SJava

October 25, 2011

`.JavaArrayConstructor`
Create and access elements of Java arrays from R.

Description

These functions allow one to create multi-dimensional Java arrays via R commands using the `.Java` function. The get and set accessors work element-wise and not in the vector fashion common in R and S. One must create and initialize the Java virtual machine before calling any of these functions. See `.JavaInit`.

Usage

```
.JavaArrayConstructor(klass, ..., dim=length(list(...)), .name=NULL,
                      .convert=FALSE)
.JavaGetArrayElement(jobj,..., .name=NULL, .convert=TRUE)
.JavaSetArrayElement(jobj, value, ..., .dims=list(...), .convert=FALSE)
.JavaArrayLength(jobj)
## S3 method for class 'JavaArrayReference'
x[...]
## S3 replacement method for class 'JavaArrayReference'
x[...] <- value
## S3 method for class 'JavaArrayReference'
length(x)
```

Arguments

`klass`	Typically a string (character vector of length 1) identifying the name of the class of the element type in the array to be created. This can also be a foreign reference to a Java class object obtained via an earlier call to `.Java`.
`...`	In the `.JavaArrayConstructor`, these are currently ignored. They are intended to be initializing values that are used to populate the top-level values of the new array. That is, they are used to set `arr[0], arr[1], arr[2], ...`. In the `JavaArrayReference` methods these are indicies passed to `.JavaGetArrayElement` and `.JavaSetArrayElement`.

`dim`	When creating an array in `.JavaArrayConstructor`, these specify both the number of dimensions and the length of each dimension in the array to be created.
`.dims`	When setting an array element in `.JavaSetArrayElement`, a `list` of `integer(1)` values corresponding to array dimensions of the element to be set.
`jobj, x`	This is the reference to the Java array returned from an earlier call to `.JavaArrayConstructor` or the return value from a call to `.Java`.
`value`	the object to be inserted as an element in the Java array. This is converted to a Java object using the usual conversion rules and then inserted into the Java array.
`.name`	The name to use to store the result in the omegahat named reference database. If this is missing, an anonymous reference is returned or the value converted to an R object. If the result of the Java method can be converted, this argument can be used to prohibit this conversion and leave the Java value in Omegahat for use in future `.Java` calls.
`.convert`	a logical value that indicates whether the Omegahat manager should attempt to convert the result of the method call. This is usually `TRUE`, but can be explicitly specified to avoid (arrays of) primitive object being converted to an R object when it is to be used in a subsequent `.Java` call.

Details

This uses the `.Java` to call methods in the Omegahat Evaluator which process the array request.

Value

`.JavaArrayConstructor` returns a reference to the newly create Java array object.

`.JavaArrayLength` returns a single integer giving the length of the top-level dimension of the array.

`.JavaGetArrayElement` returns the value of the specified element of the given array, converted to an R object as usual. Thus it may be a Java reference.

`.JavaSetArrayElement` returns `NULL`.

Author(s)

Duncan Temple Lang, John Chambers

References

http://www.javasoft.com, http://www.omegahat.org

See Also

`.Java`

Examples

```
if (!isJavaInitialized())
    .JavaInit(verbose=TRUE)
a <- .JavaArrayConstructor("java.lang.String", dim=3)
.JavaArrayLength(a)
.JavaSetArrayElement(a, "First", 1)
```

```
.JavaSetArrayElement(a, "Second", 2)
.JavaSetArrayElement(a, "Third", 3)
.JavaGetArrayElement(a, 2)
```

`OmegahatReference` *Accessing Java classes, methods and field*

Description

The `$` methods allow one to invoke Java methods in the form

```
jobj$methodName(arg1,arg2,....)
```

The `[[` methods allow access to class fields as

```
jobj[["fieldName"]]
```

```
jobj[["fieldName"]] <- value
```

Usage

```
## S3 method for class 'OmegahatReference'
obj$name
## S3 method for class 'OmegahatReference'
x[[name, ...]]
## S3 replacement method for class 'OmegahatReference'
x[[name, ...]] <- value
## S3 method for class 'OmegahatReference'
print(x, ...)
```

Arguments

`obj`	A reference to the Java object.
`x`	A reference to the Java object.
`name`	The name of the Java method or field being accessed.
`value`	The value to be assigned to the field.
`...`	Additional arguments; ignored for `print`.

Details

The `$` method is equivalent to `.Java(obj, name,)`

The `[[` method is equivalent to `.Java(NULL, "getField", name, x)`.

The `[[<-` method is equivalent to `.Java(NULL, "setField", name, x, value)`.

Value

The return value of `$` is a function which remembers the arguments to this function call and, when called, results in a call to `.Java` using those arguments.

The return value of `[[` is the field value.

`print` is called for its side effect, i.e., providing a compact represntation of the object.

Author(s)

Duncan Temple Lang

References

http://www.omegahat.org/RSJava

See Also

.Java

Examples

```
p <- .Java("System", "getProperties", .convert=FALSE)
p$getProperty("java.class.path")
```

.JavaSigs *A vector of Java type specifiers*

Description

This is a named vector containing the pairs of Java primitive type names (e.g. double, boolean, etc.) and the corresponding low-level JNI type identifier. These are things such as The values ca be used in in the .sigs arguments of .Java and .JavaConstructor.

.javaConfig *The default options for initializing the Java Virtual Machine*

Description

This is a list containing elements for

classpath elements in which collections of Java classes are found

system properties name-value pairs used to customize the JVM's environment, the Omegahat classes and any other classes that read these properties

library path directories in which Java can find shared libraries.

The values in this object are created during the configuration of the package so that they refer to files in the directories into which the package is installed.

`.JClass` *Returns a reference to a Java class.*

Description

This returns the class of the specified object or that found by resolving the (partially qualfied) class name. The resulting class reference can be used to access static fields and method, and a description of the class. This is most conveniently done using the `$` operator.

Usage

```
.JClass(x, name = NULL)
```

Arguments

`x`	a (partially qualified) class name or a reference to a Java object whose class name is used to resolve the class.
`name`	an optional string to use as the name to use for the resulting class reference in the Omegahat database. This is passed as the `.name` argument to `.Java`.

Details

This calls the Omegahat evaluator's `findClass` method with the name of the class to be found.

Value

An anonymous reference to a Java class. This can then be used to access static methods and fields.

Author(s)

Duncan Temple Lang, John Chambers

References

http://www.omegahat.org/RSJava

See Also

`.Java`

Examples

```
jsys <- .JClass("System")
jsys$getProperties()
jsys$getProperty("java.class.path")

 rt <- .JClass("Runtime")$getRuntime()
 rt$exec("whoami")
 rt$exec(paste("find",system.file("scripts", pkg="Java")))

 frame <- .JNew("JFrame")
 frame$setBackground(.JClass("Color")$red())
```

`.Java` *Invokes a Java method*

Description

Calls a Java method, transferring R arguments to the Java (Omegahat) system as needed. This can be used to call methods in the Omegahat evaluator, not just on previously created user-level objects. One must create and initialize the Java virtual machine before calling this function. See `.JavaInit`.

