

widgetTools

October 25, 2011

basicPW-class *Class "basicPW", a basic class for primary widgets*

Description

This class defines the behavior shared by primary widget object used to build a GUI type interface

Objects from the Class

Objects can be created by calls of the form `new("basicPW", ...)`. Constructors have been defined to create objects of this class for specific widgets such as buttons, list boxes, ..

Slots

wName: Object of class "character" - a string for the name of the object

wType: Object of class "character" - a string defining the type of the primary widget. (e.g. button)

wValue: Object of class "ANY" - the initial value to be associated with the object

wWidth: Object of class "numeric" - an integer for the width of the object to be rendered (if applicable)

wHeight: Object of class "numeric" - an integer for the height of the object to be rendered (if applicable)

wFuns: Object of class "list" - a list of R functions to be executed before the widget is activated

wPreFun: Object of class "function" - a list of functions to be executed before the value of the widget to be updated

wPostFun: Object of class "function" - a list of functions to be executed before the value of the widget to be retrieved

wNotify: Object of class "list" - a list of functions to be executed each time when the value of the widget changes

wEnv: Object of class "environment" - an R environment object within which the value of the object is stored

wView: Object of class "widgetView" - a object of the class widgetView to which the widget is rendered

Methods

wEnv<- signature(object = "basicPW"): Set the value for wEnv slot
wEnv signature(object = "basicPW"): Get the value for wEnv slot
wFuns<- signature(object = "basicPW"): Set the value for wFuns slot
wFuns signature(object = "basicPW"): Get the value for wFuns slot
wHeight<- signature(object = "basicPW"): Set the value for wHeight slot
wHeight signature(object = "basicPW"): Get the value for wHeight slot
wName<- signature(object = "basicPW"): Set the value for wName slot
wName signature(object = "basicPW"): Get the value for wName slot
wNotify<- signature(object = "basicPW"): Set the value for wNotify slot
wNotify signature(object = "basicPW"): Get the value for wNotify slot
wPostFun<- signature(object = "basicPW"): Set the value for wPostFun slot
wPostFun signature(object = "basicPW"): Get the value for wPostFun slot
wPreFun<- signature(object = "basicPW"): Set the value for wPreFun slot
wPreFun signature(object = "basicPW"): Get the value for wPreFun slot
wType<- signature(object = "basicPW"): Set the value for wType slot
wType signature(object = "basicPW"): Get the value for wType slot
wValue<- signature(object = "basicPW"): Set the value for wValue slot
wValue signature(object = "basicPW"): Get the value for wValue slot
wView<- signature(object = "basicPW"): Set the value for wView slot
wView signature(object = "basicPW"): Get the value for wView slot
wWidth<- signature(object = "basicPW"): Set the value for wWidth slot
wWidth signature(object = "basicPW"): Get the value for wWidth slot

Author(s)

Jianhua Zhang

References

Programming with data

See Also

[widgetView-class](#), [widget-class](#)

Examples

```
# Create an R environment to store the values of primary widgets
PWEnv <- new.env(hash = TRUE, parent = parent.frame(1))

# Create a label
labell <- label(wName = "labell", wValue = "File Name: ", wEnv = PWEnv)

# Create an entry box with "Feed me using brows" as the default value
entry1 <- entryBox(wName = "entry1", wValue = "Feed me using browse",
                  wEnv = PWEnv)
```

 button

Functions to construct objects of primary widgets and render them

Description

All the primary widgets such as button, text box, and so on are objects of basicPW class. The functions are constructors of primary widgets that are subjects of basicPW class with behaviors specific to primary widgets.

