The SciViews Package

April 24, 2006

Contains  svMisc svIO svViews svDialogs svSocket svGUI svIDE svWidgets

Version  0.9-5

Date  2006-04-24

Author  Philippe Grosjean & Eric Lecoutre

Maintainer  Philippe Grosjean <phgrosjean@sciviews.org>

License  GPL 2 or above

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

#### Package ‘svMisc’
- Require ........................................... 1
- TempEnv .......................................... 2
- assignTemp ....................................... 3
- compareRVersion ................................. 3
- existsTemp ....................................... 4
- findhtmlhelp .................................... 5
- getEnvironment ................................ 5
- getTemp .......................................... 6
- guiCmd .......................................... 7
- guiSource ....................................... 7
- help.search.web ................................. 8
- isRgui ........................................... 9
- isSDI ........................................... 10
- isWin .......................................... 10
- listCustoms ................................... 11
- listMethods ................................... 12
- progress ....................................... 12
- rmTemp ......................................... 13
- tempdirWin ................................... 14
- tempvar ....................................... 15
- userdir ....................................... 15

#### Package ‘svIO’
- clippaste ...................................... 17
- copy ........................................... 18
- export ......................................... 19

#### Package ‘svViews’
- guiViews ....................................... 21
- report ......................................... 22
- view .......................................... 23
- viewHTMLinit ................................ 24
- wordReport ................................... 25

#### Package ‘svDialogs’
- display ......................................... 29
- guiDlg ......................................... 30
- guiDlgAssistant .............................. 31
- guiDlgColor ................................. 31
- guiDlgDir .................................... 32
CONTENTS

Package 'svSocket'
  getSocketClients .............................................. 49
  getSocketServerName ........................................ 50
  getSocketServers ........................................... 50
  processSocket ................................................ 51
  sendSocket .................................................. 52
  startSocketServer ........................................... 52

Package 'svGUI'
  Windows ......................................................... 55
  guiInfo ........................................................ 56
  guiObjBrowse ................................................ 57
  guiRemove ..................................................... 58
  guiSave ......................................................... 59
  guiStart ......................................................... 59
  svGUI-internal ................................................ 60

Package 'svIDE'
  Args ............................................................ 61
  Complete ......................................................... 62
  TinnR ........................................................... 63
  createSyntaxFile .............................................. 64
  getFunctions .................................................. 65
  getKeywords ................................................... 66
  guiDDEInstall ................................................ 67

Package 'svWidgets'
  Img ............................................................. 69
  Menu ............................................................ 70
  Tool ............................................................. 73
  Win ............................................................... 74
  tkMenu .......................................................... 75
## CONTENTS

<table>
<thead>
<tr>
<th>winMenu</th>
<th>76</th>
</tr>
</thead>
<tbody>
<tr>
<td>Index</td>
<td>79</td>
</tr>
</tbody>
</table>
Package ‘svMisc’

**Title** SciViews GUI API - Miscellaneous functions

**Imports** utils, methods

**Description** Supporting functions for the GUI API (manage local variables, GUI to install packages, etc...)

---

**Usage**

```
Require(package, bundle = NULL, quietly = FALSE, warn.conflicts = TRUE,
        keep.source = getOption("keep.source.pkgs"),
        character.only = FALSE, version, save = TRUE, gui=getOption("guiRequire"))
```

**Arguments**

- **package**
  
  the name of a package, given as a name or literal character string, or a character string, depending on whether character.only is FALSE (default) or TRUE.

- **bundle**
  
  a character string specifying the name of the bundle where the required package can be found. This is useful to specify, otherwise, Require() will look for the package name in the repository, while only the bundle name can be found there!

- **quietly**
  
  a logical. If TRUE, no message confirming package loading is printed.

- **warn.conflicts**
  
  logical. If TRUE, warnings are printed about code conflicts from attaching the new package, unless that package contains an object .conflicts.OK.

- **keep.source**
  
  logical. If TRUE, functions “keep their source” including comments, see argument keep.source to options.

- **character.only**
  
  a logical indicating whether package or help can be assumed to be character strings.

- **version**
  
  A character string denoting a version number of the package to be loaded. If no version is given, a suitable default is chosen.
TempEnv

save logical or environment. If TRUE, a call to require from the source for a package will save the name of the required package in the variable ".required", allowing function detach to warn if a required package is detached. See section ‘Packages that require other packages’ below.

gui logical. If FALSE, this function behaves just like the require function. If TRUE, then if a package fails to load, the user will be prompted with a message box and dialog asking whether they would like to install the package. If the user chooses to install the package, they will be asked which repository or local file they would like to install it from.

Author(s)

James Wettenhall (⟨wettenhall@wehi.edu.au⟩), adaptation by Philippe Grosjean (⟨phgrosjean@sciviews.org⟩)

Examples

## Not run:
options(guiRequire = TRUE)
# For packages which are already installed, Require works in the same way as require.
# Require(base)

# For packages which are not installed, Require asks the user whether they would like to install the package.
# Require(foo)
## End(Not run)

TempEnv Get (and create it if needed) a temporary environment

Description

This function creates and returns the reference to a temporary environment called TempEnv.

Usage

TempEnv()

Value

TempEnv (an environment object)

Author(s)

Philippe Grosjean <phgrosjean@sciviews.org>

See Also

assignTemp, getTemp, rmTemp, existsTemp

Examples

TempEnv()
assignTemp

Assign a temporary variable in the TempEnv environment

Description

The function assigns a variable to TempEnv, an environment dedicated to temporary variables.

Usage

assignTemp(x, value, replace.existing = TRUE)

Arguments

- **x** The name of the variable
- **value** The value of the variable
- **replace.existing** Do we replace an existing variable?

Author(s)

Philippe Grosjean <phgrosjean@sciviews.org>

See Also

TempEnv, getTemp, rmTemp, existsTemp

Examples

assignTemp("test", TRUE)
# Retrieve this variable
getTemp("test")

compareRVersion

Compare current R version with a specified one

Description

Determine if R is older (return -1), or not (return 1) than a given version number.

Usage

compareRVersion(version)

Arguments

- **version** A string defining the version, like ‘2.0’ or ‘1.9.1’

Value

-1 if R is older, 1 otherwise.
existsTemp

Determine if a variable exists in TempEnv

Description

Does a variable exists in the temporary environment, TempEnv?

Usage

```r
existsTemp(x, mode = "any")
```

Arguments

- `x`: The name of the variable (string)
- `mode`: The mode of the sought variable

Value

TRUE if the variable exists in TempEnv (and is of the correct mode), FALSE otherwise.

Author(s)

Philippe Grosjean <phgrosjean@sciviews.org>

See Also

`TempEnv, assignTemp, getTemp, rmTemp`

Examples

```r
assignTemp("test", TRUE)
# Retrieve this variable
existsTemp("test")
# Remove it
rmTemp("test")
# Does it exists?
existsTemp("test")
```
findhtmlhelp

Find the HTML help file associated with a topic

Description
Get the path of the HTML help file associated with a topic, but do not display it. It can be used for contextual menu of objects in an object browser.

Usage
findhtmlhelp(topic, package = .packages(), lib.loc = NULL, verbose = getOption("verbose"))

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>topic</td>
<td>The topic to search</td>
</tr>
<tr>
<td>package</td>
<td>An optional package where to look at</td>
</tr>
<tr>
<td>lib.loc</td>
<td>The path of the local library</td>
</tr>
<tr>
<td>verbose</td>
<td>Verbose output</td>
</tr>
</tbody>
</table>

Value
A character string containing the path of the HTML file, or "" if this file does not exist.

Author(s)
Eric Lecoutre <lecoutre@stat.ucl.ac.be>

See Also
listMethods

Examples

findhtmlhelp("plot")
findhtmlhelp("ls")

getEnvironment
Get an environment, knowing its position

Description
Given a position in the search path, the corresponding environment is returned.

Usage
getEnvironment(pos)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pos</td>
<td>The position in the search path</td>
</tr>
</tbody>
</table>
**getTemp**

**Value**

An environment.

**Author(s)**

Philippe Grosjean <phgrosjean@sciviews.org>

**Examples**

```r
getEnvironment(1)
getEnvironment(length(search())) # "package:base" => NULL
```

---

**getTemp**  
*Get a temporary variable from the TempEnv environment*

**Description**

The function gets a variable from TempEnv, an environment dedicated to temporary variables.

**Usage**

```r
getTemp(x, default = NULL, mode = "any")
```

**Arguments**

- `x`  
The name of the variable
- `default`  
The default value to return, in case the variable does not exist
- `mode`  
The mode of the variable (if the variable exists, but is not of correct mode, nothing is retrieved). Use `mode = "any"` (default value) to retrieve the variable whatever its mode.

**Value**

The content of the variable, or the default value if the variable is not found in TempEnv.

**Author(s)**

Philippe Grosjean <phgrosjean@sciviews.org>

**See Also**

`TempEnv`, `assignTemp`, `rmTemp`, `existsTemp`

**Examples**

```r
assignTemp("test", TRUE)
# Retrieve this variable
getTemp("test")
# Retrieve a non existing variable (returns default value)
getTemp("nonexistent", default = "default value")
```
guiCmd

Send a command to the GUI client

Description

This function is not intended to be used at the command line (except for debugging purposes). It sends a command string to a GUI client.

Usage

guiCmd(command)

Arguments

command The command string to send

Value

TRUE if the command succeed, the content of command, otherwise.

Author(s)

Philippe Grosjean (phgrosjean@sciviews.org)

See Also

guiInstall, guiStart

guiSource

Source R code, capture its output and convert it in a different format

Description

guiSource is like source, but it allows to rework the output (for instance to print it in HTML format)

Usage

guiSource(file, out.form = getOption("R.output.format"), local = FALSE, 
   echo = FALSE, print.eval = TRUE, verbose = getOption("verbose"), 
   prompt.echo = getOption("prompt"), max.deparse.length = 150, chdir = FALSE)
help.search.web

Arguments

file a connection or a character string giving the name of the file or URL to read from.
out.form a string indicating which output format to use (for instance, "html").
local if 'local' is 'FALSE', the statements scanned are evaluated in the user’s workspace (the global environment), otherwise in the environment calling 'source'.
echo logical; if 'TRUE', each expression is printed after parsing, before evaluation.
print.eval logical; if 'TRUE', the result of `eval(i)` is printed for each expression 'i'; defaults to 'echo'.
verbose if 'TRUE', more diagnostics (than just 'echo = TRUE') are printed during parsing and evaluation of input, including extra info for each expression.
prompt.echo character; gives the prompt to be used if 'echo = TRUE'.
max.deparse.length integer; is used only if 'echo' is 'TRUE' and gives the maximal length of the "echo" of a single expression.
chdir logical; if 'TRUE', the R working directory is changed to the directory containing 'file' for evaluating

Details

This function is usually called by functions that processes commands send by GUI clients.

Value

The formatted output is returned invisibly.

See Also

source, guiCmd

help.search.web  Search web documents about R and R functions

Description

Retrieve web documents, messages in R mailing lists, or wiki containing string.

