# Package: testRapi (via r-universe)

August 14, 2024

**R6GeneratorNote** Generated by r6-generator-maven-plugin: remove this line if you want to make manual changes and dont want them to get overwritten

**Type** Package

**Title** A Test Library

**Version** 1.1.0

**Description** Documents the features of the 'r6-generator-maven-plugin' by providing an example of an R package automatically generated from Java code by the plugin. It is not intended to be useful beyond testing, demonstrating and documenting the features of the r6 generator plugin.

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**VignetteBuilder** knitr

**Suggests** here, roxygen2, devtools, tidyverse, knitr, rmarkdown, testthat

**Imports** ggplot2, readr, tibble, dplyr, rJava, R6, fs, cli, rappdirs, utils, magrittr, rlang, xml2, stringr

**URL** https://terminological.github.io/r6-generator-docs/index.html, https://github.com/terminological/r6-generator-docs

**BugReports** https://github.com/terminological/r6-generator-docs/issues

**RoxygenNote** 7.3.1

**Repository** https://terminological.r-universe.dev

**RemoteUrl** https://github.com/terminological/r6-generator-docs

**RemoteRef** 1.1.0

**RemoteSha** 927590af07bb1258c1c4b90a5a5c007716ae1dd5

# Contents

testRapi-package . . . 2
.background_cancel . . . 3
.background_cancel_all . . . 3
.background_get_by_id . . . 4
.background_status . . . 4
.background_tidy_up . . . 5
async_factory . . . 5
async_static_countdown . . . 6
BounceTest . . . 6
collider.feature_test . . . 17
collider.more_feature_test . . . 17
concat . . . 18
create . . . 18
demo_static . . . 19
deserialise_dataframe . . . 20
deserialise_list . . . 20
deserialise_named_list . . . 21
diamonds . . . 21
FactoryTest . . . 22
FeatureTest . . . 29
JavaApi . . . 37
manualFunction . . . 50
MinimalExample . . . 51
MoreFeatureTest . . . 52
Serialiser . . . 55
serialise_dataframe . . . 56
serialise_list . . . 56
serialise_named_list . . . 57

**Index** **58**

---

`testRapi-package` *testRapi: A Test Library*

---

## Description

Documents the features of the 'r6-generator-maven-plugin' by providing an example of an R package automatically generated from Java code by the plugin. It is not intended to be useful beyond testing, demonstrating and documenting the features of the r6 generator plugin.

Version: 1.1.0

Classes:

- `JavaApi`
- `MinimalExample`
- `BounceTest`

- `FeatureTest`
- `MoreFeatureTest`
- `FactoryTest`
- `Serialiser`

## Author(s)

- Rob Challen rob@terminological.co.uk
- Rob Challen rob@terminological.co.uk 0000-0002-5504-7768
- terminological

---

`.background_cancel` *Async cancel one*

---

## Description

Cancel an asynchronous operation by id.

## Usage

```
.background_cancel(id)
```

## Arguments

| | |
|---|---|
| `id` | the id from ‘.background_status()‘ |

## Value

nothing, called for side effects

---

`.background_cancel_all` *Async cancel all*

---

## Description

Cancel all asynchronous operations.

## Usage

```
.background_cancel_all()
```

## Value

nothing, called for side effects

## .background_get_by_id *Get a RFuture by id*

### Description

Get a background task from the ‘.background_status()‘ list and wrap it in an R6 RFuture class. This

### Usage

```
.background_get_by_id(id)
```

### Arguments

| | |
|---|---|
| id | the id from ‘.background_status()‘ |

### Value

an RFuture

## .background_status *Async function status*

### Description

Print any asynchronous java function calls that are in progress, complete or cancelled. The function returns a thread id which may be used to retrieve the thread itself

### Usage

```
.background_status()
```

### Value

a maybe empty dataframe with ‘id‘ and ‘status‘ columns

## .background_tidy_up *Tidy up completed async operations*

### Description

Remove all processed and cancelled aync operations from the status list. This will potentially free up some system resources

### Usage

```
.background_tidy_up()
```

### Value

nothing, called for side effects

## async_factory *no title*

### Description

no description

### Usage

```
async_factory()
```

### Value

an ‘RFuture‘ with methods ‘cancel()‘, ‘isCancelled()‘, ‘isDone()‘ and ‘get()‘. The result of a ‘get()‘ call will be an R6 FactoryTest object:

`async_static_countdown` *no title*

## Description

no description

## Usage

```
async_static_countdown(
 label,
 rtimer
)
```

## Arguments

| | |
|---|---|
| `label` | label - (java expects a RCharacter) |
| `rtimer` | rtimer - (java expects a RInteger) |

## Value

an ‘RFuture‘ with methods ‘cancel()‘, ‘isCancelled()‘, ‘isDone()‘ and ‘get()‘. The result of a ‘get()‘ call will be an RCharacter:

`BounceTest` *BounceTest*

## Description

missing description

Version: 1.1.0

Generated: 2024-05-16T16:22:15.451498127

## Details

no details

## Methods

### Constructors:

- J$BounceTest$new()

### Class methods:

- instance$bounceNull(x)
- instance$bounceVoid()
- instance$bounceString(x)
- instance$bounceCharacter(x)
- instance$bounceCharacterVector(x)
- instance$bounceDouble(x)
- instance$bounceNumeric(x)
- instance$bounceNumericVector(x)
- instance$bounceInt(x)
- instance$bounceInteger(x)
- instance$bounceIntegerVector(x)
- instance$bounceFactor(x)
- instance$bounceFactorVector(x)
- instance$bounceDate(x)
- instance$bounceDateVector(x)
- instance$bounceLogical(x)
- instance$bounceLogicalVector(x)
- instance$bounceFile(x)
- instance$bounceDataframe(x)
- instance$bounceList(x)
- instance$bounceNamedList(x)
- instance$bounceArray(x)
- instance$clone()
- instance$print()

**Constructor** `new()`**:** the default no-args constructor

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$BounceTest$new();
```

*Arguments:*

- none

*Returns:* the new R6 BounceTest object

**Method** `bounceNull()`**:** no title

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$BounceTest$new();
instance$bounceNull(x)
```

*Arguments:*

**x** - (java expects a RNull)

*Returns:* RNull:

**Method** `bounceVoid()`**:** no title

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$BounceTest$new();
instance$bounceVoid()
```

*Arguments:*

- none

*Returns:* void:

**Method** `bounceString()`**:** no title

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$BounceTest$new();
instance$bounceString(x)
```

*Arguments:*

**x** - (java expects a String)

*Returns:* String:

**Method** `bounceCharacter()`**:** no title

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$BounceTest$new();
instance$bounceCharacter(x)
```

*Arguments:*

**x** - (java expects a RCharacter)

*Returns:* RCharacter:

**Method** `bounceCharacterVector()`**:** no title

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$BounceTest$new();
instance$bounceCharacterVector(x)
```

*Arguments:*

**x** - (java expects a RCharacterVector)

*Returns:* RCharacterVector:

**Method** `bounceDouble()`: no title

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$BounceTest$new();
instance$bounceDouble(x)
```

*Arguments:*

**x** - (java expects a double)

*Returns:* double:

**Method** `bounceNumeric()`: no title

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$BounceTest$new();
instance$bounceNumeric(x)
```

*Arguments:*

**x** - (java expects a RNumeric)

*Returns:* RNumeric:

**Method** `bounceNumericVector()`: no title

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$BounceTest$new();
instance$bounceNumericVector(x)
```

*Arguments:*

**x** - (java expects a RNumericVector)

*Returns:* RNumericVector:

**Method** `bounceInt()`: no title

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$BounceTest$new();
instance$bounceInt(x)
```

