Package 'traitr'

February 20, 2015

Type Package

Title An interface for creating GUIs modeled in part after traits UI module for python.

Version 0.14

Author John Verzani <jverzani@gmail.com>

Maintainer John Verzani <jverzani@gmail.com>

URL https://r-forge.r-project.org/R/?group_id=761

Description An interface for creating GUIs modeled in part after the traits UI module for python. The basic design takes advantage of the model-view-controller design pattern. The interface makes basic dialogs quite easy to produce.

Depends digest, proto, gWidgets

Suggests testthat

License GPL (>= 2)

Collate 'helpers.R' 'base.R' 'model.R' 'view.R' 'controller.R' 'container.R' 'editor.R' 'itemgroup.R' 'dialog.R' 'items.R' 'itemlist.R' 'loadanimation.R' 'traitR-package.R'

NeedsCompilation no

Repository CRAN

Date/Publication 2014-09-01 16:10:50

R topics documented:

traitR-package 3
aContainer 3
aContext 4
aController 5
Adapter 5
aDialog 6
aFrame 8
aGroup 9

aModel . . . 10
anExpandGroup . . . 11
anItemGroup . . . 12
aNotebook . . . 13
aNotebookPage . . . 14
aPanedGroup . . . 15
aTableLayout . . . 16
aView . . . 17
BaseTrait . . . 17
BooleanEditor . . . 18
buttonItem . . . 19
choiceItem . . . 20
Container . . . 22
Controller . . . 23
dataframeItem . . . 24
dateItem . . . 25
dfEditItem . . . 26
Dialog . . . 26
dialogMaker . . . 28
Editor . . . 29
expressionItem . . . 31
fileItem . . . 32
formulaItem . . . 32
get_with_default . . . 33
graphicDeviceItem . . . 34
imageItem . . . 35
integerItem . . . 36
Item . . . 37
ItemGroup . . . 40
itemList . . . 42
ItemListEditor . . . 44
labelItem . . . 45
Model . . . 45
numericItem . . . 46
rangeItem . . . 47
separatorItem . . . 48
stringItem . . . 49
tableItem . . . 50
trueFalseItem . . . 51
variableSelectorItem . . . 52
View . . . 53
wrap_in_tag . . . 54

Index **56**

traitR-package *An interface for GUI creation using gWidgets*

Description

This package provides an alternate interface for creating graphical user interfaces. The design was inspired by the Traits UI module for python developed by enthought.com. The implementation uses the MVC design pattern in the background, although the user need not be aware of this.

Details

For basic use, the user creates a bunch of items (the model), specifies how these will be layed out in a simple manner (the view), specifies actions to happen (the controller) and then creates a dialog. See `aDialog` for examples.

Creating basic dialogs requires no actual GUI programming knowledge. One specifies the items by type of variable (`numericItem` or `stringItem`, say) and the "action" through a method call.

The package uses the **proto** package so at some level, the R user must use that OO syntax. In particular, methods calls are done with the notation `obj$method_name` and method definitions have an initial argument `.` for passing in a reference to the `proto` object. (The name `.` is a convention, but can be changed to `self` or `this`, if that naming convention is preferred. Methods for `proto` objects are documented (to some degree anyways). Their help page is shown through the method `show_help`, as in `obj$show_help()`.

aContainer *A container to give a different context than the default for a set of items*

Description

A container to give a different context than the default for a set of items

Usage

```
aContainer(..., context = NULL, attr = list(), enabled_when, visible_when)
```

Arguments

`context`	ItemGroup or item to get context from. Typically just NULL.
`attr`	gWidget values passed to constructor
`enabled_when`	Method to determine when items in container should be enabled
`visible_when`	Method to determine when items in container should be visible
`...`	children items specified by character strings

Value

Returns a `proto` object. Call `obj$show_help()` to view its methods and properties.

See Also

`Container`

Examples

```
i <- aDialog(items=list(x=numericItem(1), y=stringItem("a")))
lay <- aContainer("x","y")
## Not run: i$make_gui(gui_layout=lay)
## how to do enabled when
lay <- aContainer("x",
         aContainer("y", enabled_when=function(.) .$get_x() > 1))
j <- i$instance()
## Not run: j$make_gui(gui_layout=lay)
## visible can be used to hide values if not needed
i <- aDialog(items=list(x=numericItem(1), y=stringItem("a")))
lay <- aContainer("x","y")
## Not run: i$make_gui(gui_layout=lay)
## how to do enabled when
lay <- aContainer("x",
         aContainer("y", visible_when=function(.) .$get_x() > 1))
k <- i$instance()
## Not run: k$make_gui(gui_layout=lay)
```

---

aContext *A container to give a different context than the default for a set of items*

Description

The basic container uses the calling model (a dialog or item group) as its context. This allows the context to be overridden, which might be desirable if the items are in more than one dialog.

Usage

```
aContext(..., context, attr = list(), enabled_when, visible_when)
```

Arguments

`context`	ItemGroup or item to get context from. Typically just NULL.
`attr`	gWidget values passed to constructor
`enabled_when`	Method to determine when items in container should be enabled
`visible_when`	Method to determine when items in container should be visible
`...`	children items specified by character strings

Value

Returns a `proto` object. Call `obj$show_help()` to view its methods and properties.

See Also

`Container`

`aController` *Constructor for a Controller proto objects*

Description

Simply provides a more typical calling interface for the Controller proto object

Usage

```
aController(...)
```

Arguments

`...`	passed to proto method for

Value

returns the Controller object

`Adapter` *Trait for Adapter object An adapter is a simple controller connecting one model property with a widget in a view by default the adapter synchronizes changes*

Description

Trait for Adapter object An adapter is a simple controller connecting one model property with a widget in a view by default the adapter synchronizes changes

Usage

```
Adapter
```

Format

```
proto object
 $ handler_user_data: NULL
 $ class            : chr [1:3] "Adapter" "Controller" "TraitR"
 $ update_from_model:function (.)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 179 49 190 29 49 29 179 190
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266cf1948>
 $ init             :function (.)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 228 34 239 29 34 29 228 239
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266cf1948>
 $ remove_view      :function (.)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 224 41 227 29 41 29 224 227
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266cf1948>
 $ property         : NULL
 $ view_widget_name : NULL
 $ update_from_view :function (.)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 192 48 222 29 48 29 192 222
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266cf1948>
 $ add_handler_name : chr "addHandlerChanged"
 parent: proto object
 .. parent: proto object
```

`aDialog` *Create a Dialog instance*

Description

A dialog is like an item group, in that it combines items into a model. However, an item group is meant to be incorporated into other GUIS, whereas a dialog creates its own window and decorations. A dialog has default buttons, and options for adding in menubars, toolbars, and statusbars. The choice of buttons can be specified at construction.

Usage

```
aDialog(items = list(), title = "", help_string = "", buttons = c("OK",
  "SPACE", "Cancel", "Help"), ...)
```

Arguments

`items`	List of item instances to create the model for the dialog object. May also be an item group (`anItemGroup`).
`title`	Title of dialog
`help_string`	String for default Help button
`buttons`	Character vector of button names. "OK","Cancel","Help","Undo","Redo" are some standard ones. "SPACE" and "SPRING" adjust the layout.

... How to pass in other properties and methods of the dialog object. For example OK_handler.

Details

Methods:

The main method that a dialog has is its OK_handler which is a method called when the "OK" button is clicked (one of the default buttons).

The getters and setters for the main value for an item are get_NAME and set_NAME, where NAME is the item name. The name is specified when the item is constructed (through its name property) or more conveniently, taken from the name of the component in the items list that defines the items for dialog or item group.

The method to_R returns the items' values as a list (useful in combination with do.call).

The method get_item_by_name returns an item object by its name. (Names should be unique.) This is useful if more properties than the main one for an item are needed to be set. (The main value is set via the setter.) The example shows how the validate property of some items can be set.

The method is_valid is TRUE if all items validate and FALSE otherwise.

The method model_value_changed(.) is called whenever an item property is changed either through the GUI. A dialog observes itself.

For each item one can listen for changes with the method property_NAME_value_changed(., value, old_value).

Properties that are of interest:

1. status_text If non-NULL, when GUI is drawn, a status bar will be made with this text. The method set_status_text can be used to update the status
2. menu_list A menu list to specify a menubar. (See gmenu.)
3. toolbar_list A menu list to specify a toolbar. (See gtoolbar.)
4. buttons A list of buttons names. The default is c("OK", "SPACE", "Cancel", "Help"). The special names SPACE and SPRING adjust their positioning, otherwise the values are button names. When a button is clicked, the handler buttonname_handler is called, where the buttonname is stripped on non-alphanumeric characters. The basic buttons and Redo and Undo have default handlers. Likely, only OK_handler will need redefining. The property default_button can be specified to make a button the default one (so that it is activated when a user presses the enter key).

