Package ‘jdx’

June 4, 2020

Type Package

Title 'Java' Data Exchange for 'R' and 'rJava'

Description Simplifies and extends data exchange between 'R' and 'Java'.

Version 0.1.4

License GPL (>= 2) | BSD_3_clause + file LICENSE

Encoding UTF-8

Imports rJava (>= 0.9-8), utils (>= 3.3.0)

SystemRequirements Java Runtime Environment (>= 8)

RoxygenNote 7.1.0

Suggests testthat, knitr, rmarkdown, pander

VignetteBuilder knitr

NeedsCompilation no

URL `https://github.com/floidgilbert/jdx`

BugReports `https://github.com/floidgilbert/jdx/issues`

Author Floid R. Gilbert [aut, cre],
David B. Dahl [aut]

Maintainer Floid R. Gilbert <`floid.r.gilbert@gmail.com`>

Repository CRAN

Date/Publication 2020-06-04 20:50:02 UTC

R topics documented:

jdx-package *Java Data Exchange for R and rJava*

Description

Builds on **rJava** to simplify and extend data exchange between R and Java.

Details

The **jdx** package works in conjunction with **rJava** to provide a low-level interface for the Java platform. The **jdx** package was originally developed to provide data exchange functionality for **jsr223**, the high-level scripting interface for the Java platform. We provide **jdx** to developers who may want to extend existing **rJava** solutions. For developers of new applications, we suggest the **jsr223** package for rapid application development with a relatively low learning curve.

The **jdx** package converts R data structures to generic Java objects and vice versa. In particular, R vectors, n-dimensional arrays, factors, data frames, tables, environments, and lists are converted to Java objects. Java scalars and n-dimensional arrays are converted to R vectors and n-dimensional arrays. Java maps and collections are converted to R lists, data frames, vectors, or n-dimensional arrays depending on content. Several options are available for data conversion including row-major and column-major ordering for arrays and data frames.

For sites that plan on designing and distributing packages that depend on **jdx**, it may be helpful to know that the **jdx** package does not use or load rJava's companion package **JRI** (the Java/R Interface).

For best results, please refer to the vignette Introduction to jdx: Java Data Exchange for R and rJava.

Author(s)

Floid R. Gilbert <`floid.r.gilbert@gmail.com`>, David B. Dahl <`dahl@stat.byu.edu`>

See Also

`convertToJava` , `convertToR` , `getJavaClassName`

Examples

```
library("jdx")

# Convert matrix using column-major ordering
m <- matrix(1:4, 2, 2)
o = convertToJava(m, array.order = "column-major")
getJavaClassName(o)
identical(m, convertToR(o, array.order = "column-major"))

# Convert 4-dimensional array using row-major ordering
dimensions <- c(3, 2, 2, 2)
a = array(1:prod(dimensions), dimensions)
o = convertToJava(a, array.order = "row-major")
```

```
getJavaClassName(o)
identical(a, convertToR(o, array.order = "row-major"))

# Convert data frame
identical(iris, convertToR(convertToJava(iris)))

# Demonstrate exact double precision
identical(pi, convertToR(convertToJava(pi, scalars.as.objects = TRUE)))
```

`convertToJava` *Convert R Objects to Java Objects*

Description

The function `convertToJava` converts R objects to generic Java objects for use with the **rJava** package. This function simplifies and extends data exchange for **rJava**. The function `convertToJava` is the inverse of `convertToR`.

Usage

```
convertToJava(
  value,
  length.one.vector.as.array = FALSE,
  scalars.as.objects = FALSE,
  array.order = "row-major",
  data.frame.row.major = TRUE,
  coerce.factors = TRUE
)
```

Arguments

`value`	An R vector, matrix, n-dimensional array, table, factor, data frame, list, or environment. Nested lists are supported. Supported data types: numeric, integer, character, logical, and raw.
`length.one.vector.as.array`	A logical vector of length one. See R Vectors of Length One in the vignette.
`scalars.as.objects`	A logical vector of length one. See R Vectors of Length One in the vignette.
`array.order`	A character vector of length one specifying the order when copying R n-dimensional arrays to Java. Valid values are `"row-major"`, `"column-major"`, and `"column-minor"`. See R Matrices and N-dimensional Arrays in the vignette.
`data.frame.row.major`	A logical vector of length one. When `TRUE` (the default), a data frame is converted to a list of map objects that represent rows. When `FALSE`, a data frame is converted to a map of arrays that represent columns. Conversion for column-major order is much faster than row-major order. See R Data Frames in the vignette.

