Package 'binman'

October 2, 2020

Title A Binary Download Manager

Version 0.1.2

Description Tools and functions for managing the download of binary files. Binary repositories are defined in 'YAML' format. Defining new pre-download, download and post-download templates allow additional repositories to be added.

Depends R (>= 3.3)

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Suggests testthat, covr, knitr, rmarkdown

Imports rappdirs, yaml, httr, jsonlite, xml2, utils, stats, assertthat, semver

URL https://docs.ropensci.org/binman/, https://github.com/ropensci/binman

BugReports https://github.com/ropensci/binman/issues

RoxygenNote 7.1.1

VignetteBuilder knitr

NeedsCompilation no

Author John Harrison [aut] (original author), Ju Yeong Kim [cre] (rOpenSci maintainer)

Maintainer Ju Yeong Kim <jkim2345@fredhutch.org>

Repository CRAN

Date/Publication 2020-10-02 21:22:05 UTC

R topics documented:

`app_dir` *Get application directory*

Description

Get application directory

Usage

```
app_dir(appname, check = TRUE)
```

Arguments

`appname`	A character string giving the name of the application
`check`	check whether the app given by appname exists or not.

Value

A character string giving the path of the directory

Examples

```
## Not run:
appdir <- app_dir("superduperapp", FALSE)

## End(Not run)
```

assign_directory *Assign directory*

Description

Assign directory to download list

Usage

```
assign_directory(dllist, appname)
```

Arguments

dllist	A named list of data.frames. The name indicates the platform. The data.frame should contain the version, url and file to be processed.
appname	Name to give the app

Value

A named list of data.frames. The data.frame should contain the version, url and file to be processed, the directory to download the file to and whether the file already exists.

Examples

```
## Not run:
tdata <- system.file("testdata", "test_dllist.Rdata", package = "binman")
load(tdata)
assign_directory(test_dllist, "myapp")

## End(Not run)
```

binman *binman*

Description

A Binary Download Manager.

Details

Tools and functions for managing the download of binary files. Binary repositories are defined in 'YAML' format. Defining new pre-download, download and post-download templates allow additional repositories to be added.

download_files *Download binaries*

Description

Download binaries from repository

Usage

```
download_files(dllist, overwrite = FALSE)
```

Arguments

dllist	A named list of data.frames. The data.frame should contain the version, url and file to be processed, the directory to download the file to and whether the file already exists.
overwrite	Overwrite existing binaries. Default value of FALSE

Value

A data.frame indicating whether a file was downloaded for a platform.

Examples

```
## Not run:
trdata <- system.file("testdata", "test_dlres.Rdata", package = "binman")
tldata <- system.file("testdata", "test_dllist.Rdata", package = "binman")
load(trdata)
load(tldata)
dllist <- assign_directory(test_dllist, "myapp")
testthat::with_mock(
  `httr::GET` = function(...) {
    test_llres
  },
  `base::dir.create` = function(...) {
    TRUE
  },
  dlfiles <- download_files(dllist)
)

## End(Not run)
```

list_versions *List app versions*

Description

List app versions by platform

Usage

```
list_versions(appname, platform = c("ALL"))
```

Arguments

appname	A character string giving the name of the application
platform	A character vector of platforms to list. Defaults to "ALL"

Value

A list of platforms with version directories

Examples

```
## Not run:
appdir <- app_dir("superduperapp", FALSE)
platforms <- LETTERS[1:4]
versions <- LETTERS[5:7]
mkdirs <- file.path(appdir, outer(platforms, versions, file.path))
chk <- vapply(mkdirs, dir.create, logical(1), recursive = TRUE)
expect_true(all(chk))
res <- list_versions("superduperapp")
unlink(appdir, recursive = TRUE)

## End(Not run)
```

noproc_dlfiles *Do not post process*

Description

Do not post process dlfiles

Usage

```
noproc_dlfiles(dlfiles)
```

Arguments

dlfiles	A data.frame of files by platform and indicating whether they were processed

Value

Returns a list of character vectors indicating files processed

Examples

```
## Not run:
ymlfile <- system.file("exdata", "sampleapp4.yml", package = "binman")
trdata <- system.file("testdata", "test_dlres.Rdata", package = "binman")
load(trdata)
testthat::with_mock(
  `httr::GET` = function(...) {
    test_llres
  },
  `base::dir.create` = function(...) {
    TRUE
  },
  procyml <- process_yaml(ymlfile)
)
procyml

## End(Not run)
```

---

predl_bitbucket_downloads

Pre download bitbucket downloads

Description

Pre download bitbucket downloads template function

Usage

```
predl_bitbucket_downloads(
  url,
  platform,
  history,
  appname,
  platformregex = platform,
  versionregex = "\\d+(?:\\.\\d+)+"
)
```

Arguments

url	A url giving the bitbucket download JSON for a project. As an example https://bitbucket.org/ariya/phantor the phantomjs project has an asset JSON available at https://api.bitbucket.org/2.0/repositories/ariya/phanto
platform	A character vector of platform names
history	The maximum number of files to get for a platform
appname	Name of the app
platformregex	A filter for platforms. Defaults to the platform
versionregex	A regex for retrieving the version.