Usage

```

button(wName, wEnv, wValue = "", wWidth = 12, wHeight = 0, wFuns = list(),
wNotify = list(), wPreFun = function(x) x, wPostFun = function(x) x,
wView = new("widgetView") )
entryBox(wName, wEnv, wValue = "", wWidth = 50, wHeight = 0, wFuns = list(),
wNotify = list(), wPreFun = function (x) x, wPostFun = function(x) x,
wView = new("widgetView"))
textBox(wName, wEnv, wValue = "", wWidth = 25, wHeight = 12, wFuns = list(),
wNotify = list(), wPreFun = function (x) x, wPostFun = function(x) x,
wView = new("widgetView"))
listBox(wName, wEnv, wValue = "", wWidth = 25, wHeight = 10, wFuns = list(),
wNotify = list(), wPreFun = function (x) x, wPostFun = function(x) x,
wView = new("widgetView"))
checkButton(wName, wEnv, wValue, wWidth = 50, wFuns = list(), wNotify =
list(), wPreFun = function (x) x, wPostFun = function(x) x,
wView = new("widgetView"))
radioButton(wName, wEnv, wValue, wWidth = 50, wFuns = list(), wNotify =
list(), wPreFun = function (x) x, wPostFun = function(x) x,
wView = new("widgetView"))
label(wName, wEnv, wValue = "", wWidth = 0, wHeight = 0, wFuns = list(),
wNotify = list(), wPreFun = function (x) x, wPostFun = function(x) x,
wView = new("widgetView"))
widget(wTitle, pWidgets, funs = list(), preFun = function()
print("Hello"), postFun = function() print("Bye"), env, defaultNames =c(
"Finish", "Cancel"))
widgetView(WVTitle, vName, widgetids = list(), theWidget = new("widget"),
winid)

```

Arguments

wName	wName a character string for the name to be associated with a given primary widget
vName	vName same as wName but for a widget object
wEnv	wEnv an R environment object within which the original values for each primary widget will be stored and updating and retrieval of the values will take place
env	env same as wEnv but for a widget object
wValue	wValue the initial values to be associated with a given primary widget
wWidth	wWidth an integer for the width of the primary widget (if applicable)
wHeight	wHeight an integer for the height of the primary widget (if applicable)

wFuns	wFuns a list of R functions that will be associated with a primary widget and invoked when an operation (e.g. click, get focus, ...) is applied to the primary widget
funcs	funcs same as wFuns but for a widget object
wNotify	wNotify a list of functions defining the actions to be performed when the value of the primary widget changes
wPreFun	wPreFun an R function that should be applied when the widget is activated
preFun	preFun same as wPreFun but for a view
wPostFun	wPostFun an R function that will be applied when the widget is inactivated
postFun	postFun same as wPostFun but for a view
wTitle	wTitle a character string for the title to be displayed when the widget is rendered
pWidgets	pWidget a list of primary widgets (e.g. button, list box, ...) to be rendered
WVTitle	WVTitle same as wTitle
widgetids	widgetids a list of tkwin ids for the primary widgets to be rendered
theWidget	theWidget a widget object to render the primary widgets
wView	wView an object of class widgetView
winid	winid an object of class winid
defaultNames	defaultName a vector of character string of length two for the text to be shown on the two default buttons. The first is to end the process and the second to abort the process

Details

[button](#) constructs a button widget object.

[button](#) constructs an entry box widget object.

[textBox](#) constructs a text box widget object.

[listBox](#) constructs a list box widget object. Value for a listbox object should be a named vector with names being the content to be shown in the list box and values being TRUE (default value) or FALSE.

[checkButton](#) constructs a group of check box widget objects. Value for check button objects should be a named vector with names being the content to be shown in the list box and values being TRUE (checked) or FALSE (not checked).

[radioButton](#) constructs a group of radio button widget objects. Value for radio button objects should be a named vector with names being the content to be shown in the list box and values being TRUE (default) or FALSE.

[label](#) constructs a text label widget object with the value displayed as the text.

[widget](#) constructs a widget object to render the primary widgets.

[widgetView](#) constructs a [widgetView](#) object. This class is for internal use by class [widget-class](#). Users trying to create GUI type widget do not need to use this class.

Value

Each constructor returns a tkwin object for the primary widget object.

