Package ‘Runiversal’

February 19, 2015

Type Package

Title Runiversal - Package for converting R objects to Java variables and XML.

Version 1.0.2

Date 2010-07-31

Author Mehmet Hakan Satman

Maintainer Mehmet Hakan Satman <mhsatman@istanbul.edu.tr>

Url http://www.mhsatman.com

Description This package contains some functions for converting R objects to Java style variables and XML. Generated Java code is interpretable by dynamic Java libraries such as Beanshell. Calling R externally and handling the Java or XML output is an other way to call R from other languages without native interfaces. For a Java implementation of this approach visit http://www.mhsatman.com/rcaller.php and http://stdioe.blogspot.com/search/label/rcaller

License GPL

LazyLoad yes

Repository CRAN

Date/Publication 2012-08-01 05:55:46

NeedsCompilation no

R topics documented:

Runiversal-package *Converts R objects to Java style interpretable codes and XML*

Description

This package contains two main functions called makejava and makexml. makejava function converts R objects to Java codes which is interpretable by Bean Shell or Dynamic Java. Other function, makexml, converts R objects to XML code for handling R object in other languages. Both of these functions can be used for calling R codes from other languages without using native codes.

Details

Package:	Runiversal
Type:	Package
Version:	1.0.1
Date:	2010-07-29
License:	GPL
LazyLoad:	yes

Author(s)

Mehmet Hakan Satman

Maintainer: Mehmet Hakan Satman <mhsatman@istanbul.edu.tr> http://www.mhsatman.com

References

See http://www.mhsatman.com/rcaller for calling R from Java without JNI. Bean Shell is a library for scripting Java language that downloadable for free in http://www.beanshell.org/

Examples

```
#Getting estimates from regression object as Java variables that be directly interpretable by Bean Shell.
x<-1:10
y<-rnorm(10)
ols<-lm(y~x)
betas<-ols$coefficients
cat(makejava(betas,"myBetas"))

#Getting summary report as Java variables
cat(makejava(summary(ols)))

#Getting regression results as xml
cat(makexml(ols))
```

```
#Saving xml to file
cat(makexml(ols), file="somefile.xml")
```

`cleanNames` *Variable name cleaner.*

Description

An utility function for deleting spacial chracters from variable names (Especially for Java).

Usage

```
cleanNames(names)
```

Arguments

`names`	Names to be cleared.

Details

Clears variable names defined in names.

Value

Returns variable names.

Note

This function is generally called by main routines of package. See 'see also' section.

Author(s)

Mehmet Hakan Satman

See Also

makexml makejava

Examples

```
varname<-"r.squared"
cleanNames(varname)
```

concat *concat*

Description

Constructs a string of Java array using given R object.

Usage

```
concat(to, ...)
```

Arguments

to	A string variable which string of Java array will be added to.
...	An R object, generally vectors or scalers.

Details

This function is a utility function for main functions 'makexml' and 'makejava'

Value

Returns a new string, which is sums of old string and R objecs.

Author(s)

Mehmet Hakan Satman

See Also

makexml makejava

Examples

```
a<-"a string"
x<-1:10
concat(a,x)
```

`makejava` *A wrapper function for converting R objects to Java style variables.*

Description

This function converts R objects to Java arrays. If R object is numeric than the Java object is double[]. Otherwise the Java object will be String[].

Usage

```
makejava(obj, name = "")
```

Arguments

`obj`	R object that to be converted to Java style variables.
`name`	New variable name for created Java style variable. If R object is only a vector, a name have to be given. If R object is a list, name is inoperative.

Details

This function returns interpretable code for most frequently used dynamic Java interpreters such as Bean Shell. So this function can be used for using R codes in Java without JNI. This is an inefficient way to use R in Java but implementation is quite easy and platform independent.

Value

Returns interpretable code for Bean Shell and Java of R objects.

Note

This function generates interpretable code for Java. For other languages you can use makexml to convert R objects to XML code. XML is readable for all of the programming languages.

