Package 'svKomodo'

May 9, 2026

Type Package

Version 1.0.0

Date 2022-05-07

Title 'SciViews' - Functions to Interface with Komodo IDE

Description R-side code to implement an R editor and IDE in Komodo IDE with the SciViews-K extension.

Maintainer Philippe Grosjean <phgrosjean@sciviews.org>

Depends R (>= 2.6.0), svMisc (>= 0.9-68)

Imports utils

Suggests svHttp, svSocket, spelling, covr, knitr, rmarkdown

SystemRequirements Komodo Edit (https://www.activestate.com/products/komodo-ide/)

License GPL-2

URL https://github.com/SciViews/svKomodo, https://www.sciviews.org/svKomodo/

BugReports https://github.com/SciViews/svKomodo/issues

RoxygenNote 7.1.1

Encoding UTF-8

Language en-US

NeedsCompilation no

Author Philippe Grosjean [aut, cre] (ORCID: <https://orcid.org/0000-0002-2694-9471>)

Repository CRAN

Date/Publication 2022-05-10 15:40:02 UTC

Contents

`svKomodo-package` *svKomodo: 'SciViews' - Functions to Interface with Komodo IDE*

Description

R-side code to implement an R editor and IDE in Komodo IDE with the SciViews-K extension.

Details

Before RStudio or R addins for vscode were developed and popularized, this was an attempt to develop a complete IDE for R by using Komodo IDE, see https://www.activestate.com/products/komodo-ide/. The additional code needed to supplement Komodo IDE with R-related features was implemented in two parts: (1) The SciViews-K (https://github.com/SciViews/sciviewsk) addin for Komodo and (2) the present svKomodo R package that work hand in hand to provide R code intelligence, an object explorer, an integrated R Console, a document format similar to R Markdown, but using ASCIIDoc instead of Markdown, and more.

The development of this extension for Komodo IDE was stopped in 2016 because it was too similar to RStudio that emerged at that time as one of the preferred editor/IDE for R. The SciViews IDE using Komodo never was enough advertised, nor documented to gain a significant useR base.

Nevertheless, the SciViews-K Komodo addin and the svKomodo R package implemented features that may be interesting in a different context. The interaction of R and Komodo and the remote R Console in Komodo use communication protocols developed in the svSocket and svHttp R packages. As such, it is an example of R -JavaScript code to interact with R in a non-blocking way that could be reused in any other software that implements its UI using HTML + JavaScript. The consequent JavaScript code developed for the R object explorer may be worth looking at, for instance.

Both SciViews-K and svKomodo should be considered archived items. I keep them "alive" in the hope that their code could be useful to others in similar contexts. See the other man pages for further explanations.

Author(s)

Maintainer: Philippe Grosjean <phgrosjean@sciviews.org> (ORCID)

See Also

Useful links:

- https://github.com/SciViews/svKomodo
- https://www.sciviews.org/svKomodo/
- Report bugs at https://github.com/SciViews/svKomodo/issues

`koCmd` *Connect to the SciViews-K (Komodo Edit/IDE) socket server and run JavaScript code in Komodo*

Description

If Komodo Edit/IDE with the SciViews-K extension is running on your machine, you can connect to its socket server and run javascript code in it with this function.

Usage

```
koCmd(
  cmd,
  data = NULL,
  async = FALSE,
  host = getOption("ko.host"),
  port = getOption("ko.port"),
  kotype = getOption("ko.kotype"),
  timeout = 2,
  type = c("js", "rjsonp", "output"),
  pad = NULL,
  ...
)
```

Arguments

`cmd`	the JavaScript you want to execute in Komodo Edit, in a character
`data`	if a named list, replace `<<<name>>>` in cmd for each name of the list by its component first. If a character string, replace `<<<data>>>` in cmd. If `NULL` (by default), do nothing to cmd before submitting it. See the last examples for using data.
`async`	not used yet!
`host`	the host where Komodo is located. Currently, only `localhost` is accepted. Can be changed by setting `options(ko.host = ....)`.
`port`	the socket port where the SciViews-K server is listening, by default, it is port 7052. Can be changed by setting `options(ko.port = ....)`.
`kotype`	the type of Komodo server in use. Currently (SciViews-K >= 0.9-25), it can be either `"socket"` or `"file"`.