Usage

```
.Java(.qualifier, .methodName, ..., .name=NULL, .sigs="", .convert=TRUE)
```

Arguments

`.qualifier`	The Java object whose method is to be invoked. This is the 'this' in the Java call and is typically a reference obtained as the result of a previous call to `.Java` or `.JavaConstructor`. If this is NULL or omitted, the Omegahat evaluator looks first for an Omegahat function and then a method within its own object.
`.methodName`	The name of the method (or function if `.qualifier` is `NULL`) that is to be invoked in the Java object.
`...`	arguments to be passed to the Java method call. Any values that are named (i.e. x = 1) are assigned persistently to the Omegahat *named reference database* and can be referenced directly in future calls.
`.name`	The name to use to store the result in the omegahat named reference database. If this is missing, an anonymous reference is returned or the value converted to an R object. If the result of the Java method can be converted, this argument can be used to prohibit this conversion and leave the Java value in Omegahat for use in future `.Java` calls.
`.sigs`	A character vector of class identifiers that help to identify the Java method to be invoked. This is used to avoid ambiguity introduced by Java's polymorphism/overloaded names and the automatic/implicit conversion performed between R and Java objects. This should have an entry for each argument passed via ... and governs how we convert that S value to a Java value. See `.JavaSigs` for possible values.
`.convert`	typically a logical value that indicates whether the Omegahat manager should attempt to convert the result of the method call. This is usually `TRUE`, but can be explicitly specified to avoid (arrays of) primitive object being converted to an R object when it is to be used in a subsequent `.Java` call. One can also provide a function which will be called with two arguments - a reference to the Java object and the class name of the Java object. This is the same as the function converters one can register via `setJavaFunctionConverter`. Also, one can specify a native routine (i.e. C/C++/Fortran) address. This can be done using `getNativeSymbolInfo` and accessing the `address` field of the returned object. See examples in the `inst/examples/` directory.

Details

This invokes a Java method on the target object by first converting the R arguments to Java objects and then searching the Java object for a method that accepts these Java argument types. Then it invokes the method and converts the result to a Java object using the basic and extensible conversion mechanism between Java and R.

Value

The return value of the Java method invocation, converted to an R object. If the Java value is considered convertible, one of the registered converters is called. By default, these handle primitives (scalars) and Java collections. The user can register others. If no converter is found, a reference to the Java object is returned as an R object. If the `.name` argument was supplied in the call to this R function, the reference is a `NamedReference`. Otherwise, it is a `AnonymousReference`.

Note

Uses the Omegahat interactive Java environment.

Author(s)

Duncan Temple Lang, John Chambers

References

http://www.omegahat.org/RSJava

See Also

`.JavaConstructor` `.OmegahatExpression` `getJavaConverterDescriptions`

`foreignReference`

Examples

```
 v <- .JavaConstructor("java.util.Vector", as.integer(10))
.Java(v, "add", "A string element")
.Java(v, "add", .JavaConstructor("java.util.Hashtable", as.integer(3)))
.Java(v, "size")

props <- .Java("System", "getProperties")
props[["java.class.path"]]

 props <- .Java("System", "getProperties", .convert=FALSE)
 props$getProperty("java.class.path")
```

JavaConfiguration-class *Class "JavaConfiguration"*

Description

Establish configuration parameters for SJava sessions

Objects from the Class

Use the constructor `JavaConfiguration` to create objects from the class.

```
JavaConfiguration(classPath= defaultJavaPath(), libraryPath = defaultJavaPath("L
properties=character(0), options=character(0), args=character(0))
```

Slots

`classPath`: `character` vector of class paths, one class path per element.

`libraryPath`: `character` vector of library paths, one path per element.

`properties`: A `character` vector of Java VM properties.

`options`: A `character` vector of Java VM options.

`args`: A `character` vector of Java VM arguments.

Methods

Manipulation.

collapse `signature(x = "JavaConfiguration")`: re-structure `x` into the equivalent command-line representation.

merge `signature(x = "JavaConfiguration", y = "JavaConfiguration")`: combine the contents of two `JavaConfiguration` objects, e.g., collating unique class paths.

Examples

```
showClass("JavaConfiguration")
JavaConfiguration()
collapse(JavaConfiguration())
```

.JavaConstructor *Create a Java object*

Description

Creates a Java object by calling a constructor from the desired class. The object is (almost always) stored in the Omegahat session and a reference to it returned. One must create and initialize the Java virtual machine before calling this function. See `.JavaInit`. `.JNew` is a simple alias of `.JavaConstructor`.

Usage

```
.JavaConstructor(className, ..., .name="", .sigs="", .convert=FALSE)
.JNew(className, ..., .name="", .sigs="", .convert=FALSE)
```

Arguments

`className`	The name of the Java class to be instantiated. This can be either the full name or a partially qualified name which will use the Omegahat class locator mechanism to find the class. It is better (but less convenient) to give the full name as this avoids the lengthy one-time construction of the class lists in Omegahat. It makes sense to give partially qualified names for a) the user's convenience, b) when one expects to substitute different packages with same-named classes that can be used in place of each other.
`...`	the arguments used to identify and be passed to the constructor in the target class being instantiated.
`.name`	The name to use to store the result in the omegahat named reference database. If this is missing, an anonymous reference is returned or the value converted to an R object. If the result of the Java method can be converted, this argument can be used to prohibit this conversion and leave the Java value in Omegahat for use in future `.Java` calls.
`.sigs`	A character vector of class identifiers that help to identify the Java method to be invoked. This is used to avoid ambiguity introduced by Java's polymorphism/overloaded names and the automatic/implicit conversion performed between R and Java objects.
`.convert`	a logical value indicating whether the Omegahat interpreter should attempt to convert the newly created object to an R object (TRUE) or simply leave it in the Omegahat database. This is ignored if a value for `.name` is supplied. One can also provide a function which will be called with two arguments - a reference to the Java object and the class name of the Java object. This is the same as the function converters one can register via `setJavaFunctionConverter`. Also, one can specify a native routine (i.e. C/C++/Fortran) address. This can be done using `getNativeSymbolInfo` and accessing the `address` field of the returned object. See examples in the `inst/examples/` directory.

Details

This creates a new Java object by first converting the R arguments to Java objects and then looking for a constructor in the target class that accepts arguments of these types. The resulting Java object is available for future computations as arguments to `.Java`, `.JavaConstructor`, and `.OmegahatExpression`. At present, the object must be explicitly freed by the caller. This is always true if a value is given for the `.name` argument.