*Arguments:*

**x** - (java expects a int)

*Returns:* int:

**Method** `bounceInteger()`: no title

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$BounceTest$new();
instance$bounceInteger(x)
```

*Arguments:*

**x** - (java expects a RInteger)

*Returns:* RInteger:

**Method** `bounceIntegerVector()`: no title

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$BounceTest$new();
instance$bounceIntegerVector(x)
```

*Arguments:*

**x** - (java expects a RIntegerVector)

*Returns:* RIntegerVector:

**Method** `bounceFactor()`: no title

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$BounceTest$new();
instance$bounceFactor(x)
```

*Arguments:*

**x** - (java expects a RFactor)

*Returns:* RFactor:

**Method** `bounceFactorVector()`: no title

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$BounceTest$new();
instance$bounceFactorVector(x)
```

*Arguments:*

**x** - (java expects a RFactorVector)

*Returns:* RFactorVector:

**Method** `bounceDate()`: no title

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$BounceTest$new();
instance$bounceDate(x)
```

*Arguments:*

**x** - (java expects a RDate)

*Returns:* RDate:

**Method** `bounceDateVector()`: no title

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$BounceTest$new();
instance$bounceDateVector(x)
```

*Arguments:*

**x** - (java expects a RDateVector)

*Returns:* RDateVector:

**Method** `bounceLogical()`: no title

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$BounceTest$new();
instance$bounceLogical(x)
```

*Arguments:*

**x** - (java expects a RLogical)

*Returns:* RLogical:

**Method** `bounceLogicalVector()`: no title

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$BounceTest$new();
instance$bounceLogicalVector(x)
```

*Arguments:*

**x** - (java expects a RLogicalVector)

*Returns:* RLogicalVector:

**Method** `bounceFile()`: no title

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$BounceTest$new();
instance$bounceFile(x)
```

*Arguments:*
**x** - (java expects a RFile)
*Returns:* RFile:

**Method** `bounceDataframe()`: no title
*Usage:*
```
J = testRapi::JavaApi$get()
instance = J$BounceTest$new();
instance$bounceDataframe(x)
```

*Arguments:*
**x** - (java expects a RDataframe)
*Returns:* RDataframe:

**Method** `bounceList()`: no title
*Usage:*
```
J = testRapi::JavaApi$get()
instance = J$BounceTest$new();
instance$bounceList(x)
```

*Arguments:*
**x** - (java expects a RList)
*Returns:* RList:

**Method** `bounceNamedList()`: no title
*Usage:*
```
J = testRapi::JavaApi$get()
instance = J$BounceTest$new();
instance$bounceNamedList(x)
```

*Arguments:*
**x** - (java expects a RNamedList)
*Returns:* RNamedList:

**Method** `bounceArray()`: no title
*Usage:*
```
J = testRapi::JavaApi$get()
instance = J$BounceTest$new();
instance$bounceArray(x)
```

*Arguments:*
**x** - (java expects a RNumericArray)
*Returns:* RNumericArray:

## Examples

```
## ------------------------------------
## Construct new instance of BounceTest
## ------------------------------------
## Not run:
J = testRapi::JavaApi$get()
# appropriate parameter values must be provided
instance = J$BounceTest$new()

## End(Not run)

## ------------------------------------
## Method `BounceTest$bounceNull(x)`
## ------------------------------------
## Not run:
# appropriate parameter values must be provided
instance$bounceNull(x)

## End(Not run)

## ------------------------------------
## Method `BounceTest$bounceVoid()`
## ------------------------------------
## Not run:
# appropriate parameter values must be provided
instance$bounceVoid()

## End(Not run)

## ------------------------------------
## Method `BounceTest$bounceString(x)`
## ------------------------------------
## Not run:
# appropriate parameter values must be provided
instance$bounceString(x)

## End(Not run)

## ------------------------------------
## Method `BounceTest$bounceCharacter(x)`
## ------------------------------------
## Not run:
# appropriate parameter values must be provided
instance$bounceCharacter(x)

## End(Not run)

## ------------------------------------
## Method `BounceTest$bounceCharacterVector(x)`
## ------------------------------------
## Not run:
# appropriate parameter values must be provided
```

```
instance$bounceCharacterVector(x)

## End(Not run)

## ------------------------------------
## Method `BounceTest$bounceDouble(x)`
## ------------------------------------
## Not run:
# appropriate parameter values must be provided
instance$bounceDouble(x)

## End(Not run)

## ------------------------------------
## Method `BounceTest$bounceNumeric(x)`
## ------------------------------------
## Not run:
# appropriate parameter values must be provided
instance$bounceNumeric(x)

## End(Not run)

## ------------------------------------
## Method `BounceTest$bounceNumericVector(x)`
## ------------------------------------
## Not run:
# appropriate parameter values must be provided
instance$bounceNumericVector(x)

## End(Not run)

## ------------------------------------
## Method `BounceTest$bounceInt(x)`
## ------------------------------------
## Not run:
# appropriate parameter values must be provided
instance$bounceInt(x)

## End(Not run)

## ------------------------------------
## Method `BounceTest$bounceInteger(x)`
## ------------------------------------
## Not run:
# appropriate parameter values must be provided
instance$bounceInteger(x)

## End(Not run)

## ------------------------------------
## Method `BounceTest$bounceIntegerVector(x)`
## ------------------------------------
## Not run:
```

```
# appropriate parameter values must be provided
instance$bounceIntegerVector(x)

## End(Not run)

## ------------------------------------
## Method `BounceTest$bounceFactor(x)`
## ------------------------------------
## Not run:
# appropriate parameter values must be provided
instance$bounceFactor(x)

## End(Not run)

## ------------------------------------
## Method `BounceTest$bounceFactorVector(x)`
## ------------------------------------
## Not run:
# appropriate parameter values must be provided
instance$bounceFactorVector(x)

## End(Not run)

## ------------------------------------
## Method `BounceTest$bounceDate(x)`
## ------------------------------------
## Not run:
# appropriate parameter values must be provided
instance$bounceDate(x)

## End(Not run)

## ------------------------------------
## Method `BounceTest$bounceDateVector(x)`
## ------------------------------------
## Not run:
# appropriate parameter values must be provided
instance$bounceDateVector(x)

## End(Not run)

## ------------------------------------
## Method `BounceTest$bounceLogical(x)`
## ------------------------------------
## Not run:
# appropriate parameter values must be provided
instance$bounceLogical(x)

## End(Not run)

## ------------------------------------
## Method `BounceTest$bounceLogicalVector(x)`
## ------------------------------------
```

```
## Not run:
# appropriate parameter values must be provided
instance$bounceLogicalVector(x)

## End(Not run)

## ------------------------------------
## Method `BounceTest$bounceFile(x)`
## ------------------------------------
## Not run:
# appropriate parameter values must be provided
instance$bounceFile(x)

## End(Not run)

## ------------------------------------
## Method `BounceTest$bounceDataframe(x)`
## ------------------------------------
## Not run:
# appropriate parameter values must be provided
instance$bounceDataframe(x)

## End(Not run)

## ------------------------------------
## Method `BounceTest$bounceList(x)`
## ------------------------------------
## Not run:
# appropriate parameter values must be provided
instance$bounceList(x)

## End(Not run)

## ------------------------------------
## Method `BounceTest$bounceNamedList(x)`
## ------------------------------------
## Not run:
# appropriate parameter values must be provided
instance$bounceNamedList(x)

## End(Not run)

## ------------------------------------
## Method `BounceTest$bounceArray(x)`
## ------------------------------------
## Not run:
# appropriate parameter values must be provided
instance$bounceArray(x)