Author(s)

Mehmet Hakan Satman

References

See an Java implementation using this functionality in http://www.mhsatman.com/rcaller.php

See Also

makexml

Examples

```
#Shows the linear regression results as Java style variables
y<-rnorm(10)
x<-1:10
ols<-lm(y~x)
cat(makejava(ols))

#Shows only the residuals as double[] Residuals=new double[]{.....};
cat(makejava(ols$residuals, "Residuals"))
```

`makevectorjava` *makevectorjava()*

Description

This is an utility function for makejava() and not generally be called by user.

Usage

```
makevectorjava(code, objt, name = "")
```

Arguments

`code`	A string which the generated code to be added to.
`objt`	An R object, to be converted to R code.
`name`	Variable name for generated Java object. Optional.

Author(s)

Mehmet Hakan Satman

Examples

```
x<-1:10
cat(makevectorjava("",x,"x"))
```

makevectorxml *makevectorxml()*

Description

This is an utility function for makexml() and not generally be called by user.

Usage

```
makevectorxml(code, objt, name = "")
```

Arguments

code	A string which the generated code to be added to.
objt	An R object, to be converted to xml code.
name	Variable name for generated xml entry. Optional.

Author(s)

Mehmet Hakan Satman

Examples

```
x<-1:10
cat(makevectorxml("",x,"x"))
```

makexml *makexml*

Description

This function converts an R object to xml code, so an output of an R script can be handled easly by any programming language. Every elements in xml is encapsulated in <root></root> tags. Any variable in list or variable itself is placed between <variable> and </variable> tags. Values of elemenets are placed between <value> and </value> tags. Xml structure is easy to parse and browse.

Usage

```
makexml(obj, name = "")
```

Arguments

obj	R object that to be converted to xml.
name	Optinal. Each element in xml code is listed as <variable name="" type="numeric | chracter">. Default name is name of R object, but a new name can be defined.

Details

A generated output of lm object is like this:

<?xml version="1.0"?> <root> <variable name="coefficients" type="numeric"> <value>0.662970909075238</value>
<value>-0.125473985248366</value> </variable> <variable name="residuals" type="numeric">
<value>-0.0856149907633715</value> <value>0.963352019137748</value> <value>-0.212667731012814</value>
<value>-0.561977205893819</value> <value>-0.133045701604119</value> <value>-0.485339628713294</value>
<value>0.107354078697277</value> <value>-0.624426907949804</value> <value>1.45763471890252</value>
<value>-0.425268650800328</value> </variable> <variable name="effects" type="numeric"> <value>0.0858115975474794
<value>-1.13967406760172</value> <value>-0.313474791725809</value> <value>-0.577239374729828</value>
<value>-0.0627629785631422</value> <value>-0.329512013795331</value> <value>0.348726585492226</value>
<value>-0.297509509277868</value> <value>1.87009700945145</value> <value>0.07273853162558</value>
</variable> <variable name="rank" type="numeric"> <value>2</value> </variable> <variable name="fitted_values"
type="numeric"> <value>0.537496923826872</value> <value>0.412022938578506</value> <value>0.28654895333014</
<value>0.161074968081774</value> <value>0.0356009828334077</value> <value>-0.0898730024149584</value>
<value>-0.215346987663324</value> <value>-0.340820972911691</value> <value>-0.466294958160057</value>
<value>-0.591768943408423</value> </variable> <variable name="assign" type="numeric"> <value>0</value>
<value>1</value> </variable> <variable name="df_residual" type="numeric"> <value>8</value>
</variable> <variable name="call" type="character"> <value>lm, y ~ x</value> </variable> <vari-
able name="terms" type="character"> <value>~, y, x</value> </variable> </root>

Value

Returns a well-formed xml file for given R object.

Note

This function can be used for handling R output from other languages such as C, C++, Java, Javascript and Pyhton. This method turns problem of "calling R from other languages" into "parsing xml".

Author(s)

Mehmet Hakan Satman

See Also

makejava

Examples

```
x<-1:10
y<-rnorm(10)
ols<-summary(lm(y~x))
cat(makexml(ols))
```

Index