Usage

help.search.web(apropos, type = c("google", "archive", "wiki"))

Arguments

apropos The string to search
type The search engine, or location to use

Value

Returns TRUE invisibly (invoked for its side effect of opening the web browser with the search result)
isRgui

Author(s)

David Forrest <drf@vims.edu> & Philippe Grosjean <phgrosjean@sciviews.org> after Barry Rowland’s original code, but see <http://fawn.unibw-hamburg.de/cgi-bin/Rwiki.pl?SearchFunctions>

See Also

help.search

Examples

```R
## Not run:
help.search.web("volatility") # Google by default, may soon show Dr. Harrell's example
help.search.web("volatility", type = "archive") # In mailing list archive
help.search.web("search", type = "wiki")
## End(Not run)
```

---

isRgui  
Is Rgui.exe (under Windows) running?

Description

Determine if the R UI is Rgui, the standard R program under Windows. This function can also be used under other platforms, but it will always return FALSE.

Usage

isRgui()

Value

TRUE if the R UI is Rgui.exe, FALSE otherwise.

Author(s)

Philippe Grosjean <phgrosjean@sciviews.org>

See Also

isSDI

Examples

isRgui()
isSDI  

Is Rgui started in SDI mode under Windows?

**Description**

Determine if R is running in Rgui under Windows and in SDI mode.

**Usage**

```r
isSDI()
```

**Value**

TRUE if it is Rgui under Windows, and it is stared in SDI mode (Single-Document Interface), FALSE otherwise.

**Note**

See the menu entry ‘Edit’ -> ‘GUI preferences’ to change it, or start Rgui with the ‘–SDI’ argument line parameter. Under another platform than Windows or if it is not Rgui, then isSDI always returns FALSE.

**Author(s)**

Philippe Grosjean <phgrosjean@sciviews.org> & Eric Lecoutre <lecoutre@stat.ucl.ac.be>

**See Also**

isWin

**Examples**

```r
isSDI()
```

---

isWin  

Is it a Windows platform?

**Description**

Determine if the platform is Windows.

**Usage**

```r
isWin()
```

**Value**

TRUE if the platform is Windows, FALSE otherwise.
**listCustoms**

*List all custom types for a method*

**Description**

List all custom types for a method.

**Usage**

```r
listCustoms(method, class)
```

**Arguments**

- `method` The method name
- `class` The class name

**Value**

A vector with character strings with custom methods names.

**Note**

This function is only useful for special generic functions with type argument like `view`, or `copy`. These functions offer a mechanism to easily add custom types, and the present function list them all.

**Author(s)**

Philippe Grosjean <phgrosjean@sciviews.org>

**See Also**

`listMethods`

**Examples**

```r
listCustoms("view", "data.frame") # None, except if you defined some of it!
```
listMethods  

List all methods associated with a generic function

Description
List all S3 and/or S4 methods for a generic function.

Usage
listMethods(f, S3 = TRUE, S4 = TRUE)

Arguments
- f: The name of the generic function
- S3: If TRUE, list of S3 methods
- S4: If TRUE, list of S4 methods

Value
A list with components:
- S3: The S3 methods for this generic function, or character(0) if none
- S4: The S4 methods for this generic function, or character(0) if none

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

See Also
listCustoms

Examples
listMethods("t.test") # S3
listMethods("show")   # S4
listMethods("ls")    # None, not a generic function!

progress  

Display progression of a long calculation on the console

Description
Display progression level of a long-running task in the console. Two mode can be used: either percent of achievement (55%), or the number of items or steps done on a total (1 file on 10 done...).

Usage
progress(value, max.value = NULL)
Arguments

value Describe value here
max.value Describe max.value here

Details

The function uses backspace (\8) to erase characters at the console.

Value

This function returns NULL invisibly. It is invoked for its side effects.

Note

In a GUI, it is preferable to use a non modal dialog box with a progress widget, or to display such a progress widget in the status bar of your main window. See package 'tk2' for further information.

Author(s)

Philippe Grosjean <phgrosjean@sciviews.org>

Examples

# 1) A simple progress indicator in %
cat("\nSimple progress indicator in %:\n")
for (i in 1:101) {
  progress(i)
  Sys.sleep(0.05)
}
cat("Done!\n")

# 2) A progress indicator with 'x on y'
cat("\nA variation...\n")
for (i in 1:31) {
  progress(i, 30)
  Sys.sleep(0.2)
}
cat("Done!\n")

---

**rmTemp**

*Remove one or several temporary variable(s) from the TempEnv environment*

Description

The function removes one or more variable(s) from TempEnv.

Usage

rmTemp(x)
Arguments

- The name of the variable (string), or a vector of characters with the name of all variables to remove from TempEnv

Value

Nothing (or a try-error object if it fails).

Warning

This command issues no error message if variable(s) do not exist in TempEnv!

Author(s)

Philippe Grosjean <phgrosjean@sciviews.org>

See Also

TempEnv, assignTemp, getTemp, existsTemp

Examples

assignTemp("test", TRUE)
# Retrieve this variable
getTemp("test")
# Remove it
rmTemp("test")
# Try to retrieve it again
getTemp("test")

tempdirWin() retrieves the Windows temporary directory, which is simply the root of tempdir() under Windows.

Usage

tempdirWin()

Value

A string with the path to the temporary directory

Author(s)

Philippe Grosjean <phgrosjean@sciviews.org>

See Also

userdir
tempvar

Get an arbitrary name for a temporary variable

Description
The function ensures that the variable name is cryptic enough and is not already used.

Usage
```
tempvar(pattern = "_.var")
```

Arguments
- **pattern**: The prefix for the variable (the rest is a random number)

Value
A string with the name of a variable

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

See Also
tempdirWin

Examples
```
tempvar()
```

userdir

Get the user directory

Description
```userdir()``` retrieves the current user directory.

Usage
```
userdir()
```

Value
A string with the path to the user directory

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

\texttt{tempdirWin}

Examples

\texttt{userdir()}
Package ‘svIO’

Title  SciViews GUI API - Data import/export

Depends  R (>= 2.1.0), svMisc, R2HTML

Imports  utils

Suggests  Hmisc

Description  Various functions to translate objects and exchange them with external applications

clippaste  Paste the content of the clipboard into an variable

Description

Create a new object by pasting the content of the clipboard in it. Contents generated with copy are recognized.

Usage

clippaste(name = "newobj", type = "ascii", objclass = "data.frame", pos = 1, ...)

Arguments

name  The name of the variable to create

 type  Specify the type of content (the format of the object currently in the clipboard)

 objclass  The class of the object to create

 pos  Where to place the new object (by default, in the global workspace)

 ...  Further arguments passed to the specific paste method

Value

The result returned by the specific paste method.

Author(s)

Eric Lecoutre

See Also

copy
### Examples

```r
## Not run:
# A trivial example... but that becomes more interesting if the copy
# and paste operations are made in different R instances
data(iris)
copy(iris, type = "ascii")
clippaste(iris2)

## End(Not run)
```

---

**copy**

*Convert an R object and copy it to the clipboard*

**Description**

This is a wrapper around `export`, using `file = "clipboard"`

**Usage**

```r
copy(x, type = "raw", objname = deparse(substitute(x)), ...)
```

**Arguments**

- `x`: The object to copy
- `type`: `type` gives the format in which the object should be converted. The list of recognized formats for this object is obtained with `type = "typelist"
- `objname`: The name of the copied object (by default, it is the name of the variable that holds it)
- `...`: Further arguments to use with the corresponding `export` method

**Value**

The result returned by the corresponding `export` method

**Author(s)**

Philippe Grosjean & Eric Lecoutre

**See Also**

`export`, `clippaste`

**Examples**

```r
## Not run:
data(iris)
copy(iris, type = "ascii")

## End(Not run)
```
export

Export the content of an object in a foreign format

Description

export is a generic function to convert objects into various foreign formats and write it to a file or a connection.

Usage

```r
export(x, type = "raw", file, append = FALSE, objname = deparse(substitute(x)), ...)
```

Arguments

- **x**: The object to export
- **type**: type gives the format in which the object should be converted. The list of recognized formats for this object is obtained with `type = "typelist"`
- **file**: The path to the file where the converted object should be extracted
- **append**: Do we append this object to this file?
- **objname**: The name of the object (by default, it is the name of the variable that holds it)
- **...**: Further argument passed to specific export methods

Value

Return the result from specific exportation methods (invisibly if the file is written)

Author(s)

Philippe Grosjean & Eric Lecoutre

See Also

copy

Examples

```r
## Not run:
data(iris)
export(iris, type = "ascii", file = "iris.txt")
# Inspect the file 'iris.txt', and then...
unlink("iris.txt")

## End(Not run)
```
Package ‘svViews’

Title SciViews GUI API - Views and report features

Depends R (>= 2.1.0), svMisc, R2HTML

Imports utils, lattice, MASS

Description Views are HTML presentations of the content of R objects, combining text, tables and graphs in the same document.

**guiViews**

*GUI functions to manipulate and generate Views*

Description

These function are internally used, and are not intended for direct manipulation.

Usage

```r
guiViewsCmd(type, typelist, command, file, ...)  
guiViewsCSS()  
guiViewsCSSChange(newCSS = "")  
guiViewsDir()  
guiViewsDisplay(viewfile, viewer = getOption("viewer"), ...)  
guiViewsFile()
```

Arguments

- **type**
  - The type of View
- **typelist**
  - The list of all existing View types for this object
- **command**
  - The code interpreted by the Viewer to switch to a different View
- **file**
  - The file containing the View
- **newCSS**
  - The path to the new CSS (Cascading Style Sheet) file to use
- **viewfile**
  - The file containing the View
- **viewer**
  - The viewer application used to display the View
- **...**
  - Further arguments passed to functions that generate the View
Value

guiViewsCSS(), guiViewsDir(), and guiViewsFile() return the corresponding file or directory. guiViewsCSSChange() return either TRUE or FALSE. guiViewsCmd() returns a string with the command. guiViewsDisplay() return the result returned by a call to the viewer.

Author(s)

Philippe Grosjean & Eric Lecoutre

See Also

view

Examples

guiViewsFile() # The name of temporary file suitable to hold a new View

Description

Given a SciViews-compatible reporting application (like SciViews R Report or Microsoft Word) is parameterized as the default application to use, this function generates and append a View to the current report.

Usage

```
report(x, objname = deparse(substitute(x)), reptype = getOption("report"),
application = getOption("reporter"), bookmark = "<End>", ...)
reportGraph(device = dev.cur(), reptype = getOption("report"),
application = getOption("reporter"), bookmark = "<End>", multiformat = FALSE,
dir = file.path(tempdir(), "svGraph"), width = 480, height = 480,
points = 12, bg = "transparent", ...)
```

Arguments

- **x**: An object
- **objname**: A name for this object (its own name, by default)
- **reptype**: The type of report. By default, it is "html" format. An alternative is "doc" for a Microsoft Word report
- **device**: The device where the graph should be appended to the report
- **application**: The reporter application, in case of reptype == "html"
- **bookmark**: A bookmark to select in a Word document before pasting the data. If "<End>" (by default), the end of the document is selected (data is appended). If NULL, then a GoTo dialog box is first displayed in Word to select a location before pasting. If a string is given, it must match a valid bookmark in the active Word document, otherwise an error will be issued.
multiformat: Do we use multiple formats (PNG + PDF, currently), or not (PNG only, default value)
dir: The directory where to place resulting files
width: The width of the generated image, in pixels
height: The height of the generated image, in pixels
pointsize: The pointsize (font size) to use
bg: The color of the background (transparent, by default)
... Further argument to use to generate the View or the graph

Details

ReportGraph() prepares graph outputs, so that they are suitable to be incorporated in a report. For an HTML report, it is a PNG image (plus a PDF one, if multiformat = TRUE). You can specify an option to tell which type of report is used by default: define options(report = "html") (used by default) or options(report = "doc"). For an "html" type, you need also to give the path to a compatible application like SciViews R Report. You can indicate it with options(reporter = "path_to_the_app")

Value

Returns invisibly TRUE for report() and the path to the created PNG/EMF file for reportGraph.

Author(s)

Eric Lecoutre & Philippe Grosjean

See Also

view

Examples