Author(s)

Jianhua Zhang

References

R tcltk

See Also[widget-class](#), [basicPW-class](#)**Examples**

```
# Create an R environment to store the values of primary widgets
PWEEnv <- new.env(hash = TRUE, parent = parent.frame(1))

# Create a label
label1 <- label(wName = "label1", wValue = "File Name: ", wEnv = PWEEnv)

# Create an entry box with "Feed me using brows" as the default value
entry1 <- entryBox(wName = "entry1", wValue = "Feed me using browse",
                  wEnv = PWEEnv)

# Create a button that will call the function browse2Entry1 when
# pressed.
browse2Entry1 <- function(){
  tempValue <- tclvalue(tkgetOpenFile())
  temp <- get(wName(entry1), env = PWEEnv)
  wValue(temp) <- paste(tempValue, sep = "", collapse = ";")
  assign(wName(entry1), temp, env = PWEEnv)
}
button1 <- button(wName = "button1", wValue = "Browse",
                 wFuns = list(command = browse2Entry1), wEnv = PWEEnv)

# Create a list box with "Option1", "Option2", and "Option3" as the
# content and "Option1" selected
list1 <- listBox(wName = "list1", wValue = c(Option1 = TRUE, Option2 = FALSE,
      Option3 = FALSE), wEnv = PWEEnv)

# Create a text box with "Feed me something" displayed
text1 <- textBox(wName = "text1", wValue = "Feed me something",
                wEnv = PWEEnv)

# Create a set of radio buttons with "radiol" as the default
label2 <- label(wName = "label2", wValue = "Select one: ", wEnv = PWEEnv)
radios1 <- radioButton(wName = "radios1", wValue = c(radiol = TRUE,
      radio2 = FALSE, radio3 = FALSE), wEnv = PWEEnv)

# Create a set of check boxes with "check1" selected and "check2" and
# "check3" not selected
label3 <- label(wName = "label3", wValue = "Select one to many: ",
              wEnv = PWEEnv)
checks1 <- checkBox(wName = "checks1", wValue = c(check1 = TRUE,
      check22 = FALSE, check3 = FALSE), wEnv = PWEEnv)

# Please not that the name of the primary widget object (e.g. checks1)
```

```

# should be the same as the value of the name slot of the object
# (e. g. name = "checks1")

# Render the widgets
pWidgets <- list(topRow = list(label1 = label1, entry1 = entry1,
                             button1 = button1), textRow = list(list1 = list1,
                             text1 = text1), radGroup = list(label2 = label2,
                             radios1 = radios1), chkGroup = list(label3 = label3,
                             checks1 = checks1))

## Not run:
## These cannot be run by examples() but should be OK when pasted
## into an interactive R session with the widgetTools package loaded

aWidget <- widget(wTitle = "A test widget", pWidgets, funs = list(),
                  preFun = function() print("Hello"),
                  postFun = function() print("Bye"), env = PWEEnv)

## End(Not run)

```

dropdownList

A widget to mimic a dropdown list

Description

The current tcltk library does not support dropdown lists unless an extension is included. The function dropdownList provide an alternative.

Usage

```

dropdownList(base, options, textvariable, width = 10, default, editable
= FALSE)
getListOption(targetWidget, options, height, vScroll = FALSE)

```

Arguments

base	base a tkwin object that is the parent frame of the dropdown list to be created
options	options a vector of character strings for the content of the dropdown list
textvariable	textvariable a tclVar object to be associated with the selected item of the dropdown list
width	width an integer for the width in number of characters of the selection containing part of the dropdown list
default	default a character string for the default selection that is going to be shown in the selection containing window of the dropdown list
targetWidget	targetWidget a tkwin object for an entry box to which a button will be associated to make the look of a dropdown list
editable	editable a boolean indicating whether the dropdown list will be editable or not
height	height an integer for the height of the dropdown list box. If missing, height will be assigned the length of the options to be shown in the list box
vScroll	vScroll a boolean indicating whether a vertical scroll bar will be associated with the dropdown list box

Details

base can be a top window or a frame.