## End(Not run)
```

## collider.feature_test *no title*

### Description

no description

### Usage

```
collider.feature_test(
 message1,
 message2
)
```

### Arguments

| | |
|---|---|
| message1 | message1 - (java expects a RCharacter) |
| message2 | message2 - (java expects a RCharacter) |

### Value

RCharacter:

## collider.more_feature_test *no title*

### Description

no description

### Usage

```
collider.more_feature_test(
 message1,
 message2
)
```

### Arguments

| | |
|---|---|
| message1 | message1 - (java expects a RCharacter) |
| message2 | message2 - (java expects a RCharacter) |

### Value

RCharacter:

### Examples

```
library(testthat)
tmp = collider.more_feature_test('should ','work')
expect_equal(tmp,'more feature test: should work')
```

---

## concat *no title*

---

### Description

no description

### Usage

```
concat(
 message1,
 message2
)
```

### Arguments

| | |
|---|---|
| message1 | message1 - (java expects a RCharacter) |
| message2 | message2 - (java expects a RCharacter) |

### Value

RCharacter:

---

## create *A static object constructor*

---

### Description

no description

### Usage

```
create(
 message1,
 message2
)
```

### Arguments

| | |
|---|---|
| message1 | message1 - the message to be printed - (java expects a RCharacter) |
| message2 | message2 - will be used for toString - (java expects a RCharacter) |

### Value

R6 MoreFeatureTest object: A MoreFeatureTest R6 object

### Examples

```
J = JavaApi$get()
J$MoreFeatureTest$create('Hello,',' World')
```

---

## demo_static *Static methods are also supported.*

---

### Description

These are accessed through the root of the R api, and as a functional interface

### Usage

```
demo_static(
 message
)
```

### Arguments

| | |
|---|---|
| message | message a message - (java expects a String) |

### Value

void:

### Examples

```
J = JavaApi$get()
J$FeatureTest$demoStatic('Ola, el mundo')
demo_static('Bonjour, le monde')
```

## deserialise_dataframe *no title*

### Description

no description

### Usage

```
deserialise_dataframe(
 filename
)
```

### Arguments

| | |
|---|---|
| filename | filename - (java expects a String) |

### Value

RDataframe:

## deserialise_list *no title*

### Description

no description

### Usage

```
deserialise_list(
 filename
)
```

### Arguments

| | |
|---|---|
| filename | filename - (java expects a String) |

### Value

RList:

## deserialise_named_list *no title*

### Description

no description

### Usage

```
deserialise_named_list(
 filename
)
```

### Arguments

| | |
|---|---|
| filename | filename - (java expects a String) |

### Value

RNamedList:

## diamonds *The ggplot2::diamonds dataframe*

### Description

A copy serialised into java, using RObject.writeRDS, saved within the jar file of the package, and exposed here using RObject.readRDS.

### Usage

```
diamonds()
```

### Value

RDataframe: the ggplot2::diamonds dataframe

### Examples

```
dplyr::glimpse( diamonds() )
```

FactoryTest *FactoryTest*

## Description

missing description

Version: 1.1.0

Generated: 2024-05-16T16:22:15.487711497

## Details

no details

## Methods

### Constructors:

- J$FactoryTest$new()

### Class methods:

- instance$generateCharacter()
- instance$generateNumeric()
- instance$generateInteger()
- instance$generateFactor()
- instance$generateLogical()
- instance$generateCharacterVec()
- instance$generateNumericVec()
- instance$generateIntegerVec()
- instance$generateFactorVec()
- instance$generateLogicalVec()
- instance$generateDataframe()
- instance$generateStreamDataframe()
- instance$generateList()
- instance$generateNamedList()
- instance$clone()
- instance$print()

**Constructor** `new()`**:** the default no-args constructor

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$FactoryTest$new();
```

*Arguments:*

- none

*Returns:* the new R6 FactoryTest object

**Method** `generateCharacter()`**:** no title

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$FactoryTest$new();
instance$generateCharacter()
```

*Arguments:*

- none

*Returns:* RCharacter:

**Method** `generateNumeric()`**:** no title

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$FactoryTest$new();
instance$generateNumeric()
```

*Arguments:*

- none

*Returns:* RNumeric:

**Method** `generateInteger()`**:** no title

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$FactoryTest$new();
instance$generateInteger()
```

*Arguments:*

- none

*Returns:* RInteger:

**Method** `generateFactor()`**:** no title

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$FactoryTest$new();
instance$generateFactor()
```

*Arguments:*

- none

*Returns:* RFactor:

**Method** `generateLogical()`**:** no title

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$FactoryTest$new();
instance$generateLogical()
```

*Arguments:*

- none

*Returns:* RLogical:

**Method** `generateCharacterVec()`**:** no title

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$FactoryTest$new();
instance$generateCharacterVec()
```

*Arguments:*

- none

*Returns:* RCharacterVector:

**Method** `generateNumericVec()`**:** no title

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$FactoryTest$new();
instance$generateNumericVec()
```

*Arguments:*

- none

*Returns:* RNumericVector:

**Method** `generateIntegerVec()`**:** no title

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$FactoryTest$new();
instance$generateIntegerVec()
```

*Arguments:*

- none

*Returns:* RIntegerVector:

**Method** `generateFactorVec()`**:** no title

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$FactoryTest$new();
instance$generateFactorVec()
```

*Arguments:*

- none

*Returns:* RFactorVector:

**Method** `generateLogicalVec()`**:** no title

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$FactoryTest$new();
instance$generateLogicalVec()
```

*Arguments:*

- none

*Returns:* RLogicalVector:

**Method** `generateDataframe()`**:** no title

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$FactoryTest$new();
instance$generateDataframe()
```

*Arguments:*

- none

*Returns:* RDataframe:

**Method** `generateStreamDataframe()`**:** no title

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$FactoryTest$new();
instance$generateStreamDataframe()
```

*Arguments:*

- none

*Returns:* RDataframe:

**Method** `generateList()`**:** Lists are much harder to type check than vectors hence RList builder methods throw checked exceptions

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$FactoryTest$new();
instance$generateList()
```

*Arguments:*

- none

*Returns:* RList: a RList containing the supplied Java objects converted into RObjects

**Method** `generateNamedList()`**:** no title

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$FactoryTest$new();
instance$generateNamedList()
```

*Arguments:*

- none

*Returns:* RNamedList:

