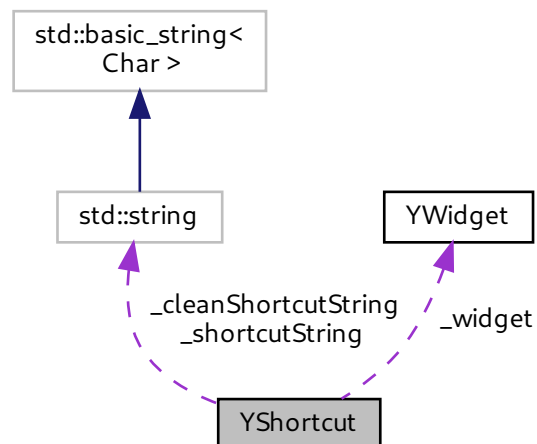
Helper class for shortcut management: This class holds data about the shortcut for one single widget.

```
#include <YShortcut.h>
```

Inheritance diagram for YShortcut:



Collaboration diagram for YShortcut:



### Public Types

- enum { **None** = 0 }  
*Marker for "no shortcut".*

## Public Member Functions

- [YShortcut](#) ([YWidget](#) \*shortcut\_widget)  
*Constructor.*
- virtual [~YShortcut](#) ()  
*Destructor.*
- [YWidget](#) \* [widget](#) () const  
*Returns the [YWidget](#) this shortcut data belong to.*
- const char \* [widgetClass](#) () const  
*Returns the textual representation of the widget class of the widget this shortcut data belongs to.*
- bool [isButton](#) () const  
*Returns 'true' if the widget that is associated with this shortcut is a button (derived from [YPushButton](#)).*
- bool [isWizardButton](#) () const  
*Returns 'true' if the widget that is associated with this shortcut is a wizard button (one of the navigation buttons of a wizard).*
- std::string [shortcutString](#) ()  
*Returns the complete shortcut string (which may or may not contain "&"), i.e.*
- std::string [cleanShortcutString](#) ()  
*Returns the shortcut string ( from the widget's shortcut property ) without any "&" markers.*
- char [preferred](#) ()  
*The preferred shortcut character, i.e.*
- char [shortcut](#) ()  
*The actual shortcut character.*
- virtual void [setShortcut](#) (char newShortcut)  
*Set (override) the shortcut character.*
- void [clearShortcut](#) ()  
*Clear the shortcut: Override the shortcut character with nothing.*
- bool [conflict](#) ()  
*Query the internal 'conflict' marker.*
- void [setConflict](#) (bool newConflictState=true)  
*Set or unset the internal 'conflict' marker.*
- int [distinctShortcutChars](#) ()  
*Obtain the number of distinct valid shortcut characters in the shortcut string, i.e.*
- bool [hasValidShortcutChar](#) ()  
*Return true if this shortcut contains any character that would be valid as a shortcut character.*

## Static Public Member Functions

- static std::string [cleanShortcutString](#) (std::string [shortcutString](#))  
*Static version of the above for general use: Returns the specified string without any "&" markers.*
- static char [shortcutMarker](#) ()  
*Static function: Returns the character used for marking keyboard shortcuts.*
- static std::string::size\_type [findShortcutPos](#) (const std::string &str, std::string::size\_type start\_pos=0)  
*Static function: Find the next occurrence of the shortcut marker ('&') in a string, beginning at starting position start\_pos.*
- static char [findShortcut](#) (const std::string &str, std::string::size\_type start\_pos=0)  
*Static function: Find the next shortcut marker in a string, beginning at starting position start\_pos.*
- static bool [isValid](#) (char c)  
*Returns 'true' if 'c' is a valid shortcut character, i.e.*
- static char [normalized](#) (char c)  
*Return the normalized version of shortcut character 'c', i.e.*
- static std::string [getShortcutString](#) (const [YWidget](#) \*widget)  
*Obtain a widget's shortcut property - the string that contains "&" to designate a shortcut.*

## Protected Member Functions

- virtual std::string [getShortcutString](#) ()  
Obtain the the shortcut property of this shortcut's widget - the string that contains "&" to designate a shortcut.

## Protected Attributes

- [YWidget](#) \* **\_widget**
- std::string **\_shortcutString**
- bool **\_shortcutStringCached**
- std::string **\_cleanShortcutString**
- bool **\_cleanShortcutStringCached**
- int **\_preferred**
- int **\_shortcut**
- bool **\_conflict**
- bool **\_isButton**
- bool **\_isWizardButton**
- int **\_distinctShortcutChars**

### 5.127.1 Detailed Description

Helper class for shortcut management: This class holds data about the shortcut for one single widget.

Definition at line 40 of file [YShortcut.h](#).

### 5.127.2 Member Function Documentation

#### 5.127.2.1 clearShortcut()

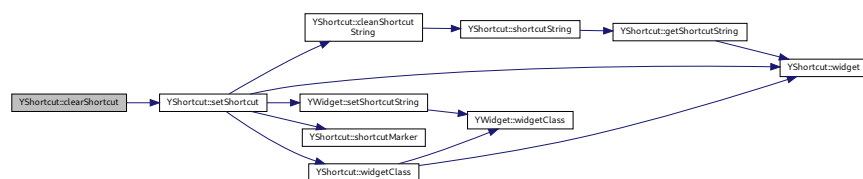
```
void YShortcut::clearShortcut ( )
```

Clear the shortcut: Override the shortcut character with nothing.

This may happen if a conflict cannot be resolved.

Definition at line 175 of file [YShortcut.cc](#).

Here is the call graph for this function:



### 5.127.2.2 conflict()

```
bool YShortcut::conflict ( ) [inline]
```

Query the internal 'conflict' marker.

This class doesn't care about that flag, it just stores it for the convenience of higher-level classes.

Definition at line 131 of file [YShortcut.h](#).

### 5.127.2.3 distinctShortcutChars()

```
int YShortcut::distinctShortcutChars ( )
```

Obtain the number of distinct valid shortcut characters in the shortcut string, i.e.

how many different shortcuts that widget could get.

Definition at line 182 of file [YShortcut.cc](#).

Here is the call graph for this function:



### 5.127.2.4 findShortcut()

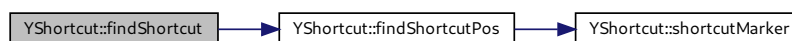
```
char YShortcut::findShortcut (
    const std::string & str,
    std::string::size_type start_pos = 0 ) [static]
```

Static function: Find the next shortcut marker in a string, beginning at starting position `start_pos`.

Returns the shortcut character or 0 if none found.

Definition at line 282 of file [YShortcut.cc](#).

Here is the call graph for this function:





### 5.127.2.5 findShortcutPos()

```
string::size_type YShortcut::findShortcutPos (
    const std::string & str,
    std::string::size_type start_pos = 0 ) [static]
```

Static function: Find the next occurrence of the shortcut marker ('&') in a string, beginning at starting position start\_pos.

Returns string::npos if not found or the position of the shortcut marker (not the shortcut character!) if found.

Definition at line 256 of file [YShortcut.cc](#).

Here is the call graph for this function:



### 5.127.2.6 isValid()

```
bool YShortcut::isValid (
    char c ) [static]
```

Returns 'true' if 'c' is a valid shortcut character, i.e.

[a-zA-Z0-9], 'false' otherwise.

Definition at line 291 of file [YShortcut.cc](#).

### 5.127.2.7 normalized()

```
char YShortcut::normalized (
    char c ) [static]
```

Return the normalized version of shortcut character 'c', i.e.

a lowercase letter or a digit [a-z0-9]. Returns 0 if 'c' is invalid.

Definition at line 301 of file [YShortcut.cc](#).

### 5.127.2.8 preferred()

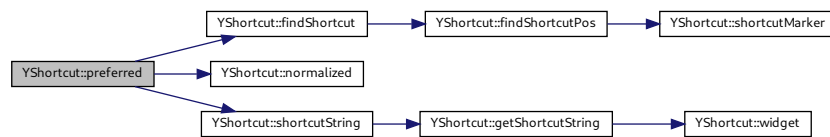
```
char YShortcut::preferred ( )
```

The preferred shortcut character, i.e.

the character that had been preceded by "&" before checking / resolving conflicts began.

Definition at line 119 of file [YShortcut.cc](#).

Here is the call graph for this function:



### 5.127.2.9 shortcut()

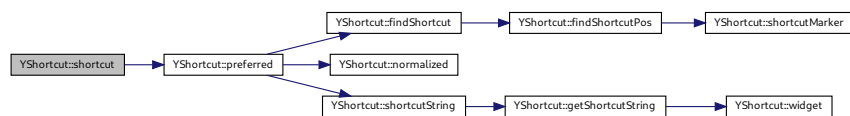
```
char YShortcut::shortcut ( )
```

The actual shortcut character.

This may be different from [preferred\(\)](#) if it is overridden.

Definition at line 131 of file [YShortcut.cc](#).

Here is the call graph for this function:



### 5.127.2.10 shortcutString()

```
string YShortcut::shortcutString ( )
```

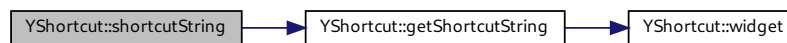
Returns the complete shortcut string (which may or may not contain "&"), i.e.

the value of the widget's shortcut property. For PushButtons, this is the label on the button ( e.g., "&Details..." ), for other widgets usually the caption above it.

This value is cached, i.e. this isn't a too expensive operation.

Definition at line 77 of file [YShortcut.cc](#).

Here is the call graph for this function:



The documentation for this class was generated from the following files:

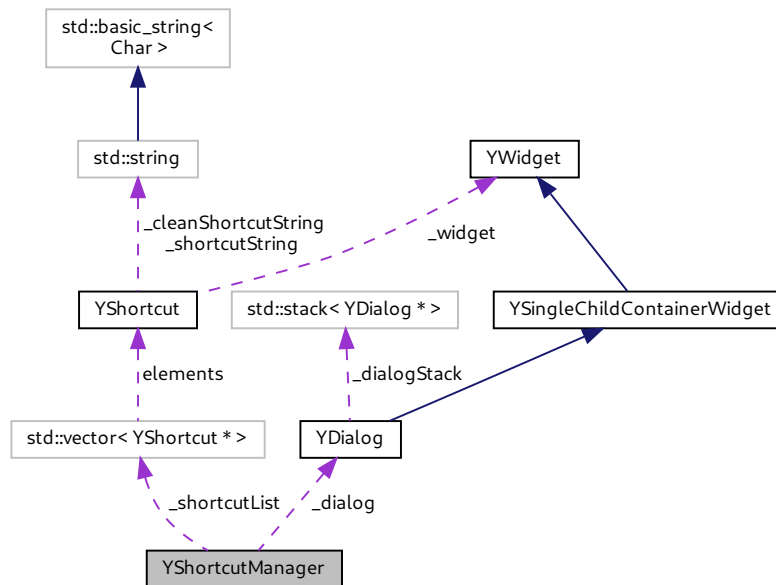
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YShortcut.h`
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YShortcut.cc`

## 5.128 YShortcutManager Class Reference

Helper class to manage keyboard shortcuts within one dialog and resolve keyboard shortcut conflicts.

```
#include <YShortcutManager.h>
```

Collaboration diagram for YShortcutManager:



## Public Member Functions

- [YShortcutManager](#) ([YDialog](#) \*[dialog](#))  
*Constructor.*
- virtual [~YShortcutManager](#) ()  
*Destructor.*
- void [checkShortcuts](#) (bool [autoResolve](#)=true)  
*Check the keyboard shortcuts of all children of this dialog (not for sub-dialogs!).*
- int [conflictCount](#) ()  
*Returns the number of shortcut conflicts.*
- void [resolveAllConflicts](#) ()  
*Resolve shortcut conflicts.*
- [YDialog](#) \* [dialog](#) ()  
*Returns the dialog this shortcut manager works on.*

## Protected Member Functions

- void [clearShortcutList](#) ()  
*Delete all members of the internal shortcut list, then empty the list.*
- void [findShortcutWidgets](#) (YWidgetListConstIterator [begin](#), YWidgetListConstIterator [end](#))  
*Recursively search all widgets between iterators 'begin' and 'end' (not those of any sub-dialogs!) for child widgets that could accept a keyboard shortcut and add these to `_shortcutList`.*

- void [resolveConflict](#) (YShortcut \*shortcut)  
*Pick a new shortcut character for 'shortcut' - one that isn't marked as used in the '\_used' array.*
- int [findShortestWizardButton](#) (const YShortcutList &conflictList)  
*Find the shortest wizard button in 'conflictList', if there is any.*
- unsigned [findShortestWidget](#) (const YShortcutList &conflictList)  
*Find the shortest widget in 'conflictList'.*

## Protected Attributes

- YDialog \* [\\_dialog](#)  
*The dialog this shortcut manager works on.*
- YShortcutList [\\_shortcutList](#)  
*List of all the shortcuts in this dialog.*
- int [\\_wanted](#) [sizeof(char)<< 8]  
*Counters for wanted shortcut characters.*
- bool [\\_used](#) [sizeof(char)<< 8]  
*Flags for used shortcut characters.*
- int [\\_conflictCount](#)  
*Counter for shortcut conflicts.*

### 5.128.1 Detailed Description

Helper class to manage keyboard shortcuts within one dialog and resolve keyboard shortcut conflicts.

Definition at line 38 of file [YShortcutManager.h](#).

### 5.128.2 Member Function Documentation

#### 5.128.2.1 checkShortcuts()

```
void YShortcutManager::checkShortcuts (
    bool autoResolve = true )
```

Check the keyboard shortcuts of all children of this dialog (not for sub-dialogs!).

Call [resolveAllConflicts\(\)](#) if 'autoResolve' is 'true'.

Definition at line 62 of file [YShortcutManager.cc](#).

```
int YShortcutManager::conflictCount ( ) [inline]
```

Valid only after `checkShortcuts()` or `resolveAllConflicts()`.

### 5.128.2.3 findShortestWidget()

Find the shortest widget in 'conflictList'.

Returns the index of the shortest widget.

Generated by Doxygen

#### 5.128.2.4 findShortestWizardButton()

```
int YShortcutManager::findShortestWizardButton (
    const YShortcutList & conflictList ) [protected]
```

Find the shortest wizard button in 'conflictList', if there is any.

Returns the index of that shortest wizard button or -1 if there is none.

Definition at line 309 of file [YShortcutManager.cc](#).

#### 5.128.2.5 resolveAllConflicts()

```
void YShortcutManager::resolveAllConflicts ( )
```

Resolve shortcut conflicts.

Requires [checkShortcuts\(\)](#) to be called first.

Note: This may or may not work. There is no general solution to that problem. This method tries its best, but you may end up with widgets that don't have any ( more ) shortcut.

Why? Just picture the following ( admittedly pathologic ) situation:

[& OK] [& OK] [& OK]

This will result in something like this:

[& OK] [O& K] [OK]

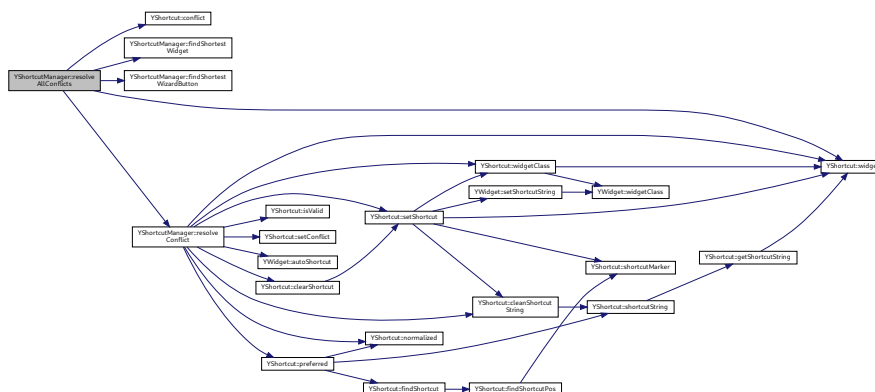
I.e. the first OK button will retain its preferred shortcut ( 'O' ), the second OK button's shortcut will be reassigned to 'K' and the third won't get any - there are simply not enough eligible shortcut characters.

This may even fail in much less pathological situations. This example is only supposed to give you a general idea why not to blindly rely on automatic shortcut resolving.

It's always best to resolve conflicts manually. This will generally result in much better shortcuts: Easier to memorize, less chance of picking characters that cannot really do a good job showing their shortcut like very narrow characters ( e.g., 'i' ) or descender characters ( e.g., 'g', 'p', 'q' - imagine those underlined! ).

Definition at line 163 of file [YShortcutManager.cc](#).

Here is the call graph for this function:



### 5.128.2.6 resolveConflict()

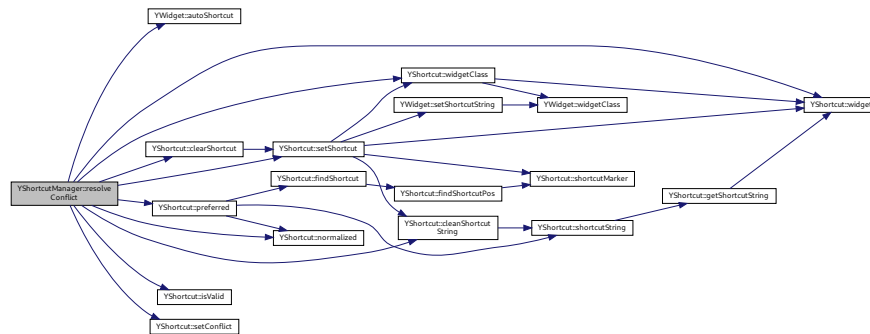
```
void YShortcutManager::resolveConflict (
    YShortcut * shortcut ) [protected]
```

Pick a new shortcut character for 'shortcut' - one that isn't marked as used in the '\_used' array.

Unset the conflict marker if that succeeded.

Definition at line 231 of file [YShortcutManager.cc](#).

Here is the call graph for this function:



The documentation for this class was generated from the following files:

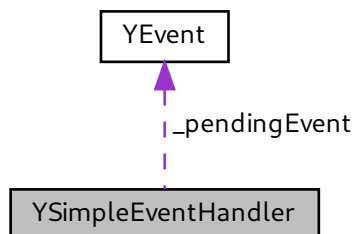
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YShortcutManager.h
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YShortcutManager.cc

## 5.129 YSimpleEventHandler Class Reference

Simple event handler suitable for most UIs.

```
#include <YSimpleEventHandler.h>
```

Collaboration diagram for YSimpleEventHandler:





## Public Member Functions

- [YSimpleEventHandler](#) ()  
*Constructor.*
- virtual [~YSimpleEventHandler](#) ()  
*Destructor.*
- void [sendEvent](#) ([YEvent](#) \*event\_disown)  
*Widget event handlers call this when an event occurred that should be the answer to a [UserInput\(\)](#) / [PollInput\(\)](#) (etc.) call.*
- bool [eventPendingFor](#) ([YWidget](#) \*widget) const  
*Returns 'true' if there is any event pending for the specified widget.*
- [YEvent](#) \* [pendingEvent](#) () const  
*Returns the last event that isn't processed yet or 0 if there is none.*
- [YEvent](#) \* [consumePendingEvent](#) ()  
*Consumes the pending event.*
- void [deletePendingEventsFor](#) ([YWidget](#) \*widget)  
*Delete any pending events for the specified widget.*
- void [clear](#) ()  
*Clears any pending event (deletes the corresponding object).*
- void [blockEvents](#) (bool block=true)  
*Block (or unblock) events.*
- void [unblockEvents](#) ()  
*Unblock events previously blocked.*
- bool [eventsBlocked](#) () const  
*Returns 'true' if events are currently blocked.*
- void [deleteEvent](#) ([YEvent](#) \*event)  
*Delete an event.*

## Protected Attributes

- [YEvent](#) \* [\\_pendingEvent](#)
- bool [\\_eventsBlocked](#)

### 5.129.1 Detailed Description

Simple event handler suitable for most UIs.

This event handler keeps track of one single event that gets overwritten when a new one arrives.

Definition at line 39 of file [YSimpleEventHandler.h](#).

### 5.129.2 Constructor & Destructor Documentation

### 5.129.2.1 ~YSimpleEventHandler()

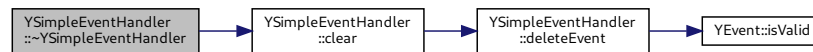
```
YSimpleEventHandler::~YSimpleEventHandler ( ) [virtual]
```

Destructor.

If there is a pending event, it is deleted here.

Definition at line 44 of file [YSimpleEventHandler.cc](#).

Here is the call graph for this function:



## 5.129.3 Member Function Documentation

### 5.129.3.1 blockEvents()

```
void YSimpleEventHandler::blockEvents (
    bool block = true )
```

Block (or unblock) events.

If events are blocked, any event sent with [sendEvent\(\)](#) from now on is ignored (and will get lost) until events are unblocked again.

Definition at line 145 of file [YSimpleEventHandler.cc](#).

### 5.129.3.2 consumePendingEvent()

```
YEvent * YSimpleEventHandler::consumePendingEvent ( )
```

Consumes the pending event.

Sets the internal pending event to 0. Does NOT delete the internal consuming event.

The caller assumes ownership of the object this pending event points to. In particular, he has to take care to delete that object when he is done processing it.

Returns the pending event or 0 if there is none.

Definition at line 62 of file [YSimpleEventHandler.cc](#).

### 5.129.3.3 deleteEvent()

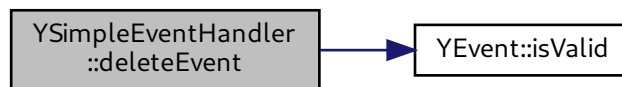
```
void YSimpleEventHandler::deleteEvent (
    YEvent * event )
```

Delete an event.

Don't call this from the outside; this is public only because of limitations of C++ .

Definition at line 156 of file [YSimpleEventHandler.cc](#).

Here is the call graph for this function:



### 5.129.3.4 deletePendingEventsFor()

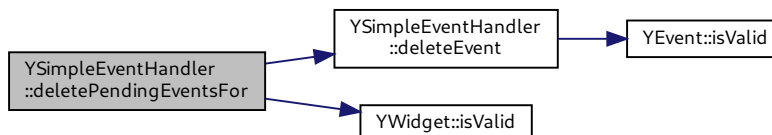
```
void YSimpleEventHandler::deletePendingEventsFor (
    YWidget * widget )
```

Delete any pending events for the specified widget.

This is useful mostly if the widget is about to be destroyed.

Definition at line 130 of file [YSimpleEventHandler.cc](#).

Here is the call graph for this function:



### 5.129.3.5 pendingEvent()

```
YEvent* YSimpleEventHandler::pendingEvent ( ) const [inline]
```

Returns the last event that isn't processed yet or 0 if there is none.

This event handler keeps track of only one single (the last one) event.

Definition at line 80 of file [YSimpleEventHandler.h](#).

### 5.129.3.6 sendEvent()

```
void YSimpleEventHandler::sendEvent (
    YEvent * event_disown )
```

Widget event handlers call this when an event occurred that should be the answer to a `UserInput()` / `PollInput()` (etc.) call.

The UI assumes ownership of the event object that 'event' points to, so the event MUST be created with `new()`. The UI is to take care to delete the event after it has been processed.

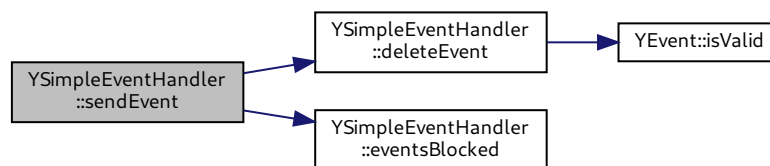
If events are blocked (see [blockEvents\(\)](#) ), the event sent with this function will be ignored (but safely deleted - no memory leak).

It is an error to pass 0 for 'event'. This simple event handler keeps track of only the latest user event. If there is more than one, older events are automatically discarded. Since Events are created on the heap with the "new" operator, discarded events need to be deleted.

Events that are not discarded are deleted later (after they are processed) by the generic UI.

Definition at line 75 of file [YSimpleEventHandler.cc](#).

Here is the call graph for this function:



### 5.129.3.7 unblockEvents()

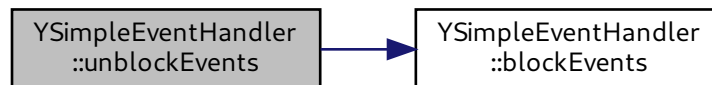
```
void YSimpleEventHandler::unblockEvents ( ) [inline]
```

Unblock events previously blocked.

This is just an alias for `blockEvents( false)` for better readability.

Definition at line 116 of file [YSimpleEventHandler.h](#).

Here is the call graph for this function:



The documentation for this class was generated from the following files:

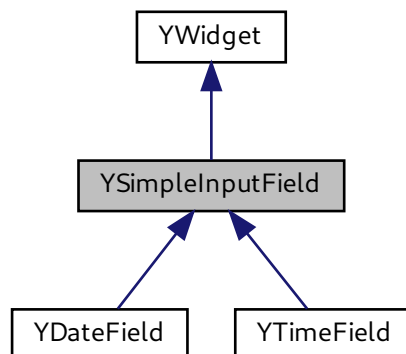
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YSimpleEventHandler.h`
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YSimpleEventHandler.cc`

## 5.130 YSimpleInputField Class Reference

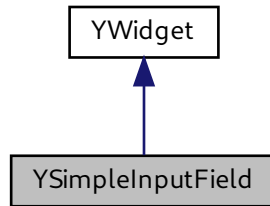
Abstract base class for simple input fields with a label above the field and a text value.

```
#include <YSimpleInputField.h>
```

Inheritance diagram for YSimpleInputField:



Collaboration diagram for YSimpleInputField:



## Public Member Functions

- virtual [~YSimpleInputField](#) ()  
*Destructor.*
- virtual std::string [value](#) ()=0  
*Get the current value (the text entered by the user or set from the outside) of this input field.*
- virtual void [setValue](#) (const std::string &text)=0  
*Set the current value (the text entered by the user or set from the outside) of this input field.*
- std::string [label](#) () const  
*Get the label (the caption above the input field).*
- virtual void [setLabel](#) (const std::string &label)  
*Set the label (the caption above the input field).*
- virtual bool [setProperty](#) (const std::string &propertyName, const [YPropertyValue](#) &val)  
*Set a property.*
- virtual [YPropertyValue](#) [getProperty](#) (const std::string &propertyName)  
*Get a property.*
- virtual const [YPropertySet](#) & [propertySet](#) ()  
*Return this class's property set.*
- virtual std::string [shortcutString](#) () const  
*Get the string of this widget that holds the keyboard shortcut.*
- virtual void [setShortcutString](#) (const std::string &str)  
*Set the string of this widget that holds the keyboard shortcut.*
- const char \* [userInputProperty](#) ()  
*The name of the widget property that will return user input.*

## Protected Member Functions

- [YSimpleInputField](#) ([YWidget](#) \*parent, const std::string &label)  
*Constructor.*

### 5.130.1 Detailed Description

Abstract base class for simple input fields with a label above the field and a text value.

Definition at line 37 of file [YSimpleInputField.h](#).

### 5.130.2 Member Function Documentation

#### 5.130.2.1 getProperty()

```
YPropertyValue YSimpleInputField::getProperty (
    const std::string & propertyName ) [virtual]
```

Get a property.

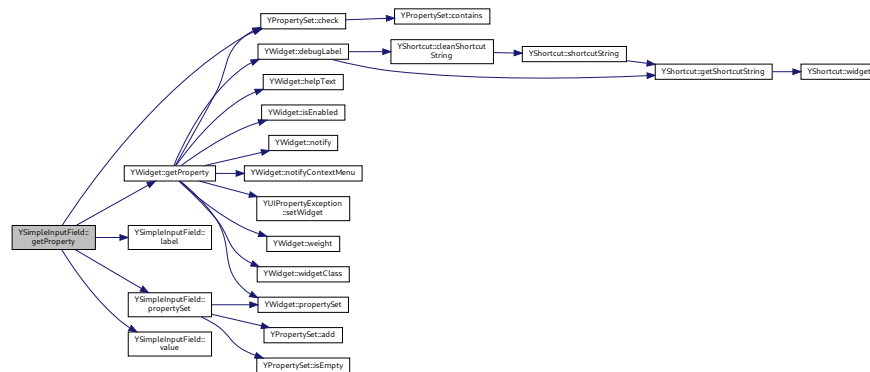
Reimplemented from [YWidget](#).

This method may throw YUIPropertyExceptions.

Reimplemented from [YWidget](#).

Definition at line 114 of file [YSimpleInputField.cc](#).

Here is the call graph for this function:



### 5.130.2.2 `propertySet()`

```
const YPropertySet & YSimpleInputField::propertySet ( ) [virtual]
```

Return this class's property set.

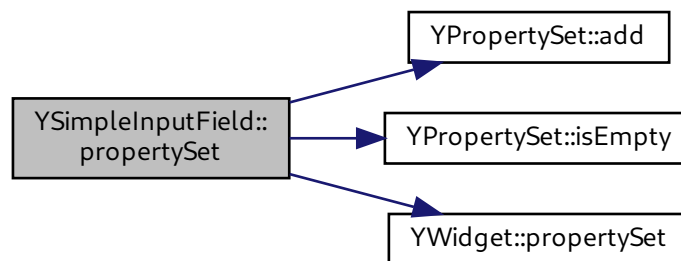
This also initializes the property upon the first call.

Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 78 of file [YSimpleInputField.cc](#).

Here is the call graph for this function:



### 5.130.2.3 `setLabel()`

```
void YSimpleInputField::setLabel (
    const std::string & label ) [virtual]
```

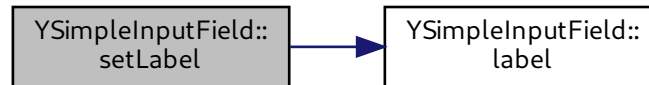
Set the label (the caption above the input field).

Derived classes are free to reimplement this, but they should call this base class method at the end of the overloaded function.

Definition at line 70 of file [YSimpleInputField.cc](#).



Here is the call graph for this function:



#### 5.130.2.4 setProperty()

```
bool YSimpleInputField::setProperty (
    const std::string & propertyName,
    const YPropertyValue & val ) [virtual]
```

Set a property.

Reimplemented from [YWidget](#).

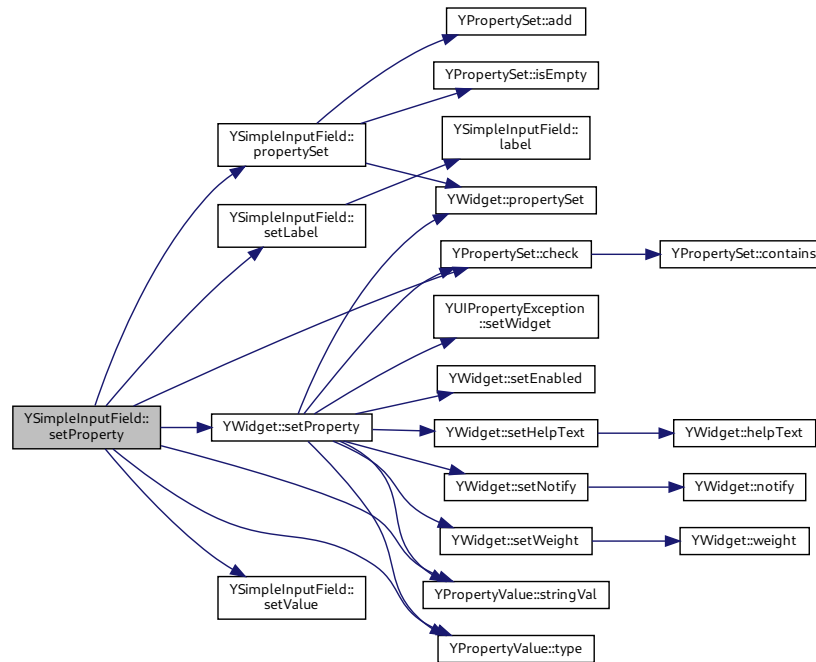
This function may throw `YUIPropertyExceptions`.

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from [YWidget](#).

Definition at line 98 of file [YSimpleInputField.cc](#).

Here is the call graph for this function:



### 5.130.2.5 setShortcutString()

```
virtual void YSimpleInputField::setShortcutString (
    const std::string & str ) [inline], [virtual]
```

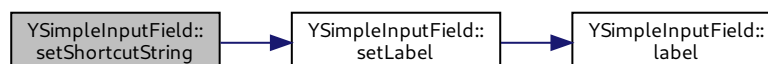
Set the string of this widget that holds the keyboard shortcut.

Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 121 of file [YSimpleInputField.h](#).

Here is the call graph for this function:



### 5.130.2.6 setValue()

```
virtual void YSimpleInputField::setValue (
    const std::string & text ) [pure virtual]
```

Set the current value (the text entered by the user or set from the outside) of this input field.

Derived classes are required to implement this.

### 5.130.2.7 shortcutString()

```
virtual std::string YSimpleInputField::shortcutString ( ) const [inline], [virtual]
```

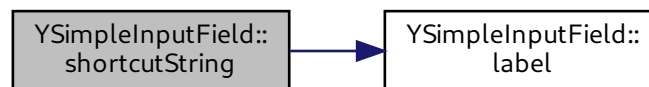
Get the string of this widget that holds the keyboard shortcut.

Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 114 of file [YSimpleInputField.h](#).

Here is the call graph for this function:



### 5.130.2.8 userInputProperty()

```
const char* YSimpleInputField::userInputProperty ( ) [inline], [virtual]
```

The name of the widget property that will return user input.

Inherited from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 128 of file [YSimpleInputField.h](#).

### 5.130.2.9 value()

```
virtual std::string YSimpleInputField::value ( ) [pure virtual]
```

Get the current value (the text entered by the user or set from the outside) of this input field.

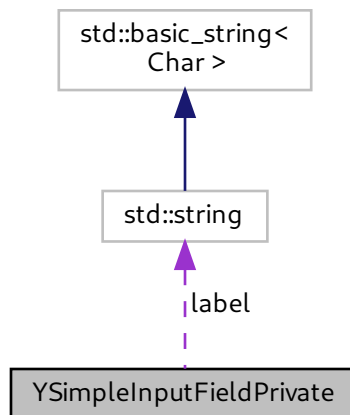
Derived classes are required to implement this.

The documentation for this class was generated from the following files:

- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YSimpleInputField.h
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YSimpleInputField.cc

## 5.131 YSimpleInputFieldPrivate Struct Reference

Collaboration diagram for YSimpleInputFieldPrivate:



### Public Member Functions

- **YSimpleInputFieldPrivate** (const string &label)

### Public Attributes

- string **label**

### 5.131.1 Detailed Description

Definition at line 35 of file [YSimpleInputField.cc](#).

The documentation for this struct was generated from the following file:

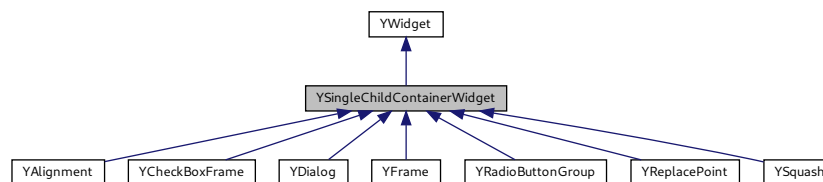
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YSimpleInputField.cc](#)

## 5.132 YSingleChildContainerWidget Class Reference

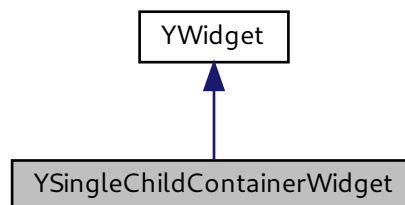
Container widget class that manages one child.

```
#include <YSingleChildContainerWidget.h>
```

Inheritance diagram for YSingleChildContainerWidget:



Collaboration diagram for YSingleChildContainerWidget:



## Public Member Functions

- virtual [~YSingleChildContainerWidget](#) ()  
*Destructor.*
- virtual int [preferredWidth](#) ()  
*Preferred width of the widget.*
- virtual int [preferredHeight](#) ()  
*Preferred height of the widget.*
- virtual void [setSize](#) (int newWidth, int newHeight)  
*Set the new size of the widget.*
- virtual bool [stretchable](#) (YUIDimension dim) const  
*Returns 'true' if this widget is stretchable in the specified dimension.*

## Protected Member Functions

- [YSingleChildContainerWidget](#) (YWidget \*parent)  
*Constructor.*

### 5.132.1 Detailed Description

Container widget class that manages one child.

Definition at line 34 of file [YSingleChildContainerWidget.h](#).

### 5.132.2 Member Function Documentation

#### 5.132.2.1 preferredHeight()

```
int YSingleChildContainerWidget::preferredHeight ( ) [virtual]
```

Preferred height of the widget.

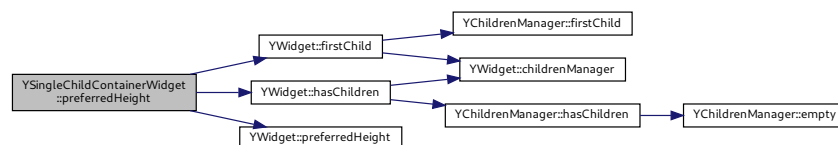
Reimplemented from [YWidget](#).

Implements [YWidget](#).

Reimplemented in [YAlignment](#).

Definition at line 51 of file [YSingleChildContainerWidget.cc](#).

Here is the call graph for this function:



### 5.132.2.2 preferredWidth()

```
int YSingleChildContainerWidget::preferredWidth ( ) [virtual]
```

Preferred width of the widget.

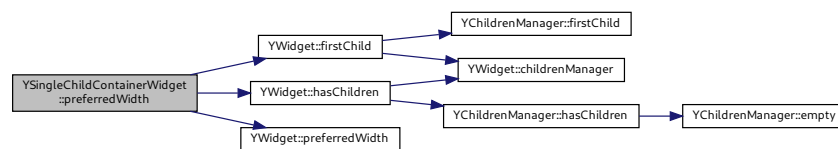
Reimplemented from [YWidget](#).

Implements [YWidget](#).

Reimplemented in [YAlignment](#).

Definition at line 42 of file [YSingleChildContainerWidget.cc](#).

Here is the call graph for this function:



### 5.132.2.3 setSize()

```
void YSingleChildContainerWidget::setSize (
    int newWidth,
    int newHeight ) [virtual]
```

Set the new size of the widget.

In this case, the size of the single child is set.

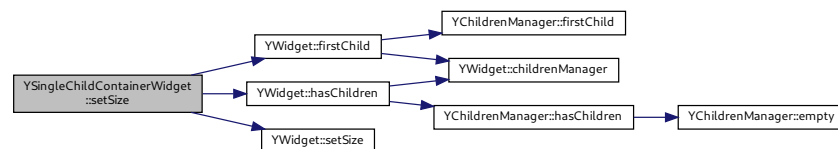
Reimplemented from [YWidget](#).

Implements [YWidget](#).

Reimplemented in [YAlignment](#).

Definition at line 60 of file [YSingleChildContainerWidget.cc](#).

Here is the call graph for this function:



### 5.132.2.4 stretchable()

```
bool YSingleChildContainerWidget::stretchable (
    YUIDimension dim ) const [virtual]
```

Returns 'true' if this widget is stretchable in the specified dimension.

In this case, the stretchability of the single child is returned.

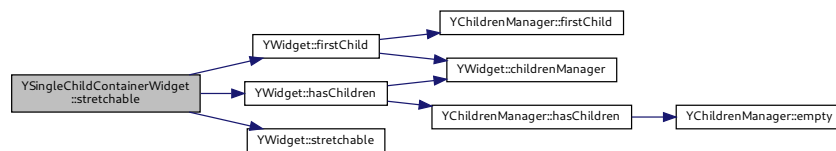
Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Reimplemented in [YSquash](#), and [YAlignment](#).

Definition at line 68 of file [YSingleChildContainerWidget.cc](#).

Here is the call graph for this function:



The documentation for this class was generated from the following files:

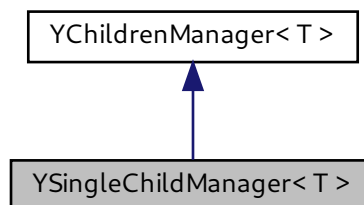
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YSingleChildContainerWidget.h](#)
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YSingleChildContainerWidget.cc](#)

## 5.133 YSingleChildManager< T > Class Template Reference

Children manager that can handle one single child (rejecting any more).

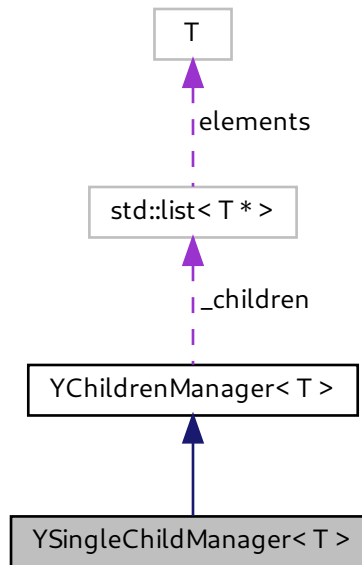
```
#include <YChildrenManager.h>
```

Inheritance diagram for YSingleChildManager< T >:





Collaboration diagram for YSingleChildManager< T >:



## Public Member Functions

- **YSingleChildManager** (T \*containerParent)
- virtual void **add** (T \*child)  
*Add a new child.*
- void **replace** (T \*newChild)  
*Replace the previous child (if any) with a new one.*

## Additional Inherited Members

### 5.133.1 Detailed Description

```
template<class T>
class YSingleChildManager< T >
```

Children manager that can handle one single child (rejecting any more).

Useful for [YAlignment](#), [YFrame](#) etc.

Definition at line 173 of file [YChildrenManager.h](#).

## 5.133.2 Member Function Documentation

### 5.133.2.1 add()

```
template<class T >
virtual void YSingleChildManager< T >::add (
    T * child ) [inline], [virtual]
```

Add a new child.

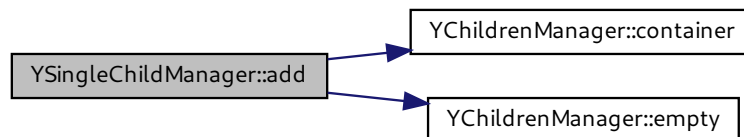
Reimplemented from [YChildrenManager](#).

This will throw a [YUITooManyChildrenException](#) if there already is a child.

Reimplemented from [YChildrenManager< T >](#).

Definition at line 189 of file [YChildrenManager.h](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

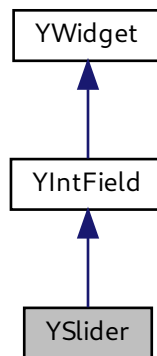
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YChildrenManager.h`

## 5.134 YSlider Class Reference

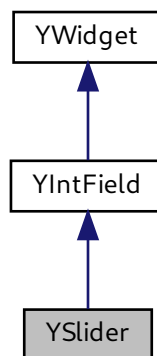
Slider: Input widget for an integer value between a minimum and a maximum value.

```
#include <YSlider.h>
```

Inheritance diagram for YSlider:



Collaboration diagram for YSlider:



## Public Member Functions

- virtual [~YSlider](#) ()  
*Destructor.*
- virtual const char \* [widgetClass](#) () const  
*Returns a descriptive name of this widget class for logging, debugging etc.*

## Protected Member Functions

- [YSlider](#) ([YWidget](#) \*[parent](#), const std::string &[label](#), int [minValue](#), int [maxValue](#))  
*Constructor.*

### 5.134.1 Detailed Description

Slider: Input widget for an integer value between a minimum and a maximum value.

Very similar to [IntField](#) in semantics, but with a graphical slider that can be dragged to the desired value. It also contains an [IntField](#) to allow entering the value directly.

Don't confuse this widget with [ProgressBar](#): [ProgressBar](#) is output-only.

This is an optional widget, i.e. not all UIs support it.

Definition at line [44](#) of file [YSlider.h](#).

### 5.134.2 Constructor & Destructor Documentation

#### 5.134.2.1 YSlider()

```
YSlider::YSlider (  
    YWidget * parent,  
    const std::string & label,  
    int minValue,  
    int maxValue ) [protected]
```

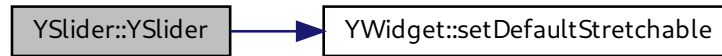
Constructor.

Create a Slider with 'label' as the caption, and the specified minimum and maximum values.

Note that [YWidgetFactory::createSlider\(\)](#) also has an 'initialValue' parameter that is not used here (because the current value is not stored in this base class, but in the derived class).

Definition at line [45](#) of file [YSlider.cc](#).

Here is the call graph for this function:



The documentation for this class was generated from the following files:

- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YSlider.h
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YSlider.cc

## 5.135 YSliderPrivate Struct Reference

### Public Attributes

- bool **dummy**

### 5.135.1 Detailed Description

Definition at line 34 of file [YSlider.cc](#).

The documentation for this struct was generated from the following file:

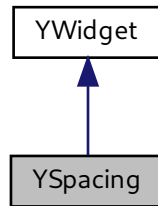
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YSlider.cc

## 5.136 YSpacing Class Reference

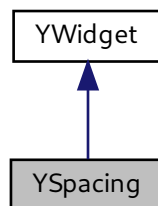
HSpacing, VSpacing, HStretch, VStretch.

```
#include <YSpacing.h>
```

Inheritance diagram for YSpacing:



Collaboration diagram for YSpacing:



## Public Member Functions

- [YSpacing](#) ([YWidget](#) \*parent, YUIDimension dim, bool [stretchable](#)=false, YLayoutSize\_t layoutUnits=0.0)  
*Constructor.*
- virtual [~YSpacing](#) ()  
*Destructor.*
- virtual const char \* [widgetClass](#) () const  
*Return a descriptive name of this widget class for logging, debugging etc.*
- YUIDimension [dimension](#) () const  
*Return the primary dimension of this Spacing/Stretch, i.e.*
- int [size](#) () const  
*Return the size in the primary dimension.*
- int [size](#) (YUIDimension dim) const  
*Return the size in the specified dimension.*
- virtual int [preferredWidth](#) ()  
*Preferred width of the widget.*
- virtual int [preferredHeight](#) ()  
*Preferred height of the widget.*

## Additional Inherited Members

### 5.136.1 Detailed Description

HSpacing, VSpacing, HStretch, VStretch.

Definition at line 37 of file [YSpacing.h](#).

### 5.136.2 Constructor & Destructor Documentation

#### 5.136.2.1 YSpacing()

```
YSpacing::YSpacing (
    YWidget * parent,
    YUIDimension dim,
    bool stretchable = false,
    YLayoutSize_t layoutUnits = 0.0 )
```

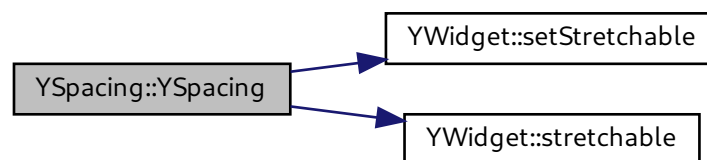
Constructor.

A Spacing/Stretch widget works only in one dimension ('dim') at the same time. But it can be stretchable and have a size at the same time, in which case the specified size acts very much like a minimal size - but not exactly, since [YLayoutBox](#) will reduce Spacings first before other widgets have to be resized below their preferred size.

'layoutUnits' is specified in abstract UI units where a main window (800x600 pixels in the Qt UI) corresponds to a 80x25 window.