Value

Returns a proto object. See its show_help method for details.

Examples

```
##
## a Basic example
dlg <- aDialog(items=list(
                 a = numericItem(0),
                 b = stringItem("a")
                 ),
               title="The title",
```

```
               help_string="Help on this dialog"
               )
## Not run: dlg$make_gui()
##
##
## example with model_value_changed
plotIt <- function(n, mean, sd, ...) hist(rnorm(n, mean, sd))
dlg <- aDialog(items=list(
  n = integerItem(10),
  mean = numericItem(0),
  sd = numericItem(1),
  out=graphicDeviceItem()
),
buttons="Cancel",
model_value_changed=function(.) if(.$is_valid()) do.call("plotIt", .$to_R())
)
##
## validation for n, sd
n <- dlg$get_item_by_name("n")
n$validate <- function(., rawvalue) {
  if(rawvalue <= 1) stop("n must be positive integer") else rawvalue
}
sd <- dlg$get_item_by_name("sd")
sd$validate <- function(., rawvalue) {
  if(rawvalue <- 0) stop("sd must be positive") else rawvalue
}
## Not run: dlg$make_gui()
##
##
## subtle point about scope. Proto methods can be defined via $<- or [[<- but there is a difference.
## $<- does not have lexical scope whereas [[<- does. The $<- might be more natural to type,
## but [[<- might be more natural to use. In this example,
## The "b" button does not work, as it can't find the
## function a -- the frame of evaluation is the environment dlg (not its enclosing frame).
## Thanks to Gabor for his help with this.
scope_example <- function() {
a <- function(...) print("hi")
dlg <- aDialog(items=list(),
               buttons=c("a","b","c"),
               a_handler=function(.) a(),   ## like [[<-, not $<-
               title="a, c work; b doesn't"
               )
 dlg$b_handler <- function(.) a()  ## $<- has unbound variables found in dlg
 dlg[['c_handler']] <- a           ## [[<- uses lexical scope for unbound variables
}
## Not run: scope_example()
## See ?anItemGroup for an example of a modal dialog
```

---

aFrame *Box container with label and visual separator to indicate grouping*

Description

Box container with label and visual separator to indicate grouping

Usage

```
aFrame(..., label = "frame label", horizontal = FALSE, spacing = 10,
  context = NULL, attr = list(), enabled_when, visible_when)
```

Arguments

label	label for frame
horizontal	If TRUE left to right, if FALSE top to bottom
spacing	Space in pixels between items
context	ItemGroup or item to get context from. Typically just NULL.
attr	gWidget values passed to constructor
enabled_when	Method to determine when items in container should be enabled
visible_when	Method to determine when items in container should be visible
...	children items specified by character strings

Value

Returns a proto object. Call obj$show_help() to view its methods and properties.

See Also

Container

Examples

```
i <- aDialog(items=list(x=numericItem(1), y=stringItem("a")))
lay <- aFrame(label="label frame",
              aContainer("x","y"))
## Not run: i$make_gui(gui_layout=lay)
```

---

aGroup *A box container. Packs in items left to right or top to bottom*

Description

A box container. Packs in items left to right or top to bottom

Usage

```
aGroup(..., horizontal = TRUE, spacing = 10, context = NULL,
  attr = list(), enabled_when, visible_when)
```

Arguments

horizontal	If TRUE left to right, if FALSE top to bottom
spacing	Space in pixels between items
context	ItemGroup or item to get context from. Typically just NULL.
attr	gWidget values passed to constructor
enabled_when	Method to determine when items in container should be enabled
visible_when	Method to determine when items in container should be visible
...	children items specified by character strings

Value

Returns a proto object. Call obj$show_help() to view its methods and properties.

See Also

Container

Examples

```
i <- aDialog(items=list(xlong=numericItem(1), y=stringItem("a")))
lay <- aGroup("xlongname","y", horizontal=FALSE)  # not in nice layout
## Not run: i$make_gui(gui_layout=lay)
```

---

aModel *Constructor for a Model proto objects*

Description

Simply provides a more typical calling interface for the Model proto object

Usage

```
aModel(...)
```

Arguments

...	passed to proto method for

Value

returns the Model object. Call obj$show_help() to view its methods and properties.

anExpandGroup *Expanding group. Has trigger to show/hide its children*

Description

Expanding group. Has trigger to show/hide its children

Usage

```
anExpandGroup(..., label = "", horizontal = FALSE, expanded = TRUE,
  context = NULL, attr = list(), enabled_when, visible_when)
```

Arguments

label	label for trigger
horizontal	If TRUE left to right, if FALSE top to bottom
expanded	Initial state of children. Set to TRUE to show
context	ItemGroup or item to get context from. Typically just NULL.
attr	gWidget values passed to constructor
enabled_when	Method to determine when items in container should be enabled
visible_when	Method to determine when items in container should be visible
...	children items specified by character strings

Value

Returns a proto object. Call obj$show_help() to view its methods and properties.

See Also

Container

Examples

```
i <- aDialog(items=list(x=numericItem(1), y=stringItem("a")))
lay <- anExpandGroup(label="label frame",
                     aContainer("x","y"))
## Not run: i$make_gui(gui_layout=lay)
```

anItemGroup *Constructor for ItemGroup instances*

Description

An ItemGroup creates a model with properties given by the items and a default layout for its items. This can also be specified when the layout is drawn through make_gui.

Usage

```
anItemGroup(items = list(), name, ...)
```

Arguments

items	List of Item instances or ItemGroup instances
name	Name of ItemGroup.
...	Passed to ItemGroup proto trait

Details

An item group bundles a list of items into a model. When the model is intialized, constructors to access the model values are created. These getters/setters use the item names, so that get_name will get the main value for the item with name attribute "name".

An item group has the useful methods to_R to return the values in the model as a named list and get_item_by_name to get the item from the list of items matching the name.

Value

A proto object. Call obj$show_help() to view its methods and properties.

See Also

aContainer for specifying a layout

Examples

```
## Not run:
## make a simple item group, show in non-default layout
i <- anItemGroup(items=list(
                 numericItem(0,"x"),
                 numericItem(0,"y"),
                 stringItem("","z")
                 ))
lay <- aContainer("x","y", aFrame("z", label="z in a box"))
## some proto methods:
i$make_gui(cont=gwindow("Example of itemGroup"), gui_layout=lay)
i$get_x()     # get x value
i$set_x(10)   # set x value to 10
```

```
i$to_R()      # get list of x,y,z values

## End(Not run)

## example of using an item group and gbasicdialog to make a modal  GUI
ig <- anItemGroup(items=list(
                    x=numericItem(2)
                    )
                  )

## using gbasicdialog from gWidgets
## Not run:
w <- gbasicdialog("testing", handler=function(h,...) {
  . <- h$action                          # action passes in itemgroup
  .$output <- sin(.$get_x())
},
                  action=ig)
ig$make_gui(container=w)
visible(w, TRUE)  ## modal now
print(ig$output)

## End(Not run)
```

---

aNotebook *A notebook container.*

Description

Pages of notebook are made with aNotebookPage container, which in turn can hold other items.

Usage

```
aNotebook(..., close_buttons = FALSE, initial_page = 1, context = NULL,
  attr = list(expand = TRUE), enabled_when, visible_when)
```

Arguments

close_buttons	Logical indicating if close buttons should be added (RGtk2 only)
initial_page	Which page to open on
context	ItemGroup or item to get context from. Typically left as NULL.
attr	gWidget values passed to constructor
enabled_when	Method to determine when items in container should be enabled
visible_when	Method to determine when items in container should be visible
...	children items specified by character strings

Value

Returns a proto object. Call obj$show_help() to view its methods and properties.

See Also

Container

Examples

```
## Not run:
i <- aDialog(items=list(x=numericItem(1), y=stringItem("a")))
lay <- aNotebook(
          aNotebookPage(label="page 1", "x"),
          aNotebookPage(label="page 2", "y")
        )
i$make_gui(gui_layout=lay)

## End(Not run)
```

---

aNotebookPage *A page in a notebook*

Description

Container to hold a page within a notebook container

Usage

```
aNotebookPage(..., label, context = NULL, attr = list(), enabled_when,
  visible_when)
```

Arguments

label	Tab label for notebook page
context	ItemGroup or item to get context from. Typically just NULL.
attr	gWidget values passed to constructor
enabled_when	Method to determine when items in container should be enabled
visible_when	Method to determine when items in container should be visible
...	children items specified by character strings

Value

Returns a proto object. Call obj$show_help() to view its methods and properties.

aPanedGroup *A two panel paned group container.*

Description

A two panel paned group container.