## Examples

```
## ------------------------------------
## Construct new instance of FactoryTest
## ------------------------------------
## Not run:
J = testRapi::JavaApi$get()
# appropriate parameter values must be provided
instance = J$FactoryTest$new()

## End(Not run)

## ------------------------------------
## Method `FactoryTest$generateCharacter()`
## ------------------------------------
## Not run:
# appropriate parameter values must be provided
instance$generateCharacter()

## End(Not run)

## ------------------------------------
## Method `FactoryTest$generateNumeric()`
## ------------------------------------
## Not run:
# appropriate parameter values must be provided
instance$generateNumeric()

## End(Not run)

## ------------------------------------
```

```
## Method `FactoryTest$generateInteger()`
## ------------------------------------
## Not run:
# appropriate parameter values must be provided
instance$generateInteger()

## End(Not run)

## ------------------------------------
## Method `FactoryTest$generateFactor()`
## ------------------------------------
## Not run:
# appropriate parameter values must be provided
instance$generateFactor()

## End(Not run)

## ------------------------------------
## Method `FactoryTest$generateLogical()`
## ------------------------------------
## Not run:
# appropriate parameter values must be provided
instance$generateLogical()

## End(Not run)

## ------------------------------------
## Method `FactoryTest$generateCharacterVec()`
## ------------------------------------
## Not run:
# appropriate parameter values must be provided
instance$generateCharacterVec()

## End(Not run)

## ------------------------------------
## Method `FactoryTest$generateNumericVec()`
## ------------------------------------
## Not run:
# appropriate parameter values must be provided
instance$generateNumericVec()

## End(Not run)

## ------------------------------------
## Method `FactoryTest$generateIntegerVec()`
## ------------------------------------
## Not run:
# appropriate parameter values must be provided
instance$generateIntegerVec()

## End(Not run)
```

```
## ------------------------------------
## Method `FactoryTest$generateFactorVec()`
## ------------------------------------
## Not run:
# appropriate parameter values must be provided
instance$generateFactorVec()

## End(Not run)

## ------------------------------------
## Method `FactoryTest$generateLogicalVec()`
## ------------------------------------
## Not run:
# appropriate parameter values must be provided
instance$generateLogicalVec()

## End(Not run)

## ------------------------------------
## Method `FactoryTest$generateDataframe()`
## ------------------------------------
## Not run:
# appropriate parameter values must be provided
instance$generateDataframe()

## End(Not run)

## ------------------------------------
## Method `FactoryTest$generateStreamDataframe()`
## ------------------------------------
## Not run:
# appropriate parameter values must be provided
instance$generateStreamDataframe()

## End(Not run)

## ------------------------------------
## Method `FactoryTest$generateList()`
## ------------------------------------
## Not run:
# appropriate parameter values must be provided
instance$generateList()

## End(Not run)

## ------------------------------------
## Method `FactoryTest$generateNamedList()`
## ------------------------------------
## Not run:
# appropriate parameter values must be provided
instance$generateNamedList()

## End(Not run)
```

FeatureTest *FeatureTest*

## Description

A test of the R6 generator templating

Version: 1.1.0

Generated: 2024-05-16T16:22:15.462768704

## Arguments

| | |
|---|---|
| `logMessage` | logMessage - a message which will be logged - (java expects a String) |

## Details

The feature test should allow mathjax in javadoc

$$e = mc^2$$

this is a details comment

## Methods

### Constructors:

- `J$FeatureTest$new(logMessage)`

### Class methods:

- `instance$doHelloWorld()`
- `instance$doSum(a, b)`
- `instance$doSum2(a, b)`
- `instance$getMessage()`
- `instance$fluentSetMessage(message)`
- `instance$factoryMethod(a, b)`
- `instance$objectAsParameter(otherObj)`
- `instance$doSomethingWithDataFrame(dataframe)`
- `instance$generateDataFrame()`
- `instance$errorThrower()`
- `instance$asyncCountdown()`
- `instance$asyncRaceCountdown()`
- `instance$blockingCountdown()`
- `instance$clone()`
- `instance$print()`

**Constructor** `new()`**:** A maximum of one constructor of any signature can be used.
If different constructors are needed then they may be used but not included in the R Api (i.e. not annotated with @RMethod.)
Static factory methods can be used instead.

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$FeatureTest$new(logMessage);
```

*Arguments:*

**logMessage - a message which will be logged** - (java expects a String)

*Returns:* the new R6 FeatureTest object

*Examples:*

```
library(roxygen2)
library(devtools)
library(here)
library(tidyverse)
library(ggplot2)
library(readr)
library(dplyr)
library(tibble)
J = JavaApi$get()
minExample = J$FeatureTest$new('Hello from Java constructor!')
```

**Method** `doHelloWorld()`**:** A hello world function

More detailed description

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$FeatureTest$new(logMessage);
instance$doHelloWorld()
```

*Arguments:*

- none

*Returns:* RCharacter: this java method returns a String

*Examples:*

```
library(roxygen2)
library(devtools)
library(here)
library(tidyverse)
library(ggplot2)
library(readr)
library(dplyr)
library(tibble)
J = JavaApi$get()
minExample = J$FeatureTest$new('Hello, R World!')
minExample$doHelloWorld()
```

**Method** `doSum()`**:** Add some numbers (1).

The doSum function description = it adds two numerics

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$FeatureTest$new(logMessage);
instance$doSum(a, b)
```

*Arguments:*

**a the A parameter, can be NA** - (java expects a RNumeric)

**b the B parameter** - (java expects a RNumeric)

*Returns:* RNumeric: A+B of course, NAs in inputs are converted to null in Java. This catches the resulting NPE in java idiom and returns an explicit NA. This only matters if you care about the difference between NA_real_ and NaN in R.

*Examples:*

```
library(roxygen2)
library(devtools)
library(here)
library(tidyverse)
library(ggplot2)
library(readr)
library(dplyr)
library(tibble)
J = JavaApi$get()
library(testthat)
minExample = J$FeatureTest$new('Hello from Java constructor!')
result = minExample$doSum(2,7.5)
testthat::expect_equal(result,9.5)
```

**Method** `doSum2()`**:** Adds some numbers

Do sum 2 uses native ints rather than RNumerics It should throw an error if given something that cannot be coerced to an integer. This also demonstrated the use of the '@RDefault' annotation

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$FeatureTest$new(logMessage);
instance$doSum2(a, b)
```

*Arguments:*

**a the A parameter** - (java expects a int)

**b the B parameter - (defaulting to '10')** - (java expects a int)

*Returns:* int: A+B of course

**Method** `getMessage()`**:** Get the message

message desciption

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$FeatureTest$new(logMessage);
instance$getMessage()
```

*Arguments:*

- none

*Returns:* RCharacter: The message (previously set by the constructor)

**Method** `fluentSetMessage()`**:** Set a message in a fluent way

A fluent method which updates the message in this object, returning the same object. This is differentiated from factory methods which produce a new instance of the same class by checking to see if the returned Java object is equal to the calling Java object.

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$FeatureTest$new(logMessage);
instance$fluentSetMessage(message)
```

*Arguments:*

**message the message is a string - (defaulting to ‘\"hello\nworld\"‘)** - (java expects a RCharacter)

*Returns:* R6 FeatureTest object: this should return exactly the same R6 object.

**Method** `factoryMethod()`**:** A factory or builder method which constructs an object of another class from some parameters

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$FeatureTest$new(logMessage);
instance$factoryMethod(a, b)
```

*Arguments:*

**a the first parameter** - (java expects a RCharacter)

**b the second parameter - (defaulting to ‘as.character(Sys.Date())‘)** - (java expects a RCharacter)

*Returns:* R6 MoreFeatureTest object: A MoreFeatureTest R6 reference

**Method** `objectAsParameter()`**:** no title

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$FeatureTest$new(logMessage);
instance$objectAsParameter(otherObj)
```

*Arguments:*

**otherObj** - (java expects a MoreFeatureTest)

*Returns:* String:

**Method** `doSomethingWithDataFrame()`**:** Consumes a data frame and logs its length

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$FeatureTest$new(logMessage);
instance$doSomethingWithDataFrame(dataframe)
```

*Arguments:*

**dataframe a dataframe** - (java expects a RDataframe)

*Returns:* void:

**Method** `generateDataFrame()`**:** Creates a basic dataframe and returns it

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$FeatureTest$new(logMessage);
instance$generateDataFrame()
```

*Arguments:*

- none

*Returns:* RDataframe: a daatframe

**Method** `errorThrower()`**:** Throws an error

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$FeatureTest$new(logMessage);
instance$errorThrower()
```

*Arguments:*

- none

*Returns:* void:

**Method** `asyncCountdown()`**:** no title

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$FeatureTest$new(logMessage);
instance$asyncCountdown()
```

*Arguments:*

- none

*Returns:* RCharacter:

**Method** `asyncRaceCountdown()`**:** no title

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$FeatureTest$new(logMessage);
instance$asyncRaceCountdown()
```

*Arguments:*

- none

*Returns:* RCharacter:

**Method** `blockingCountdown()`**:** no title

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$FeatureTest$new(logMessage);
instance$blockingCountdown()
```