Definition at line 45 of file [YSpacing.cc](#).

Here is the call graph for this function:



### 5.136.3 Member Function Documentation

#### 5.136.3.1 dimension()

```
YUIDimension YSpacing::dimension ( ) const
```

Return the primary dimension of this Spacing/Stretch, i.e.

the dimension in which it uses space or stretches.

Definition at line 62 of file [YSpacing.cc](#).

#### 5.136.3.2 preferredHeight()

```
int YSpacing::preferredHeight ( ) [virtual]
```

Preferred height of the widget.

Reimplemented from [YWidget](#).

Implements [YWidget](#).

Definition at line 90 of file [YSpacing.cc](#).

#### 5.136.3.3 preferredWidth()

```
int YSpacing::preferredWidth ( ) [virtual]
```

Preferred width of the widget.

Reimplemented from [YWidget](#).

Implements [YWidget](#).

Definition at line 81 of file [YSpacing.cc](#).



#### 5.136.3.4 size() [1/2]

```
int YSpacing::size ( ) const
```

Return the size in the primary dimension.

This is the device dependent size (pixels or character cells), not the abstract UI layout unit from the constructor.

Definition at line 68 of file [YSpacing.cc](#).

#### 5.136.3.5 size() [2/2]

```
int YSpacing::size (
    YUIDimension dim ) const
```

Return the size in the specified dimension.

This is the device dependent size (pixels or character cells), not the abstract UI layout unit from the constructor.

Definition at line 74 of file [YSpacing.cc](#).

The documentation for this class was generated from the following files:

- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YSpacing.h](#)
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YSpacing.cc](#)

## 5.137 YSpacingPrivate Struct Reference

### Public Member Functions

- **YSpacingPrivate** (YUIDimension dim, int size)

### Public Attributes

- YUIDimension **dim**
- int **size**

#### 5.137.1 Detailed Description

Definition at line 31 of file [YSpacing.cc](#).

The documentation for this struct was generated from the following file:

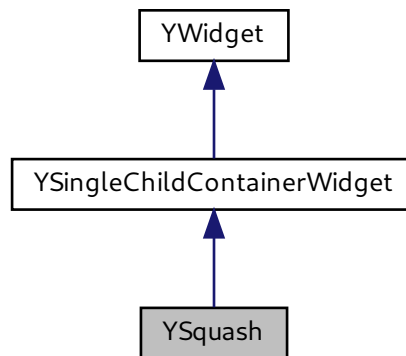
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YSpacing.cc](#)

## 5.138 YSquash Class Reference

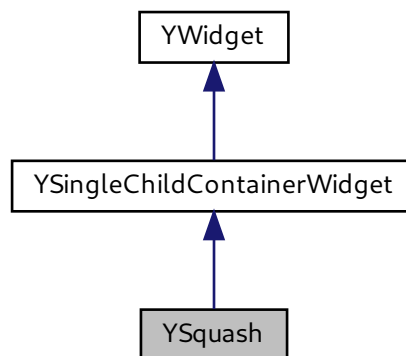
HSquash, VSquash HVSquash: reduce child to its preferred size.

```
#include <YSquash.h>
```

Inheritance diagram for YSquash:



Collaboration diagram for YSquash:



## Public Member Functions

- virtual `~YSquash ()`  
*Destructor.*
- virtual const char \* `widgetClass () const`  
*Returns a descriptive name of this widget class for logging, debugging etc.*
- bool `horSquash () const`  
*Returns 'true' if this widget squashes horizontally.*
- bool `vertSquash () const`  
*Returns 'true' if this widget squashes vertically.*
- bool `stretchable (YUIDimension dim) const`  
*In a squashed dimension the widget NOT stretchable.*

## Protected Member Functions

- `YSquash (YWidget *parent, bool horSquash, bool vertSquash)`  
*Constructor.*

### 5.138.1 Detailed Description

HSquash, VSquash HVSquash: reduce child to its preferred size.

Squash is a widget that "squashes" its one child during layout, i.e., it reduces it in size down to its preferred size. It may squash vertically, horizontally or in both dimensions.

Definition at line 41 of file [YSquash.h](#).

### 5.138.2 Constructor & Destructor Documentation

#### 5.138.2.1 YSquash()

```
YSquash::YSquash (  
    YWidget * parent,  
    bool horSquash,  
    bool vertSquash ) [protected]
```

Constructor.

Squashes horizontally if 'horSquash' is 'true', vertically if 'vertSquash' is 'true'.

Definition at line 44 of file [YSquash.cc](#).

### 5.138.3 Member Function Documentation

#### 5.138.3.1 stretchable()

```
bool YSquash::stretchable (
    YUIDimension dim ) const [virtual]
```

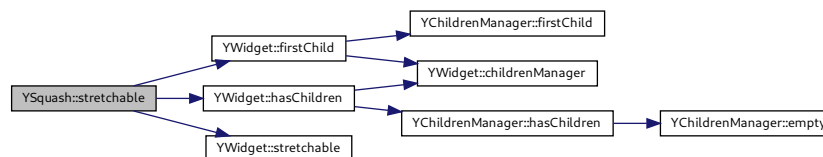
In a squashed dimension the widget NOT stretchable.

In an unsquashed dimension the widget is stretchable if the child is stretchable.

Reimplemented from [YSingleChildContainerWidget](#).

Definition at line 70 of file [YSquash.cc](#).

Here is the call graph for this function:

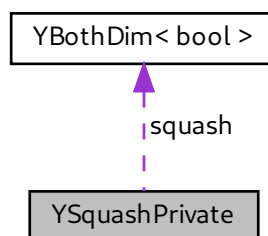


The documentation for this class was generated from the following files:

- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YSquash.h](#)
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YSquash.cc](#)

## 5.139 YSquashPrivate Struct Reference

Collaboration diagram for YSquashPrivate:



## Public Member Functions

- [YSquashPrivate](#) (bool horSquash, bool vertSquash)  
*Constructor.*

## Public Attributes

- [YBothDim](#)< bool > **squash**

### 5.139.1 Detailed Description

Definition at line 29 of file [YSquash.cc](#).

The documentation for this struct was generated from the following file:

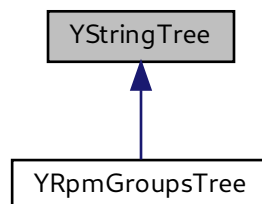
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YSquash.cc

## 5.140 YStringTree Class Reference

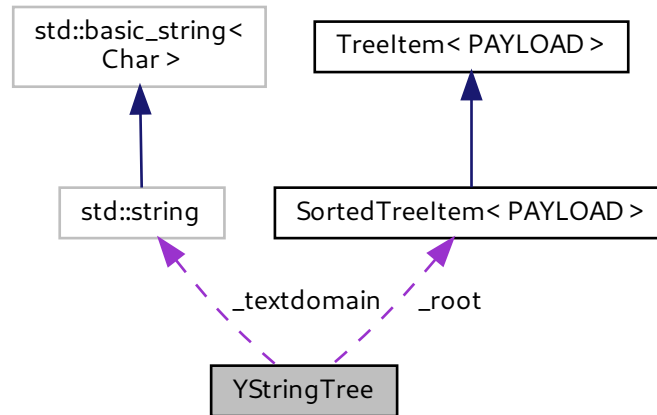
Abstract base class for filter views with hierarchical filter criteria - e.g., RPM group tags, MIME types.

```
#include <YStringTree.h>
```

Inheritance diagram for YStringTree:



Collaboration diagram for YStringTree:



## Public Member Functions

- [YStringTree](#) (const char \*[textdomain](#))  
*Constructor.*
- virtual [~YStringTree](#) ()  
*Destructor.*
- [YStringTreeItem](#) \* [addBranch](#) (const std::string &content, char delimiter=0, [YStringTreeItem](#) \*parent=0)  
*Add a unique new branch with text content 'content' to the tree, beginning at 'parent' (root if parent == 0).*
- std::string [origPath](#) (const [YStringTreeItem](#) \*item, char delimiter, bool startWithDelimiter=true)  
*Construct a complete original path for the specified tree item.*
- std::string [translatedPath](#) (const [YStringTreeItem](#) \*item, char delimiter, bool startWithDelimiter=true)  
*Construct a complete original path for the specified tree item.*
- [YTransText](#) path (const [YStringTreeItem](#) \*item, char delimiter, bool startWithDelimiter=true)  
*Construct a complete path (both original and translated) for the specified tree item.*
- void [logTree](#) ()  
*Debugging - dump the tree into the log file.*
- [YStringTreeItem](#) \* [root](#) () const  
*Returns the root of the filter view tree.*
- const char \* [textdomain](#) () const  
*Returns the textdomain used internally for translation of pathname components.*
- void [setTextdomain](#) (const char \*domain)  
*Set the textdomain used internally for translation of pathname components.*
- std::string [translate](#) (const std::string &orig)  
*Translate message 'orig' using the internal textdomain.*

## Protected Member Functions

- `std::string completePath` (const [YStringTreeItem](#) \*item, bool translated, char delimiter, bool startWithDelimiter)  
*Construct a complete original or translated path for the specified tree item.*
- `void logBranch` ([YStringTreeItem](#) \*branch, std::string indentation)  
*Debugging - dump one branch of the tree into the log file.*

## Protected Attributes

- [YStringTreeItem](#) \* `_root`
- std::string `_textdomain`

### 5.140.1 Detailed Description

Abstract base class for filter views with hierarchical filter criteria - e.g., RPM group tags, MIME types.

Definition at line 41 of file [YStringTree.h](#).

### 5.140.2 Constructor & Destructor Documentation

#### 5.140.2.1 YStringTree()

```
YStringTree::YStringTree (
    const char * textdomain )
```

Constructor.

'textdomain' specifies the gettext textdomain to use to translate pathname components as new branches are added.

NOTE: This will NOT change the gettext environment in any way - the tree uses `dgettext()` internally. The caller is responsible to bind that textdomain to a message catalog (`bindtextdomain()` etc.).

Definition at line 33 of file [YStringTree.cc](#).

Here is the call graph for this function:



### 5.140.3 Member Function Documentation

#### 5.140.3.1 addBranch()

```
YStringTreeItem * YStringTree::addBranch (
    const std::string & content,
    char delimiter = 0,
    YStringTreeItem * parent = 0 )
```

Add a unique new branch with text content 'content' to the tree, beginning at 'parent' (root if parent == 0).

This content can be a path specification delimited with character 'delimiter' (if not 0), i.e. this method will split 'content' up into path components and insert tree items for each level as appropriate. Leading delimiters will be ignored. If 'delimiter' is 0, 'content' is not split but used 'as is'. Items are automatically sorted alphabetically. Pathname components are automatically translated using the textdomain specified in the constructor.

Returns the tree node for this branch - either newly created or the existing one.

Example: `addBranch( "/usr/local/bin", '/' )` `addBranch( "/usr/lib", '/' )`

"usr" "lib" "local" "bin"

Definition at line 49 of file [YStringTree.cc](#).

Here is the call graph for this function:





### 5.140.3.2 completePath()

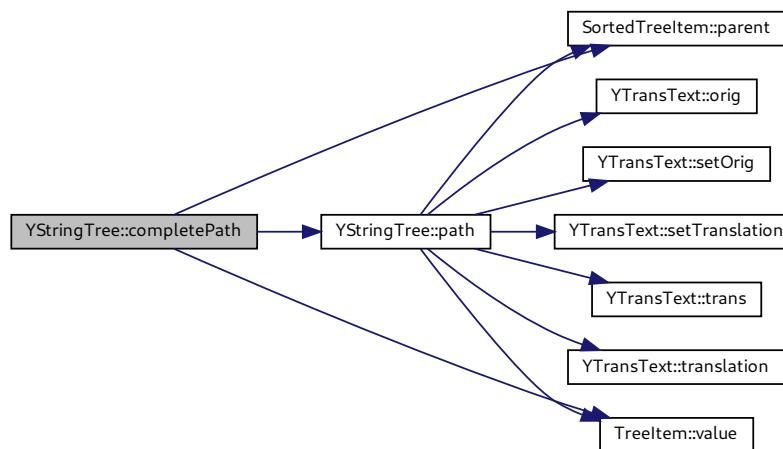
```
string YStringTree::completePath (
    const YStringTreeItem * item,
    bool translated,
    char delimiter,
    bool startWithDelimiter ) [protected]
```

Construct a complete original or translated path for the specified tree item.

'startWithDelimiter' specifies whether or not the complete path should start with the delimiter character.

Definition at line 128 of file [YStringTree.cc](#).

Here is the call graph for this function:



### 5.140.3.3 origPath()

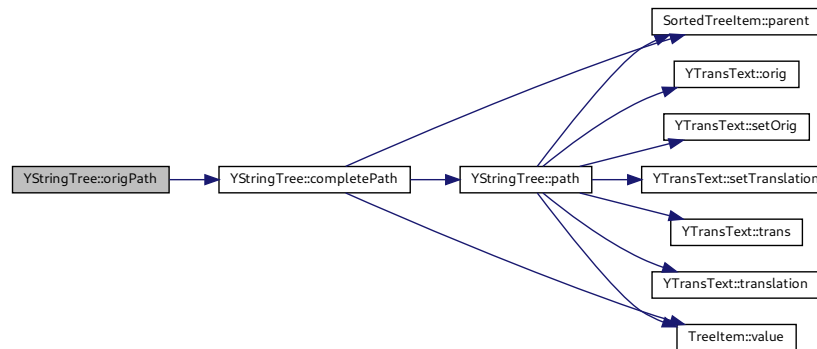
```
std::string YStringTree::origPath (
    const YStringTreeItem * item,
    char delimiter,
    bool startWithDelimiter = true ) [inline]
```

Construct a complete original path for the specified tree item.

'startWithDelimiter' specifies whether or not the complete path should start with the delimiter character.

Definition at line 97 of file [YStringTree.h](#).

Here is the call graph for this function:



#### 5.140.3.4 path()

```

YTransText YStringTree::path (
    const YStringTreeItem * item,
    char delimiter,
    bool startWithDelimiter = true )
  
```

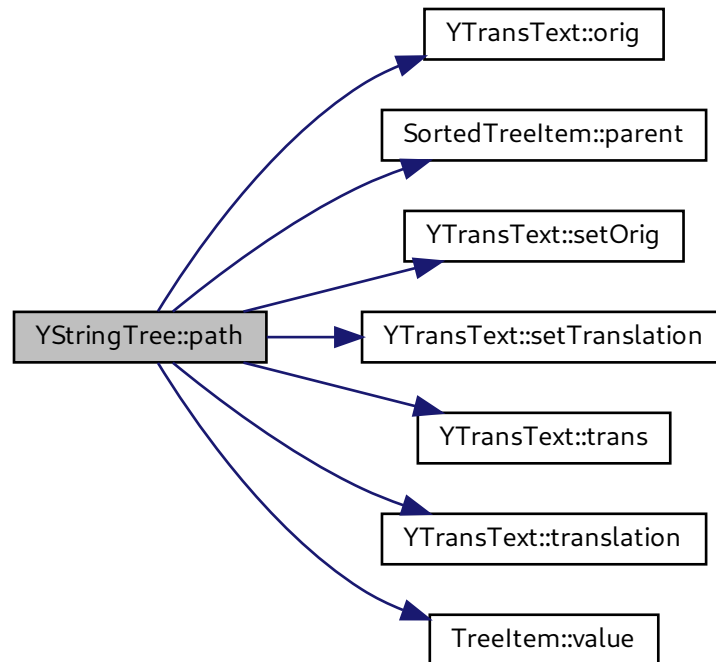
Construct a complete path (both original and translated) for the specified tree item.

'startWithDelimiter' specifies whether or not the complete path should start with the delimiter character.

Note: [origPath\(\)](#) or [translatedPath\(\)](#) are much cheaper if only one version (original or translated) is required.

Definition at line 159 of file [YStringTree.cc](#).

Here is the call graph for this function:



#### 5.140.3.5 root()

```
YStringTreeItem* YStringTree::root ( ) const [inline]
```

Returns the root of the filter view tree.

Note: In most cases, the root item itself will not contain any useful information. Consider it the handle for the entire tree, not an actual data element.

Definition at line 139 of file [YStringTree.h](#).

### 5.140.3.6 setTextdomain()

```
void YStringTree::setTextdomain (
    const char * domain ) [inline]
```

Set the textdomain used internally for translation of pathname components.

NOTE: This will NOT change the gettext environment in any way - the tree uses dgettext() internally. The caller is responsible to bind that textdomain to a message catalog (bindtextdomain() etc.).

Definition at line 157 of file [YStringTree.h](#).

### 5.140.3.7 translate()

```
string YStringTree::translate (
    const std::string & orig )
```

Translate message 'orig' using the internal textdomain.

Returns the translated text or the original if there is no translation.

Definition at line 119 of file [YStringTree.cc](#).

### 5.140.3.8 translatedPath()

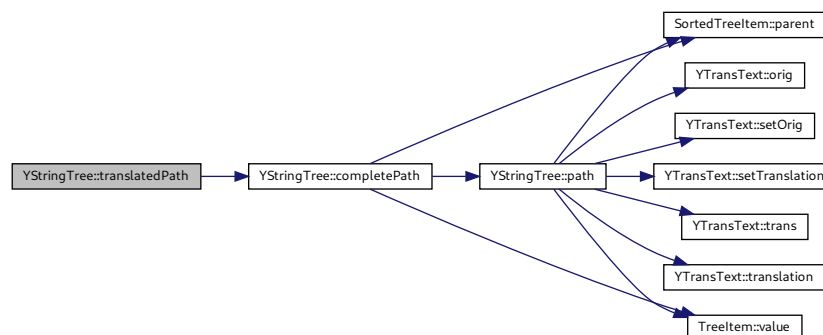
```
std::string YStringTree::translatedPath (
    const YStringTreeItem * item,
    char delimiter,
    bool startWithDelimiter = true ) [inline]
```

Construct a complete original path for the specified tree item.

'startWithDelimiter' specifies whether or not the complete path should start with the delimiter character.

Definition at line 108 of file [YStringTree.h](#).

Here is the call graph for this function:



The documentation for this class was generated from the following files:

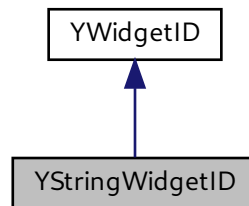
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YStringTree.h
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YStringTree.cc

## 5.141 YStringWidgetID Class Reference

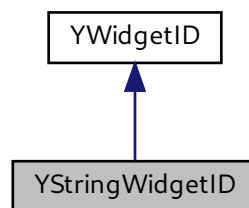
Simple widget ID class based on strings.

```
#include <YWidgetID.h>
```

Inheritance diagram for YStringWidgetID:



Collaboration diagram for YStringWidgetID:



### Public Member Functions

- [YStringWidgetID](#) (const std::string &[value](#))  
*Constructor.*
- virtual [~YStringWidgetID](#) ()  
*Destructor.*

- virtual bool `isEqual` (`YWidgetID` \*otherID) const  
*Check if this ID is equal to another.*
- virtual std::string `toString` () const  
*Convert the ID value to string.*
- std::string `value` () const  
*Return the ID value.*
- const std::string & `valueConstRef` () const  
*Return the ID value as a const ref.*

## Additional Inherited Members

### 5.141.1 Detailed Description

Simple widget ID class based on strings.

Definition at line 72 of file `YWidgetID.h`.

### 5.141.2 Member Function Documentation

#### 5.141.2.1 `isEqual()`

```
bool YStringWidgetID::isEqual (
    YWidgetID * otherID ) const [virtual]
```

Check if this ID is equal to another.

Reimplemented from `YWidgetID`.

Implements `YWidgetID`.

Definition at line 46 of file `YWidgetID.cc`.

Here is the call graph for this function:



### 5.141.2.2 toString()

```
string YStringWidgetID::toString ( ) const [virtual]
```

Convert the ID value to string.

Used for logging and debugging.

Reimplemented from [YWidgetID](#).

Implements [YWidgetID](#).

Definition at line 59 of file [YWidgetID.cc](#).

The documentation for this class was generated from the following files:

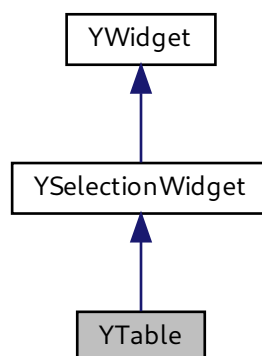
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YWidgetID.h
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YWidgetID.cc

## 5.142 YTable Class Reference

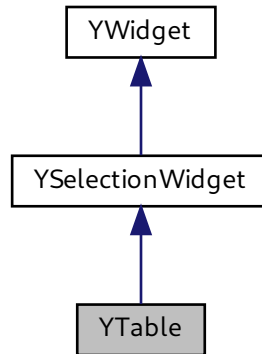
Table: Selection list with multiple columns.

```
#include <YTable.h>
```

Inheritance diagram for YTable:



Collaboration diagram for YTable:



## Public Member Functions

- virtual `~YTable()`  
*Destructor.*
- virtual const char \* `widgetClass()` const  
*Return a descriptive name of this widget class for logging, debugging etc.*
- int `columns()` const  
*Return the number of columns of this table.*
- bool `hasColumn(int column)` const  
*Return 'true' if this table has a column no.*
- std::string `header(int column)` const  
*Return the header text for the specified column.*
- YAlignmentType `alignment(int column)` const  
*Return the alignment for the specified column.*
- bool `immediateMode()` const  
*Deliver even more events than with `notify()` set.*
- void `setImmediateMode(bool immediateMode=true)`  
*Set `immediateMode()` on or off.*
- bool `keepSorting()` const  
*Return 'true' if the sort order is to be kept in item insertion order, i.e.*
- virtual void `setKeepSorting(bool keepSorting)`  
*Switch between sorting by item insertion order (`keepSorting: true`) or allowing the user to sort by an arbitrary column (by clicking on the column header).*
- bool `hasMultiSelection()` const  
*Return 'true' if the user can select multiple items at the same time (e.g., with shift-click or ctrl-click).*
- YItem \* `findItem(const std::string &wantedItemLabel, int column)` const  
*Try to find an item with label 'wantedItemLabel' in column 'column' between iterators 'begin' and 'end'.*



- `YItem * findItem` (const std::string &wantedItemLabel, int column, [YItemConstIterator](#) begin, [YItemConstIterator](#) end) const
- virtual void `cellChanged` (const [YTableCell](#) \*cell)=0  
*Notification that a cell (its text, its icon and/or sort-key) was changed from the outside.*
- virtual bool `setProperty` (const std::string &propertyName, const [YPropertyValue](#) &val)  
*Set a property.*
- virtual [YPropertyValue](#) `getProperty` (const std::string &propertyName)  
*Get a property.*
- virtual const [YPropertySet](#) & `propertySet` ()  
*Return this class's property set.*
- const char \* `userInputProperty` ()  
*The name of the widget property that will return user input.*

## Protected Member Functions

- `YTable` ([YWidget](#) \*parent, [YTableHeader](#) \*header, bool multiSelection)  
*Constructor.*
- void `setTableHeader` ([YTableHeader](#) \*newHeader)  
*Exchange the previous table header with a new one.*

### 5.142.1 Detailed Description

Table: Selection list with multiple columns.

The user can select exactly one row (with all its columns) from that list. Each cell (each column within each row) has a label text, an optional icon (\*) and an optional sort-key (used instead of the label text during sort).

This widget is similar to `SelectionBox`, but it has several columns for each item (each row). If just one column is desired, consider using `SelectionBox` instead.

Note: This is not something like a spread sheet, and it doesn't pretend or want to be. Actions are performed on rows, not on individual cells (columns within one row).

(\*) Not all UIs (in particular not text-based UIs) support displaying icons, so an icon should never be an exclusive means to display any kind of information.

Definition at line 56 of file [YTable.h](#).

### 5.142.2 Constructor & Destructor Documentation

### 5.142.2.1 YTable()

```
YTable::YTable (
    YWidget * parent,
    YTableHeader * header,
    bool multiSelection ) [protected]
```

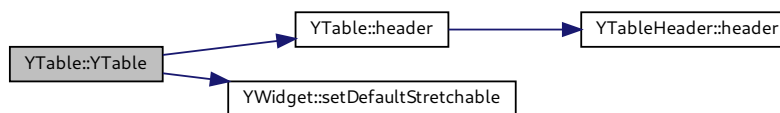
Constructor.

'header' describes the table's headers: Number of columns, column headings, and column alignment. The widget assumes ownership of this object and will delete it when appropriate. The header cannot be changed after creating the widget.

'multiSelection' indicates whether or not the user can select multiple items at the same time (e.g., with shift-click or ctrl-click). This can only be set in the constructor.

Definition at line 52 of file [YTable.cc](#).

Here is the call graph for this function:



## 5.142.3 Member Function Documentation

### 5.142.3.1 cellChanged()

```
virtual void YTable::cellChanged (
    const YTableCell * cell ) [pure virtual]
```

Notification that a cell (its text, its icon and/or sort-key) was changed from the outside.

Applications are required to call this whenever a table cell is changed after adding the corresponding table item (the row) to the table widget.

Derived classes are required to implement this and update the display accordingly.

Note that the position of this cell can be retrieved with `cell->column()` and `cell->itemIndex()`.

### 5.142.3.2 findItem()

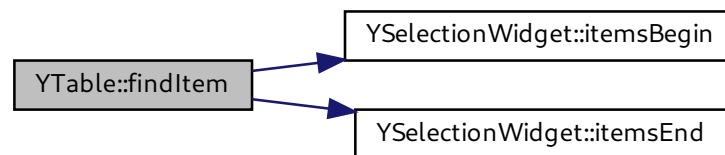
```
YItem * YTable::findItem (
    const std::string & wantedItemLabel,
    int column ) const
```

Try to find an item with label 'wantedItemLabel' in column 'column' between iterators 'begin' and 'end'.

Return that item or 0 if there is none.

Definition at line 152 of file [YTable.cc](#).

Here is the call graph for this function:



### 5.142.3.3 getProperty()

```
YPropertyValue YTable::getProperty (
    const std::string & propertyName ) [virtual]
```

Get a property.

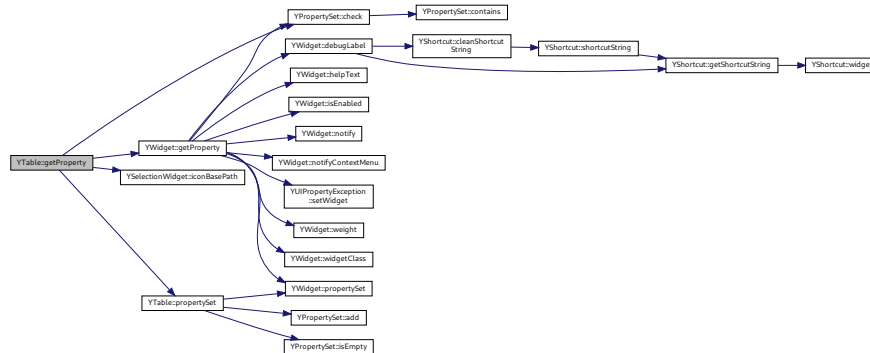
Reimplemented from [YWidget](#).

This method may throw `YUIPropertyExceptions`.

Reimplemented from [YWidget](#).

Definition at line 235 of file [YTable.cc](#).

Here is the call graph for this function:



#### 5.142.3.4 hasColumn()

```
bool YTable::hasColumn (
    int column ) const
```

Return 'true' if this table has a column no.

'column' (counting from 0 on).

Definition at line 94 of file [YTable.cc](#).

Here is the call graph for this function:



#### 5.142.3.5 immediateMode()

```
bool YTable::immediateMode ( ) const
```

Deliver even more events than with [notify\(\)](#) set.

With "notify" alone, a table widget sends an ActivatedEvent when the user double-clicks an item or presses the "space" key on it. It does not send an event when the user just sends another item.

With "immediate", it also sends a SelectionChangedEvent when the user selects another item. "immediate" implicitly includes "notify".

Definition at line 115 of file [YTable.cc](#).

### 5.142.3.6 keepSorting()

```
bool YTable::keepSorting ( ) const
```

Return 'true' if the sort order is to be kept in item insertion order, i.e.

if sorting the table by clicking on a column header should be disabled.

Definition at line 132 of file [YTable.cc](#).

### 5.142.3.7 propertySet()

```
const YPropertySet & YTable::propertySet ( ) [virtual]
```

Return this class's property set.

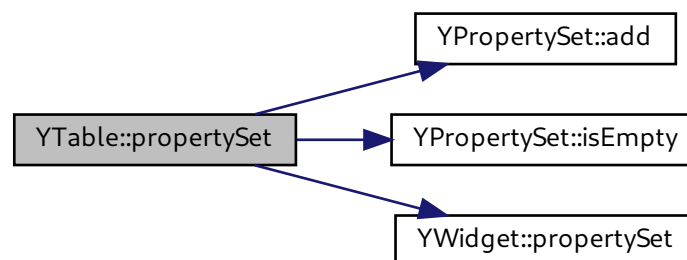
This also initializes the property upon the first call.

Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 179 of file [YTable.cc](#).

Here is the call graph for this function:



### 5.142.3.8 setKeepSorting()

```
void YTable::setKeepSorting (
    bool keepSorting ) [virtual]
```

Switch between sorting by item insertion order (`keepSorting: true`) or allowing the user to sort by an arbitrary column (by clicking on the column header).

Derived classes can overwrite this function, but they should call this base class function in the new implementation.

Definition at line 139 of file [YTable.cc](#).

Here is the call graph for this function:



### 5.142.3.9 setProperty()

```
bool YTable::setProperty (
    const std::string & propertyName,
    const YPropertyValue & val ) [virtual]
```

Set a property.

Reimplemented from [YWidget](#).

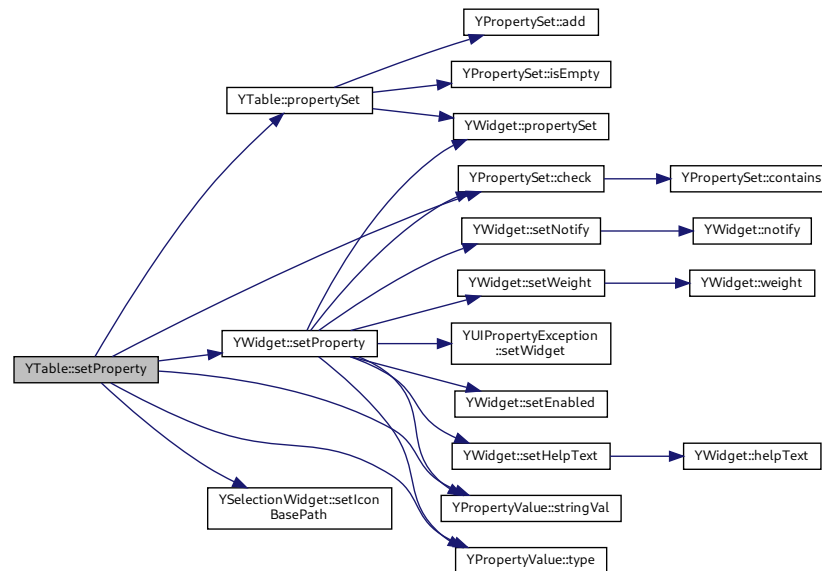
This function may throw `YUIPropertyExceptions`.

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from [YWidget](#).

Definition at line 214 of file [YTable.cc](#).

Here is the call graph for this function:



#### 5.142.3.10 setTableHeader()

```
void YTable::setTableHeader (
    YTableHeader * newHeader ) [protected]
```

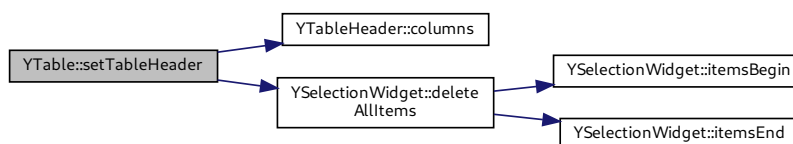
Exchange the previous table header with a new one.

This will delete the old [YTableHeader](#) object.

If the new header has a different number of columns than the old one, all items will implicitly be deleted.

Definition at line 74 of file [YTable.cc](#).

Here is the call graph for this function:



### 5.142.3.11 userInputProperty()

```
const char* YTable::userInputProperty ( ) [inline], [virtual]
```

The name of the widget property that will return user input.

Inherited from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 208 of file [YTable.h](#).

The documentation for this class was generated from the following files:

- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YTable.h
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YTable.cc

## 5.143 YTableCell Class Reference

One cell (one column in one row) of a [YTableItem](#).

```
#include <YTableWidgetItem.h>
```

### Public Member Functions

- [YTableCell](#) (const std::string &[label](#), const std::string &[iconName](#)="", const std::string &[sortKey](#)="")  
*Constructor with label and optional icon name and optional sort key for cells that don't have a parent item yet (that will be added to a parent later with [setParent\(\)](#)).*
- [YTableCell](#) ([YTableItem](#) \*[parent](#), int [column](#), const std::string &[label](#), const std::string &[iconName](#)="", const std::string &[sortKey](#)="")  
*Constructor with parent, column no., label and optional icon name for cells that are created with a parent.*
- virtual [~YTableCell](#) ()  
*Destructor.*
- std::string [label](#) () const  
*Return this cells's label.*
- void [setLabel](#) (const std::string &[newLabel](#))  
*Set this cell's label.*
- std::string [iconName](#) () const  
*Return this cell's icon name.*
- bool [hasIconName](#) () const  
*Return 'true' if this cell has an icon name.*
- void [setIconName](#) (const std::string &[newIconName](#))  
*Set this cell's icon name.*
- std::string [sortKey](#) () const  
*Return this cell's sort key.*
- bool [hasSortKey](#) () const



*Return 'true' if this cell has a sort key.*

- void [setSortKey](#) (const std::string &newSortKey)

*Set this cell's sort key.*

- [YTableItem](#) \* [parent](#) () const

*Return this cell's parent item or 0 if it doesn't have one yet.*

- int [column](#) () const

*Return this cell's column no.*

- int [itemIndex](#) () const

*Convenience function: Return this cell's parent item's index within its table widget or -1 if there is no parent item or no parent table.*

- void [reparent](#) ([YTableItem](#) \*parent, int column)

*Set this cell's parent item and column no.*

### 5.143.1 Detailed Description

One cell (one column in one row) of a [YTableItem](#).

Each cell has a label (a user visible text), optionally an icon (\*) and also optionally a sort-key.

Note that cells don't have individual IDs; they have just an index. The first cell in an item is cell(0). In an ideal world, each [YTableItem](#) would have exactly as many cells as there are columns in the [YTable](#), but these classes make no such assumptions. A [YTableItem](#) might have any number of cells, including none.

The [YTable](#) widget is free to ignore any excess cells if there are more than the [YTable](#) widget has columns. If there are less cells than the table has columns, the nonexistent cells will be treated as empty.

(\*) Not all UIs can handle icons. UIs that can't handle them will simply ignore any icons specified for YTableCells. Thus, applications should either check the UI capabilities if it can handle icons or use icons only as an additional visual cue that still has a text counterpart (so the user can still make sense of the table content when no icons are visible).

Definition at line 219 of file [YTableItem.h](#).

### 5.143.2 Constructor & Destructor Documentation

#### 5.143.2.1 ~YTableCell()

```
virtual YTableCell::~YTableCell ( ) [inline], [virtual]
```

Destructor.

Not strictly needed inside this class, but useful for derived classes. Since this is the only virtual method of this class, the cost of this is a vtable for this class and a pointer to the vtable in each instance.

Definition at line 258 of file [YTableItem.h](#).

### 5.143.3 Member Function Documentation

#### 5.143.3.1 column()

```
int YTableCell::column ( ) const [inline]
```

Return this cell's column no.

(counting from 0on) or -1 if it doesn't have a parent yet.

Definition at line 322 of file [YTableItem.h](#).

#### 5.143.3.2 label()

```
std::string YTableCell::label ( ) const [inline]
```

Return this cells's label.

This is what the user sees in a dialog, so this will usually be a translated text.

Definition at line 264 of file [YTableItem.h](#).

#### 5.143.3.3 reparent()

```
void YTableCell::reparent (
    YTableItem * parent,
    int column )
```

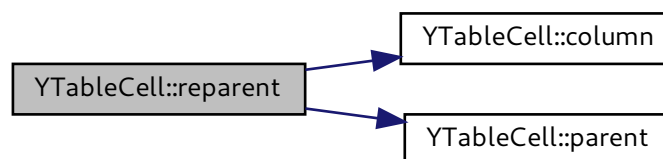
Set this cell's parent item and column no.

if it doesn't have a parent yet.

This method will throw an exception if the cell already has a parent.

Definition at line 173 of file [YTableItem.cc](#).

Here is the call graph for this function:



#### 5.143.3.4 setIconName()

```
void YTableCell::setIconName (
    const std::string & newIconName ) [inline]
```

Set this cell's icon name.

If this is called after the corresponding table item (table row) is added to the table widget, call [YTable::cellChanged\(\)](#) to notify the table widget about the fact. Only then will the display be updated.

Definition at line 292 of file [YTableItem.h](#).

#### 5.143.3.5 setLabel()

```
void YTableCell::setLabel (
    const std::string & newLabel ) [inline]
```

Set this cell's label.

If this is called after the corresponding table item (table row) is added to the table widget, call [YTable::cellChanged\(\)](#) to notify the table widget about the fact. Only then will the display be updated.

Definition at line 273 of file [YTableItem.h](#).

#### 5.143.3.6 setSortKey()

```
void YTableCell::setSortKey (
    const std::string & newSortKey ) [inline]
```

Set this cell's sort key.

If this is called after the corresponding table item (table row) is added to the table widget, call [YTable::cellChanged\(\)](#) to notify the table widget about the fact. Only then will the display be updated.

Definition at line 311 of file [YTableItem.h](#).

The documentation for this class was generated from the following files:

- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YTableItem.h](#)
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YTableItem.cc](#)

## 5.144 YTableHeader Class Reference

Helper class for [YTable](#) for table column properties:

```
#include <YTableHeader.h>
```

### Public Member Functions

- [YTableHeader](#) ()  
*Constructor.*
- virtual [~YTableHeader](#) ()  
*Destructor.*
- void [addColumn](#) (const std::string &[header](#), YAlignmentType [alignment](#)=YAlignBegin)  
*Add a column with the specified column header text and alignment.*
- int [columns](#) () const  
*Return the number of columns.*
- bool [hasColumn](#) (int column) const  
*Return 'true' if this table header has a column no.*
- std::string [header](#) (int column) const  
*Return the header text for the specified column.*
- YAlignmentType [alignment](#) (int column) const  
*Return the alignment for the specified column.*

### 5.144.1 Detailed Description

Helper class for [YTable](#) for table column properties:

- number of columns
- header for each column
- alignment for each column

Definition at line [43](#) of file [YTableHeader.h](#).

### 5.144.2 Member Function Documentation

### 5.144.2.1 hasColumn()

```
bool YTableHeader::hasColumn (
    int column ) const
```

Return 'true' if this table header has a column no.

'column' (counting from 0 on).

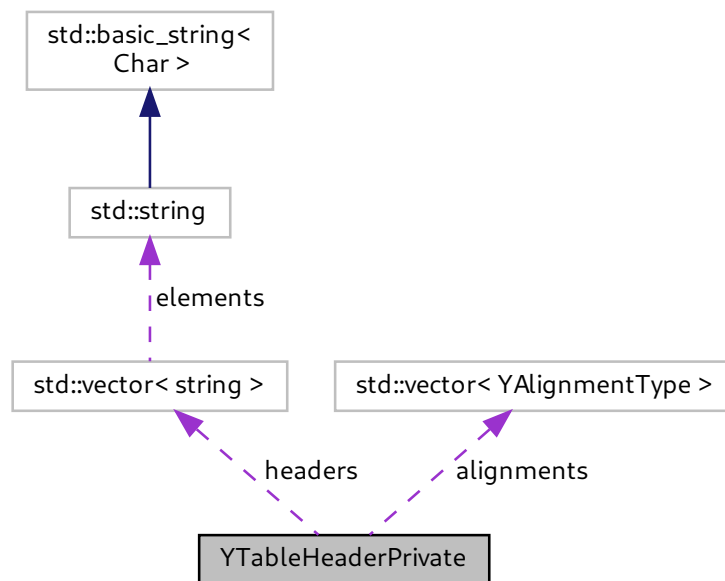
Definition at line 79 of file [YTableHeader.cc](#).

The documentation for this class was generated from the following files:

- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YTableHeader.h
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YTableHeader.cc

## 5.145 YTableHeaderPrivate Struct Reference

Collaboration diagram for YTableHeaderPrivate:



### Public Attributes

- `vector< string >` **headers**
- `vector< YAlignmentType >` **alignments**

### 5.145.1 Detailed Description

Definition at line 38 of file [YTableHeader.cc](#).

The documentation for this struct was generated from the following file:

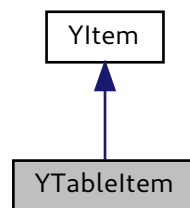
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YTableHeader.cc](#)

## 5.146 YTableItem Class Reference

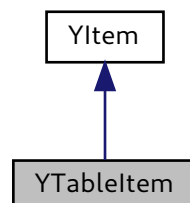
Item class for [YTable](#) items.

```
#include <YTableItem.h>
```

Inheritance diagram for YTableItem:



Collaboration diagram for YTableItem:



## Public Member Functions

- [YTableItem](#) ()  
*Default constructor.*
- [YTableItem](#) (const std::string &label\_0, const std::string &label\_1=std::string(), const std::string &label\_2=std::string(), const std::string &label\_3=std::string(), const std::string &label\_4=std::string(), const std::string &label\_5=std::string(), const std::string &label\_6=std::string(), const std::string &label\_7=std::string(), const std::string &label\_8=std::string(), const std::string &label\_9=std::string())  
*Convenience constructor for table items without any icons.*
- virtual [~YTableItem](#) ()  
*Destructor.*
- void [addCell](#) ([YTableCell](#) \*cell\_disown)  
*Add a cell.*
- void [addCell](#) (const std::string &label, const std::string &iconName=std::string(), const std::string &sortKey=std::string())  
*Create a new cell and add it (even if all 'label', 'iconName' and 'sortKey' are empty).*
- void [deleteCells](#) ()  
*Delete all cells.*
- [YTableCellIterator](#) [cellsBegin](#) ()  
*Return an iterator that points to the first cell of this item.*
- [YTableCellConstIterator](#) [cellsBegin](#) () const
- [YTableCellIterator](#) [cellsEnd](#) ()  
*Return an iterator that points after the last cell of this item.*
- [YTableCellConstIterator](#) [cellsEnd](#) () const
- const [YTableCell](#) \* [cell](#) (int [index](#)) const  
*Return the cell at the specified index (counting from 0 on) or 0 if there is none.*
- [YTableCell](#) \* [cell](#) (int [index](#))
- int [cellCount](#) () const  
*Return the number of cells this item has.*
- bool [hasCell](#) (int [index](#)) const  
*Return 'true' if this item has a cell with the specified index (counting from 0 on), 'false' otherwise.*
- std::string [label](#) (int [index](#)) const  
*Return the label of cell no.*
- std::string [iconName](#) (int [index](#)) const  
*Return the icon name of cell no.*
- bool [hasIconName](#) (int [index](#)) const  
*Return 'true' if there is a cell with the specified index that has an icon name.*
- std::string [label](#) () const  
*Just for debugging.*

### 5.146.1 Detailed Description

Item class for [YTable](#) items.

Each [YTableItem](#) corresponds to one row in a [YTable](#).

A [YTableItem](#) might have any number of cells (columns within this row), including none. The [YTable](#) widget is free to ignore any excess cells if there are more than the [YTable](#) widget has columns. The [YTable](#) widget is to treat nonexistent cells like empty ones.

Note that while [YTable](#) items and their cells can be manipulated through pointers, their visual representation on screen might be updated only upon calling certain methods of the [YTable](#) widget. See the [YTable](#) reference for details.

Definition at line 58 of file [YTableItem.h](#).

## 5.146.2 Constructor & Destructor Documentation

### 5.146.2.1 YTableItem() [1/2]

`YTableItem::YTableItem ( )`

Default constructor.

Use [addCell\(\)](#) to give it any content.

Definition at line 31 of file [YTableItem.cc](#).

### 5.146.2.2 YTableItem() [2/2]

```
YTableItem::YTableItem (
    const std::string & label_0,
    const std::string & label_1 = std::string(),
    const std::string & label_2 = std::string(),
    const std::string & label_3 = std::string(),
    const std::string & label_4 = std::string(),
    const std::string & label_5 = std::string(),
    const std::string & label_6 = std::string(),
    const std::string & label_7 = std::string(),
    const std::string & label_8 = std::string(),
    const std::string & label_9 = std::string() )
```

Convenience constructor for table items without any icons.

This will create up to 10 (0..9) cells. Empty cells for empty labels at the end of the labels are not created, but empty cells in between are.

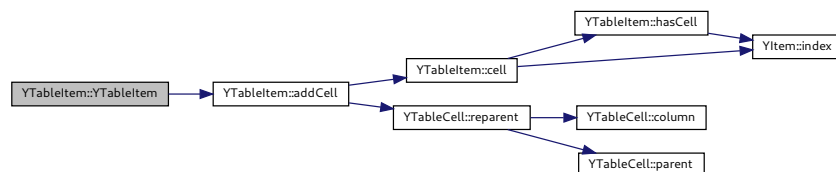
```
new YTableItem( "one", "two", "", "", "five" );
```

will create an item with 5 cells:

```
cell[0] ==> "one"
cell[1] ==> "two"
cell[2] ==> ""
cell[3] ==> ""
cell[4] ==> "five"
```

Definition at line 38 of file [YTableItem.cc](#).

Here is the call graph for this function:





### 5.146.2.3 ~YTableItem()

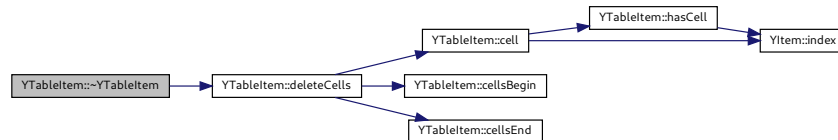
```
YTableItem::~YTableItem ( ) [virtual]
```

Destructor.

This will delete all cells.

Definition at line 84 of file [YTableItem.cc](#).

Here is the call graph for this function:



## 5.146.3 Member Function Documentation

### 5.146.3.1 addCell()

```
void YTableItem::addCell (
    YTableCell * cell_disown )
```

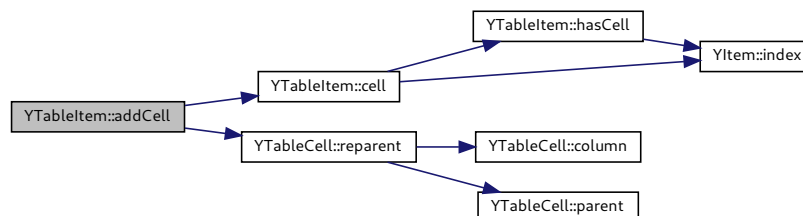
Add a cell.

This item will assume ownership over the cell and delete it when appropriate (when the table is destroyed or when table items are replaced), at which time the pointer will become invalid.

Cells can still be changed after they (and the item they belong to) are added, but in that case, [YTable::cellChanged\(\)](#) needs to be called to update the table display accordingly.

Definition at line 107 of file [YTableItem.cc](#).