timeout	number of seconds to wait for a response.
type	which type of Komodo command do we send? If type = "js" (by default), cmd is considered to be JavaScript code to be evaluated in Komodo. If type = "rjsonp", cmd is parsed as RJson object with padding (included in a JavaScript function call) and will be evaluated as such in Komodo. if type = "output", the string in cmd is considered to be some R output and will be displayed in the Komodo local R console (no evaluation).
pad	a string naming the JavaScript function to run in Komodo, with the constructed RJson object as argument. If NULL (by default), the RJson object is evaluated directly without padding.
...	further arguments to pass to toRjson().

Details

Komodo Edit (https://www.activestate.com/products/komodo-ide/) is an Open Source (MPL, GPL & LGPL) editor based on the excellent Mozilla platform and the powerful Scintilla text editor widget. It runs on many Linux distributions, on Windows and on MacOS. Komodo IDE was a commercial equivalent, but with many tools for developers, especially targeting languages like Perl, Tcl, Python, Ruby, etc. This product is now freely distributed as well and Komodo Edit was deprecated. However, the project does not appear to be actively maintained any more and it may not work on more recent MacOS or Windows versions.

koCmd() can only talk to Komodo if the SciViews-K socket server is installed. This server is contained in the SciViews-K extension that you can download from https://github.com/SciViews/sciviewsk. See Komodo documentation to know how to install this extension (drag and drop of the extension on the Komodo window works in most platforms).

Value

Returns the results of the evaluation of the javascript code in Komodo Edit/IDE if async = FALSE. Note that async = TRUE is not supported yet.

If there is an error, or cmd is an invalid JavaScript code, a character string containing javascript error message is returned (this is changed from version 0.9-47, previously a 'try-error' was returned).

Note

Because of security concerns, the SciViews-K server only allows connections from local clients (running on the same computer). This limitation would be relatively easy to eliminate, but at your own risks!

Data are returned from Komodo to R by using the JavaScript function sv.socket.serverWrite(), see the examples bellow.

See Also

svSocket::startSocketServer(), svSocket::processSocket()

Examples

```
## Not run:
# Make sure you have started Komodo Edit or IDE with the SciViews-K extension
# installed on the same machine, and the socket server started and then...

# Send JavaScript commands to Komodo
# Alert box in Komodo, and then reply to R
koCmd(c('alert("Hello from R!");',
  'sv.socket.serverWrite("Hello from OpenKomodo (" + ko.interpolate.currentFilePath() + ")");'))

# Open a web page wih Komodo configuration
koCmd("ko.open.URI('about:config','browser');")

# Get info from Komodo
koCmd("sv.socket.serverWrite(ko.logging.getStack());")

# Passing a large amount of data to Komodo, and then, back to R
koCmd(paste0('sv.socket.serverWrite("', rep(paste(iris, collapse = "\\\\n"), 10), '");'))

# It is easier to use 'data =' instead of paste() for constructing the JS command
koCmd('alert("<<<data>>>");', data = search())

# Using a named list for data to replace in the cmd
koCmd('alert("This is R version <<<major>>>.<<<minor>>>");', R.version)

# Sending incorrect JavaScript instruction
koCmd('nonexistingJSfunction();')
# Should return something like:
# "ReferenceError: nonexistingJSfunction is not defined"

# Sending RJsonP (RJson with padding) instruction to Komodo
koCmd("Hello with RJsonP!", type = "rjsonp", pad = "alert")

# This is more useful to pass complex R objects to Komodo
koCmd(head(iris), type = "rjsonp", pad = "sv.socket.serverWrite")

# Send simple text (no evaluation) to the Komodo R console
koCmd("Hello again from R!", type = "output")

## End(Not run)
```

`koInstall` *Install and uninstall hooks for communicating with Komodo Edit/IDE*

Description

Install functions in `SciViews:TempEnv` and callbacks required to communicate with Komodo Edit with the SciViews-K extension (see https://github.com/SciViews/sciviewsk).