Usage

```
aPanedGroup(..., horizontal = TRUE, context = NULL, attr = list(),
  enabled_when, visible_when)
```

Arguments

horizontal	If TRUE left to right, if FALSE top to bottom
context	ItemGroup or item to get context from. Typically just NULL.
attr	gWidget values passed to constructor
enabled_when	Method to determine when items in container should be enabled
visible_when	Method to determine when items in container should be visible
...	children items specified by character strings

Value

Returns a proto object. Call obj$show_help() to view its methods and properties.

See Also

Container

Examples

```
## Not run:
i <- aDialog(items=list(x=numericItem(1), y=stringItem("a")))
lay <- aPanedGroup("x","y") ## just two children,
i$make_gui(gui_layout=lay)
## can put other children into a container to make just two children for aPanedGroup instance
j <- aDialog(items=list(x=numericItem(1), y=stringItem("a"),
                        z=trueFalseItem(TRUE, label="check me")))
lay <- aPanedGroup("x", aContainer("y", "z"))
j$make_gui(gui_layout=lay)

## End(Not run)
```

aTableLayout *A container for tabular layout*

Description

The basic container has one column for the item's labels and one column for the item's editors.

Usage

```
aTableLayout(..., no_cols = 1, context = NULL, attr = list(),
  enabled_when, visible_when)
```

Arguments

no_cols	Number of columns. Fills in row by row.
context	ItemGroup or item to get context from. Typically just NULL.
attr	gWidget values passed to constructor
enabled_when	Method to determine when items in container should be enabled
visible_when	Method to determine when items in container should be visible
...	children items specified by character strings

Value

Returns a proto object. Call obj$show_help() to view its methods and properties.

See Also

aContainer constructor, Container base trait

Examples

```
## simple example
i <- aDialog(items=list(x=numericItem(1), y=stringItem("a")))
lay <- aTableLayout("x","y", no_cols=2)
## Not run: i$make_gui(gui_layout=lay)
```

aView *Constructor for a View proto object*

Description

Simply provides a more typical calling interface for the View proto object

Usage

```
aView(...)
```

Arguments

...	passed to proto method for

Value

Returns the View object. Call `obj$show_help()` to view its methods and properties.

BaseTrait *Base Trait to place common properties and methods*

Description

Base Trait to place common properties and methods

Usage

```
BaseTrait
```

Format

```
proto object
 $ traitr         : logi TRUE
 $ do_call        :function (., fun, lst = list())
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 231 31 236 21 31 21 231 236
  .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266aff340>
 $ show_help      :function (.)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 269 33 297 21 33 21 269 297
  .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266aff340>
 $ list_objects   :function (., all.names = FALSE)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 153 35 177 20 35 20 153 177
  .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266aff340>
 $ append         :function (., name, value, key)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 68 28 79 21 28 21 68 79
  .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266aff340>
```

```
$ new            :function (., ...)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 44 27 48 21 27 21 44 48
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266aff340>
$ list_properties:function (., all.names = FALSE)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 186 36 186 100 36 100 186 186
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266aff340>
$ class          : chr "TraitR"
$ has_slot       :function (., key)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 102 29 105 20 29 20 102 105
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266aff340>
$ next_method    :function (., meth_name)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 199 34 222 20 34 20 199 222
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266aff340>
$ has_local_slot :function (., key)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 109 35 111 20 35 20 109 111
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266aff340>
$ create_doc_list:function (.)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 239 39 263 21 39 21 239 263
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266aff340>
$ add_class      :function (., newclass)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 37 33 37 85 33 85 37 37
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266aff340>
$ get_slot       :function (., key)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 118 29 123 20 29 20 118 123
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266aff340>
$ set_slot       :function (., key, value, initialize = TRUE)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 140 29 147 20 29 20 140 147
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266aff340>
$ get_local_slot :function (., key)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 128 37 133 20 37 20 128 133
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266aff340>
$ list_methods   :function (., all.names = FALSE)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 191 33 191 94 33 94 191 191
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266aff340>
$ is             :function (., class = NULL)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 89 26 94 21 26 21 89 94
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266aff340>
$ assign_if_null :function (., key, value)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 55 35 59 20 35 20 55 59
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266aff340>
```

`BooleanEditor` *Trait for Editor for TRUE/FALSE selection*

Description

Editor has regular or compact style

Usage

```
BooleanEditor
```

Format

```
proto object
 $ class          : chr [1:4] "BooleanEditor" "Editor" "View" "TraitR"
 $ editor_name    : chr "gcombobox"
 $ make_ui_compact:function (., container, attr = .$attr, context, ...)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 380 47 383 31 47 31 380 383
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc2651fc120>
 $ make_ui        :function (., container, attr = .$attr, context, ...)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 376 39 379 31 39 31 376 379
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc2651fc120>
 parent: proto object
 .. parent: proto object
 .. .. parent: proto object
```

---

buttonItem *Button item to initiate an action*

Description

While dialogs have a buttons property for the main buttons, this item allows other buttons to be used within a dialog. One must define an action (a callback) to call when the button is clicked. There are some issues with how this method is defined and where it is evaluated.

Usage

```
buttonItem(value = "button label", action = NULL, name, label = name,
  help = "", tooltip = "", attr, model, editor, ...)
```

Arguments

value	Default value for the model
action	function to call when clicked. Signature is function(., h, ...) {} (like gWidgets with extra leading .). The "." is the button item, not the itemgroup or dialog that this item may be a part of. When that is the case, .$parent refers to the parent itemgroup or dialog. The evaluation environment is not that where the action is defined. This can lead to unexpected sources of error.
name	Required name for object. Names should be unique within a group of items
label	Optional label, default value is the name. Use "" to have not label text.
help	Optional help string
tooltip	Optional tooltip to display
attr	A list of attributes to pass to widget on construction. Eg. attr=list(size=c(100,200))

model	Optional model. Useful if one wishes to use same model for multiple views
editor	Specification of editor (a view) to override default
...	Passed to parent proto object during call to proto

Value

A proto object. Call obj$show_help() to view its methods and properties.

See Also

Item

Examples

```
## basic button. Note the extra "." compared to gWidgets handler
b <- buttonItem("click me", action=function(.,h,...) {
       print("hi")

     })
## An example within a dialog
dlg <- aDialog(items=list(
                a = stringItem(""),
                b = buttonItem("Click me", label="", action=function(., h, ...) {
                  galert(sprintf("Item a is \%s\n", .$parent$get_a()))
                })
                ),
              title="A dialog with a button item",
              buttons=c()               # no standard buttons
              )
   ## Not run: dlg$make_gui()
```

---

choiceItem *Item for choosing one of several values*

Description

Item for choosing one of several values

Usage

```
choiceItem(value = "", values = "", by_index = FALSE, multiple = FALSE,
  editable = FALSE, name, label = name, help = "", tooltip = "", attr,
  model, editor, editor_type = c("", "gradio", "gcombobox", "gtable", "gedit",
  "gcheckboxgroup"), ...)
```

Arguments

value	Default value for the model. This is specified by index (or indices if multiple=TRUE)
values	Values that one can select from. May be a data frame or vector. The editor depends on the size of this: small will be radio button or checkboxes; medium is combobox; large is a table. One can override the behaviour by passing in a value to editor_type.
by_index	Do we get and set the main value by index or by value?
multiple	Multiple selection is allowed? If so, then only checkboxes or table widget is used
editable	Can user edit value to be selected? If so, the combobox is used
name	Required name for object. Names should be unique within a group of items
label	Optional label, default value is the name
help	Optional help string
tooltip	Optional tooltip to display
attr	A list of attributes to pass to widget on construction. Eg. attr=list(size=c(100,200)). The expand=TRUE value is a default for this.
model	Optional model. Useful if one wishes to use same model for multiple views
editor	Specification of editor (a view) to override default
editor_type	overide choice of editor by heuristic based on the number of possible values. Must set attr to match desired.
...	Passed to parent proto object during call to proto

Value

A proto object. Call obj$show_help() to view its methods and properties.