```r
## Not run:
data(iris)
report(iris)
## End(Not run)
```

Description

Views are rich-formatted presentations of the content of R objects. They provide an easy way to inspect them, combining formatted text, tables and graphs.

Usage

```r
view(x, type = "summary", objname = deparse(substitute(x)), ...)
```
viewHTMLinit

Initialize a View

Create a HTML file and place a correct header in it.

Usage

```
viewHTMLinit(objname, type, file = guiViewsFile(), CSSFile = guiViewsCSS(),
```

Arguments

- **objname**: Name of the object
- **type**: Type of View
- **file**: Name of the file where the View is created
- **CSSFile**: Name of the CSS (Cascading Style Sheet) file used
- **command**: The command to use for switching to a different view (internally used by SciViews-compatible applications)
- **view.title**: If TRUE, a title is added, otherwise no
wordReport

Value

Returns the file name invisibly.

Author(s)

Eric Lecoutre & Philippe Grosjean

See Also

view

wordReport

Manage a report in Microsoft Word from within R

Description

These various functions allow to control Microsoft Word (currently, only through DDE under Windows) to append results to a document from within R.

Usage

WordOpen(report = NULL, warn = TRUE)
WordActivate(async = FALSE)
WordExit()
WordGoto(bookmark = NULL)
WordGotoEnd()
WordInsertPara()
WordInsertPageBreak()
WordInsertText(text)
WordInsertFile(file, keep.bookmark = FALSE)
WordInsertPictureFile(file, keep.bookmark = FALSE)
WordExec(command, type = "DDE", async = FALSE)

Arguments

report
The path to a Word .doc file to use as a starting basis for the report. If report == NULL (by default), Word is just opened with a new empty document, if it was not already running.

warn
Should the function issue a warning if it cannot find or start Microsoft Word?

async
Would the command run asynchronously (returns immediately, before the command is fully processed by Word), on not (by default).

bookmark
A bookmark in the current Word document to select. If bookmark = NULL (by default), the GoTo dialog box of Word is opened, and you have an opportunity to navigate to a given place in your Word document before continuing (to indicate where to place the next piece of report in your file). If you specify bookmark = "", the insertion point is moved to the beginning of the current document.

text
A text to insert in the Word document. Note that you can use \n or \v for forced line breaks inside a paragraph, \t for going to the next paragraph, \f for inserting a page break and \t for a tabulation.
**file**

The path to a file containing data or an image to insert in the current Word document. All formats supported by Microsoft Word are accepted and automatically converted (e.g., HTML, RTF, ... and for images, BMP, TIFF, GIF, WMF, EMF, ...)

**keep.bookmark**

A flag to indicate that we want to keep the currently selected bookmark. When using `WordInsertFile()` or `WordInsertPictureFile()` with this option set to FALSE, any selected bookmark is deleted. With this option set to TRUE, the bookmark is preserved, but note that it requires a non-empty selection in the active Word document!

**command**

A command to execute in Word

**type**

The type of communication to use with Word. Currently, only DDE is valid.

### Details

These are various functions to control Microsoft Word: open it, open or activate a given document, insert text, pictures, or other items, and select a position in the document. All functions return invisibly `TRUE` in case of success and `FALSE` otherwise. Note that `WordGoto("my bookmark")` will fail if there is no bookmark called `my bookmark` in the active document... and R will not be advised of the error, that is, the function still returns `TRUE` in this case! This is because there is no feedback mechanism from Word to R implemented yet! `WordExec()` is the low-level function that allows to pass directly WordBasic commands to Word. It is not normally used directly, but it can be useful for commands not included in the other functions. Look at the WordBasic help and the code of the other functions to learn how you should enter these commands...

### Value

Returns invisibly `TRUE` for `report()` and the path to the created PNG file for `reportGraph()`.

### Author(s)

Philippe Grosjean <phgrosjean@sciviews.org>

### See Also

`report`

### Examples

```r
## Not run:
WordOpen() # Open Microsoft Word with a new empty document (or activate current)
WordGotoEnd() # Move insertion point to the end of that document
# Insert some text in it
WordInsertText("Some \ttext...\nOther text after line feed in same para\rOther para\fOther page")
# Insert a new paragraph
WordInsertPara()

# Open an existing Word document (it must exist there, of course!)
WordOpen("c:/temp/test.doc")
WordGoto() # Display the GoTo dialog box in Word, and allow to navigate to
WordInsertPictureFile("c:/temp/test.emf") # Insert a picture there
WordGoto("my bookmark") # Go to the "my bookmark" bookmark (it must exist)
WordInsertFile("c:/temp/test.html", "testit") # Insert a piece of HTML there
```
WordExit()  # Close word, possibly prompting to save all unsaved documents.
## End(Not run)
Package ‘svDialogs’

Title SciViews GUI API - Dialog boxes
Depends R (>= 2.1.0), tcltk
Imports svMisc
Description Rapidly construct dialog boxes for your GUI, including an automatic function assistant

---

display The display method (for flexible dialog boxes)

Description
Display a dialog box previously calculated. For instance, display a flexible dialog box.

Usage
```r
display(x, ...)  
display(x, parent = 0, GUI = getOption("guiWidgets"), debug = FALSE, ...)
```

Arguments
- `x`: An object that accepts a display method
- `parent`: Which window is the parent of this flexible dialog box?
- `GUI`: Which GUI to use to display this dialog box?
- `debug`: Do we debug this dialog box?
- `...`: Further arguments passed to the display method

Value
The R code constructed by this dialog box

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

See Also
guiDlg, guiEval

Examples
```r
### TO DO...
```
guiDlg  

Create a flexible dialog box

Description

This function creates a flexible dialog box, constituted by one or several 'panes'

Usage

guiDlg(title = "Input", message = NULL, help = NULL, sep = NULL, width = 50,  
labelwidth = 0, panes = list(list(type = "entry", message = "Enter data:",  
default = NULL)), GUI = getOption("guiWidgets"))

Arguments

title  The title of the dialog box
message The message to display on top of the dialog box
help    The help topic to trigger with the button 'help'
sep     Do we use separators?
width   The width of the dialog box
labelwidth The width reserved for labels
panes   A list with the panes to add to this flexible dialog box
GUI     Which GUI toolkit do we use?

Value

Returns a guiDlg object that can be shown using display method.

Author(s)

Philippe Grosjean (phgrosjean@sciviews.org)

See Also

display, guiEval, guiPane.tcltk

Examples

### TO DO...
**guiDlgAssistant**  
*An assistant dialog box*

---

**Description**

The assistant allows step by step execution of a given task.

**Usage**

`guiDlgAssistant(...)`

**Arguments**

...  
Arguments for the assistant dialog box

**Value**

This function is invoked for its side effect: to display an assistant dialog box.

**Author(s)**

Philippe Grosjean ([phgrosjean@sciviews.org](mailto:phgrosjean@sciviews.org))

**See Also**

`guiDlgFunction`

**Examples**

```r
### TO DO...
```

---

**guiDlgColor**  
*A color selection dialog box*

---

**Description**

Allows to select a color graphically.

**Usage**

`guiDlgColor(...)`

**Arguments**

...  
Further arguments to the color dialog box

**Value**

Returns selected colors.
guiDlgDir

Author(s)
Philippe Grosjean (⟨phgrosjean@sciviews.org⟩)

See Also
guiDlgFont, guiDlgGraphOptions

Examples

#### TO DO...

guiDlgDir      A select directory dialog box

Description
Allows to easily select a directory

Usage

guiDlgDir(title = "Select a directory", dir = getwd(), new = TRUE, parent = 0, GUI = getOption("guiWidgets"))

Arguments

title            the title of the dialog box
dir              the default directory
new              is a new directory allowed?
parent           the parent of this dialog box (not implemented yet)
GUI              which GUI widgets do you want to use (tcltk, wxPython, ...)

Value
The choosen directory.

Author(s)
Philippe Grosjean (⟨phgrosjean@sciviews.org⟩)

See Also
guiDlgOpen, guiDlgSave

Examples

#### TO DO...
**guiDlgDoubleList**  
*A double list dialog box*

**Description**
Select items in two (un)related lists at once

**Usage**
```r
guiDlgDoubleList(list1, list2, title = "Select", default1 = ", default2 = ", multi = c(TRUE, TRUE), new = c(FALSE, FALSE), sort = c(TRUE, TRUE), transfer = FALSE, parent = 0, GUI = getOption("guiWidgets"))
```

**Arguments**
- `list1` The first list items
- `list2` The second list items
- `title` A title for this dialog box
- `default1` Selected items by default for the first list
- `default2` Selected items by default for the second list
- `multi` Do we allow multiple selection for first and second list
- `new` Do we allow to create new items for first and second list
- `sort` Do we sort first and second list
- `transfer` The transfer mode allows to transfer items form one list to the other
- `parent` Which window is the parent of this dialog box
- `GUI` Which GUI toolkit would we use

**Value**
A list of two lists with selected items in both lists.

**Author(s)**
Philippe Grosjean (phgrosjean@sciviews.org)

**See Also**
- `guiDlgList`

**Examples**
```r
### TO DO...
```
guiDlgFont  
A font selection dialog box

Description
Select a font and some attributes

Usage
guiDlgFont(...)

Arguments
... further arguments to the function

Value
A string describing the selected font.

Author(s)
Philippe Grosjean (⟨phgrosjean@sciviews.org⟩)

See Also
guiDlgOptions

Examples
#### TO DO...

---

guiDlgFormula  
A formula dialog box

Description
Eases the creation of formulas

Usage
guiDlgFormula(...) 

Arguments
... further arguments to the function

Value
A formula build with this dialog box
guiDlgFunction

Author(s)
Philippe Grosjean (⟨phgrosjean@sciviews.org⟩)

See Also
guiDlgFunction

Examples
#### TO DO...

---

guiDlgFunction A dialog box prompting for function arguments

Description
Given a function, this dialog box prompts for values for its arguments

Usage
guiDlgFunction(fun, template = NULL, maxargs = 7, var = "res", width = 40, labelwidth = 10, displayit = TRUE, execfun = getOption("guiExecFun"))

Arguments
- **fun** The name of the function to prompt for
- **template** An optional template for the dialog box
- **maxargs** The maximum number of arguments to include in the dialog box
- **var** The default name of the variable where to assign result
- **width** The width of the dialog box
- **labelwidth** The width reserved for labels
- **displayit** Do we display it, or just construct it?
- **execfun** The function to use to execute resulting code

Value
A line of R code generated by the dialog box

Author(s)
Philippe Grosjean (⟨phgrosjean@sciviews.org⟩)

See Also
guiDlgInput, guiDlgVarSel

Examples
#### TO DO...
guiDlgGraphOptions  *A graph options dialog box*

**Description**
Change various graph options at once with this dialog box

**Usage**
guiDlgGraphOptions(...)

**Arguments**
... further arguments to the function

**Value**
This function is invoked for its side effect to change graph options with a GUI dialog box.

**Author(s)**
Philippe Grosjean (phgrosjean@sciviews.org)

**See Also**
guiDlgOptions

**Examples**