Here is the call graph for this function:



### 5.146.3.2 iconName()

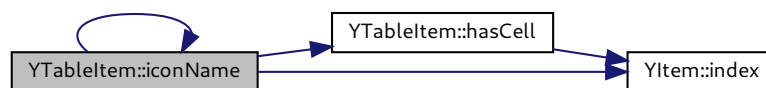
```
string YTableItem::iconName (  
    int index ) const
```

Return the icon name of cell no.

'index' (counting from 0 on) or an empty string if there is no cell with that index.

Definition at line 157 of file [YTableItem.cc](#).

Here is the call graph for this function:



### 5.146.3.3 label()

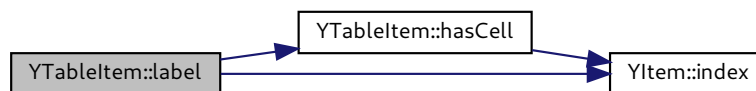
```
string YTableItem::label (  
    int index ) const
```

Return the label of cell no.

'index' (counting from 0 on) or an empty string if there is no cell with that index.

Definition at line 150 of file [YTableItem.cc](#).

Here is the call graph for this function:

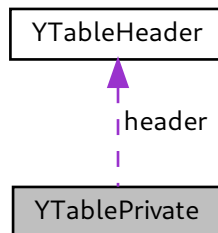


The documentation for this class was generated from the following files:

- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YTableItem.h](#)
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YTableItem.cc](#)

## 5.147 YTablePrivate Struct Reference

Collaboration diagram for YTablePrivate:



### Public Member Functions

- **YTablePrivate** ([YTableHeader](#) \*header)

### Public Attributes

- [YTableHeader](#) \* **header**
- bool **keepSorting**
- bool **immediateMode**

### 5.147.1 Detailed Description

Definition at line 35 of file [YTable.cc](#).

The documentation for this struct was generated from the following file:

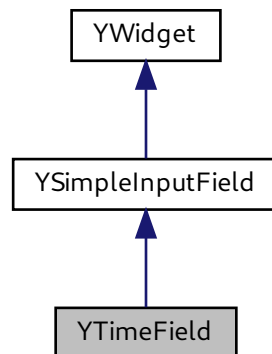
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YTable.cc

## 5.148 YTimeField Class Reference

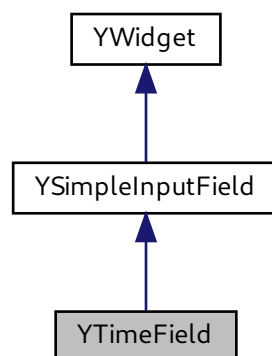
Input field for entering a time in "hh:mm:ss" format.

```
#include <YTimeField.h>
```

Inheritance diagram for YTimeField:



Collaboration diagram for YTimeField:



## Public Member Functions

- virtual [~YTimeField](#) ()  
*Destructor.*
- virtual const char \* [widgetClass](#) () const  
*Returns a descriptive name of this widget class for logging, debugging etc.*

## Protected Member Functions

- [YTimeField](#) (YWidget \*parent, const std::string &label)  
*Constructor.*

### 5.148.1 Detailed Description

Input field for entering a time in "hh:mm:ss" format.

Derived classes are required to implement: [value\(\)](#) [setValue\(\)](#) See [YSimpleInputField.h](#) for details.

Definition at line 41 of file [YTimeField.h](#).

The documentation for this class was generated from the following files:

- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YTimeField.h
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YTimeField.cc

## 5.149 YTimeFieldPrivate Struct Reference

### Public Attributes

- bool **dummy**

### 5.149.1 Detailed Description

Definition at line 34 of file [YTimeField.cc](#).

The documentation for this struct was generated from the following file:

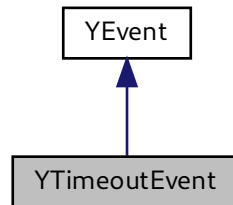
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YTimeField.cc

## 5.150 YTimeoutEvent Class Reference

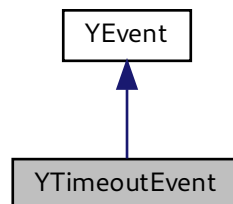
Event to be returned upon timeout (i.e.

```
#include <YEvent.h>
```

Inheritance diagram for YTimeoutEvent:



Collaboration diagram for YTimeoutEvent:



### Protected Member Functions

- virtual [~YTimeoutEvent](#) ()

*Protected destructor - events can only be deleted via [YDialog::deleteEvent\(\)](#).*

### Additional Inherited Members

#### 5.150.1 Detailed Description

Event to be returned upon timeout (i.e.

no event available in the specified timeout)

Definition at line [346](#) of file [YEvent.h](#).

## 5.150.2 Constructor & Destructor Documentation

### 5.150.2.1 ~YTimeoutEvent()

```
virtual YTimeoutEvent::~YTimeoutEvent ( ) [inline], [protected], [virtual]
```

Protected destructor - events can only be deleted via [YDialog::deleteEvent\(\)](#).

The associated dialog will take care of this event and delete it when appropriate.

Definition at line [358](#) of file [YEvent.h](#).

The documentation for this class was generated from the following file:

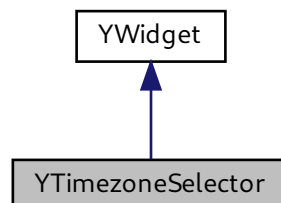
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YEvent.h](#)

## 5.151 YTimezoneSelector Class Reference

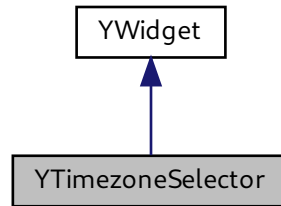
A fancy widget with a world map.

```
#include <YTimezoneSelector.h>
```

Inheritance diagram for YTimezoneSelector:



Collaboration diagram for YTimezoneSelector:



## Public Member Functions

- virtual `~YTimezoneSelector ()`  
*Destructor.*
- virtual const char \* `widgetClass ()` const  
*Return a descriptive name of this widget class for logging, debugging etc.*
- virtual bool `setProperty (const std::string &propertyName, const YPropertyValue &val)`  
*Set a property.*
- virtual `YPropertyValue getProperty (const std::string &propertyName)`  
*Get a property.*
- virtual const `YPropertySet & propertySet ()`  
*Return this class's property set.*
- virtual std::string `currentZone ()` const =0  
*subclasses have to implement this to return value*
- virtual void `setCurrentZone (const std::string &zone, bool zoom)=0`  
*subclasses have to implement this to set value*

## Protected Member Functions

- `YTimezoneSelector (YWidget *parent, const std::string &ixmap, const std::map< std::string, std::string > &time-zones)`  
*Constructor.*

### 5.151.1 Detailed Description

A fancy widget with a world map.

Definition at line 38 of file `YTimezoneSelector.h`.



## 5.151.2 Constructor & Destructor Documentation

### 5.151.2.1 YTimeZoneSelector()

```
YTimeZoneSelector::YTimeZoneSelector (
    YWidget * parent,
    const std::string & pixmap,
    const std::map< std::string, std::string > & timezones ) [protected]
```

Constructor.

This widget isn't doing much on it's own, but the UI may have some fancy use.

- pixmap should be a png or jpg of a world map with centered 0°0° and the timezones are a map between zone.tab entry and user visible string.

The widget is only displaying timezones/cities in that map

Definition at line 43 of file [YTimeZoneSelector.cc](#).

## 5.151.3 Member Function Documentation

### 5.151.3.1 getProperty()

```
YPropertyValue YTimeZoneSelector::getProperty (
    const std::string & propertyName ) [virtual]
```

Get a property.

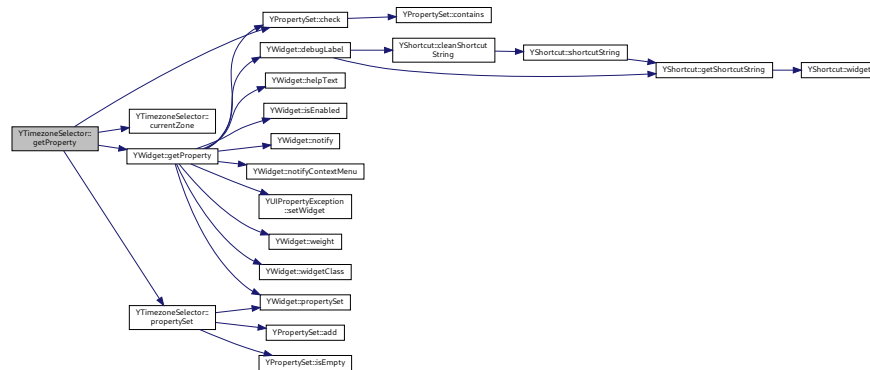
Reimplemented from [YWidget](#).

This method may throw YUIPropertyExceptions.

Reimplemented from [YWidget](#).

Definition at line 93 of file [YTimeZoneSelector.cc](#).

Here is the call graph for this function:



### 5.151.3.2 propertySet()

```
const YPropertySet & YTimezoneSelector::propertySet ( ) [virtual]
```

Return this class's property set.

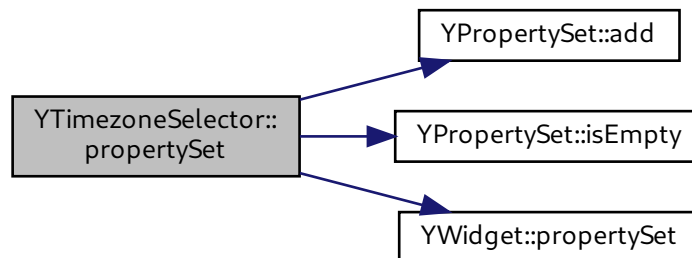
This also initializes the property upon the first call.

Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 58 of file [YTimezoneSelector.cc](#).

Here is the call graph for this function:



### 5.151.3.3 setProperty()

```
bool YTimezoneSelector::setProperty (
    const std::string & propertyName,
    const YPropertyValue & val ) [virtual]
```

Set a property.

Reimplemented from [YWidget](#).

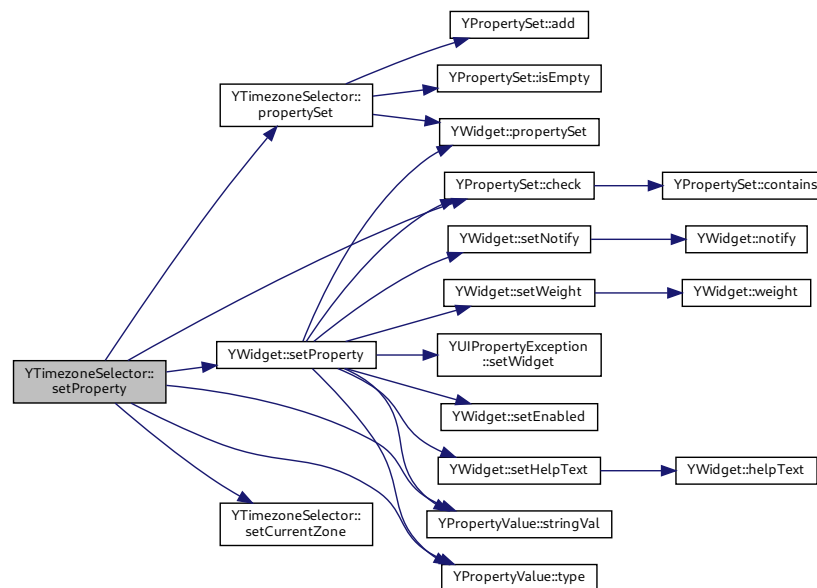
This function may throw YUIPropertyExceptions.

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from [YWidget](#).

Definition at line 74 of file [YTimezoneSelector.cc](#).

Here is the call graph for this function:



The documentation for this class was generated from the following files:

- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YTimezoneSelector.h`
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YTimezoneSelector.cc`

## 5.152 YTimezoneSelectorPrivate Class Reference

### 5.152.1 Detailed Description

Definition at line 35 of file [YTimezoneSelector.cc](#).

The documentation for this class was generated from the following file:

- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YTimezoneSelector.cc

## 5.153 YTransText Class Reference

Helper class for translated strings: Stores a message in the original (untranslated) version along with the translation into the current locale.

```
#include <YTransText.h>
```

### Public Member Functions

- [YTransText](#) (const std::string &[orig](#), const std::string &[translation](#))  
*Constructor with both original and translated message.*
- [YTransText](#) (const std::string &[orig](#))  
*Constructor that automatically translates the original message.*
- [YTransText](#) (const [YTransText](#) &src)  
*Copy constructor.*
- [YTransText](#) & [operator=](#) (const [YTransText](#) &src)  
*Assignment operator.*
- const std::string & [orig](#) () const  
*Return the original message.*
- const std::string & [translation](#) () const  
*Return the translation.*
- const std::string & [trans](#) () const  
*Return the translation.*
- void [setOrig](#) (const std::string &newOrig)  
*Set the original message.*
- void [setTranslation](#) (const std::string &newTrans)  
*Set the translation.*
- bool [operator<](#) (const [YTransText](#) &other) const  
*operator< : Compares translations.*
- bool [operator>](#) (const [YTransText](#) &other) const  
*operator> : Compares translations.*
- bool [operator==](#) (const [YTransText](#) &other) const  
*operator== : Compares translations.*

### 5.153.1 Detailed Description

Helper class for translated strings: Stores a message in the original (untranslated) version along with the translation into the current locale.

Definition at line 36 of file [YTransText.h](#).

### 5.153.2 Member Function Documentation

#### 5.153.2.1 setOrig()

```
void YTransText::setOrig (  
    const std::string & newOrig )    [inline]
```

Set the original message.

Does not touch the translation, so make sure you change both if you want to keep them synchronized!

Definition at line 95 of file [YTransText.h](#).

#### 5.153.2.2 trans()

```
const std::string& YTransText::trans ( ) const    [inline]
```

Return the translation.

( alias, just as a shortcut )

Definition at line 89 of file [YTransText.h](#).

The documentation for this class was generated from the following file:

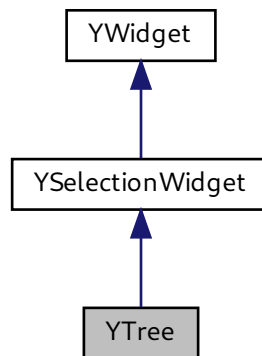
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YTransText.h](#)

## 5.154 YTree Class Reference

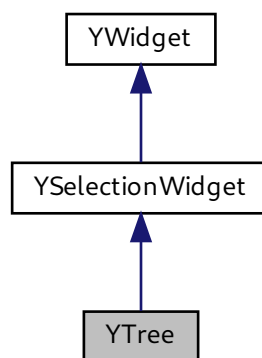
Tree: List box that displays a (scrollable) list of hierarchical items from which the user can select exactly one.

```
#include <YTree.h>
```

Inheritance diagram for YTree:



Collaboration diagram for YTree:



## Public Member Functions

- virtual `~YTree ()`  
*Destructor.*
- virtual const char \* `widgetClass ()` const  
*Returns a descriptive name of this widget class for logging, debugging etc.*
- virtual void `rebuildTree ()`=0  
*Rebuild the displayed tree from the internally stored YTreeItems.*
- virtual void `addItems` (const `YItemCollection` &itemCollection)  
*Add multiple items.*
- bool `immediateMode ()` const  
*Deliver even more events than with `notify()` set.*
- void `setImmediateMode` (bool on=true)  
*Set `immediateMode()` on or off.*
- virtual bool `setProperty` (const std::string &propertyName, const `YPropertyValue` &val)  
*Set a property.*
- virtual `YPropertyValue` `getProperty` (const std::string &propertyName)  
*Get a property.*
- virtual const `YPropertySet` & `propertySet ()`  
*Return this class's property set.*
- const char \* `userInputProperty ()`  
*The name of the widget property that will return user input.*
- bool `hasMultiSelection ()` const  
*Return 'true' if the user can select multiple items at the same time.*
- virtual `YTreeItem` \* `currentItem ()`=0  
*Return the the item that currently has the keyboard focus or 0 if no item currently has the keyboard focus.*
- `YTreeItem` \* `findItem` (std::vector< std::string > &path) const  
*Return item in the tree which matches path of labels or nullptr in case no item with such label was found.*
- virtual void `activate ()`=0  
*Activate the item selected in the tree.*

## Protected Member Functions

- `YTree` (`YWidget` \*parent, const std::string &label, bool multiSelection, bool `recursiveSelection`)  
*Constructor.*
- `YTreeItem` \* `findItem` (std::vector< std::string >::iterator path\_begin, std::vector< std::string >::iterator path\_end, `YItemConstIterator` begin, `YItemConstIterator` end) const  
*Recursively looks for the first item in the tree of the menu items using depth first search.*

### 5.154.1 Detailed Description

Tree: List box that displays a (scrollable) list of hierarchical items from which the user can select exactly one.

Each item has a label text and an optional icon (\*).

This is very similar to `SelectionBox`, but each item can have subitems that can be open (expanded) or closed (collapsed).

The tree widget also has a caption label that is displayed above the tree. The hotkey displayed in that caption label will move the keyboard focus into the tree item list.

(\*) Not all UIs (in particular not text-based UIs) support displaying icons, so an icon should never be an exclusive means to display any kind of information.

'multiSelection' indicates whether or not the user can select multiple items at the same time. This can only be set in the constructor.

Definition at line 56 of file [YTree.h](#).

### 5.154.2 Member Function Documentation

#### 5.154.2.1 activate()

```
virtual void YTree::activate ( ) [pure virtual]
```

Activate the item selected in the tree.

Can be used in tests to simulate user input.

Derived classes are required to implement this.

#### 5.154.2.2 addItem()

```
void YTree::addItem (
    const YItemCollection & itemCollection ) [virtual]
```

Add multiple items.

For some UIs, this can be more efficient than calling [addItem\(\)](#) multiple times. This function also automatically calls [rebuildTree\(\)](#) at the end.

Derived classes can overwrite this function, but they should call this base class function at the end of the new implementation.

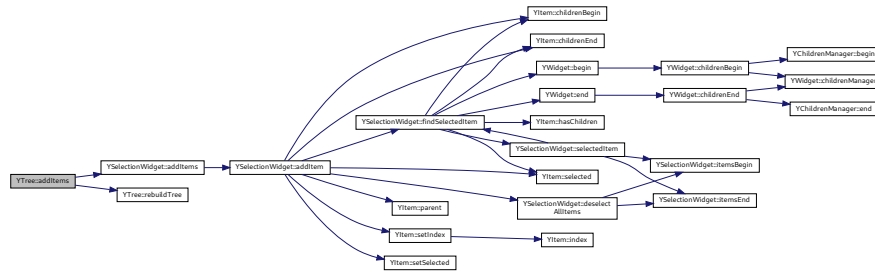
Reimplemented from [YSelectionWidget](#).

Reimplemented from [YSelectionWidget](#).



Definition at line 84 of file [YTree.cc](#).

Here is the call graph for this function:



### 5.154.2.3 currentItem()

```
virtual YTreeItem* YTree::currentItem ( ) [pure virtual]
```

Return the the item that currently has the keyboard focus or 0 if no item currently has the keyboard focus.

Notice that for a MultiSelectionBox the current item is not necessarily selected, i.e., its check box may or may not be checked.

Derived classes are required to implement this function.

### 5.154.2.4 findItem() [1/2]

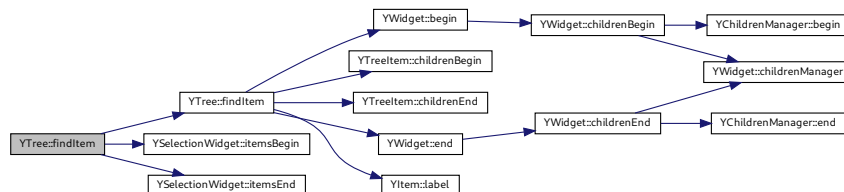
```
YTreeItem * YTree::findItem (
    std::vector< std::string > & path ) const
```

Return item in the tree which matches path of labels or nullptr in case no item with such label was found.

Accepts vector of strings which denote path to the node.

Definition at line 175 of file [YTree.cc](#).

Here is the call graph for this function:



### 5.154.2.5 findItem() [2/2]

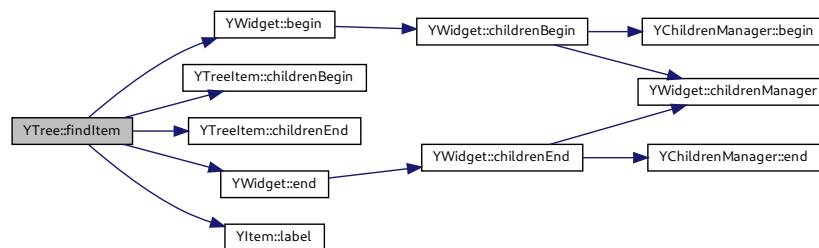
```
YTreeItem * YTree::findItem (
    std::vector< std::string >::iterator path_begin,
    std::vector< std::string >::iterator path_end,
    YItemConstIterator begin,
    YItemConstIterator end ) const [protected]
```

Recursively looks for the first item in the tree of the menu items using depth first search.

Return nullptr if item which matches full path is not found.

Definition at line 182 of file [YTree.cc](#).

Here is the call graph for this function:



### 5.154.2.6 getProperty()

```
YPropertyValue YTree::getProperty (
    const std::string & propertyName ) [virtual]
```

Get a property.

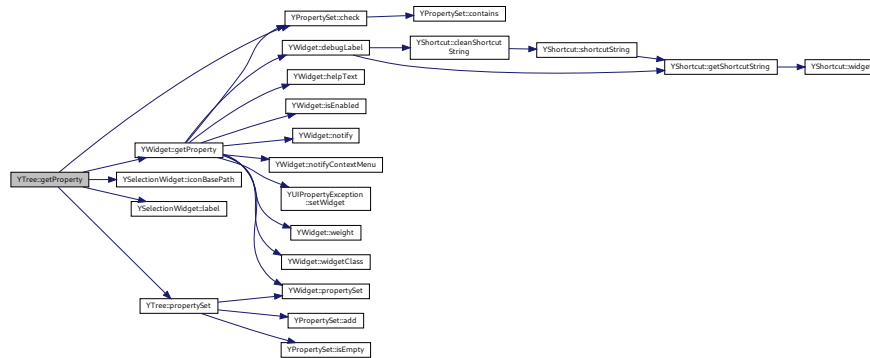
Reimplemented from [YWidget](#).

This method may throw `YUIPropertyExceptions`.

Reimplemented from [YWidget](#).

Definition at line 149 of file [YTree.cc](#).

Here is the call graph for this function:



### 5.154.2.7 immediateMode()

```
bool YTree::immediateMode ( ) const
```

Deliver even more events than with [notify\(\)](#) set.

For [YTree](#), this is relevant mostly for the Ncurses UI:

In graphical UIs like the Qt UI, the user can use the mouse to select an item in a tree. With [notify\(\)](#) set, this will send an event right away (i.e., it will make `UserInput` and related return, while normally it would only return when the user clicks a `PushButton`).

In the Ncurses UI, there is no mouse, so the user has to use the cursor keys to move to the item he wants to select. In [immediateMode\(\)](#), every cursor key press will make the tree send an event. Without [immediateMode\(\)](#), the `NCTree` will wait until the user hits the [Return] key until an event is sent. Depending on what the application does upon each selection box event, [immediateMode\(\)](#) might make the application less responsive.

Definition at line 67 of file [YTree.cc](#).

### 5.154.2.8 propertySet()

```
const YPropertySet & YTree::propertySet ( ) [virtual]
```

Return this class's property set.

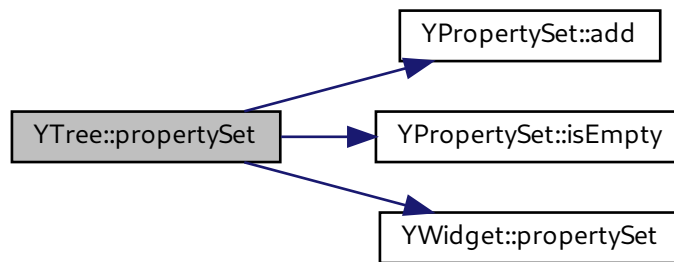
This also initializes the property upon the first call.

Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 92 of file [YTree.cc](#).

Here is the call graph for this function:



#### 5.154.2.9 `rebuildTree()`

```
virtual void YTree::rebuildTree ( ) [pure virtual]
```

Rebuild the displayed tree from the internally stored `YTreeItems`.

The application should call this (once) after all items have been added with [addItem\(\)](#). [YTree::addItem\(\)](#) calls this automatically.

Derived classes are required to implement this.

#### 5.154.2.10 `setProperty()`

```
bool YTree::setProperty (
    const std::string & propertyName,
    const YPropertyValue & val ) [virtual]
```

Set a property.

Reimplemented from [YWidget](#).

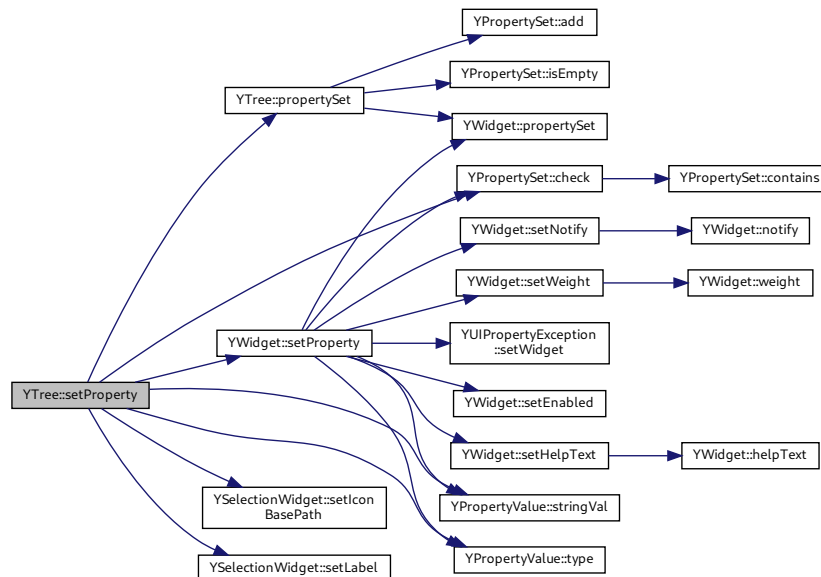
This function may throw `YUIPropertyExceptions`.

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented from [YWidget](#).

Definition at line 126 of file [YTree.cc](#).

Here is the call graph for this function:



#### 5.154.2.11 userInputProperty()

```
const char* YTree::userInputProperty ( ) [inline], [virtual]
```

The name of the widget property that will return user input.

Inherited from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 165 of file [YTree.h](#).

The documentation for this class was generated from the following files:

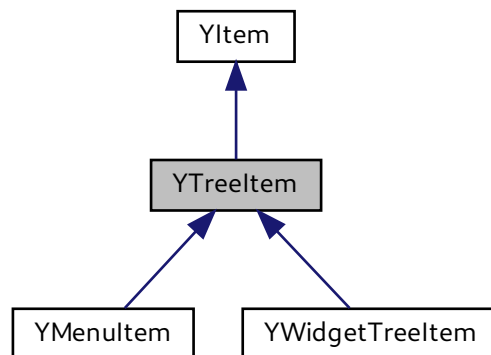
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YTree.h`
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YTree.cc`

## 5.155 YTreeItem Class Reference

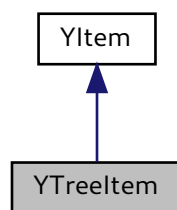
Item class for tree items.

```
#include <YTreeItem.h>
```

Inheritance diagram for YTreeItem:



Collaboration diagram for YTreeItem:



### Public Member Functions

- [YTreeItem](#) (const std::string &label, bool isOpen=false)  
*Constructors for toplevel items.*
- **YTreeItem** (const std::string &label, const std::string &iconName, bool isOpen=false)
- [YTreeItem](#) ([YTreeItem](#) \*parent, const std::string &label, bool isOpen=false)

*Constructors for items that have a parent item.*

- **YTreeItem** (YTreeItem \*parent, const std::string &label, const std::string &iconName, bool isOpen=false)
- virtual ~YTreeItem ()

*Destructor.*

- virtual bool hasChildren () const

*Return 'true' if this item has any child items.*

- virtual YItemIterator childrenBegin ()

*Return an iterator that points to the first child item of this item.*

- virtual YItemConstIterator childrenBegin () const
- virtual YItemIterator childrenEnd ()

*Return an iterator that points after the last child item of this item.*

- virtual YItemConstIterator childrenEnd () const
- virtual void addChild (YItem \*item\_disown)

*Add a child item to this item.*

- virtual void deleteChildren ()

*Delete all child items.*

- bool isOpen () const

*Return 'true' if this tree item should be displayed open (with its children visible) by default.*

- void setOpen (bool open)

*Change the 'isOpen' flag.*

- virtual YTreeItem \* parent () const

*Returns this item's parent item or 0 if it is a toplevel item.*

## 5.155.1 Detailed Description

Item class for tree items.

This class implements children management.

Definition at line 35 of file YTreeItem.h.

## 5.155.2 Constructor & Destructor Documentation

### 5.155.2.1 YTreeItem()

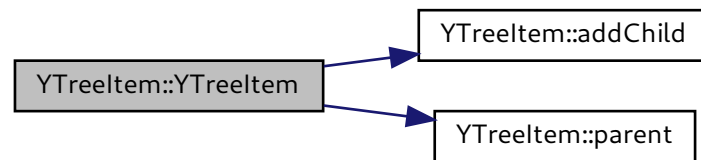
```
YTreeItem::YTreeItem (
    YTreeItem * parent,
    const std::string & label,
    bool isOpen = false )
```

Constructors for items that have a parent item.

They will automatically register this item with the parent item. The parent assumes ownership of this item and will delete it in its (the parent's) destructor.

Definition at line 49 of file [YTreeItem.cc](#).

Here is the call graph for this function:



#### 5.155.2.2 `~YTreeItem()`

```
YTreeItem::~~YTreeItem ( ) [virtual]
```

Destructor.

This will delete all children.

Definition at line 74 of file [YTreeItem.cc](#).

Here is the call graph for this function:



### 5.155.3 Member Function Documentation



### 5.155.3.1 addChild()

```
void YTreeItem::addChild (
    YItem * item_disown ) [virtual]
```

Add a child item to this item.

Note that the constructors that accept a parent pointer will automatically add themselves to their parent, so applications will normally not have to call this function.

Definition at line 80 of file [YTreeItem.cc](#).

### 5.155.3.2 childrenBegin()

```
virtual YItemIterator YTreeItem::childrenBegin ( ) [inline], [virtual]
```

Return an iterator that points to the first child item of this item.

Reimplemented from [YItem](#).

Reimplemented from [YItem](#).

Definition at line 83 of file [YTreeItem.h](#).

### 5.155.3.3 childrenEnd()

```
virtual YItemIterator YTreeItem::childrenEnd ( ) [inline], [virtual]
```

Return an iterator that points after the last child item of this item.

Reimplemented from [YItem](#).

Reimplemented from [YItem](#).

Definition at line 91 of file [YTreeItem.h](#).

### 5.155.3.4 hasChildren()

```
virtual bool YTreeItem::hasChildren ( ) const [inline], [virtual]
```

Return 'true' if this item has any child items.

Reimplemented from [YItem](#).

Reimplemented from [YItem](#).

Definition at line 76 of file [YTreeItem.h](#).

### 5.155.3.5 isOpen()

```
bool YTreeWidgetItem::isOpen ( ) const
```

Return 'true' if this tree item should be displayed open (with its children visible) by default.

Notice that this will always return 'false' for tree items without children.

Definition at line 101 of file [YTreeWidgetItem.cc](#).

Here is the call graph for this function:



### 5.155.3.6 parent()

```
virtual YTreeWidgetItem\* YTreeWidgetItem::parent ( ) const [inline], [virtual]
```

Returns this item's parent item or 0 if it is a toplevel item.

Reimplemented from [YItem](#).

Reimplemented from [YItem](#).

Reimplemented in [YMenuItem](#).

Definition at line 127 of file [YTreeWidgetItem.h](#).

The documentation for this class was generated from the following files:

- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YTreeWidgetItem.h](#)
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YTreeWidgetItem.cc](#)

## 5.156 YTreePrivate Struct Reference

### Public Attributes

- bool **immediateMode**

### 5.156.1 Detailed Description

Definition at line 37 of file [YTree.cc](#).

The documentation for this struct was generated from the following file:

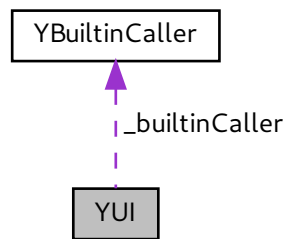
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YTree.cc](#)

## 5.157 YUI Class Reference

Abstract base class of a libYUI user interface.

```
#include <YUI.h>
```

Collaboration diagram for YUI:



### Public Member Functions

- virtual `~YUI ()`  
*Destructor.*
- void `shutdownThreads ()`  
*Shut down multithreading.*
- virtual void `blockEvents (bool block=true)`  
*Block (or unblock) events.*
- void `unblockEvents ()`  
*Unblock events previously blocked.*
- virtual bool `eventsBlocked () const`  
*Returns 'true' if events are currently blocked.*
- virtual void `deleteNotify (YWidget *widget)`  
*Notification that a widget is being deleted.*
- void `topmostConstructorHasFinished ()`

*Must be called after the constructor of the Qt/NCurses ui is ready.*

- bool `runningWithThreads` () const  
*Running with threads?*
- void `uiThreadMainLoop` ()  
*This method implements the UI thread in case it is existing.*
- `YBuiltinCaller` \* `builtinCaller` () const  
*Return the transparent inter-thread communication.*
- void `setBuiltinCaller` (`YBuiltinCaller` \*caller)  
*Set the transparent inter-thread communication.*
- virtual `YEvent` \* `runPkgSelection` (`YWidget` \*packageSelector)=0  
*UI-specific runPkgSelection method.*
- `YWidget` \* `sendWidgetID` (const std::string &id)  
*Send a widget ID.*

## Static Public Member Functions

- static `YUI` \* `ui` ()  
*Access the global UI.*
- static `YWidgetFactory` \* `widgetFactory` ()  
*Return the widget factory that provides all the createXY() methods for standard (mandatory, i.e.*
- static `YOptionalWidgetFactory` \* `optionalWidgetFactory` ()  
*Return the widget factory that provides all the createXY() methods for optional ("special") widgets and the corresponding hasXYWidget() methods.*
- static `YApplication` \* `app` ()  
*Return the global YApplication object.*
- static `YApplication` \* `application` ()  
*Aliases for YUI::app()*
- static `YApplication` \* `yApp` ()
- static void `ensureUICreated` ()  
*Make sure there is a UI (with a UI plug-in) created.*

## Protected Member Functions

- `YUI` (bool withThreads)  
*Constructor.*
- virtual `YWidgetFactory` \* `createWidgetFactory` ()=0  
*Create the widget factory that provides all the createXY() methods for standard (mandatory, i.e.*
- virtual `YOptionalWidgetFactory` \* `createOptionalWidgetFactory` ()=0  
*Create the widget factory that provides all the createXY() methods for optional ("special") widgets and the corresponding hasXYWidget() methods.*
- virtual `YApplication` \* `createApplication` ()=0  
*Create the YApplication object that provides global methods.*
- virtual void `idleLoop` (int fd\_ycp)=0  
*This virtual method is called when threads are activated in case the execution control is currently on the side of the module.*
- void `terminateUIThread` ()  
*Tells the ui thread that it should terminate and waits until it does so.*

- void [createUIThread](#) ()  
*Creates and launches the ui thread.*
- virtual void [uiThreadDestructor](#) ()  
*Destructor for the UI thread.*
- void [signalUIThread](#) ()  
*Signals the ui thread by sending one byte through the pipe to it.*
- bool [waitForUIThread](#) ()  
*Waits for the ui thread to send one byte through the pipe to the ycp thread and reads this byte from the pipe.*
- void [signalYCPThread](#) ()  
*Signals the ycp thread by sending one byte through the pipe to it.*
- bool [waitForYCPThread](#) ()  
*Waits for the ycp thread to send one byte through the pipe to the ycp thread and reads this byte from the pipe.*
- void [setButtonOrderFromEnvironment](#) ()  
*Set the button order (in [YButtonBox](#) widgets) from environment variables:*

## Protected Attributes

- bool [\\_withThreads](#)  
*true if a seperate UI thread is created*
- pthread\_t [\\_uiThread](#)  
*Handle to the ui thread.*
- [YBuiltinCaller](#) \* [\\_builtinCaller](#)  
*Inter-thread communication between the YCP thread and the UI thread: The YCP thread supplies data here and signals the UI thread, the UI thread picks up the data, executes the function, puts the result here and signals the YCP thread that waits until the result is available.*
- int [pipe\\_to\\_ui](#) [2]  
*Used to synchronize data transfer with the ui thread.*
- int [pipe\\_from\\_ui](#) [2]  
*Used to synchronize data transfer with the ui thread.*
- bool [\\_terminate\\_ui\\_thread](#)  
*This is a flag that signals the ui thread that it should terminate.*
- bool [\\_eventsBlocked](#)  
*Flag that keeps track of blocked events.*

## Friends

- class **YUIFunction**
- class **YUILoader**
- void \* **start\_ui\_thread** (void \*ui\_int)

### 5.157.1 Detailed Description

Abstract base class of a libYUI user interface.

Definition at line 48 of file [YUI.h](#).

## 5.157.2 Member Function Documentation

### 5.157.2.1 app()

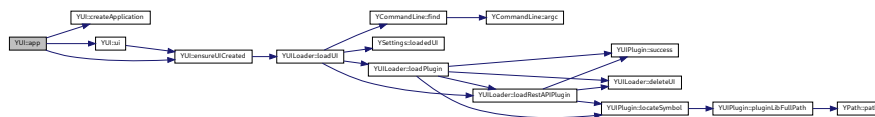
```
YApplication * YUI::app ( ) [static]
```

Return the global [YApplication](#) object.

This will create the [YApplication](#) upon the first call and return a pointer to the one and only (singleton) [YApplication](#) upon each subsequent call. This may throw exceptions if the [YApplication](#) cannot be created.

Definition at line 162 of file [YUI.cc](#).

Here is the call graph for this function:



### 5.157.2.2 blockEvents()

```
virtual void YUI::blockEvents (
    bool block = true ) [inline], [virtual]
```

Block (or unblock) events.

If events are blocked, any event sent should be ignored until events are unblocked again.

This default implementation keeps track of a simple internal flag that can be queried with [eventsBlocked\(\)](#), so if you reimplement [blockEvents\(\)](#), be sure to reimplement [eventsBlocked\(\)](#) as well.

Definition at line 161 of file [YUI.h](#).

### 5.157.2.3 builtinCaller()

```
YBuiltinCaller* YUI::builtinCaller ( ) const [inline]
```

Return the transparent inter-thread communication.

This will return 0 until set from the outside.

Definition at line 212 of file [YUI.h](#).

#### 5.157.2.4 createApplication()

```
virtual YApplication* YUI::createApplication ( ) [protected], [pure virtual]
```

Create the [YApplication](#) object that provides global methods.

Derived classes are required to implement this.

#### 5.157.2.5 createOptionalWidgetFactory()

```
virtual YOptionalWidgetFactory* YUI::createOptionalWidgetFactory ( ) [protected], [pure virtual]
```

Create the widget factory that provides all the createXY() methods for optional ("special") widgets and the corresponding hasXYWidget() methods.

Derived classes are required to implement this.

#### 5.157.2.6 createWidgetFactory()

```
virtual YWidgetFactory* YUI::createWidgetFactory ( ) [protected], [pure virtual]
```

Create the widget factory that provides all the createXY() methods for standard (mandatory, i.e. non-optional) widgets.

Derived classes are required to implement this.

#### 5.157.2.7 deleteNotify()

```
virtual void YUI::deleteNotify (
    YWidget * widget ) [inline], [virtual]
```

Notification that a widget is being deleted.

This is called from the [YWidget](#) destructor.

Derived classes can implement this for any clean-up actions such as deleting any events that might be pending for that widget.

Definition at line 185 of file [YUI.h](#).

### 5.157.2.8 ensureUICreated()

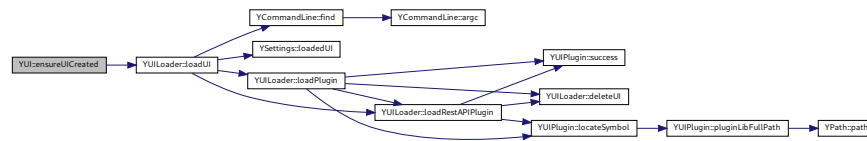
```
void YUI::ensureUICreated ( ) [static]
```

Make sure there is a UI (with a UI plug-in) created.

If there is none yet, this will use all-default parameters to load a UI plug-in and create a UI (without threads).

Definition at line 176 of file [YUI.cc](#).

Here is the call graph for this function:



### 5.157.2.9 eventsBlocked()

```
virtual bool YUI::eventsBlocked ( ) const [inline], [virtual]
```

Returns 'true' if events are currently blocked.

Reimplement this if you reimplement [blockEvents\(\)](#).

Definition at line 176 of file [YUI.h](#).

### 5.157.2.10 idleLoop()

```
virtual void YUI::idleLoop (
    int fd_ycp ) [protected], [pure virtual]
```

This virtual method is called when threads are activated in case the execution control is currently on the side of the module.

This means that no `UserInput()` or `PollInput()` is pending. The module just does some work. The UI <-> module protocol is in the "UI waits for the next command" state. The UI can override this method when it wants to react to user input or other external events such as repaint requests from the X server.

'fd\_ycp' file descriptor that should be used to determine when to leave the idle loop. As soon as it is readable, the loop must be left. In order to avoid polling you can combine it with other ui-specific fds and do a common `select()` call.



### 5.157.2.11 optionalWidgetFactory()

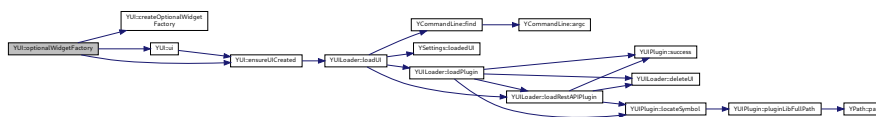
```
YOptionalWidgetFactory * YUI::optionalWidgetFactory ( ) [static]
```

Return the widget factory that provides all the createXY() methods for optional ("special") widgets and the corresponding hasXYWidget() methods.

This will create the factory upon the first call and return a pointer to the one and only (singleton) factory upon each subsequent call. This may throw exceptions if the factory cannot be created.

Definition at line 147 of file [YUI.cc](#).

Here is the call graph for this function:



### 5.157.2.12 runPkgSelection()

```
virtual YEvent* YUI::runPkgSelection (
    YWidget * packageSelector ) [pure virtual]
```

UI-specific runPkgSelection method.

Derived classes are required to implement this.

The packageSelector's dialog will take care of the event and delete it when appropriate. The returned pointer is valid until the next call to YDialog::userInput(), YDialog::pollInput(), or [YUI::runPkgSelection\(\)](#) or until the dialog with the packageSelector is destroyed.

### 5.157.2.13 sendWidgetID()

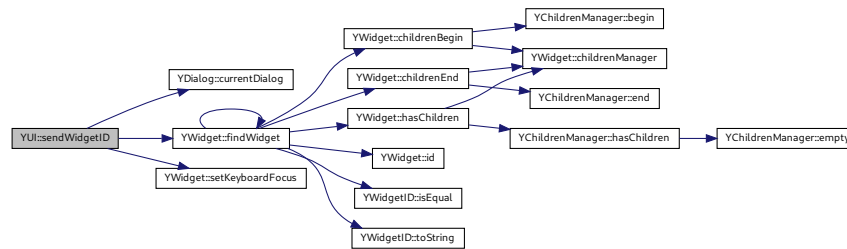
```
YWidget * YUI::sendWidgetID (
    const std::string & id )
```

Send a widget ID.

This implementation simply sets the keyboard focus to that widget. If there is no widget with that ID, this will throw a [YUIWidgetNotFoundException](#). This function returns the widget that was found in case the caller wants to do more with it than just set the keyboard focus to it.

Definition at line 487 of file [YUI.cc](#).

Here is the call graph for this function:



#### 5.157.2.14 setBuiltinCaller()

```
void YUI::setBuiltinCaller (
    YBuiltinCaller * caller ) [inline]
```

Set the transparent inter-thread communication.

Built-ins are only really called if there is a valid [YBuiltinCaller](#) set.

Definition at line 218 of file [YUI.h](#).

#### 5.157.2.15 setButtonOrderFromEnvironment()

```
void YUI::setButtonOrderFromEnvironment ( ) [protected]
```

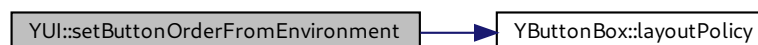
Set the button order (in [YButtonBox](#) widgets) from environment variables:

```
$Y2_BUTTON_ORDER="KDE"
$Y2_BUTTON_ORDER="Gnome"
```

(all case insensitive)

Definition at line 392 of file [YUI.cc](#).

Here is the call graph for this function:



### 5.157.2.16 shutdownThreads()

```
void YUI::shutdownThreads ( )
```

Shut down multithreading.

This needs to be called before the destructor if the UI was created with threads. If the UI was created without threads, this does nothing.

Definition at line 265 of file [YUI.cc](#).

Here is the call graph for this function:



### 5.157.2.17 topmostConstructorHasFinished()

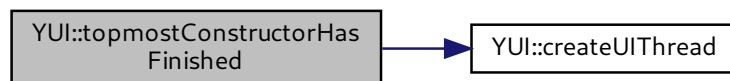
```
void YUI::topmostConstructorHasFinished ( )
```

Must be called after the constructor of the Qt/NCurses ui is ready.

Starts the ui thread.

Definition at line 188 of file [YUI.cc](#).

Here is the call graph for this function:



### 5.157.2.18 uiThreadDestructor()

```
void YUI::uiThreadDestructor ( ) [protected], [virtual]
```

Destructor for the UI thread.

This will be called as the last thing the UI thread does.

Derived classes can overwrite this. In most cases it makes sense to call this base class method in the new implementation.

Definition at line 117 of file [YUI.cc](#).

Here is the call graph for this function:



### 5.157.2.19 uiThreadMainLoop()

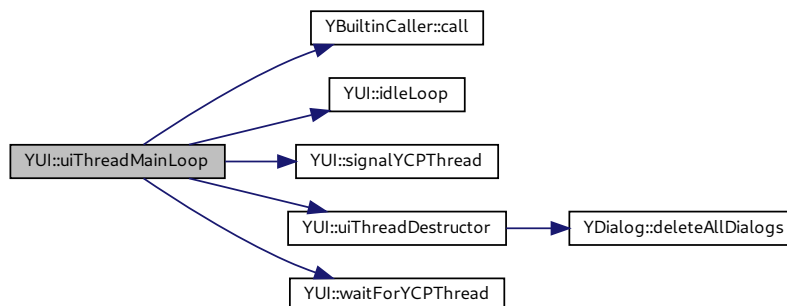
```
void YUI::uiThreadMainLoop ( )
```

This method implements the UI thread in case it is existing.

The loop consists of calling `idleLoop`, getting the next command from the `YCPUComponent`, evaluating it, which possibly involves calling `userInput()` or `pollInput()` and writes the answer back to the other thread where the request came from.