See Also

Item

Examples

```
## default is to get/set by value
        a <- choiceItem("a", letters, name="x")
        a$get_x()
        a$set_x("b")
        a$get_x()
## or by index, which can be easier to do
        b <- choiceItem("a", letters, name="x", by_index=TRUE)
        b$get_x()
        b$set_x(2)
        b$get_x()
## Size determines widget, unless you set editor_type
## a radio group
        rg <- choiceItem("a", letters[1:3], name="x")
## a combobox
```

```
        cb <- choiceItem("a", letters[1:8], name="x")
## a table
        tb <- choiceItem("a", letters[1:26], name="x")
## adjust size of table widget
      tb <- choiceItem("a", letters[1:26], name="x", attr=list(size=c(width=300,height=400)))
## Multiple and size determines widget type
## smaller uses checkboxgroup
        cbg <- choiceItem("a", letters[1:5], multiple=TRUE)
## larger uses table
        tbl <- choiceItem("a", letters[1:15], multiple=TRUE)
## place values in data frame to avoid generic header
       tbl <- choiceItem("a", data.frame("Column header"=letters[1:15]), multiple=TRUE)
```

`Container` *Base Trait for Container objects. Containers are used to make views.*

Description

Basic container is a glayout object for tabular layout There are various types of layouts. The most basic, and default view, is simply `aContainer(...names of items...)` which simply uses a table to display the item's label and editor. Other containers can be used to adjust this.

Usage

```
Container
```

Format

```
proto object
 $ attr          :List of 1
 $ class         : chr [1:2] "Container" "TraitR"
 $ make_ui       :function (., container, attr = .$attr, context, ...)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 124 40 171 31 40 31 124 171
  .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266c4e4a0>
 $ init_ui       :function (.)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 115 38 115 51 38 51 115 115
  .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266c4e4a0>
 $ context       : NULL
 $ validate_ui   :function (.)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 199 43 199 56 43 56 199 199
  .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266c4e4a0>
 $ is_realized   :function (.)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 48 43 48 100 43 100 48 48
  .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266c4e4a0>
 $ type          : chr "glayout"
 $ visible_when  :function (.)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 190 46 190 65 46 65 190 190
  .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266c4e4a0>
```

```
$ instance      :function (.)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 236 39 248 30 39 30 236 248
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266c4e4a0>
$ enabled       :function (., value)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 182 41 184 31 41 31 182 184
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266c4e4a0>
$ no_cols       : num 1
$ update_ui     :function (.)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 204 43 232 31 43 31 204 232
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266c4e4a0>
$ make_container:function (., container, attr = .$attr)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 50 46 111 31 46 31 50 111
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266c4e4a0>
$ enabled_when  :function (.)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 177 46 180 31 46 31 177 180
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266c4e4a0>
$ children      : list()
$ container     : NULL
$ visible       :function (., value)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 192 41 194 31 41 31 192 194
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266c4e4a0>
parent: proto object
```

Details

Containers have a few methods, notably the `is_visible` and `is_enabled` methods which can be used to hide or set sensitive to user input the container's components.

`Controller` *Trait for Controller objects*

Description

A controller connects a model and an associated view to synchronize changes in one with another This implementation rests on the controller having some suitably named methods

Usage

```
Controller
```

Format

```
proto object
 $ get_model        :function (.)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 43 42 43 60 42 60 43 43
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266cf1948>
 $ remove_view      :function (.)
```

```
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 76 44 78 32 44 32 76 78
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266cf1948>
$ class            : chr [1:2] "Controller" "TraitR"
$ view             : NULL
$ update_from_model:function (.)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 85 50 85 63 50 63 85 85
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266cf1948>
$ get_view         :function (.)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 61 41 61 58 41 58 61 61
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266cf1948>
$ set_model        :function (., model)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 48 44 55 32 44 32 48 55
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266cf1948>
$ adapters         : list()
$ init             :function (.)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 119 39 144 32 39 32 119 144
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266cf1948>
$ model            : NULL
$ register_adapters:function (.)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 97 52 112 32 52 32 97 112
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266cf1948>
$ update_from_view :function (.)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 91 51 91 64 51 64 91 91
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266cf1948>
$ set_view         :function (., view)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 67 43 74 32 43 32 67 74
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266cf1948>
parent: proto object
```

`dataframeItem` *Item to select a data frame from the available data frames in .GlobalEnv*

Description

This widget checks every second or so for new data frames and updates selection accordingly

Usage

```
dataframeItem(value = "", name, label = name, help = "", tooltip = "",
  attr, model, editor, ...)
```

Arguments

`value`	Default data frame for the model, defaults to .GlobalEnv
`name`	Required name for object. Names should be unique within a group of items
`label`	Optional label, default value is the name

help	Optional help string
tooltip	Optional tooltip to display
attr	A list of attributes to pass to widget on construction. Eg. attr=list(size=c(100,200)) This widget uses a gtable instance and specifying the size is suggested
model	Optional model. Useful if one wishes to use same model for multiple views
editor	Specification of editor (a view) to override default
...	Passed to parent proto object during call to proto

Value

A proto object. Call obj$show_help() to view its methods and properties.

---

dateItem *A calendar date selection item*

Description

A calendar date selection item

Usage

```
dateItem(value = "", format_string, name, label = name, help = "",
  tooltip = "", attr, model, editor, ...)
```

Arguments

value	Default data frame for the model
format_string	String to specify format of date to return. See strftime for codes. default value is '%Y-%m-%d'
name	Required name for object. Names should be unique within a group of items
label	Optional label, default value is the name
help	Optional help string
tooltip	Optional tooltip to display
attr	A list of attributes to pass to widget on construction. Eg. attr=list(size=c(100,200))
model	Optional model. Useful if one wishes to use same model for multiple views
editor	Specification of editor (a view) to override default
...	Passed to parent proto object during call to proto

Value

A proto object. Call obj$show_help() to view its methods and properties.

See Also

Item

Examples

```
d <- dateItem(name="d") ## basic usage, no initial date.
# specify intial date and reformat -- can't start in that format, it is amibiguous
d <- dateItem('2000-12-25', format_string='\%m-\%d-\%Y', name='d')
```

---

dfEditItem *data frame editor item. Needs writing*

Description

Write me

Usage

```
dfEditItem(...)
```

Arguments

...	to be replaced with actual arguments

Value

A proto object. Call obj$show_help() to view its methods and properties.

Dialog *A Dialog wraps a top-level window around a collection of items which may be item groups*

Description

One can specify the parent when making the UI Buttons are specified through the buttons property

Usage

```
Dialog
```

Format

```
proto object
 $ make_gui        :function (., gui_layout = .$make_default_gui_layout(), parent = NULL, visible = TR
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 127 36 214 27 36 27 127 214
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc2668d25f8>
 $ OK_handler        :function (.)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 80 38 82 27 38 27 80 82
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc2668d25f8>
 $ Redo_handler      :function (.)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 102 40 102 59 40 59 102 102
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc2668d25f8>
 $ doc_Cancel_handler: chr "<p class=description>Description: Handler called when 'Cancel' button is c
 $ default_button    : NULL
 $ update_ui         :function (.)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 233 37 243 27 37 27 233 243
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc2668d25f8>
 $ doc_OK_handler    : chr "<p class=description>Description: Handler called when 'OK' button is clicke
 $ items             : list()
 $ status_text       : NULL
 $ doc_default_button: chr "<p class=description>Description: Name of default button. Leave empty (<co
 $ Undo_handler      :function (.)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 98 40 98 59 40 59 98 98
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc2668d25f8>
 $ Help_handler      :function (.)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 92 42 94 27 42 27 92 94
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc2668d25f8>
 $ class             : chr [1:4] "Dialog" "ItemGroup" "Model" "TraitR"
 $ toolbar_list      : NULL
 $ on_realized       :function (.)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 104 41 106 27 41 27 104 106
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc2668d25f8>
 $ visible           :function (., value = TRUE)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 228 35 231 27 35 27 228 231
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc2668d25f8>
 $ title             : chr "Dialog title"
 $ close_gui         :function (.)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 219 39 222 27 39 27 219 222
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc2668d25f8>
 $ doc_Help_handler  : chr "<p class=description>Description: Handler called when 'Help' button is cli
 $ doc_Redo_handler  : chr "<p class=description>Description: Handler called when 'Redo' button is cli
 $ Cancel_handler    :function (.)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 86 44 88 27 44 27 86 88
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc2668d25f8>
 $ doc_Undo_handler  : chr "<p class=description>Description: Handler called when 'Undo' button is cli
 $ buttons           : chr [1:4] "OK" "Cancel" "SPACE" "Help"
 $ get_widget        :function (., key)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 112 38 112 73 38 73 112 112
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc2668d25f8>
```

```
$ widget_list       : list()
$ menu_list         : NULL
$ help_string       : chr ""
parent: proto object
.. parent: proto object
.. .. parent: proto object
```

---

dialogMaker *Automatically create a dialog for a function*

Description

Function must have a special markup for its argument. A named argument a..b is interpreted with b determining the type of item to use. We support numeric, string, choice, range, ??? Within the body of the function, the variable a..b should be referred to by a. The idea is that you write and debug the function as usual, then simply modify the argument list to include the types. This function will not work for functions whose arguments use lazy evaluation referring to other argument's values.