```r
### TO DO...
```

---

guiDlgGrid  *A grid (data display/editor) window*

**Description**
Display or edit matrices or data frames with this grid window

**Usage**
guiDlgGrid(table, title = deparse(substitute(table)), edit = TRUE, edit.vars = TRUE, add.vars = TRUE, add.rows = TRUE, parent = -1, GUI = getOption("guiWidgets"))
guiDlgInput

Arguments

- **table**: The table to display or edit in the grid
- **title**: The title of the grid window
- **edit**: Can we edit its content?
- **edit.vars**: Can we edit the name of its variables?
- **add.vars**: Can we add new variables?
- **add.rows**: Can we add rows?
- **parent**: Which window is the parent of the grid window
- **GUI**: Which GUI toolkit do we use?

Value

A modified version of the table if `edit == TRUE`

Author(s)

Philippe Grosjean (phgrosjean@sciviews.org)

See Also

- guiDlgText

Examples

```r
# TO DO...
```

---

**guiDlgInput**

A simple input box

Description

Enter one (textual) value with this input box

Usage

```r
guiDlgInput(message = "Enter a value", title = "Input", default = ", parent = 0
```

Arguments

- **message**: the message to display on top of the text box
- **title**: the title of the dialog box
- **default**: the default value in the text box
- **parent**: the parent of this dialog box (not implemented yet)
- **GUI**: which GUI widgets do you want to use (tcltk, wxPython, ...)

Value

The text entered by the user at the input box
guiDlgItemSel

Author(s)

Philippe Grosjean (⟨phgrosjean@sciviews.org⟩)

See Also

guiDlgMessage, guiDlgText

Examples

#### TO DO...

guiDlgItemSel A select item(s) dialog box

Description

Select item(s) in a list or a data frame

Usage

guiDlgItemSel(list, classes = NULL, title = "Select items", default = ",
default.items = ", all.names = FALSE, multi = FALSE, sort = TRUE,
sort.items = FALSE, subset = FALSE, save.restore = FALSE, parent = 0, GUI =

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>list</td>
<td>The list to display in the dialog box</td>
</tr>
<tr>
<td>classes</td>
<td>Classes to use</td>
</tr>
<tr>
<td>title</td>
<td>The title of the dialog box</td>
</tr>
<tr>
<td>default</td>
<td>The default value</td>
</tr>
<tr>
<td>default.items</td>
<td>The default items selected</td>
</tr>
<tr>
<td>all.names</td>
<td>Do we use all.names?</td>
</tr>
<tr>
<td>multi</td>
<td>Is a multiple selection allowed?</td>
</tr>
<tr>
<td>sort</td>
<td>Do we sort the list?</td>
</tr>
<tr>
<td>sort.items</td>
<td>Do we sort items in the list?</td>
</tr>
<tr>
<td>subset</td>
<td>Do we allow subsetting the list?</td>
</tr>
<tr>
<td>save.restore</td>
<td>Do we save restore information?</td>
</tr>
<tr>
<td>parent</td>
<td>Which window is the parent of this dialog box?</td>
</tr>
<tr>
<td>GUI</td>
<td>Which GUI toolkit do we use?</td>
</tr>
</tbody>
</table>

Value

A list containing selected items

Author(s)

Philippe Grosjean (⟨phgrosjean@sciviews.org⟩)
See Also

`guiDlgVarSel`

Examples

#### TO DO...

---

**guiDlgList**  
* A list selection dialog box

**Description**

Select an item in a list

**Usage**

```r
guiDlgList(items, multi = FALSE, message = if (multi) "Select item(s):" else "Select one item:", title = "Selection", default = 1, nsel = NULL, new = FALSE, sort = FALSE, width = 50, parent = 0, GUI = getOption("guiWidgets"))
```

**Arguments**

- `items`: The list of items
- `multi`: Is it a multiple selection dialog box?
- `message`: The message to display on top of the list
- `title`: The title of the dialog box
- `default`: The default selected item
- `nsel`: Maximum number of selected items allowed
- `new`: Do we allow to create new items?
- `sort`: Do we sort the list?
- `width`: The width of the dialog box
- `parent`: Which window is the parent of this dialog box?
- `GUI`: Which GUI toolkit do we use for this dialog box?

**Value**

A list with selected items

**Author(s)**

Philippe Grosjean (⟨phgrosjean@sciviews.org⟩)

**See Also**

`guiDlgDoubleList`

**Examples**

#### TO DO...
guiDlgMessage  

function to do ...

Description

A concise (1-5 lines) description of what the function does.

Usage

guiDlgMessage(message, title = "Message", type = c("ok", "okcancel", "yesno", "yesnocancel"), default = 1, icon = c("info", "question", "error", "warning"), parent = 0, GUI = getOption("guiWidgets"))

Arguments

message  the message to display in the dialog box

title  the title of the dialog box

type  the type of dialog box: ok, okcancel, yesno or yesnocancel

default  the default button (if more than one)

icon  the icon: info, question, error or warning

parent  the parent of this dialog box (not implemented yet)

GUI  which GUI widgets do you want to use (tcltk, wxPython, ...)

Value

A string with the caption of the button that the user pressed.

Author(s)

Philippe Grosjean (phgrosjean@sciviews.org)

See Also

guiDlgInput

Examples

#### TO DO...
guiDlgOpen

An open file(s) dialog box

Description

Allows to select one or more existing files to be opened

Usage

guiDlgOpen(title = "Select file", defaultFile = "", defaultDir = "", multi = FALSE, filters = c("All files (.**)", ".*.*"), parent = 0, GUI = getOption("GUIwidgets"))

Arguments

title the title of the dialog box
defaultFile the file name to display by default
defaultDir the default dir where to open the dialog box
multi is it allowed to select several files at once?
filters the filters to use. It is a n*2 matrix of characters, with first column being the label and the second being the mask. For instance: "Text files", "*.txt".
parent the parent of this dialog box (not implemented yet)
GUI which GUI widgets do you want to use (tcltk, wxPython, ...)

Value

A list with the path to the selected files.

Author(s)

Philippe Grosjean (<phgrosjean@sciviews.org>)

See Also

guiDlgSave, guiDlgDir

Examples

#### TO DO...
guiDlgOptions  A general options dialog box

Description
Display and allow to change various options

Usage
guiDlgOptions(...)

Arguments
... further arguments to the function

Value
This function is invoked for its side effect to change options through a dialog box

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

See Also
guiDlgGraphOptions

Examples
### TO DO...

guiDlgProgress  A progress window

Description
Display the progression of a long task

Usage
guiDlgProgress(value, range = c(0, 100), message = "Please wait...", title = "Progress", percent = TRUE, cancel = TRUE, icon = "none", parent = -1, GUI = getOption("GUI"), ...)
**guiDlgSave**

**Arguments**

- **value**: The current value of the progression
- **range**: The range for the progression
- **message**: The message to display on top of the progression bar
- **title**: The title of the dialog box
- **percent**: Is it a progression in percent?
- **cancel**: Is the cancel button enabled?
- **icon**: An icon to display on the top left part of the dialog box
- **parent**: Which window is the parent of this dialog box?
- **GUI**: Which GUI toolkit to use to display this dialog box?

**Value**

Nothing

**Author(s)**

Philippe Grosjean ([phgrosjean@sciviews.org](mailto:phgrosjean@sciviews.org))

**See Also**

- guiDlgMessage

**Examples**

```r
### TO DO...
```

---

**guiDlgSave**  
a save to file dialog box

**Description**

Use this dialog box to ask a name for saving a file

**Usage**

```r
guiDlgSave(title = "Save As", defaultFile = "", defaultDir = "", defaultExt = "")
```

**Arguments**

- **title**: the title of the dialog box
- **defaultFile**: the default file name
- **defaultDir**: the default directory where to open this dialog box
- **defaultExt**: the default extension for the file
- **filters**: the filters to use. It is a n*2 matrix of characters, with first column being the label and the second being the mask. For instance: "Text files", "*.txt".
- **parent**: the parent of this dialog box (not implemented yet)
- **GUI**: which GUI widgets do you want to use (tktk, wxPython, ...)

---
guiDlgText

Value

The path to the file where to save data.

Author(s)

Philippe Grosjean (phgrosjean@sciviews.org)

See Also

guiDlgOpen

Examples

### TO DO...

---

A simple text display/editor window

Description

Display or edit the content of ASCII files

Usage

```r
guiDlgText(text, file = NULL, title = deparse(substitute(text)), edit = TRUE,
submit = TRUE, parent = -1, GUI = getOption("guiWidgets"))
```

Arguments

text The text to display
file The name of a file, if the text is contained in a file
title The title of the window
edit Do we allow to edit this text?
submit If this text is R code, allow to submit it to R
parent Which window is the parent of this text window
GUI Which GUI toolkit to use to display this window?

Value

The modified text, or the path to the file containing the modified text.

Author(s)

Philippe Grosjean (phgrosjean@sciviews.org)

See Also

guiDlgGrid

Examples

#### TO DO...
guiDlgVarSel

A select variable dialog box

Description

Present a list of variables of a given class and allow to choose one or several

Usage

guiDlgVarSel(list, classes = NULL, title = "Select a variable", default = "", all.names = FALSE, multi = FALSE, new = FALSE, sort = TRUE, parent = 0, GUI = getOption("guiWidgets"))

Arguments

list A list of variables
classes Classes for the variables to search for
title A title of the dialog box
default The default selected item
all.names Do we search for all names (including variables starting with ".")?
multi Do we allow multiple selection?
new Do we allow to create new variables?
sort Do we sort the list?
parent Which window is the parent of this dialog box?
GUI Which GUI toolkit do we use?

Value

A list with selected variables names

Author(s)

Philippe Grosjean ([phgrosjean@sciviews.org](mailto:phgrosjean@sciviews.org))

See Also

guiDlgItemSel

Examples

```r
### TO DO...
```
guiDlgView

A view window

Description
Display a view window. A view is an HTML presentation of an R object.

Usage
```
guiDlgView(file, CSSfile, title = "View", report = TRUE, parent = -1, GUI = getOption("guiWidgets"))
```

Arguments
- **file**: The path of the file that contains the view
- **CSSfile**: The CSS file (Cascaded Style Sheet) to use
- **title**: The title of the view window
- **report**: Do we allow to report this view?
- **parent**: Which window is the parent of this view window
- **GUI**: Which GUI toolkit to use for this view window

Value
Nothing

Author(s)
Philippe Grosjean (⟨phgrosjean@sciviews.org⟩)

See Also
- `guiDlgText`

Examples
```
#### TO DO...
```

guiEval

Evaluate R code constructed using a flexible dialog box

Description
This function is usually called by a flexible dialog box to execute R code that was build with it.

Usage
```
guiEval(code, ident = "GUI ")
```

```
guiPane.tcltk

Arguments

code A string wuth some R code
ident Do we identify this execution (change the prompt)

Value

The result of the evaluated R code

Author(s)

Philippe Grosjean (⟨phgrosjean@sciviews.org⟩)

See Also

guiDlg.display

Examples

#### TO DO...

---

guiPane.tcltk Construct a 'pane', that is a zone in a flexible dialog box

Description

This function construct a 'pane'. Panes are then assembled to form a flexible dialog box.