`coerce.factors`	A logical vector of length one. When `TRUE` (the default), an attempt is made to coerce the character values backing factors to integer, numeric, or logical vectors. If coercion fails, the factor is converted to a character vector. When `FALSE`, the factor is converted to a character vector. This parameter affects standalone factors as well as factors present in data frames and lists. See R Factors in the vignette.

Details

The `convertToJava` function is used to create objects that can be used as method parameters in the **rJava** package. R vectors, matrices, n-dimensional arrays, tables, factors, data frames, environments, lists, named lists, and nested lists are supported as well as data types numeric, integer, logical, character and raw.

The vignette contains all documentation for `convertToJava` and its inverse function `convertToR`. Note that these functions are not always perfect inverses of each other. See Conversion Issues for more information.

Value

A Java object reference or an R vector. See the vignette for details.

See Also

`convertToR` , `getJavaClassName`

Examples

```
library("jdx")

# Convert matrix using column-major ordering
m <- matrix(1:4, 2, 2)
o = convertToJava(m, array.order = "column-major")
getJavaClassName(o)
identical(m, convertToR(o, array.order = "column-major"))

# Convert 4-dimensional array using row-major ordering
dimensions <- c(3, 2, 2, 2)
a = array(1:prod(dimensions), dimensions)
o = convertToJava(a, array.order = "row-major")
getJavaClassName(o)
identical(a, convertToR(o, array.order = "row-major"))

# Convert data frame
identical(iris, convertToR(convertToJava(iris)))

# Demonstrate exact double precision
identical(pi, convertToR(convertToJava(pi, scalars.as.objects = TRUE)))
```

`convertToR` *Convert Java Objects to R Objects*

Description

The function `convertToR` converts the Java objects referenced by **rJava** objects to R objects. The function `convertToR` is the inverse of `convertToJava`.

Usage

```
convertToR(
  value,
  strings.as.factors = NULL,
  array.order = "row-major"
)
```

Arguments

`value`	An **rJava** object reference.
`strings.as.factors`	A logical vector of length one specifying whether string vectors are automatically converted to factors when Java objects are converted to R data frames. This parameter is discussed in the vignette under Java Maps.
`array.order`	A character vector of length one specifying the order used to copy Java n-dimensional arrays to R. Valid values are `"row-major"`, `"column-major"`, and `"column-minor"`. See Java One-dimensional Arrays and N-dimensional Rectangular Arrays in the vignette.

Details

The `convertToR` function is not thread-safe. Do not simultaneously call `convertToR` from different threads in the same process. A thread-safe alternative is presented in the R documentation for `convertToRlowLevel`.

The vignette contains all documentation for `convertToR` and its inverse function `convertToJava`. Note that these functions are not always perfect inverses of each other. See Conversion Issues for more information.

Value

An R object. See the vignette for details.

See Also

`convertToJava` , `getJavaClassName`

Examples

```
library("jdx")

# Convert matrix using column-major ordering
m <- matrix(1:4, 2, 2)
o = convertToJava(m, array.order = "column-major")
getJavaClassName(o)
identical(m, convertToR(o, array.order = "column-major"))

# Convert 4-dimensional array using row-major ordering
dimensions <- c(3, 2, 2, 2)
a = array(1:prod(dimensions), dimensions)
o = convertToJava(a, array.order = "row-major")
getJavaClassName(o)
identical(a, convertToR(o, array.order = "row-major"))

# Convert data frame
identical(iris, convertToR(convertToJava(iris)))

# Demonstrate exact double precision
identical(pi, convertToR(convertToJava(pi, scalars.as.objects = TRUE)))
```

`convertToRlowLevel` *Low-level Interface for jdx*

Description

The functions listed here are the low-level interface for **jdx** and are primarily used behind the scenes in **jsr223**, the high-level integration package for Java. However, these functions may also be useful for **rJava** developers interested in a thread-safe alternative to `convertToR`. See the code examples for a brief outline. If multi-threaded access is not required, please use `convertToR`.

