Package 'rJython'

February 20, 2015

Version 0.0-4

Date 2011-12-29

Title R interface to Python via Jython

Author G. Grothendieck and Carlos J. Gil Bellosta (authors of Jython itself are Jim Hugunin, Barry Warsaw, Samuele Pedroni, Brian Zimmer, Frank Wierzbicki and others; Bob Ippolito is the author of the simplejson Python module)

Maintainer Carlos J. Gil Bellosta <cgb@datanalytics.com>

Description R interface to Python via Jython allowing R to call python code.

SystemRequirements Java

Depends rJava (>= 0.8-1), rjson (>= 0.1.9)

License GPL (>= 2)

URL https://r-forge.r-project.org/projects/rjython/

Repository CRAN

Date/Publication 2012-07-30 18:38:02

NeedsCompilation no

R topics documented:

jython.assign *Assign and get variables in Python from R*

Description

Functions that assign and get Python variables from R.

Usage

```
jython.assign( rJython, var.name, value )
jython.get( rJython, py.var )
```

Arguments

rJython	rJython interpreter as instantiated by function rJython
var.name	a character string containing a valid python variable name
value	an R object whose equivalent wants to be assigned to the variable in python
py.var	Name of a python variable

Details

These functions can assign values to variables in Python as well as get their values back to R. Objects are serialized as json strings while being transferred between R and Python.

Value

Function jython.get returns a R version of the Python variable py.var.

References

http://code.google.com/p/simplejson

Examples

```
rJython <- rJython()

a <- 1:4
jython.assign( rJython, "a", a )
jython.exec( rJython, "b = len( a )" )
jython.get( rJython, "b" )

rJython$exec( "import math" )
jython.get( rJython, "math.pi" )
```

jython.call *jython.call*

Description

Calls Python functions from R

Usage

```
jython.call( rJython, py.foo, ... )
```

Arguments

rJython	rJython interpreter as instantiated by function rJython
py.foo	Name of a python function
...	R objects to pass as arguments to the Python function

Details

This function runs a python function taking as arguments R objects and returning an R object. Some limitations exist as to the nature of the objects that can be passed between R and Jython. As of this writing, atomic arguments and vectors are supported.

The user has to be careful to indicate named parameters as required according to python conventions.

Value

An R representation of the object returned by the call to the python function.

References

http://code.google.com/p/simplejson

Examples

```
rJython <- rJython()
jython.call( rJython, "len", 1:3 )

a <- 1:4
b <- 5:8
rJython$exec( "def concat(a,b): return a+b" )
jython.call( rJython, "concat", a, b)
```

jython.exec *jython.exec*

Description

Executes python code via the Jython interpreter

Usage

```
jython.exec( rJython, python.code )
```

Arguments

rJython	rJython interpreter as instantiated by function rJython
python.code	a character vector containing python code, typically containing a single line with indentation and EOL characters as required by Python syntax

Details

This function runs Python code. It needs to be provided by the caller in a character vector.

The vector may consists of a single string with EOL and indentation characters embedded.

Alternatively, it can be a character vector, each entry containing one or more lines of python code.

Value

None. If the code produces some output, it is up to the caller to go and fetch if from Jython.

References

http://code.google.com/p/simplejson

Examples

```
rJython <- rJython()

a <- 1:4
b <- 5:8
jython.exec( rJython, c( "def concat(a,b):", "\treturn a+b" ) )
jython.call( rJython, "concat", a, b)
```

jython.method.call *jython.method.call*

Description

Calls Python functions from R

Usage

```
jython.method.call( rJython, py.object, py.method, ... )
```

Arguments

rJython	rJython interpreter as instantiated by function rJython
py.object	an existing object
py.method	a method of such object
...	arguments for the method

Details

This function runs a python function taking as arguments R objects and returning an R object. Some limitations exist as to the nature of the objects that can be passed between R and Jython. As of this writing, atomic arguments and vectors are supported.

The user has to be careful to indicate named parameters as required according to python conventions.

Value

An R representation of the object returned by the call to the python function.

References

http://code.google.com/p/simplejson

Examples

```
rJython <- rJython()

rJython$exec( 'a = "hola hola"' )
jython.method.call( rJython, "a", "split", " " )
```

rJython *rJython*

Description

Interface for using python from R via Jython

Usage

```
rJython( jython.jar = NULL, modules = NULL )
```

Arguments

jython.jar	Optional alternative location of the jython.jar file. The package is distributed with its 'jython.jar' file, but the user can invoke their own by providing its path.
modules	List containing optional paths for required modules. Packages utilizing rJython can add their own extra python modules to the 'inst' directory and then add their base path using function system.file so that they can be later found when issuing 'import module' directives.

Details

This function is not primarily intended to be run directly by users but to be used by R packages with python dependencies. Their authors are invited to hide the internals of this package in their code at their convenience.

Value

An object of class rJava that can exectute python code as per the examples below.

References

http://rsympy.googlecode.com, http://www.jython.org/Project/, http://www.jython.org/javadoc/org/python/util/PythonInterpreter.html

Examples

```
rJython <- rJython()

# Now you can use the rJava low level interface

rJython$exec( "a = 2*2" )
four <- rJython$get("a")
.jstrVal( four )

# Alternatively, a higher level interface is provided by the rJython package:
```

```
a <- 1:4
jython.assign( rJython, "a", a )
jython.exec( rJython, "b = len( a )" )
jython.get( rJython, "b" )


## Not run:

# If package 'foo' contains python code that depends on module 'mod',
#   'mod' can be included in the 'inst' directory of the package.
# Then, in order to make it available to Jython, the interpreter can be
#   started as follows:

rJython <- rJython( modules = system.file("python-modules", package = "foo") )


## End(Not run)
```

Index