Definition at line 359 of file [YUI.cc](#).

Here is the call graph for this function:





### 5.157.3 Member Data Documentation

#### 5.157.3.1 `_eventsBlocked`

```
bool YUI::_eventsBlocked [protected]
```

Flag that keeps track of blocked events.

Never query this directly, use [eventsBlocked\(\)](#) instead.

Definition at line [367](#) of file [YUI.h](#).

#### 5.157.3.2 `_terminate_ui_thread`

```
bool YUI::_terminate_ui_thread [protected]
```

This is a flag that signals the ui thread that it should terminate.

This is done by setting the flag to true. The ui thread replies by setting the flag back to false directly after terminating itself.

Definition at line [361](#) of file [YUI.h](#).

#### 5.157.3.3 `pipe_from_ui`

```
int YUI::pipe_from_ui[2] [protected]
```

Used to synchronize data transfer with the ui thread.

It stores a pair of file descriptors of a pipe. For each YCP value we get from the ui thread, we read one arbitrary byte from here.

Definition at line [353](#) of file [YUI.h](#).

#### 5.157.3.4 pipe\_to\_ui

```
int YUI::pipe_to_ui[2] [protected]
```

Used to synchronize data transfer with the ui thread.

It stores a pair of file descriptors of a pipe. For each YCP value we send to the ui thread, we write one arbitrary byte here.

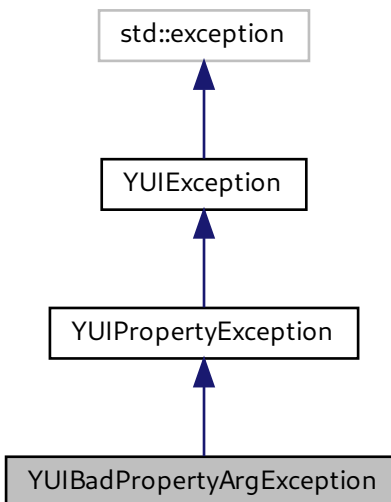
Definition at line 346 of file [YUI.h](#).

The documentation for this class was generated from the following files:

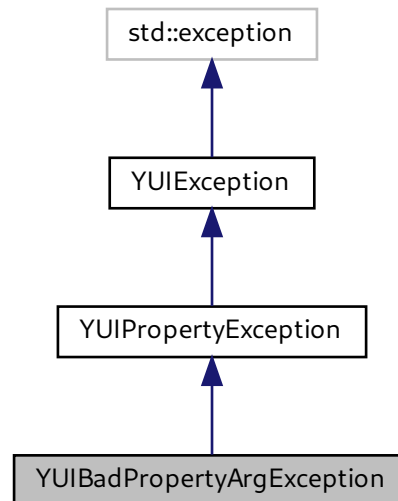
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YUI.h](#)
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YUI.cc](#)

## 5.158 YUIBadPropertyArgException Class Reference

Inheritance diagram for YUIBadPropertyArgException:



Collaboration diagram for YUIBadPropertyArgException:



## Public Member Functions

- **YUIBadPropertyArgException** (const [YProperty](#) &property, [YWidget](#) \*widget, const std::string &message="")

## Protected Member Functions

- virtual std::ostream & [dumpOn](#) (std::ostream &str) const  
*Write proper error message with all relevant data.*

## Additional Inherited Members

### 5.158.1 Detailed Description

Definition at line 635 of file [YUIException.h](#).

### 5.158.2 Member Function Documentation



### 5.158.2.1 dumpOn()

```
ostream & YUIBadPropertyArgException::dumpOn (
    std::ostream & str ) const [protected], [virtual]
```

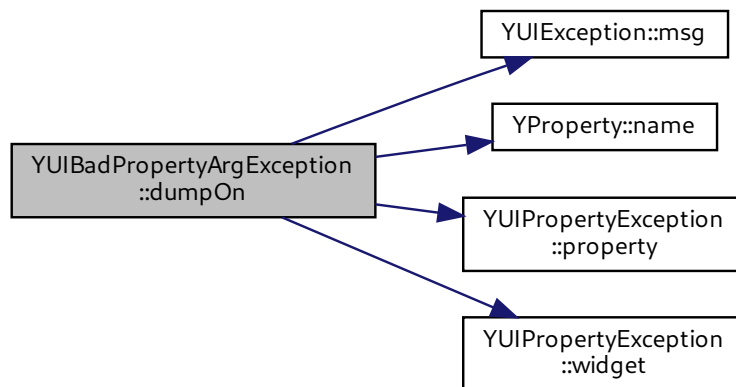
Write proper error message with all relevant data.

Reimplemented from [YUIException](#).

Implements [YUIPropertyException](#).

Definition at line 197 of file [YUIException.cc](#).

Here is the call graph for this function:



The documentation for this class was generated from the following files:

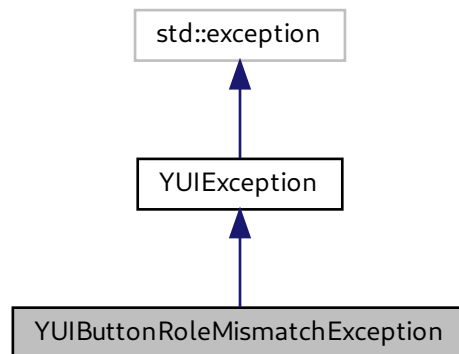
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YUIException.h`
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YUIException.cc`

## 5.159 YUIButtonRoleMismatchException Class Reference

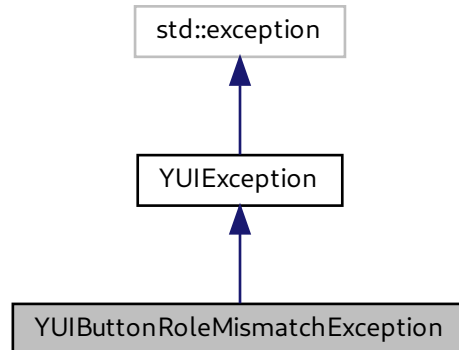
Exception class for "wrong button roles in YButtonBox".

```
#include <YUIException.h>
```

Inheritance diagram for `YUIButtonRoleMismatchException`:



Collaboration diagram for `YUIButtonRoleMismatchException`:



## Public Member Functions

- `YUIButtonRoleMismatchException` (const std::string &[msg](#))

## Additional Inherited Members

### 5.159.1 Detailed Description

Exception class for "wrong button roles in `YButtonBox`".

Definition at line 905 of file [YUIException.h](#).

The documentation for this class was generated from the following file:

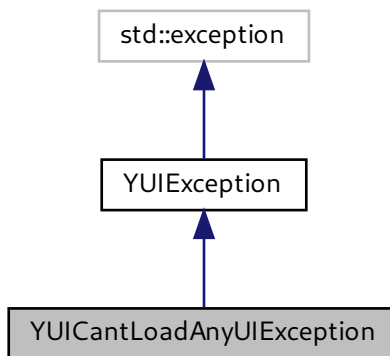
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YUIException.h

## 5.160 YUICantLoadAnyUIException Class Reference

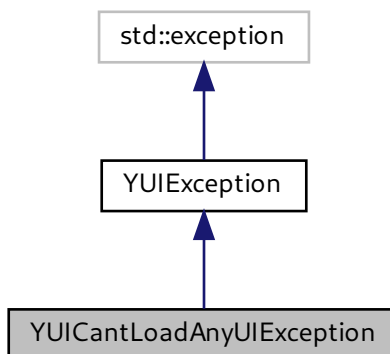
Exception class for UI plugin load failure.

```
#include <YUIException.h>
```

Inheritance diagram for YUICantLoadAnyUIException:



Collaboration diagram for YUICantLoadAnyUIException:



## Additional Inherited Members

### 5.160.1 Detailed Description

Exception class for UI plugin load failure.

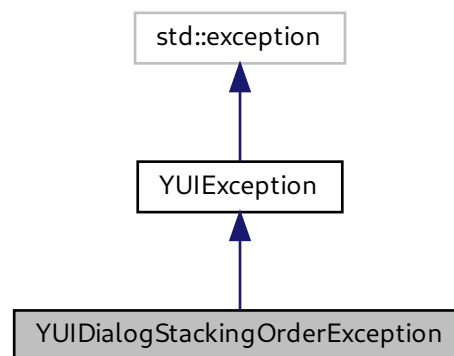
Definition at line 890 of file [YUIException.h](#).

The documentation for this class was generated from the following file:

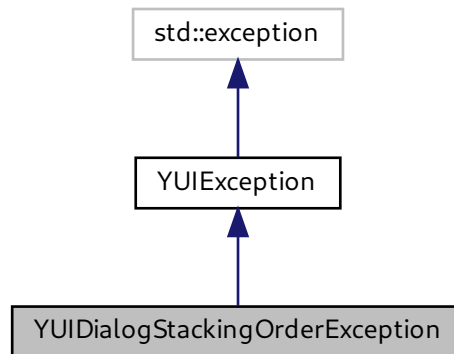
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YUIException.h`

## 5.161 YUIDialogStackingOrderException Class Reference

Inheritance diagram for YUIDialogStackingOrderException:



Collaboration diagram for YUIDialogStackingOrderException:



## Additional Inherited Members

### 5.161.1 Detailed Description

Definition at line [479](#) of file [YUIException.h](#).

The documentation for this class was generated from the following file:

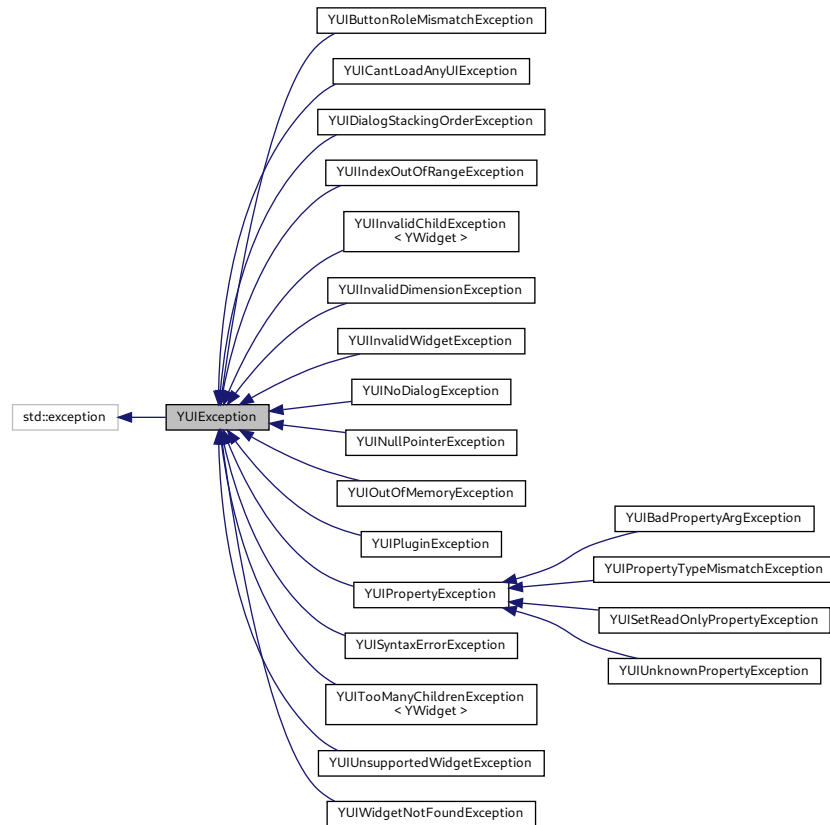
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YUIException.h](#)

## 5.162 YUIException Class Reference

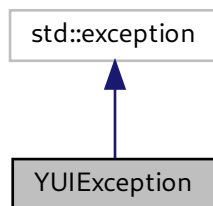
Base class for UI Exceptions.

```
#include <YUIException.h>
```

Inheritance diagram for YUIException:



Collaboration diagram for YUIException:



## Public Member Functions

- [YUIException](#) ()

*Default constructor.*

- [YUIException](#) (const std::string &msg\_r)

*Constructor taking a message.*

- virtual [~YUIException](#) () throw ()

*Destructor.*

- const [YCodeLocation](#) & [where](#) () const

*Return [YCodeLocation](#).*

- void [relocate](#) (const [YCodeLocation](#) &newLocation) const

*Exchange location on rethrow.*

- const std::string & [msg](#) () const

*Return the message string provided to the constructor.*

- void [setMsg](#) (const std::string &msg)

*Set a new message string.*

- std::string [asString](#) () const

*Error message provided by dumpOn as string.*

- virtual const char \* [what](#) () const throw ()

*Return message string.*

## Static Public Member Functions

- static std::string [strErrno](#) (int errno\_r)

*Make a string from errno\_r.*

- static std::string [strErrno](#) (int errno\_r, const std::string &msg)

*Make a string from errno\_r and msg\_r.*

- static void [log](#) (const [YUIException](#) &exception, const [YCodeLocation](#) &location, const char \*const prefix)

*Drop a log line on throw, catch or rethrow.*

## Protected Member Functions

- virtual std::ostream & [dumpOn](#) (std::ostream &str) const

*Overload this to print a proper error message.*

## Friends

- std::ostream & [operator<<](#) (std::ostream &str, const [YUIException](#) &obj)

*[YUIException](#) stream output.*

### 5.162.1 Detailed Description

Base class for UI Exceptions.

Exception offers to store a message string passed to the constructor. Derived classes may provide additional information. Overload dumpOn to provide a proper error text.

Definition at line 297 of file [YUIException.h](#).

## 5.162.2 Constructor & Destructor Documentation

### 5.162.2.1 YUIException() [1/2]

```
YUIException::YUIException ( )
```

Default constructor.

Use YUI\_THROW to throw exceptions.

Definition at line 63 of file [YUIException.cc](#).

### 5.162.2.2 YUIException() [2/2]

```
YUIException::YUIException (
    const std::string & msg_r )
```

Constructor taking a message.

Use YUI\_THROW to throw exceptions.

Definition at line 68 of file [YUIException.cc](#).

## 5.162.3 Member Function Documentation

### 5.162.3.1 log()

```
void YUIException::log (
    const YUIException & exception,
    const YCodeLocation & location,
    const char *const prefix ) [static]
```

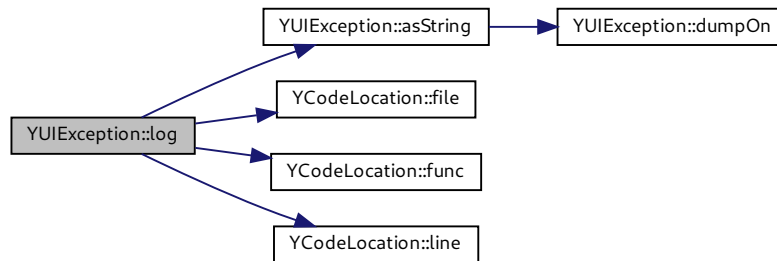
Drop a log line on throw, catch or rethrow.

Used by YUI\_THROW macros.

Definition at line 128 of file [YUIException.cc](#).



Here is the call graph for this function:



### 5.162.3.2 msg()

```
const std::string& YUIException::msg ( ) const [inline]
```

Return the message string provided to the constructor.

Note: This is not necessarily the complete error message. The whole error message is provided by `asString` or `dumpOn`.

Definition at line 334 of file [YUIException.h](#).

### 5.162.3.3 what()

```
virtual const char* YUIException::what ( ) const throw ( ) [inline], [virtual]
```

Return message string.

Reimplemented from `std::exception`.

Definition at line 370 of file [YUIException.h](#).

The documentation for this class was generated from the following files:

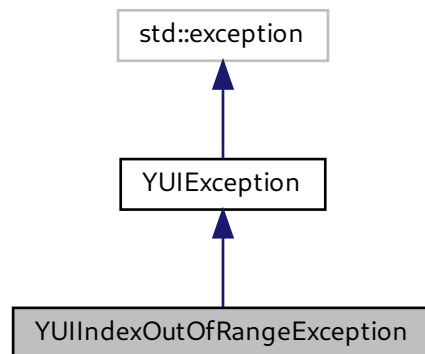
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YUIException.h`
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YUIException.cc`

## 5.163 YUIIndexOutOfRangeException Class Reference

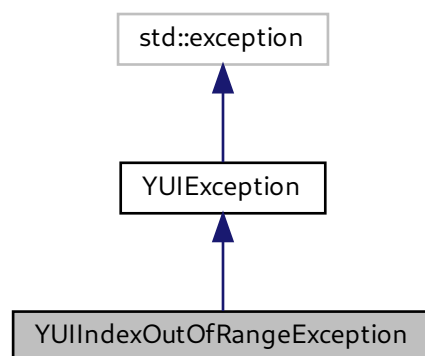
Exception class for "index out of range".

```
#include <YUIException.h>
```

Inheritance diagram for YUIIndexOutOfRangeException:



Collaboration diagram for YUIIndexOutOfRangeException:



## Public Member Functions

- [YUIIndexOutOfRangeException](#) (int [invalidIndex](#), int [validMin](#), int [validMax](#), const std::string &[msg](#)="")  
*Constructor.*
- int [invalidIndex](#) () const  
*Return the offending index value.*
- int [validMin](#) () const  
*Return the valid minimum index.*
- int [validMax](#) () const  
*Return the valid maximum index.*

## Protected Member Functions

- virtual std::ostream & [dumpOn](#) (std::ostream &str) const  
*Write proper error message with all relevant data.*

## Additional Inherited Members

### 5.163.1 Detailed Description

Exception class for "index out of range".

Definition at line [807](#) of file [YUIException.h](#).

### 5.163.2 Constructor & Destructor Documentation

#### 5.163.2.1 YUIIndexOutOfRangeException()

```
YUIIndexOutOfRangeException::YUIIndexOutOfRangeException (
    int invalidIndex,
    int validMin,
    int validMax,
    const std::string & msg = "" ) [inline]
```

Constructor.

'invalidIndex' is the offending index value. It should be between 'validMin' and 'validMax':

```
validMin <= index <= validMax
```

Definition at line [818](#) of file [YUIException.h](#).

### 5.163.3 Member Function Documentation

#### 5.163.3.1 dumpOn()

```
virtual std::ostream& YUIIndexOutOfRangeException::dumpOn (
    std::ostream & str ) const [inline], [protected], [virtual]
```

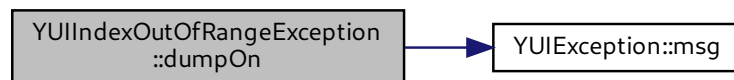
Write proper error message with all relevant data.

Reimplemented from [YUIException](#).

Reimplemented from [YUIException](#).

Definition at line [852](#) of file [YUIException.h](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

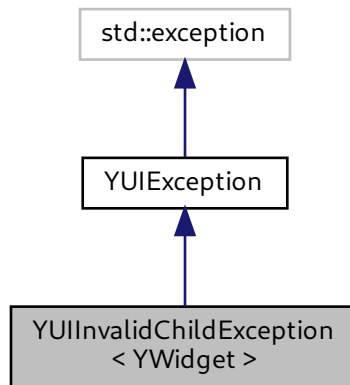
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YUIException.h`

## 5.164 YUIInvalidChildException< YWidget > Class Template Reference

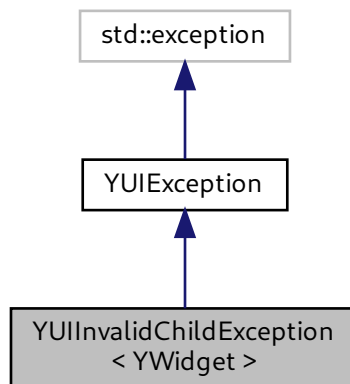
Exception class for "invalid child".

```
#include <YUIException.h>
```

Inheritance diagram for YUIInvalidChildException< YWidget >:



Collaboration diagram for YUIInvalidChildException< YWidget >:



## Public Member Functions

- **YUIInvalidChildException** ([YWidget](#) \*[container](#), [YWidget](#) \*[child](#)=0)
- [YWidget](#) \* [container](#) () const  
*Returns the container widget whose child should be removed etc.*
- [YWidget](#) \* [child](#) () const  
*Returns the child widget.*

## Protected Member Functions

- virtual std::ostream & [dumpOn](#) (std::ostream &str) const  
*Write proper error message with all relevant data.*

## Additional Inherited Members

### 5.164.1 Detailed Description

```
template<class YWidget>
class YUIInvalidChildException< YWidget >
```

Exception class for "invalid child".

One of:

- Attempt to remove a child from a children manager that is not in that manager's children list.
- Child widget of wrong type added to a container widget, e.g., anything other than a [YPushButton](#) added to a [YButtonBox](#).

Definition at line [712](#) of file [YUIException.h](#).

### 5.164.2 Member Function Documentation

#### 5.164.2.1 dumpOn()

```
template<class YWidget >
virtual std::ostream& YUIInvalidChildException< YWidget >::dumpOn (
    std::ostream & str ) const [inline], [protected], [virtual]
```

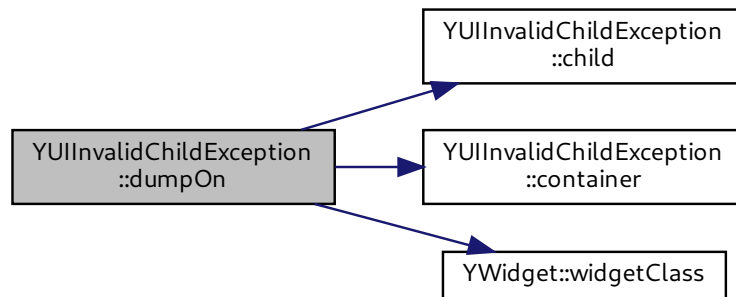
Write proper error message with all relevant data.

Reimplemented from [YUIException](#).

Reimplemented from [YUIException](#).

Definition at line [742](#) of file [YUIException.h](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

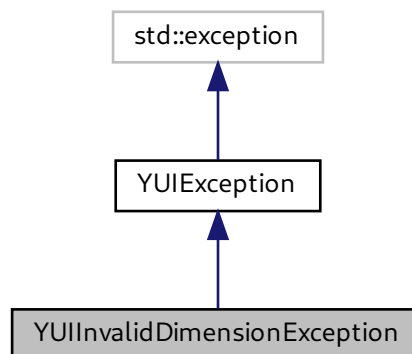
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YUIException.h`

## 5.165 YUIInvalidDimensionException Class Reference

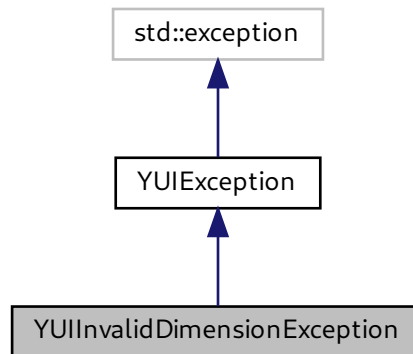
Exception class for "value other than YD\_HORIZ or YD\_VERT used for dimension".

```
#include <YUIException.h>
```

Inheritance diagram for YUIInvalidDimensionException:



Collaboration diagram for YUIInvalidDimensionException:



## Additional Inherited Members

### 5.165.1 Detailed Description

Exception class for "value other than YD\_HORIZ or YD\_VERT used for dimension".

Definition at line 792 of file [YUIException.h](#).

The documentation for this class was generated from the following file:

- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YUIException.h`

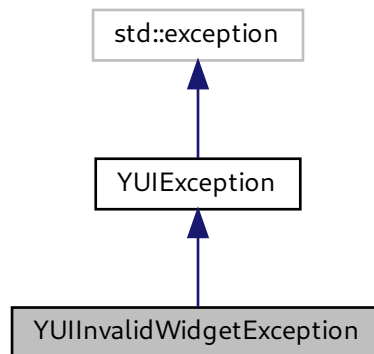
## 5.166 YUIInvalidWidgetException Class Reference

Exception class for invalid widgets.

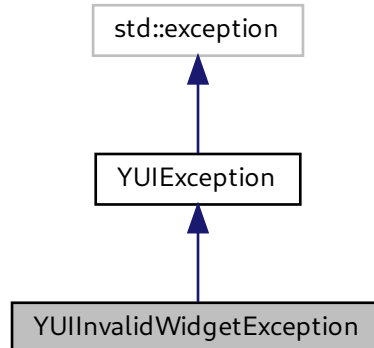
```
#include <YUIException.h>
```



Inheritance diagram for YUIInvalidWidgetException:



Collaboration diagram for YUIInvalidWidgetException:



## Additional Inherited Members

### 5.166.1 Detailed Description

Exception class for invalid widgets.

This is typically caused by widget pointers that continue living after the corresponding widget has already been deleted.

Definition at line 440 of file [YUIException.h](#).

The documentation for this class was generated from the following file:

- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YUIException.h`

## 5.167 YUILoader Class Reference

Class to load one of the concrete UI plug-ins: Qt, NCurses, Gtk.

```
#include <YUILoader.h>
```

### Static Public Member Functions

- static void `loadUI` (bool withThreads=false)  
*Load any of the available UI-plugins by this order and criteria:*
- static void `deleteUI` ()  
*This will make sure the UI singleton is deleted.*
- static void `loadRestAPIPlugin` (const std::string &wantedGUI, bool withThreads=false)  
*Method handles loading integration test framework and load underlying GUI using hints from loadUI.*
- static void `loadPlugin` (const std::string &name, bool withThreads=false)  
*Load a UI plug-in.*
- static bool `pluginExists` (const std::string &pluginBaseName)
- static void `loadExternalWidgets` (const std::string &name, const std::string &symbol="\_Z21createExternal↵ WidgetsPKc")  
*Load the given External Widgets plugin followed by its graphical extension implementation in the following order in the same way as loadUI:*

### 5.167.1 Detailed Description

Class to load one of the concrete UI plug-ins: Qt, NCurses, Gtk.

Definition at line 47 of file [YUILoader.h](#).

### 5.167.2 Member Function Documentation

#### 5.167.2.1 deleteUI()

```
void YUILoader::deleteUI ( ) [static]
```

This will make sure the UI singleton is deleted.

If the UI is already destroyed, it will do nothing. If there still is a UI object, it will be deleted.

This is particularly important for the NCurses UI so that the terminal settings are properly restored.

Definition at line 230 of file [YUILoader.cc](#).

### 5.167.2.2 loadExternalWidgets()

```
void YUILoader::loadExternalWidgets (
    const std::string & name,
    const std::string & symbol = "_Z21createExternalWidgetsPKc" ) [static]
```

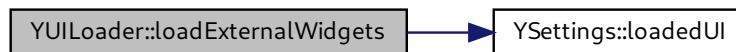
Load the given External Widgets plugin followed by its graphical extension implementation in the following order in the same way as loadUI:

- Qt, Gtk or NCurses

'name' is the user defined plugin name, graphical extension implementations have to be called 'name'-qt, 'name'-gtk and 'name'-ncurses. Following this rule plugin file names are as libyui-XX-YY.so.VER where: XX is the user defined name YY is the UI used (ncurses, gtk, qt) VER is the libyui so version 'symbol' is the function symbol to be loaded, e.g. YExternalWidgets\* 'symbol'(void) (e.g. default YExternalWidgets\* createExternalWidgets(const char \*) see createExternalWidgetsFunction\_t definition)

Definition at line 295 of file [YUILoader.cc](#).

Here is the call graph for this function:



### 5.167.2.3 loadPlugin()

```
void YUILoader::loadPlugin (
    const std::string & name,
    bool withThreads = false ) [static]
```

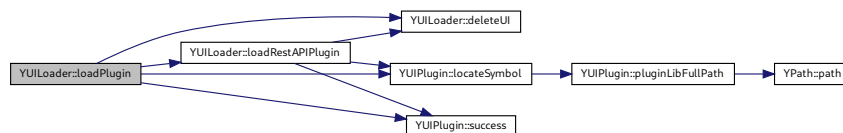
Load a UI plug-in.

'name' is one of the YUIPlugin\_ -defines above.

This might throw exceptions.

Definition at line 241 of file [YUILoader.cc](#).

Here is the call graph for this function:



### 5.167.2.4 loadUI()

```
void YUILoader::loadUI (
    bool withThreads = false ) [static]
```

Load any of the available UI-plugins by this order and criteria:

- Qt:
  - if \$DISPLAY is set
  - NCurses is user-selected and stdout is *not* a TTY
- Gtk:
  - if \$DISPLAY is set and Qt is not available,
  - a GTK-based desktop environment is detected from the environment variable XDG\_CURRENT\_DESKTOP
  - any of the above pre-conditions are met and NCurses is user-selected, but stdout is *not* a TTY
- NCurses:
  - if \$DISPLAY is *not* set and stdout is a TTY
  - Qt and Gtk are not available and stdout is a TTY

This can be overridden by either:

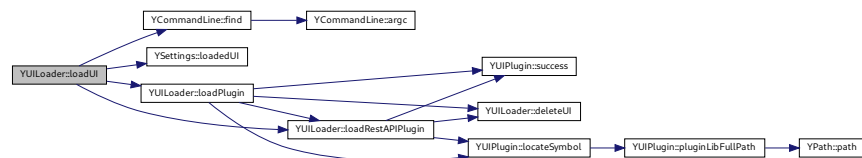
- specifying one of the switches on the command-line of the program
  - '-gtk',
  - '-ncurses', or
  - '-qt'
- setting the environment variable YUI\_PREFERED\_BACKEND to one of
  - 'gtk',
  - 'ncurses', or
  - 'qt'

If a command-line switch is given to the program, the setting from the environment variable will be overridden by the UI-plugin chosen with the switch.

If the user-selected UI-plugin is not installed on the system, an installed UI-plugin will be chosen by the above criteria.

Definition at line 51 of file [YUILoader.cc](#).

Here is the call graph for this function:



The documentation for this class was generated from the following files:

- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YUILoader.h
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YUILoader.cc

## 5.168 YUILog Class Reference

UI logging.

```
#include <YUILog.h>
```

### Public Member Functions

- `std::ostream & log` (YUILogLevel\_t logLevel, const char \*logComponent, const char \*sourceFileName, int lineNo, const char \*functionName)  
*Generic log function.*

### Static Public Member Functions

- static `std::ostream & debug` (const char \*logComponent, const char \*sourceFileName, int lineNo, const char \*functionName)  
*Logging functions for each log level.*
- static `std::ostream & milestone` (const char \*logComponent, const char \*sourceFileName, int lineNo, const char \*functionName)
- static `std::ostream & warning` (const char \*logComponent, const char \*sourceFileName, int lineNo, const char \*functionName)
- static `std::ostream & error` (const char \*logComponent, const char \*sourceFileName, int lineNo, const char \*functionName)
- static `YUILog * instance` ()  
*Return the singleton object for this class.*
- static void `enableDebugLogging` (bool debugLogging=true)  
*Enable or disable debug logging.*
- static bool `debugLoggingEnabled` ()  
*Return 'true' if debug logging is enabled, 'false' if not.*
- static bool `setLogFileName` (const std::string &logFileName)  
*Set the log file name to be used with the standard logger function.*
- static std::string `logFileName` ()  
*Return the current log file name or an empty string if stderr is used.*
- static void `setLoggerFunction` (YUILoggerFunction loggerFunction)  
*Set the UI logger function.*
- static YUILoggerFunction `loggerFunction` (bool returnStdLogger=false)  
*Return the UI logger function.*
- static void `setEnableDebugLoggingHooks` (YUIEnableDebugLoggingFunction enableFunction, YUIDebugLoggingEnabledFunction isEnabledFunction)  
*Set the hook functions to enable/disable debug logging and to query if debug logging is enabled:*
- static YUIEnableDebugLoggingFunction `enableDebugLoggingHook` ()  
*Return the hook function that enables or disables debug logging or 0 if no such hook function is set.*
- static YUIDebugLoggingEnabledFunction `debugLoggingEnabledHook` ()  
*Return the hook function that checks if debug logging is enabled or 0 if no such hook function is set.*
- static std::string `basename` (const std::string &fileNameWithPath)  
*Return the base name without path from a file name with path.*

### 5.168.1 Detailed Description

UI logging.

Definition at line 99 of file [YUILog.h](#).

### 5.168.2 Member Function Documentation

#### 5.168.2.1 debug()

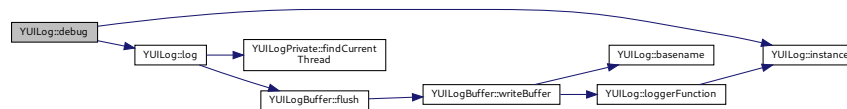
```
ostream & YUILog::debug (
    const char * logComponent,
    const char * sourceFileName,
    int lineNo,
    const char * functionName ) [static]
```

Logging functions for each log level.

They all access the singleton object for this class. This means that the first call to any of those functions will create the singleton [YUILog](#) object.

Definition at line 487 of file [YUILog.cc](#).

Here is the call graph for this function:



#### 5.168.2.2 instance()

```
YUILog * YUILog::instance ( ) [static]
```

Return the singleton object for this class.

This will create the singleton if it doesn't exist yet.

Definition at line 333 of file [YUILog.cc](#).

### 5.168.2.3 log()

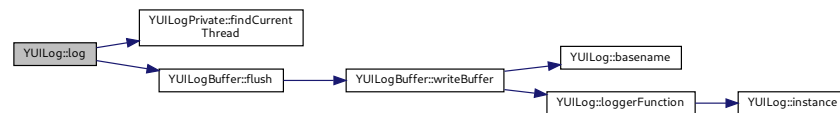
```
ostream & YUILog::log (
    YUILogLevel_t logLevel,
    const char * logComponent,
    const char * sourceFileName,
    int lineNo,
    const char * functionName )
```

Generic log function.

[debug\(\)](#), [milestone\(\)](#) etc. ultimately all call this function.

Definition at line [456](#) of file [YUILog.cc](#).

Here is the call graph for this function:



### 5.168.2.4 logFileName()

```
string YUILog::logFileName ( ) [static]
```

Return the current log file name or an empty string if stderr is used.

Notice that this information is only relevant as long as the standard logger function is used.

Definition at line [384](#) of file [YUILog.cc](#).

Here is the call graph for this function:



### 5.168.2.5 loggerFunction()

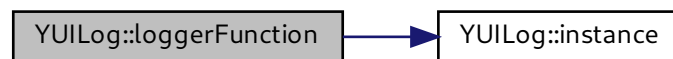
```
YUILoggerFunction YUILog::loggerFunction (
    bool returnStdLogger = false ) [static]
```

Return the UI logger function.

If stderr is used for logging (i.e. no logger function set), 0 is returned (unless 'returnStdLogger' is 'true', in which case the internally used stderr-logger is returned).

Definition at line 421 of file [YUILog.cc](#).

Here is the call graph for this function:



### 5.168.2.6 setEnableDebugLoggingHooks()

```
void YUILog::setEnableDebugLoggingHooks (
    YUIEnableDebugLoggingFunction enableFunction,
    YUIDebugLoggingEnabledFunction isEnabledFunction ) [static]
```

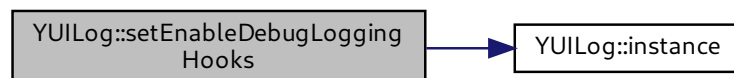
Set the hook functions to enable/disable debug logging and to query if debug logging is enabled:

```
void enableDebugLogging( bool enable );
bool debugLoggingEnabled();
```

If those functions are set, they will be used instead of the internal "debugLogging" flag.

Definition at line 433 of file [YUILog.cc](#).

Here is the call graph for this function:





### 5.168.2.7 setLogFileName()

```
bool YUILog::setLogFileName (
    const std::string & logFileName ) [static]
```

Set the log file name to be used with the standard logger function.

Output will be appended to this file.

Until this file name is set, the standard logger function logs to stderr. Set the log file name to an empty string to log to stderr again.

This returns 'true' upon success (opening the file was successful), 'false' upon error.

Notice:

(1) This file name is only relevant as long as the standard logger function is used. Custom logger functions may or may not use this file name.

(2) No attempt is made to do anything fancy with the log file like log file rotation when a certain file size is reached. Applications that need this should use a custom logger function. See also [setLoggerFunction\(\)](#).

Definition at line 348 of file [YUILog.cc](#).

Here is the call graph for this function:



### 5.168.2.8 setLoggerFunction()

```
void YUILog::setLoggerFunction (
    YUILoggerFunction loggerFunction ) [static]
```

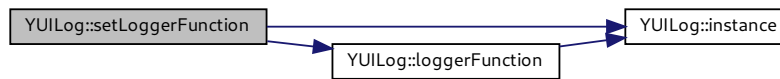
Set the UI logger function.

This is the function that will ultimately receive all UI log output (except debug logging if debug logging is disabled).

By default, all logging is output to stderr. This behaviour can be restored if 0 is passed as a function pointer here.

Definition at line 411 of file [YUILog.cc](#).

Here is the call graph for this function:



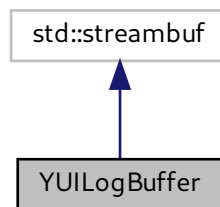
The documentation for this class was generated from the following files:

- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YUILog.h`
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YUILog.cc`

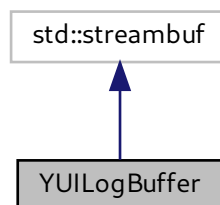
## 5.169 YUILogBuffer Class Reference

Stream buffer class that will use the [YUILog](#)'s logger function.

Inheritance diagram for YUILogBuffer:



Collaboration diagram for YUILogBuffer:



## Public Member Functions

- [YUILogBuffer](#) ()  
*Constructor.*
- virtual [~YUILogBuffer](#) ()  
*Destructor.*
- virtual std::streamsize [xspu](#)tn (const char \*sequence, std::streamsize maxLength)  
*Write (no more than maxLength characters of) a sequence of characters and return the number of characters written.*
- virtual int [overflow](#) (int ch=EOF)  
*Write one character in case of buffer overflow.*
- std::streamsize [writeBuffer](#) (const char \*sequence, std::streamsize seqLen)  
*Write (no more than maxLength characters of) a sequence of characters and return the number of characters written.*
- void [flush](#) ()  
*Flush the output buffer: Write any data unwritten so far.*

## Friends

- class [YUILog](#)

### 5.169.1 Detailed Description

Stream buffer class that will use the [YUILog](#)'s logger function.

See also <http://blogs.awesomeplay.com/elanthis/archives/2007/12/10/>

Definition at line 58 of file [YUILog.cc](#).

### 5.169.2 Member Function Documentation

#### 5.169.2.1 overflow()

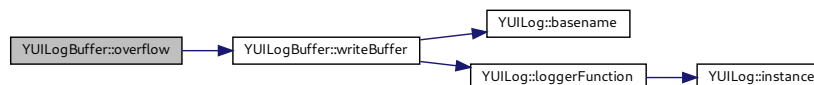
```
int YUILogBuffer::overflow (
    int ch = EOF ) [virtual]
```

Write one character in case of buffer overflow.

Reimplemented from streambuf.

Definition at line 170 of file [YUILog.cc](#).

Here is the call graph for this function:



### 5.169.2.2 writeBuffer()

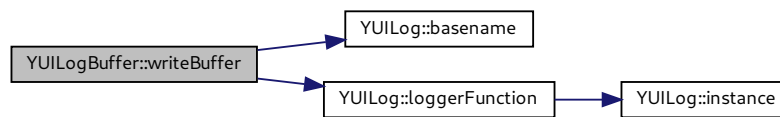
```
std::streamsize YUILogBuffer::writeBuffer (
    const char * sequence,
    std::streamsize seqLen )
```

Write (no more than `maxLength` characters of) a sequence of characters and return the number of characters written.

This is the actual worker function that uses the [YUILog::loggerFunction](#) to actually write characters.

Definition at line 125 of file [YUILog.cc](#).

Here is the call graph for this function:



### 5.169.2.3 xspn()

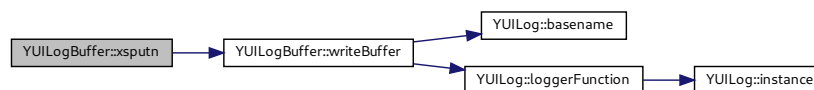
```
std::streamsize YUILogBuffer::xspn (
    const char * sequence,
    std::streamsize maxLength ) [virtual]
```

Write (no more than `maxLength` characters of) a sequence of characters and return the number of characters written.

Reimplemented from `streambuf`. This is called for all output operations on the associated ostream.

Definition at line 163 of file [YUILog.cc](#).

Here is the call graph for this function:

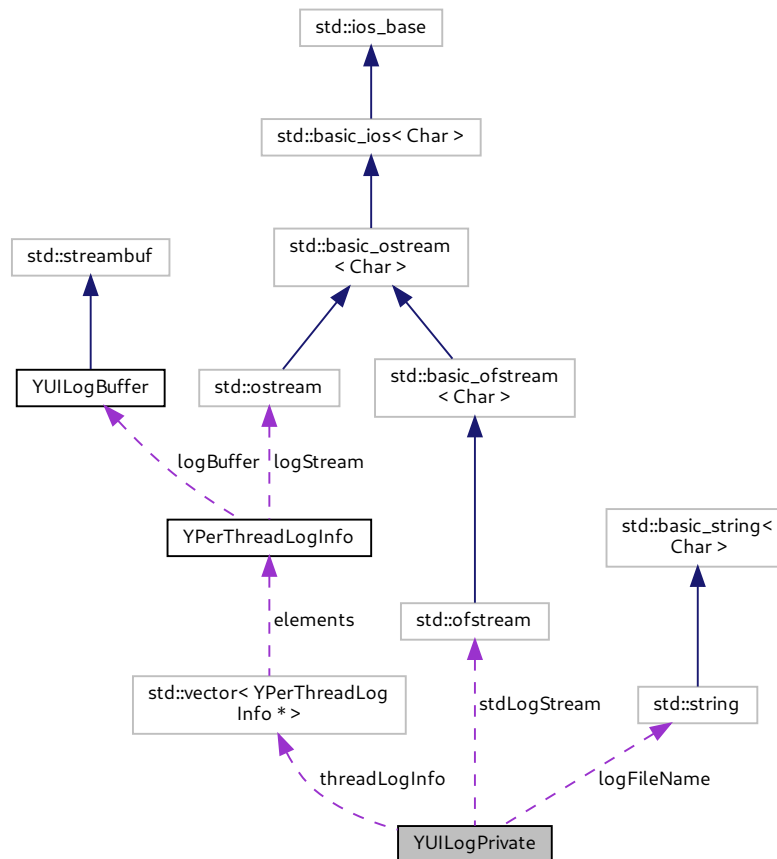


The documentation for this class was generated from the following file:

- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YUILog.cc`

## 5.170 YUILogPrivate Struct Reference

Collaboration diagram for YUILogPrivate:



### Public Member Functions

- [YUILogPrivate](#) ()  
*Constructor.*
- [~YUILogPrivate](#) ()  
*Destructor.*
- [YPerThreadLogInfo \\* findCurrentThread](#) ()  
*Find the per-thread logging information for the current thread.*

### Public Attributes

- string **logFileName**

- `std::ofstream` **stdLogStream**
- `YUILoggerFunction` **loggerFunction**
- `YUIEnableDebugLoggingFunction` **enableDebugLoggingHook**
- `YUIDebugLoggingEnabledFunction` **debugLoggingEnabledHook**
- `bool` **enableDebugLogging**
- `std::vector< YPerThreadLogInfo * >` **threadLogInfo**

### 5.170.1 Detailed Description

Definition at line 254 of file [YUILog.cc](#).

### 5.170.2 Member Function Documentation

#### 5.170.2.1 findCurrentThread()

```
YPerThreadLogInfo* YUILogPrivate::findCurrentThread ( ) [inline]
```

Find the per-thread logging information for the current thread.

Create a new one if it doesn't exist yet.

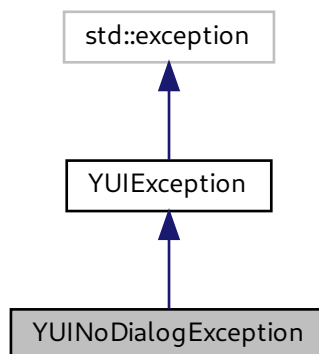
Definition at line 279 of file [YUILog.cc](#).

The documentation for this struct was generated from the following file:

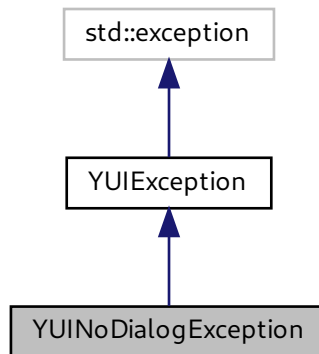
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YUILog.cc`

## 5.171 YUINoDialogException Class Reference

Inheritance diagram for YUINoDialogException:



Collaboration diagram for YUINoDialogException:



## Additional Inherited Members

### 5.171.1 Detailed Description

Definition at line [467](#) of file [YUIException.h](#).

The documentation for this class was generated from the following file:

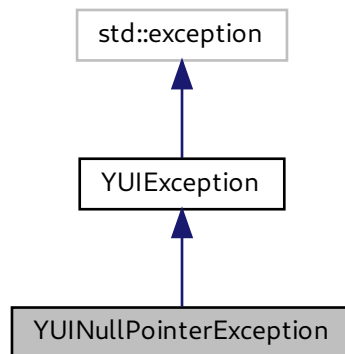
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YUIException.h`

## 5.172 YUINullPointerException Class Reference

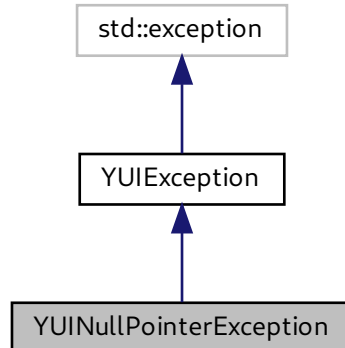
Exception class for generic null pointer exceptions.

```
#include <YUIException.h>
```

Inheritance diagram for YUINullPointerException:



Collaboration diagram for YUINullPointerException:



## Additional Inherited Members

### 5.172.1 Detailed Description

Exception class for generic null pointer exceptions.

When available, a more specialized exception class should be used.

Definition at line 407 of file [YUIException.h](#).

The documentation for this class was generated from the following file:



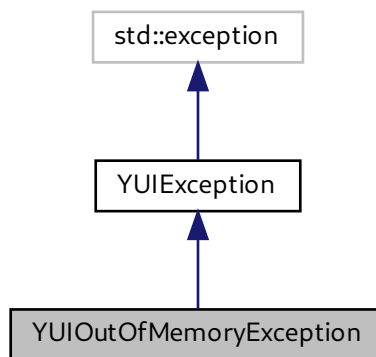
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YUIException.h

## 5.173 YUIOutOfMemoryException Class Reference

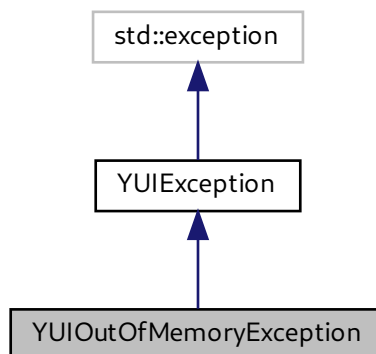
Exception class for "out of memory".

```
#include <YUIException.h>
```

Inheritance diagram for YUIOutOfMemoryException:



Collaboration diagram for YUIOutOfMemoryException:



## Additional Inherited Members

### 5.173.1 Detailed Description

Exception class for "out of memory".

Typically used if operator new returned 0.