Usage

```
dialogMaker(f, title = "Dialog", help_string = "", make_gui = TRUE,
  add_graphic_device = FALSE, ...)
```

Arguments

f	function to make dialog for. Its arguments must be specified in a certain way.
title	Title for dialog window
help_string	String for help information
make_gui	If TRUE or add_graphic_device=TRUE then call dialogs make_gui method
add_graphic_device	If TRUE add an graphicDeviceItem to dialog
...	passed to make_gui when no graphic device asked for

Details

All arguments should have a default A choice items should have its default with a vector. The first argument is the selected one A range item is specified with values c(from=., to=..., by=..., [value=from]). If value not give, then from is used. The OK_handler will call f.

Value

Returns an instance of aDialog.

Examples

```
f <- function(x..numeric=1, y..string="a") print(list(x,y))
## Not run: dialogMaker(f)
## can have missing arguments
f <- function(x, y..numeric=1) print(list(x,y))
## Not run: dialogMaker(f)
## a choice item. Sizing is funny for tables
f <- function(x..choice=letters) print(x)
## Not run: dialogMaker(f)
## range items
f <- function(x..numeric=0, mu..numeric=0,
              alternative..choice=c("two.sided","less","greater"),
              conf.level..range=c(.80,1.00, .01, .95)) {
              out <- capture.output(t.test(x, alt=alternative, conf.level=conf.level))
              print(out)
}
## Not run: dialogMaker(f, title="CI from t.test with summarized values")
```

`Editor` *Base Trait for Editor.*

Description

An editor is a basic view for a widget, essentially a map from gedit, say, to a view.

A Base Trait for an editor using the entry widget

editor has regular or compact style

Trait for making a range editor (slider, spinbox)

Trait for button editor

Trait for embedding an image file

Trait for embedding graphics (Qt, RGtk2 only)

Trait for making File browser editor

Trait for data selection editor

No selection, just display. For selection use ChoiceItem

Trait for a label

Trait for making a visual separator

Usage

```
Editor

EntryEditor

ObjectWithValuesEditor
```

RangeEditor

ButtonEditor

ImageEditor

GraphEditor

FileBrowseEditor

DateEditor

TableEditor

LabelEditor

SeparatorEditor

Format

```
proto object
 $ attr               :List of 1
 $ class              : chr [1:3] "Editor" "View" "TraitR"
 $ make_ui            :function (., container, attr = .$attr, context = ., ...)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 52 32 66 22 32 22 52 66
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc2651fc120>
 $ editor_name        : chr "gedit"
 $ set_valid          :function (.)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 187 32 187 45 32 45 187 187
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc2651fc120>
 $ instance           :function (.)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 179 31 183 22 31 22 179 183
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc2651fc120>
 $ enabled            :function (., value)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 134 32 136 22 32 22 134 136
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc2651fc120>
 $ view_widget_name   : chr "editor"
 $ editor_style       : NULL
 $ set_value_in_view  :function (., widget_name = .$view_widget_name, value)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 162 42 174 22 42 22 162 174
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc2651fc120>
 $ set_invalid        :function (., mesg)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 191 34 194 22 34 22 191 194
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc2651fc120>
 $ get_value_from_view:function (.)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 142 44 152 22 44 22 142 152
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc2651fc120>
 $ visible            :function (., value)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 131 32 133 22 32 22 131 133
```

```
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc2651fc120>
parent: proto object
.. parent: proto object
```

`expressionItem` *Item for typing in R expressions. These are eval-parsed in .GlobalEnv prior to return*

Description

Item for typing in R expressions. These are eval-parsed in .GlobalEnv prior to return

Usage

```
expressionItem(value = "", name, label = name, help = "", tooltip = "",
  attr, model, editor, ...)
```

Arguments

`value`	Default value for the model
`name`	Required name for object. Names should be unique within a group of items
`label`	Optional label, default value is the name
`help`	Optional help string
`tooltip`	Optional tooltip to display
`attr`	A list of attributes to pass to widget on construction. Eg. attr=list(size=c(100,200))
`model`	Optional model. Useful if one wishes to use same model for multiple views
`editor`	Specification of editor (a view) to override default
`...`	Passed to parent proto object during call to proto

Value

A `proto` object. Call `obj$show_help()` to view its methods and properties.

See Also

`numericItem`, `integerItem`, `stringItem`, as these are similar, but also validate the final results

fileItem *A file selection item*

Description

A file selection item

Usage

```
fileItem(value = "", name, label = name, help = "", tooltip = "", attr,
  model, editor, ...)
```

Arguments

value	Default data frame for the model
name	Required name for object. Names should be unique within a group of items
label	Optional label, default value is the name
help	Optional help string
tooltip	Optional tooltip to display
attr	A list of attributes to pass to widget on construction. Eg. attr=list(size=c(100,200))
model	Optional model. Useful if one wishes to use same model for multiple views
editor	Specification of editor (a view) to override default
...	Passed to parent proto object during call to proto

Value

A proto object. Call obj$show_help() to view its methods and properties.

formulaItem *A formula Item*

Description

A formula Item

Usage

```
formulaItem(value = "", dataframeItem, name, label = name, help = "",
  tooltip = "", attr, model, editor, ...)
```

Arguments

value	Default data frame for the model
dataframeItem	A required dataframeItem instance. This need not be in same display, or even displayed
name	Required name for object. Names should be unique within a group of items
label	Optional label, default value is the name
help	Optional help string
tooltip	Optional tooltip to display
attr	A list of attributes to pass to widget on construction. Eg. attr=list(size=c(100,200)) This widget uses a gtable instance and specifying the size is suggested
model	Optional model. Useful if one wishes to use same model for multiple views
editor	Specification of editor (a view) to override default
...	Passed to parent proto object during call to proto

Value

A proto object. Call obj$show_help() to view its methods and properties.

---

get_with_default *Function to return value or an object (or default if value is null, NA or "")*

Description

Function to return value or an object (or default if value is null, NA or "")

Usage

```
get_with_default(x, default)
```

Arguments

x	object
default	default value

Value

Returns default if x is NA, null or "", otherwise x

graphicDeviceItem *A graphic device item. (Only with RGtk2 and cairoDevice!)*

Description

This device will become the current one if the mouse clicks in the window, This isn't perfect, but should be easy enough to get used to. This only works with gWidgetsRGtk2, gWidgetsQt

Usage

```
graphicDeviceItem(value = "", name, label = name, help = "",
  tooltip = "", attr = list(size = c(480, 480)), model, editor, ...)
```

Arguments

value	ingored
name	Required name for object. Names should be unique within a group of items
label	Optional label, default value is the name
help	Optional help string
tooltip	Optional tooltip to display
attr	A list of attributes to pass to widget on construction. Eg. attr=list(size=c(100,200)).
model	ignored
editor	ignored
...	Passed to parent proto object during call to proto

Value

A proto object. Call obj$show_help() to view its methods and properties.

Note

With **gWidgetsRGtk2**, there is some thing odd that causes a display to pop up before the cairo Device if no devices are open.

See Also

Item

Examples

```
graphIt <- function(n, ...) hist(rnorm(n))
dlg <- aDialog(items=list(n=integerItem(10), out=graphicDeviceItem()),
model_value_changed=function(.) do.call("graphIt", .$to_R()) ## ... allows out to pass in unnoticed
)
## Not run: dlg$make_gui()
graphIt(dlg$get_n()) ## initial graphic

## End(Not run)
```

imageItem *Display an image specified by its filename.*

Description

Display an image specified by its filename.

Usage

```
imageItem(value = "", name, label = name, help = "", tooltip = "",
  attr = list(), model, editor, ...)
```

Arguments

value	name of file
name	Required name for object. Names should be unique within a group of items
label	ignored
help	Optional help string
tooltip	Optional tooltip to display
attr	A list of attributes to pass to widget on construction. Eg. attr=list(size=c(100,200)).
model	ignored
editor	ignored
...	Passed to parent proto object during call to proto

Value

A proto object. Call obj$show_help() to view its methods and properties.

See Also

Item

Examples

```
img <- system.file("images/plot.gif", package="gWidgets")   ## some image
i <- imageItem(img)                                         ## constructor
## Not run: i$make_ui(container=gwindow("Image"))                ## show item directly
```

integerItem *Item for integers*

Description

Item for integers

Usage

```
integerItem(value = integer(0), name, label = name, help = "",
  tooltip = "", eval_first = FALSE, attr, model, editor, ...)
```

Arguments

value	Default value for the model
name	Required name for object. Names should be unique within a group of items
label	Optional label, default value is the name
help	Optional help string
tooltip	Optional tooltip to display
eval_first	Should value be run through eval/parse before coercion.
attr	A list of attributes to pass to widget on construction. Eg. attr=list(size=c(100,200))
model	Optional model. Useful if one wishes to use same model for multiple views
editor	Specification of editor (a view) to override default
...	Passed to parent proto object during call to proto

Value

A proto object. Call obj$show_help() to view its methods and properties.