*Arguments:*

- none

*Returns:* RCharacter:

## Examples

```
## ------------------------------------
## Construct new instance of FeatureTest
## ------------------------------------
library(roxygen2)
library(devtools)
library(here)
library(tidyverse)
library(ggplot2)
library(readr)
library(dplyr)
library(tibble)
J = JavaApi$get()
minExample = J$FeatureTest$new('Hello from Java constructor!')

## ------------------------------------
## Method `FeatureTest$doHelloWorld()`
## ------------------------------------
library(roxygen2)
library(devtools)
library(here)
library(tidyverse)
library(ggplot2)
library(readr)
library(dplyr)
library(tibble)
J = JavaApi$get()
minExample = J$FeatureTest$new('Hello, R World!')
minExample$doHelloWorld()
```

```
## ------------------------------------
## Method `FeatureTest$doSum(a, b)`
## ------------------------------------
library(roxygen2)
library(devtools)
library(here)
library(tidyverse)
library(ggplot2)
library(readr)
library(dplyr)
library(tibble)
J = JavaApi$get()
library(testthat)
minExample = J$FeatureTest$new('Hello from Java constructor!')
result = minExample$doSum(2,7.5)
testthat::expect_equal(result,9.5)

## ------------------------------------
## Method `FeatureTest$doSum2(a, b)`
## ------------------------------------
## Not run:
# appropriate parameter values must be provided
instance$doSum2(a, b)

## End(Not run)

## ------------------------------------
## Method `FeatureTest$getMessage()`
## ------------------------------------
## Not run:
# appropriate parameter values must be provided
instance$getMessage()

## End(Not run)

## ------------------------------------
## Method `FeatureTest$fluentSetMessage(message)`
## ------------------------------------
## Not run:
# appropriate parameter values must be provided
instance$fluentSetMessage(message)

## End(Not run)

## ------------------------------------
## Method `FeatureTest$factoryMethod(a, b)`
## ------------------------------------
## Not run:
# appropriate parameter values must be provided
instance$factoryMethod(a, b)

## End(Not run)
```

```
## ------------------------------------
## Method `FeatureTest$objectAsParameter(otherObj)`
## ------------------------------------
## Not run:
# appropriate parameter values must be provided
instance$objectAsParameter(otherObj)

## End(Not run)

## ------------------------------------
## Method `FeatureTest$doSomethingWithDataFrame(dataframe)`
## ------------------------------------
## Not run:
# appropriate parameter values must be provided
instance$doSomethingWithDataFrame(dataframe)

## End(Not run)

## ------------------------------------
## Method `FeatureTest$generateDataFrame()`
## ------------------------------------
## Not run:
# appropriate parameter values must be provided
instance$generateDataFrame()

## End(Not run)

## ------------------------------------
## Method `FeatureTest$errorThrower()`
## ------------------------------------
## Not run:
# appropriate parameter values must be provided
instance$errorThrower()

## End(Not run)

## ------------------------------------
## Method `FeatureTest$asyncCountdown()`
## ------------------------------------
## Not run:
# appropriate parameter values must be provided
instance$asyncCountdown()

## End(Not run)

## ------------------------------------
## Method `FeatureTest$asyncRaceCountdown()`
## ------------------------------------
## Not run:
# appropriate parameter values must be provided
instance$asyncRaceCountdown()
```

```
## End(Not run)

## ------------------------------------
## Method `FeatureTest$blockingCountdown()`
## ------------------------------------
## Not run:
# appropriate parameter values must be provided
instance$blockingCountdown()

## End(Not run)
```

---

JavaApi *A Test Library*

---

## Description

Documents the features of the 'r6-generator-maven-plugin' by providing an example of an R package automatically generated from Java code by the plugin. It is not intended to be useful beyond testing, demonstrating and documenting the features of the r6 generator plugin.

Version: 1.1.0

Generated: 2024-05-16T16:22:15.382427434

## Arguments

| | |
|---|---|
| logLevel | optional - the slf4j log level as a string - one of OFF (most specific, no logging), FATAL (most specific, little data), ERROR, WARN, INFO, DEBUG, TRACE (least specific, a lot of data), ALL (least specific, all data) |

## Usage

```
J = testRapi::JavaApi$get(logLevel)
```

## Package initialisation and control

- JavaApi$installDependencies()
- JavaApi$rebuildDependencies()
- JavaApi$versionInformation()
- J = JavaApi$get(logLevel)
- J$changeLogLevel(logLevel)
- J$reconfigureLog(log4jproperties)
- J$printMessages()

### Package classes and static methods

- J$MinimalExample$new()
- J$BounceTest$new()
- J$FeatureTest$new(logMessage)
- J$FeatureTest$demoStatic(message)
- J$FeatureTest$diamonds()
- J$FeatureTest$collider(message1, message2)
- J$FeatureTest$asyncStaticCountdown(label, rtimer)
- J$FeatureTest$asyncFactory()
- J$MoreFeatureTest$new(message1, message2)
- J$MoreFeatureTest$create(message1, message2)
- J$MoreFeatureTest$concat(message1, message2)
- J$MoreFeatureTest$collider(message1, message2)
- J$FactoryTest$new()
- J$Serialiser$new()
- J$Serialiser$serialiseDataframe(dataframe, filename)
- J$Serialiser$deserialiseDataframe(filename)
- J$Serialiser$serialiseList(dataframe, filename)
- J$Serialiser$deserialiseList(filename)
- J$Serialiser$serialiseNamedList(dataframe, filename)
- J$Serialiser$deserialiseNamedList(filename)

### Package initialisation and control

**Package method** `JavaApi$installDependencies()`**:** This package level method checks for, and installs any dependencies needed for the running of the package. It is called automatically on first package load and so in general does not need to be used directly.

*Usage:*

```
testRapi::JavaApi$installDependencies()
```

*Arguments:*

- none

*Returns:* nothing. called for side effects.

**Package method** `JavaApi$rebuildDependencies()`**:** This package level method removes existing dependencies and re-installs dependencies needed for the running of the package. It is called automatically on first package load and so in general does not need to be called.

*Usage:*

```
testRapi::JavaApi$rebuildDependencies()
```

*Arguments:*

- none

*Returns:* nothing. called for side effects.

**Package method** `JavaApi$versionInformation()`**:** This package level method returns debugging version information for the package

*Usage:*

```
testRapi::JavaApi$versionInformation()
```

*Arguments:*

- none

*Returns:* A list containing a set of versioning information about this package.

**Package method** `JavaApi$get()`**:** This is the main entry point for the package and the root of the Java API in this package. All classes defined in the package are made available as items under this root. The JavaApi object manages the communication between R and Java.

*Usage:*

```
J = testRapi::JavaApi$get()
# package classes and functions are nested under the `J` api object.
```

*Arguments:*

**logLevel** The desired verbosity of the package. One of "OFF", "FATAL", "ERROR", "WARN", "INFO", "DEBUG", "TRACE", "ALL".

*Returns:* A R6 testRapi::JavaApi object containing the access point to the objects and functions defined in this package

**Api method** `J$changeLogLevel(logLevel)`**:** Once the package is initialised the log level can be changed to increase the level of messages from the api.

*Usage:*

```
J = testRapi::JavaApi$get()
J$changeLogLevel("DEBUG")
```

*Arguments:*

**logLevel** The desired verbosity of the package. One of "OFF", "FATAL", "ERROR", "WARN", "INFO", "DEBUG", "TRACE", "ALL".

*Returns:* nothing. used for side effects.

**Api method** `J$reconfigureLog(log4jproperties)`**:** Experimental / Advanced use: Once the package is initialised the log configureation can be changed to log to an external file for example.