Usage

guiPane.tcltk(dlg, item, ...)
guiPane.entry.tcltk(paneFrame, butArg, onArgEdit, varUseIt, dlg, item, ...)
guiPane.list.tcltk(paneFrame, butArg, onArgEdit, varUseIt, dlg, item, ...)

Arguments

dlg A flexible dialog box
paneFrame The frame where this pane should be incrusted
butArg Arguments for buttons
onArgEdit Edit it on arg?
varUseIt var uses it?
item An item in this dialog box
... further arguments passed to the function

Value

Build a pane in a flexible dialog box

Author(s)

Philippe Grosjean (⟨phgrosjean@sciviews.org⟩)
guiSetStyle.tcltk

See Also

guiDlg

Examples

### TO DO...

---

`guiSetStyle.tcltk` *Change the style of compatible tcltk dialog boxes*

Description

This functions switches to a different style for compatible tcltk dialog boxes (fixed and flexible dialog boxes defined in this package).

Usage

```r
guiSetStyle.tcltk(style = c("classic", "SciViews", "Rcmdr", "large"), ask = FALSE)
```

Arguments

- **style**  
  The style to use
- **ask**  
  Do we ask the user?

Value

This function is invoked for its side effect to change the default style of tcltk dialog boxes

Author(s)

Philippe Grosjean (phgrosjean@sciviews.org)

See Also

guiDlg

Examples

### TO DO...
Package ‘svSocket’

Title SciViews GUI API - R Socket Server

Depends R (>= 2.1.0), tcltk

Imports svMisc

Description Implements a simple socket server allowing to connect GUI clients to R

getSocketClients  Get infos about socket clients

Description
List all clients currently connected to a given R socket server.

Usage
getSocketClients(port = 8888)

Arguments
port The port of the R socket server.

Value
A vector of character string with the address of clients in the form XXX.XXX.XXX.XXX:YYY
where XXX.XXX.XXX.XXX is their ip address and YYY is their port. The name of these items
equals the corresponding Tcl socket name.

Author(s)
Philippe Grosjean ⟨phgrosjean@sciviews.org⟩

See Also
getSocketServers

Examples
  # TO DO...
**getSocketServerName**
*Get the name of a R socket server*

Description
Get the internal name given to a particular R socket server.

Usage
`getSocketServerName(port = 8888)`

Arguments
- **port**
  THE port of the R socket server.

Value
A string with the server name, or NULL if it does not exist.

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

See Also
`getSocketServers`

Examples
```r
##TO DO...
```

**getSocketServers**
*Get the ports of current R socket servers.*

Description
Returns a list with all the ports of currently running R socket servers.

Usage
`getSocketServers()`

Value
A character vector, or NULL if no R socket server is currently running.

Author(s)
Philippe Grosjean ((phgrosjean@sciviews.org))
processSocket

See Also

getSocketClients, getSocketServerName, startSocketServer

Examples

```r
## TO DO...
```

---

**processSocket**  
*The function that processes a command from the socket.*

**Description**

This is the default R function called each time data is send by a client through a socket. It is possible to customize this function and to use customized versions for particular R socket servers.

**Usage**

```r
processSocket(msg)
```

**Arguments**

- `msg`  
The message send by the client, to be processed.

**Value**

Returns a string with the result of the process.

**Author(s)**

Philippe Grosjean (phgrosjean@sciviews.org)

**See Also**

startSocketServer, sendSocket

**Examples**

```r
## TO DO...
```
**sendSocket**

*Send data to a client through a socket.*

**Description**

The text is send to one of the R socket server’s clients currently connected.

**Usage**

```r
text, socket)
```

**Arguments**

- `text`: The text to send to the client.
- `socket`: The Tcl name of the client socket currently connected.

**Author(s)**

Philippe Grosjean (⟨phgrosjean@sciviews.org⟩)

**See Also**

`processSocket`

**Examples**

```r
##TO DO...
```

---

**startSocketServer**

*Start and stop a R socket server.*

**Description**

A R socket server is listening for command send by clients to a TCP port. This server is implemented in Tcl/Tk, using the powerful `socket` command. Since it runs in the separate tcltk event loop, it is not blocking R, and it runs in the background; the user can still enter commands at the R prompt while one or several R socket servers are running and even, possibly, processing socket clients requests.

**Usage**

```r
startSocketServer(port = 8888, server.name = "Rserver", procfun = processSocket)
stopSocketServer(port = 8888)
```

**Arguments**

- `port`: The TCP port of the R socket server
- `server.name`: The internal name of this server
- `procfun`: The function to use to process client’s commands. By default, it is `processSocket()`. 
Details

One can write a different procfun function than the default one for special servers. That function must accept one argument (a string with the command send by the client) and it must return a character string containing the result of the computation.

Note

This server is currently synchronous in the processing of the command. However, neither R, nor the client are blocked during exchange of data (communication is asynchronous).

Author(s)

Philippe Grosjean (⟨phgrosjean@sciviews.org⟩)

See Also

processSocket

Examples

##TO DO...
Package ‘svGUI’

**Title**  SciViews GUI API - Main GUI features

**Depends**  R (>= 2.1.0), tcltk, R2HTML, svMisc, svIO, svViews

**Imports**  grDevices

**SystemRequirements**  SciViews-R (the GUI client under Windows) can be obtained from http://www.sciviews.org/SciViews-R

**Description**  Functions to communicate with a GUI client, to implement an object browser, etc...

---

**Usage**

```r
Windows(width = 7, height = 7, pointsize = 12, record = getOption("graphics.record"),
  rescale = c("R", "fit", "fixed"), xpinch, ypinch, bg = "transparent", canvas = "white",
  gamma = getOption("gamma"), xpos = NA, ypos = NA, buffered = getOption("windowsBuffered"))
```

**Arguments**

- `width`  The width of the graph device
- `height`  The height of the graph device
- `pointsize`  The base size of the font
- `record`  Should recording mode be turned on?
- `rescale`  The way the graph is rescaled
- `xpinch`  X aspect ratio of the screen
- `ypinch`  Y aspect ratio of the screen
- `bg`  Color for the background
- `canvas`  Color for the canvas
- `gamma`  Gamma correction to apply
- `xpos`  X position of the window
- `ypos`  Y position of the window
- `buffered`  Should the window be buffered?
guiInfo

Value

A new graph device is created.

Author(s)

Philippe Grosjean ([phgrosjean@sciviews.org])

See Also

guiStart

Examples

#### TO DO...

---

**guiInfo**

*Get an info tip or a completion list about an object*

Description

This function provides call tip, or completion list to GUI clients.

Usage

`guiInfo(fname, firstarg = NULL, type = "tip")`

Arguments

- **fname**: The name of the function or of the object
- **firstarg**: The first argument (to determine which method is called, not used yet)
- **type**: The type of info required. Currently, 'tip', 'list' or 'listitems'

Value

The content of the tip, or of the list.

Author(s)

Philippe Grosjean ([phgrosjean@sciviews.org])

See Also

guiObjList

Examples

#### TO DO...
guiObjBrowse

Function to implement an object browser

Description

These functions provide features required to implement a complete object browser in the GUI client.

Usage

```r
guiObjBrowse(id = "default", env.name = NULL, pos = NULL, all.names = NULL,
             pattern = NULL, group = NULL, regenerate = FALSE)
guiObjClear(id)
guiObjDir()
guiObjList(id = "default", env.name = NULL, pos = 1, all.names = FALSE,
            pattern = "", group = "", path = NULL, compare = TRUE)
guiObjMenu(id = "default", selobject, objects, envir, path = NULL)
guiObjSearch(path = NULL, compare = TRUE)
```

Arguments

- `id`: The id of the object browser (you can run several ones concurrently)
- `env.name`: The name of the environment
- `pos`: The position in the search path
- `all.names`: Do we display all names (including hidden variables starting with `.`)?
- `pattern`: A pattern to match for selecting variables
- `group`: A group to filter
- `regenerate`: Do we force to regenerate the information?
- `path`: The path where to write a temporary file with the requested information
- `compare`: If `compare == TRUE`, result is compared with last cached value and the client is updated only if something changed
- `selobject`: The current selection in the object browser
- `objects`: A list with selected items in the object browser
- `envir`: The currently selected environment in the object browser

Details

guiObjBrowse do the horsework. guiObjDir gets the temporary directory where exchange files are stored. guiObjList lists objects in a given workspace. guiObjSearch lists the search path. guiObjClear clears any reference to a given object browser. guiObjMenu computes a context menu for a given object.

Value

- Depending on the function, a list, a string, or a reference to an external, temporary file where data is written.

Author(s)

Philippe Grosjean ((phgrosjean@sciviews.org))
See Also

`guiRemove`, `guiSave`

Examples

```r
#### TO DO...
```

---

**guiRemove**

Remove one or several objects (and prompt for confirmation)

---

**Description**

This function is intended to be called by an object browser context menu. It deletes variable(s) after prompting for confirmation.

**Usage**

```r
guiRemove(list = character(0), inherits = FALSE, ask = TRUE)
```

**Arguments**

- `list`: A list of variables to delete
- `inherits`: Do the variables inherit from lower environments in the search path?
- `ask`: Do we ask for confirmation to the user?

**Value**

Nothing if the function succeed.

**Author(s)**

Philippe Grosjean (phgrosjean@sciviews.org)

**See Also**

`guiObjBrowse`

**Examples**

```r
#### TO DO...
```
guiSave

Save one or several objects to a .Rdata file

Description

This function is intended to be called by an object explorer context menu. It saves one or several objects to a .Rdata file.

Usage

```r
guiSave(..., list = character(0), ascii = FALSE, version = NULL, envir = parent.frame(), compress = TRUE)
```

Arguments

- `...` The list of variables to save
- `list` Idem, but provided in a list
- `ascii` Should the data be written in ASCII or binary mode?
- `version` Which version of .Rdata to use
- `envir` From which environment objects should be retrieved?
- `compress` Do we compress the file?

Value

The path of the .Rdata file where variables are written.

Author(s)

Philippe Grosjean (⟨phgrosjean@sciviews.org⟩)

See Also

`guiObjBrowse`

Examples

```r
### TO DO...
```

guiStart

Install/Uninstall and Start/Stop a GUI client

Description

A GUI needs several changes in R, and needs also a way to communicate with it. These functions provide required features for SciViews compatible GUI clients.
Usage

```r
svGUI-internal
guiInstall(HideTkWindow = TRUE)
guiUninstall()
guiStart(gui = "\"%SciViews_Home%/bin/RConsole.exe\" -connect")
guiStop()
```

Arguments

- **HideTkWindow**: Whether the Tk communication window should be hidden or not
- **gui**: The path to the GUI client executable

Value

TRUE if success, error message otherwise.