See Also

numericItem

Item *Base Trait for an Item*

Description

An Item combines a model, view and controller interface into one convenient package. Items may be combined into an ItemGroup or a Dialog to be shown.

Usage

```
Item
```

Format

```
proto object
 $ set_model              :function (., model)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 255 36 283 26 36 26 255 283
  .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265c14c40>
 $ attr                   : list()
 $ tooltip                : chr ""
 $ init_ui                :function (., container, attr = .$attr, context, ...)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 496 33 504 25 33 25 496 504
  .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265c14c40>
 $ new                    :function (., ...)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 514 30 518 26 30 26 514 518
  .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265c14c40>
 $ set_validate           :function (., f)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 208 40 208 68 40 68 208 208
  .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265c14c40>
 $ show_label             : logi TRUE
 $ to_string              :function (., drop = TRUE)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 462 38 466 26 38 26 462 466
  .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265c14c40>
 $ instance               :function (.)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 526 35 532 26 35 26 526 532
  .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265c14c40>
 $ visible_when           :function (.)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 183 42 183 59 42 59 183 183
  .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265c14c40>
 $ make_ui                :function (., container, attr = .$attr, context, ...)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 164 36 168 26 36 26 164 168
  .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265c14c40>
 $ make_default_gui_layout:function (.)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 174 51 174 63 51 63 174 174
  .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265c14c40>
 $ get_model              :function (.)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 295 36 295 54 36 54 295 295
```

```
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265c14c40>
$ handler_user_data      : NULL
$ setattr                :function (., key, value, notify_private = TRUE)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 329 34 360 26 34 26 329 360
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265c14c40>
$ set_model_from_item    :function (., item)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 285 46 291 26 46 26 285 291
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265c14c40>
$ parent                 : NULL
$ validate               :function (., rawvalue)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 225 35 233 26 35 26 225 233
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265c14c40>
$ update_ui              :function (.)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 537 36 542 26 36 26 537 542
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265c14c40>
$ visible                :function (., value)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 188 37 190 27 37 27 188 190
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265c14c40>
$ label                  : chr "label"
$ init                   :function (.)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 508 31 510 26 31 26 508 510
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265c14c40>
$ to_R                   :function (., drop = TRUE)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 445 33 454 26 33 26 445 454
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265c14c40>
$ init_model             :function (.)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 382 37 434 26 37 26 382 434
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265c14c40>
$ init_controller        :function (.)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 477 41 488 26 41 26 477 488
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265c14c40>
$ class                  : chr [1:3] "Item" "Model" "TraitR"
$ on_realized            : NULL
$ remove_observer        :function (., o)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 308 44 310 26 44 26 308 310
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265c14c40>
$ coerce_with            : NULL
$ set_editor             :function (., editor)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 143 39 150 26 39 26 143 150
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265c14c40>
$ enabled                :function (., value)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 202 37 204 27 37 27 202 204
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265c14c40>
$ default_get            :function (.)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 365 37 367 25 37 25 365 367
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265c14c40>
$ controller             : NULL
$ default_set            :function (.)
```

```
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 373 37 375 25 37 25 373 375
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265c14c40>
$ excluded_property_names: chr [1:2] "editor" "model"
$ model                  : NULL
$ is_valid               :function (.)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 240 34 243 25 34 25 240 243
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265c14c40>
$ icon                   : chr ""
$ name                   : chr ""
$ enabled_when           :function (.)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 197 42 197 59 42 59 197 197
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265c14c40>
$ add_observer           :function (., o)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 301 41 303 26 41 26 301 303
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265c14c40>
$ get_editor             :function (.)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 154 39 154 58 39 58 154 154
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265c14c40>
$ editor                 : NULL
$ properties             : NULL
$ getattr                :function (., key)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 316 34 321 26 34 26 316 321
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265c14c40>
$ add_handler_name       : chr "addHandlerChanged"
$ value                  : chr "default value"
$ help                   : chr ""
$ get_controller         :function (.)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 475 42 475 65 42 65 475 475
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265c14c40>
parent: proto object
```

Details

The `make_ui` method creates the user interface, initializes the model and the controller. The `init_model`, `init_controller` and `init_ui` do the work.

The model may be shared with different items. See `set_model_from_item` or the `instance` proto method.

Items implement the observer interface, so one can add observers to listen for changes to the properties. (Properties are listed in the property "`properties`".)

Items use the Adpater interface to link the model with the view (an Editor). The "`properties`" property lists the names of model properties. One should use "value" for the special one to be returned by the method `to_R`. (This method gathers values from the items after coercion)

When an item's user interface is made, the method `on_realized` is called.

ItemGroup *Base Trait to group items together to form a model. ItemGroups may be viewed as a model, view and controller bundled together in a tidy package.*

Description

An item group is a collection of Item instances. These are specified through the `items` property as a list.

Usage

```
ItemGroup
```

Format

```
proto object
 $ to_R                  :function (., drop = TRUE)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 101 33 106 26 33 26 101 106
  .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265eedae8>
 $ add_observer          :function (., o)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 308 41 314 26 41 26 308 314
  .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265eedae8>
 $ remove_observer       :function (., o)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 315 44 321 26 44 26 315 321
  .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265eedae8>
 $ undo_ptr              : num 0
 $ init_controller       :function (.)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 271 42 280 26 42 26 271 280
  .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265eedae8>
 $ class                 : chr [1:3] "ItemGroup" "Model" "TraitR"
 $ items                 : list()
 $ init                  :function (.)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 282 31 286 26 31 26 282 286
  .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265eedae8>
 $ make_default_gui_layout:function (.)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 165 50 169 26 50 26 165 169
  .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265eedae8>
 $ visible               :function (., value)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 175 34 177 26 34 26 175 177
  .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265eedae8>
 $ instance              :function (.)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 291 35 307 26 35 26 291 307
  .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265eedae8>
 $ to_string             :function (., drop = TRUE)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 112 38 117 26 38 26 112 117
  .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265eedae8>
```

```
$ make_gui            :function (., container, gui_layout, parent = NULL, visible = TRUE, ...)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 137 37 159 26 37 26 137 159
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265eedae8>
$ undo_trim_stack       :function (.)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 334 42 343 26 42 26 334 343
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265eedae8>
$ icon                  : chr ""
$ redo                  :function (.)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 365 31 372 26 31 26 365 372
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265eedae8>
$ undo                  :function (.)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 356 31 363 26 31 26 356 363
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265eedae8>
$ get_items             :function (.)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 54 38 59 26 38 26 54 59
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265eedae8>
$ undo_add              :function (., property, value, old_value, ...)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 344 35 354 26 35 26 344 354
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265eedae8>
$ is_valid              :function (.)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 122 36 124 27 36 27 122 124
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265eedae8>
$ gui_layout            : NULL
$ get_item_by_name      :function (., name)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 86 45 88 26 45 26 86 88
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265eedae8>
$ get_items_only        :function (.)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 65 43 77 26 43 26 65 77
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265eedae8>
$ init_model            :function (.)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 204 37 248 26 37 26 204 248
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265eedae8>
$ get_item              :function (., key)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 82 37 82 67 37 67 82 82
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265eedae8>
$ undo_can_redo         :function (.)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 364 40 364 84 40 84 364 364
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265eedae8>
$ enabled               :function (., value)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 183 34 185 26 34 26 183 185
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265eedae8>
$ undo_stack_length     : num 25
$ update_ui             :function (.)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 379 36 385 26 36 26 379 385
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265eedae8>
$ make_ui             :function (., container, attr = .$attr, context = ., ...)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 187 34 189 26 34 26 187 189
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265eedae8>
```

```
$ undo_stack            : list()
$ set_model             :function (., itemgroup)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 255 37 264 26 37 26 255 264
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265eedae8>
$ undo_can_undo         :function (.)
 ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 355 40 355 65 40 65 355 355
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265eedae8>
$ undo_do               : logi TRUE
parent: proto object
.. parent: proto object
```

Details

ItemGroups implement the observer pattern, so are models and can have observers listen for changes ItemGroups observe themselves to update the user interface on model changes. One can add other observers if desired. See the `init` method for an example.

If a method `model_value_changed` is defined, then it will be called with the ItemGroup instance being in whenever a property of the model has a new value. The handlers `property_NAME_value_changed` is called when the main value in item NAME is changed. (An item can have several properties, one of which is the main one.)