Value

If a value for the argument `.name` is provided, this returns a `NamedReference` to a Java object stored in the Omegahat session. Otherwise, usually an `AnonymousReference` is returned. However, if a converter to R exists for the particular Java class being created *and* no value for the `.name` argument is given in the call, the Java object will be converted directly to an R object. This is sometimes useful when the constructor populates the object's fields and one has no further user for the object itself, but just its contents. For example, the basic constructor for the class `StatDataURL` takes a URL name and reads its contents. A converter could be registered for this class that returns the lines of text.

Note

Uses the Omegahat interactive Java environment.

Author(s)

Duncan Temple Lang, John Chambers

References

http://www.omegahat.org/RSJava

See Also

.Java .OmegahatExpression

Examples

```
tmp <- .JavaConstructor("util.Vector", as.integer(10))
.Java(tmp, "add", "This is a string")
.Java(tmp, "add", 1.5)

b <- .JavaConstructor("JButton","R Java Button")
.Java(tmp, "add", b)

f <- .JavaConstructor("GenericFrame", b)

f <- .JNew("GenericFrame", b)
```

.JavaInit *Initialize or terminate the Java Virtual Machine*

Description

.JavaInit loads and starts the Java Virtual Machine and the Omegahat session which brokers requests to Java classes and objects. The arguments to this function control the initial configuration and environment of the JVM.

Usage

```
.JavaInit(..., config=NULL, default=javaConfig(), verbose=FALSE,
          callbackHandler=javaHandlerGenerator())
```

Arguments

config	an object containing elements to be used in the classpath of the new JVM, system properties, and libraryPath elements for loading code via JNI (from Java). See javaConfig
default	the default configuration options (classpath, properties, etc.) that are merged with those from the config argument.
verbose	logical value indicating whether diagnostic information should be displayed on the screen as the JVM and Omegahat session are initialized. This is for use in debugging failures or anomalies in the startup.

`callbackHandler`
: an object (usually a closure) that handles requests from the Java/Omegahat system for method invocations on R objects exported to that foreign system. This is usually `javaHandlerGenerator`

`...`
: additional arguments passed to JavaConfiguration to influence settings.

Details

Creates and starts the JVM and Omegahat session. Also, registers a function or list of functions (closure) to handle callbacks from Java to R objects and functions.

Value

A logical value indicating whether the initialization was successful.

Author(s)

Duncan Temple Lang, John Chambers

References

http://www.omegahat.org/RSJava

See Also

javaConfig

Examples

```
## Not run:
 .JavaInit()

## End(Not run)
```

.JavaTerminate *Terminates the Java Virtual Machine*

Description

Unloads the Java Virtual Machine, releasing its resources and terminating the Omegahat session. *Once the JVM is terminated, it cannot be restarted within this R session.*

Usage

```
.JavaTerminate()
```

Details

This just calls the internal routine which notifies the JVM that it should terminate. Exactly how this action is performed depends on the current state of the JVM and the threads that are active.

Value

`TRUE` indicating that the JVM is terminated and should not be used.

Author(s)

Duncan Temple Lang

References

http://www.omegahat.org/RSJava

See Also

`.JavaInit`

Examples

```
## Not run:
   # active the JVM only to find out what
   # version of Java it supports.
   # No further activity can take place in the
   # Java session.
   #
 .JavaInit()
 jversion <- .Java("System", "getProperty", "java.version")
 .JavaTerminate()

## End(Not run)
```

.OmegahatExpression

Execute an Omegahat/Java expression

Description

This evaluates the specified expression in the Omegahat sub-system, resolving references from the Omegahat databases and the list of arguments provided in this call.

This is no longer active in the current version (0.69-0) of the package. This is done to avoid a dependency on an older version of ANTLR. If this feature is needed, use `.Java` instead or please ask for it to be reintroduced.

Usage

```
.OmegahatExpression(expr, ..., .name=NULL, .sigs="", .convert = TRUE)
```

Arguments

`expr`	A string value that is a valid Omegahat expression.
`...`	a collection of named arguments which are converted to Java objects and available to the Omegahat expression when it is evaluated using the names of the arguments.

.name	The name to use to store the result in the omegahat named reference database. If this is missing, an anonymous reference is returned or the value converted to an R object. If the result of the Java method can be converted, this argument can be used to prohibit this conversion and leave the Java value in Omegahat for use in future .Java calls.
.sigs	not really needed here, but can be used to control the conversion of the arguments in ...
.convert	logical value indicating whether the Omegahat interpreter should attempt to convert the result of the expression to an R object (TRUE), or alternatively just assign the value to a local database and return a reference. This is useful when one wishes to avoid converting an object back to its R counterpart because you wish to use it in subsequent .OmegahatExpression or .Java calls.

Details

This can be used to create functions, assign multiple values in a single call, create arrays easily, etc. One of the drawbacks of using this is that the details of the Omegahat and Java languages are exposed to the code that calls them in this manner. By using the .Java and .JavaConstructor functions, one can easily subsititute different implementations that for example, use CORBA to invoke methods in remote objects written in different languages. \ In some ways, this has similarities to substitute.

Value

The result of the Omegahat evaluation of the expression, converted from a Java object to an R object using the basic and extensible conversion mechanism between Java and R.

See Also

.Java .JavaConstructor

Examples

```
.OmegahatExpression("show(1::10)")
.OmegahatExpression("show(1::z);", z=10)
```

.RSJava.symbol *Expands a name to a C routine name in this package.*

Description

In order to avoid conflicts with other packages having the same C routine names, we use a macro RS_Java to identify the names of C routines. This akes it easy for us to generate unique names. This function allows callers of these routines from R (via the .C and .Call functions) to refer to them via their non-expanded name and have this function perform the appropriate expansion. Currently, this prefixes the regular name with "RS_JAVA_".

Usage

```
.RSJava.symbol(name)
```

Arguments

name	The unexpanded name of the C routine, i.e. without the prefix.

Value

The string identifying the C routine corresponding to the short (unexpanded) reference given in name.

Author(s)

Duncan Temple Lang

References

http://www.omegahat.com/RSJava

Examples

```
.C(.RSJava.symbol("isJVMInitialized"), logical(1))
```

RtoJavaSig *Returns the Java type identifier for an R object*

Description

This takes the given object and returns a string that can be used in the This is of most use for primitivesso that one doesn't have to remember the different Java characters representing its primitive types. This is currently not useful for non-primitive objects (e.g. lists) until javaSig is enhanced.

Usage

```
RtoJavaSig(obj)
```

Arguments

obj	Any R object, but the result is currently only meaningful if this is a primitive.

Details

This examines the class and/or mode of the specified object and then calls javaSig to find the name of the Java class corresponding to the name of the given object's type.

Value

A character vector of length 1 identifying the Java type (primitive or class) corresponding to the type of the input object.

Note

This will be enhanced in future versions as more elaborate conversion mechanisms are added.