The widget returns a frame that contains a dropdown list. The frame need to be placed using any of the layout methods of tcltk. The value of the selection will be accessed through the tclVar object passed to the function.

getListOptions is called by dropdown list to get the selected item

Value

dropdownList returns a tkwin object for the frame that contains a dropdown list

getListOptions returns a character string for the selected item

Author(s)

Jianhua Zhang

References

tcltk

See Also

[tooltip](#)

Examples

```
## Not run:
## These cannot be run by examples() but should be OK when pasted
## into an interactive R session with the widgetTools package loaded

base <- tkoplevel()
selection <- tclVar()
dropdownList(base, c("Option 1", "Option 2", "Option 3"),
             selection, 15, "Option 2")
tclvalue(selection)
# Destroy toplevel widget
# tkdestroy(base)

## End(Not run)
```

makeViewer

Put a Scrollable List Box into a tkWidget.

Description

This function associates a tk listbox with a scroll bar and then puts them into a given tk widget.

Usage

```
makeViewer(target, vWidth = "", vHeight = "", hScroll = FALSE,
           vScroll = TRUE, what = "list", side = "left", text = "")
```

Arguments

target	tk widget that can accommodate a list box.
vWidth, vHeight	integers giving width and height of the listbox.
hScroll, vScroll	logicals indicating whether a horizontal or vertical scroll bar should be associated with the list box.
what	A character string indicating the type of the viewer to be put on a widget. Valid types include "list" for list box, "canvas", and "text" for text box
side	A character string for the geometry management of the viewer on the widget. Valid values include "left", "right", "top", and "bottom"
text	A character string to be displayed

Details

Tk list boxes (or canvas, text box) and scroll bars are separate widgets. This function provides a common interface to put them together and functionally associated.

Value

This function does not return any value.

Author(s)

Jianhua (John) Zhang

See Also

`tklistbox` (from the 'tcltk' package).

Examples

```
## Not run:
## These cannot be run by examples() but should be OK when pasted
## into an interactive R session with the widgetTools package loaded

# Create a top level window and put a list box in it
base <- tkoplevel()
listBox <- makeViewer(base)

# Destroy toplevel widget
# tkdestroy(base)

## End(Not run)
```

oneVScrList *A function that creates a groups of list boxes sharing a single*

Description

This function creates a group of list boxes what share a common vertical scroll bar. Values in all the list boxes scroll up or down when the scroll bar is dragged

Usage

```
oneVScrList(base, data)
```

Arguments

base	base a tkwin object that will be the container of the list boxes to be created
data	data a matrix with data to be put in the list boxes

Details

The matrix should have names for its columns. The names of the list boxes to be created will be the same as the corresponding columns of the matrix.

Data in the list boxes can be sorted based on values in any of the list boxes.

Value

This function returns a list containing the tkwin objects of the list boxes created.

Author(s)

Jianhua Zhang

References

tcltk

See Also

[dropdownList](#), [tooltip](#)

Examples

```
## Not run:
## These cannot be run by examples() but should be OK when pasted
## into an interactive R session with the widgetTools package loaded

testData <- matrix(c(1:50, 100:51), ncol = 2)
colnames(testData) <- c("Column 1", "Column 2")
base <- tkoplevel()
tt <- oneVScrList(base, testData)

# Destroy toplevel widget
# tkdestroy(base)

## End(Not run)
```

`safeFileOpen`*A function that checks to see if a connection can be made to a given*

Description

This function checks to see if a given file name exists. If so, the function returns a connection to the file. Otherwise, it returns "fileName does not exist".

Usage

```
safeFileOpen(fileName)
```

Arguments

`fileName` `fileName` a character string for the name of a file to which a connection is to be opened

Details

When this function is used, users have to make sure to check to see if the returned object inherits object "connection". Otherwise, the file does not exist or a connection has not been made.

Value

The function returns a connection object that inherits class "connection" if the file exists and is opened. Otherwise, the string "fileName does not exist"

Note

This function is not placed here to be used by various widgets. May be moved to a more suitable place later

Author(s)

Jianhua Zhang

See Also

[file](#)

Examples

```
write("A test file", "testFile4safeFileOpen")
tt <- safeFileOpen("testFile4safeFileOpen")
inherits(tt, "connection")
unlink("testFile4safeFileOpen")
tt <- safeFileOpen("testFile4safeFileOpen")
inherits(tt, "connection")
```

tooltip	<i>A tcltk widget to mimic a tooltip</i>
---------	--

Description

Current tcltk library does not support tooltip unless an extension is included. The function tooltip is implemented as an alternative.