*Usage:*

```
J = testRapi::JavaApi$get()
prp = fs::path(getwd(),"log4j.properties")
if (fs::file_exists(prp)) {
 J$changeLogLevel(prp)
}
```

*Arguments:*

**log4jproperties** a full path to a log4jproperies file

*Returns:* nothing. used for side effects.

**Api method** `J$printMessages()`**:** Experimental / Internal use: Messages from Java to R are queued and printed after each function call. It is unlikely that any will be not printed so in normal circumstances this function should do nothing.

*Usage:*

```
J = testRapi::JavaApi$get()
J$printMessages()
```

*Arguments:*

- none

*Returns:* nothing. used for side effects.

### Static methods and constructors

**Method** `MinimalExample$new()`**:** the default no-args constructor

*Usage:*

```
J = testRapi::JavaApi$get()
J$MinimalExample$new()
```

*Arguments:*

- none

*Returns:* R6 MinimalExample object:

**Method** `BounceTest$new()`**:** the default no-args constructor

*Usage:*

```
J = testRapi::JavaApi$get()
J$BounceTest$new()
```

*Arguments:*

- none

*Returns:* R6 BounceTest object:

**Method** `FeatureTest$new()`**:** A maximum of one constructor of any signature can be used.
If different constructors are needed then they may be used but not included in the R Api (i.e. not annotated with @RMethod.)
Static factory methods can be used instead.

*Usage:*

```
J = testRapi::JavaApi$get()
J$FeatureTest$new(logMessage)
```

*Arguments:*

**logMessage - a message which will be logged** - (java expects a String)

*Returns:* R6 FeatureTest object:

**Examples:**

```
minExample = J$FeatureTest$new('Hello from Java constructor!')
```

**Method** `FeatureTest$demoStatic()`**:** Static methods are also supported.
These are accessed through the root of the R api, and as a functional interface

*Usage:*

```
J = testRapi::JavaApi$get()
J$FeatureTest$demoStatic(message)
# this method is also exposed as a package function:
testRapi::demo_static(message)
```

*Arguments:*

**message a message** - (java expects a String)

*Returns:* void:

**Examples:**

```
J = JavaApi$get()
J$FeatureTest$demoStatic('Ola, el mundo')
demo_static('Bonjour, le monde')
```

**Method** `FeatureTest$diamonds()`**:** The ggplot2::diamonds dataframe
A copy serialised into java, using RObject.writeRDS, saved within the jar file of the package, and exposed here using RObject.readRDS.

*Usage:*

```
J = testRapi::JavaApi$get()
J$FeatureTest$diamonds()
# this method is also exposed as a package function:
testRapi::diamonds()
```

*Arguments:*

- none

*Returns:* RDataframe: the ggplot2::diamonds dataframe

**Examples:**

```
dplyr::glimpse( diamonds() )
```

**Method** `FeatureTest$collider()`**:** no title

*Usage:*

```
J = testRapi::JavaApi$get()
J$FeatureTest$collider(message1, message2)
# this method is also exposed as a package function:
testRapi::collider.feature_test(message1, message2)
```

*Arguments:*

**message1** - (java expects a RCharacter)

**message2** - (java expects a RCharacter)

*Returns:* RCharacter:

**Method** `FeatureTest$asyncStaticCountdown()`**:** no title

*Usage:*

```
J = testRapi::JavaApi$get()
J$FeatureTest$asyncStaticCountdown(label, rtimer)
# this method is also exposed as a package function:
testRapi::async_static_countdown(label, rtimer)
```

*Arguments:*

**label** - (java expects a RCharacter)

**rtimer** - (java expects a RInteger)

*Returns:* An ‘RFuture‘ with methods ‘cancel()‘, ‘isCancelled()‘, ‘isDone()‘ and ‘get()‘. The result of a ‘get()‘ call will be an RCharacter:

**Method** `FeatureTest$asyncFactory()`**:** no title

*Usage:*

```
J = testRapi::JavaApi$get()
J$FeatureTest$asyncFactory()
# this method is also exposed as a package function:
testRapi::async_factory()
```

*Arguments:*

- none

*Returns:* An ‘RFuture‘ with methods ‘cancel()‘, ‘isCancelled()‘, ‘isDone()‘ and ‘get()‘. The result of a ‘get()‘ call will be an R6 FactoryTest object:

**Method** `MoreFeatureTest$new()`**:** the first constructor is used if there are none annotated

*Usage:*

```
J = testRapi::JavaApi$get()
J$MoreFeatureTest$new(message1, message2)
```

*Arguments:*

**message1 - the message to be printed** - (java expects a RCharacter)

**message2 - will be used for toString** - (java expects a RCharacter)

*Returns:* R6 MoreFeatureTest object:

**Method** `MoreFeatureTest$create()`**:** A static object constructor

*Usage:*

```
J = testRapi::JavaApi$get()
J$MoreFeatureTest$create(message1, message2)
# this method is also exposed as a package function:
testRapi::create(message1, message2)
```

*Arguments:*

**message1 - the message to be printed** - (java expects a RCharacter)

**message2 - will be used for toString** - (java expects a RCharacter)

*Returns:* R6 MoreFeatureTest object: A MoreFeatureTest R6 object

**Examples:**

```
J = JavaApi$get()
J$MoreFeatureTest$create('Hello,' World')
```

**Method** `MoreFeatureTest$concat()`**:** no title

*Usage:*

```
J = testRapi::JavaApi$get()
J$MoreFeatureTest$concat(message1, message2)
# this method is also exposed as a package function:
testRapi::concat(message1, message2)
```

*Arguments:*

**message1** - (java expects a RCharacter)

**message2** - (java expects a RCharacter)

*Returns:* RCharacter:

**Method** `MoreFeatureTest$collider()`**:** no title

*Usage:*

```
J = testRapi::JavaApi$get()
J$MoreFeatureTest$collider(message1, message2)
# this method is also exposed as a package function:
testRapi::collider.more_feature_test(message1, message2)
```

*Arguments:*

**message1** - (java expects a RCharacter)

**message2** - (java expects a RCharacter)

*Returns:* RCharacter:

**Examples:**

```
library(testthat)
tmp = collider.more_feature_test('should ','work')
expect_equal(tmp,'more feature test: should work')
```

**Method** `FactoryTest$new()`**:** the default no-args constructor

*Usage:*

```
J = testRapi::JavaApi$get()
J$FactoryTest$new()
```

*Arguments:*

- none

*Returns:* R6 FactoryTest object:

**Method** `Serialiser$new()`**:** the default no-args constructor

*Usage:*

```
J = testRapi::JavaApi$get()
J$Serialiser$new()
```

*Arguments:*

- none

*Returns:* R6 Serialiser object:

**Method** `Serialiser$serialiseDataframe()`**:** no title

*Usage:*

```
J = testRapi::JavaApi$get()
J$Serialiser$serialiseDataframe(dataframe, filename)
# this method is also exposed as a package function:
testRapi::serialise_dataframe(dataframe, filename)
```

*Arguments:*

**dataframe** - (java expects a RDataframe)

**filename** - (java expects a String)

*Returns:* void:

**Method** `Serialiser$deserialiseDataframe()`**:** no title

*Usage:*

```
J = testRapi::JavaApi$get()
J$Serialiser$deserialiseDataframe(filename)
# this method is also exposed as a package function:
testRapi::deserialise_dataframe(filename)
```

*Arguments:*

**filename** - (java expects a String)

*Returns:* RDataframe:

**Method** `Serialiser$serialiseList()`**:** no title

*Usage:*

```
J = testRapi::JavaApi$get()
J$Serialiser$serialiseList(dataframe, filename)
# this method is also exposed as a package function:
testRapi::serialise_list(dataframe, filename)
```