Author(s)

Philippe Grosjean (<phgrosjean@sciviews.org>)

See Also

Windows

Examples

```r
### TO DO...
```

---

svGUI-internal

**Internal svGUI objects**

Description

Internal svGUI objects.

Details

These are not to be called by the user.
Package ‘svIDE’

**Title**  SciViews GUI API - IDE and code editor functions

**Depends**  R (>= 2.1.0), tcltk

**Imports**  utils, methods

**Description**  Supporting functions for the GUI API (create files and interact with external IDE/code editors)

---

**Args**  
Show function arguments in a human-readable way - get a call tip

---

**Description**

`Args()` displays function arguments in a better way than `args()` does. It is primarily intended for code tips in GUIs.

**Usage**

```r
Args(name, only.args = FALSE)
CallTip(code, only.args = FALSE, location = FALSE)
```

**Arguments**

- `name`  A string with the name of a function
- `code`  A fraction of R code ending with the name of a function, eventually followed by ‘(‘
- `only.args`  Do we return only arguments of the function (arg1, arg2 = TRUE, ...), or the full call, like (myfun(arg1, arg2 = TRUE, ...)).
- `location`  If TRUE then the location (in which package the function resides) is appended to the calltip between square brackets

**Value**

A string with the calling syntax of the function

**Note**

`Args()` is supposed to display S3/S4 methods, and primitives adequately,... but this is not implemented yet in the current version!
Complete

Get a completion list for a R code fragment

Description
A list with matching items is returned in a string.

Usage
Complete(code, givetype = FALSE, fieldsep = "|")

Arguments
- code: A fraction of R code needing completion
- givetype: Do we append the type of each entry to the list?
- fieldsep: The separator to use between the item and its type

Details
The completion list is context-dependent, and it calculated as if the code was entered at the command line.
If the code ends with $ or [ [, then the function look for items in a list or data.frame whose name is the last identifier.
If the code ends with @, then the function look for slots of the corresponding S4 object.
If the code ends with a partial identifier name, the function returns all visible matching keywords.
In all other situations, the function returns a list of the objects in .GlobalEnv.

Value
A string with the completion list

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

See Also
guiComplete
Examples

## a data.frame

```r
data(iris)
cat(Complete("item <- iris$"), "\n")
cat(Complete("item <- iris[[", TRUE), "\n")
```

## An S4 object

```r
setClass("track", representation(x = "numeric", y = "numeric"))
t1 <- new("track", x=1:20, y=(1:20)^2)
cat(Complete("item2 <- t1@", TRUE), "\n")
```

## A partial identifier

```r
cat(Complete("item3 <- va", TRUE), "\n")
```

## Otherwise, a list with the content of .GlobalEnv

```r
cat(Complete("item4 <- ", "\n")
```

Description

These functions provide features required to implement the Tinn-R object browser.

Usage

```r
trObjSearch(path = NULL, compare = TRUE)
trObjList(id = "default", env.name = NULL, pos = 1, all.names = FALSE,
          pattern = "", group = "", path = NULL, compare = TRUE)
```

Arguments

- `path`: The path where to write a temporary file with the requested information
- `compare`: If `compare == TRUE`, result is compared with last cached value and the client is updated only if something changed
- `id`: The id of the object browser (you can run several ones concurrently)
- `env.name`: The name of the environment
- `pos`: The position in the search path
- `all.names`: Do we display all names (including hidden variables starting with `.`)?
- `pattern`: A pattern to match for selecting variables
- `group`: The group to which variables belong

Details

`trObjSearch` list search paths, `guiObjList` lists objects in a given environment. These functions are not intended to be called directly by the end-user. They are used internally by Tinn-R to get data for its own R objects explorer. Note that, for Tinn-R to be able to run these commands and to communicate with R, you have to load the svIDE package. You must also activate one, or both communication means: sockets, by loading the svSocket package, or DDE, by starting the `guiDDEInstall()` function from the svIDE package (you can activate both as well).
Finally, you need also function from the svIO package. The best way to get R operational to communicate with Tinn-R from start is to add the following commands to `/etc/profile.site` in the R install directory:

```r
library(svIDE); library(svSocket); library(svIO);
guiDDEInstall()
```

**Value**

Depending on the function, a list, a string, or a reference to an external, temporary file where data is written.

**Author(s)**

Jose Claudio Faria (joseclaudio.faria@terra.com.br) & Philippe Grosjean (phgrosjean@sciviews.org)

**See Also**

`guiDDEInstall`

---

**createSyntaxFile**  
Create a syntax definition or a calltip file for R language

**Description**

A `.svl` syntax file describes the syntax of the language for SciViews GUIs. It is used mainly for syntax coloring of text in editors. The calltip file (by default, `Rcalltips.txt`) is a database with formal calls of R functions, to be used by code editors to display function calltips.

**Usage**

```r
createSyntaxFile(svfile = "R.svl", pos = 2:length(search()))
createCallTipFile(file = "Rcalltips.txt", pos = 2:length(search()),
    field.sep = ":", only.args = FALSE, return.location = FALSE)
```

**Arguments**

- `svfile`  
The name or location of the `.svl` file you want to create
- `file`  
The name or location of the calltip file you want to create
- `pos`  
A vector of integers indicating which positions in the search path should be recorded in the file
- `field.sep`  
The field separator to use between the function name and its calltip in the file
- `only.args`  
Do we record the full calltip (myfun(arg1, arg2 = TRUE, ...)), or only the function arguments (arg1, arg2, ...)
- `return.location`  
When TRUE, the package where this function is located in returned between square brackets at the end of the line

**Value**

These functions return nothing. They are invoked for their side effects of creating a file.
getFunctions

Note
SciViews-R uses a file named 'R.svl' and located in <SciViewsDir>/bin/languages. This function generates such a file. Do resist to the temptation to generate a very long list of keywords by loading many packages. SciViews cannot handle a list longer that 32k, that is roughly, 2000 - 2500 keywords.

createCallTipFile() sometimes issues warnings because it cannot get arguments from some keywords. You can ignore these warnings.

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

See Also
getFunctions, getKeywords, CallTip

Examples
## Not run:
# Create a syntax highlighting file for all currently loaded R packages
createSyntaxFile("Rtemp.svl")
# Show and delete it
file.show("Rtemp.svl", delete.file = TRUE)

# Create a calltips file for all currently loaded R packages
createCallTipFile("Rtemp.ctip", return.location = TRUE)
# Show and delete it
file.show("Rtemp.ctip", delete.file = TRUE)

# You can also customize the calltip file and select the package
# Here we include only functions from base package (the last item
# in the search path)
createCallTipFile("Rtemp2.ctip", pos = length(search()),
  field.sep = ";", only.args = TRUE)
# Show and delete it
file.show("Rtemp2.ctip", delete.file = TRUE)

## End(Not run)
getKeywords

Arguments

pos The position in the search path

Value

A list of character strings with functions names.

Author(s)

Philippe Grosjean <phgrosjean@sciviews.org>

See Also

getKeywords, createSyntaxFile

Examples

getFunctions(1) # Functions defined in .GlobalEnv
length(getFunctions(length(search()))) # Number of functions in package:base

getKeywords

get all keywords for syntax highlighting

Description

Get all visible keywords in one or several environment, excluding operators and reserved keywords.

Usage

getKeywords(pos = 2:length(search()))

Arguments

pos A vector of integers with all positions in the search path where to look at

Value

A vector of character strings with keywords.

Note

This function is used by createSyntaxFile() to list all keyword2 items (thus excluding reserved keywords and operators)

Author(s)

Philippe Grosjean <phgrosjean@sciviews.org>

See Also

getFunctions, createSyntaxFile
guiDDEInstall

Examples

guiDDEInstall()

desc

install a DDE server (Windows only) for external IDE/code editor

Description

These functions install and manage a DDE server to return context-sensitive calltips or completion lists to external IDE/code editors under Windows.

Usage

guiDDEInstall()
guiCallTip(code, file = NULL, onlyargs = FALSE, maxwidth = 60, location = FALSE)
guiComplete(code, file = NULL, givetype = FALSE, fieldsep = "|")

Arguments

code A piece of R code (in a character string) to analyze
file A file where to return the result ("", or NULL for none. You can use "clipboard" to send the result to the clipboard under Windows only; it raises an error currently on other platforms)
onlyargs Do we return the whole calltip or only the function arguments?
maxwidth Reformat the calltip to maxwith (use 0 for not reformatting it)
location
givetype Return also the type of each object in the completion list (possibly to use set a different icon in the list displayed by the external program)
fieldsep The separator to use between the item and its type in the list

Value

These functions should be used to interact with an external program. Their result is returned invisibly for debugging purposes and is not intended to be use in R.

Note

DDE is a communication protocol that exists only under Windows. Consequently, those functions cannot be used (yet) on other platforms.

On loading of the package, if the option(use.DDE = TRUE) is defined, the DDE server (guiDDEInstall()) is automatically installed when the package is loaded. Also if options(IDE = "[path.to.exe]") is defined, then that IDE is automatically started afterward.

Author(s)

Philippe Grosjean <phgrosjean@sciviews.org>

See Also

CallTip, Complete
## Not run:

## DDE exchange protocol is available ONLY under Windows!

## Also, this cannot be run by examples() but should be OK when pasted
## into an interactive R session with the tcltk package loaded

## Here is how you can test these features under Windows

```r
options(use.DDE = TRUE)
library(svIDE) # This should automatically start the
# DDE server named 'TclEval SciViewsR', according to the option set

# Get some data in the user workspace
data(trees)
a <- 1
b <- "some text in the first instance of R"
```

# To test these functions in an external program, we need now
# to start a second instance of R. In this second instance, enter:
library(tcltk)
.Tcl("package require dde")
.Tcl("dde services TclEval {}")
# You should get 'TclEval SciViewsR' in the list
# if the server in the first instance is running

# Now, request a calltip for the function 'ls'
# This is done in two steps:
# 1) Execute the command 'guiCallTip' with this code chunk as argument
.Tcl("dde execute TclEval SciViewsR (guiCallTip {res <- ls()}")
# 2) Retrieve the calltip from the variable SciViewsR_CallTip
.Tcl("dde request TclEval SciViewsR SciViewsR_CallTip")

# Another way to trigger DDE commands (for programs that do not support
# DDE is to use 'execdde.exe' of the tcltk2 package (see ?tk2dde)

# It is also possible to copy the calltip to a file, or to the clipboard
# by specifying it after the command (also the two additional arguments
# have their default values changed)
.Tcl("dde execute TclEval SciViewsR (guiCallTip {library()} clipboard TRUE 40 TRUE")
# Look at what is in the clipboard
cat(readClipboard(), "\n")

# The process is similar to get completion lists
.Tcl("dde execute TclEval SciViewsR (guiComplete {iris$})")
.Tcl("dde request TclEval SciViewsR SciViewsR_Complete")