Definition at line 423 of file [YUIException.h](#).

The documentation for this class was generated from the following file:

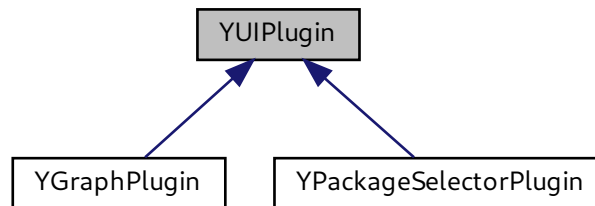
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YUIException.h](#)

## 5.174 YUIPlugin Class Reference

Wrapper class for dlopen() and related.

```
#include <YUIPlugin.h>
```

Inheritance diagram for YUIPlugin:



## Public Member Functions

- [YUIPlugin](#) (const char \*[pluginLibBaseName](#))  
*Constructor: Load the specified plugin library from the standard UI plugin directory (/usr/lib/yui/).*
- virtual [~YUIPlugin](#) ()  
*Destructor.*
- void [unload](#) ()  
*Unload this plugin.*
- void \* [locateSymbol](#) (const char \*symbol)  
*Try to locate the specified symbol (function or global variable) in the plugin library.*
- bool [error](#) () const  
*Returns 'true' if there was an error loading the plugin.*
- bool [success](#) () const  
*Returns 'true' if there was no error loading the plugin.*
- std::string [errorMsg](#) () const  
*Returns a human readable (but in most cases untranslated) error message if there was an error.*

## Protected Member Functions

- void \* [pluginLibHandle](#) ()  
*Returns the dlopen() handle of the plugin library.*
- std::string [pluginLibBaseName](#) () const  
*Returns the base name of the plugin library.*
- std::string [pluginLibFullPath](#) () const  
*Returns the full path of the plugin library.*

### 5.174.1 Detailed Description

Wrapper class for dlopen() and related.

Definition at line 35 of file [YUIPlugin.h](#).

### 5.174.2 Constructor & Destructor Documentation

#### 5.174.2.1 ~YUIPlugin()

```
YUIPlugin::~YUIPlugin ( ) [virtual]
```

Destructor.

Please note that this will NOT attempt to unload the plugin library since this is usually counterproductive. If unloading the plugin is desired, call [unload\(\)](#) manually.

Definition at line 60 of file [YUIPlugin.cc](#).

### 5.174.3 Member Function Documentation

### 5.174.3.1 locateSymbol()

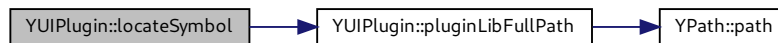
```
void * YUIPlugin::locateSymbol (
    const char * symbol )
```

Try to locate the specified symbol (function or global variable) in the plugin library.

Returns the in-memory address of that symbol or 0 if it could not be found or if loading the plugin library had failed in the constructor.

Definition at line 89 of file [YUIPlugin.cc](#).

Here is the call graph for this function:



### 5.174.3.2 unload()

```
void YUIPlugin::unload ( )
```

Unload this plugin.

This calls `dlclose()` which will unload the plugin library if it is no longer used, i.e. if the reference count `dlopen()` uses reaches 0.

Definition at line 68 of file [YUIPlugin.cc](#).

The documentation for this class was generated from the following files:

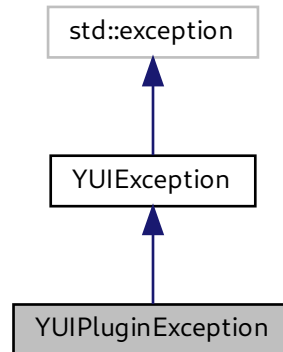
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YUIPlugin.h`
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YUIPlugin.cc`

## 5.175 YUIPluginException Class Reference

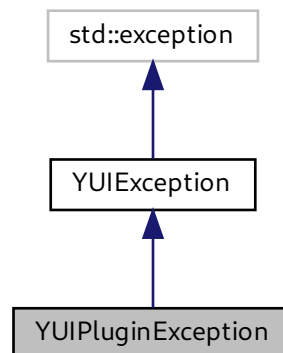
Exception class for plugin load failure.

```
#include <YUIException.h>
```

Inheritance diagram for YUIPluginException:



Collaboration diagram for YUIPluginException:



### Public Member Functions

- **YUIPluginException** (const std::string &pluginName)

## Additional Inherited Members

### 5.175.1 Detailed Description

Exception class for plugin load failure.

Definition at line 875 of file [YUIException.h](#).

The documentation for this class was generated from the following file:

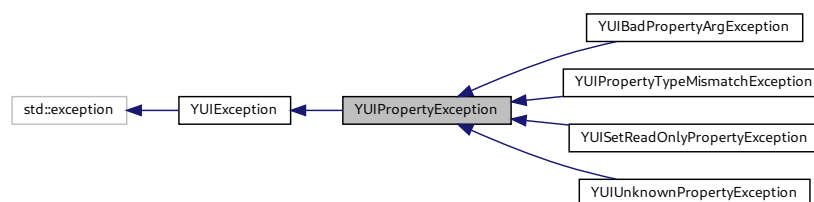
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YUIException.h](#)

## 5.176 YUIPropertyException Class Reference

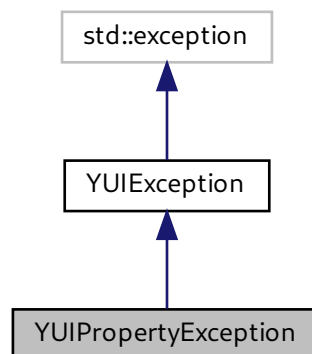
Abstract base class for widget property exceptions.

```
#include <YUIException.h>
```

Inheritance diagram for YUIPropertyException:



Collaboration diagram for YUIPropertyException:



## Public Member Functions

- [YProperty](#) [property](#) () const  
*Returns the property that caused this exception.*
- [YWidget](#) \* [widget](#) () const  
*Returns the corresponding widget or 0 if there was none.*
- void [setWidget](#) ([YWidget](#) \*w)  
*Set the corresponding widget.*

## Protected Member Functions

- **YUIPropertyException** (const [YProperty](#) &prop, [YWidget](#) \*widget=0)
- virtual std::ostream & [dumpOn](#) (std::ostream &str) const =0  
*Write proper error message with all relevant data.*

## Additional Inherited Members

### 5.176.1 Detailed Description

Abstract base class for widget property exceptions.

Definition at line 506 of file [YUIException.h](#).

### 5.176.2 Member Function Documentation

#### 5.176.2.1 [dumpOn\(\)](#)

```
virtual std::ostream& YUIPropertyException::dumpOn (  
    std::ostream & str ) const [protected], [pure virtual]
```

Write proper error message with all relevant data.

Reimplemented from [YUIException](#).

Reimplemented from [YUIException](#).

Implemented in [YUIBadPropertyArgException](#), [YUISetReadOnlyPropertyException](#), [YUIPropertyTypeMismatchException](#), and [YUIUnknownPropertyException](#).

The documentation for this class was generated from the following file:

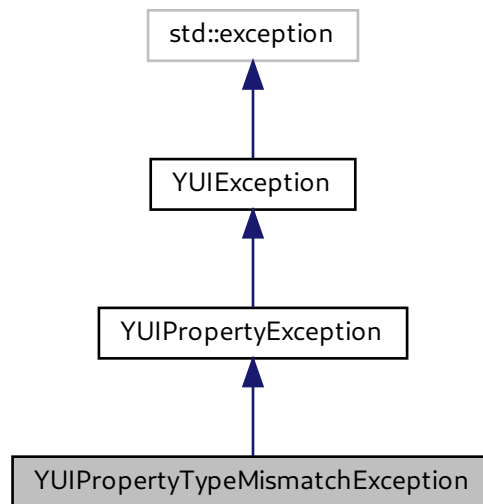
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YUIException.h

## 5.177 YUIPropertyTypeMismatchException Class Reference

Exception class for "property type mismatch": The application tried to set a property with a wrong type.

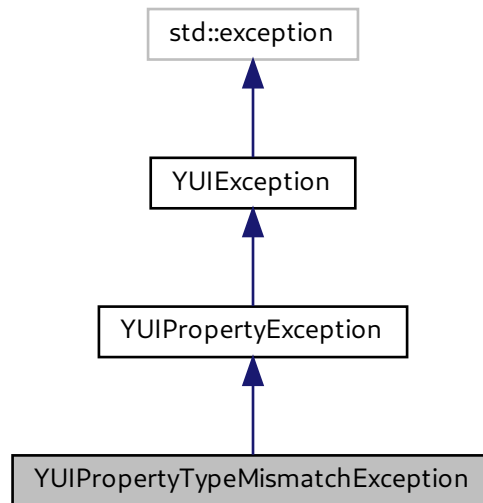
```
#include <YUIException.h>
```

Inheritance diagram for YUIPropertyTypeMismatchException:





Collaboration diagram for YUIPropertyTypeMismatchException:



## Public Member Functions

- **YUIPropertyTypeMismatchException** (const [YProperty](#) &`property`, `YPropertyType` `type`, `YWidget` \*`widget`=0)
- `YPropertyType` `type` () const  
*Return the property type the application tried to set.*

## Protected Member Functions

- virtual `std::ostream` & `dumpOn` (`std::ostream` &`str`) const  
*Write proper error message with all relevant data.*

## Additional Inherited Members

### 5.177.1 Detailed Description

Exception class for "property type mismatch": The application tried to set a property with a wrong type.

Definition at line 578 of file [YUIException.h](#).

### 5.177.2 Member Function Documentation

### 5.177.2.1 dumpOn()

```
ostream & YUIPropertyTypeMismatchException::dumpOn (
    std::ostream & str ) const [protected], [virtual]
```

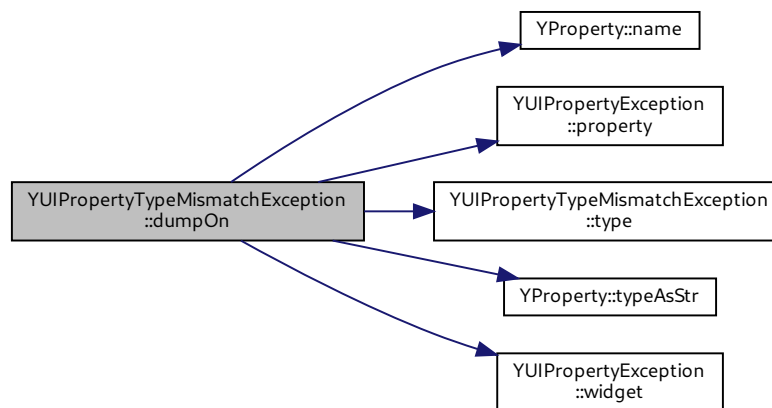
Write proper error message with all relevant data.

Reimplemented from [YUIException](#).

Implements [YUIPropertyException](#).

Definition at line 162 of file [YUIException.cc](#).

Here is the call graph for this function:



The documentation for this class was generated from the following files:

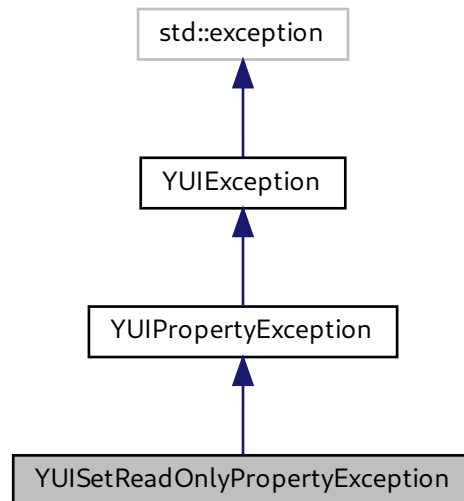
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YUIException.h`
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YUIException.cc`

## 5.178 YUISetReadOnlyPropertyException Class Reference

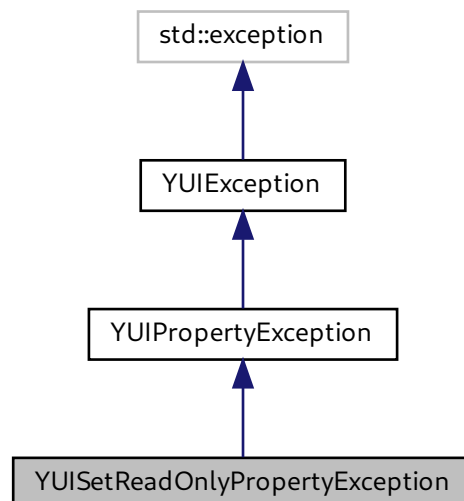
Exception class for attempt to set a read-only property.

```
#include <YUIException.h>
```

Inheritance diagram for YUISetReadOnlyPropertyException:



Collaboration diagram for YUISetReadOnlyPropertyException:



## Public Member Functions

- **YUISetReadOnlyPropertyException** (const [YProperty](#) &[property](#), [YWidget](#) \*[widget](#)=0)

## Protected Member Functions

- virtual `std::ostream & dumpOn (std::ostream &str) const`  
*Write proper error message with all relevant data.*

## Additional Inherited Members

### 5.178.1 Detailed Description

Exception class for attempt to set a read-only property.

Definition at line [613](#) of file [YUIException.h](#).

### 5.178.2 Member Function Documentation

#### 5.178.2.1 `dumpOn()`

```
ostream & YUISetReadOnlyPropertyException::dumpOn (
    std::ostream & str ) const [protected], [virtual]
```

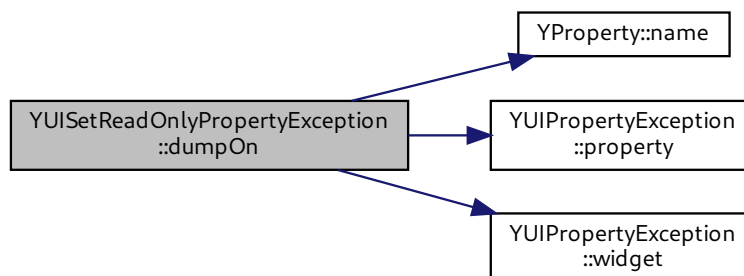
Write proper error message with all relevant data.

Reimplemented from [YUIException](#).

Implements [YUIPropertyException](#).

Definition at line [181](#) of file [YUIException.cc](#).

Here is the call graph for this function:

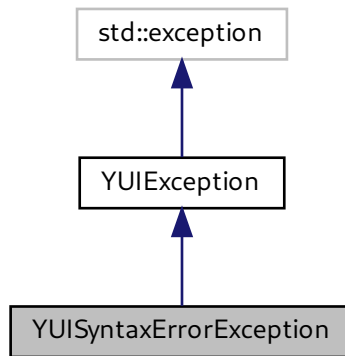


The documentation for this class was generated from the following files:

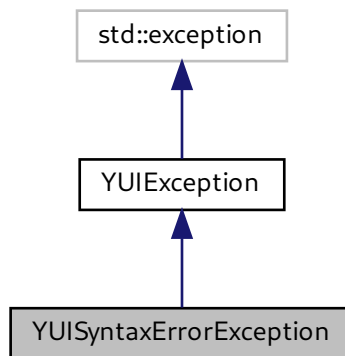
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YUIException.h`
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YUIException.cc`

## 5.179 YUISyntaxErrorException Class Reference

Inheritance diagram for YUISyntaxErrorException:



Collaboration diagram for YUISyntaxErrorException:



### Public Member Functions

- **YUISyntaxErrorException** (const std::string &msg)

## Additional Inherited Members

### 5.179.1 Detailed Description

Definition at line 491 of file [YUIException.h](#).

The documentation for this class was generated from the following file:

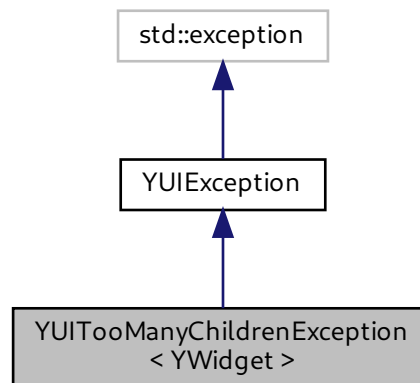
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YUIException.h](#)

## 5.180 YUITooManyChildrenException< YWidget > Class Template Reference

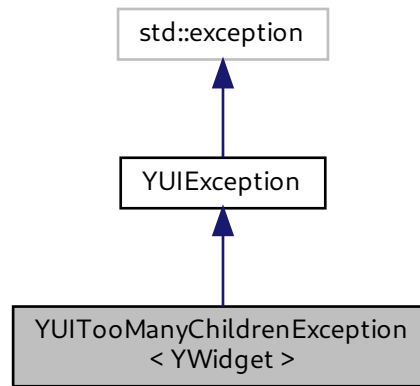
Exception class for "too many children": Attempt to add a child to a widget class that can't handle children ([YPushButton](#) etc.) or just one child ([YFrame](#), [YDialog](#)).

```
#include <YUIException.h>
```

Inheritance diagram for YUITooManyChildrenException< YWidget >:



Collaboration diagram for YUITooManyChildrenException< YWidget >:



## Public Member Functions

- **YUITooManyChildrenException** ([YWidget](#) \*[container](#))
- [YWidget](#) \* [container](#) () const  
*Returns the container widget that can't handle that many children.*

## Protected Member Functions

- virtual std::ostream & [dumpOn](#) (std::ostream &str) const  
*Write proper error message with all relevant data.*

## Additional Inherited Members

### 5.180.1 Detailed Description

```
template<class YWidget>
class YUITooManyChildrenException< YWidget >
```

Exception class for "too many children": Attempt to add a child to a widget class that can't handle children ([YPushButton](#) etc.) or just one child ([YFrame](#), [YDialog](#)).

Definition at line [663](#) of file [YUIException.h](#).

## 5.180.2 Member Function Documentation

### 5.180.2.1 dumpOn()

```
template<class YWidget >
virtual std::ostream& YUITooManyChildrenException< YWidget >::dumpOn (
    std::ostream & str ) const [inline], [protected], [virtual]
```

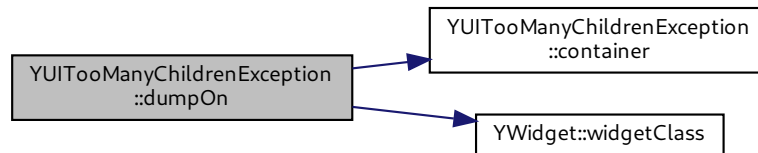
Write proper error message with all relevant data.

Reimplemented from [YUIException](#).

Reimplemented from [YUIException](#).

Definition at line [686](#) of file [YUIException.h](#).

Here is the call graph for this function:



The documentation for this class was generated from the following file:

- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YUIException.h`

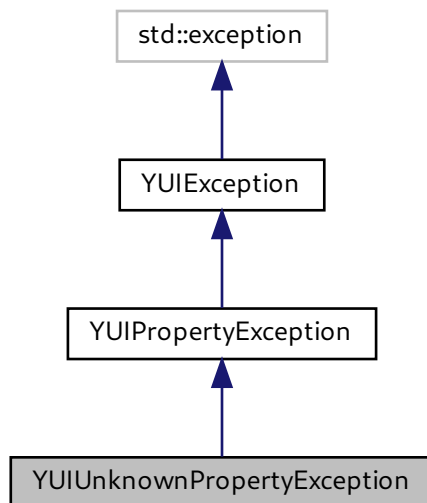
## 5.181 YUIUnknownPropertyException Class Reference

Exception class for "unknown property name": The application tried to set (or query) a property that doesn't exist.

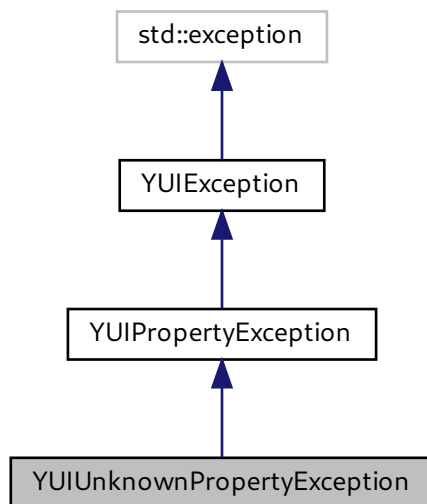
```
#include <YUIException.h>
```



Inheritance diagram for YUIUnknownPropertyException:



Collaboration diagram for YUIUnknownPropertyException:



## Public Member Functions

- **YUIUnknownPropertyException** (const std::string &propertyName, [YWidget](#) \*widget=0)

## Protected Member Functions

- virtual std::ostream & [dumpOn](#) (std::ostream &str) const  
*Write proper error message with all relevant data.*

## Additional Inherited Members

### 5.181.1 Detailed Description

Exception class for "unknown property name": The application tried to set (or query) a property that doesn't exist.

Definition at line [553](#) of file [YUIException.h](#).

### 5.181.2 Member Function Documentation

#### 5.181.2.1 dumpOn()

```
ostream & YUIUnknownPropertyException::dumpOn (
    std::ostream & str ) const [protected], [virtual]
```

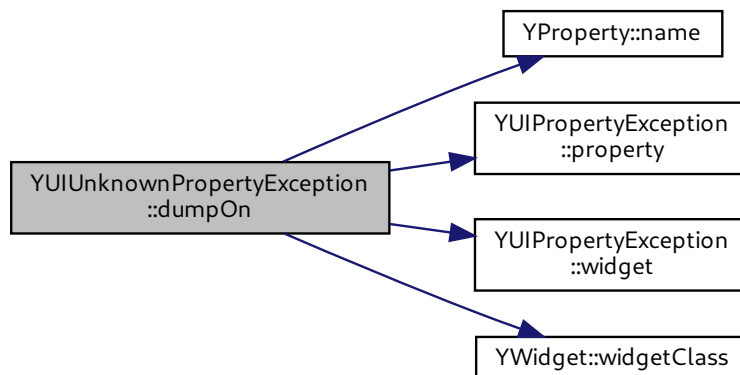
Write proper error message with all relevant data.

Reimplemented from [YUIException](#).

Implements [YUIPropertyException](#).

Definition at line [141](#) of file [YUIException.cc](#).

Here is the call graph for this function:



The documentation for this class was generated from the following files:

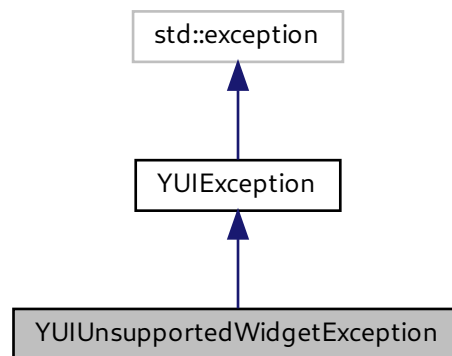
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YUIException.h
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YUIException.cc

## 5.182 YUIUnsupportedWidgetException Class Reference

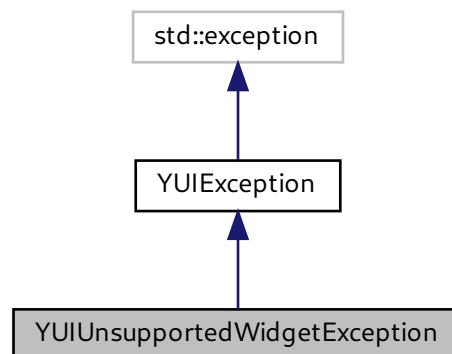
Exception class for "optional widget not supported".

```
#include <YUIException.h>
```

Inheritance diagram for YUIUnsupportedWidgetException:



Collaboration diagram for YUIUnsupportedWidgetException:



## Public Member Functions

- **YUIUnsupportedWidgetException** (const std::string &widgetType)

## Additional Inherited Members

### 5.182.1 Detailed Description

Exception class for "optional widget not supported".

Note that applications are supposed to check with [YUI::optionalWidgetFactory\(\)](#)->hasXYWidget() before trying to create such a widget. This exception is thrown if that check wasn't done, the application tried to create that kind of widget anyway, but the UI doesn't support that widget.

Definition at line 775 of file [YUIException.h](#).

The documentation for this class was generated from the following file:

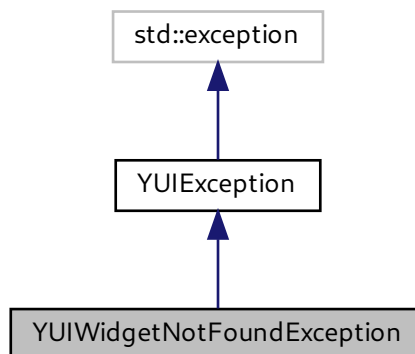
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YUIException.h

## 5.183 YUIWidgetNotFoundException Class Reference

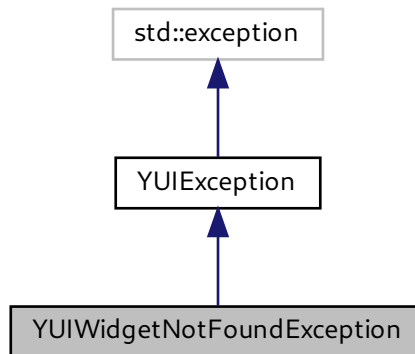
Exception class for "No widget found with that ID".

```
#include <YUIException.h>
```

Inheritance diagram for YUIWidgetNotFoundException:



Collaboration diagram for YUIWidgetNotFoundException:



## Public Member Functions

- **YUIWidgetNotFoundException** (const std::string &idString)

## Additional Inherited Members

### 5.183.1 Detailed Description

Exception class for "No widget found with that ID".

Definition at line 455 of file [YUIException.h](#).

The documentation for this class was generated from the following file:

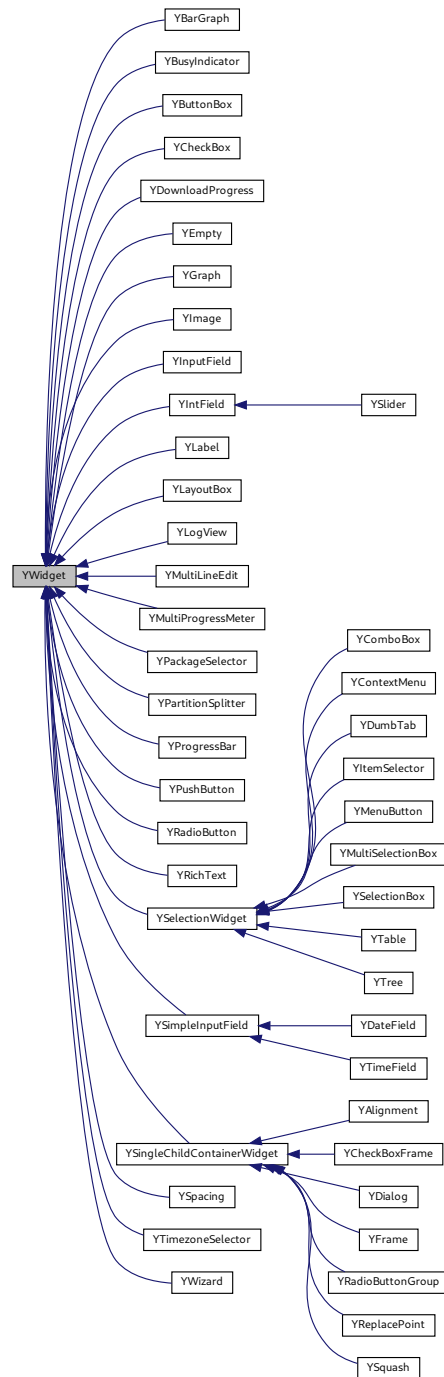
- /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YUIException.h

## 5.184 YWidget Class Reference

Abstract base class of all UI widgets.

```
#include <YWidget.h>
```

Inheritance diagram for YWidget:



## Public Member Functions

- virtual [~YWidget](#) ()

*Destructor.*

- virtual const char \* [widgetClass](#) () const  
*Returns a descriptive name of this widget class for logging, debugging etc.*
- virtual std::string [debugLabel](#) () const  
*Returns a descriptive label of this widget instance.*
- std::string [helpText](#) () const  
*Return the help text for this widget.*
- void [setHelpText](#) (const std::string &[helpText](#))  
*Set a help text for this widget.*
- virtual const [YPropertySet](#) & [propertySet](#) ()  
*Return this class's property set.*
- virtual bool [setProperty](#) (const std::string &propertyName, const [YPropertyValue](#) &val)  
*Set a property.*
- virtual [YPropertyValue](#) [getProperty](#) (const std::string &propertyName)  
*Get a property.*
- bool [hasChildren](#) () const  
*Returns 'true' if this widget has any children.*
- [YWidget](#) \* [firstChild](#) () const  
*Returns the first child or 0 if there is none.*
- [YWidget](#) \* [lastChild](#) () const  
*Returns the last child or 0 if there is none.*
- [YWidgetListIterator](#) [childrenBegin](#) () const  
*Return an iterator that points to the first child or to [childrenEnd\(\)](#) if there are no children.*
- [YWidgetListIterator](#) [childrenEnd](#) () const  
*Return an iterator that points after the last child.*
- [YWidgetListConstIterator](#) [childrenConstBegin](#) () const  
*Return a const iterator that points to the first child or to [childrenEnd\(\)](#) if there are no children.*
- [YWidgetListConstIterator](#) [childrenConstEnd](#) () const  
*Return a const iterator that points after the last child.*
- [YWidgetListIterator](#) [begin](#) ()  
*A helper for the range-based "for" loop.*
- [YWidgetListIterator](#) [end](#) ()  
*A helper for the range-based "for" loop.*
- int [childrenCount](#) () const  
*Returns the current number of children.*
- bool [contains](#) ([YWidget](#) \*child) const  
*Checks if 'child' is a (direct!) child of this widget.*
- virtual void [addChild](#) ([YWidget](#) \*child)  
*Add a new child.*
- virtual void [removeChild](#) ([YWidget](#) \*child)  
*Remove a child.*
- void [deleteChildren](#) ()  
*Delete all children and remove them from the children manager's list.*
- [YWidget](#) \* [parent](#) () const  
*Return this widget's parent or 0 if it doesn't have a parent.*
- bool [hasParent](#) () const  
*Return 'true' if this widget has a parent, 'false' if not.*

- void **setParent** (YWidget \*newParent)  
*Set this widget's parent.*
- YDialog \* **findDialog** ()  
*Traverse up the widget hierarchy and find the dialog this widget belongs to.*
- YWidget \* **findWidget** (YWidgetID \*id, bool doThrow=true) const  
*Recursively find a widget by its ID.*
- virtual int **preferredWidth** ()=0  
*Preferred width of the widget.*
- virtual int **preferredHeight** ()=0  
*Preferred height of the widget.*
- virtual int **preferredSize** (YUIDimension dim)  
*Preferred size of the widget in the specified dimension.*
- virtual void **setSize** (int newWidth, int newHeight)=0  
*Set the new size of the widget.*
- bool **isValid** () const  
*Checks whether or not this object is valid.*
- bool **beingDestroyed** () const  
*Check if this widget is in the process of being destroyed.*
- void \* **widgetRep** () const  
*Return a pointer to the underlying toolkit's (Qt, ...) widget representing this abstract UI widget.*
- void **setWidgetRep** (void \*toolkitWidgetRep)  
*Set the pointer to the underlying toolkit's (Qt, ...) widget representing this abstract UI widget.*
- bool **hasId** () const  
*Returns 'true' if this widget has an ID.*
- YWidgetID \* **id** () const  
*Returns this widget's ID.*
- void **setId** (YWidgetID \*newId\_disown)  
*Set this widget's ID.*
- virtual void **setEnabled** (bool enabled=true)  
*Enable or disable this widget, i.e.*
- void **setDisabled** ()  
*Disable this widget (overloaded for better readability).*
- virtual bool **isEnabled** () const  
*Returns 'true' if this widget is enabled.*
- virtual bool **stretchable** (YUIDimension dim) const  
*This is a boolean value that determines whether the widget is resizable beyond its preferred size in the specified dimension.*
- void **setStretchable** (YUIDimension dim, bool newStretch)  
*Set the stretchable state to "newStretch" regardless of any `hstretch` or `vstretch` options.*
- void **setDefaultStretchable** (YUIDimension dim, bool newStretch)  
*Set the stretchable state to "newStretch".*
- virtual int **weight** (YUIDimension dim)  
*The weight is used in situations where all widgets can get their preferred size and yet space is available.*
- bool **hasWeight** (YUIDimension dim)  
*Return whether or not the widget has a weight in the specified dimension.*
- void **setWeight** (YUIDimension dim, int weight)  
*Set a weight in the specified dimension.*
- void **setNotify** (bool notify=true)



- Sets the Notify property.*

  - bool `notify` () const

*Returns whether the widget will notify, i.e.*
- void `setNotifyContextMenu` (bool `notifyContextMenu`=true)

*Sets the notifyContextMenu property.*
- bool `notifyContextMenu` () const

*Returns whether the widget will send an event when the user clicks selects the context menu e.g.*
- bool `sendKeyEvents` () const

*Returns 'true' if this widget should send key events, i.e.*
- void `setSendKeyEvents` (bool `doSend`)

*Specify whether or not this widget should send key events.*
- bool `autoShortcut` () const

*Returns 'true' if a keyboard shortcut should automatically be assigned to this widget - without complaints in the log file.*
- void `setAutoShortcut` (bool `_newAutoShortcut`)

*Sets the 'autoShortcut' flag.*
- int `functionKey` () const

*Return a function key number that is assigned to this widget.*
- bool `hasFunctionKey` () const

*Check if a function key is assigned to this widget.*
- virtual void `setFunctionKey` (int `fkey_no`)

*Assign a function key to this widget (1 for F1, 2 for F2, etc.*
- virtual bool `setKeyboardFocus` ()

*Set the keyboard focus to this widget.*
- virtual std::string `shortcutString` () const

*Get the string of this widget that holds the keyboard shortcut, if any.*
- virtual void `setShortcutString` (const std::string &`str`)

*Set the string of this widget that holds the keyboard shortcut, if any.*
- virtual const char \* `userInputProperty` ()

*The name of the widget property that will return user input, if there is any.*
- void `dumpWidgetTree` (int `indentationLevel`=0)

*Debugging function: Dump the widget tree from here on to the log file.*
- void `dumpDialogWidgetTree` ()

*Debugging function: Dump the widget tree from this widget's dialog parent.*
- void `setChildrenEnabled` (bool `enabled`)

*Enable or disable all widgets in this widget tree.*
- virtual void `saveUserInput` (YMacroRecorder \*`macroRecorder`)

*Recursively save the user input of all child widgets to a macro recorder:*
- void \* `operator new` (size\_t `size`)

*Overloaded operator new to ensure widgets are always created on the heap, never on the stack.*
- virtual void `startMultipleChanges` ()

*In some UIs updating the screen content is an expensive operation.*
- virtual void `doneMultipleChanges` ()

## Protected Member Functions

- [YWidget](#) ([YWidget](#) \*parent)  
*Constructor.*
- [YWidgetChildrenManager](#) \* childrenManager () const  
*Returns this widget's children manager.*
- void [setChildrenManager](#) ([YWidgetChildrenManager](#) \*manager)  
*Sets a new children manager for this widget.*
- void [setBeingDestroyed](#) ()  
*Set the "being destroyed" flag, i.e.*
- void [dumpWidget](#) ([YWidget](#) \*w, int indentationLevel)  
*Helper function for [dumpWidgetTree\(\)](#): Dump one widget to the log file.*

### 5.184.1 Detailed Description

Abstract base class of all UI widgets.

Definition at line 54 of file [YWidget.h](#).

### 5.184.2 Member Function Documentation

#### 5.184.2.1 addChild()

```
void YWidget::addChild (
    YWidget * child ) [virtual]
```

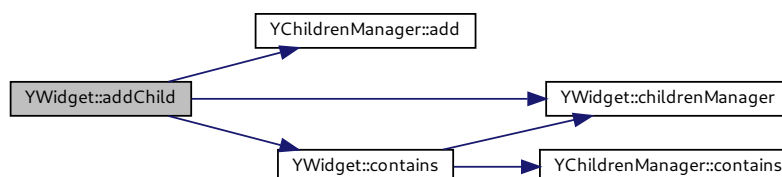
Add a new child.

This may throw exceptions if more children are added than this widget can handle.

Reimplemented in [YAlignment](#).

Definition at line 176 of file [YWidget.cc](#).

Here is the call graph for this function:



### 5.184.2.2 begin()

```
YWidgetListIterator YWidget::begin ( ) [inline]
```

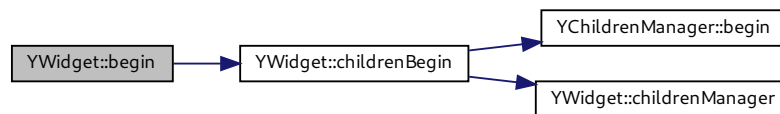
A helper for the range-based "for" loop.

#### Returns

Iterator pointing to the beginning of the children list

Definition at line 238 of file [YWidget.h](#).

Here is the call graph for this function:



### 5.184.2.3 debugLabel()

```
string YWidget::debugLabel ( ) const [virtual]
```

Returns a descriptive label of this widget instance.

This default implementation returns this widget's "shortcut property" (possibly truncated to avoid over-long texts) - the property that contains the keyboard shortcut used to activate this widget or to move the keyboard focus to it. In most cases this is this widget's label.

Note: This is usually translated to the user's target language. This makes this useful for debugging only.

Reimplemented in [YLabel](#), and [YDumbTab](#).

Definition at line 223 of file [YWidget.cc](#).

Here is the call graph for this function:



#### 5.184.2.4 dumpDialogWidgetTree()

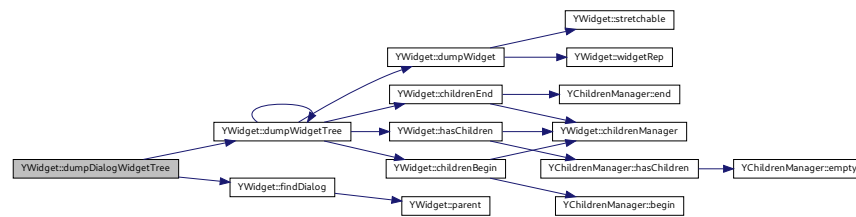
```
void YWidget::dumpDialogWidgetTree ( )
```

Debugging function: Dump the widget tree from this widget's dialog parent.

If there is no such dialog parent, dump the widget tree from here on.

Definition at line 663 of file [YWidget.cc](#).

Here is the call graph for this function:



#### 5.184.2.5 end()

```
YWidgetListIterator YWidget::end ( ) [inline]
```

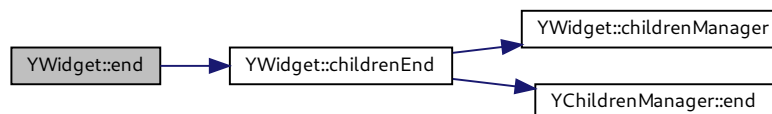
A helper for the range-based "for" loop.

##### Returns

Iterator pointing to the end of the children list

Definition at line 245 of file [YWidget.h](#).

Here is the call graph for this function:



### 5.184.2.6 findDialog()

```
YDialog * YWidget::findDialog ( )
```

Traverse up the widget hierarchy and find the dialog this widget belongs to.

Returns 0 if there is none.

Definition at line 376 of file [YWidget.cc](#).

Here is the call graph for this function:



### 5.184.2.7 findWidget()

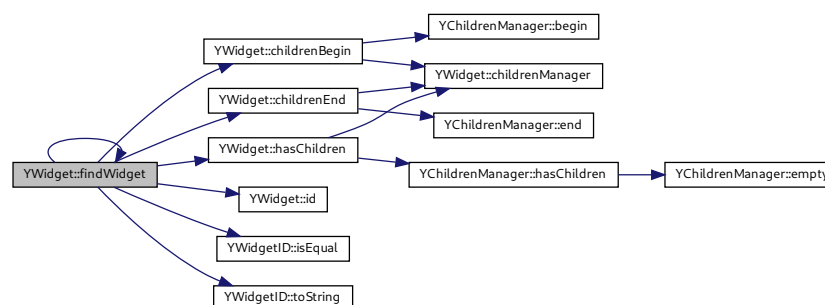
```
YWidget * YWidget::findWidget (
    YWidgetID * id,
    bool doThrow = true ) const
```

Recursively find a widget by its ID.

If there is no widget with that ID, this function throws a [YUIWidgetNotFoundException](#) if 'doThrow' is 'true'. It returns 0 if 'doThrow' is 'false'.

Definition at line 607 of file [YWidget.cc](#).

Here is the call graph for this function:



### 5.184.2.8 firstChild()

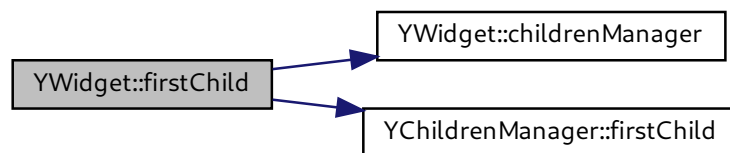
```
YWidget* YWidget::firstChild ( ) const [inline]
```

Returns the first child or 0 if there is none.

Useful mostly for children managers that handle only one child.

Definition at line 199 of file [YWidget.h](#).

Here is the call graph for this function:



### 5.184.2.9 functionKey()

```
int YWidget::functionKey ( ) const
```

Return a function key number that is assigned to this widget.

(1 for F1, 2 for F2, etc.; 0 for none)

Definition at line 324 of file [YWidget.cc](#).

### 5.184.2.10 getProperty()

```
YPropertyValue YWidget::getProperty (
    const std::string & propertyName ) [virtual]
```

Get a property.

Derived classes need to implement this.

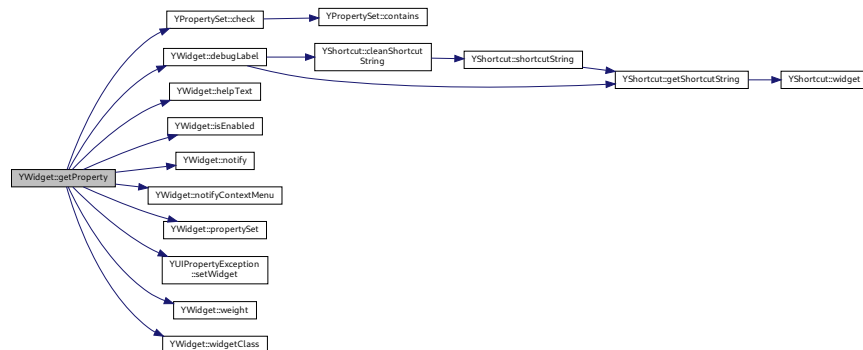
This method may throw exceptions, for example

- if there is no property with that name

Reimplemented in [YWizard](#), [YTree](#), [YTimezoneSelector](#), [YTable](#), [YSimpleInputField](#), [YSelectionBox](#), [YRichText](#), [YRadioButtonGroup](#), [YRadioButton](#), [YPushButton](#), [YProgressBar](#), [YPartitionSplitter](#), [YMultiSelectionBox](#), [YMultiProgressMeter](#), [YMultiLineEdit](#), [YMenuButton](#), [YLogView](#), [YLabel](#), [YItemSelector](#), [YIntField](#), [YInputField](#), [YGraph](#), [YFrame](#), [YDumbTab](#), [YDownloadProgress](#), [YContextMenu](#), [YComboBox](#), [YCheckBoxFrame](#), [YCheckBox](#), [YBusyIndicator](#), and [YBarGraph](#).

Definition at line 457 of file [YWidget.cc](#).

Here is the call graph for this function:



### 5.184.2.11 isValid()

```
bool YWidget::isValid ( ) const
```

Checks whether or not this object is valid.

This is to enable dangling pointer error checking (i.e. this object is already deallocated, but a pointer to it is still in use).

See also the `YUI_CHECK_WIDGET()` macro in [YUIException.h](#)

Definition at line 244 of file [YWidget.cc](#).

**5.184.2.12 notify()**

```
bool YWidget::notify ( ) const
```

Returns whether the widget will notify, i.e.

will case UserInput to return.

Definition at line [534](#) of file [YWidget.cc](#).

**5.184.2.13 notifyContextMenu()**

```
bool YWidget::notifyContextMenu ( ) const
```

Returns whether the widget will send an event when the user clicks selects the context menu e.g.

via right click.

Definition at line [540](#) of file [YWidget.cc](#).

**5.184.2.14 operator new()**

```
void * YWidget::operator new (
    size_t size )
```

Overloaded operator new to ensure widgets are always created on the heap, never on the stack.

Simpler implementations of this have a tendency to be fooled by poorly implemented derived classes.

Definition at line [130](#) of file [YWidget.cc](#).

**5.184.2.15 preferredHeight()**

```
virtual int YWidget::preferredHeight ( ) [pure virtual]
```

Preferred height of the widget.

Derived classes are required to implement this.

Implemented in [YSpacing](#), [YSingleChildContainerWidget](#), [YLayoutBox](#), [YEmpty](#), [YButtonBox](#), and [YAlignment](#).



#### 5.184.2.16 preferredSize()

```
int YWidget::preferredSize (
    YUIDimension dim ) [virtual]
```

Preferred size of the widget in the specified dimension.

This default implementation calls [preferredWidth\(\)](#) or [preferredHeight\(\)](#) which makes sense for most cases.

Derived classes can reimplement this, but this is discouraged.

Note: Even in that case, [preferredWidth\(\)](#) and [preferredHeight\(\)](#) need to be implemented, but they might then call [preferredSize\(\)](#).

Reimplemented in [YLayoutBox](#).

Definition at line 546 of file [YWidget.cc](#).

#### 5.184.2.17 preferredWidth()

```
virtual int YWidget::preferredWidth ( ) [pure virtual]
```

Preferred width of the widget.

Derived classes are required to implement this.

Implemented in [YSpacing](#), [YSingleChildContainerWidget](#), [YLayoutBox](#), [YEmpty](#), [YButtonBox](#), and [YAlignment](#).

#### 5.184.2.18 propertySet()

```
const YPropertySet & YWidget::propertySet ( ) [virtual]
```

Return this class's property set.

This also initializes the property upon the first call.

Derived classes should reimplement this.

Remember to add the base class's property set to your own in reimplemented versions, e.g.:

```

const YPropertySet &
MyWidgetClass::propertySet()
{
    static YPropertySet propSet;

    if ( propSet.isEmpty() )
    {
        // Add properties for the derived class
        propSet.add( YProperty( YUIProperty_Value, YStringProperty ) );
        propSet.add( YProperty( YUIProperty_Label, YStringProperty ) );

        // Add base class properties
        propSet.add( YWidget::propertySet() );
    }

    return propSet;
}

```

Otherwise the base class's properties will not be available in the derived class. It is also important that the base class's properties are added after those of the derived class so the derived class's properties have priority over those of the base class.

Reimplemented in [YWizard](#), [YTree](#), [YTimezoneSelector](#), [YTable](#), [YSimpleInputField](#), [YSelectionBox](#), [YRichText](#), [YRadioButtonGroup](#), [YRadioButton](#), [YPushButton](#), [YProgressBar](#), [YPartitionSplitter](#), [YMultiSelectionBox](#), [YMultiProgressMeter](#), [YMultiLineEdit](#), [YMenuButton](#), [YLogView](#), [YLabel](#), [YItemSelector](#), [YIntField](#), [YInputField](#), [YGraph](#), [YFrame](#), [YDumbTab](#), [YDownloadProgress](#), [YContextMenu](#), [YComboBox](#), [YCheckBoxFrame](#), [YCheckBox](#), [YBusyIndicator](#), and [YBarGraph](#).