Usage

```
tooltip(text, targetWidget, width = 350)
```

Arguments

text	text a character string for the content of the tooltip
targetWidget	targetWidget a tkwin object for the target tcltk widget to which a tool tip will be associated
width	width an integer for the width (in pixels) of the tooltip

Details

Given a target tcltk widget, a tooltip will be associated with the widget. The content of the tooltip will be shown when mouse moves over the widget and disappear when mouse moves out of the widget.

Value

This function returns invisible()

Author(s)

Jianhua Zhang

References

tcltk

See Also

[dropdownList](#)

Examples

```
## Not run:
## These cannot be run by examples() but should be OK when pasted
## into an interactive R session with the widgetTools package loaded

base <- tktoplevel()
but <- tkbutton(base, text = "Move Mouse Over Me")
tkpack(but)
tkbind(but, "<Enter>", expression(tooltip("Move mouse off me", but)))
```

```

# Destroy toplevel widget
# tkdestroy(base)

## End(Not run)

```

widget-class *Class "widget" creates a widget with primary widgets contained in the*

Description

This class takes a list of primary widgets and then creates a "widgetView" object that renders the primary widgets

Objects from the Class

Objects can be created by calls of the form `new("widget", ...)`.

Slots

wTitle: Object of class "character" - a character string for the title of the widget to be created

pWidgets: Object of class "list" - a list of "basicPW" objects representing widget elements to be rendered

env: Object of class "environment" - an R environment for the object to work within

funcs: Object of class "list" - a list of functions that will be associated with buttons on the widget to be rendered. The name of the function in the list will be the text appears on the button and the function will be executed when the button is pressed

preFun: Object of class "function" - a function that will be executed before the widget is constructed

postFun: Object of class "function" - a function that will be executed before the widget is destroyed

Methods

env<- signature(object = "widget"): set the value for env

wEnv signature(object = "widget"): get the value for env

funcs<- signature(object = "widget"): set the value for funcs

funcs signature(object = "widget"): get the value for funcs

postFuncs<- signature(object = "widget"): set the value for postFuncs

postFun signature(object = "widget"): get the value for postFuncs

preFuncs<- signature(object = "widget"): set the value for preFun

preFun signature(object = "widget"): get the value for preFun

pWidgets<- signature(object = "widget"): set the value for pWidgets

pWidgets signature(object = "widget"): get the value for pWidgets

updateCheck signature(object = "widget"): update the value of check buttons of the widget to be rendered

updateList signature(object = "widget"): update the value of list box/entry of the widget to be rendered

updateRadio signature(object = "widget"): update the value of radio buttons of the widget to be rendered

updateText signature(object = "widget"): update the value of text box of the widget to be rendered

wTitle<- signature(object = "widget"): set the value of wTitle

wTitle signature(object = "widget"): get the value of wTitle

Author(s)

Jianhua Zhang

References

Programming with data

See Also

[basicPW-class](#), [widgetView-class](#)