# Get the list of variables in the user workspace of the first R instance
# into the clipboard (use also the other arguments to get the type of objects)
.Tcl("dde execute TclEval SciViewsR (guiComplete {} clipboard TRUE {-})")
# Look at what is in the clipboard
cat(readClipboard(), "\n")
```

## End(Not run)
Package ‘svWidgets’

Title  SciViews GUI API - Widgets & Windows

Depends  R (>= 2.1.0), tcltk

Imports  utils, svMisc

Suggests  tcltk2

Description  High level management of widgets, windows and other graphical resources

| Img | Manipulate image resources for the GUIs |

Description

Mechanism provided here is very simple and allows for automatic loading of image resources from any package subdirectory. Currently, only Tk images loaded from GIF files are supported... but more formats will be added in the future.

Usage

ImgGet(image)
ImgNames()
ImgType(image, warn = TRUE)
tkImgAdd(file, type = "gif")
tkImgDel(image)
tkImgReadPackage(package, subdir = "gui", type = "gif")

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>image</td>
<td>Name of an image</td>
</tr>
<tr>
<td>warn</td>
<td>Do we issue a warning if the type of image is not recognized?</td>
</tr>
<tr>
<td>file</td>
<td>Image file from where to download the resource</td>
</tr>
<tr>
<td>type</td>
<td>Type of image. Currently, only ’gif’ is supported.</td>
</tr>
<tr>
<td>package</td>
<td>Name of a package from where to load image resources</td>
</tr>
<tr>
<td>subdir</td>
<td>Subdirectory in the package where the graphical resources are stored. By default, it is the ”gui” subdirectory.</td>
</tr>
</tbody>
</table>
Details

These functions care about (un)loading image resources. A list of these resources is maintained in `.gui.Imgs` located in the TempEnv environment.

Value

`tkImgAdd()` and `ImgGet()` return the handle to the newly created image. `ImgNames()` return the list of all images registered in `.gui.Imgs`.

Author(s)

Philippe Grosjean

See Also

`WinNames`

Examples

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

ImgNames()
tkImgAdd(file.path(.path.package(package = "svWidgets")[1], "gui", "logoSciViews.png"))
ImgNames()
ImgType("$Tk.logoSciViews")
# Place that logo in a Tk window
timg <- tkWinAdd("timg", title = "A Tk window with image", pos = "-40+20")
labImg <- tklabel(timg, image = ImgGet("$Tk.logoSciViews"), bg = "white")
tkpack(labImg)
# When the image resource is deleted, it is not displayed any more, but there is
tkImgDel("$Tk.logoSciViews")
ImgNames()
tkWinDelete("timg")
# To read all image resources at once (place this in .Lib.first())
tkImgReadPackage("svWidgets")
ImgNames()

## End(Not run)
```

Menu

Conveniently manipulate menus, whatever the window

Description

These functions provide an unifying way of dealing with menus in R. It is ispired from the winMenuXXX() functions under Windows that allow easy manipulation of custom menus in Rgui.exe. Currently, they support all the functionnalities of winMenuXXX() functions and they bring a convenient similar layer on top of Tk menus to manipulate them in a similar way.
Menu

Usage

MenuAdd(menu, ...)  
MenuAddItem(menu, item, action, image = "", accel = "", options = "")  
MenuDel(menu)  
MenuDelItem(menu, item)  
MenuNames()  
MenuItems(menu)  
MenuType(menu, warn = TRUE)  
MenuChangeItem(menu, item, action = "", options = "")  
MenuStateItem(menu, item, active = TRUE)  
MenuRead(file = "Menus.txt")  
MenuReadPackage(package, subdir = "gui", file = "Menus.txt")

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>menu</td>
<td>Name of a menu</td>
</tr>
<tr>
<td>item</td>
<td>Name of a menu item</td>
</tr>
<tr>
<td>action</td>
<td>Action the menu triggers (R code)</td>
</tr>
<tr>
<td>image</td>
<td>Name of an image to display at left of the menu item</td>
</tr>
<tr>
<td>accel</td>
<td>Accelerator (keystroke) to use to trigger this menu item</td>
</tr>
<tr>
<td>options</td>
<td>Additional options, for instance <code>state = &quot;disable&quot;</code> to disable the menu at creation.</td>
</tr>
<tr>
<td></td>
<td>Further options</td>
</tr>
<tr>
<td>warn</td>
<td>Do we issue a warning if the type of menu is not recognized?</td>
</tr>
<tr>
<td>active</td>
<td>Do we enable or disable the menu item?</td>
</tr>
<tr>
<td>file</td>
<td>A file containing menu specifications to read</td>
</tr>
<tr>
<td>package</td>
<td>Name of a package from where to load menu specifications</td>
</tr>
<tr>
<td>subdir</td>
<td>Subdirectory in the package where the menu specifications are stored. By default, it is the &quot;gui&quot; subdirectory.</td>
</tr>
</tbody>
</table>

Details

These functions care about creating, deleting and managing custom menus. Informations and handles to the various menus created with these functions are stored in the .gui.Menus variable, located in the TempEnv environment.

Value

MenuAdd() return the handle to the newly created menu. MenuNames() return the list of all menus registered in .gui.Menus.

Author(s)

Philippe Grosjean

See Also

MenuNames, winMenuChangeItem
Examples

## Not run:
## These cannot be run by examples() but should be OK when pasted
## into an interactive R session with the tcltk package loaded
## Run these commands one at a time and look at menus...

# Create menus in Rgui, using a specification file
MenuReadPackage("svWidgets")
MenuNames()
(MenuItems("$ConsoleMain/Testit"))

# Create menus manually in Rgui
MenuAdd("$ConsoleMain/Testit2")
MenuAddItem("$ConsoleMain/Testit2", "Trial", "ls()")
MenuNames()
(MenuItems("$ConsoleMain/Testit2"))
MenuStateItem("$ConsoleMain/Testit2", "Trial", FALSE)
MenuStateItem("$ConsoleMain/Testit2", "Trial", TRUE)
# This is buggy! -> MenuChangeItem("$ConsoleMain/Testit2", "Trial", "search()")
(MenuItems("$ConsoleMain/Testit2"))

# Create and manipulate Tk menus
TkWinAdd("tt", title = "A Tk window with menus", pos = "-40+20")
MenuAdd("$Tk.tt/Misc")
MenuNames()
(MenuItems("$Tk.tt/Misc"))
Still nothing in it
MenuAddItem("$Tk.tt/Misc", "List &variables", "print(ls(envir = .GlobalEnv))")
MenuAddItem("$Tk.tt/Misc", "Say &yo!", "cat('yo!\n')")
MenuDelItem("$Tk.tt/Misc", "Say &yo!")
MenuAddItem("$Tk.tt/Misc", "Say yo! twice", "cat('yo! yo!\n')")
(MenuItems("$Tk.tt/Misc"))

TkImgReadPackage("svWidgets") # Make sure images are loaded
MenuAdd("$Tk.tt/Misc/Sub&Menu")
MenuAddItem("$Tk.tt/Misc/Sub&Menu", "T&rial", "cat('Trial triggered!\n')")
MenuAddItem("$Tk.tt/Misc", "Tria&l2", "cat('Trail with image and accel!\n')", image = "$Tk.butOpen", accel = "Ctrl+T")
MenuNames()
(MenuItems("$Tk.tt/Misc"))
MenuStateItem("$Tk.tt/Misc", "Tria&l2", FALSE)
MenuStateItem("$Tk.tt/Misc", "Tria&l2", TRUE)
MenuStateItem("$Tk.tt/Misc", "Sub&Menu", FALSE)
MenuStateItem("$Tk.tt/Misc", "Sub&Menu", TRUE)
MenuChangeItem("$Tk.tt/Misc", "Tria&l2", options = "underline = 1")
# This is the way to change binding
tkbind(WinGet("tt"), "<Control-r>", function() tkMenuInvoke("$Tk.tt/Misc", "Tria&l2")
MenuChangeItem("$Tk.tt/Misc", "Tria&l2", action = 'cat("new action for Tria&l2!\n")'
tkMenuInvoke("$Tk.tt/Misc", "Tria&l2")
(MenuItems("$Tk.tt/Misc"))
MenuDelItem("$Tk.tt/Misc", "Tria&l2")
MenuDel("$Tk.tt/Misc")
MenuNames()
(MenuItems("$Tk.tt/Misc"))
tkWinDelete("tt")
Tool

Conveniently manipulate tools and toolbars, whatever the widgets used

Description

These functions provide an unifying way of dealing with tools and toolbars in R. Currently, they support a little bit a functionality for tcltk2 widgets (tile) and most functions, except ToolRead are not finished.

Usage

ToolbarAdd(toolbar, ...)
ToolbarDel(toolbar)
ToolbarType(toolbar, warn = TRUE)
ToolAdd(tool, ...)
ToolDel(tool)
ToolNames()
ToolItems(toolbar)
ToolType(tool, warn = TRUE)
ToolChange(tool, action = "", image = "", accel = "", options = "")
ToolState(tool, active = TRUE)
ToolRead(file = "Tools.txt")
ToolReadPackage(package, subdir = "gui", file = "Tools.txt")

Arguments

toolbar Name of a toolbar
...
Further options
warn Do we issue a warning if the type of tool or toolbar is not recognized?
tool The name of a tool
action Action the tool triggers (R code)
image Name of an image to display at left of the menu item
accel Accelerator (keystroke) to use to trigger this menu item
options Additional options, for instance "state = "disable" to disable the tool at creation.
active Do we enable or disable the tool?
file A file containing menu specifications to read
package Name of a package from where to load tools and toolbars specifications
subdir Subdirectory in the package where the tools and toolbars specifications are stored. By default, it is the "gui" subdirectory.

Details

These functions care about creating, deleting and managing custom tools and toolbars. Informations and handles to the various tools created with these functions are stored in the .gui.Tools variable, located in the TempEnv environment.
Win

Value

ToolAdd() return the handle to the newly created tool. ToolNames() return the list of all tools registered in .gui.Tools.

Author(s)

Philippe Grosjean

See Also

MenuAdd

Examples

## Not run:
## These cannot be run by examples() but should be OK when pasted
## into an interactive R session with the tcltk or tcltk2 package loaded
## Run these commands one at a time and look at the toolbars being created...

### TODO: examples here!

## End(Not run)

---

Win

Manipulate Windows

Description

R can combine various windows (native, Tk, Gtk, etc.). There could be problems when a GUI uses various kinds of windows together. For instance, it is very difficult to define a modal window that is modal for the whole application. These functions manage windows and ease their clean creation an destruction.

Usage

```r
tkWinAdd(name = "tkwin1", parent = .TkRoot, title = NULL, pos = NULL,
         bind.delete = TRUE, ...)
tkWinDelete(name)
WinGet(name)
WinNames()
```

Arguments

- **name**: Name of a window
- **parent**: Parent of this window
- **title**: Title of the window
- **pos**: Where to place the window. A string like '+XX+YY' where XX is the horizontal position in pixels, and YY is the vertical position. Using negative values place the window relative to the right or bottom side of the screen. Specifying NULL (by default) allows for automatic placement of the window.
- **bind.delete**: Do we automatically bind tkWinDelete to the windows delete envent?
- **...**: Additional options to pass to the window creator
tkMenu

Details

The list of windows and pointers to their handles are stored in `.gui.Wins` in the `TempEnv` environment.

Value

tkWinAdd() and WinGet() return the handle to the newly created window. WinNames() return the list of all windows registered in `.gui.Wins`.