Usage

```
arrayOrderToString(value)

convertToRlowLevel(
  j2r,
  data.code = NULL,
  strings.as.factors = NULL
)

createJavaToRobject()

jdxConstants()

processCompositeDataCode(
  j2r,
  composite.data.code,
  throw.exceptions = TRUE,
  warn.missing.logical = TRUE,
  warn.missing.raw = TRUE
)
```

Arguments

`value`	An **rJava** object reference to a `org.fgilbert.jdx.JavaToR$ArrayOrder` enumeration value.
`j2r`	An **rJava** object reference to a `org.fgilbert.jdx.JavaToR` object. The `createJavaToRobject` function creates an instance.
`data.code`	A **jdx** data code value created with `processCompositeDataCode`.
`strings.as.factors`	Same as in `convertToR`.
`composite.data.code`	A **jdx** composite data code returned by the `initialize` method of `org.fgilbert.jdx.JavaToR`.
`throw.exceptions`	A logical value indicating whether to throw exceptions.
`warn.missing.logical`	A logical value indicating whether to raise a specific warning.
`warn.missing.raw`	A logical value indicating whether to raise a specific warning.

Details

See the code examples below for a thread-safe alternative to `convertToR`. The low-level functional interface presented here is awkward, but it was designed to limit type inference and the number of transactions between R and the JVM, ultimately maximizing performance for **jsr223**.

See Also

`convertToJava` , `convertToR` , `getJavaClassName`

Examples

```
library("jdx")

# Create org.fgilbert.jdx.JavaToR object used to convert
# java objects to R objects. Create one of these objects
# per thread for thread-safe execution.
#
# It is also possible to create and use the JavaToR
# object in Java and return a reference to R via rJava.
j2r <- createJavaToRobject()

# Pass the Java object to be converted to the initialize
# method of the JavaToR object. Note that the Java object
# must be cast as java.lang.Object. The initialize method
# returns an integer value known as a composite data code
# that is used to derive the R structure.
composite.data.code <- rJava::.jcall(
  j2r
  , "I"
  , "initialize"
  , rJava::.jcast(convertToJava(iris))
)

# Process the resulting composite data code to get a data
# code vector. This step also raises any applicable
# errors/warnings.
data.code <- processCompositeDataCode(j2r, composite.data.code)

# Pass the JavaToR object and the data code to
# convertToRlowLevel to get the R object.
convertToRlowLevel(j2r, data.code, strings.as.factors = FALSE)

# When converting n-dimensional arrays, pass an array
# ordering constant to the initialize method.
array.order.constants <- jdxConstants()$ARRAY_ORDER
array <- convertToJava(as.matrix(iris[1:4]), array.order = "column-major")
composite.data.code <- rJava::.jcall(
  j2r
  , "I"
  , "initialize"
  , rJava::.jcast(array)
  , array.order.constants[["column-major"]]
)
data.code <- processCompositeDataCode(j2r, composite.data.code)
convertToRlowLevel(j2r, data.code)
```

`getJavaClassName` *Get a Java Object's Class Name*

Description

The function `getJavaClassName` returns the class name for the Java object behind an **rJava** object reference.

Usage

```
getJavaClassName(value)
```

Arguments

`value`	An **rJava** object reference.

Value

A character vector of length one containing the class name of the Java object.

See Also

convertToJava , convertToR

Examples

```
library("jdx")
getJavaClassName(convertToJava(matrix(1:4, 2, 2)))
getJavaClassName(convertToJava(iris))
```

Index