Definition at line 402 of file [YWidget.cc](#).

#### 5.184.2.19 removeChild()

```

void YWidget::removeChild (
    YWidget * child ) [virtual]

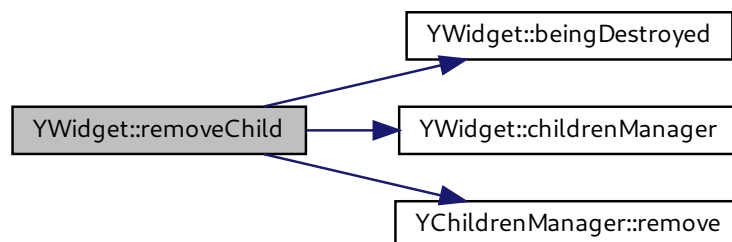
```

Remove a child.

This only removes the child from the children manager's list; it does not delete it.

Definition at line 191 of file [YWidget.cc](#).

Here is the call graph for this function:



**5.184.2.20 saveUserInput()**

```
void YWidget::saveUserInput (
    YMacroRecorder * macroRecorder ) [virtual]
```

Recursively save the user input of all child widgets to a macro recorder:

All child widgets that could contain data entered by the user are requested to send their contents to the macro recorder, e.g. input fields, check boxes etc.

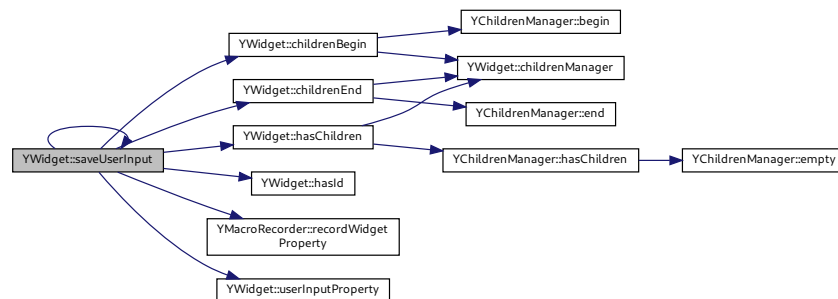
This default implementation records this widget's user input property (the property returned by `userInputProperty`) and then recursively calls `saveUserInput()` for all child widgets. This is suitable for most cases, for container widgets as well as for leaf widgets that have no or exactly one property that needs to be recorded.

Widgets that need another number of properties recorded should reimplement this method (and NOT call this default method in the new implementation).

Reimplemented in [YRadioButton](#), [YMultiSelectionBox](#), and [YInputField](#).

Definition at line 719 of file [YWidget.cc](#).

Here is the call graph for this function:

**5.184.2.21 sendKeyEvents()**

```
bool YWidget::sendKeyEvents ( ) const
```

Returns 'true' if this widget should send key events, i.e.

if it has `opt (keyEvent)` set.

Definition at line 300 of file [YWidget.cc](#).

#### 5.184.2.22 `setBeingDestroyed()`

```
void YWidget::setBeingDestroyed ( ) [protected]
```

Set the "being destroyed" flag, i.e.

indicate that this widget is in the process of being destroyed. The base class method already sets this, but sometimes it might be useful to call this in a derived class's destructor so certain optimizations work better.

This status intentionally cannot be reverted to "not being destroyed".

Definition at line [264](#) of file [YWidget.cc](#).

#### 5.184.2.23 `setChildrenManager()`

```
void YWidget::setChildrenManager (
    YWidgetChildrenManager * manager ) [protected]
```

Sets a new children manager for this widget.

The widget assumes ownership of this children manager and will delete it when appropriate.

The default children manager (a `YWidgetChildrenRejector`) rejects all children. This is useful for leaf widgets such as `PushButton`, `ComboBox` etc.

Derived classes that can handle children might want to set the children manager to a `YWidgetChildrenManager` (the base class that does not reject children) or to a `YSingleWidgetChildManager` (the class that handles exactly one child widget).

Definition at line [166](#) of file [YWidget.cc](#).

#### 5.184.2.24 `setDefaultStretchable()`

```
void YWidget::setDefaultStretchable (
    YUIDimension dim,
    bool newStretch )
```

Set the stretchable state to "newStretch".

`hstretch` or `rvstretch` options may override this.

Definition at line [566](#) of file [YWidget.cc](#).

#### 5.184.2.25 `setEnabled()`

```
void YWidget::setEnabled (
    bool enabled = true ) [virtual]
```

Enable or disable this widget, i.e.

make it accept or reject user input.

Derived classes should call the base class method to update the internal "enabled" flag.

Definition at line 500 of file [YWidget.cc](#).

#### 5.184.2.26 `setFunctionKey()`

```
void YWidget::setFunctionKey (
    int fkey_no ) [virtual]
```

Assign a function key to this widget (1 for F1, 2 for F2, etc.

; 0 for none)

Derived classes may want to overwrite this function, but they should call this base class function in the new function.

Reimplemented in [YPushButton](#).

Definition at line 336 of file [YWidget.cc](#).

#### 5.184.2.27 `setHelpText()`

```
void YWidget::setHelpText (
    const std::string & helpText )
```

Set a help text for this widget.

Currently, the UI does not do anything with this text but store it. Displaying the text at a convenient time is currently the application's responsibility. This may change in future versions.

Definition at line 348 of file [YWidget.cc](#).

Here is the call graph for this function:



#### 5.184.2.28 setId()

```
void YWidget::setId (
    YWidgetID * newId_disown )
```

Set this widget's ID.

The widget assumes ownership of this ID and will delete it when needed. (In the widget's destructor or when a new ID is set)

Widget IDs are purely for application use. C++ applications don't need to use them; they are much better off using widget pointers. For other languages, though, that can't use C++ pointers (e.g. Ruby) it makes sense to have widget IDs to identify widgets.

Definition at line 361 of file [YWidget.cc](#).

#### 5.184.2.29 setKeyboardFocus()

```
bool YWidget::setKeyboardFocus ( ) [virtual]
```

Set the keyboard focus to this widget.

The default implementation just emits a warning message. Overwrite this function for all widgets that can accept the keyboard focus.

This function returns true if the widget did accept the keyboard focus, and false if not.

Definition at line 599 of file [YWidget.cc](#).

#### 5.184.2.30 setProperty()

```
bool YWidget::setProperty (
    const std::string & propertyName,
    const YPropertyValue & val ) [virtual]
```

Set a property.

Derived classes need to implement this.

This method may throw exceptions, for example

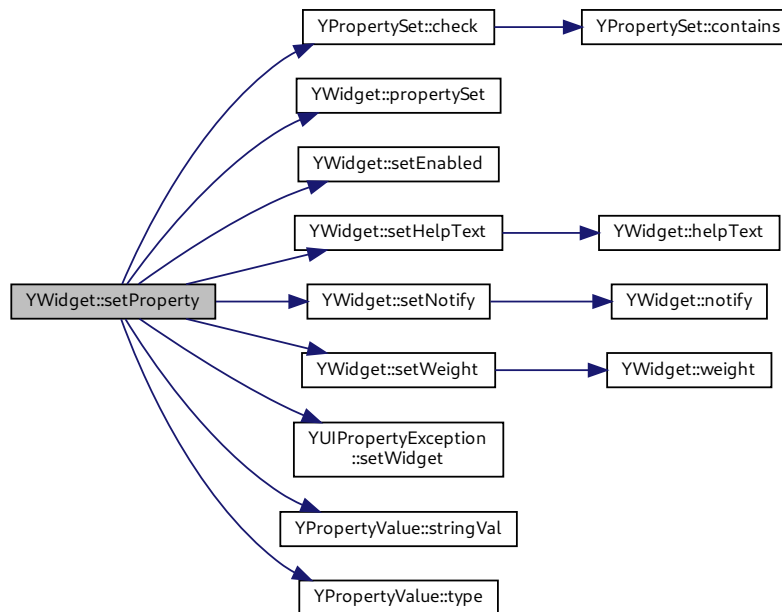
- if there is no property with that name
- if the expected type and the type mismatch
- if the value is out of range

This function returns 'true' if the value was successfully set and 'false' if that value requires special handling (not in error cases: those are covered by exceptions).

Reimplemented in [YTree](#), [YTimezoneSelector](#), [YTable](#), [YSimpleInputField](#), [YSelectionBox](#), [YRichText](#), [YRadioButtonGroup](#), [YRadioButton](#), [YPushButton](#), [YProgressBar](#), [YPartitionSplitter](#), [YMultiSelectionBox](#), [YMultiProgressMeter](#), [YMultiLineEdit](#), [YMenuButton](#), [YLogView](#), [YLabel](#), [YItemSelector](#), [YIntField](#), [YInputField](#), [YGraph](#), [YFrame](#), [YDumbTab](#), [YDownloadProgress](#), [YContextMenu](#), [YComboBox](#), [YCheckBoxFrame](#), [YCheckBox](#), [YBusyIndicator](#), and [YBarGraph](#).

Definition at line 432 of file [YWidget.cc](#).

Here is the call graph for this function:



### 5.184.2.31 setShortcutString()

```
void YWidget::setShortcutString (
    const std::string & str ) [virtual]
```

Set the string of this widget that holds the keyboard shortcut, if any.

Most widgets will call `setLabel()`.

Overwrite this for widgets that can have keyboard shortcuts.

Reimplemented in [YSimpleInputField](#), [YSelectionWidget](#), [YRadioButton](#), [YPushButton](#), [YMultiLineEdit](#), [YLogView](#), [YIntField](#), [YInputField](#), [YDumbTab](#), [YCheckBoxFrame](#), and [YCheckBox](#).

Definition at line 513 of file [YWidget.cc](#).

Here is the call graph for this function:



#### 5.184.2.32 `setSize()`

```
virtual void YWidget::setSize (
    int newWidth,
    int newHeight ) [pure virtual]
```

Set the new size of the widget.

Layout manager widgets (like [YLayoutBox](#)) call this during geometry management after all widgets are queried about their preferred widths and heights. Depending on layout constraints, widgets might be resized beyond or below their preferred size.

The sizes passed here are not meant to affect any future [preferredWidth\(\)](#) or [preferredHeight\(\)](#) calls; they are just the outcome of all kinds of compromises (too little screen space or too much) for the current geometry management calculation.

Derived classes are required to implement this function.

Implemented in [YSingleChildContainerWidget](#), [YLayoutBox](#), [YButtonBox](#), and [YAlignment](#).

#### 5.184.2.33 `setWidgetRep()`

```
void YWidget::setWidgetRep (
    void * toolkitWidgetRep )
```

Set the pointer to the underlying toolkit's (Qt, ...) widget representing this abstract UI widget.

This pointer might be useful for derived UIs to store a counterpart of the toolkit widget in each [YWidget](#). The abstract UI does not need that, though; this is purely for the convenience of derived UIs. All the abstract UI ever does with that pointer is store it.

Definition at line 493 of file [YWidget.cc](#).



#### 5.184.2.34 shortcutString()

```
virtual std::string YWidget::shortcutString ( ) const [inline], [virtual]
```

Get the string of this widget that holds the keyboard shortcut, if any.

Most widgets will return label().

Overwrite this for widgets that can have keyboard shortcuts.

Reimplemented in [YSimpleInputField](#), [YSelectionWidget](#), [YRadioButton](#), [YPushButton](#), [YMultiLineEdit](#), [YLogView](#), [YIntField](#), [YInputField](#), [YDumbTab](#), [YCheckBoxFrame](#), and [YCheckBox](#).

Definition at line 560 of file [YWidget.h](#).

#### 5.184.2.35 startMultipleChanges()

```
virtual void YWidget::startMultipleChanges ( ) [inline], [virtual]
```

In some UIs updating the screen content is an expensive operation.

Use [startMultipleChanges\(\)](#) to tell the ui that you're going to perform multiple changes to the widget. The UI may delay any screen updates until [doneMultipleChanges\(\)](#) is called.

Definition at line 640 of file [YWidget.h](#).

#### 5.184.2.36 stretchable()

```
bool YWidget::stretchable (
    YUIDimension dim ) const [virtual]
```

This is a boolean value that determines whether the widget is resizable beyond its preferred size in the specified dimension.

A selection box is stretchable in both dimensions, a push button is not stretchable by default, a frame is stretchable if its contents are stretchable. Most widgets accept a `hstretch` or `vrstretch` option to become stretchable even when by default they are not.

Reimplemented in [YLayoutBox](#), [YButtonBox](#), [YSquash](#), [YSingleChildContainerWidget](#), [YDumbTab](#), and [YAlignment](#).

Definition at line 572 of file [YWidget.cc](#).

### 5.184.2.37 userInputProperty()

```
virtual const char* YWidget::userInputProperty ( ) [inline], [virtual]
```

The name of the widget property that will return user input, if there is any.

Widgets that do have user input (such as [InputField](#), [ComboBox](#), [SelBox](#)) should overwrite this methods. Widgets that are purely passive (such as [Label](#), [RichText](#)) should not.

Reimplemented in [YTree](#), [YTable](#), [YSimpleInputField](#), [YSelectionBox](#), [YRadioButton](#), [YPartitionSplitter](#), [YMultiSelectionBox](#), [YMultiLineEdit](#), [YItemSelector](#), [YIntField](#), [YInputField](#), [YComboBox](#), [YCheckBoxFrame](#), and [YCheckBox](#).

Definition at line 576 of file [YWidget.h](#).

### 5.184.2.38 weight()

```
int YWidget::weight (
    YUIDimension dim ) [virtual]
```

The weight is used in situations where all widgets can get their preferred size and yet space is available.

The remaining space will be devided between all stretchable widgets according to their weights. A widget with greater weight will get more space. The default weight for all widgets is 0.

Derived classes can overwrite this function, but they should call this base class function in the new function.

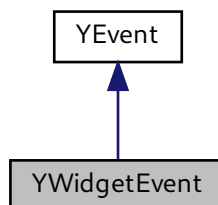
Definition at line 578 of file [YWidget.cc](#).

The documentation for this class was generated from the following files:

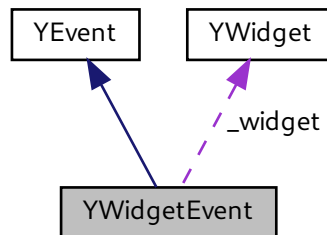
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YWidget.h](#)
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YWidget.cc](#)

## 5.185 YWidgetEvent Class Reference

Inheritance diagram for YWidgetEvent:



Collaboration diagram for YWidgetEvent:



## Public Member Functions

- [YWidgetEvent](#) ([YWidget](#) \*[widget](#)=0, [EventReason](#) [reason](#)=Activated, [EventType](#) [eventType](#)=WidgetEvent)  
*Constructor.*
- virtual [YWidget](#) \* [widget](#) () const  
*Returns the widget that caused this event.*
- [EventReason](#) [reason](#) () const  
*Returns the reason for this event.*

## Protected Member Functions

- virtual [~YWidgetEvent](#) ()  
*Protected destructor - events can only be deleted via [YDialog::deleteEvent\(\)](#).*

## Protected Attributes

- [YWidget](#) \* [\\_widget](#)
- [EventReason](#) [\\_reason](#)

## Additional Inherited Members

### 5.185.1 Detailed Description

Definition at line 165 of file [YEvent.h](#).

### 5.185.2 Constructor & Destructor Documentation

### 5.185.2.1 ~YWidgetEvent()

```
virtual YWidgetEvent::~YWidgetEvent ( ) [inline], [protected], [virtual]
```

Protected destructor - events can only be deleted via [YDialog::deleteEvent\(\)](#).

The associated dialog will take care of this event and delete it when appropriate.

Definition at line [194](#) of file [YEvent.h](#).

## 5.185.3 Member Function Documentation

### 5.185.3.1 reason()

```
EventReason YWidgetEvent::reason ( ) const [inline]
```

Returns the reason for this event.

This very much like an event sub-type.

Definition at line [185](#) of file [YEvent.h](#).

### 5.185.3.2 widget()

```
virtual YWidget* YWidgetEvent::widget ( ) const [inline], [virtual]
```

Returns the widget that caused this event.

Reimplemented from [YEvent](#).

Reimplemented from [YEvent](#).

Definition at line [180](#) of file [YEvent.h](#).

The documentation for this class was generated from the following files:

- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YEvent.h](#)
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YEvent.cc](#)

## 5.186 YWidgetFactory Class Reference

Abstract widget factory for mandatory widgets.

```
#include <YWidgetFactory.h>
```

### Public Member Functions

- [YDialog](#) \* **createMainDialog** (YDialogColorMode colorMode=YDialogNormalColor)
- [YDialog](#) \* **createPopupDialog** (YDialogColorMode colorMode=YDialogNormalColor)
- virtual [YDialog](#) \* **createDialog** (YDialogType dialogType, YDialogColorMode colorMode=YDialogNormalColor)=0
- [YLayoutBox](#) \* **createVBox** (YWidget \*parent)
- [YLayoutBox](#) \* **createHBox** (YWidget \*parent)
- virtual [YLayoutBox](#) \* **createLayoutBox** (YWidget \*parent, YUIDimension dimension)=0
- virtual [YButtonBox](#) \* **createButtonBox** (YWidget \*parent)=0
- virtual [YPushButton](#) \* **createPushButton** (YWidget \*parent, const std::string &label)=0
- virtual [YLabel](#) \* **createLabel** (YWidget \*parent, const std::string &text, bool isHeading=false, bool isOutputField=false)=0
- [YLabel](#) \* **createHeading** (YWidget \*parent, const std::string &label)
- virtual [YInputField](#) \* **createInputField** (YWidget \*parent, const std::string &label, bool passwordMode=false)=0
- virtual [YCheckBox](#) \* **createCheckBox** (YWidget \*parent, const std::string &label, bool isChecked=false)=0
- virtual [YRadioButton](#) \* **createRadioButton** (YWidget \*parent, const std::string &label, bool isChecked=false)=0
- virtual [YComboBox](#) \* **createComboBox** (YWidget \*parent, const std::string &label, bool editable=false)=0
- virtual [YSelectionBox](#) \* **createSelectionBox** (YWidget \*parent, const std::string &label)=0
- virtual [YTree](#) \* **createTree** (YWidget \*parent, const std::string &label, bool multiselection=false, bool recursive-selection=false)=0
- virtual [YTable](#) \* **createTable** (YWidget \*parent, YTableHeader \*header\_disown, bool multiSelection=false)=0
- virtual [YProgressBar](#) \* **createProgressBar** (YWidget \*parent, const std::string &label, int maxValue=100)=0
- virtual [YRichText](#) \* **createRichText** (YWidget \*parent, const std::string &text=std::string(), bool plainTextMode=false)=0
- virtual [YBusyIndicator](#) \* **createBusyIndicator** (YWidget \*parent, const std::string &label, int timeout=1000)=0
- [YPushButton](#) \* **createIconButton** (YWidget \*parent, const std::string &iconName, const std::string &fallbackTextLabel)
- [YLabel](#) \* **createOutputField** (YWidget \*parent, const std::string &label)
- virtual [YIntField](#) \* **createIntField** (YWidget \*parent, const std::string &label, int minVal, int maxVal, int initialValue)=0
- [YInputField](#) \* **createPasswordField** (YWidget \*parent, const std::string &label)
- virtual [YMenuButton](#) \* **createMenuButton** (YWidget \*parent, const std::string &label)=0
- virtual [YMultiLineEdit](#) \* **createMultiLineEdit** (YWidget \*parent, const std::string &label)=0
- virtual [YImage](#) \* **createImage** (YWidget \*parent, const std::string &imageFileName, bool animated=false)=0
- virtual [YLogView](#) \* **createLogView** (YWidget \*parent, const std::string &label, int visibleLines, int storedLines)=0
- virtual [YMultiSelectionBox](#) \* **createMultiSelectionBox** (YWidget \*parent, const std::string &label)=0
- virtual [YPackageSelector](#) \* **createPackageSelector** (YWidget \*parent, long ModeFlags=0)=0
- virtual [YWidget](#) \* **createPkgSpecial** (YWidget \*parent, const std::string &subwidgetName)=0
- [YSpacing](#) \* **createHStretch** (YWidget \*parent)
- [YSpacing](#) \* **createVStretch** (YWidget \*parent)
- [YSpacing](#) \* **createHSpacing** (YWidget \*parent, YLayoutSize\_t size=1.0)
- [YSpacing](#) \* **createVSpacing** (YWidget \*parent, YLayoutSize\_t size=1.0)

- virtual [YSpacing](#) \* **createSpacing** ([YWidget](#) \*parent, YUIDimension dim, bool stretchable=false, YLayoutSize\_t size=0.0)=0
- virtual [YEmpty](#) \* **createEmpty** ([YWidget](#) \*parent)=0
- [YAlignment](#) \* **createLeft** ([YWidget](#) \*parent)
- [YAlignment](#) \* **createRight** ([YWidget](#) \*parent)
- [YAlignment](#) \* **createTop** ([YWidget](#) \*parent)
- [YAlignment](#) \* **createBottom** ([YWidget](#) \*parent)
- [YAlignment](#) \* **createHCenter** ([YWidget](#) \*parent)
- [YAlignment](#) \* **createVCenter** ([YWidget](#) \*parent)
- [YAlignment](#) \* **createHVCenter** ([YWidget](#) \*parent)
- [YAlignment](#) \* **createMarginBox** ([YWidget](#) \*parent, YLayoutSize\_t horMargin, YLayoutSize\_t vertMargin)
- [YAlignment](#) \* **createMarginBox** ([YWidget](#) \*parent, YLayoutSize\_t leftMargin, YLayoutSize\_t rightMargin, YLayoutSize\_t topMargin, YLayoutSize\_t bottomMargin)
- [YAlignment](#) \* **createMinWidth** ([YWidget](#) \*parent, YLayoutSize\_t minWidth)
- [YAlignment](#) \* **createMinHeight** ([YWidget](#) \*parent, YLayoutSize\_t minHeight)
- [YAlignment](#) \* **createMinSize** ([YWidget](#) \*parent, YLayoutSize\_t minWidth, YLayoutSize\_t minHeight)
- virtual [YAlignment](#) \* **createAlignment** ([YWidget](#) \*parent, YAlignmentType horAlignment, YAlignmentType vert←Alignment)=0
- [YSquash](#) \* **createHSquash** ([YWidget](#) \*parent)
- [YSquash](#) \* **createVSquash** ([YWidget](#) \*parent)
- [YSquash](#) \* **createHVSquash** ([YWidget](#) \*parent)
- virtual [YSquash](#) \* **createSquash** ([YWidget](#) \*parent, bool horSquash, bool vertSquash)=0
- virtual [YFrame](#) \* **createFrame** ([YWidget](#) \*parent, const std::string &label)=0
- virtual [YCheckBoxFrame](#) \* **createCheckBoxFrame** ([YWidget](#) \*parent, const std::string &label, bool checked)=0
- virtual [YRadioButtonGroup](#) \* **createRadioButtonGroup** ([YWidget](#) \*parent)=0
- virtual [YReplacePoint](#) \* **createReplacePoint** ([YWidget](#) \*parent)=0
- virtual [YItemSelector](#) \* **createItemSelector** ([YWidget](#) \*parent, bool enforceSingleSelection=true)
- [YItemSelector](#) \* **createSingleItemSelector** ([YWidget](#) \*parent)
- [YItemSelector](#) \* **createMultiItemSelector** ([YWidget](#) \*parent)
- virtual [YItemSelector](#) \* **createCustomStatusItemSelector** ([YWidget](#) \*parent, const YItemCustomStatusVector &customStates)

## Protected Member Functions

- [YWidgetFactory](#) ()  
*Constructor.*
- virtual [~YWidgetFactory](#) ()  
*Destructor.*

### 5.186.1 Detailed Description

Abstract widget factory for mandatory widgets.

Use [YOptionalWidgetFactory](#) for optional ("special") widgets.

Refer to the respective widget's documentation (in the header file) for documentation about the function parameters.

Definition at line 78 of file [YWidgetFactory.h](#).

## 5.186.2 Constructor & Destructor Documentation

### 5.186.2.1 YWidgetFactory()

```
YWidgetFactory::YWidgetFactory ( ) [protected]
```

Constructor.

Use [YUI::widgetFactory\(\)](#) to get the singleton for this class.

Definition at line 37 of file [YWidgetFactory.cc](#).

The documentation for this class was generated from the following files:

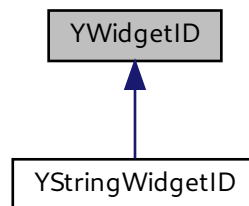
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YWidgetFactory.h](#)
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YWidgetFactory.cc](#)

## 5.187 YWidgetID Class Reference

Abstract base class for widget IDs.

```
#include <YWidgetID.h>
```

Inheritance diagram for YWidgetID:



### Public Member Functions

- virtual [~YWidgetID](#) ()  
*Destructor.*
- virtual bool [isEqual](#) ([YWidgetID](#) \*otherID) const =0  
*Check if this ID is equal to another.*
- virtual std::string [toString](#) () const =0  
*Convert the ID value to string.*

## Protected Member Functions

- [YWidgetID](#) ()  
*Constructor.*

### 5.187.1 Detailed Description

Abstract base class for widget IDs.

Definition at line 36 of file [YWidgetID.h](#).

### 5.187.2 Constructor & Destructor Documentation

#### 5.187.2.1 YWidgetID()

```
YWidgetID::YWidgetID ( ) [inline], [protected]
```

Constructor.

Protected since this is an abstract base class.

Definition at line 42 of file [YWidgetID.h](#).

### 5.187.3 Member Function Documentation

#### 5.187.3.1 toString()

```
virtual std::string YWidgetID::toString ( ) const [pure virtual]
```

Convert the ID value to string.

Used for logging and debugging.

Implemented in [YStringWidgetID](#).

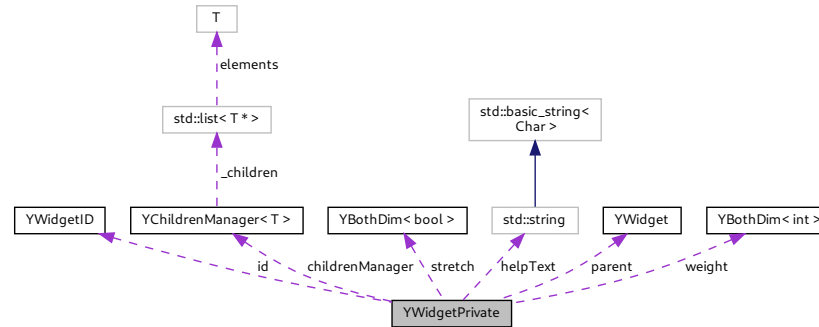
The documentation for this class was generated from the following file:

- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YWidgetID.h](#)



## 5.188 YWidgetPrivate Struct Reference

Collaboration diagram for YWidgetPrivate:



### Public Member Functions

- `YWidgetPrivate` (`YWidgetChildrenManager` \*manager, `YWidget` \*parentWidget=0)  
*Constructor.*

### Public Attributes

- `YWidgetChildrenManager` \* **childrenManager**
- `YWidget` \* **parent**
- bool **beingDestroyed**
- bool **enabled**
- bool **notify**
- bool **notifyContextMenu**
- bool **sendKeyEvents**
- bool **autoShortcut**
- void \* **toolkitWidgetRep**
- `YWidgetID` \* **id**
- `YBothDim< bool >` **stretch**
- `YBothDim< int >` **weight**
- int **functionKey**
- string **helpText**

### 5.188.1 Detailed Description

Definition at line 56 of file `YWidget.cc`.

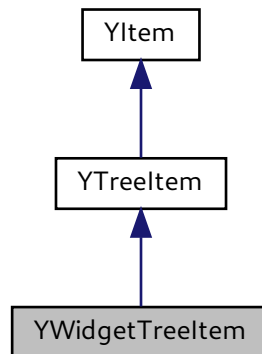
The documentation for this struct was generated from the following file:

- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YWidget.cc`

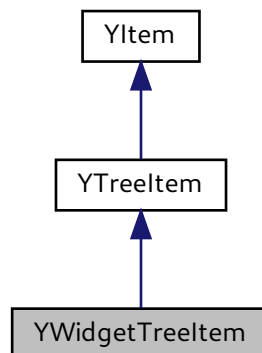
## 5.189 YWidgetTreeltem Class Reference

Custom tree item class to map tree items to widgets.

Inheritance diagram for YWidgetTreeltem:



Collaboration diagram for YWidgetTreeltem:



### Public Member Functions

- **YWidgetTreeltem** ([YWidget](#) \*widget, bool [isOpen](#))
- **YWidgetTreeltem** ([YWidgetTreeltem](#) \*parent, [YWidget](#) \*widget, bool [isOpen](#))
- [YWidget](#) \* **widget** () const

## Protected Member Functions

- void `setWidgetLabel()`

### 5.189.1 Detailed Description

Custom tree item class to map tree items to widgets.

Definition at line 86 of file [YDialogSpy.cc](#).

The documentation for this class was generated from the following file:

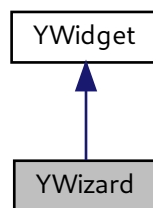
- `/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YDialogSpy.cc`

## 5.190 YWizard Class Reference

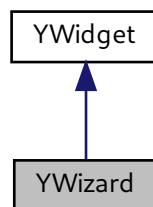
A wizard is a more complex frame typically used for multi-step workflows:

```
#include <YWizard.h>
```

Inheritance diagram for YWizard:



Collaboration diagram for YWizard:



## Public Member Functions

- virtual [~YWizard](#) ()  
*Destructor.*
- virtual const char \* [widgetClass](#) () const  
*Returns a descriptive name of this widget class for logging, debugging etc.*
- [YWizardMode](#) [wizardMode](#) () const  
*Return the wizard mode (what kind of wizard this is): YWizardMode\_Standard, YWizardMode\_Steps, YWizardMode\_Tree, YWizardMode\_TitleOnLeft.*
- virtual [YPushButton](#) \* [backButton](#) () const =0  
*Return the wizard buttons or 0 if there is no such button.*
- virtual [YPushButton](#) \* [abortButton](#) () const =0
- virtual [YPushButton](#) \* [nextButton](#) () const =0
- virtual [YReplacePoint](#) \* [contentsReplacePoint](#) () const =0  
*Return the internal contents ReplacePoint.*
- void [protectNextButton](#) (bool protect)  
*Protect the wizard's "Next" button against disabling.*
- bool [nextButtonIsProtected](#) () const  
*Check if the wizard's "Next" button is currently protected against disabling.*
- virtual void [setButtonLabel](#) ([YPushButton](#) \*button, const std::string &newLabel)  
*Set the label of one of the wizard buttons ([backButton\(\)](#), [abortButton\(\)](#), [nextButton\(\)](#) ) if that button is non-null.*
- virtual void [setHelpText](#) (const std::string &[helpText](#))=0  
*Set the help text.*
- virtual void [setDialogIcon](#) (const std::string &iconName)=0  
*Set the dialog icon.*
- virtual void [setDialogTitle](#) (const std::string &titleText)=0  
*Set the dialog title shown in the window manager's title bar.*
- virtual std::string [getDialogTitle](#) ()=0  
*Get the current dialog title shown in the window manager's title bar.*
- virtual void [setDialogHeading](#) (const std::string &headingText)=0  
*Set the dialog heading.*
- virtual std::string [getDialogHeading](#) ()=0  
*Get the dialog heading.*
- virtual void [addStep](#) (const std::string &text, const std::string &[id](#))=0  
*Add a step for the steps panel on the side bar.*
- virtual void [addStepHeading](#) (const std::string &text)=0  
*Add a step heading for the steps panel on the side bar.*
- virtual void [deleteSteps](#) ()=0  
*Delete all steps and step headings from the internal lists.*
- virtual void [setCurrentStep](#) (const std::string &[id](#))=0  
*Set the current step.*
- virtual void [updateSteps](#) ()=0  
*Update the steps display: Reflect the internal steps and heading lists in the layout.*
- virtual void [addTreeItem](#) (const std::string &parentID, const std::string &text, const std::string &[id](#))=0  
*Add a tree item.*
- virtual void [selectTreeItem](#) (const std::string &[id](#))=0  
*Select the tree item with the specified ID, if such an item exists.*
- virtual std::string [currentTreeSelection](#) ()=0

*Returns the current tree selection or an empty string if nothing is selected or there is no tree.*

- virtual void `deleteTreeItems` ()=0  
*Delete all tree items.*
- virtual void `addMenu` (const std::string &text, const std::string &id)=0  
*Add a menu to the menu bar.*
- virtual void `addSubMenu` (const std::string &parentMenuID, const std::string &text, const std::string &id)=0  
*Add a submenu to the menu with ID 'parentMenuID'.*
- virtual void `addMenuEntry` (const std::string &parentMenuID, const std::string &text, const std::string &id)=0  
*Add a menu entry to the menu with ID 'parentMenuID'.*
- virtual void `addMenuSeparator` (const std::string &parentMenuID)=0  
*Add a menu separator to a menu.*
- virtual void `deleteMenus` ()=0  
*Delete all menus and hide the menu bar.*
- virtual void `showReleaseNotesButton` (const std::string &label, const std::string &id)=0  
*Show a "Release Notes" button above the "Help" button in the steps panel with the specified label that will return the specified id to UI::UserInput() when clicked.*
- virtual void `hideReleaseNotesButton` ()=0  
*Hide an existing "Release Notes" button.*
- virtual void `retranslateInternalButtons` ()=0  
*Retranslate internal buttons that are not accessible from the outside:*
- void `ping` ()  
*NOP command to check if a YWizard is running.*
- virtual `YPropertyValue` `getProperty` (const std::string &propertyName)  
*Get a property.*
- virtual const `YPropertySet` & `propertySet` ()  
*Return this class's property set.*

## Protected Member Functions

- `YWizard` (`YWidget` \*parent, const std::string &backButtonLabel, const std::string &abortButtonLabel, const std::string &nextButtonLabel, `YWizardMode` wizardMode=`YWizardMode_Standard`)  
*Constructor.*

### 5.190.1 Detailed Description

A wizard is a more complex frame typically used for multi-step workflows:

+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+

Side bar	Content Area

	(YReplacePoint)
	[Back] [Abort] [Next]

+-----+-----+-----+

The side bar can contain help text, a list of steps that are performed, or an embedded tree (much like the [YTree](#) widget).

The client application creates the wizard and replaces the widget in the content area for each step.

The wizard buttons can theoretically be used to do anything, but good UI design will stick to the model above: [Back], [Abort], [Next].

If only two buttons are desired, leave the [Back] button's label empty. The remaining two buttons will be rearranged accordingly in the button area.

In the last step of a multi-step workflow, the [Next] button's label is customarily replaced with a label that indicates that this is the last step. [Accept] is recommended for that button label: [Finish] (as sometimes used in other environments) by no means clearly indicates that this is the positive ending, the final "do it" button. Worse, translations of that are often downright miserable: To German, [Finish] gets translated as [Beenden] which is the same word as "Quit" (used in menus). This does not at all tell the user that that button really performs the requested action the multi-step wizard is all about.

Definition at line 96 of file [YWizard.h](#).

## 5.190.2 Constructor & Destructor Documentation

### 5.190.2.1 YWizard()

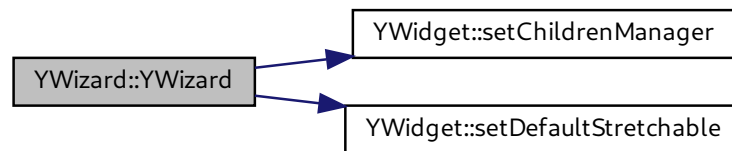
```
YWizard::YWizard (
    YWidget * parent,
    const std::string & backButtonLabel,
    const std::string & abortButtonLabel,
    const std::string & nextButtonLabel,
    YWizardMode wizardMode = YWizardMode_Standard ) [protected]
```

Constructor.

If only two buttons are desired, leave 'backButtonLabel' empty.

Definition at line 49 of file [YWizard.cc](#).

Here is the call graph for this function:



### 5.190.3 Member Function Documentation

#### 5.190.3.1 addMenu()

```
virtual void YWizard::addMenu (
    const std::string & text,
    const std::string & id ) [pure virtual]
```

Add a menu to the menu bar.

If the menu bar is not visible yet, it will be made visible. 'text' is the user-visible text for the menu bar (including keyboard shortcuts marked with '&'), 'id' is the menu ID for later [addMenuEntry\(\)](#) etc. calls.

#### 5.190.3.2 addMenuEntry()

```
virtual void YWizard::addMenuEntry (
    const std::string & parentMenuID,
    const std::string & text,
    const std::string & id ) [pure virtual]
```

Add a menu entry to the menu with ID 'parentMenuID'.

'id' is what will be returned by `UI::UserInput()` etc. when a user activates this menu entry.

#### 5.190.3.3 addStep()

```
virtual void YWizard::addStep (
    const std::string & text,
    const std::string & id ) [pure virtual]
```

Add a step for the steps panel on the side bar.

This only adds the step to the internal list of steps. The display is only updated upon calling [updateSteps\(\)](#).

#### 5.190.3.4 addStepHeading()

```
virtual void YWizard::addStepHeading (
    const std::string & text ) [pure virtual]
```

Add a step heading for the steps panel on the side bar.

This only adds the heading to the internal list of steps. The display is only updated upon calling [updateSteps\(\)](#).

#### 5.190.3.5 addTreeItem()

```
virtual void YWizard::addTreeItem (
    const std::string & parentID,
    const std::string & text,
    const std::string & id ) [pure virtual]
```

Add a tree item.

If "parentID" is an empty string, it will be a root item. 'text' is the text that will be displayed in the tree, 'id' the ID with which this newly created item can be referenced - and that will be returned when the user clicks on a tree item.

#### 5.190.3.6 backButton()

```
virtual YPushButton* YWizard::backButton ( ) const [pure virtual]
```

Return the wizard buttons or 0 if there is no such button.

Derived classes are required to implement this.

#### 5.190.3.7 contentsReplacePoint()

```
virtual YReplacePoint* YWizard::contentsReplacePoint ( ) const [pure virtual]
```

Return the internal contents ReplacePoint.

Derived classes are required to implement this.

#### 5.190.3.8 deleteSteps()

```
virtual void YWizard::deleteSteps ( ) [pure virtual]
```

Delete all steps and step headings from the internal lists.

The display is only updated upon calling [updateSteps\(\)](#).



### 5.190.3.9 getProperty()

```
YPropertyValue YWizard::getProperty (
    const std::string & propertyName ) [virtual]
```

Get a property.

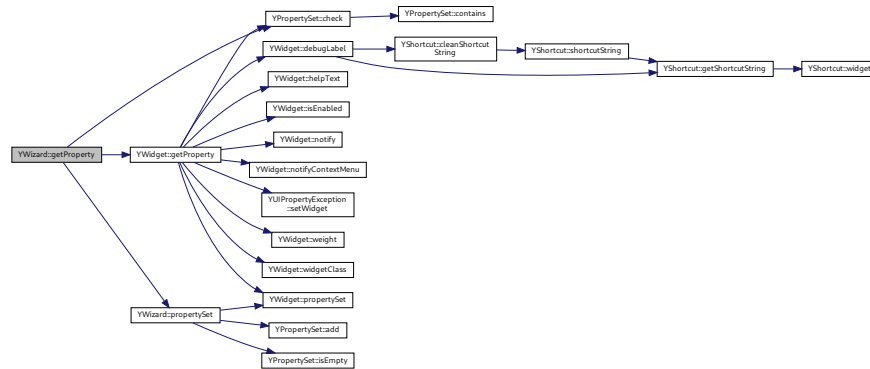
Reimplemented from [YWidget](#).

This method may throw YUIPropertyExceptions.

Reimplemented from [YWidget](#).

Definition at line 133 of file [YWizard.cc](#).

Here is the call graph for this function:



### 5.190.3.10 propertySet()

```
const YPropertySet & YWizard::propertySet ( ) [virtual]
```

Return this class's property set.

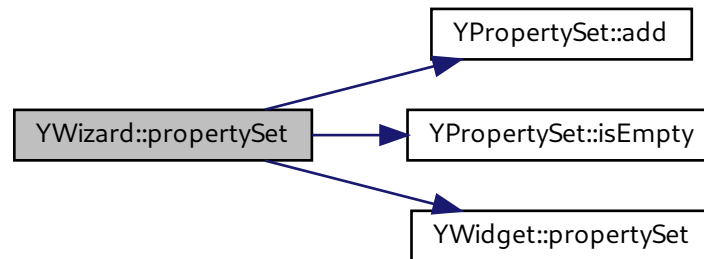
This also initializes the property upon the first call.

Reimplemented from [YWidget](#).

Reimplemented from [YWidget](#).

Definition at line 115 of file [YWizard.cc](#).

Here is the call graph for this function:



#### 5.190.3.11 retranslateInternalButtons()

```
virtual void YWizard::retranslateInternalButtons ( ) [pure virtual]
```

Retranslate internal buttons that are not accessible from the outside:

- [Help]
- [Steps]
- [Tree]

#### 5.190.3.12 setButtonLabel()

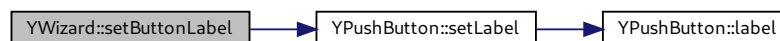
```
void YWizard::setButtonLabel (
    YPushButton * button,
    const std::string & newLabel ) [virtual]
```

Set the label of one of the wizard buttons (`backButton()`, `abortButton()`, `nextButton()` ) if that button is non-null.

The default implementation simply calls `button->setLabel( newLabel )`.

Definition at line 96 of file [YWizard.cc](#).

Here is the call graph for this function:



### 5.190.3.13 setCurrentStep()

```
virtual void YWizard::setCurrentStep (
    const std::string & id ) [pure virtual]
```

Set the current step.

This also triggers [updateSteps\(\)](#) if necessary.

### 5.190.3.14 setDialogIcon()

```
virtual void YWizard::setDialogIcon (
    const std::string & iconName ) [pure virtual]
```

Set the dialog icon.

An empty icon name clears the current icon.

### 5.190.3.15 setDialogTitle()

```
virtual void YWizard::setDialogTitle (
    const std::string & titleText ) [pure virtual]
```

Set the dialog title shown in the window manager's title bar.

An empty string clears the current title.

The documentation for this class was generated from the following files:

- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YWizard.h](#)
- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YWizard.cc](#)

## 5.191 YWizardPrivate Struct Reference

### Public Member Functions

- **YWizardPrivate** ([YWizardMode](#) wizardMode)

### Public Attributes

- [YWizardMode](#) **wizardMode**
- bool **nextButtonIsProtected**

### 5.191.1 Detailed Description

Definition at line [35](#) of file [YWizard.cc](#).

