Package ‘svGUI’

February 15, 2013

Type Package

Version 0.9-54

Date 2013-01-27

Title SciViews GUI API - Functions to manage GUIs

Author Philippe Grosjean

Maintainer Philippe Grosjean <phgrosjean@sciviews.org>

Depends R (>= 2.6.0)

Description Functions to manage GUIs from R

License GPL-2

URL http://www.sciviews.org/SciViews-R

BugReports https://r-forge.r-project.org/tracker/?group_id=194

Repository CRAN

Date/Publication 2013-01-28 07:37:41

NeedsCompilation no

R topics documented:

svGUI-package ...................................................... 2
dontAsk ............................................................... 2
gui ................................................................. 3
guiAdd ............................................................... 4
setUI ............................................................... 6

Index 8
Description

The SciViews svGUI package provides function to implement GUI (Graphical User Interface) in R. It is independent from any particular GUI toolkit, centralize info about GUI elements currently used, and dispatch GUI functions to the particular toolkits used.

Details

Package: svGUI
Type: Package
Version: 0.9-54
Date: 2013-01-27
License: GPL 2 or above, at your convenience

Author(s)

Philippe Grosjean
Maintainer: Ph. Grosjean <phgrosjean@sciviews.org>

Description
dontAsk

Don’t we ask something though the GUI?

Usage
dontAsk(gui = .GUI)

Arguments
gui a ’gui’ object.

Details

Methods for 'gui' objects can dispatch as usual using amethod(..., gui = agui) but note that these methods do not dispatch on first provided argument, but to the named argument gui. There is another way to call 'gui' methods: agui$amethod(...). This may be a convenient alternative for those who prefer this style of calling object’s methods.
Value

TRUE if the GUI cannot interrupt. The function triggering the dialog box should then not display it and should return the default value as the result. The function returns TRUE if R is run in a non interactive session, or if ask is set to FALSE for the GUI, or if it is not specified, that is, its value is NULL, if getOptions("gui.ask") is FALSE.

Author(s)

Philippe Grosjean <phgrosjean@sciviews.org>

See Also

guiAsk, gui

Examples

## What's the current state for the default GUI
dontAsk()

---

**gui**

A GUI object

Description

The 'gui' object contains and manages GUI-related data.

Usage

## S3 method for class 'gui'
gui$x, args
## S3 method for class 'gui'
print(x, ...)
is.gui(x)

Arguments

x a function for $, or an object for the others.
gui a 'gui' object.
args arguments to pass to the function.
... further arguments (not used yet).

Details

Methods for 'gui' objects can dispatch as usual using amethod(...., gui = agui) but note that these methods do not dispatch on first provided argument, but to the named argument gui. There is another way to call 'gui' methods: agui$amethod(....). This may be a convenient alternative for those who prefer this style of calling object's methods.
Value

The result of the evaluation of the method of function applied for `agui$mamethod(...).` The
'gui' object, invisibly, for `print()`. `TRUE` for `is.gui(x)` if the object `x` inherits from 'gui', `FALSE`
otherwise.

Note

The 'gui' objects are not terribly useful by themselve, but thay provide the fundation for a flexible
organisation of GUI elements in R (see for instance svDialogs or svWidgets).

Author(s)

Philippe Grosjean <phgrosjean@sciviews.org>

See Also

guiAdd

Examples

```r
## Create a GUI
guiAdd("myGUI")
is.gui(myGUI)
myGUI
guiRemove("myGUI")
```
guiAdd

Arguments

**gui.name**  
the name of the GUI. It is also the name of the object stored in SciViews:TempEnv where you can access it. Although not required, you are better to use short, syntactically correct names.

**widgets**  
the list of widgets that GUI uses, listed in a priority order.

**ask**  
logical indicating if simple dialog boxes should be display (ask = TRUE), or if those dialog boxes are by-passed, using default values to simulate script running in non interactive mode, or to test scripts without interruption, using only provided default values (useful for automated tests).

**reset**  
should the GUI's main parameters (widgets, ask) be reset to default values?

**gui**  
a 'gui' object. If provided, it supersedes any value provided in gui.name.

**x**  
a 'gui' object.

**value**  
the list of widgets to add to this GUI, in priority order, or should we change ask to TRUE, FALSE or NULL (then, use default value stored in getOption("gui.ask")).

**gui.or.name**  
a 'gui' object or its name.

Value

The 'gui' object just created or changed for guiAdd(), guiChange(), guiWidgets(agui) <- value or guiAsk(agui) <- value. For guiRemove(), TRUE if an object was removed, or FALSE if the object was not there, invisibly. For guiList(), the list of current 'gui' objects in SciViews:TempEnv, or a zero-length character if none (should not happen, since .GUI, the default GUI, cannot be removed).

