Package: rmaven (via r-universe)

August 13, 2024

Title Run Java Build Tool Maven from R

Version 0.1.3

Description Distributing large files in packages is contrary to 'CRAN'
policies. Java libraries frequently have large dependencies,
and this makes them hard to submit to CRAN. Java has a
sophisticated dependency management tool (Maven) which can
calculate and cache dependencies. Running Maven from R allows
for better integration of Java libraries into R. Maven provides
various options for resolving complied Java code dependencies,
configuring class path, or compiling Java source code, all of
which are useful for efficient use of Java within R.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

Imports fs, magrittr, rappdirs, rlang, stringr, utils, xml2

Suggests rmarkdown, knitr, rJava

URL <https://terminological.github.io/rmaven>,
<https://github.com/terminological/rmaven>

VignetteBuilder knitr

Language en-US

RoxygenNote 7.2.3

BugReports <https://github.com/terminological/rmaven/issues>

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

RemoteUrl https://github.com/terminological/rmaven

RemoteRef 0.1.3

RemoteSha 2bed4e880170761a1e7306a3afe04f60631d52bf

Contents

---

as.coordinates   *Maven coordinates*

Description

Maven coordinates

Usage

```
as.coordinates(groupId, artifactId, version, ...)
```

Arguments

groupId	the maven groupId
artifactId	the maven artifactId
version	the maven version
...	other parameters ignored apart from packaging (one of jar,war,pom or ejb) and classifier (one of tests, client, sources, javadoc, jar-with-dependencies, or src)

Value

a coordinates object containing the Maven artifact coordinates

Examples

```
as.coordinates("org.junit.jupiter","junit-jupiter-api","4.13.2")
```

---

clear_rmaven_cache *Clear out the* rmaven *cache*

Description

Deletes all content in the rmaven cache. This should not be necessary, but never say never, and if there is really a problem with the cache, then deleting it may be the best thing. This will wait for confirmation from the user. If running unattended the options("rmaven.allow.cache.delete"=TRUE) must be set for the action to occur, otherwise it will generate a warning and do nothing.

Usage

```
clear_rmaven_cache()
```

Value

nothing, called for side effects

Examples

```
# need to set the following option to allow cache to be deleted in non
# interactive session
opts = options("rmaven.allow.cache.delete"=TRUE)
clear_rmaven_cache()
options(opts)
```

---

compile_jar *Compile and package Java code*

Description

Compilation will package the Java source code in to a Jar file for further use. It will resolve dependencies and optionally package them into a single uber jar (using maven assembly).

Usage

```
compile_jar(
  path,
  nocache = FALSE,
  verbose = c("normal", "quiet", "debug"),
  with_dependencies = FALSE,
  ...
)
```

Arguments

`path`	the path to - either a java source code directory containing a `pom.xml` file, the `pom.xml` file itself, or a `...-src.jar` assembled by the maven assembly plugin,
`nocache`	normally compilation is only performed if the input has changed. nocache forces recompilation
`verbose`	how much output from maven, one of "normal", "quiet", "debug"
`with_dependencies`	
	compile the Java code to a '...-jar-with-dependencies.jar' including transitive dependencies which may be easier to embed into R code as does not need a class path (however may be large if there are a lot of dependencies)
`...`	passed to `execute_maven(...)`, e.g. could include `settings` parameter

Value

the path to the compiled 'jar' file. If this is a fat jar this can be passed straight to `rJava`, otherwise an additional `resolve_dependencies(...)` call is required

Examples

```
# This code can take quite a while to run as has to
# download a lot of plugins, especially on first run
path = package_jars("rmaven","src")
compile_jar(path,nocache=TRUE)
path2 = system.file("testdata/test-project",package = "rmaven")
compile_jar(path2,nocache=TRUE,with_dependencies=TRUE)
```

---

copy_artifact *Copy an artifact from a repository to a local directory*

Description

This essentially runs a `maven-dependency-plugin:copy` goal to copy a JAR file from (usually) a remote repository to a local directory. The directory is under the users control but defaults to the `.m2` repository.