Examples

```
PWEnv <- new.env(hash = TRUE, parent = parent.frame(1))

label1 <- label(wName = "label1", wValue = "File Name: ", wEnv = PWEnv)
entry1 <- entryBox(wName = "entry1", wValue = "Feed me using browse",
                  wEnv = PWEnv)
browse2Entry1 <- function(){
  tempValue <- fileBrowser()
  temp <- get(wName(entry1), wEnv = PWEnv)
  wValue(temp) <- paste(tempValue, sep = "", collapse = ";")
  assign(wName(entry1), temp, env = PWEnv)
}
button1 <- button(wName = "button1", wValue = "Browse",
                  wFuns = list(command = browse2Entry1), wEnv = PWEnv)
list1 <- listBox(wName = "list1", wValue = c(Option1 = TRUE, Option2 = FALSE,
                                           Option3 = FALSE), wEnv = PWEnv)
text1 <- textBox(wName = "text1", wValue = "Feed me something",
                 wEnv = PWEnv)
label2 <- label(wName = "label2", wValue = "Select one: ", wEnv = PWEnv)
radios1 <- radioButton(wName = "radios1", wValue = c(radiol = TRUE,
                                                     radio2 = FALSE, radio3 = FALSE), wEnv = PWEnv)
label3 <- label(wName = "label3", wValue = "Select one to many: ",
               wEnv = PWEnv)
checks1 <- checkButton(wName = "checks1", wValue = c(check1 = TRUE,
                                                      check22 = FALSE, check3 = FALSE), wEnv = PWEnv)
pWidgets <- list(topRow = list(label1 = label1, entry1 = entry1,
                              button1 = button1), textRow = list(list1 = list1,
                                                                    text1 = text1), radGroup = list(label2 = label2,
                                                                    radios1 = radios1), chkGroup = list(label3 = label3,
                                                                    checks1 = checks1))

## Not run:
## These cannot be run by examples() but should be OK when pasted
```

```
## into an interactive R session with the widgetTools package loaded

aWidget <- widget(wTitle = "A test widget", pWidgets, funs = list(),
  preFun = function() print("Hello"),
  postFun = function() print("Bye"), env = PWEnv)

## End(Not run)
```

```
widgetView-class   Class "widgetView", a class for a GUI type widget holding widget
```

Description

"widgetView" renders element widgets

Objects from the Class

Objects can be created by calls of the form `new("widgetView", ...)`. This class is for internal use by class `widget-class`. Users trying to create GUI type widget do not need to use this class.

Slots

WVTitle: Object of class "character" - a character string that will be displayed as the title of the widget to be created

vName: Object of class "character" - a character string for the vName of the widget

winid: Object of class "tkwin" - a tkwin object for the id of the top window for the widget

widgetids: Object of class "list" - a list of tkwin ids for element widgets

theWidget: Object of class "widget" - a widget object that creates the widgetView

Methods

killWin signature (tkWidget = "widgetView"): destroys the window representing the widgetView

vName<- signature (object = "widgetView"): set the value for vName

vName signature (object = "widgetView"): get the value for vName

renderWidgets signature (widgetView = "widgetView", pWidgets = "list"): takes a list of "basicPW" objects (pWidgets) and renders them accordingly

renewView signature (widgetView = "widgetView", pWidgets = "list"): using values contained by the "basicPW" objects of pWidgets to update the values of widget elements displayed

theWidget<- signature (object = "widgetView"): set the value for theWidget

theWidget signature (object = "widgetView"): get the value for theWidget

updateDisplay signature (widgetView = "widgetView"): update the value of list box or text box element widgets

widgetids<- signature (object = "widgetView"): set the value of widgetids

widgetids signature (object = "widgetView"): get the value of widgetids

winid<- signature (object = "widgetView"): set the value of winid

winid signature(object = "widgetView"): set the value of winid
winWait signature(tkWidget = "widgetView"): make widgetView modal
WVTitle signature(object = "widgetView"): get the value for WVTitle

Author(s)

Jianhua Zhang

References

Programming with data

See Also

[widget-class,basicPW-class](#)

Examples

```
## Not run:
  ## These cannot be run by examples() but should be OK when pasted
  ## into an interactive R session with the widgetTools package loaded

widgetView <- widgetView(WVTitle = "demo", vName = "widget1")

## End(Not run)
```

writeText

Functions that read from and write to tcltk widgets

Description

These functions provide some of the common read and write operations for tcltk widgets

Usage

```
writeText(widget, value, clear = TRUE)
writeList(widget, value, clear = TRUE)
getListValue(which)
getTextValue(which)
getEntryValue(which)
```

Arguments

widget	widget a tkwin object for the tcltk widget to be read or written to
value	value the text of numerical value to be written to a tcltk widget
clear	clear a boolean to indicate whether a value will append to the existing one (FALSE)
which	which a tkwin object for the tcltk widget whose value will be retrieved

Details

`writeText` writes to a given tcltk text box widget.
`writeList` writes to a given tcltk list or entry box widget.
`getListValue` retrieves the selected value in a tcltk list widget.
`getTextValue` retrieves the value of a text box.
`getEntryValue` retrieves the value of an entry box.