*Arguments:*

**dataframe** - (java expects a RList)

**filename** - (java expects a String)

*Returns:* void:

**Method** `Serialiser$deserialiseList()`**:** no title

*Usage:*

```
J = testRapi::JavaApi$get()
J$Serialiser$deserialiseList(filename)
# this method is also exposed as a package function:
testRapi::deserialise_list(filename)
```

*Arguments:*

**filename** - (java expects a String)

*Returns:* RList:

**Method** `Serialiser$serialiseNamedList()`**:** no title

*Usage:*

```
J = testRapi::JavaApi$get()
J$Serialiser$serialiseNamedList(dataframe, filename)
# this method is also exposed as a package function:
testRapi::serialise_named_list(dataframe, filename)
```

*Arguments:*

**dataframe** - (java expects a RNamedList)

**filename** - (java expects a String)

*Returns:* void:

**Method** `Serialiser$deserialiseNamedList()`**:** no title

*Usage:*

```
J = testRapi::JavaApi$get()
J$Serialiser$deserialiseNamedList(filename)
# this method is also exposed as a package function:
testRapi::deserialise_named_list(filename)
```

*Arguments:*

**filename** - (java expects a String)

*Returns:* RNamedList:

## Author(s)

<rob@terminological.co.uk>

## Examples

```
## ------------------------------------
## Check library dependencies for testRapi
## ------------------------------------
testRapi::JavaApi$installDependencies()

## ------------------------------------
## Construct a testRapi Java API instance
## ------------------------------------

J = testRapi::JavaApi$get()
# or a more verbose configuration
# J = testRapi::JavaApi$get("DEBUG")


## ------------------------------------
## Method `J$MinimalExample$new(...)`
## ------------------------------------
## Not run:
# no example given - appropriate parameter values must be provided:
J$MinimalExample$new()
# or alternatively:
testRapi::new()

## End(Not run)

## ------------------------------------
## Method `J$BounceTest$new(...)`
## ------------------------------------
## Not run:
# no example given - appropriate parameter values must be provided:
J$BounceTest$new()
# or alternatively:
testRapi::new()
```

```
## End(Not run)

## ------------------------------------
## Method `J$FeatureTest$new(...)`
## ------------------------------------
minExample = J$FeatureTest$new('Hello from Java constructor!')

## ------------------------------------
## Method `J$FeatureTest$demoStatic(...)`
## Aliased as `testRapi::demo_static(...)`
## ------------------------------------
J = JavaApi$get()
J$FeatureTest$demoStatic('Ola, el mundo')
demo_static('Bonjour, le monde')

## ------------------------------------
## Method `J$FeatureTest$diamonds(...)`
## Aliased as `testRapi::diamonds(...)`
## ------------------------------------
dplyr::glimpse( diamonds() )

## ------------------------------------
## Method `J$FeatureTest$collider(...)`
## Aliased as `testRapi::collider.feature_test(...)`
## ------------------------------------
## Not run:
# no example given - appropriate parameter values must be provided:
J$FeatureTest$collider(message1, message2)
# or alternatively:
testRapi::collider.feature_test(message1, message2)

## End(Not run)

## ------------------------------------
## Method `J$FeatureTest$asyncStaticCountdown(...)`
## Aliased as `testRapi::async_static_countdown(...)`
## ------------------------------------
## Not run:
# no example given - appropriate parameter values must be provided:
J$FeatureTest$asyncStaticCountdown(label, rtimer)
# or alternatively:
testRapi::async_static_countdown(label, rtimer)

## End(Not run)

## ------------------------------------
## Method `J$FeatureTest$asyncFactory(...)`
## Aliased as `testRapi::async_factory(...)`
## ------------------------------------
## Not run:
# no example given - appropriate parameter values must be provided:
J$FeatureTest$asyncFactory()
```

```
# or alternatively:
testRapi::async_factory()

## End(Not run)

## ------------------------------------
## Method `J$MoreFeatureTest$new(...)`
## ------------------------------------
## Not run:
# no example given - appropriate parameter values must be provided:
J$MoreFeatureTest$new(message1, message2)
# or alternatively:
testRapi::new(message1, message2)

## End(Not run)

## ------------------------------------
## Method `J$MoreFeatureTest$create(...)`
## Aliased as `testRapi::create(...)`
## ------------------------------------
J = JavaApi$get()
J$MoreFeatureTest$create('Hello,',' World')

## ------------------------------------
## Method `J$MoreFeatureTest$concat(...)`
## Aliased as `testRapi::concat(...)`
## ------------------------------------
## Not run:
# no example given - appropriate parameter values must be provided:
J$MoreFeatureTest$concat(message1, message2)
# or alternatively:
testRapi::concat(message1, message2)

## End(Not run)

## ------------------------------------
## Method `J$MoreFeatureTest$collider(...)`
## Aliased as `testRapi::collider.more_feature_test(...)`
## ------------------------------------
library(testthat)
tmp = collider.more_feature_test('should ','work')
expect_equal(tmp,'more feature test: should work')

## ------------------------------------
## Method `J$FactoryTest$new(...)`
## ------------------------------------
## Not run:
# no example given - appropriate parameter values must be provided:
J$FactoryTest$new()
# or alternatively:
testRapi::new()

## End(Not run)
```

```
## ------------------------------------
## Method `J$Serialiser$new(...)`
## ------------------------------------
## Not run:
# no example given - appropriate parameter values must be provided:
J$Serialiser$new()
# or alternatively:
testRapi::new()

## End(Not run)

## ------------------------------------
## Method `J$Serialiser$serialiseDataframe(...)`
## Aliased as `testRapi::serialise_dataframe(...)`
## ------------------------------------
## Not run:
# no example given - appropriate parameter values must be provided:
J$Serialiser$serialiseDataframe(dataframe, filename)
# or alternatively:
testRapi::serialise_dataframe(dataframe, filename)

## End(Not run)

## ------------------------------------
## Method `J$Serialiser$deserialiseDataframe(...)`
## Aliased as `testRapi::deserialise_dataframe(...)`
## ------------------------------------
## Not run:
# no example given - appropriate parameter values must be provided:
J$Serialiser$deserialiseDataframe(filename)
# or alternatively:
testRapi::deserialise_dataframe(filename)

## End(Not run)

## ------------------------------------
## Method `J$Serialiser$serialiseList(...)`
## Aliased as `testRapi::serialise_list(...)`
## ------------------------------------
## Not run:
# no example given - appropriate parameter values must be provided:
J$Serialiser$serialiseList(dataframe, filename)
# or alternatively:
testRapi::serialise_list(dataframe, filename)

## End(Not run)

## ------------------------------------
## Method `J$Serialiser$deserialiseList(...)`
## Aliased as `testRapi::deserialise_list(...)`
## ------------------------------------
## Not run:
```

```
# no example given - appropriate parameter values must be provided:
J$Serialiser$deserialiseList(filename)
# or alternatively:
testRapi::deserialise_list(filename)

## End(Not run)

## ------------------------------------
## Method `J$Serialiser$serialiseNamedList(...)`
## Aliased as `testRapi::serialise_named_list(...)`
## ------------------------------------
## Not run:
# no example given - appropriate parameter values must be provided:
J$Serialiser$serialiseNamedList(dataframe, filename)
# or alternatively:
testRapi::serialise_named_list(dataframe, filename)

## End(Not run)

## ------------------------------------
## Method `J$Serialiser$deserialiseNamedList(...)`
## Aliased as `testRapi::deserialise_named_list(...)`
## ------------------------------------
## Not run:
# no example given - appropriate parameter values must be provided:
J$Serialiser$deserialiseNamedList(filename)
# or alternatively:
testRapi::deserialise_named_list(filename)

## End(Not run)
```

---

`manualFunction` *A very simple test*

---

## Description

A very simple test

## Usage

```
manualFunction()
```

## Value

a string

`MinimalExample` *MinimalExample*

## Description

This class is a very basic example of the features of the rJava maven plugin.