Usage

```
copy_artifact(
  groupId = NULL,
  artifactId = NULL,
  version = NULL,
  ...,
  coordinates = NULL,
  artifact = NULL,
  outputDirectory = .working_dir(artifact),
```

```
    repoUrl = .default_repos(),
    nocache = FALSE,
    verbose = c("normal", "quiet", "debug")
)
```

Arguments

`groupId`	optional, the maven `groupId`,
`artifactId`	optional, the maven `artifactId`,
`version`	optional, the maven version,
`...`	other maven coordinates such as classifier or packaging
`coordinates`	optional, coordinates as a coordinates object,
`artifact`	optional, coordinates as an artifact string `groupId:artifactId:version[:packaging[:classifier]]` string
`outputDirectory`	
	optional path, defaults to the `rmaven` cache directory
`repoUrl`	the URLs of the repositories to check (defaults to maven central, `Sonatype snaphots` and `jitpack`)
`nocache`	normally artifacts are only fetched if required, `nocache` forces fetching
`verbose`	how much output from maven, one of "normal", "quiet", "debug"

Value

the output of the system2 call. 0 on success.

Examples

```
# This code can take quite a while to run as has to
# download a lot of plugins, especially on first run
tmp = copy_artifact("org.junit.jupiter","junit-jupiter-api","5.9.0")
print(tmp)
```

---

developer_mode	*Use the default Maven repository location*

Description

It should be clear that running this function will affect files in the users filesystem.

Usage

```
developer_mode()
```

Details

The default Maven directory is located in user space and writing to it is forbidden by CRAN policies. This plugin is set up to use a cache directory for the local Maven repository but if you are developing Java code and using it in R then your Java build tools will be installing content to the default Maven directory, and not the CRAN sanctioned cache. Thus rmaven wont by default be able to pick up jar files locally installed through standard Java tooling. This function sets the rmaven repository location to the Maven standard location allowing local jar files to be used. Obviously this is not portable until the Java packages are deployed to a Maven repository and is only an useful option during development.

Value

nothing. called for side effects

Examples

```
# set the repository location to the usual location for Java development
# developer_mode()

# We don't run this above example as it creates an empty directory in the
# userspace and doing so in an example violates CRAN principles.
```

---

execute_maven   *Executes a maven goal*

Description

Maven goals are defined either as life-cycle goals (e.g. "clean", "compile") or as plugin goals (e.g. "help:system"). Some Maven goals may be executed without a pom.xml file, others require one. Some maven goals (e.g. compilation) require the use of a JDK.

Usage

```
execute_maven(
  goal,
  opts = c(),
  pom_path = NULL,
  quiet = .quietly(verbose),
  debug = .debug(verbose),
  verbose = c("normal", "debug", "quiet"),
  require_jdk = FALSE,
  settings = .settings_path(),
  ...
)
```

Arguments

`goal`	the goal of the `mvn` command (can be multiple) e.g. `c("clean","compile")`
`opts`	provided options in the form `c("-Doption1=value2","-Doption2=value2")`
`pom_path`	optional. the path to a `pom.xml` file for goals that need one.
`quiet`	should output from maven be suppressed? (`-q` flag)
`debug`	should output from maven be verbose? (`-X` flag)
`verbose`	how much output from maven, one of "normal", "quiet", "debug"
`require_jdk`	does the goal you are executing require a JDK (e.g. compilation does, fetching artifacts and calculating class path does not)
`settings`	the path to a `settings.xml` file controlling Maven. The default is a configuration with a local repository in the `rmaven` cache directory (and not the Java maven repository).
`...`	non-empty named parameters are passed to maven as options in the form `-Dname=value`

Value

nothing, invisibly

Examples

```
# This code can take quite a while to run as has to
# download a lot of plugins, especially on first run on a clean system
execute_maven("help:system")
```

`fetch_artifact`	*Fetch an artifact, and dependencies, into the local* `.m2` *repository*

Description

This can be used to get a JAR file from the maven repositories into a local `.m2` repository. The local path is made available for importing it into the `rJava` classpath for example.