Author(s)

John Chmabers, Duncan Temple Lang

References

http://www.omegahat.org, http://www.javasoft.com

See Also

javaSig

Examples

```
javaSig(1)
javaSig(as.integer(10))
javaSig("a string")
javaSig(list(a=1))
```

---

internal *Functions for internal or illustrative use by SJava*

Description

These functions are for internal use by SJava, or are used in primarily outdated examples.

Usage

```
## S3 method for class 'JavaException'
conditionMessage(c)
## S3 method for class 'JavaException'
conditionCall(c)
getLastJavaException()
createListener(methods, jinterface, jname = "",
               superClass = "org.omegahat.R.Java.RJavaInstance")
```

Arguments

c	A condition object.
methods	a named list of functions to be invoked, in particular actionPerformed to implement callback methods of jinterface.
jinterface	A Java class (e.g., java.awt.event.ActionListener) for which the listener is to be created.
jname	A name (as used by .JNew) used to refer to this dynamically compiled class.
superClass	The super class for the defined and dynamically compiled class.

Author(s)

Duncan Temple Lang

See Also

conditions

convertFromToJava *Convert Java instance to R instances and vice versa*

Description

These functions convert Java language structures (e.g., a Java vector) to the corresponding R structure (e.g., an R vector).

Usage

```
convertListToJava(x, ...)
convertNamedListToJava(x, ...)
convertArrayFromJava(x, ...)
convertOrderedTableFromJava(x, ...)
convertVectorFromJava(x, ...)
simplifyListToVector(x)
setDefaultConverters()
```

Arguments

x	The R object or Java reference to be convereted or simplified.
...	Additional (unused) arguments.

Value

The convert functions return the converted R or Java object. setDefaultConverters sets the converts enumerated here as the default for the corresponding data types.

Author(s)

Duncan Temple Lang

References

http://www.omegahat.org, http://www.javasoft.com

defaultJavaPath *Obtain the Java path or other variable Sys.getenv*

Description

This function queries Sys.getenv for the value of the named variable.

Usage

```
defaultJavaPath(varName="CLASSPATH")
```

Arguments

varName	The variable to be queried.

Value

The value of `varName` as defined in `Sys.getenv`.

Author(s)

Duncan Temple Lang

`expandClassName` *The fully qualified name of a Java class*

Description

This takes the partially qualified name of a Java class and queries the Omegahat class list to resolve the appropriate Java class. This then returns the name (via the Java method `getName()`) of that class with all the package information in the name.

Usage

```
expandClassName(klass)
```

Arguments

`klass`	the partially qualified name of the Java class which is to be resolved.

Details

This uses the Omegahat evaluator's class list, including the locally added classes (i.e. those not in the Java classpath, but added to the Omegahat class search path) and the dynamically generated and loaded classes.

Value

a character vector of length 1 containing the fully qualified name of the Java class.

Note

Note that this causes the Java class to be loaded. If one wanted to simply determine from which Java package a partially qualified class name would be loaded without loading it (e.g. one might want to test whether Vector is from the antlr package or the core `java.util` package) then one should loop over the evaluator's class list (`classLists()`) object and use its `matchesClassName` method.

Author(s)

Duncan Temple Lang

References

`http://www.omegahat.org/RSJava`

See Also

`.JClass`

Examples

```
expandClassName("util.Vector")
```

`foreignReference` *Create a reference to an R object for use by a foreing system.*

Description

Create an object that is used to represent a local R object in a call to a foreign system (Java), optionally specifying a name by which it is to be stored locally and known externally and the name of one or more classes/interfaces that should be used to represent it. The last of these is interpreted by the remote system.

Usage

```
foreignReference(obj, id="", className=NULL, targetClasses=NULL,
                 register=TRUE)
```

Arguments

`obj`	The R value/object that is to be represented by this reference. When methods are invoked on the reference, they are applied to this object.
`id`	An identifier for the reference by which it can be known to foreign systems and internally. This is just a name.
`className`	Name of the (R) class of this object.
`targetClasses`	The name of a class or of interfaces which the foreign system should use when representing this object. This allows the reference to implement different e.g. Java interfaces so it can be used as an argument to different methods.
`register`	A logical value indicating whether the object should be "exported" by the foreign system (`TRUE`), or otherwise just a local object created to represent the R value without making it accessible remotely. If no value for `id` is supplied and the reference is registered, a counter used to generate unique references is incremented. (Is this true?)

Value

An object of class `foreignReference`.

Author(s)

Duncan Temple Lang, John Chambers

References

http://www.omegahat.org/RSJava

See Also

`.Java` `.JavaConstructor`

Examples

```
data(mtcars)
foreignReference(mtcars, targetClasses="DataFrameInt")
```

`getForeignReferences` *Get the names of the objects in the Omegahat system*

Description

This queries the Omegahat manager for the names of the different objects it manages for use by `.Java`, `.JavaConstructor` and `.OmegahatExpression` calls. This allows the elements of either the named or anonymous or both databases to be queried.

Usage

```
getForeignReferences(which=c(named = TRUE, anonymous = FALSE))
```

Arguments

`which`	a logical vector identifying the named (`TRUE`) and anonymous (`FALSE`) databases. The default is both.

Details

This uses the `.Java` function to invoke the `getReferences` method of the Omegahat manager/evaluator. This aids one to query and control the interface manager.

Value

A list with the same length as the argument `which`. Each element is a list of This is not working as designed yet. We need to add a converter for an `InterfaceReference`.

Author(s)

Duncan Temple Lang, John Chambers

References

http://www.omegahat.org/RSJava

See Also

`.Java` `.JavaConstructor`

Examples

```
getForeignReferences()

  # Just the named values
getForeignReferences(TRUE)

  # Just the anonymous references
getForeignReferences(FALSE)
```

`getJavaConverterDescriptions` *Retrieves descriptions for the registered converters between R and Java*

Description

The conversion between R and Java objects is controlled by a list of actions. Each contains a description string to help the user understand what it does. This function returns these descriptions for the converters in one or both directions (i.e. from R to Java or vice-versa).

Usage

```
getJavaConverterDescriptions(which=c(fromJava = FALSE, toJava = TRUE))
```

Arguments

`which`	A logical vector in which `FALSE` identifies the converters from Java to R and `TRUE` indicates from R to Java.

Details

This examines the internal C data structures used to maintain the conversion tables.

Value

Returns a list with the same length as `which` in which each element is a character vector containing the description strings from the different registered conveters for that conversion direction. These include the default converters that handle the conversion of primitives between the two systems.

Author(s)

Duncan Temple Lang

References

http://www.omegahat.org/RSJava

See Also

`getNumJavaConverters` `setJavaConvertible`

Examples

```
getNumJavaConverters()
getNumJavaConverters(TRUE)
getNumJavaConverters(FALSE)
```

getJavaHandler *Obtains the current R foreign reference manager*

Description

This queries the C code to retrieve the R object that manages the exporting of R objects to foreign systems such as Omegahat and Java.