Value

`getListValue` returns the selected value in a tcltk list widget.
`getTextValue` returns the value of a text box.
`getEntryValue` returns the value of an entry box.

Author(s)

Jianhua Zhang

References

R tcltk

See Also

[basicPW-class](#), [widget-class](#)

Examples

```
## Not run:
## These cannot be run by examples() but should be OK when pasted
## into an interactive R session with the widgetTools package loaded

# Create the widgets
base <- tkoplevel()
list <- tklistbox(base, width = 20, height = 5)

entry <- tkentry(base)
text <- tktext(base, width = 20, height = 5)
tkpack(list, entry, text)
# Write and read from the widgets
writeList(list, c("Option1", "Option2", "Option3"))
writeList(entry, "An Entry box")
writeText(text, "A text box")
# Will be NULL if not selected
getListValue(list)
getTextValue(text)
getEntryValue(entry)
# Destroy toplevel widget
# tkdestroy(base)

## End(Not run)
```


Index

*Topic classes

basicPW-class, 1
widget-class, 12
widgetView-class, 14

*Topic file

safeFileOpen, 10

*Topic interface

button, 3
makeViewer, 7
oneVScrlList, 9
writeText, 15

*Topic misc

dropdownList, 6
tooltip, 11

basicPW-class, 5, 13, 15, 16

basicPW-class, 1

button, 3, 4

checkButton, 4

checkButton (button), 3

dropdownList, 6, 9, 11

entryBox (button), 3

env<- (widget-class), 12

env<- , widget-method
(widget-class), 12

file, 10

funs (widget-class), 12

funs, widget-method
(widget-class), 12

funs<- (widget-class), 12

funs<- , widget-method
(widget-class), 12

getEntryValue, 16

getEntryValue (writeText), 15

getListOption (dropdownList), 6

getListValue, 16

getListValue (writeText), 15

getTextValue, 16

getTextValue (writeText), 15

killWin (widgetView-class), 14

killWin, widgetView-method
(widgetView-class), 14

label, 4

label (button), 3

listBox, 4

listBox (button), 3

makeViewer, 7

oneVScrlList, 9

postFun (widget-class), 12

postFun, widget-method
(widget-class), 12

postFuns<- (widget-class), 12

postFuns<- , widget-method
(widget-class), 12

preFun (widget-class), 12

preFun, widget-method
(widget-class), 12

preFuns<- (widget-class), 12

preFuns<- , widget-method
(widget-class), 12

pWidgets (widget-class), 12

pWidgets, widget-method
(widget-class), 12

pWidgets<- (widget-class), 12

pWidgets<- , widget-method
(widget-class), 12

radioButton, 4

radioButton (button), 3

renderWidgets (widgetView-class),
14

renderWidgets, widgetView, list-method
(widgetView-class), 14

renewView (widgetView-class), 14

renewView, widgetView, list-method
(widgetView-class), 14

safeFileOpen, 10

textBox, 4

- textBox (*button*), 3
- theWidget (*widgetView-class*), 14
- theWidget, widgetView-method
(*widgetView-class*), 14
- theWidget<- (*widgetView-class*), 14
- theWidget<-, widgetView-method
(*widgetView-class*), 14
- tklistbox, 8
- tooltip, 7, 9, 11

- updateCheck (*widget-class*), 12
- updateCheck, widget-method
(*widget-class*), 12
- updateDisplay (*widgetView-class*),
14
- updateDisplay, widgetView-method
(*widgetView-class*), 14
- updateList (*widget-class*), 12
- updateList, widget-method
(*widget-class*), 12
- updateRadio (*widget-class*), 12
- updateRadio, widget-method
(*widget-class*), 12
- updateText (*widget-class*), 12
- updateText, widget-method
(*widget-class*), 12