Usage

```
fetch_artifact(
  groupId = NULL,
  artifactId = NULL,
  version = NULL,
  ...,
  coordinates = NULL,
  artifact = NULL,
  repoUrl = .default_repos(),
  nocache = FALSE,
  verbose = c("normal", "quiet", "debug")
)
```

Arguments

`groupId`	optional, the maven `groupId`,
`artifactId`	optional, the maven `artifactId`,
`version`	optional, the maven version,
`...`	other maven coordinates such as classifier or packaging
`coordinates`	optional, but if not supplied `groupId` and `artifactId` must be, coordinates as a coordinates object (see `as.coordinates()`)
`artifact`	optional, coordinates as an artifact string `groupId:artifactId:version[:packaging[:classifier]]` string
`repoUrl`	the URLs of the repositories to check (defaults to Maven central, 'Sonatype' snapshots and 'jitpack', defined in `options("rmaven.default_repos")`)
`nocache`	normally artifacts are only fetched if required, nocache forces fetching
`verbose`	how much output from maven, one of "normal", "quiet", "debug"

Value

the path of the artifact within the local maven cache

Examples

```
# This code can take quite a while to run as has to
# download a lot of plugins, especially on first run
fetch_artifact(artifact="com.google.guava:guava:31.1-jre")
fetch_artifact(coordinates = as.coordinates("org.junit.jupiter",
  "junit-jupiter-api","5.9.0"))
```

---

get_repository_location

Get the location of the Maven repository

Description

In general this function is mainly for internal use but maybe handy for debugging. The maven repository location can be defined by `set_repository_location(...)` or through the option `options("rmaven.m2.repository"=...)` option but defaults to a `.m2/repository` directory in the `rmaven` cache directory. This is not the default location for Maven when used from Java writing to the default Maven directory in user space is forbidden by CRAN policies. The result of this is that `rmaven` will have to unnecessarily download additional copies of Java libraries, onto the users computer and cannot re-use already cached copies. Also Maven wont be able to pick up jar files locally installed through standard Java tooling, unless the default CRAN approved location is overridden using `set_repository_location("~/.m2/repository")`. This is more of an issue for developers rather than users.

Usage

```
get_repository_location(settings_path = .settings_path())
```

Arguments

settings_path   the file path of the settings.xml to update (generally the supplied default is what you want to use)

Value

the location of the maven repository

Examples

```
# the default location:
get_repository_location()
# change the location to the Java default. This change will not persist between sessions.
opt = options("rmaven.m2.repository"=paste0(tempdir(),"/.m2/repository/"))
set_repository_location()
get_repository_location()
# revert to rmaven defaults
options(opt)
set_repository_location()
```

---

package_jars   *Find location of some or all of the jars in a particular package.*

Description

Find location of some or all of the jars in a particular package.

Usage

```
package_jars(
  package_name,
  types = c("all", "thin-jar", "fat-jar", "shaded", "src")
)
```

Arguments

package_name   the R package name

types   the jar types to look for in the package: one of all,thin-jar,fat-jar,shaded,src

Value

a vector of paths to jar files in the package

Examples

```
package_jars("rmaven")
package_jars("rmaven","thin-jar")
```

---

print.coordinates   *Prints a coordinates object*

Description

Prints a coordinates object

Usage

```
## S3 method for class 'coordinates'
print(x, ...)
```

Arguments

x	a maven coordinates object
...	ignored

Value

nothing. for side effects.

Examples

```
print(as.coordinates("org.junit.jupiter","junit-jupiter-api","4.13.2"))
```

---

resolve_dependencies   *Resolve dependencies and calculate the* classpath *for an artifact.*

Description

This function makes sure the transitive dependencies for a maven artifact are available locally in the
.m2 maven cache and calculates a local classpath which can be provided to rJava. The artifact may
be specified either as a set of maven coordinates (in which case the artifact itself is also downloaded,
and included in the classpath) or as a path to a jar file containing a pom.xml (e.g. a compiled jar
file, a compiled ...-jar-with-dependencies, or a assembled ...-src.jar).