Version: 1.1.0

Generated: 2024-05-16T16:22:15.444089981

## Details

The class is annotated with an @RClass to identify it as part of the R API.

## Methods

### Constructors:

- `J$MinimalExample$new()`

### Class methods:

- `instance$demo(dataframe, message)`
- `instance$clone()`
- `instance$print()`

**Constructor** `new()`**:** the default no-args constructor

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$MinimalExample$new();
```

*Arguments:*

- none

*Returns:* the new R6 MinimalExample object

**Method** `demo()`**:** Documentation of the method can be done in JavaDoc and these will be present in the R documentation

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$MinimalExample$new();
instance$demo(dataframe, message)
```

*Arguments:*

**dataframe - a dataframe with an arbitrary number of columns** - (java expects a RDataframe)

**message - a message** - (java expects a String)

*Returns:* RDataframe: the dataframe unchanged

*Examples:*

```
J = JavaApi$get()
minExample = J$MinimalExample$new()
minExample$demo(dataframe=tibble::tibble(input=c(1,
2,3)), message='Hello world')
```

## Examples

```
## ------------------------------------
## Construct new instance of MinimalExample
## ------------------------------------
## Not run:
J = testRapi::JavaApi$get()
# appropriate parameter values must be provided
instance = J$MinimalExample$new()

## End(Not run)

## ------------------------------------
## Method `MinimalExample$demo(dataframe, message)`
## ------------------------------------
J = JavaApi$get()
minExample = J$MinimalExample$new()
minExample$demo(dataframe=tibble::tibble(input=c(1,
2,3)), message='Hello world')
```

---

`MoreFeatureTest` *MoreFeatureTest*

---

## Description

This has no documentation

Version: 1.1.0

Generated: 2024-05-16T16:22:15.480444175

## Arguments

| | |
|---|---|
| `message1` | message1 - the message to be printed - (java expects a RCharacter) |
| `message2` | message2 - will be used for toString - (java expects a RCharacter) |

## Details

This has no documentation

## Methods

### Constructors:

- J$MoreFeatureTest$new(message1, message2)

### Class methods:

- instance$testLogging()
- instance$throwCatchable()
- instance$printMessage()
- instance$throwRuntime()
- instance$clone()
- instance$print()

**Constructor** `new()`**:** the first constructor is used if there are none annotated

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$MoreFeatureTest$new(message1, message2);
```

*Arguments:*

**message1 - the message to be printed** - (java expects a RCharacter)

**message2 - will be used for toString** - (java expects a RCharacter)

*Returns:* the new R6 MoreFeatureTest object

**Method** `testLogging()`**:** no title

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$MoreFeatureTest$new(message1, message2);
instance$testLogging()
```

*Arguments:*

- none

*Returns:* void:

**Method** `throwCatchable()`**:** no title

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$MoreFeatureTest$new(message1, message2);
instance$throwCatchable()
```

*Arguments:*

- none

*Returns:* RCharacter:

**Method** `printMessage()`**:** no title

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$MoreFeatureTest$new(message1, message2);
instance$printMessage()
```

*Arguments:*

- none

*Returns:* void:

**Method** `throwRuntime()`**:** no title

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$MoreFeatureTest$new(message1, message2);
instance$throwRuntime()
```

*Arguments:*

- none

*Returns:* RCharacter:

## Examples

```
## ------------------------------------
## Construct new instance of MoreFeatureTest
## ------------------------------------
## Not run:
J = testRapi::JavaApi$get()
# appropriate parameter values must be provided
instance = J$MoreFeatureTest$new(message1, message2)

## End(Not run)

## ------------------------------------
## Method `MoreFeatureTest$testLogging()`
## ------------------------------------
## Not run:
# appropriate parameter values must be provided
instance$testLogging()

## End(Not run)

## ------------------------------------
## Method `MoreFeatureTest$throwCatchable()`
## ------------------------------------
## Not run:
# appropriate parameter values must be provided
instance$throwCatchable()

## End(Not run)
```

```
## ------------------------------------
## Method `MoreFeatureTest$printMessage()`
## ------------------------------------
## Not run:
# appropriate parameter values must be provided
instance$printMessage()

## End(Not run)

## ------------------------------------
## Method `MoreFeatureTest$throwRuntime()`
## ------------------------------------
## Not run:
# appropriate parameter values must be provided
instance$throwRuntime()

## End(Not run)
```

---

`Serialiser` *Serialiser*

---

## Description

Actually useful class which takes an R dataframe and a filename

Version: 1.1.0

Generated: 2024-05-16T16:22:15.497640740

## Details

string and serialises the dataframe so it can be used natively in java for testing purposes.

## Methods

### Constructors:

- J$Serialiser$new()

### Class methods:

- none
- `instance$clone()`
- `instance$print()`

**Constructor** `new()`**:** the default no-args constructor

*Usage:*

```
J = testRapi::JavaApi$get()
instance = J$Serialiser$new();
```

*Arguments:*

- none

*Returns:* the new R6 Serialiser object

## Examples

```
## ------------------------------------
## Construct new instance of Serialiser
## ------------------------------------
## Not run:
J = testRapi::JavaApi$get()
# appropriate parameter values must be provided
instance = J$Serialiser$new()

## End(Not run)
```

---

`serialise_dataframe` *no title*

---

## Description

no description

## Usage

```
serialise_dataframe(
 dataframe,
 filename
)
```

## Arguments

| | |
|---|---|
| `dataframe` | dataframe - (java expects a RDataframe) |
| `filename` | filename - (java expects a String) |

## Value

void:

---

`serialise_list` *no title*

---

## Description

no description

## Usage

```
serialise_list(
 dataframe,
 filename
)
```

### Arguments

| | |
|---|---|
| dataframe | dataframe - (java expects a RList) |
| filename | filename - (java expects a String) |

### Value

void:

---

## serialise_named_list *no title*

---

### Description

no description

### Usage

```
serialise_named_list(
 dataframe,
 filename
)
```

### Arguments

| | |
|---|---|
| dataframe | dataframe - (java expects a RNamedList) |
| filename | filename - (java expects a String) |

### Value

void:

# Index


∗ **java api**
async_factory, 5
async_static_countdown, 6
BounceTest, 6
collider.feature_test, 17
collider.more_feature_test, 17
concat, 18
create, 18
demo_static, 19
deserialise_dataframe, 20
deserialise_list, 20
deserialise_named_list, 21
diamonds, 21
FactoryTest, 22
FeatureTest, 29
JavaApi, 37
MinimalExample, 51
MoreFeatureTest, 52
serialise_dataframe, 56
serialise_list, 56
serialise_named_list, 57
Serialiser, 55
testRapi-package, 2
.background_cancel, 3
.background_cancel_all, 3
.background_get_by_id, 4
.background_status, 4
.background_tidy_up, 5

async_factory, 5
async_static_countdown, 6

BounceTest, 6

collider.feature_test, 17
collider.more_feature_test, 17
concat, 18
create, 18

demo_static, 19
deserialise_dataframe, 20
deserialise_list, 20
deserialise_named_list, 21
diamonds, 21

FactoryTest, 22
FeatureTest, 29

JavaApi, 37

manualFunction, 50
MinimalExample, 51
MoreFeatureTest, 52

serialise_dataframe, 56
serialise_list, 56
serialise_named_list, 57
Serialiser, 55

testRapi *(*testRapi-package*)*, 2
testRapi-package, 2