ItemGroups have a `make_gui` method to make a view of the model. The layout of this GUI can be specified through its `gui_layout` argument, or by default have a simple table layout. ItemGroup instances are meant to be embedded into a GUI, so the `cont` argument is needed to pass in the desired container.

`itemList` *An itemList is used to store a list of similar items or itemgroups with a means to edit individually*

Description

An itemList is used to store a list of similar items or itemgroups with a means to edit individually

Usage

```
itemList(items = list(), items_name = "", item_factory = NULL, name,
  label = name, help = "", tooltip = "", attr = list(), model, editor,
  ...)
```

Arguments

`items`	list of similar items, may be empty list
`items_name`	Header name on top of table displaying item list
`item_factory`	function to call to produce a new item, e.g. `function(.) numericItem(1)`
`name`	name of itemList object

label	label for itemList object
help	help string
tooltip	tooltip
attr	attributes passed to make_ui constructor
model	optional model to pass in
editor	optional editor to pass in
...	passed along to Item$proto() call

Value

A proto object. Call obj$show_help() to view its methods and properties.

Note

This item's model is a a list storing child items or item groups. To create new items, the item_factory method should be provided. It provides a template for a new item, the editor allows the user to modify its values When a child item is edited the "done" button is clicked to close. The method post_process is called. (The edited changes may already have been sent back to the model.) The child items to_string method is called to make the label in the table that allows the user to select the child item to edit. This should be a character vector of length 1. The table can display an icon. Simply set the icon property of the icon to a **gWidgets** stock icon name.

The child items can be returned via the get_value method or the get_NAME method, where NAME is that passed into the name argument of the constructor. The to_R method can be modified to manipulate the return value. The vignette has an example where the output is coerced into a data frame. The default is a list with each child items to_R method called to form the numbered components.

Examples

```
## Not run:
## make icons
imagedir <- system.file("images",package="traitr")
addStockIcons(gsub("\\\\.png","", list.files(path=imagedir)),
              list.files(path=imagedir, full.names=TRUE))
## make item
item <- itemList(items=list(),
                 items_name="Personnel",
                 item_factory = function(.) {
                   a <- anItemGroup(items=list(
                                    name=stringItem(""),
                                    rank=choiceItem("Scout",
                                         values=c("Scout","Captain","General")),
                                serial.number = stringItem("", label="Serial number")))
                   a$post_process <- function(.) {
                    .$icon <- tolower(.$get_rank())
                    }
                   a$to_string <- function(., drop=TRUE) .$to_R()$name
                   return(a)
                 },
                 name="itemlist")
```

```
item$make_ui(container=gwindow("itemList test"))

## End(Not run)
```

ItemListEditor *trait for editor for itemList*

Description

trait for editor for itemList

Usage

```
ItemListEditor
```

Format

```
proto object
 $ update_ui        :function (., context)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 350 44 370 32 44 32 350 370
  .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265e2c778>
 $ class            : chr [1:4] "ItemListEditor" "Editor" "View" "TraitR"
 $ get_table_entries:function (., items, name)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 219 50 232 32 50 32 219 232
  .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265e2c778>
 $ clear_edit_area  :function (., message = "")
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 235 50 241 32 50 32 235 241
  .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265e2c778>
 $ make_ui          :function (., container, attr = .attr, context = ., ...)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 269 42 348 32 42 32 269 348
  .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265e2c778>
 $ edit_item        :function (., context, item)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 243 44 267 32 44 32 243 267
  .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc265e2c778>
 parent: proto object
 .. parent: proto object
 .. .. parent: proto object
```

labelItem *Simple label item*

Description

Useful to adding text to a dialog. Has no interactivity.

Usage

```
labelItem(value = "label", name, label, help = "", tooltip = "", attr,
  model, editor, ...)
```

Arguments

value	Default value for the label
name	Required name for object. Names should be unique within a group of items
label	Same as value. Here for consistency, but needn't be used
help	Optional help string
tooltip	Optional tooltip to display
attr	A list of attributes to pass to widget on construction. Eg. attr=list(size=c(100,200))
model	Optional model. Useful if one wishes to use same model for multiple views
editor	Specification of editor (a view) to override default
...	Passed to parent proto object during call to proto

Value

A proto object. Call obj$show_help() for its methods and properties

Model *Trait for a model object.*

Description

Model objects consist of properties and the methods that manipulate them Models are initialized by init so that getter and setter pairs are made When setter functions are called, the model's observers are notified

Usage

```
Model
```

Format

```
proto object
 $ remove_observer :function (., observer)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 86 45 92 27 45 27 86 92
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266bf3298>
 $ setattr         :function (., key, value, notify_private = TRUE)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 116 37 122 27 37 27 116 122
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266bf3298>
 $ class           : chr [1:2] "Model" "TraitR"
 $ init            :function (.)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 128 34 139 27 34 27 128 139
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266bf3298>
 $ list_observers  :function (.)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 97 41 97 64 41 64 97 97
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266bf3298>
 $ getattr         :function (., key)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 107 37 107 70 37 70 107 107
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266bf3298>
 $ add_observer    :function (., observer)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 71 42 80 27 42 27 71 80
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266bf3298>
 $ notify_observers:function (., key = NULL, value = NA, old_value = NA, notify_private = TRUE)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 47 46 64 27 46 27 47 64
 .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266bf3298>
 parent: proto object
```

`numericItem` *Item for numbers*

Description

Item for numbers

Usage

```
numericItem(value = numeric(0), name, label = name, help = "",
  tooltip = "", eval_first = FALSE, attr, model, editor, ...)
```

Arguments

`value`	Default value for the model
`name`	Required name for object. Names should be unique within a group of items
`label`	Optional label, default value is the name
`help`	Optional help string
`tooltip`	Optional tooltip to display

eval_first	Should value be run through eval/parse before coercion.
attr	A list of attributes to pass to widget on construction. Eg. attr=list(size=c(100,200))
model	Optional model. Useful if one wishes to use same model for multiple views
editor	Specification of editor (a view) to override default
...	Passed to parent proto object during call to proto

Value

A proto object. Call obj$show_help() to view its methods and properties.

See Also

Item

Examples

```
## basic use
a <- numericItem(0, name="x")
a$set_x(10)
a$get_x()
## eval can be instructed
a <- numericItem(0, name="x", eval_first=TRUE)
a$set_x("1:5")
a$get_x()
a$to_R()
```

---

rangeItem *A range selection item*

Description

Editor is a slider (with spinbutton when by value is an integer).

Usage

```
rangeItem(value = "", from = 0, to = 10, by = 1, name, label = name,
  help = "", tooltip = "", attr, model, editor, ...)
```

Arguments

value	Default data frame for the model
from	Starting value of range
to	Ending value of range
by	Step size to step through range. If an integer, a spinbutton is also displaye
name	Required name for object. Names should be unique within a group of items
label	Optional label, default value is the name

help	Optional help string
tooltip	Optional tooltip to display
attr	A list of attributes to pass to widget on construction. Eg. attr=list(size=c(100,200))
model	Optional model. Useful if one wishes to use same model for multiple views
editor	Specification of editor (a view) to override default
...	Passed to Item trait

Value

A proto object. Call obj$show_help() to view its methods and properties.

Examples

```
i <- rangeItem(value=5, from=0, to=10, by=1, name="rng")
i$get_rng()
i$set_rng(10)
i$get_rng()
```

---

separatorItem *Visual separator item*

Description

Creates a horizontal (or vertical if done through "attr") line to separate GUI elements.

Usage

```
separatorItem(name = ".seperator", attr = list(horizontal = TRUE), ...)
```

Arguments

name	name of widget
attr	passed to widget constructor. Use list(horizontal=FALSE) to get vertical
...	ignored

Value

A proto object. Call obj$show_help() to view its methods and properties.

stringItem *A string item*

Description

A string item

Usage

```
stringItem(value = "", regex = NULL, name, label = name, help = "",
  tooltip = "", eval_first = FALSE, attr, model, editor, ...)
```

Arguments

value	Default value for the model
regex	If non NULL specifies a regular expression for validation.
name	Required name for object. Names should be unique within a group of items
label	Optional label, default value is the name
help	Optional help string
tooltip	Optional tooltip to display
eval_first	Should value be run through eval/parse before coercion.
attr	A list of attributes to pass to widget on construction. Eg. attr=list(size=c(100,200))
model	Optional model. Useful if one wishes to use same model for multiple views
editor	Specification of editor (a view) to override default
...	Passed to parent proto object during call to proto

Value

A proto object. Call obj$show_help() to view its methods and properties.