Author(s)

Philippe Grosjean (<phgrosjean@sciviews.org>)

See Also

gui, setUI, dontAsk

Examples

```r
## A 'gui' object named .GUI is automatically created in SciViews:TempEnv
guilist()

## Create a new GUI object to manage a separate GUI in the same R session
guiAdd("myGUI")
guilist()

## Change general properties of this GUI
guiAsk(myGUI) <- FALSE
## Add widgets to this GUI (you must provide methods for them)
## see the svDialogstcltk package for examples how to do this
guiWidgets(myGUI) <- "tcltkWidgets"
guiWidgets(myGUI) # Added to existing ones if reset is FALSE
```
## Remove this new GUI

`guiRemove("myGUI")`

---

### setUI

Set a property in the UI (User Interface), or start an action

---

**Description**

This is the preferred way to set a property in a `gui` object or to indicate that an UI action is about to start.

**Usage**

```r
setUI(..., gui = .GUI)
```

## S3 method for class 'gui'

```
setUI(fun, call, args, res, widgets, status, msg = NULL, 
     ..., gui = .GUI)
```

```r
startUI(..., gui = .GUI)
```

## S3 method for class 'gui'

```
startUI(fun, call, default, widgets = NULL, status = "busy-modal", 
        msg = "Displaying a modal dialog box", 
        msg.no.ask = "A modal dialog box was by-passed", ..., gui = .GUI)
```

**Arguments**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fun</td>
<td>the name of the calling function. Only required if call is provided.</td>
</tr>
<tr>
<td>call</td>
<td>the call in the generic as obtained by <code>match.call()</code>.</td>
</tr>
<tr>
<td>args</td>
<td>a list with checked and/or reworked arguments for a method. The generic can</td>
</tr>
<tr>
<td></td>
<td>do this work, so that code does not need to be duplicated in all its methods.</td>
</tr>
<tr>
<td>res</td>
<td>any data returned by the GUI (the results).</td>
</tr>
<tr>
<td>default</td>
<td>the default value to return if the UI is by-passed because in non interactive mode, or ask is FALSE.</td>
</tr>
<tr>
<td>widgets</td>
<td>the class name of the current widgets implementation.</td>
</tr>
<tr>
<td>status</td>
<td>description of the current GUI status. Could be &quot;ok&quot;, &quot;busy&quot;, &quot;busy-modal&quot; (a modal dialog box is currently displayed), &quot;by-passed&quot; (the GUI was by-passed because <code>dontAsk()</code> returns TRUE), &quot;error&quot;, or any other status indicator suitable for the current state of your GUI.</td>
</tr>
<tr>
<td>msg</td>
<td>the message expliciting the status. Cannot be provided without status.</td>
</tr>
<tr>
<td>msg.no.ask</td>
<td>the message expliciting the status in case the UI is by-passed.</td>
</tr>
<tr>
<td>...</td>
<td>any other property of the GUI, provided as named arguments.</td>
</tr>
<tr>
<td>gui</td>
<td>a <code>gui</code> object.</td>
</tr>
</tbody>
</table>

**Value**

The modified `gui` object is returned invisibly by `setUI()`. For `startUI()` either TRUE (can start the UI), or FALSE if the UI should be by-passed.
setUI

Author(s)
Philippe Grosjean <phgrosjean@sciviews.org>

See Also

guiAdd, $.gui

Examples

## Imagine you implement a new input box
## In your function, you have this code:
myInput <- function (default = "an answer", gui = .GUI) {
## Start a GUI action... or by-pass it!
if (gui$startUI("myInput", call = match.call(), default = default,
   msg = "Displaying an input dialog box",
   msg.no.ask = "An input dialog box was by-passed")) {
   ## Here the input dialog box is displayed and R waits from user's action
   ## ... [your code here]
   res <- "some results" # Imagine this is the text typed in the box

   ## When the dialog box is closed, the function should do:
   setUI(res = res, status = NULL)
}
return(invisible(gui))
}
## Index

*Topic **misc**
  - dontAsk, 2
  - gui, 3
  - guiAdd, 4
  - setUI, 6

*Topic **package**
  - svGUI-package, 2

*Topic **utilities**
  - svGUI-package, 2

$.gui, 7
$.gui (gui), 3
dontAsk, 2, 5
gui, 3, 3, 5
guiAdd, 4, 4, 7
guiAsk, 3
guiAsk (guiAdd), 4
guiAsk<- (guiAdd), 4
guiChange (guiAdd), 4
guiList (guiAdd), 4
guiRemove (guiAdd), 4
guiWidgets (guiAdd), 4
guiWidgets<- (guiAdd), 4

is.gui (gui), 3
print.gui (gui), 3

setUI, 5, 6
startUI (setUI), 6
svGUI (svGUI-package), 2
svGUI-package, 2