Usage

```
getJavaHandler()
```

Value

A list (or object) that provides the functions needed by a reference manager. See `setJavaHandler` and `javaHandlerGenerator` for a description of these methods.

Author(s)

Duncan Temple Lang, John Chambers

References

http://www.omegahat.org/RSJava

See Also

`setJavaHandler` `javaHandlerGenerator`

Examples

```
old <- getJavaHandler()
old$references()
old$total()
old$createReference(rnorm(100))
old$addReference(foreignReference(rnorm(100), "mydata"))
old$remove("mydata")
```

getJavaMethods *List the methods or constructors of a Java object.*

Description

This is a convenient method for obtaining a list of all the methods a Java object provides.

Usage

```
getJavaMethods(what)
getJavaConstructors(what)
```

Arguments

what	the (partially qualified) name of a Java class, or a reference to a Java object managed by the Omegahat evaluator. The latter contains the class name of the object.

Details

This is a simple use of .Java and the evaluator's methods getMethods and getConstructors. The getJavaMethods also adds the names to the resulting R list.

Value

A list of Java Method objects converted to their R equivalents. The names of the elements in the list are given by the name of the Java method. In the case of getJavaConstructors, no names are given since these have no explicit name. Each element describes the corresponding Java method in terms of the number and types of arguments, its accessibility, in which class it was defined, and the exceptions it may throw.

Author(s)

Duncan Temple Lang

References

http://www.omegahat.org/RSJava

See Also

.Java .JavaConstructor .JNew

Examples

```
v <- .JNew("util.Vector")
  # get all the methods
head(getJavaMethods(v))
  # get all the add() methods
getJavaMethods(v)[["add"]]

constr <- getJavaConstructors("util.Vector")
length(constr)
constr[[1]]
```

---

getNumJavaConverters

Returns the number of converters registered between R and Java

---

Description

This returns the number of converters currently registered between R and Java. The argument specifies the desired direction of the conversion, by default querying both from Java to R and from R to Java.

Usage

```
getNumJavaConverters(which=c(fromJava = FALSE, toJava = TRUE))
```

Arguments

which	a logical vector in which FALSE indicates from Java to R and TRUE indicates from R to Java.

Details

This accesses the internal C data structures that maintain the converter lists. These are in C so that the low-level JNI code can access them directly without the overhead of converting to reference objects and losing contextual information.

Value

An integer vector with the same length as the argument which. The value of each element is the number of registered converters in the corresponding list.

Author(s)

Duncan Temple Lang

References

http://www.omegahat.org/RSJava

See Also

getJavaConverterDescriptions

Examples

```
getNumJavaConverters()
getNumJavaConverters(FALSE)
getNumJavaConverters(TRUE)
```

---

interfaceGenerator *Generates a template "closure" to represent a Java interface/class.*

Description

To use an R variable as a Java object, one can create a closure or list of functions that implement the methods of that Java class. The template of such a closure can be generated automatically using the reflectance capabilities of both Java and R. This function generates such a template and can be used as an initial step in implementing an R version of a Java class.

Usage

```
interfaceGenerator(name, file="")
```

Arguments

name	the name of the Java class or interface whose methods are to be duplicated locally via R functions. This is resolved using the usual Omegahat rules so this can be a partially qualified class name.
file	The name of a file to which the template functions are written. This can then be edited to provide an implementation of the Java class via an R object.

Details

This is useful for allowing an R object to be converted to a proxy Java object. For example, consider using an R object as a callback for a Swing button. The object must implement the `actionPerformed()` method of the `ActionListener`. This function calls the Omegahat evaluator's `getMethods()` method to retrieve a list of Java method descriptions and then converts them to an R closure definition.

Value

This function has the side-effect of writing the definition of a function closure definition to standard output (the console) or to a file. In the future, we will generate the actual function objects. The idea is merely to show the possibilities available to us using reflectance.

Author(s)

Duncan Temple Lang

References

http://www.omegahat.org/RSJava

See Also

foreignReference

Examples

```
interfaceGenerator("java.awt.event.ActionListener")
interfaceGenerator("java.awt.event.ActionListener", "MyFile")
```

---

isJavaInitialized *Determines whether the JVM has been created.*

Description

This determines whether the Java Virtual machine has already been initialized within this R session, usually via the `.JavaInit` This is useful when we want to use the Java interface, but want to avoid an error being thrown if the user hasn't already created Java.

Usage

```
isJavaInitialized(msg=NULL)
```

Arguments

msg	A character string, which if specified and the virtual machine has not been initialized, is passed as the single argument in a call to stop.

Details

This checks the state of the internal C variables to determine if the user has initialized the JVM. It does not attempt to create the JVM. This allows the user to specify different arguments to customize the VM.

Value

A logical value indicating whether the JVM has been created earlier (TRUE) or not (FALSE). If the msg argument is specified and the JVM has not been initialized, an error is thrown and there is no return value.

Author(s)

Duncan Temple Lang

References

http://www.omegahat.org/RSJava

See Also

.JavaInit

Examples

```
 isJavaInitialized("initialize Java with .JavaInit()")
 jlabel <- .JavaConstructor("javax.swing.JButton",
                            "Welcome the R-Java interface")
 f <- .JavaConstructor("org.omegahat.Environment.GUITools.GenericFrame",
                       jlabel, TRUE)

   # optionally execute code.
if(isJavaInitialized()) {
  .Java("System", "getProperty", "java.class.path")
}
```

---

javaConfig *Returns a configuration object for initializing the Java Virtual*

Description

This integrates user arguments with default values to create an object containing the information to parameterize the Java Virtual Machine's and the Omegahat session's initial environments. The object includes a classpath specification, system properties for the JVM that can also be read by the Omegahat system to govern how its elements are instantiated and also a library path for loading shared libraries as JNI code for Java classes.

Usage

```
javaConfig(classPath=character(0), properties=character(0),
           libraryPath=character(0), options=character(0),
           default=if (exists(".javaConfig")) .javaConfig
                   else JavaConfiguration())
```

Arguments

classPath	a character vector identifying locations of Java classes in the form of URLs, Jar files or simple directories. This is passed to the JVM as the argument to classpath. \ If this is not specified, the environment variable CLASSPATH is queried and if this set, its elements are prefixed to the default ones.
properties	a named character vector of system properties that are passed to the JVM initialization as -Dname=value for each element.
libraryPath	a character vector identifying directories which are to be used by Java when loading shared libraries/DLLs via System.loadLibrary()
options	a character vector of strings that are passed as is to the initialization of the virtual machine. These are basically non-property command line arguments that one can pass when starting the Java virtual machine in the usual way (e.g. calling the java executable). These include arguments such as mx of specifying the maximum amount of memory (e.g."-Xmx128m" for 128 megabytes), verbose option (e.g. "-verbose" or "-verbose:gc,class"), etc. See the documentation for your JVM or "The Java Native Interface" by Sheng Liang (page 250–251) or any JNI book.
default	a list containing the default values for each of the 3 fields/groups of parameters to which are added the user-specified values in the corresponding earlier arguments.