Author(s)

Philippe Grosjean

See Also

MenuReadPackage, tkImgReadPackage

Examples

```r
## Not run:
## These cannot be run by examples() but should be OK when pasted
## into an interactive R session with the tcltk package loaded
#
# Creating and destroying a Tk window and inspecting the list
WinNames()
tkWinAdd("tt", title = "My win", pos ="-40+20")
WinNames()
tkm.deiconify(WinGet("tt")) # Standard tcltk functions on the window
tkWinDelete("tt")
WinNames()
## End(Not run)
```

Description

These functions provide an easier way to create and manipulate Tk menus under R. Note that the corresponding MenuXXX() function also manipulate Tk menus the same way, but are capable of manipulating other menus as well. One should, thus, preferably use MenuXXX() that tkMenuXXX()!

Usage

tkMenuAdd(menu, tearoff = FALSE)
tkMenuAddItem(menu, item, action, image = "", accel = "", options = "")
tkMenuDel(menu)
tkMenuDelItem(menu, item)
tkMenuItems(menu)
tkMenuChangeItem(menu, item, action = "", options = "")
tkMenuStateItem(menu, item, active = TRUE)
tkMenuInvoke(menu, item)
Arguments

- menu: Name of a menu
- tearoff: Should the menu be detachable?
- item: Name of a menu item
- action: Action the menu triggers (R code)
- image: Name of an image to display at left of the menu item
- accel: Accelerator (keystroke) to use to trigger this menu item
- options: Additional options, for instance 'state = "disable"' to disable the menu at creation.
- active: Do we enable or disable the menu item?

Value

tkMenuAdd() return the handle to the newly created menu.

Author(s)

Philippe Grosjean

See Also

MenuNames

Examples

```r
## Look at the extensive example in ?MenuNames
```

Description

A couple of winMenuXXX() functions are defined in the package 'utils' to manipulate custom menus of Rgui (under windows only). Here are some additional ones. note that you should preferably use the corresponding MenuXXX() function defined in this package (they work with windows menus as well as other menus, like Tk.

Usage

```r
winMenuChangeItem(menu, item, action, options = "")
winMenuStateItem(menu, item, active = TRUE)
```

Arguments

- menu: Name of a menu
- item: Name of a menu item
- action: Action the menu triggers (R code)
- options: Additional options, for instance 'state = "disable"' to disable the menu at creation.
- active: Do we enable or disable the menu item?
Details

These functions complement the winMenuXXX() functions in package 'utils'. Do prefer to use the corresponding MenuXXX() functions that work with all types of menus (currently, only Windows and Tk, but more will be added in the future).

Author(s)

Philippe Grosjean

See Also

MenuChangeItem, MenuStateItem

Examples

## See ?MenuNames for extensive examples
Index

*Topic IO
  clippaste, 17
  copy, 18
  export, 19
  getSocketClients, 49
  getSocketServerName, 50
  getSocketServers, 50
  processSocket, 51
  sendSocket, 52
  startSocketServer, 52

*Topic environment
  getEnvironment, 5

*Topic file
  clippaste, 17
  copy, 18
  export, 19

*Topic interface
  guiViews, 21
  report, 22
  view, 23
  viewHTMLInit, 24
  wordReport, 25

*Topic internal
  svGUI-internal, 60

*Topic misc
  display, 29
  guiCmd, 7
  guiDlg, 30
  guiDlgAssistant, 31
  guiDlgColor, 31
  guiDlgDir, 32
  guiDlgDoubleList, 33
  guiDlgFont, 34
  guiDlgFormula, 34
  guiDlgFunction, 35
  guiDlgGraphOptions, 36
  guiDlgGrid, 36
  guiDlgInput, 37
  guiDlgItemSel, 38
  guiDlgList, 39
  guiDlgMessage, 40
  guiDlgOpen, 41
  guiDlgOptions, 42
  guiDlgProgress, 42
  guiDlgSave, 43
  guiDlgText, 44
  guiDlgVarSel, 45
  guiDlgView, 46
  guiEval, 46
  guiInfo, 56
  guiObjBrowse, 57
  guiPane.tcltk, 47
  guiRemove, 58
  guiSave, 59
  guiSetStyle.tcltk, 48
  guiStart, 59
  Require, 1
  TinnR, 63
  Windows, 55

*Topic print
  guiViews, 21
  report, 22
  view, 23
  viewHTMLInit, 24
  wordReport, 25

*Topic utilities
  Args, 61
  assignTemp, 3
  compareRVersion, 3
  Complete, 62
  createSyntaxFile, 64
  existsTemp, 4
  findhtmlHelp, 5
  getEnvironment, 5
  getFunctions, 65
  getKeywords, 66
  getSocketClients, 49
  getSocketServerName, 50
  getSocketServers, 50
  getTemp, 6
  guiDDEInstall, 67
  guiSource, 7
  help.search.web, 8
  Img, 69
  isRgui, 9
  isSDI, 10

79
isWin, 10
listCustoms, 11
listMethods, 12
Menu, 70
progress, 12
Require, 1
rmTemp, 13
tempdirWin, 14
TempEnv, 2
tempvar, 15
tkMenu, 75
Tool, 73
userdir, 15
Win, 74
winMenu, 76
.onLoad(svGUI-internal), 60
.onUnload(svGUI-internal), 60
.packageName(svGUI-internal), 60

Args, 61
assignTemp, 2, 3, 4, 6, 14
CallTip, 65, 67
CallTip(Args), 61
clippaste, 17, 18
compareRVersion, 3
Complete, 62, 67
copy, 17, 18, 19
createCallTipFile, 62
createCallTipFile
(createSyntaxFile), 64
createSyntaxFile, 64, 66
detach, 2
display, 29, 30, 47
evaluate, 2, 3, 4, 6, 14
export, 18, 19
findhtmlhelp, 5
getEnvironment, 5
getFunctions, 65, 65, 66
getKeywords, 65, 66, 66
getSocketClients, 49, 51
getSocketServerName, 50, 51
getSocketServers, 49, 50, 50
getTemp, 2–4, 6, 14
guiCallTip, 62
guiCallTip(guiDDEInstall), 67
guiCmd, 7, 8
guiComplete, 62
guiComplete(guiDDEInstall), 67
guiDDEInstall, 64, 67
guiDlg, 29, 30, 47, 48
guiDlgAssistant, 31
guiDlgColor, 31
guiDlgDir, 32, 41
guiDlgDoubleList, 33, 39
guiDlgFont, 32, 34
guiDlgFormula, 34
guiDlgFunction, 31, 35, 35
guiDlgGraphOptions, 32, 36, 42
guiDlgGrid, 36, 44
guiDlgInput, 35, 37, 40
guiDlgItemSel, 38, 45
guiDlgList, 33, 39
guiDlgMessage, 38, 40, 43
guiDlgOpen, 32, 41, 44
guiDlgOptions, 34, 36, 42
guiDlgProgress, 42
guiDlgSave, 32, 41, 43
guiDlgText, 37, 38, 44, 46
guiDlgVarSel, 35, 39, 45
guiDlgView, 46
guiEval, 29, 30, 46
guiInfo, 56
guiInstall, 7
guiInstall(guiStart), 59
guiObjBrowse, 57, 58, 59
guiObjClear (guiObjBrowse), 57
guiObjDir (guiObjBrowse), 57
guiObjList, 56
guiObjList (guiObjBrowse), 57
guiObjMenu (guiObjBrowse), 57
guiObjSearch (guiObjBrowse), 57
guiPane.entry.tcltk
(guiPane.tcltk), 47
guiPane.list.tcltk
(guiPane.tcltk), 47
guiPane.tcltk, 30, 47
guiRemove, 58, 58
guiSave, 58, 59
guiSetStyle.tcltk, 48
guiSource, 7
guiStart, 7, 56, 59
guiStop (guiStart), 59
guiUninstall (guiStart), 59
guiViews, 21
guiViewsCmd (guiViews), 21
guiViewsCSS (guiViews), 21
guiViewsCSSChange (guiViews), 21
guiViewsDir (guiViews), 21
guiViewsDisplay (guiViews), 21
guiViewsFile (guiViews), 21
help.search, 9
help.search.web, 8
Img, 69
ImgGet (Img), 69
ImgNames (Img), 69
ImgType (Img), 69
isRgui, 9
isSDI, 4, 9, 10, 11
isWin, 4, 10, 10
listCustoms, 11, 12
listMethods, 5, 11, 12
Menu, 70
MenuAdd, 74
MenuAddItem (Menu), 70
MenuChangeItem, 77
MenuChangeItem (Menu), 70
MenuDel (Menu), 70
MenuDelItem (Menu), 70
MenuItems (Menu), 70
MenuNames, 71, 76
MenuNames (Menu), 70
MenuRead (Menu), 70
MenuReadPackage, 75
MenuReadPackage (Menu), 70
MenuStateItem, 77
MenuStateItem (Menu), 70
MenuType (Menu), 70
name, 1
options, 1
processSocket, 51, 52, 53
progress, 12
report, 22, 24, 26
reportGraph (report), 22
Require, 1
require, 2
rmTemp, 2–4, 6, 13
sendSocket, 51, 52
source, 8
startSocketServer, 51, 52
stopSocketServer
   (startSocketServer), 52
svGUI-internal, 60
tempdirWin, 14, 15, 16
TempEnv, 2, 3, 4, 6, 14
tempvar, 15
TinnR, 63
tkImgAdd (Img), 69
tkImgDel (Img), 69
tkImgReadPackage, 75
tkImgReadPackage (Img), 69
tkMenu, 75
tkMenuAdd (tkMenu), 75
tkMenuAddItem (tkMenu), 75
tkMenuChangeItem (tkMenu), 75
tkMenuDel (tkMenu), 75
tkMenuDelItem (tkMenu), 75
tkMenuInvoke (tkMenu), 75
tkMenuItems (tkMenu), 75
tkMenuStateItem (tkMenu), 75
tkWinAdd (Win), 74
tkWinDelete (Win), 74
Tool, 73
ToolAdd (Tool), 73
ToolbarAdd (Tool), 73
ToolbarDel (Tool), 73
ToolbarType (Tool), 73
ToolChange (Tool), 73
ToolDel (Tool), 73
ToolItems (Tool), 73
ToolNames (Tool), 73
ToolRead (Tool), 73
ToolReadPackage (Tool), 73
ToolState (Tool), 73
ToolType (Tool), 73
trObjList (TinnR), 63
trObjSearch (TinnR), 63
userdir, 14, 15
view, 22, 23, 23, 25
viewHTMLInit, 24, 24
Win, 74
Windows, 55, 60
WinGet (Win), 74
winMenu, 76
winMenuChangeItem, 71
winMenuChangeItem (winMenu), 76
winMenuStateItem (winMenu), 76
WinNames, 70
WinNames (Win), 74
WordActivate (wordReport), 25
WordExec (wordReport), 25
WordExit (wordReport), 25
WordGoto (wordReport), 25
WordGotoEnd (wordReport), 25
WordInsertFile (wordReport), 25
WordInsertPageBreak (wordReport), 25
WordInsertPara (wordReport), 25
WordInsertPictureFile (wordReport), 25
WordInsertText (wordReport), 25
WordOpen (wordReport), 25
wordReport, 25