The documentation for this struct was generated from the following file:

- [/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YWizard.cc](#)



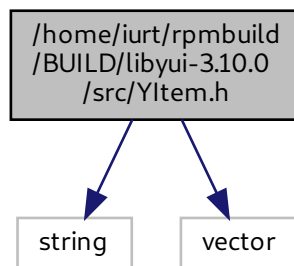
## Chapter 6

# File Documentation

### 6.1 /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YItem.h File Reference

```
#include <string>
#include <vector>
```

Include dependency graph for YItem.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [YItem](#)

*Simple item class for `SelectionBox`, `ComboBox`, `MultiSelectionBox` etc.*

## Typedefs

- typedef std::vector< YItem \* > YItemCollection  
*Collection of pointers to YItem.*
- typedef YItemCollection::iterator YItemIterator  
*Mutable iterator over YItemCollection.*
- typedef YItemCollection::const\_iterator YItemConstIterator  
*Const iterator over YItemCollection.*

## 6.2 YItem.h

```

00001 /*
00002 Copyright (C) 2000-2012 Novell, Inc
00003 This library is free software; you can redistribute it and/or modify
00004 it under the terms of the GNU Lesser General Public License as
00005 published by the Free Software Foundation; either version 2.1 of the
00006 License, or (at your option) version 3.0 of the License. This library
00007 is distributed in the hope that it will be useful, but WITHOUT ANY
00008 WARRANTY; without even the implied warranty of MERCHANTABILITY or
00009 FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public
00010 License for more details. You should have received a copy of the GNU
00011 Lesser General Public License along with this library; if not, write
00012 to the Free Software Foundation, Inc., 51 Franklin Street, Fifth
00013 Floor, Boston, MA 02110-1301 USA
00014 */
00015
00016
00017 /*-
00018
00019 File:      YItem.h
00020
00021 Author:     Stefan Hundhammer <sh@suse.de>
00022
00023 -*/
00024
00025 #ifndef YItem_h
00026 #define YItem_h
00027
00028 #include <string>
00029 #include <vector>
00030
00031
00032 class YItem;
00033
00034 // without "documenting" the file, typedefs will be dropped
00035 /*! @file
00036
00037 /*! Collection of pointers to YItem.
00038 typedef std::vector<YItem *> YItemCollection;
00039 /*! Mutable iterator over @ref YItemCollection.
00040 typedef YItemCollection::iterator YItemIterator;
00041 /*! Const iterator over @ref YItemCollection.
00042 typedef YItemCollection::const_iterator YItemConstIterator;
00043
00044
00045 /**
00046 * Simple item class for SelectionBox, ComboBox, MultiSelectionBox etc. items.
00047 * This class provides stubs for children management.
00048 */
00049 class YItem
00050 {
00051 public:
00052     /**
00053      * Constructor with just the label and optionally the selected state.
00054      */
00055     YItem( const std::string & label, bool selected = false )
00056         : _label( label )
00057         , _status( selected ? 1 : 0 )
00058         , _index( -1 )
00059         , _data( 0 )
00060     {}
00061

```

```

00062     /**
00063      * Constructor with label and icon name and optionally the selected state.
00064      */
00065     YItem( const std::string & label, const std::string & iconName, bool selected = false )
00066         : _label( label )
00067         , _iconName( iconName )
00068         , _status( selected ? 1 : 0 )
00069         , _index( -1 )
00070         , _data( 0 )
00071     {}
00072
00073     /**
00074      * Destructor.
00075      */
00076     virtual ~YItem() {}
00077
00078     /**
00079      * Return this item's label. This is what the user sees in a dialog, so
00080      * this will usually be a translated text.
00081      */
00082     std::string label() const { return _label; }
00083
00084     /**
00085      * Set this item's label.
00086      */
00087     void setLabel( const std::string & newLabel ) { _label = newLabel; }
00088
00089     /**
00090      * Return this item's icon name.
00091      */
00092     std::string iconName() const { return _iconName; }
00093
00094     /**
00095      * Return 'true' if this item has an icon name.
00096      */
00097     bool hasIconName() const { return ! _iconName.empty(); }
00098
00099     /**
00100      * Set this item's icon name.
00101      */
00102     void setIconName( const std::string & newIconName ) { _iconName = newIconName; }
00103
00104     /**
00105      * Return 'true' if this item is currently selected.
00106      */
00107     bool selected() const { return _status != 0; }
00108
00109     /**
00110      * Select or unselect this item. This does not have any effect on any other
00111      * item; if it is desired that only one item is selected at any time, the
00112      * caller has to take care of that.
00113      */
00114     void setSelected( bool sel = true ) { _status = sel ? 1 : 0; }
00115
00116     /**
00117      * Return the status of this item. This is a bit more generalized than
00118      * 'selected'. Values other than 0 or 1 can mean different things to the
00119      * application or to the specific widget.
00120      */
00121     int status() const { return _status; }
00122
00123     /**
00124      * Set the status of this item. Most widgets only use 0 for "not selected"
00125      * or nonzero for "selected". Some widgets may make use of other values as
00126      * well.
00127      */
00128     void setStatus( int newStatus ) { _status = newStatus; }
00129
00130     /**
00131      * Set this item's index.
00132      */
00133     void setIndex( int index ) { _index = index; }
00134
00135     /**
00136      * Return the index of this item (as set with setIndex() ).
00137      */
00138     int index() const { return _index; }
00139
00140     /**
00141      * Set the opaque data pointer for application use.
00142      *

```

```

00143     * Applications can use this to store the pointer to a counterpart of this
00144     * tree item. It is the application's responsibility to watch for dangling
00145     * pointers and possibly deleting the data. All this class ever does with
00146     * this pointer is to store it.
00147     **/
00148 void setData( void * newData ) { _data = newData; }
00149
00150 /**
00151  * Return the opaque data pointer.
00152  **/
00153 void * data() const { return _data; }
00154
00155 //
00156 // Children management stubs.
00157 //
00158 // Derived classes that can handle child items should reimplement those
00159 // functions.
00160 // The following default implementations don't do anything with children;
00161 // they act as if this item didn't have any children.
00162 //
00163
00164 /**
00165  * Return 'true' if this item has any child items.
00166  **/
00167 virtual bool hasChildren() const { return false; }
00168
00169 /**
00170  * Return an iterator that points to the first child item of this item.
00171  *
00172  * This default implementation returns the 'end' iterator of the
00173  * class-static always empty _noChildren YItemCollection.
00174  * It is safe to use this iterator in classic iterator loops:
00175  *
00176  *   for ( YItemIterator it = myItem->childrenBegin();
00177  *         it != myItem->childrenEnd();
00178  *         ++it )
00179  *   {
00180  *       ...
00181  *   }
00182  *
00183  * The loop body will only ever be executed if this item is a derived class
00184  * that actually manages child items.
00185  **/
00186 virtual YItemIterator childrenBegin()          { return _noChildren.end(); }
00187 virtual YItemConstIterator childrenBegin() const { return _noChildren.end(); }
00188
00189 /**
00190  * Return an iterator that points after the last child item of this item.
00191  *
00192  * This default implementation returns the 'end' iterator of the
00193  * class-static always empty _noChildren YItemCollection.
00194  **/
00195 virtual YItemIterator childrenEnd()          { return _noChildren.end(); }
00196 virtual YItemConstIterator childrenEnd() const { return _noChildren.end(); }
00197
00198 /**
00199  * Returns this item's parent item or 0 if it is a toplevel item.
00200  * This default implementation always returns 0.
00201  * Derived classes that handle children should reimplement this.
00202  **/
00203 virtual YItem * parent() const { return 0; }
00204
00205 private:
00206
00207     std::string _label;
00208     std::string _iconName;
00209     int         _status;
00210     int         _index;
00211     void *      _data;
00212
00213 /**
00214  * Static children collection that is always empty so the children
00215  * iterators of this base class have something valid to return.
00216  **/
00217 static YItemCollection _noChildren;
00218 };
00219
00220
00221
00222
00223 #endif // YItem_h

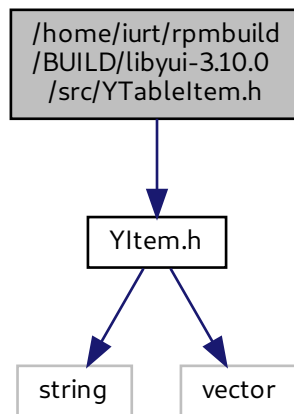
```



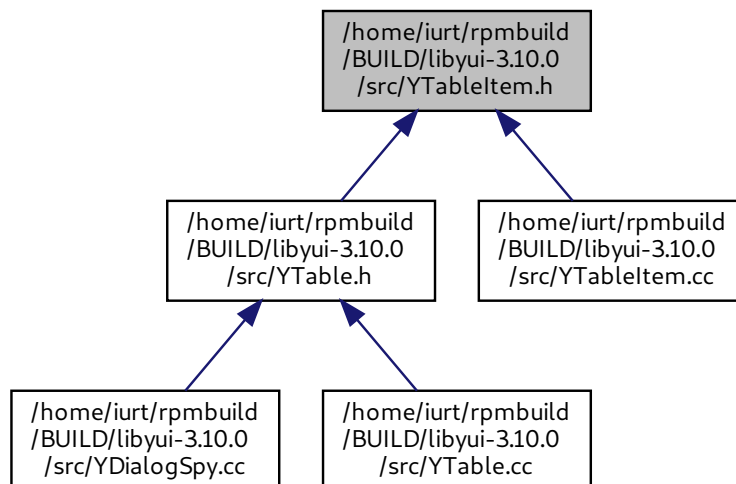
## 6.3 /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YTableItem.h File Reference

```
#include "YItem.h"
```

Include dependency graph for YTableItem.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [YTableItem](#)  
*Item class for [YTable](#) items.*
- class [YTableCell](#)  
*One cell (one column in one row) of a [YTableItem](#).*

## Typedefs

- typedef std::vector< [YTableCell](#) \* > [YTableCellCollection](#)  
*Collection of pointers to [YTableCell](#).*
- typedef YTableCellCollection::iterator [YTableCellIterator](#)  
*Mutable iterator over [YTableCellCollection](#).*
- typedef YTableCellCollection::const\_iterator [YTableCellConstIterator](#)  
*Const iterator over [YTableCellCollection](#).*

## 6.4 YTableItem.h

```

00001 /*
00002  Copyright (C) 2000-2012 Novell, Inc
00003  This library is free software; you can redistribute it and/or modify
00004  it under the terms of the GNU Lesser General Public License as
00005  published by the Free Software Foundation; either version 2.1 of the
00006  License, or (at your option) version 3.0 of the License. This library
00007  is distributed in the hope that it will be useful, but WITHOUT ANY
00008  WARRANTY; without even the implied warranty of MERCHANTABILITY or
00009  FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public
00010  License for more details. You should have received a copy of the GNU
00011  Lesser General Public License along with this library; if not, write
00012  to the Free Software Foundation, Inc., 51 Franklin Street, Fifth
00013  Floor, Boston, MA 02110-1301 USA
00014 */
00015
00016
00017 /*-
00018  File:          YTableItem.h
00019
00020  Author:        Stefan Hundhammer <sh@suse.de>
00021
00022
00023 */-
00024
00025 #ifndef YTableItem_h
00026 #define YTableItem_h
00027
00028 #include "YItem.h"
00029
00030
00031 class YTableCell;
00032
00033 // without "documenting" the file, typedefs will be dropped
00034 //! @file
00035
00036 //! Collection of pointers to YTableCell
00037 typedef std::vector<YTableCell *> YTableCellCollection;
00038 //! Mutable iterator over @ref YTableCellCollection
00039 typedef YTableCellCollection::iterator YTableCellIterator;
00040 //! Const iterator over @ref YTableCellCollection
00041 typedef YTableCellCollection::const_iterator YTableCellConstIterator;
00042
00043
00044 /**
00045  * Item class for YTable items. Each YTableItem corresponds to one row in a
00046  * YTable.
00047  *
00048  * A YTableItem might have any number of cells (columns within this row),

```

```

00049  * including none. The YTable widget is free to ignore any excess cells if
00050  * there are more than the YTable widget has columns. The YTable widget is to
00051  * treat nonexistent cells like empty ones.
00052  *
00053  * Note that while YTable items and their cells can be manipulated through
00054  * pointers, their visual representation on screen might be updated only upon
00055  * calling certain methods of the YTable widget. See the YTable reference for
00056  * details.
00057  */
00058  class YTableItem: public YItem
00059  {
00060  public:
00061
00062      /**
00063       * Default constructor. Use addCell() to give it any content.
00064       */
00065      YTableItem();
00066
00067      /**
00068       * Convenience constructor for table items without any icons.
00069       *
00070       * This will create up to 10 (0..9) cells. Empty cells for empty labels at
00071       * the end of the labels are not created, but empty cells in between are.
00072       *
00073       * new YTableItem( "one", "two", "", "", "five" );
00074       *
00075       * will create an item with 5 cells:
00076       *
00077       * cell[0] ==> "one"
00078       * cell[1] ==> "two"
00079       * cell[2] ==> ""
00080       * cell[3] ==> ""
00081       * cell[4] ==> "five"
00082       */
00083      YTableItem( const std::string & label_0,
00084                  const std::string & label_1 = std::string(),
00085                  const std::string & label_2 = std::string(),
00086                  const std::string & label_3 = std::string(),
00087                  const std::string & label_4 = std::string(),
00088                  const std::string & label_5 = std::string(),
00089                  const std::string & label_6 = std::string(),
00090                  const std::string & label_7 = std::string(),
00091                  const std::string & label_8 = std::string(),
00092                  const std::string & label_9 = std::string() );
00093
00094      /**
00095       * Destructor.
00096       *
00097       * This will delete all cells.
00098       */
00099      virtual ~YTableItem();
00100
00101      /**
00102       * Add a cell. This item will assume ownership over the cell and delete it
00103       * when appropriate (when the table is destroyed or when table items are
00104       * replaced), at which time the pointer will become invalid.
00105       *
00106       * Cells can still be changed after they (and the item they belong to) are
00107       * added, but in that case, YTable::cellChanged() needs to be called to
00108       * update the table display accordingly.
00109       */
00110      void addCell( YTableCell * cell_disown );
00111
00112      /**
00113       * Create a new cell and add it (even if all 'label',
00114       * 'iconName' and 'sortKey' are empty).
00115       */
00116      void addCell( const std::string & label, const std::string & iconName = std::string(),
00117                   const std::string & sortKey = std::string() );
00118
00119      /**
00120       * Delete all cells.
00121       */
00122      void deleteCells();
00123
00124      /**
00125       * Return an iterator that points to the first cell of this item.
00126       */
00127      YTableCellIterator      cellsBegin()          { return _cells.begin(); }
00128      YTableCellConstIterator cellsBegin() const    { return _cells.begin(); }
00129

```

```

00130     /**
00131      * Return an iterator that points after the last cell of this item.
00132      */
00133     YTableCellIterator      cellsEnd()          { return _cells.end(); }
00134     YTableCellConstIterator cellsEnd() const    { return _cells.end(); }
00135
00136     /**
00137      * Return the cell at the specified index (counting from 0 on)
00138      * or 0 if there is none.
00139      */
00140     const YTableCell * cell( int index ) const;
00141     YTableCell * cell( int index );
00142
00143     /**
00144      * Return the number of cells this item has.
00145      */
00146     int cellCount() const { return _cells.size(); }
00147
00148     /**
00149      * Return 'true' if this item has a cell with the specified index
00150      * (counting from 0 on), 'false' otherwise.
00151      */
00152     bool hasCell( int index ) const;
00153
00154     /**
00155      * Return the label of cell no. 'index' (counting from 0 on) or an empty
00156      * string if there is no cell with that index.
00157      */
00158     std::string label( int index ) const;
00159
00160     /**
00161      * Return the icon name of cell no. 'index' (counting from 0 on) or an empty
00162      * string if there is no cell with that index.
00163      */
00164     std::string iconName( int index ) const;
00165
00166     /**
00167      * Return 'true' if there is a cell with the specified index that has an
00168      * icon name.
00169      */
00170     bool hasIconName( int index ) const;
00171
00172     /**
00173      * Just for debugging.
00174      */
00175     std::string label() const { return label(0); }
00176
00177 private:
00178
00179     // Disable unwanted base class methods. They don't make sense in this
00180     // context since there is not just one single label or icon name, but one
00181     // for each cell.
00182
00183     std::string iconName() const          { return ""; }
00184     bool hasIconName() const              { return false; }
00185     void setLabel      ( const std::string & ) {}
00186     void setIconName   ( const std::string & ) {}
00187
00188     //
00189     // Data members
00190     //
00191     YTableCellCollection _cells;
00192 };
00193
00194 /**
00195  * One cell (one column in one row) of a YTableItem. Each cell has a label (a
00196  * user visible text), optionally an icon (*) and also optionally a sort-key.
00197  *
00198  * Note that cells don't have individual IDs; they have just an index.
00199  * The first cell in an item is cell(0). In an ideal world, each YTableItem
00200  * would have exactly as many cells as there are columns in the YTable, but
00201  * these classes make no such assumptions. A YTableItem might have any number
00202  * of cells, including none.
00203  *
00204  * The YTable widget is free to ignore any excess cells if there are more than
00205  * the YTable widget has columns. If there are less cells than the table has
00206  * columns, the nonexistent cells will be treated as empty.

```

```

00211 *
00212 *
00213 * (*) Not all UIs can handle icons. UIs that can't handle them will simply
00214 * ignore any icons specified for YTableCells. Thus, applications should either
00215 * check the UI capabilities if it can handle icons or use icons only as an
00216 * additional visual cue that still has a text counterpart (so the user can
00217 * still make sense of the table content when no icons are visible).
00218 **/
00219 class YTableCell
00220 {
00221 public:
00222     /**
00223      * Constructor with label and optional icon name and optional sort
00224      * key for cells that don't have a parent item yet (that will be
00225      * added to a parent later with setParent()).
00226      **/
00227     YTableCell( const std::string & label, const std::string & iconName = "",
00228                const std::string & sortKey = "" )
00229         : _label( label )
00230         , _iconName( iconName )
00231         , _sortKey( sortKey )
00232         , _parent( 0 )
00233         , _column ( -1 )
00234     {}
00235
00236     /**
00237      * Constructor with parent, column no., label and optional icon name for
00238      * cells that are created with a parent.
00239      **/
00240     YTableCell( YTableItem *          parent,
00241                int                    column,
00242                const std::string &    label,
00243                const std::string &    iconName = "",
00244                const std::string &    sortKey = "" )
00245         : _label( label )
00246         , _iconName( iconName )
00247         , _sortKey( sortKey )
00248         , _parent( parent )
00249         , _column ( column )
00250     {}
00251
00252     /**
00253      * Destructor. Not strictly needed inside this class, but useful for
00254      * derived classes. Since this is the only virtual method of this class,
00255      * the cost of this is a vtable for this class and a pointer to the vtable
00256      * in each instance.
00257      **/
00258     virtual ~YTableCell() {}
00259
00260     /**
00261      * Return this cell's label. This is what the user sees in a dialog, so
00262      * this will usually be a translated text.
00263      **/
00264     std::string label() const { return _label; }
00265
00266     /**
00267      * Set this cell's label.
00268      *
00269      * If this is called after the corresponding table item (table row) is
00270      * added to the table widget, call YTable::cellChanged() to notify the
00271      * table widget about the fact. Only then will the display be updated.
00272      **/
00273     void setLabel( const std::string & newLabel ) { _label = newLabel; }
00274
00275     /**
00276      * Return this cell's icon name.
00277      **/
00278     std::string iconName() const { return _iconName; }
00279
00280     /**
00281      * Return 'true' if this cell has an icon name.
00282      **/
00283     bool hasIconName() const { return ! _iconName.empty(); }
00284
00285     /**
00286      * Set this cell's icon name.
00287      *
00288      * If this is called after the corresponding table item (table row) is
00289      * added to the table widget, call YTable::cellChanged() to notify the
00290      * table widget about the fact. Only then will the display be updated.
00291      **/

```

```

00292 void setIconName( const std::string & newIconName ) { _iconName = newIconName; }
00293
00294 /**
00295  * Return this cell's sort key.
00296  */
00297 std::string sortKey() const { return _sortKey; }
00298
00299 /**
00300  * Return 'true' if this cell has a sort key.
00301  */
00302 bool hasSortKey() const { return ! _sortKey.empty(); }
00303
00304 /**
00305  * Set this cell's sort key.
00306  *
00307  * If this is called after the corresponding table item (table row) is
00308  * added to the table widget, call YTable::cellChanged() to notify the
00309  * table widget about the fact. Only then will the display be updated.
00310  */
00311 void setSortKey( const std::string & newSortKey ) { _sortKey = newSortKey; }
00312
00313 /**
00314  * Return this cell's parent item or 0 if it doesn't have one yet.
00315  */
00316 YTableItem * parent() const { return _parent; }
00317
00318 /**
00319  * Return this cell's column no. (counting from 0on) or -1 if it doesn't
00320  * have a parent yet.
00321  */
00322 int column() const { return _column; }
00323
00324 /**
00325  * Convenience function: Return this cell's parent item's index within its
00326  * table widget or -1 if there is no parent item or no parent table.
00327  */
00328 int itemIndex() const { return _parent ? _parent->index() : -1; }
00329
00330 /**
00331  * Set this cell's parent item and column no. if it doesn't have a parent
00332  * yet.
00333  *
00334  * This method will throw an exception if the cell already has a parent.
00335  */
00336 void reparent( YTableItem * parent, int column );
00337
00338 private:
00339
00340     std::string      _label;
00341     std::string      _iconName;
00342     std::string      _sortKey;
00343     YTableItem *     _parent;
00344     int              _column;
00345
00346 };
00347
00348
00349
00350 #endif // YTableItem_h

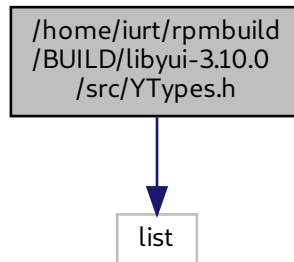
```

## 6.5 /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YTypes.h File Reference

Author: Stefan Hundhammer [sh@suse.de](mailto:sh@suse.de)

```
#include <list>
```

Include dependency graph for YTypes.h:



This graph shows which files directly or indirectly include this file:



## Macros

- `#define YUIAllDimensions 2`

## Typedefs

- `typedef double YLayoutSize_t`
- `typedef long long YFileSize_t`
- `typedef std::list< YWidget * > YWidgetList`
- `typedef std::list< YWidget * >::iterator YWidgetListIterator`
- `typedef std::list< YWidget * >::const_iterator YWidgetListConstIterator`
- `typedef std::list< YWidget * >::reverse_iterator YWidgetListReverseIterator`
- `typedef std::list< YWidget * >::const_reverse_iterator YWidgetListConstReverseIterator`

## Enumerations

- `enum YUIDimension { YD_HORIZ = 0 , YD_VERT = 1 }`
- `enum YAlignmentType { YAlignUnchanged , YAlignBegin , YAlignEnd , YAlignCenter }`
- `enum YDialogType { YMainDialog , YPopupDialog , YWizardDialog }`  
*Type of dialog: Main / Popup / Wizard.*
- `enum YDialogColorMode { YDialogNormalColor , YDialogInfoColor , YDialogWarnColor }`
- `enum YButtonRole {`  
`YCustomButton = 0 , YOKButton , YApplyButton , YCancelButton ,`  
`YHelpButton , YRelNotesButton , YMaxButtonRole }`
- `enum YButtonOrder { YKDEButtonOrder , YGnomeButtonOrder }`

### 6.5.1 Detailed Description

Author: Stefan Hundhammer [sh@suse.de](mailto:sh@suse.de)

Header file for frequently used simple types to reduce interdependencies between important headers (e.g., [YWidget.h](#), [YUI.h](#)).

Definition in file [YTypes.h](#).

## 6.6 YTypes.h

```

00001 /*
00002  Copyright (C) 2000-2012 Novell, Inc
00003  This library is free software; you can redistribute it and/or modify
00004  it under the terms of the GNU Lesser General Public License as
00005  published by the Free Software Foundation; either version 2.1 of the
00006  License, or (at your option) version 3.0 of the License. This library
00007  is distributed in the hope that it will be useful, but WITHOUT ANY
00008  WARRANTY; without even the implied warranty of MERCHANTABILITY or
00009  FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public
00010  License for more details. You should have received a copy of the GNU
00011  Lesser General Public License along with this library; if not, write
00012  to the Free Software Foundation, Inc., 51 Franklin Street, Fifth
00013  Floor, Boston, MA 02110-1301 USA
00014 */
00015
00016
00017 /**
00018  @file          YTypes.h
00019
00020  Author:        Stefan Hundhammer <sh@suse.de>
00021
00022  Header file for frequently used simple types to reduce interdependencies
00023  between important headers (e.g., YWidget.h, YUI.h).
00024
00025  **/
00026
00027
00028 #ifndef YTypes_h
00029 #define YTypes_h
00030
00031 #include <list>
00032
00033 typedef double          YLayoutSize_t;
00034 typedef long long      YFileSize_t;
00035
00036 class YWidget;
00037
00038 typedef std::list<YWidget *>          YWidgetList;
00039 typedef std::list<YWidget *>::iterator YWidgetListIterator;
00040 typedef std::list<YWidget *>::const_iterator YWidgetListConstIterator;
00041 typedef std::list<YWidget *>::reverse_iterator YWidgetListReverseIterator;
00042 typedef std::list<YWidget *>::const_reverse_iterator YWidgetListConstReverseIterator;
00043
00044
00045 #define YUIAllDimensions      2
00046
00047 enum YUIDimension
00048 {
00049     YD_HORIZ    = 0,
00050     YD_VERT     = 1
00051 };
00052
00053
00054 enum YAlignmentType
00055 {
00056     YAlignUnchanged,
00057     YAlignBegin,
00058     YAlignEnd,
00059     YAlignCenter
00060 };
00061
00062

```



```

00063 /**
00064  * Type of dialog: Main / Popup / Wizard.
00065  */
00066 enum YDialogType
00067 {
00068     YMainDialog,
00069     YPopupDialog,
00070     YWizardDialog,
00071 };
00072
00073
00074 enum YDialogColorMode
00075 {
00076     YDialogNormalColor, // Default
00077     YDialogInfoColor,   // Brighter colors
00078     YDialogWarnColor    // Very bright Warning colors
00079 };
00080
00081
00082 enum YButtonRole
00083 {
00084     YCustomButton = 0, // No predefined role
00085     YOKButton,         // [OK], [Continue], [Yes], [Accept], [Next]
00086     YApplyButton,      // [Apply]
00087     YCancelButton,     // [Cancel]
00088     YHelpButton,       // [Help]
00089     YRelNotesButton,   // [Release Notes]
00090
00091     YMaxButtonRole     // For use as array size
00092 };
00093
00094
00095 enum YButtonOrder
00096 {
00097     YKDEButtonOrder,   // [OK] [Apply] [Cancel] [Custom1] [Custom2] [Help]
00098                       // [Yes] [No]
00099                       // [Continue] [Cancel]
00100
00101     YGnomeButtonOrder // [Help] [Custom1] [Custom2] [Apply] [Cancel] [OK]
00102                       // [No] [Yes]
00103                       // [Cancel] [Continue]
00104 };
00105
00106
00107
00108 #endif // YTypes_h

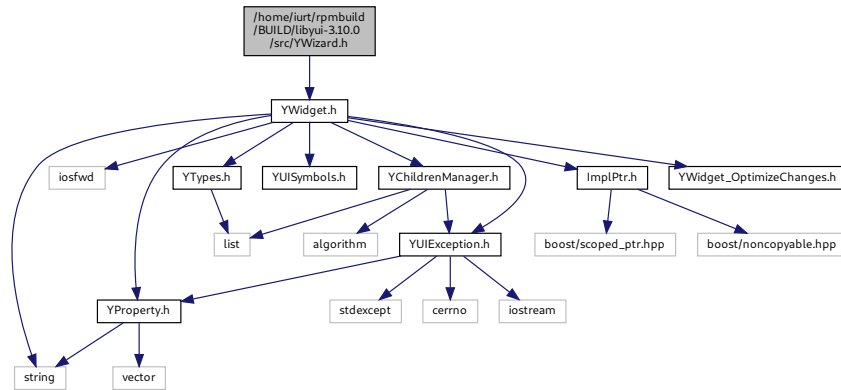
```

## 6.7 /home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YWizard.h File Reference

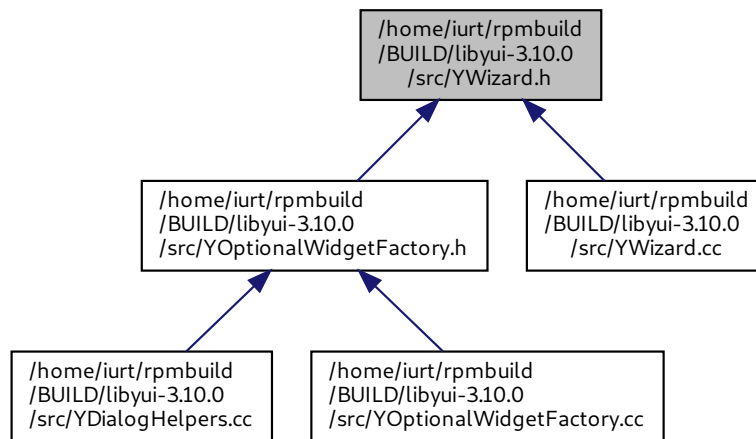
Author: Stefan Hundhammer [sh@suse.de](mailto:sh@suse.de)

```
#include "YWidget.h"
```

Include dependency graph for YWizard.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [YWizard](#)

*A wizard is a more complex frame typically used for multi-step workflows:*

## Macros

- #define **YWizardID** "wizard"
- #define **YWizardContentsReplacePointID** "contents"

## Enumerations

- enum [YWizardMode](#) { [YWizardMode\\_Standard](#) , [YWizardMode\\_Steps](#) , [YWizardMode\\_Tree](#) , [YWizardMode\\_TitleOnLeft](#) }

*Kind of the wizard layout.*

### 6.7.1 Detailed Description

Author: Stefan Hundhammer [sh@suse.de](mailto:sh@suse.de)

Definition in file [YWizard.h](#).

### 6.7.2 Enumeration Type Documentation

#### 6.7.2.1 YWizardMode

enum [YWizardMode](#)

Kind of the wizard layout.

Enumerator

<a href="#">YWizardMode_Standard</a>	Normal wizard (help panel or nothing)
<a href="#">YWizardMode_Steps</a>	Steps visible in left side panel.
<a href="#">YWizardMode_Tree</a>	Tree in left side panel.
<a href="#">YWizardMode_TitleOnLeft</a>	Title on the left side.

Definition at line 42 of file [YWizard.h](#).

## 6.8 YWizard.h

```

00001 /*
00002  Copyright (C) 2000-2012 Novell, Inc
00003  This library is free software; you can redistribute it and/or modify
00004  it under the terms of the GNU Lesser General Public License as
00005  published by the Free Software Foundation; either version 2.1 of the
00006  License, or (at your option) version 3.0 of the License. This library
00007  is distributed in the hope that it will be useful, but WITHOUT ANY
00008  WARRANTY; without even the implied warranty of MERCHANTABILITY or
00009  FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public
00010  License for more details. You should have received a copy of the GNU
00011  Lesser General Public License along with this library; if not, write
00012  to the Free Software Foundation, Inc., 51 Franklin Street, Fifth
00013  Floor, Boston, MA 02110-1301 USA
00014 */
00015
00016

```

```

00017 /**
00018
00019     @file          YWizard.h
00020
00021     Author:        Stefan Hundhammer <sh@suse.de>
00022
00023 */
00024
00025 #ifndef YWizard_h
00026 #define YWizard_h
00027
00028 #include "YWidget.h"
00029
00030 class YMacroRecorder;
00031 class YWizardPrivate;
00032 class YPushButton;
00033 class YReplacePoint;
00034
00035 #define YWizardID                "wizard"
00036 #define YWizardContentsReplacePointID  "contents"
00037
00038
00039 /**
00040  * Kind of the wizard layout
00041  */
00042 enum YWizardMode
00043 {
00044     YWizardMode_Standard,    ///< Normal wizard (help panel or nothing)
00045     YWizardMode_Steps,       ///< Steps visible in left side panel
00046     YWizardMode_Tree,        ///< Tree in left side panel
00047     YWizardMode_TitleOnLeft  ///< Title on the left side
00048 };
00049
00050
00051 /**
00052  * A wizard is a more complex frame typically used for multi-step workflows:
00053  *
00054  *   +-----+-----+-----+
00055  *   |         |         |         |
00056  *   |         |         |         |
00057  *   |         |         |         |
00058  *   |         |         |         |
00059  *   |         |         |         |
00060  *   |         |         |         |
00061  *   |         |         |         |
00062  *   |         |         |         |
00063  *   | Side bar |         | Content Area |
00064  *   |         |         | (YReplacePoint) |
00065  *   |         |         |         |
00066  *   |         |         |         |
00067  *   |         |         |         |
00068  *   |         |         |         |
00069  *   |         |         |         |
00070  *   |         |         |         |
00071  *   |         | [Back]  | [Abort]  | [Next] |
00072  *   +-----+-----+-----+
00073  *
00074  * The side bar can contain help text, a list of steps that are performed, or
00075  * an embedded tree (much like the YTree widget).
00076  *
00077  * The client application creates the wizard and replaces the widget in the
00078  * content area for each step.
00079  *
00080  * The wizard buttons can theoretically be used to do anything, but good UI
00081  * design will stick to the model above: [Back], [Abort], [Next].
00082  *
00083  * If only two buttons are desired, leave the [Back] button's label empty. The
00084  * remaining two buttons will be rearranged accordingly in the button area.
00085  *
00086  * In the last step of a multi-step workflow, the [Next] button's label is
00087  * customarily replaced with a label that indicates that this is the last
00088  * step. [Accept] is recommended for that button label: [Finish] (as sometimes
00089  * used in other environments) by no means clearly indicates that this is the
00090  * positive ending, the final "do it" button. Worse, translations of that are
00091  * often downright miserable: To German, [Finish] gets translated as [Beenden]
00092  * which is the same word as "Quit" (used in menus). This does not at all tell
00093  * the user that that button really performs the requested action the
00094  * multi-step wizard is all about.
00095  */
00096 class YWizard: public YWidget
00097 {

```

```

00098 protected:
00099     /**
00100      * Constructor.
00101      *
00102      * If only two buttons are desired, leave 'backButtonLabel' empty.
00103      */
00104     YWizard( YWidget *          parent,
00105              const std::string & backButtonLabel,
00106              const std::string & abortButtonLabel,
00107              const std::string & nextButtonLabel,
00108              YWizardMode        wizardMode = YWizardMode_Standard );
00109
00110 public:
00111
00112     /**
00113      * Destructor.
00114      */
00115     virtual ~YWizard();
00116
00117     /**
00118      * Returns a descriptive name of this widget class for logging,
00119      * debugging etc.
00120      */
00121     virtual const char * widgetClass() const { return "YWizard"; }
00122
00123
00124     //
00125     // Wizard basics
00126     //
00127
00128     /**
00129      * Return the wizard mode (what kind of wizard this is):
00130      * YWizardMode_Standard, YWizardMode_Steps, YWizardMode_Tree, YWizardMode_TitleOnLeft
00131      */
00132     YWizardMode wizardMode() const;
00133
00134     /**
00135      * Return the wizard buttons or 0 if there is no such button.
00136      *
00137      * Derived classes are required to implement this.
00138      */
00139     virtual YPushButton * backButton() const = 0;
00140     virtual YPushButton * abortButton() const = 0;
00141     virtual YPushButton * nextButton() const = 0;
00142
00143     /**
00144      * Return the internal contents ReplacePoint.
00145      *
00146      * Derived classes are required to implement this.
00147      */
00148     virtual YReplacePoint * contentsReplacePoint() const = 0;
00149
00150     /**
00151      * Protect the wizard's "Next" button against disabling.
00152      */
00153     void protectNextButton( bool protect );
00154
00155     /**
00156      * Check if the wizard's "Next" button is currently protected against
00157      * disabling.
00158      */
00159     bool nextButtonIsProtected() const;
00160
00161     /**
00162      * Set the label of one of the wizard buttons (backButton(), abortButton(),
00163      * nextButton() ) if that button is non-null.
00164      *
00165      * The default implementation simply calls button->setLabel( newLabel ).
00166      */
00167     virtual void setButtonLabel( YPushButton * button, const std::string & newLabel );
00168
00169     /**
00170      * Set the help text.
00171      */
00172     virtual void setHelpText( const std::string & helpText ) = 0;
00173
00174     /**
00175      * Set the dialog icon. An empty icon name clears the current icon.
00176      */
00177     virtual void setDialogIcon( const std::string & iconName ) = 0;
00178

```

```

00179  /**
00180   * Set the dialog title shown in the window manager's title bar.
00181   * An empty string clears the current title.
00182   */
00183  virtual void setDialogTitle( const std::string & titleText ) = 0;
00184
00185  /**
00186   * Get the current dialog title shown in the window manager's title bar.
00187   */
00188  virtual std::string getDialogTitle() = 0;
00189
00190  /**
00191   * Set the dialog heading.
00192   */
00193  virtual void setDialogHeading( const std::string & headingText ) = 0;
00194
00195  /**
00196   * Get the dialog heading.
00197   */
00198  virtual std::string getDialogHeading() = 0;
00199
00200  //
00201  // Steps handling
00202  //
00203
00204  /**
00205   * Add a step for the steps panel on the side bar.
00206   * This only adds the step to the internal list of steps.
00207   * The display is only updated upon calling updateSteps().
00208   */
00209  virtual void addStep( const std::string & text, const std::string & id ) = 0;
00210
00211  /**
00212   * Add a step heading for the steps panel on the side bar.
00213   * This only adds the heading to the internal list of steps.
00214   * The display is only updated upon calling updateSteps().
00215   */
00216  virtual void addStepHeading( const std::string & text ) = 0;
00217
00218  /**
00219   * Delete all steps and step headings from the internal lists.
00220   * The display is only updated upon calling updateSteps().
00221   */
00222  virtual void deleteSteps() = 0;
00223
00224  /**
00225   * Set the current step. This also triggers updateSteps() if necessary.
00226   */
00227  virtual void setCurrentStep( const std::string & id ) = 0;
00228
00229  /**
00230   * Update the steps display: Reflect the internal steps and heading lists
00231   * in the layout.
00232   */
00233  virtual void updateSteps() = 0;
00234
00235  //
00236  // Tree handling
00237  //
00238  //
00239
00240  /**
00241   * Add a tree item. If "parentID" is an empty string, it will be a root
00242   * item. 'text' is the text that will be displayed in the tree, 'id' the ID
00243   * with which this newly created item can be referenced - and that will be
00244   * returned when the user clicks on a tree item.
00245   */
00246  virtual void addTreeItem( const std::string & parentID,
00247                           const std::string & text,
00248                           const std::string & id ) = 0;
00249
00250  /**
00251   * Select the tree item with the specified ID, if such an item exists.
00252   */
00253  virtual void selectTreeItem( const std::string & id ) = 0;
00254
00255  /**
00256   * Returns the current tree selection or an empty string if nothing is
00257   * selected or there is no tree.
00258   */
00259  virtual std::string currentTreeSelection() = 0;

```

```

00260
00261     /**
00262      * Delete all tree items.
00263      */
00264     virtual void deleteTreeItems() = 0;
00265
00266
00267     //
00268     // Menu handling
00269     //
00270
00271     /**
00272      * Add a menu to the menu bar. If the menu bar is not visible yet, it will
00273      * be made visible. 'text' is the user-visible text for the menu bar
00274      * (including keyboard shortcuts marked with '&'), 'id' is the menu ID for
00275      * later addMenuEntry() etc. calls.
00276      */
00277     virtual void addMenu( const std::string & text,
00278                          const std::string & id ) = 0;
00279
00280     /**
00281      * Add a submenu to the menu with ID 'parentMenuID'.
00282      */
00283     virtual void addSubMenu( const std::string & parentMenuID,
00284                             const std::string & text,
00285                             const std::string & id ) = 0;
00286
00287     /**
00288      * Add a menu entry to the menu with ID 'parentMenuID'. 'id' is what will
00289      * be returned by UI::UserInput() etc. when a user activates this menu entry.
00290      */
00291     virtual void addMenuEntry( const std::string & parentMenuID,
00292                               const std::string & text,
00293                               const std::string & id ) = 0;
00294
00295     /**
00296      * Add a menu separator to a menu.
00297      */
00298     virtual void addMenuSeparator( const std::string & parentMenuID ) = 0;
00299
00300     /**
00301      * Delete all menus and hide the menu bar.
00302      */
00303     virtual void deleteMenus() = 0;
00304
00305     /**
00306      * Show a "Release Notes" button above the "Help" button in the steps panel
00307      * with the specified label that will return the specified id to
00308      * UI::UserInput() when clicked.
00309      */
00310     virtual void showReleaseNotesButton( const std::string & label,
00311                                         const std::string & id ) = 0;
00312
00313     //
00314     // Misc
00315     //
00316
00317     /**
00318      * Hide an existing "Release Notes" button.
00319      */
00320     virtual void hideReleaseNotesButton() = 0;
00321
00322     /**
00323      * Retranslate internal buttons that are not accessible from the outside:
00324      * - [Help]
00325      * - [Steps]
00326      * - [Tree]
00327      */
00328     virtual void retranslateInternalButtons() = 0;
00329
00330     /**
00331      * NOP command to check if a YWizard is running.
00332      */
00333     void ping();
00334
00335
00336     //
00337     // Property handling
00338     //
00339
00340     /**

```

```
00341     * Get a property.
00342     * Reimplemented from YWidget.
00343     *
00344     * This method may throw YUIPropertyExceptions.
00345     **/
00346     virtual YPropertyValue getProperty( const std::string & propertyName );
00347
00348     /**
00349     * Return this class's property set.
00350     * This also initializes the property upon the first call.
00351     *
00352     * Reimplemented from YWidget.
00353     **/
00354     virtual const YPropertySet & propertySet();
00355
00356
00357 private:
00358
00359     ImplPtr<YWizardPrivate> priv;
00360 };
00361
00362
00363 #endif // YWizard_h
```



# Index

/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YItem.h, [641](#), [642](#)  
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YTableItem.h, [645](#), [646](#)  
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YTypes.h, [650](#), [652](#)  
/home/iurt/rpmbuild/BUILD/libyui-3.10.0/src/YWizard.h, [653](#), [655](#)  
\_eventsBlocked  
  YUI, [546](#)  
\_terminate\_ui\_thread  
  YUI, [546](#)  
~TreetItem  
  TreetItem< PAYLOAD >, [33](#)  
~YBarGraphMultiUpdate  
  YBarGraphMultiUpdate, [64](#)  
~YCancelEvent  
  YCancelEvent, [90](#)  
~YDebugEvent  
  YDebugEvent, [145](#)  
~YDialog  
  YDialog, [152](#)  
~YEvent  
  YEvent, [191](#)  
~YEventFilter  
  YEventFilter, [195](#)  
~YExternalWidgetsTerminator  
  YExternalWidgetsTerminator, [202](#)  
~YGraphPlugin  
  YGraphPlugin, [217](#)  
~YKeyEvent  
  YKeyEvent, [267](#)  
~YMenuEvent  
  YMenuEvent, [313](#)  
~YMenuItem  
  YMenuItem, [316](#)  
~YPackageSelectorPlugin  
  YPackageSelectorPlugin, [344](#)  
~YSimpleEventHandler  
  YSimpleEventHandler, [453](#)  
~YTableCell  
  YTableCell, [501](#)  
~YTableItem  
  YTableItem, [508](#)  
~YTimeoutEvent  
  YTimeoutEvent, [515](#)  
~YTreetItem  
  YTreetItem, [532](#)  
~YUIPlugin  
  YUIPlugin, [583](#)  
~YWidgetEvent  
  YWidgetEvent, [623](#)  
activate  
  YDialog, [152](#)  
  YDumbTab, [180](#)  
  YPushButton, [375](#)  
  YTree, [524](#)  
activatedNode  
  YGraph, [211](#)  
activateItem  
  YItemSelector, [256](#)  
  YMenuButton, [302](#)  
add  
  YChildrenManager< T >, [111](#)  
  YChildrenRejector< T >, [114](#)  
  YPropertySet, [368](#)  
  YSingleChildManager< T >, [470](#)  
addBranch  
  YStringTree, [484](#)  
addCell  
  YTableItem, [509](#)  
addChild  
  TreetItem< PAYLOAD >, [34](#)  
  YAlignment, [37](#)  
  YTreetItem, [532](#)  
  YWidget, [606](#)  
addEventFilter  
  YDialog, [153](#)  
addItem  
  YContextMenu, [136](#)  
  YDumbTab, [181](#)  
  YMenuButton, [302](#)  
  YSelectionWidget, [427](#)  
addItems  
  YContextMenu, [136](#)  
  YMenuButton, [303](#)  
  YSelectionWidget, [428](#)  
  YTree, [524](#)  
addMenu  
  YWizard, [635](#)

- addMenuEntry
  - YWizard, 635
- addRadioButton
  - YRadioButtonGroup, 394
- addSegment
  - YBarGraph, 58
- addStep
  - YWizard, 635
- addStepHeading
  - YWizard, 635
- addTreeWidgetItem
  - YWizard, 636
- addWidget
  - YDialogSpyPrivate, 169
- app
  - YUI, 538
- applicationIcon
  - YApplication, 45
- applicationTitle
  - YApplication, 45
- arg
  - YCommandLine, 129
- argc
  - YCommandLine, 129
- argv
  - YCommandLine, 130
- askForExistingDirectory
  - YApplication, 46
- askForExistingFile
  - YApplication, 46
- askForSaveFileName
  - YApplication, 46
- autoScrollDown
  - YRichText, 407
- backButton
  - YWizard, 636
- beep
  - YApplication, 47
- begin
  - YWidget, 606
- blockEvents
  - YSimpleEventHandler, 454
  - YUI, 538
- builtinCaller
  - YUI, 538
- busyCursor
  - YApplication, 47
- buttonsByButtonOrder
  - YButtonBox, 80
- call
  - YBuiltinCaller, 70
- cellChanged
  - YTable, 494
- check
  - YPropertySet, 368, 369
- checkShortcuts
  - YDialog, 153
  - YShortcutManager, 449
- childrenBegin
  - YItem, 248
  - YTreeWidgetItem, 533
- childrenEnd
  - YItem, 249
  - YTreeWidgetItem, 533
- clear
  - YChildrenManager< T >, 111
- clearShortcut
  - YShortcut, 443
- column
  - YTableCell, 502
- completePath
  - YStringTree, 484
- conflict
  - YShortcut, 443
- conflictCount
  - YShortcutManager, 450
- consumePendingEvent
  - YSimpleEventHandler, 454
- container
  - YChildrenManager< T >, 111
- contains
  - YChildrenManager< T >, 111
  - YPropertySet, 369
- contentsReplacePoint
  - YWizard, 636
- countLayoutStretchChildren
  - YLayoutBox, 280
- countStretchableChildren
  - YLayoutBox, 280
- createApplication
  - YUI, 538
- createExternalWidgetFactory
  - YExternalWidgets, 200
- createGraph
  - YGraphPlugin, 217
- createOptionalWidgetFactory
  - YUI, 539
- createPackageSelector
  - YPackageSelectorPlugin, 344
- createWidgetFactory
  - YUI, 539
- currentDialog
  - YDialog, 153
- currentFileSize
  - YDownloadProgress, 173
- currentItem
  - YMultiSelectionBox, 333

- YTree, [525](#)
- currentValue
  - YMultiProgressMeter, [327](#)
- customStatus
  - YItemSelector, [256](#)
- cycleCustomStatus
  - YItemSelector, [256](#)
- debug
  - YUILog, [570](#)
- debugLabel
  - YDumbTab, [181](#)
  - YLabel, [270](#)
  - YWidget, [607](#)
- defaultButton
  - YDialog, [154](#)
- defaultFunctionKey
  - YApplication, [47](#)
- defaultVisibleLines
  - YMultiLineEdit, [318](#)
- deleteAllItems
  - YContextMenu, [137](#)
  - YMenuButton, [304](#)
  - YSelectionWidget, [428](#)
- deleteEvent
  - YSimpleEventHandler, [454](#)
- deleteNotify
  - YUI, [539](#)
- deletePendingEventsFor
  - YSimpleEventHandler, [455](#)
- deleteSteps
  - YWizard, [636](#)
- deleteTopmostDialog
  - YDialog, [154](#)
- deleteUI
  - YUILoader, [566](#)
- description
  - YDescribedItem, [147](#)
- deselectAllItems
  - YSelectionWidget, [429](#)
- destroy
  - YDialog, [154](#)
- deviceUnits
  - YApplication, [48](#)
- dimension
  - YSpacing, [476](#)
- displayLogText
  - YLogView, [289](#)
- distinctShortcutChars
  - YShortcut, [444](#)
- doLayout
  - YButtonBox, [80](#)
- doResize
  - YLayoutBox, [280](#)
- doUpdate
  - YBarGraph, [58](#)
  - YMultiProgressMeter, [327](#)
- dumpDialogWidgetTree
  - YWidget, [607](#)
- dumpOn
  - YUIBadPropertyArgException, [548](#)
  - YUIIndexOutOfRangeException, [560](#)
  - YUIInvalidChildException< YWidget >, [562](#)
  - YUIPropertyException, [587](#)
  - YUIPropertyTypeMismatchException, [589](#)
  - YUISetReadOnlyPropertyException, [592](#)
  - YUITooManyChildrenException< YWidget >, [596](#)
  - YUIUnknownPropertyException, [598](#)
- edit
  - YPropertyEditor, [366](#)
- editable
  - YComboBox, [119](#)
- editNewStringArray
  - YPopupInternal, [355](#)
- editStringArray
  - YPopupInternal, [356](#)
- empty
  - YChildrenManager< T >, [112](#)
- enabled
  - YDescribedItem, [147](#)
- end
  - YWidget, [608](#)
- enforceRange
  - YIntField, [239](#)
- ensureUICreated
  - YUI, [539](#)
- eventsBlocked
  - YUI, [540](#)
- exec
  - YDialogSpy, [166](#)
- externalWidgetFactory
  - YExternalWidgets, [200](#)
- externalWidgets
  - YExternalWidgets, [201](#)
- filter
  - YEventFilter, [196](#)
  - YHelpButtonHandler, [220](#)
  - YRelNotesButtonHandler, [402](#)
- filterInvalidEvents
  - YDialog, [155](#)
- find
  - YCommandLine, [130](#)
- findButton
  - YButtonBox, [81](#)
- findCurrentThread
  - YUILogPrivate, [578](#)
- findDialog

- YWidget, 608
- findDominatingChild
  - YLayoutBox, 281
- findItem
  - YMenuButton, 304, 305
  - YSelectionWidget, 429, 430
  - YTable, 494
  - YTree, 525
- findMenuItem
  - YContextMenu, 137, 138
  - YMenuButton, 306
- findRadioButtonGroup
  - YRadioButton, 386
- findSelectedItem
  - YSelectionWidget, 430
- findShortcut
  - YShortcut, 444
- findShortcutPos
  - YShortcut, 444
- findShortestWidget
  - YShortcutManager, 450
- findShortestWizardButton
  - YShortcutManager, 450
- findWidget
  - YApplication, 48
  - YWidget, 609
- firstChild
  - YChildrenManager< T >, 112
  - YWidget, 609
- focusWidget
  - YKeyEvent, 267
- form
  - FSize, 25
- FSize, 21
  - form, 25
  - FSize, 23, 24
  - operator long long, 25
- functionKey
  - YWidget, 610
- getProperty
  - YBarGraph, 58
  - YBusyIndicator, 72
  - YCheckBox, 92
  - YCheckBoxFrame, 100
  - YComboBox, 119
  - YContextMenu, 138
  - YDownloadProgress, 173
  - YDumbTab, 182
  - YFrame, 204
  - YGraph, 211
  - YInputField, 228
  - YIntField, 240
  - YItemSelector, 256
  - YLabel, 270
  - YLogView, 289
  - YMenuButton, 307
  - YMultiLineEdit, 318
  - YMultiProgressMeter, 327
  - YMultiSelectionBox, 334
  - YPartitionSplitter, 348
  - YProgressBar, 359
  - YPushButton, 375
  - YRadioButton, 386
  - YRadioButtonGroup, 394
  - YRichText, 408
  - YSelectionBox, 419
  - YSimpleInputField, 459
  - YTable, 495
  - YTimezoneSelector, 517
  - YTree, 526
  - YWidget, 610
  - YWizard, 636
- glyph
  - YApplication, 49
- handleChildrenEnablement
  - YCheckBoxFrame, 100
- hasChildren
  - YTreeItem, 533
- hasColumn
  - YTable, 496
  - YTableHeader, 504
- hasSegmentColor
  - YBarGraphSegment, 67
- hasTextColor
  - YBarGraphSegment, 67
- hasZeroSize
  - YImage, 223
- highlight
  - YDialog, 156
- hScrollValue
  - YRichText, 408
- iconBasePath
  - YApplication, 50
- iconFullPath
  - YSelectionWidget, 431
- iconName
  - YTableItem, 509
- id
  - YMenuEvent, 313
- idleLoop
  - YUI, 540
- immediateMode
  - YSelectionBox, 419
  - YTable, 496
  - YTree, 527
- ImplPtr< \_Impl >, 26