Value

A list that can be used to customize the initialization of the Java Virtual Machine embedded within the R session.

classPath	a character vector whose elements are sources of Java class files. These can be directories, URLs or Jar files (or any other form understood by the JVM). This is collapsed and specified as the value of the classpath argument to the JVM initialization.
properties	a named character vector whose values correspond to name=value pairs that are passed to the JVM as system properties in the form -Dname=value.
libraryPath	a character vector whose elements specify directories which are searched by the JVM when loading native code for a Java class via the System.loadLibrary() method.

Author(s)

John Chambers, Duncan Temple Lang

References

http://www.omegahat.org/RSJava

See Also

.JavaInit mergePath mergeProperties .javaConfig

Examples

```
javaConfig()
javaConfig(classPath=c("/home/duncan/Java", "/home/duncan/xml.jar",
                       paste(system.file("org","omegahat","Jars"),
                             "DataStructures.jar",sep="/")))
```

---

javaHandlerGenerator

Manages exporting of R objects to Java/Omegahat and calls from the

---

Description

This creates a closure that manages objects exported from the R session to Java/Omegahat as arguments to constructors and methods in that system. When the Java code invokes a method on such a reference, the R object is resolved by this closure and the appropriate R function invoked on that object.

Usage

```
javaHandlerGenerator()
```

Value

A closure containing the "methods"

handler()	brokers a method request for a reference under the management of this handler, taking care of passing the arguments, identifying the appropriate method, and catching errors.
createReference()	creates an actual foreignReference object by calling the foreignReference function.
addReference	adds an object to the list being managed by this reference handler. An explicit name can be provided in the call to this method, or otherwise a unique one is generated by the manager itself.
remove	discards the identified object from the list of objects being managed by this reference handler.
getReference	retrieves a particular object being managed by this reference manager using the name of the reference.
references	returns a (named) list of all the objects being managed by this reference manager.
total	returns the number of references that have been managed by this object. This is used in constructing new unique names when an object is registered without an explicit identifier.

Author(s)

Duncan Temple Lang, John Chambers

References

http://www.omegahat.org/RSJava

Examples

```
## Not run:
.JavaInit(callbackHandler = javaHandlerGenerator())

## End(Not run)
```

---

javaIs *Performs class comparisons for Java objects*

Description

Allows one to test if a Java object (in Omegahat) is an object of a particular class, or implements a particular Java interface.

Usage

```
javaIs(obj, klass, instanceof=TRUE)
```

Arguments

obj	the Java object whose class is being queried and compared
klass	the name of a Java class or the Class reference object with which the object obj is being compared.
instanceof	a logical value indicating whether the comparison should be done using the equality of classes or the Java instanceof semantics. The former tests whether the class of obj is the same as the class identified by klass. The latter identifies whether obj implements the Java interface class klass. (There is also the *assignable from* semantics which may or may not be currently present).

Details

This calls the Omegahat evaluator's is method.

Value

A logical value indicating whether the class of obj is related to klass in the specified comparison. This is TRUE or FALSE.

Author(s)

Duncan Temple Lang, John Chambers

References

http://www.omegahat.org/RSJava

See Also

.JNew .Java

Examples

```
x <- .JNew("java.util.Vector")

  # TRUE
javaIs(x, "java.util.Vector")
javaIs(x, "java.util.List", TRUE)

 # FALSE
javaIs(x, "java.util.Hashtable")
```

.javaMatchFunctions *Symbolic constants for how classes are matched in conversion*

Description

The R-Omegahat interface provides 3 built-in routines for determining whether a Java object matches a particular class or not. These routines are used in determining whether a converter applies to a particular Java object and is capable of converting it to an R object. This vector is a named integer vector where the names are symbolic identifiers for the integers that allow the R and C code to identify which of these 3 built-in routines is meant.

---

javaSig *Converts an R type name to a Java type*

Description

When specifying a Java type in the .sigs argument of the different method/constructor calls to influence which method is dispatched in the remote system and how R objects are converted, one must use the appropriate type specifier. This function converts R types to the corresponding Java string. This handles converting R primitive types such as integer to "I", double to "D", logical to "Z", etc. and classes to "Lpkg/subpkg/.../className;"

Usage

```
javaSig(name)
```

Arguments

name	The name of the R type whose corresponding Java type identifier is being sought. If this is specified as an object, its mode is take. See RtoJavaSig

Details

This searches the mapping contained in .JavaSigs

Value

a string (character vector of length 1) with the Java type identifier corresponding to the input.

Author(s)

John Chambers, Duncan Temple Lang

References

http://www.omegahat.org/RSJava

See Also

RtoJavaSig

---

jdynamicCompile *Dynamically Compile a Java Class*

Description

This uses the dynamic byte-code generator facility in Omegahat to create a new Java class with a given name. By default, the new class extends the `RForeignReference` class and implements a specifiable collection of Java interfaces. Each method in these interfaces is implemented by calling the corresponding R function in the R reference object.

Usage

```
jdynamicCompile(interface, newClass,
                generatorClass="ForeignReferenceClassGenerator",
                load=TRUE)
```

Arguments

`interface`	the (fully qualified) name of one or more Java interface classes that the new class should implement. The methods in these interfaces are implemented as calls to the correspondingly named R function in the R object identified by the `RForeignReference`'s key.
`newClass`	the name of the new class to be created.
`generatorClass`	the name of the Java class to which is to be used as the dynamic compiler. This is rarely specified, but allows one to use classes that implement the code differently, e.g. by extending `ForeignReferenceClassGenerator`.
`load`	logical value indicating whether the new class should be loaded into the Omegahat list of available classes. If this is `FALSE`, one usually writes the newly generated class definition to a file.

Author(s)

Duncan Temple Lang

References

http://www.omegahat.org

Examples

```
 ## Not run:
  jdynamicCompile("java.awt.event.ActionListener", "RActionListener")
  l <- .JNew("RActionListener",
             foreignReference(list(actionPerformed=function() print("ok"))))
  button$add(l)

## End(Not run)

## Not run:
  def <- jdynamicCompile("java.awt.event.ActionListener",
                         "RActionListener", load=F)
  def$write()

## End(Not run)
```

---

mergePath *Merges classpath specifications*

Description

Merges two character vectors of classpaths for use in creating arguments to initialize the Java Virtual Machine within R. This avoids duplicates.

Usage

```
mergePath(path, default, collapse=NULL)
```

Arguments

path	character vector of path elements
default	a character vector containing the default or previous classpath elements
collapse	a character vector used the value of a the collapse argument in a call to paste. If this is non-null, the resulting vector is converted to a single string. This is usually given as ":" on Unix machines and ";" on Windows machines.