Usage

```
koInstall()

koUninstall()
```

Details

The minimum instruction to install the communication with Komodo/SciViews-K (so called, SciViews Komodo) is to use: `options(ko.serve = 8888); require(svKomodo)`. When the `ko.serve` option is set, svKomodo loads svSocket, starts the socket server listening to the port you have selected (8888 by default), and install the hooks and callbacks required to communicate with SciViews Komodo.

Before loading svKomodo, you can also set `option(ko.port = 7052)` or another port number where the Komodo SciViews-K server is listening (7052 is the default value). If the Komodo client is running on a different machine, you should also set `ko.host = "xxx.xxx.xxx.xxx"`, where `xxx.xxx.xxx.xxx` is the IP address of the Komodo client's machine, before loading svKomodo (note that running R and Komodo on separate machines is not supported yet, but this is a planned feature and corresponding configurations are already recognized; just, distant server is currently locked until we will build a better security mechanism in the server (SSL, TSL, ...).

All these operations are done by Komodo if you start R from Komodo with SciViews-K extension installed.

Value

Returns nothing.

See Also

`koCmd()`

---

koRefresh *Refresh Komodo interface (active object lists and R object browser)*

Description

These function manage to refresh the list of active objects in Komodo and also the data displayed in the Komodo R object browser. They should not be called directly by the end-user.

Usage

```
koRefresh(force = FALSE)

koAutoRefresh(...)
```

Arguments

force	do we force refresh, even if data have not changed?
...	any argument (ignored, but useful for addTaskCallback()).

Value

Both functions return TRUE on success.

See Also

koCmd()

---

svTaskCallbackManager *Create task callbacks that are evaluated both from R and socket/http server*

Description

svTaskCallbackManager() is a copy of taskCallbackManager() in R base package, as of version 4.0.5 of R. Two important differences: (1) the top task created is named SV-taskCallbackManager instead of R-taskCallbackManager, and its tasks are executed after each top-level task in R console, or after execution of non-hidden R code from the socket or http server (take care: only once per set of code, no matter the number of top-level task in the R code send by the client in the second case). All taskCallbacks defined by addTaskCallback() or taskCallbackManager$add() from R base package are not executed when code is invoked from the R socket or http server!

Usage

```
svTaskCallbackManager(handlers = list(), registered = FALSE, verbose = FALSE)
```

Arguments

handlers	this can be a list of callbacks in which each element is a list with an element named f which is a callback function, and an optional element named data which is the 5-th argument to be supplied to the callback when it is invoked. Typically this argument is not specified, and one uses add to register callbacks after the manager is created.
registered	a logical value indicating whether the evaluate function has already been registered with the internal task callback mechanism. This is usually FALSE and the first time a callback is added via the add function, the evaluate function is automatically registered. One can control when the function is registered by specifying TRUE for this argument and calling addTaskCallback() manually.
verbose	a logical value, which if TRUE, causes information to be printed to the console about certain activities this dispatch manager performs. This is useful for debugging callbacks and the handler itself.

Value

See `?taskCallbackManager` for both the returned object and how to use it.

Author(s)

Slightly modified from the original R core team's function by Ph. Grosjean phgrosjean@sciviews.org

See Also

`taskCallbackManager()`

Examples

```
# create a task callback manager
cbman <- svTaskCallbackManager()
# Add a function to activate after each code evaluation in R
cbman$add(function(expr, value, ok, visible) {
  cat("Hi from the callback manager!\n")
  return(TRUE)
  }, name = "exampleHandler")
# Just issue a command and see the callback function activated
1 + 1
# List defined callbacks
cbman$callbacks()
# Remove the callback we just defined
cbman$remove("exampleHandler")
1 + 1
# Remove the task callback manager (base R function)
removeTaskCallback("SV-taskCallbackManager")
```

Index