- initConsoleKeyboard
  - YApplication, 50
- inputMaxLength
  - YComboBox, 120
  - YInputField, 229
  - YMultiLineEdit, 319
- insertChildSorted
  - SortedTreeItem< PAYLOAD >, 30
- instance
  - YUILog, 570
- invalidate
  - YEvent, 192
- isDefaultButton
  - YPushButton, 375
- isEqual
  - YStringWidgetID, 490
- isHeading
  - YLabel, 271
- isHelpButton
  - YPushButton, 376
- isLayoutStretch
  - YLayoutBox, 282
- isOpen
  - YTreeItem, 533
- isOutputField
  - YLabel, 271
- isRelNotesButton
  - YPushButton, 376
- isValid
  - YEvent, 192
  - YShortcut, 445
  - YWidget, 611
- item
  - YEvent, 192
  - YMenuEvent, 314
- itemAt
  - YContextMenu, 139
  - YMenuButton, 308
- itemsBegin
  - YSelectionWidget, 432
- itemsCount
  - YSelectionWidget, 432
- keepSorting
  - YTable, 496
- label
  - YBarGraphSegment, 68
  - YItem, 249
  - YTableCell, 502
  - YTableItem, 510
- language
  - YApplication, 50
- layoutPass
  - YLabel, 272
- layoutUnits
  - YApplication, 50
- loadedUI
  - YSettings, 439
- loadExternalWidgets
  - YUILoader, 566
- loadPlugin
  - YUILoader, 567
- loadUI
  - YUILoader, 567
- locateSymbol
  - YUIPlugin, 583
- log
  - YUIException, 556
  - YUILog, 570
- logFileName
  - YUILog, 571
- loggerFunction
  - YUILog, 571
- makeScreenShot
  - YApplication, 51
- margins
  - YButtonBox, 82
- maxLines
  - YLogView, 290
- maxValue
  - YProgressBar, 359
- message
  - YPopupInternal, 356
- minHeight
  - YAlignment, 37
- minWidth
  - YAlignment, 38
- moveChild
  - YButtonBox, 82
  - YLayoutBox, 282
- msg
  - YUIException, 557
- nextStatus
  - YItemCustomStatus, 252
- normalCursor
  - YApplication, 51
- normalized
  - YShortcut, 445
- notify
  - YWidget, 611
- notifyContextMenu
  - YWidget, 612
- open
  - YDialog, 156
- openContextMenu
  - YApplication, 51

- openInternal
  - YDialog, [157](#)
- openUI
  - YApplication, [52](#)
- operator long long
  - FSize, [25](#)
- operator new
  - YWidget, [612](#)
- operator!=
  - YPropertyValue, [371](#)
- operator==
  - YPropertyValue, [371](#)
- operator[]
  - YCommandLine, [131](#)
- OptimizeChanges, [28](#)
- optionalWidgetFactory
  - YUI, [540](#)
- origPath
  - YStringTree, [485](#)
- overflow
  - YUILogBuffer, [575](#)
- parent
  - YItem, [249](#)
  - YTreeWidgetItem, [534](#)
- passwordMode
  - YInputField, [229](#)
- path
  - YStringTree, [486](#)
- pendingEvent
  - YSimpleEventHandler, [455](#)
- pipe\_from\_ui
  - YUI, [546](#)
- pipe\_to\_ui
  - YUI, [546](#)
- plainTextMode
  - YRichText, [409](#)
- pollEvent
  - YDialog, [157](#)
- pollEventInternal
  - YDialog, [158](#)
- postponeShortcutCheck
  - YDialog, [158](#)
- preferred
  - YShortcut, [445](#)
- preferredHeight
  - YAlignment, [38](#)
  - YButtonBox, [82](#)
  - YEmpty, [188](#)
  - YLayoutBox, [283](#)
  - YSingleChildContainerWidget, [466](#)
  - YSpacing, [476](#)
  - YWidget, [612](#)
- preferredSize
  - YLayoutBox, [283](#)
  - YWidget, [612](#)
- preferredWidth
  - YAlignment, [38](#)
  - YButtonBox, [83](#)
  - YEmpty, [188](#)
  - YLayoutBox, [284](#)
  - YSingleChildContainerWidget, [466](#)
  - YSpacing, [476](#)
  - YWidget, [613](#)
- propertySet
  - YBarGraph, [59](#)
  - YBusyIndicator, [73](#)
  - YCheckBox, [93](#)
  - YCheckBoxFrame, [101](#)
  - YComboBox, [120](#)
  - YContextMenu, [139](#)
  - YDownloadProgress, [174](#)
  - YDumbTab, [183](#)
  - YFrame, [205](#)
  - YGraph, [211](#)
  - YInputField, [229](#)
  - YIntField, [240](#)
  - YItemSelector, [257](#)
  - YLabel, [272](#)
  - YLogView, [290](#)
  - YMenuButton, [308](#)
  - YMultiLineEdit, [319](#)
  - YMultiProgressMeter, [328](#)
  - YMultiSelectionBox, [334](#)
  - YPartitionSplitter, [348](#)
  - YProgressBar, [360](#)
  - YPushButton, [376](#)
  - YRadioButton, [387](#)
  - YRadioButtonGroup, [395](#)
  - YRichText, [409](#)
  - YSelectionBox, [419](#)
  - YSimpleInputField, [459](#)
  - YTable, [497](#)
  - YTimezoneSelector, [518](#)
  - YTree, [527](#)
  - YWidget, [613](#)
  - YWizard, [637](#)
- radioButtonsBegin
  - YRadioButtonGroup, [396](#)
- reason
  - YWidgetEvent, [624](#)
- rebuildMenuTree
  - YContextMenu, [140](#)
  - YMenuButton, [309](#)
- rebuildTree
  - YTree, [528](#)
- recalcLayout

- YDialog, 158
- recordMakeScreenShot
  - YMacroRecorder, 299
- redrawScreen
  - YApplication, 52
- remove
  - YChildrenManager< T >, 112
  - YCommandLine, 131
- removeChild
  - YWidget, 614
- removeEventFilter
  - YDialog, 159
- removeRadioButton
  - YRadioButtonGroup, 396
- renderGraph
  - YGraph, 212
- reparent
  - YTableCell, 502
- replace
  - YCommandLine, 132
- requestMultiPassLayout
  - YDialog, 159
- resolveAllConflicts
  - YShortcutManager, 451
- resolveConflict
  - YShortcutManager, 451
- resolveShortcutConflicts
  - YContextMenu, 140
  - YMenuButton, 309
- retranslateInternalButtons
  - YWizard, 638
- root
  - YStringTree, 487
- runInTerminal
  - YApplication, 52
- runPkgSelection
  - YUI, 541
- sanityCheck
  - YButtonBox, 83
- saveUserInput
  - YInputField, 230
  - YMultiSelectionBox, 335
  - YRadioButton, 388
  - YWidget, 614
- segment
  - YBarGraph, 60
- selectedItem
  - YComboBox, 121
- selectedItems
  - YComboBox, 122
  - YSelectionWidget, 432
- selectedWidget
  - YDialogSpyPrivate, 169
- selectItem
  - YComboBox, 122
  - YSelectionWidget, 433
- sendEvent
  - YSimpleEventHandler, 456
- sendKeyEvents
  - YWidget, 615
- sendWidgetID
  - YUI, 541
- serial
  - YEvent, 192
- setAlive
  - YBusyIndicator, 73
- setAutoEnable
  - YCheckBoxFrame, 102
- setAutoScale
  - YImage, 223
- setAutoScrollDown
  - YRichText, 409
- setAutoWrap
  - YLabel, 273
- setBackgroundPixmap
  - YAlignment, 39
- setBeingDestroyed
  - YWidget, 615
- setBuiltinCaller
  - YUI, 542
- setButtonLabel
  - YWizard, 638
- setButtonOrderFromEnvironment
  - YUI, 542
- setChildrenManager
  - YWidget, 616
- setConsoleFont
  - YApplication, 52
- setCurrentItem
  - YMultiSelectionBox, 336
- setCurrentStep
  - YWizard, 638
- setCurrentValue
  - YMultiProgressMeter, 329
- setData
  - YItem, 250
- setDefaultButton
  - YDialog, 159
  - YPushButton, 377
- setDefaultFunctionKey
  - YApplication, 53
- setDefaultStretchable
  - YWidget, 616
- setDefaultVisibleLines
  - YMultiLineEdit, 320
- setDialogIcon
  - YWizard, 639

- setDialogTitle
  - YWizard, [639](#)
- setEnabled
  - YDescribedItem, [147](#)
  - YWidget, [616](#)
- setEnabledDebugLoggingHooks
  - YUILog, [572](#)
- setEnforceInitialSelection
  - YSelectionWidget, [434](#)
- setEnforceSingleSelection
  - YSelectionWidget, [434](#)
- setExpectedSize
  - YDownloadProgress, [175](#)
- setFilename
  - YDownloadProgress, [175](#)
  - YGraph, [212](#)
- setFunctionKey
  - YPushButton, [378](#)
  - YWidget, [617](#)
- setGraph
  - YGraph, [213](#)
- setHelpButton
  - YPushButton, [378](#)
- setHelpText
  - YWidget, [617](#)
- setHScrollValue
  - YRichText, [410](#)
- setIcon
  - YPushButton, [379](#)
- setIconBasePath
  - YSelectionWidget, [434](#)
- setIconDir
  - YSettings, [439](#)
- setIconName
  - YTableCell, [502](#)
- setId
  - YWidget, [617](#)
- setImage
  - YImage, [223](#)
- setInputMaxLength
  - YComboBox, [123](#)
  - YInputField, [231](#)
  - YMultiLineEdit, [320](#)
- setInvertAutoEnable
  - YCheckBoxFrame, [102](#)
- setItemStatus
  - YItemSelector, [258](#)
  - YSelectionWidget, [435](#)
- setKeepSorting
  - YTable, [497](#)
- setKeyboardFocus
  - YWidget, [618](#)
- setLabel
  - YBarGraph, [60](#)
  - YBarGraphSegment, [68](#)
  - YBusyIndicator, [74](#)
  - YCheckBox, [94](#)
  - YCheckBoxFrame, [103](#)
  - YDownloadProgress, [176](#)
  - YFrame, [205](#)
  - YInputField, [231](#)
  - YIntField, [241](#)
  - YLogView, [291](#)
  - YMultiLineEdit, [320](#)
  - YProgressBar, [360](#)
  - YPushButton, [379](#)
  - YRadioButton, [388](#)
  - YSelectionWidget, [435](#)
  - YSimpleInputField, [460](#)
  - YTableCell, [503](#)
- setLanguage
  - YApplication, [53](#)
- setLayoutAlgorithm
  - YGraph, [213](#)
- setLayoutPolicy
  - YButtonBox, [84](#)
- setLocaleDir
  - YSettings, [440](#)
- setLogFileName
  - YUILog, [572](#)
- setLoggerFunction
  - YUILog, [573](#)
- setMargins
  - YButtonBox, [85](#)
- setMaxLines
  - YLogView, [291](#)
- setMaxValue
  - YIntField, [241](#)
- setMinValue
  - YIntField, [242](#)
- setNextStatus
  - YItemCustomStatus, [252](#)
- setOrig
  - YTransText, [521](#)
- setPlainTextMode
  - YRichText, [410](#)
- setPlayer
  - YMacro, [297](#)
- setProductName
  - YApplication, [54](#)
- setProgDir
  - YSettings, [440](#)
- setProperty
  - YBarGraph, [60](#)
  - YBusyIndicator, [74](#)
  - YCheckBox, [94](#)
  - YCheckBoxFrame, [103](#)
  - YComboBox, [123](#)



- YContextMenu, [141](#)
- YDownloadProgress, [176](#)
- YDumbTab, [183](#)
- YFrame, [206](#)
- YGraph, [214](#)
- YInputField, [231](#)
- YIntField, [242](#)
- YItemSelector, [258](#)
- YLabel, [273](#)
- YLogView, [292](#)
- YMenuButton, [310](#)
- YMultiLineEdit, [321](#)
- YMultiProgressMeter, [329](#)
- YMultiSelectionBox, [336](#)
- YPartitionSplitter, [349](#)
- YProgressBar, [361](#)
- YPushButton, [379](#)
- YRadioButton, [389](#)
- YRadioButtonGroup, [396](#)
- YRichText, [411](#)
- YSelectionBox, [420](#)
- YSimpleInputField, [461](#)
- YTable, [498](#)
- YTimezoneSelector, [518](#)
- YTree, [528](#)
- YWidget, [618](#)
- setRecorder
  - YMacro, [297](#)
- setRelNotesButton
  - YPushButton, [380](#)
- setReverseLayout
  - YApplication, [54](#)
- setRole
  - YPushButton, [380](#)
- setSanityCheckRelaxed
  - YButtonBox, [85](#)
- setSegmentColor
  - YBarGraph, [61](#)
- setSelected
  - YItem, [250](#)
- setShortcut
  - YItemShortcut, [264](#)
- setShortcutString
  - YCheckBox, [95](#)
  - YCheckBoxFrame, [104](#)
  - YDumbTab, [184](#)
  - YInputField, [232](#)
  - YIntField, [243](#)
  - YLogView, [293](#)
  - YMultiLineEdit, [322](#)
  - YPushButton, [381](#)
  - YRadioButton, [390](#)
  - YSelectionWidget, [435](#)
  - YSimpleInputField, [462](#)
  - YWidget, [619](#)
- setShrinkable
  - YInputField, [233](#)
  - YMultiSelectionBox, [337](#)
  - YRichText, [412](#)
  - YSelectionBox, [421](#)
- setSize
  - YAlignment, [39](#)
  - YButtonBox, [86](#)
  - YLayoutBox, [285](#)
  - YSingleChildContainerWidget, [467](#)
  - YWidget, [620](#)
- setSortKey
  - YTableCell, [503](#)
- setStatus
  - YItem, [250](#)
- setTableHeader
  - YTable, [499](#)
- setText
  - YComboBox, [124](#)
  - YLabel, [274](#)
- setTextColor
  - YBarGraph, [62](#)
- setTextdomain
  - YStringTree, [487](#)
- setThemeDir
  - YSettings, [440](#)
- setTimeout
  - YBusyIndicator, [75](#)
- setUseBoldFont
  - YCheckBox, [96](#)
  - YLabel, [275](#)
  - YRadioButton, [390](#)
- setValidChars
  - YComboBox, [124](#)
  - YInputField, [233](#)
- setValue
  - TreelItem< PAYLOAD >, [34](#)
  - YBarGraph, [62](#)
  - YCheckBox, [96](#)
  - YCheckBoxFrame, [105](#)
  - YComboBox, [125](#)
  - YInputField, [234](#)
  - YIntField, [244](#)
  - YMultiLineEdit, [322](#)
  - YPartitionSplitter, [350](#)
  - YProgressBar, [362](#)
  - YRadioButton, [390](#)
  - YRichText, [412](#)
  - YSimpleInputField, [462](#)
- setValueInternal
  - YIntField, [244](#)
- setVisibleItems
  - YItemSelector, [259](#)

- setVisibleLines
  - YLogView, 293
- setVScrollValue
  - YRichText, 413
- setWidgetRep
  - YWidget, 620
- setZeroSize
  - YImage, 224
- shortcut
  - YShortcut, 446
- shortcutChanged
  - YDumbTab, 185
- shortcutString
  - YCheckBox, 96
  - YCheckBoxFrame, 105
  - YDumbTab, 185
  - YInputField, 234
  - YIntField, 245
  - YLogView, 294
  - YMultiLineEdit, 323
  - YPushButton, 382
  - YRadioButton, 390
  - YSelectionWidget, 436
  - YShortcut, 446
  - YSimpleInputField, 463
  - YWidget, 620
- showChild
  - YReplacePoint, 404
- showDialogSpy
  - YDialogSpy, 166
- showHelpText
  - YDialog, 160
- showRelNotesText
  - YDialog, 160
- showText
  - YDialog, 161
- shrinkable
  - YRichText, 413
- shutdownThreads
  - YUI, 542
- size
  - YSpacing, 476, 477
- SortedTreelItem
  - SortedTreelItem< PAYLOAD >, 29
- SortedTreelItem< PAYLOAD >, 28
  - insertChildSorted, 30
  - SortedTreelItem, 29
- startMultipleChanges
  - YWidget, 621
- status
  - YItem, 250
- stretchable
  - YAlignment, 40
  - YButtonBox, 86
  - YDumbTab, 185
  - YLayoutBox, 285
  - YSingleChildContainerWidget, 467
  - YSquash, 480
  - YWidget, 621
- stringVal
  - YPropertyValue, 372
- text
  - YComboBox, 125
- textIndicator
  - YItemCustomStatus, 253
- toggleProperties
  - YDialogSpyPrivate, 170
- topmostConstructorHasFinished
  - YUI, 543
- toString
  - YStringWidgetID, 490
  - YWidgetID, 628
- trans
  - YTransText, 521
- translate
  - YStringTree, 488
- translatedPath
  - YStringTree, 488
- TreelItem
  - TreelItem< PAYLOAD >, 32, 33
- TreelItem< PAYLOAD >, 31
  - ~TreelItem, 33
  - addChild, 34
  - setValue, 34
  - TreelItem, 32, 33
- type
  - YPropertyValue, 372
- uiThreadDestructor
  - YUI, 543
- uiThreadMainLoop
  - YUI, 544
- unlockEvents
  - YSimpleEventHandler, 456
  - YUI, 544
- uncheckOtherButtons
  - YRadioButtonGroup, 397
- unload
  - YUIPlugin, 584
- updateCustomStatusIndicator
  - YItemSelector, 259
- userInputProperty
  - YCheckBox, 97
  - YCheckBoxFrame, 105
  - YComboBox, 126
  - YInputField, 234
  - YIntField, 245
  - YItemSelector, 260

- YMultiLineEdit, 323
- YMultiSelectionBox, 337
- YPartitionSplitter, 350
- YRadioButton, 391
- YSelectionBox, 421
- YSimpleInputField, 463
- YTable, 499
- YTree, 529
- YWidget, 621
- usingCustomStatus
  - YItemSelector, 260
- validChars
  - YComboBox, 126
  - YInputField, 235
- validCustomStatusIndex
  - YItemSelector, 260
- value
  - YCheckBox, 97
  - YCheckBoxFrame, 106
  - YComboBox, 126
  - YInputField, 235
  - YIntField, 245
  - YMultiLineEdit, 323
  - YPartitionSplitter, 350
  - YRadioButton, 391
  - YSimpleInputField, 463
- visibleItems
  - YItemSelector, 261
- vScrollValue
  - YRichText, 413
- waitForEvent
  - YDialog, 161
- waitForEventInternal
  - YDialog, 162
- weight
  - YWidget, 622
- what
  - YUIException, 557
- widget
  - YEvent, 193
  - YWidgetEvent, 624
- widgetClass
  - YLabel, 275
  - YRadioButton, 391
- widgetFactory
  - YUI, 545
- writeBuffer
  - YUILogBuffer, 575
- xspuIn
  - YUILogBuffer, 576
- YAlignment, 35
- addChild, 37
- minHeight, 37
- minWidth, 38
- preferredHeight, 38
- preferredWidth, 38
- setBackgroundPixmap, 39
- setSize, 39
- stretchable, 40
- YAlignmentPrivate, 41
- YApplication, 42
  - applicationIcon, 45
  - applicationTitle, 45
  - askForExistingDirectory, 46
  - askForExistingFile, 46
  - askForSaveFileName, 46
  - beep, 47
  - busyCursor, 47
  - defaultFunctionKey, 47
  - deviceUnits, 48
  - findWidget, 48
  - glyph, 49
  - iconBasePath, 50
  - initConsoleKeyboard, 50
  - language, 50
  - layoutUnits, 50
  - makeScreenShot, 51
  - normalCursor, 51
  - openContextMenu, 51
  - openUI, 52
  - redrawScreen, 52
  - runInTerminal, 52
  - setConsoleFont, 52
  - setDefaultFunctionKey, 53
  - setLanguage, 53
  - setProductName, 54
  - setReverseLayout, 54
- YApplication, 45
- YApplicationPrivate, 55
- YBarGraph, 56
  - addSegment, 58
  - doUpdate, 58
  - getProperty, 58
  - propertySet, 59
  - segment, 60
  - setLabel, 60
  - setProperty, 60
  - setSegmentColor, 61
  - setTextColor, 62
  - setValue, 62
- YBarGraphMultiUpdate, 63
  - ~YBarGraphMultiUpdate, 64
  - YBarGraphMultiUpdate, 63
- YBarGraphPrivate, 65
- YBarGraphSegment, 65

- hasSegmentColor, 67
- hasTextColor, 67
- label, 68
- setLabel, 68
- YBarGraphSegment, 66
- YBothDim< T >, 69
- YBuiltinCaller, 70
  - call, 70
- YBusyIndicator, 71
  - getProperty, 72
  - propertySet, 73
  - setAlive, 73
  - setLabel, 74
  - setProperty, 74
  - setTimeout, 75
- YBusyIndicatorPrivate, 76
- YButtonBox, 77
  - buttonsByButtonOrder, 80
  - doLayout, 80
  - findButton, 81
  - margins, 82
  - moveChild, 82
  - preferredHeight, 82
  - preferredWidth, 83
  - sanityCheck, 83
  - setLayoutPolicy, 84
  - setMargins, 85
  - setSanityCheckRelaxed, 85
  - setSize, 86
  - stretchable, 86
- YButtonBoxLayoutPolicy, 87
- YButtonBoxMargins, 88
- YButtonBoxPrivate, 88
- YCancelEvent, 89
  - ~YCancelEvent, 90
- YCheckBox, 91
  - getProperty, 92
  - propertySet, 93
  - setLabel, 94
  - setProperty, 94
  - setShortcutString, 95
  - setUseBoldFont, 96
  - setValue, 96
  - shortcutString, 96
  - userInputProperty, 97
  - value, 97
- YCheckBoxFrame, 98
  - getProperty, 100
  - handleChildrenEnablement, 100
  - propertySet, 101
  - setAutoEnable, 102
  - setInvertAutoEnable, 102
  - setLabel, 103
  - setProperty, 103
- setShortcutString, 104
- setValue, 105
- shortcutString, 105
- userInputProperty, 105
- value, 106
- YCheckBoxFramePrivate, 106
- YCheckBoxPrivate, 107
- YChildrenManager
  - YChildrenManager< T >, 110
- YChildrenManager< T >, 108
  - add, 111
  - clear, 111
  - container, 111
  - contains, 111
  - empty, 112
  - firstChild, 112
  - remove, 112
  - YChildrenManager, 110
- YChildrenRejector< T >, 113
  - add, 114
- YCodeLocation, 115
  - YCodeLocation, 115
- YColor, 116
- YComboBox, 117
  - editable, 119
  - getProperty, 119
  - inputMaxLength, 120
  - propertySet, 120
  - selectedItem, 121
  - selectedItems, 122
  - selectItem, 122
  - setInputMaxLength, 123
  - setProperty, 123
  - setText, 124
  - setValidChars, 124
  - setValue, 125
  - text, 125
  - userInputProperty, 126
  - validChars, 126
  - value, 126
  - YComboBox, 119
- YComboBoxPrivate, 127
- YCommandLine, 128
  - arg, 129
  - argc, 129
  - argv, 130
  - find, 130
  - operator[], 131
  - remove, 131
  - replace, 132
  - YCommandLine, 129
- YCommandLinePrivate, 133
- YContextMenu, 134
  - addItem, 136

- addItem, 136
- deleteAllItems, 137
- findMenuItem, 137, 138
- getProperty, 138
- itemAt, 139
- propertySet, 139
- rebuildMenuTree, 140
- resolveShortcutConflicts, 140
- setProperty, 141
- YContextMenu, 136
- YContextMenuPrivate, 142
- YDateField, 142
- YDateFieldPrivate, 144
- YDebugEvent, 144
  - ~YDebugEvent, 145
- YDescribedItem, 146
  - description, 147
  - enabled, 147
  - setEnabled, 147
- YDialog, 148
  - ~YDialog, 152
  - activate, 152
  - addEventFilter, 153
  - checkShortcuts, 153
  - currentDialog, 153
  - defaultButton, 154
  - deleteTopmostDialog, 154
  - destroy, 154
  - filterInvalidEvents, 155
  - highlight, 156
  - open, 156
  - openInternal, 157
  - pollEvent, 157
  - pollEventInternal, 158
  - postponeShortcutCheck, 158
  - recalcLayout, 158
  - removeEventFilter, 159
  - requestMultiPassLayout, 159
  - setDefaultButton, 159
  - showHelpText, 160
  - showRelNotesText, 160
  - showText, 161
  - waitForEvent, 161
  - waitForEventInternal, 162
  - YDialog, 151
- YDialogPrivate, 163
- YDialogSpy, 164
  - exec, 166
  - showDialogSpy, 166
  - YDialogSpy, 165
- YDialogSpyPrivate, 168
  - addWidget, 169
  - selectedWidget, 169
  - toggleProperties, 170
- YDownloadProgress, 171
  - currentFileSize, 173
  - getProperty, 173
  - propertySet, 174
  - setExpectedSize, 175
  - setFilename, 175
  - setLabel, 176
  - setProperty, 176
  - YDownloadProgress, 172
- YDownloadProgressPrivate, 178
- YDumbTab, 179
  - activate, 180
  - addItem, 181
  - debugLabel, 181
  - getProperty, 182
  - propertySet, 183
  - setProperty, 183
  - setShortcutString, 184
  - shortcutChanged, 185
  - shortcutString, 185
  - stretchable, 185
- YDumbTabPrivate, 186
- YEmpty, 187
  - preferredHeight, 188
  - preferredWidth, 188
- YEmptyPrivate, 189
- YEnvVar, 189
- YEvent, 190
  - ~YEvent, 191
  - invalidate, 192
  - isValid, 192
  - item, 192
  - serial, 192
  - widget, 193
- YEventFilter, 193
  - ~YEventFilter, 195
  - filter, 196
  - YEventFilter, 194
- YEventFilterPrivate, 197
- YExternalWidgetFactory, 198
  - YExternalWidgetFactory, 198
- YExternalWidgets, 199
  - createExternalWidgetFactory, 200
  - externalWidgetFactory, 200
  - externalWidgets, 201
  - YExternalWidgets, 199
- YExternalWidgetsTerminator, 202
  - ~YExternalWidgetsTerminator, 202
- YFrame, 203
  - getProperty, 204
  - propertySet, 205
  - setLabel, 205
  - setProperty, 206
- YFramePrivate, 207

- YGraph, 208
  - activatedNode, 211
  - getProperty, 211
  - propertySet, 211
  - renderGraph, 212
  - setFilename, 212
  - setGraph, 213
  - setLayoutAlgorithm, 213
  - setProperty, 214
  - YGraph, 209, 210
- YGraphPlugin, 215
  - ~YGraphPlugin, 217
  - createGraph, 217
- YGraphPrivate, 218
- YHelpButtonHandler, 219
  - filter, 220
- YIconLoader, 221
- YImage, 221
  - hasZeroSize, 223
  - setAutoScale, 223
  - setImage, 223
  - setZeroSize, 224
  - YImage, 223
- YImagePrivate, 225
- YInputField, 226
  - getProperty, 228
  - inputMaxLength, 229
  - passwordMode, 229
  - propertySet, 229
  - saveUserInput, 230
  - setInputMaxLength, 231
  - setLabel, 231
  - setProperty, 231
  - setShortcutString, 232
  - setShrinkable, 233
  - setValidChars, 233
  - setValue, 234
  - shortcutString, 234
  - userInputProperty, 234
  - validChars, 235
  - value, 235
  - YInputField, 228
- YInputFieldPrivate, 236
- YIntField, 237
  - enforceRange, 239
  - getProperty, 240
  - propertySet, 240
  - setLabel, 241
  - setMaxValue, 241
  - setMinValue, 242
  - setProperty, 242
  - setShortcutString, 243
  - setValue, 244
  - setValueInternal, 244
  - shortcutString, 245
  - userInputProperty, 245
  - value, 245
  - YIntField, 239
- YIntFieldPrivate, 246
- YItem, 247
  - childrenBegin, 248
  - childrenEnd, 249
  - label, 249
  - parent, 249
  - setData, 250
  - setSelected, 250
  - setStatus, 250
  - status, 250
- YItemCustomStatus, 251
  - nextStatus, 252
  - setNextStatus, 252
  - textIndicator, 253
- YItemSelector, 253
  - activateItem, 256
  - customStatus, 256
  - cycleCustomStatus, 256
  - getProperty, 256
  - propertySet, 257
  - setItemStatus, 258
  - setProperty, 258
  - setVisibleItems, 259
  - updateCustomStatusIndicator, 259
  - userInputProperty, 260
  - usingCustomStatus, 260
  - validCustomStatusIndex, 260
  - visibleItems, 261
  - YItemSelector, 255
- YItemSelectorPrivate, 262
- YItemShortcut, 263
  - setShortcut, 264
- YKeyEvent, 265
  - ~YKeyEvent, 267
  - focusWidget, 267
  - YKeyEvent, 267
- YLabel, 268
  - debugLabel, 270
  - getProperty, 270
  - isHeading, 271
  - isOutputField, 271
  - layoutPass, 272
  - propertySet, 272
  - setAutoWrap, 273
  - setProperty, 273
  - setText, 274
  - setUseBoldFont, 275
  - widgetClass, 275
  - YLabel, 269
- YLabelPrivate, 276

- YLayoutBox, 277
  - countLayoutStretchChildren, 280
  - countStretchableChildren, 280
  - doResize, 280
  - findDominatingChild, 281
  - isLayoutStretch, 282
  - moveChild, 282
  - preferredHeight, 283
  - preferredSize, 283
  - preferredWidth, 284
  - setSize, 285
  - stretchable, 285
  - YLayoutBox, 279
- YLayoutBoxPrivate, 286
- YLogView, 287
  - displayLogText, 289
  - getProperty, 289
  - maxLines, 290
  - propertySet, 290
  - setLabel, 291
  - setMaxLines, 291
  - setProperty, 292
  - setShortcutString, 293
  - setVisibleLines, 293
  - shortcutString, 294
  - YLogView, 289
- YLogViewPrivate, 295
- YMacro, 296
  - setPlayer, 297
  - setRecorder, 297
- YMacroPlayer, 298
- YMacroRecorder, 298
  - recordMakeScreenShot, 299
- YMenuButton, 300
  - activateItem, 302
  - addItem, 302
  - addItems, 303
  - deleteAllItems, 304
  - findItem, 304, 305
  - findMenuItem, 306
  - getProperty, 307
  - itemAt, 308
  - propertySet, 308
  - rebuildMenuTree, 309
  - resolveShortcutConflicts, 309
  - setProperty, 310
  - YMenuButton, 302
- YMenuButtonPrivate, 311
- YMenuEvent, 312
  - ~YMenuEvent, 313
  - id, 313
  - item, 314
- YMenuItem, 314
  - ~YMenuItem, 316
- YMenuItem, 315
- YMultiLineEdit, 316
  - defaultVisibleLines, 318
  - getProperty, 318
  - inputMaxLength, 319
  - propertySet, 319
  - setDefaultVisibleLines, 320
  - setInputMaxLength, 320
  - setLabel, 320
  - setProperty, 321
  - setShortcutString, 322
  - setValue, 322
  - shortcutString, 323
  - userInputProperty, 323
  - value, 323
- YMultiLineEditPrivate, 324
- YMultiProgressMeter, 325
  - currentValue, 327
  - doUpdate, 327
  - getProperty, 327
  - propertySet, 328
  - setCurrentValue, 329
  - setProperty, 329
- YMultiProgressMeterPrivate, 331
- YMultiSelectionBox, 332
  - currentItem, 333
  - getProperty, 334
  - propertySet, 334
  - saveUserInput, 335
  - setCurrentItem, 336
  - setProperty, 336
  - setShrinkable, 337
  - userInputProperty, 337
- YMultiSelectionBoxPrivate, 338
- YOptionalWidgetFactory, 338
  - YOptionalWidgetFactory, 340
- YPackageSelector, 341
  - YPackageSelector, 342
- YPackageSelectorPlugin, 343
  - ~YPackageSelectorPlugin, 344
  - createPackageSelector, 344
- YPartitionSplitter, 345
  - getProperty, 348
  - propertySet, 348
  - setProperty, 349
  - setValue, 350
  - userInputProperty, 350
  - value, 350
  - YPartitionSplitter, 347
- YPartitionSplitterPrivate, 351
- YPath, 352
  - YPath, 353
- YPerThreadLogInfo, 354
- YPopupInternal, 355

- editNewStringArray, 355
- editStringArray, 356
- message, 356
- YProgressBar, 357
  - getProperty, 359
  - maxValue, 359
  - propertySet, 360
  - setLabel, 360
  - setProperty, 361
  - setValue, 362
- YProgressBarPrivate, 363
- YProperty, 363
  - YProperty, 364
- YPropertyEditor, 365
  - edit, 366
  - YPropertyEditor, 365
- YPropertyEditorPriv, 366
- YPropertySet, 367
  - add, 368
  - check, 368, 369
  - contains, 369
- YPropertyValue, 370
  - operator!=, 371
  - operator==, 371
  - stringVal, 372
  - type, 372
- YPushButton, 373
  - activate, 375
  - getProperty, 375
  - isDefaultButton, 375
  - isHelpButton, 376
  - isRelNotesButton, 376
  - propertySet, 376
  - setDefaultButton, 377
  - setFunctionKey, 378
  - setHelpButton, 378
  - setIcon, 379
  - setLabel, 379
  - setProperty, 379
  - setRelNotesButton, 380
  - setRole, 380
  - setShortcutString, 381
  - shortcutString, 382
- YPushButtonPrivate, 383
- YRadioButton, 384
  - findRadioButtonGroup, 386
  - getProperty, 386
  - propertySet, 387
  - saveUserInput, 388
  - setLabel, 388
  - setProperty, 389
  - setShortcutString, 390
  - setUseBoldFont, 390
  - setValue, 390
  - shortcutString, 390
  - userInputProperty, 391
  - value, 391
  - widgetClass, 391
- YRadioButton, 386
- YRadioButtonGroup, 392
  - addRadioButton, 394
  - getProperty, 394
  - propertySet, 395
  - radioButtonsBegin, 396
  - removeRadioButton, 396
  - setProperty, 396
  - uncheckOtherButtons, 397
- YRadioButtonGroupPrivate, 399
- YRadioButtonPrivate, 400
- YRelNotesButtonHandler, 401
  - filter, 402
- YReplacePoint, 403
  - showChild, 404
- YRichText, 405
  - autoScrollDown, 407
  - getProperty, 408
  - hScrollValue, 408
  - plainTextMode, 409
  - propertySet, 409
  - setAutoScrollDown, 409
  - setHScrollValue, 410
  - setPlainTextMode, 410
  - setProperty, 411
  - setShrinkable, 412
  - setValue, 412
  - setVScrollValue, 413
  - shrinkable, 413
  - vScrollValue, 413
  - YRichText, 407
- YRichTextPrivate, 414
- YRpmGroupsTree, 415
- YSelectionBox, 417
  - getProperty, 419
  - immediateMode, 419
  - propertySet, 419
  - setProperty, 420
  - setShrinkable, 421
  - userInputProperty, 421
- YSelectionBoxPrivate, 422
- YSelectionWidget, 423
  - addItem, 427
  - addItems, 428
  - deleteAllItems, 428
  - deselectAllItems, 429
  - findItem, 429, 430
  - findSelectedItem, 430
  - iconFullPath, 431
  - itemsBegin, 432



- itemsCount, [432](#)
- selectedItems, [432](#)
- selectItem, [433](#)
- setEnforceInitialSelection, [434](#)
- setEnforceSingleSelection, [434](#)
- setIconBasePath, [434](#)
- setItemStatus, [435](#)
- setLabel, [435](#)
- setShortcutString, [435](#)
- shortcutString, [436](#)
- YSelectionWidget, [426](#)
- YSelectionWidgetPrivate, [437](#)
- YSettings, [438](#)
  - loadedUI, [439](#)
  - setIconDir, [439](#)
  - setLocaleDir, [440](#)
  - setProgDir, [440](#)
  - setThemeDir, [440](#)
- YShortcut, [441](#)
  - clearShortcut, [443](#)
  - conflict, [443](#)
  - distinctShortcutChars, [444](#)
  - findShortcut, [444](#)
  - findShortcutPos, [444](#)
  - isValid, [445](#)
  - normalized, [445](#)
  - preferred, [445](#)
  - shortcut, [446](#)
  - shortcutString, [446](#)
- YShortcutManager, [447](#)
  - checkShortcuts, [449](#)
  - conflictCount, [450](#)
  - findShortestWidget, [450](#)
  - findShortestWizardButton, [450](#)
  - resolveAllConflicts, [451](#)
  - resolveConflict, [451](#)
- YSimpleEventHandler, [452](#)
  - ~YSimpleEventHandler, [453](#)
  - blockEvents, [454](#)
  - consumePendingEvent, [454](#)
  - deleteEvent, [454](#)
  - deletePendingEventsFor, [455](#)
  - pendingEvent, [455](#)
  - sendEvent, [456](#)
  - unblockEvents, [456](#)
- YSimpleInputField, [457](#)
  - getProperty, [459](#)
  - propertySet, [459](#)
  - setLabel, [460](#)
  - setProperty, [461](#)
  - setShortcutString, [462](#)
  - setValue, [462](#)
  - shortcutString, [463](#)
  - userInputProperty, [463](#)
  - value, [463](#)
- YSimpleInputFieldPrivate, [464](#)
- YSingleChildContainerWidget, [465](#)
  - preferredHeight, [466](#)
  - preferredWidth, [466](#)
  - setSize, [467](#)
  - stretchable, [467](#)
- YSingleChildManager< T >, [468](#)
  - add, [470](#)
- YSlider, [471](#)
  - YSlider, [472](#)
- YSliderPrivate, [473](#)
- YSpacing, [473](#)
  - dimension, [476](#)
  - preferredHeight, [476](#)
  - preferredWidth, [476](#)
  - size, [476](#), [477](#)
  - YSpacing, [475](#)
- YSpacingPrivate, [477](#)
- YSquash, [478](#)
  - stretchable, [480](#)
  - YSquash, [479](#)
- YSquashPrivate, [480](#)
- YStringTree, [481](#)
  - addBranch, [484](#)
  - completePath, [484](#)
  - origPath, [485](#)
  - path, [486](#)
  - root, [487](#)
  - setTextdomain, [487](#)
  - translate, [488](#)
  - translatedPath, [488](#)
  - YStringTree, [483](#)
- YStringWidgetID, [489](#)
  - isEqual, [490](#)
  - toString, [490](#)
- YTable, [491](#)
  - cellChanged, [494](#)
  - findItem, [494](#)
  - getProperty, [495](#)
  - hasColumn, [496](#)
  - immediateMode, [496](#)
  - keepSorting, [496](#)
  - propertySet, [497](#)
  - setKeepSorting, [497](#)
  - setProperty, [498](#)
  - setTableHeader, [499](#)
  - userInputProperty, [499](#)
  - YTable, [493](#)
- YTableCell, [500](#)
  - ~YTableCell, [501](#)
  - column, [502](#)
  - label, [502](#)
  - reparent, [502](#)

- setIconName, 502
- setLabel, 503
- setSortKey, 503
- YTableHeader, 504
  - hasColumn, 504
- YTableHeaderPrivate, 505
- YTableItem, 506
  - ~YTableItem, 508
  - addCell, 509
  - iconName, 509
  - label, 510
  - YTableItem, 508
- YTablePrivate, 511
- YTimeField, 512
- YTimeFieldPrivate, 513
- YTimeoutEvent, 514
  - ~YTimeoutEvent, 515
- YTimezoneSelector, 515
  - getProperty, 517
  - propertySet, 518
  - setProperty, 518
  - YTimezoneSelector, 517
- YTimezoneSelectorPrivate, 520
- YTransText, 520
  - setOrig, 521
  - trans, 521
- YTree, 522
  - activate, 524
  - addItems, 524
  - currentItem, 525
  - findItem, 525
  - getProperty, 526
  - immediateMode, 527
  - propertySet, 527
  - rebuildTree, 528
  - setProperty, 528
  - userInputProperty, 529
- YTreeItem, 530
  - ~YTreeItem, 532
  - addChild, 532
  - childrenBegin, 533
  - childrenEnd, 533
  - hasChildren, 533
  - isOpen, 533
  - parent, 534
  - YTreeItem, 531
- YTreePrivate, 534
- YUI, 535
  - \_eventsBlocked, 546
  - \_terminate\_ui\_thread, 546
  - app, 538
  - blockEvents, 538
  - builtinCaller, 538
  - createApplication, 538
  - createOptionalWidgetFactory, 539
  - createWidgetFactory, 539
  - deleteNotify, 539
  - ensureUICreated, 539
  - eventsBlocked, 540
  - idleLoop, 540
  - optionalWidgetFactory, 540
  - pipe\_from\_ui, 546
  - pipe\_to\_ui, 546
  - runPkgSelection, 541
  - sendWidgetID, 541
  - setBuiltinCaller, 542
  - setButtonOrderFromEnvironment, 542
  - shutdownThreads, 542
  - topmostConstructorHasFinished, 543
  - uiThreadDestructor, 543
  - uiThreadMainLoop, 544
  - unblockEvents, 544
  - widgetFactory, 545
- YUIBadPropertyArgException, 547
  - dumpOn, 548
- YUIButtonRoleMismatchException, 549
- YUICantLoadAnyUIException, 551
- YUIDialogStackingOrderException, 552
- YUIException, 553
  - log, 556
  - msg, 557
  - what, 557
  - YUIException, 556
- YUIIndexOutOfRangeException, 558
  - dumpOn, 560
  - YUIIndexOutOfRangeException, 559
- YUIInvalidChildException< YWidget >, 560
  - dumpOn, 562
- YUIInvalidDimensionException, 563
- YUIInvalidWidgetException, 564
- YUILoader, 566
  - deleteUI, 566
  - loadExternalWidgets, 566
  - loadPlugin, 567
  - loadUI, 567
- YUILog, 569
  - debug, 570
  - instance, 570
  - log, 570
  - logFileName, 571
  - loggerFunction, 571
  - setEnableDebugLoggingHooks, 572
  - setLogFileName, 572
  - setLoggerFunction, 573
- YUILogBuffer, 574
  - overflow, 575
  - writeBuffer, 575
  - xspuIn, 576

- YUILogPrivate, [577](#)
  - findCurrentThread, [578](#)
- YUINoDialogException, [578](#)
- YUINullPointerException, [579](#)
- YUIOutOfMemoryException, [581](#)
- YUIPlugin, [582](#)
  - ~YUIPlugin, [583](#)
  - locateSymbol, [583](#)
  - unload, [584](#)
- YUIPluginException, [585](#)
- YUIPropertyException, [586](#)
  - dumpOn, [587](#)
- YUIPropertyTypeMismatchException, [588](#)
  - dumpOn, [589](#)
- YUISetReadOnlyPropertyException, [590](#)
  - dumpOn, [592](#)
- YUISyntaxErrorException, [593](#)
- YUITooManyChildrenException< YWidget >, [594](#)
  - dumpOn, [596](#)
- YUIUnknownPropertyException, [596](#)
  - dumpOn, [598](#)
- YUIUnsupportedWidgetException, [599](#)
- YUIWidgetNotFoundException, [600](#)
- YWidget, [601](#)
  - addChild, [606](#)
  - begin, [606](#)
  - debugLabel, [607](#)
  - dumpDialogWidgetTree, [607](#)
  - end, [608](#)
  - findDialog, [608](#)
  - findWidget, [609](#)
  - firstChild, [609](#)
  - functionKey, [610](#)
  - getProperty, [610](#)
  - isValid, [611](#)
  - notify, [611](#)
  - notifyContextMenu, [612](#)
  - operator new, [612](#)
  - preferredHeight, [612](#)
  - preferredSize, [612](#)
  - preferredWidth, [613](#)
  - propertySet, [613](#)
  - removeChild, [614](#)
  - saveUserInput, [614](#)
  - sendKeyEvents, [615](#)
  - setBeingDestroyed, [615](#)
  - setChildrenManager, [616](#)
  - setDefaultStretchable, [616](#)
  - setEnabled, [616](#)
  - setFunctionKey, [617](#)
  - setHelpText, [617](#)
  - setId, [617](#)
  - setKeyboardFocus, [618](#)
  - setProperty, [618](#)
  - setShortcutString, [619](#)
  - setSize, [620](#)
  - setWidgetRep, [620](#)
  - shortcutString, [620](#)
  - startMultipleChanges, [621](#)
  - stretchable, [621](#)
  - userInputProperty, [621](#)
  - weight, [622](#)
- YWidgetEvent, [622](#)
  - ~YWidgetEvent, [623](#)
  - reason, [624](#)
  - widget, [624](#)
- YWidgetFactory, [625](#)
  - YWidgetFactory, [627](#)
- YWidgetID, [627](#)
  - toString, [628](#)
  - YWidgetID, [628](#)
- YWidgetPrivate, [629](#)
- YWidgetTreeItem, [630](#)
- YWizard, [631](#)
  - addMenu, [635](#)
  - addMenuEntry, [635](#)
  - addStep, [635](#)
  - addStepHeading, [635](#)
  - addTreeItem, [636](#)
  - backButton, [636](#)
  - contentsReplacePoint, [636](#)
  - deleteSteps, [636](#)
  - getProperty, [636](#)
  - propertySet, [637](#)
  - retranslateInternalButtons, [638](#)
  - setButtonLabel, [638](#)
  - setCurrentStep, [638](#)
  - setDialogIcon, [639](#)
  - setDialogTitle, [639](#)
  - YWizard, [634](#)
- YWizard.h
  - YWizardMode, [655](#)
  - YWizardMode\_Standard, [655](#)
  - YWizardMode\_Steps, [655](#)
  - YWizardMode\_TitleOnLeft, [655](#)
  - YWizardMode\_Tree, [655](#)
- YWizardMode
  - YWizard.h, [655](#)
- YWizardMode\_Standard
  - YWizard.h, [655](#)
- YWizardMode\_Steps
  - YWizard.h, [655](#)
- YWizardMode\_TitleOnLeft
  - YWizard.h, [655](#)
- YWizardMode\_Tree
  - YWizard.h, [655](#)
- YWizardPrivate, [639](#)