Value

A character vector containing the union of the two arguments. If the collapse argument is specified, the elements of the resulting character vector are concatenated/pasted together to yield a single string.

Author(s)

Duncan Temple Lang, John Chambers

References

http://www.omegahat.org/RSJava

See Also

.JavaInit javaConfig

Examples

```
mergePath("~/Java/MySQL/mm.mysql-2.0.1/mysql.jar",c("$OMEGA_HOME/Jars/antlr.jar","$OMEG
mergePath("~/Java/MySQL/mm.mysql-2.0.1/mysql.jar",c("$OMEGA_HOME/Jars/antlr.jar","$OMEG
```

---

mergeProperties *Creates the union of two named character vectors, converting to a*

Description

Utility function to merge the two named objects, with elements single character strings, and convert the result to a vector of Java property settings.

Usage

```
mergeProperties(props, default, convert=TRUE)
```

Arguments

props	named character vector of properties.
default	named character vector of properties with which the elements in props are to be merged. The values in props take
convert	a logical value which, if TRUE causes the the resulting character vector to be converted to Java property specifications for use in intializing the Java Virtual machine, each of the form -Dname=value

Value

Augments the default with the named values that are in props and not in default and also replaces those shared by both vectors with those in props. If the argument convert is TRUE, elements of the vector are converted to Java properties suitable for initializing the JVM.

See Also

mergePath .JavaInit javaConfig

Examples

```
props <- c(java.compiler="", myProperty="abc", "X_Y"="Hi there")
mergeProperties(props, javaConfig()@properties)
```

`omegahatReference` *Creates a local object representing a Java reference*

Description

On occassion, one can lose a reference to a Java object stored in the Omegahat databases. If one knows its identifier (i.e. Omegahat name) and whether it is an anonymous or named reference, one can recreate an R object that refers to this Java object and use this R object in subsequent calls to the Java interface.

Usage

```
omegahatReference(key, named=TRUE)
```

Arguments

`key`	the name used by Omegahat to store the Java object.
`named`	a logical value indicating whether this is a named (`TRUE`) or anonymous (`FALSE`) reference.

Value

An object of class either `AnononymousOmegahatReference` or `NamedOmegahatReference`. This has fields

`key`	The Omegahat name by which the Java object is known. This is the value of the argument `key`.
`className`	the class name of the Java object. This is always empty.

Author(s)

Duncan Temple Lang

References

http://www.omegahat.org/RSJava

See Also

`.Java` `.JavaConstructor`

Examples

```
omegahatReference("x")
omegahatReference("x",FALSE)
```

`removeJavaConverter` *Removes a converter for R and Java objects in the R-Java interface.*

Description

This manipulates the internal list of object converters that control how objects are transferred from R to Java and from Java to R. It allows the R user to remove entries and control how objects are converted.

Usage

```
removeJavaConverter(id, fromJava=TRUE)
```

Arguments

`id`	the index or position of the converter to be removed in the specified converter list.
`fromJava`	logical value indicating which set of converters on which we are operating: from Java to R (`TRUE`) or from R to Java (`FALSE`). The latter is currently not implemented and awaits the next version which will use C++.

Value

Returns a integer identifying the position in the list in which the converter was located. This is a named vector and the name is the description of the converter. This allows one to ensure that you got the correct one.

Author(s)

Duncan Temple Lang

References

http://www.omegahat.org/RSJava

See Also

`setJavaConverter` `setJavaConvertible` the `.convert` argument of `.Java` and `.JavaConstructor`

Examples

```
   # remove the Constructor converter
## Not run:
 removeJavaConverter(3)

   # add a converter -- userObject must be valid
 setJavaConverter(.RSJava.symbol("RealVariableConverter"),
                  matcher="AssignableFrom",
                  autoArray=TRUE,
                  description="Omegahat RealVariable to numeric vector",
                  userObject="RealVariable")
```

```
  # and remove it by specifying its description.
removeJavaConverter("Omegahat RealVariable to numeric vector")

  # add the converter again
cvt <- setJavaConverter(.RSJava.symbol("RealVariableConverter"),
                        matcher="AssignableFrom",
                        autoArray=TRUE,
                        description="Omegahat RealVariable to numeric vector",
                        userObject="RealVariable")

  # and remove it by specifying its position
  # which is given to us by the setJavaConverter call.
removeJavaConverter(cvt$index)

## End(Not run)
```

`setJavaConvertible` *Register a Java class as being convertible to an R object*

Description

When a value is to be returned from Java to R, the Omegahat evaluator examines a table to determine if an object of that type can be converted to R. This function manipulates that table and allows one to control for what types of objects conversion is attempted.

This is not used currently. The Java objects are returned directly and the user-level converters determine whether the object is 'convertible'.

Usage

```
setJavaConvertible(klass, ok=TRUE, matching=0)
```

Arguments

`klass`	The name of a Java class or interface which is to be added or removed from the set of known convertible classes. This can be a partially qualified class name that is resolved by the Omegahat evaluator.
`ok`	logical value indicating whether objects of class `klass` are to be considered convertible or not by the Omegahat sub-system.
`matching`	an integer value from the set 0, 1 and 2. These values indicate how comparisons between the registerd class and the object to be converted are performed. **0** An exact match, meaning that the class of the object must be the same as the class being registered as convertible. **1** the target object must be an instance of the class being registered, meaning that it implements this Java interface or is an instance of a class that is derived from the registered one. **2** the target object must be compatible with the registered class in the sense of `isAssignableFrom` method between two classes. These constants are defined in the Java interface `ConvertibleClassifierInt`.

Details

Simply calls `setJavaConvertible` in the Omegahat evaluator which passes the request on to the `ConverterClassifierInt` object employed by that evaluator.

Value

NULL corresponding to a call to a Java method returning `void`.

Author(s)

Duncan Temple Lang

References

http://www.omegahat.org/RSJava

See Also

`setJavaConverter` `getJavaConverterDescriptions` `getNumJavaConverters`

`setJavaConverter` *Add a converter from Java to an R object*

Description

Register a C routine which converts a Java object to an R object. This occurs when a value is returned from a Java method (or constructor) call via `.Java` or `.JavaConstructor`.

Usage

```
setJavaConverter(handler, matcher=-1, autoArray=TRUE, description="",
                 userObject=NULL, register = TRUE)
```

Arguments

`handler`	The name of a C routine that performs the conversion from the Java object to the R object. This is given the Java object, the class of that object, the JNI environment and the element in the converter object is to be called when the the `matcher` determines that
`matcher`	The name of a routine that is used to determine whether this converter can handle a specific object. This can also be specified as an element of the vector `.javaMatchFunctions`, either as (part of) a name of an element or the integer value. These are then used to identify one of the built-in converter matching functions.
`autoArray`	A logical value indicating whether this converter routine can be called element-wise for an array of the class type it matches (TRUE) , or whether it wishes to defer the handling of such an array to another converter or deal with it all in one step.
`description`	A string that describes the action of the converter (e.g. the type of source Java class and target R object on which it operates). This is stored with the internal converter and accessible to users via the `getJavaConverterDescriptions`.