Usage

```
resolve_dependencies(
  groupId = NULL,
  artifactId = NULL,
  version = NULL,
  ...,
  coordinates = NULL,
  artifact = NULL,
  path = NULL,
  include_self = NULL,
  nocache = FALSE,
  verbose = c("normal", "quiet", "debug")
)
```

Arguments

groupId	the maven `groupId`, optional
artifactId	the maven `artifactId`, optional
version	the maven version, optional
...	passed on to as.coordinates()
coordinates	the maven coordinates, optional (either `groupId`,`artifactId` and 'version' must be specified, or 'coordinates', or 'artifact')
artifact	optional, coordinates as an artifact string `groupId:artifactId:version[:packaging[:classifier]]` string
path	the path to the source directory, pom file or jar file. if not given rmaven will get the artifact from the maven central repositories
include_self	do you want include this path in the `classpath`. optional, if missing the path will be included if it is a regular jar, or a fat jar, otherwise not.
nocache	do not used cached version, by default we use a cached version of the `classpath` unless the `pom.xml` is newer that the cached `classpath`.
verbose	how much output from maven, one of "normal", "quiet", "debug"

Value

a character vector of the `classpath` jar files (including the current one if appropriate)

Examples

```
# This code can take quite a while to run as has
# to download a lot of plugins, especially on first run

# classpath would be cached if possible
resolve_dependencies(groupId = "commons-io", artifactId = "commons-io",
  version="2.11.0")

# forcing download and classpath calculation of an artifact
resolve_dependencies(artifact = "org.junit.jupiter:junit-jupiter-api:5.9.0",
```

```
  nocache=TRUE)

# find the test jar in this package and calculate its stated dependencies
resolve_dependencies(path=
  system.file("testdata/test-project-0.0.1-SNAPSHOT.jar",package="rmaven"))

# find the test source code jar in this package and calculate its stated
# dependencies
resolve_dependencies(path=
  system.file("testdata/test-project-0.0.1-SNAPSHOT-src.jar",
  package="rmaven")
)
```

---

set_repository_location

Sets the local maven repository location

Description

This writes a maven repository location to a temporary `settings.xml` file which persists only for
the R session. The location of the maven repository is either specified here, or can be defined by
the `options("rmaven.m2.repository"=...)` option. If neither of these is provided, the location
will revert to a default location within the `rmaven` cache. (Approved by CRAN for a local cache
location) e.g. on 'Linux' this will default to `~/.cache/rmaven/.m2/repository/`

Usage

```
set_repository_location(
  repository_location = getOption("rmaven.m2.repository", default = .working_dir(subpath
    = ".m2/repository/")),
   settings_path = .settings_path()
)
```

Arguments

repository_location

a file path (which will be expanded to a full path) where the repository should
be based, e.g. `~/.m2/repository/`. Defaults to a sub-directory of the `rmaven`
cache.

settings_path   the file path of the settings.xml to update (generally the supplied default is what
you want to use)

Value

the expanded path of the new repository location

Examples

```
# Setting the repository to be a temp dir as an example:
set_repository_location(paste0(tempdir(), "/.m2/repository"))
# you would never want to do this in real life as then the maven repository
# would be rebuilt on every new R session.

# set the repository location to the usual location for Java development
# set_repository_location("~/.m2/repository")
# We don't run this above example as it creates an empty directory in the
# userspace and doing so in an example violates CRAN principles.

# set the repository location back to the CRAN approved default location
set_repository_location()
```

---

start_jvm   *Start an* rJava *JVM* *with or without debugging options*

Description

This does not do anything if the JVM has already been started. Otherwise starts the JVM via rJava with a set of options Additional JVM options (beyond debugging) can be set with the options("java.parameters"=c("-Xprof","-Xrunhprof"))

Usage

```
start_jvm(
  debug = FALSE,
  quiet = getOption("rmaven.quiet", TRUE),
  max_heap = NULL,
  thread_stack = NULL,
  ...
)
```

Arguments

debug	turn on debugging
quiet	don't report messages (defaults to getOption("rmaven.quiet") or TRUE)
max_heap	optional. if a string like "2048m" the -Xmx option value to start the JVM - if a string like "75%" the -XX:MaxRAMPercentage, if a numeric - number of megabytes.
thread_stack	optional. sensible values range from '1m' to '128m' (max is '1g'). Can be important with deeply nested structures.
...	any other named parameters are passed as -name=value

Value

nothing - called for side effects

Examples

```
start_jvm()
## Not run:
# this may try to rebind debugging port
start_jvm(debug = TRUE)

## End(Not run)
```

Index