See Also

Item

Examples

```
## basic usage
a <- stringItem("ac", name="x")
a$get_x()
a$set_x("abc213")
a$get_x()
## eval first
a <- stringItem("ac", name="x", eval_first=TRUE)
a$set_x("2 + 2")
a$get_x()
a$to_R()
```

tableItem — *List editor – list <-> tree, must have special structure to list? XXX This needs writing An item to display a table of data (given as a matrix or data.frame)*

Description

List editor – list <-> tree, must have special structure to list? XXX This needs writing An item to display a table of data (given as a matrix or data.frame)

Usage

```
tableItem(value = data.frame(V1 = "", V2 = ""), name, label = name,
  help = "", tooltip = "", attr = list(expand = TRUE), model, editor, ...)
```

Arguments

value	Default value of data frame
name	Required name for object. Names should be unique within a group of items
label	ignored
help	Optional help string
tooltip	Optional tooltip to display
attr	A list of attributes to pass to widget on construction. Eg. attr=list(size=c(100,200)).
model	ignored
editor	ignored
...	Passed to parent proto object during call to proto

Value

A proto object. Call obj$show_help() to view its methods and properties.

See Also

Item

Examples

```
## to change data frame
i <- tableItem(mtcars, name="a")
i$set_a(mtcars[1:3, 1:3])
```

`trueFalseItem` *Item for Boolean values*

Description

Item for Boolean values

Usage

```
trueFalseItem(value = TRUE, name, label = name, help = "", tooltip = "",
  attr, model, editor, ...)
```

Arguments

`value`	Default value for the model
`name`	Required name for object. Names should be unique within a group of items
`label`	Optional label, default value is the name
`help`	Optional help string
`tooltip`	Optional tooltip to display
`attr`	A list of attributes to pass to widget on construction. Eg. attr=list(size=c(100,200))
`model`	Optional model. Useful if one wishes to use same model for multiple views
`editor`	Specification of editor (a view) to override default
`...`	Passed to parent proto object during call to proto. The value `editor_style="compact"` will pass this information to the editor causing it to render as a checkbox.

Value

A `proto` object. Call `obj$show_help()` to view its methods and properties.

See Also

`Item`

Examples

```
## basic usage
a <- trueFalseItem(TRUE, name="x")
a$get_x()
a$set_x(FALSE)
a$get_x()
```

`variableSelectorItem` *Item to select a variable (or variables) from a selected data frame*

Description

Needs to have a dataframeItem specified to be useful.

Usage

```
variableSelectorItem(value = NA, multiple = FALSE, dataframeItem, name,
  label = name, help = "", tooltip = "", attr, model, editor, ...)
```

Arguments

`value`	Default data frame for the model, defaults to .GlobalEnv
`multiple`	Allow multiple selection?
`dataframeItem`	A required dataframeItem instance. This need not be in same display, or even displayed
`name`	Required name for object. Names should be unique within a group of items
`label`	Optional label, default value is the name
`help`	Optional help string
`tooltip`	Optional tooltip to display
`attr`	A list of attributes to pass to widget on construction. Eg. attr=list(size=c(100,200)) This widget uses a gtable instance and specifying the size is suggested
`model`	Optional model. Useful if one wishes to use same model for multiple views
`editor`	Specification of editor (a view) to override default
`...`	Passed to parent proto object during call to proto

Value

A `proto` object. Call `obj$show_help()` to view its methods and properties.

See Also

`dataframeItem`, `Item`

Examples

```
df <- data.frame(a=1:3, b= letters[1:3], c=rnorm(3)) # make a data frame
dfI <- dataframeItem(value="df", name="dfI")
dlg <- aDialog(items=list(
       dfI,                      ## a bit awkward -- can't define dfI in list of items
variable=variableSelectorItem(dataframeItem=dfI))
)
## Not run:  dlg$make_gui()
```

View *Trait for View objects.*

Description

A view "displays" the values in an associated model. The association is through a controller the view does not know the controller or the model Views are initialized through the `make_ui` method.

Usage

```
View
```

Format

```
proto object
 $ class              : chr [1:2] "View" "TraitR"
 $ is_realized        :function (.)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 62 38 64 26 38 26 62 64
  .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266c784a0>
 $ set_value_in_view  :function (., widget_name, value)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 79 45 87 25 45 25 79 87
  .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266c784a0>
 $ widgets            : list()
 $ attr               : list()
 $ get_value_from_view:function (.)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 73 47 73 60 47 60 73 73
  .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266c784a0>
 $ enabled            :function (., value)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 94 36 96 26 36 26 94 96
  .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266c784a0>
 $ get_widgets        :function (.)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 43 39 43 59 39 59 43 43
  .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266c784a0>
 $ visible            :function (., value)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 101 36 103 26 36 26 101 103
  .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266c784a0>
 $ make_ui            :function (., cont, attr = .$attr)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 57 34 57 66 34 66 57 57
  .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266c784a0>
 $ get_widget_by_name :function (., key)
  ..- attr(*, "srcref")=Class 'srcref'  atomic [1:8] 48 45 48 83 45 83 48 48
  .. .. ..- attr(*, "srcfile")=Classes 'srcfilecopy', 'srcfile' <environment: 0x7fc266c784a0>
 parent: proto object
```

wrap_in_tag *write values in paired tags with optional class*

Description

internal function for writing proto docs

internal function for writing proto docs. return a string

internal function for writing proto docs.

internal function for writing proto docs. return a string

internal function for writing proto docs.

merge two lists, possibly overwriting

A window to show a loading animation

Usage

```
wrap_in_tag(tag, ..., class = "")

desc(...)

param(value, ...)

returns(...)

ul(values)

## S3 method for class 'list'
merge(x, y, overwrite = TRUE)

loadingAnimation(message = "<b>Loading...</b>")
```

Arguments

tag	tag to wrap in, eg. "ul"
...	values to be wrapped. Pasted together
class	optional call to add to tag
value	name of parameter return a string
values	values to put into list items
x	a list
y	a list. Named values of y are assigned to x and then x is returned.
overwrite	If TRUE named values in y clobber similarly named values in x
message	A message to display along with graphic while loading. PANGO markup is okay.

Value

a string

returns a string

Returns a list

An item group instance with a `close` method to call to dismiss window

Examples

```
## we call, something happens, then we close
## Not run:
w <- loadingAnimation()
## .... something long, like dlg$make_gui() ...
w$close()

## End(Not run)
```

Index

∗Topic **datasets**
Adapter, 5
BaseTrait, 17
BooleanEditor, 18
Container, 22
Controller, 23
Dialog, 26
Editor, 29
Item, 37
ItemGroup, 40
ItemListEditor, 44
Model, 45
View, 53

aContainer, 3, *12*, *16*
aContext, 4
aController, 5
Adapter, 5
aDialog, *3*, 6
aFrame, 8
aGroup, 9
aModel, 10
anExpandGroup, 11
anItemGroup, *6*, 12
aNotebook, 13
aNotebookPage, 14
aPanedGroup, 15
aTableLayout, 16
aView, 17

BaseTrait, 17
BooleanEditor, 18
ButtonEditor *(*Editor*)*, 29
buttonItem, 19

choiceItem, 20
Container, *4*, *5*, *9–11*, *14–16*, 22
Controller, 23

dataframeItem, 24, *52*
DateEditor *(*Editor*)*, 29
dateItem, 25
desc *(*wrap_in_tag*)*, 54
dfEditItem, 26
Dialog, 26
dialogMaker, 28

Editor, 29
EntryEditor *(*Editor*)*, 29
expressionItem, 31

FileBrowseEditor *(*Editor*)*, 29
fileItem, 32
formulaItem, 32

get_with_default, 33
gmenu, *7*
GraphEditor *(*Editor*)*, 29
graphicDeviceItem, 34
gtoolbar, *7*

ImageEditor *(*Editor*)*, 29
imageItem, 35
integerItem, *31*, 36
Item, *20*, *21*, *26*, *34*, *35*, 37, *47*, *49–52*
ItemGroup, 40
itemList, 42
ItemListEditor, 44

LabelEditor *(*Editor*)*, 29
labelItem, 45
loadingAnimation *(*wrap_in_tag*)*, 54

merge.list *(*wrap_in_tag*)*, 54
Model, 45

numericItem, *31*, *36*, 46

ObjectWithValuesEditor *(*Editor*)*, 29

param *(*wrap_in_tag*)*, 54

RangeEditor (Editor), 29
rangeItem, 47
returns (wrap_in_tag), 54

SeparatorEditor (Editor), 29
separatorItem, 48
strftime, *25*
stringItem, *31*, 49

TableEditor (Editor), 29
tableItem, 50
traitR-package, 3
trueFalseItem, 51

ul (wrap_in_tag), 54

variableSelectorItem, 52
View, 53

wrap_in_tag, 54