`userObject` If the `matcher` argument identifies one of the built-in matching routines (i.e. assignable from, instance of, equals) this is interpreted as a Java class identifier. That is either a class name (which is resoloved, and expanded as necessary, by Omegahat) and used to parameterize the particular use matching routine.

`register` a logical value indicating whether this call should also notify Java that the specified class (i.e. that given in `userObject`) is convertible. This calls `setJavaConvertible` with the class and matching mechanism specified for this function.

Value

This returns the expanded named of the class used to parameterize the matching function and the identifier for the matching function itself.

`match` the value passed to the C routine identifying the matching function. This is either an element from `.javaMatchFunctions` (hence a named integer) or a string identifying the C routine.

`class` The name of the class used to parameterize the matching function, if the latter is one of the built-in routines named in `.javaMatchFunctions`. The class name is resolved by Omegahat and converted to use '/' instead of '.' to separate the Java packages. This is so that it can be easily used in the native C code.

`index` the position in the list into which this converter was added. This is useful if we want to remove the converter at a later stage via `removeJavaConverter`.

`description` the description argument passed to this function call. As with the `index` field, this is useful when we wish to remove the converter as it acts as an identifier for the converter. See `removeJavaConverter`.

Note

In the near future, we will re-establish the mechanism for specifying R functions or closure instances for the handler and matcher. This has become more complicated than intended and will probably be restricted to work only for C routines. All of the cases have not been tested entirely.

Author(s)

Duncan Temple Lang

References

http://www.omegahat.org, http://www.javasoft.com

See Also

`getJavaConverterDescriptions` `getNumJavaConverters` `setJavaConvertible`

Examples

```
## Not run:
 setJavaConverter(.RSJava.symbol("RealVariableConverter"),
                  matcher="AssignableFrom",
                  autoArray=TRUE,
                  description="Omegahat RealVariable to numeric vector",
                  userObject="RealVariable")
```

```
## End(Not run)
```

`setJavaFunctionConverter`
Registers a function to convert between R and Java objects

Description

This allows one to register two functions that are used to convert an object from Omegahat/Java to an R object. One function (`match`) determines whether the other function (`handler`) that actually performs the computation is suitable for converting the target object. The result of the `handler` function is an R object that represents the Java object being converted.

Usage

```
setJavaFunctionConverter(handler, match, description=NULL, fromJava=TRUE,
       autoArray = TRUE, position = -1)
```

Arguments

`handler`	a function that takes two arguments: a reference to the Java object to be converted and the name of its Java class. This function should return the converted value or the reference to the Java object if it cannot convert it meaningfully.
`match`	a function that deterines whether the associated `handler` function is appropriate for converting the target Java object. This function should expect two arguments: a reference to the Java object and its class name. It *must* return a logical value indicating whether the `handler` is capable of converting the Java object.
`description`	a descripion that is stored internally with the converter and accessible to users via the function `getJavaConverterDescriptions`.
`fromJava`	a logical value indicating whether the functions are intended for converting from Java to R or vice-versa. Currently, the R to Java mechanism is not implemented.
`autoArray`	a logical value indicating whether R is to deal with Java arrays in relation to this converter by calling the match/predicate function with an element of the array (`TRUE`) or the array itself (`FALSE`).
`position`	the index (starting at 1) at which to insert the converter. If this is non-positive, the converter entry is appended at the end of the list.

Value

An object of class `"JavaFunctionConverter"` with fields

`index`	the position of this converter in the internal list of converters. This is a useful identifier for removing the converter.
`description`	the value of the `description` argument. This is also a useful and preferred identifier for removing the converter.
`handler`	the value of the `handler` argument.
`match`	the value of the `match` argument.

Note

The R to Java converter mechanism will be added in the next release.

Author(s)

Duncan Temple Lang

References

http://www.omegahat.org/RSJava/index.html

See Also

setJavaConverter setJavaConvertible .Java .JavaConstructor

Examples

```
 # this must be run wit the ROmegahatExamples.jar
 # file in the classpath (e.g.
 #     .JavaInit(list(classPath=system.file("org/omegahat/Jars/ROmegahatExamples.jar")))
 # so as to be able to find RealVariable!

if(!is.null(.Java("__Evaluator", "expandClassName", "RealVariable"))) {
  cvt <- setJavaFunctionConverter(function(jobj,className) {
      .Java(jobj, "getValues")
  }, function(jobj, className) {
      return(className == "org.omegahat.DataStructures.Data.RealVariable")
  }, "Omegahat RealVariable to numeric vector")

  setJavaConvertible("RealVariable")
  .JavaConstructor("RealVariable", rnorm(10))

    # now unregister the converter
  setJavaConvertible("RealVariable", FALSE)
  removeJavaConverter(cvt$index)
}
```

---

setJavaHandler *Register a handler for exporting R objects to foreign systems.*

Description

Registers a manager for R objects that are to be used by foreign systems such as Omegahat and Java. The registration provides access to the object from C routines that bridge the two systems - R and the foreign one.

Usage

```
setJavaHandler(handler)
```

Arguments

handler	An object that manage the references and provide the different functions for accessing and manipulating those references. These functions are described below and implemented in javaHandlerGenerator()

Details

This registers the handler object with the C code so that it is known to the Java methods. The methods that must be provided are

handler() brokers a method request for a reference under the management of this handler, taking care of passing the arguments, identifying the appropriate method, and catching errors.

createReference() creates an actual foreignReference object by calling the foreignReference function.

addReference adds an object to the list being managed by this reference handler. An explicit name can be provided in the call to this method, or otherwise a unique one is generated by the manager itself.

remove discards the identified object from the list of objects being managed by this reference handler.

getReference retrieves a particular object being managed by this reference manager using the name of the reference.

references returns a (named) list of all the objects being managed by this reference manager.

total returns the number of references that have been managed by this object. This is used in constructing new unique names when an object is registered without an explicit identifier.

The intent of the handler is to allow the management of the objects being exported to the foreign system(s) (Java and Omegahat)

Value

The previous value of the registered handler that has been replaced with this value. This allows one to temporarily replace a handler with a new version and then swap the original back at the end of an operation/transaction.

Author(s)

Duncan Temple Lang

References

http://www.omegahat.org/RSJava/Conversion.html

See Also

getJavaHandler

Examples

```
## Not run:
  old <- setJavaHandler(javaHandlerGenerator())
  setJavaHandler(old)

## End(Not run)
```

Index