- vName (*widgetView-class*), 14
- vName, widgetView-method
(*widgetView-class*), 14
- vName<- (*widgetView-class*), 14
- vName<-, widgetView-method
(*widgetView-class*), 14

- wEnv (*basicPW-class*), 1
- wEnv, basicPW-method
(*basicPW-class*), 1
- wEnv, widget-method
(*widget-class*), 12
- wEnv<- (*basicPW-class*), 1
- wEnv<-, basicPW-method
(*basicPW-class*), 1
- wFuns (*basicPW-class*), 1
- wFuns, basicPW-method
(*basicPW-class*), 1
- wFuns<- (*basicPW-class*), 1
- wFuns<-, basicPW-method
(*basicPW-class*), 1
- wHeight (*basicPW-class*), 1
- wHeight, basicPW-method
(*basicPW-class*), 1
- wHeight<- (*basicPW-class*), 1

- wHeight<-, basicPW-method
(*basicPW-class*), 1
- widget, 4
- widget (*button*), 3
- widget-class, 2, 4, 5, 14–16
- widget-class, 12
- widgetids (*widgetView-class*), 14
- widgetids, widgetView-method
(*widgetView-class*), 14
- widgetids<- (*widgetView-class*), 14
- widgetids<-, widgetView-method
(*widgetView-class*), 14
- widgetView, 4
- widgetView (*button*), 3
- widgetView-class, 2, 13
- widgetView-class, 14
- winid (*widgetView-class*), 14
- winid, widgetView-method
(*widgetView-class*), 14
- winid<- (*widgetView-class*), 14
- winid<-, widgetView-method
(*widgetView-class*), 14
- winWait (*widgetView-class*), 14
- winWait, widgetView-method
(*widgetView-class*), 14
- wName (*basicPW-class*), 1
- wName, basicPW-method
(*basicPW-class*), 1
- wName<- (*basicPW-class*), 1
- wName<-, basicPW-method
(*basicPW-class*), 1
- wNotify (*basicPW-class*), 1
- wNotify, basicPW-method
(*basicPW-class*), 1
- wNotify<- (*basicPW-class*), 1
- wNotify<-, basicPW-method
(*basicPW-class*), 1
- wPostFun (*basicPW-class*), 1
- wPostFun, basicPW-method
(*basicPW-class*), 1
- wPostFun<- (*basicPW-class*), 1
- wPostFun<-, basicPW-method
(*basicPW-class*), 1
- wPreFun (*basicPW-class*), 1
- wPreFun, basicPW-method
(*basicPW-class*), 1
- wPreFun<- (*basicPW-class*), 1
- wPreFun<-, basicPW-method
(*basicPW-class*), 1
- writeList, 16
- writeList (*writeText*), 15
- writeText, 15, 16

wTitle (*widget-class*), 12
wTitle, widget-method
 (*widget-class*), 12
wTitle<- (*widget-class*), 12
wTitle<-, widget-method
 (*widget-class*), 12
wType (*basicPW-class*), 1
wType, basicPW-method
 (*basicPW-class*), 1
wType<- (*basicPW-class*), 1
wType<-, basicPW-method
 (*basicPW-class*), 1
wValue (*basicPW-class*), 1
wValue, basicPW-method
 (*basicPW-class*), 1
wValue<- (*basicPW-class*), 1
wValue<-, basicPW-method
 (*basicPW-class*), 1
wView (*basicPW-class*), 1
wView, basicPW-method
 (*basicPW-class*), 1
wView<- (*basicPW-class*), 1
wView<-, basicPW-method
 (*basicPW-class*), 1
WVTitle (*widgetView-class*), 14
WVTitle, widgetView-method
 (*widgetView-class*), 14
wWidth (*basicPW-class*), 1
wWidth, basicPW-method
 (*basicPW-class*), 1
wWidth<- (*basicPW-class*), 1
wWidth<-, basicPW-method
 (*basicPW-class*), 1