Value

A named list of data.frames. The name indicates the platform. The data.frame should contain the version, url and file to be processed. Used as input for `download_files` or an equivalent.

Examples

```
## Not run:
bbdata <- system.file("testdata", "test_bitbucketdl.json",
  package = "binman"
)
platform <- c("linux64", "linux32", "windows", "macosx")
platformregex <- c("linux-x86_64", "linux-i686", "windows", "macosx")
bbdllist <-
  predl_bitbucket_downloads(
    url = bbdata, platform, history = 3L,
    appname = "binman_chromedriver",
    platformregex
  )

## End(Not run)
```

---

predl_github_assets *Pre download Github assets*

Description

Pre download Github assets template function

Usage

```
predl_github_assets(
  url,
  platform,
  history,
  appname,
  platformregex = platform,
  versionregex = c("", "")
)
```

Arguments

url	A url giving the github asset JSON for a project. As an example https://github.com/mozilla/geckodriver/rel the geckodriver project has an asset JSON available at https://api.github.com/repos/mozilla/geckodriver/re
platform	A character vector of platform names
history	The maximum number of files to get for a platform
appname	Name of the app
platformregex	A filter for platforms. Defaults to the platform
versionregex	A regex for retrieving the version.

Value

A named list of data.frames. The name indicates the platform. The data.frame should contain the version, url and file to be processed. Used as input for `download_files` or an equivalent.

Examples

```
## Not run:
gadata <- system.file("testdata", "test_gitassets.json",
  package = "binman"
)
platform <- c("linux64", "win64", "macos")
gadllist <- predl_github_assets(
  url = gadata, platform, history = 3L,
  appname = "binman_chromedriver"
)

## End(Not run)
```

---

predl_google_storage *Pre-Download Google Storage*

Description

Pre-Download Google Storage template function

Usage

```
predl_google_storage(
  url,
  platform,
  history,
  appname,
  fileregex = "\\.zip$",
  platformregex = platform,
  versionregex = c(paste0("(.*)/.*", fileregex), "\\1")
)
```

Arguments

url	A url giving the JSON bucket listings for a project. For example: http://chromedriver.storage.googleapis.c lists the chromedriver files but https://www.googleapis.com/storage/v1/b/chromedriver/o/ is the JSON listings for the project.
platform	A character vector of platform names
history	The maximum number of files to get for a platform
appname	Name of the app
fileregex	A filter for files
platformregex	A filter for platforms. Defaults to the platform names.
versionregex	A regex for retrieving the version.

Value

A named list of data.frames. The name indicates the platform. The data.frame should contain the version, url and file to be processed. Used as input for download_files or an equivalent.

Examples

```
## Not run:
gsdata <- system.file("testdata", "test_googstor.json",
  package = "binman"
)
platform <- c("linux64", "win32", "mac64")
gsdllist <- predl_google_storage(
  url = gsdata, platform, history = 5L,
  appname = "binman_chromedriver"
)

## End(Not run)
```

---

process_yaml *Process a yaml file*

Description

Process a yaml file. The file defines the pre-download function, the download function and the post download function.

Usage

```
process_yaml(ymlfile, verbose = TRUE)
```

Arguments

ymlfile	A file in a YAML format defining the pre-download/ download and post download functions together with their arguments.
verbose	If TRUE, include status messages (if any)

Value

A list of files processed (downloaded and post processed)

Examples

```
## Not run:
ymlfile <- system.file("exdata", "sampleapp.yml", package = "binman")
trdata <- system.file("testdata", "test_dlres.Rdata", package = "binman")
load(trdata)
testthat::with_mock(
  `httr::GET` = function(...) {
    test_llres
  },
  `base::dir.create` = function(...) {
    TRUE
  },
  `utils::unzip` = function(zipfile, ...) {
    zipfile
  },
  procyml <- process_yaml(ymlfile)
)
procyml

## End(Not run)
```

---

rm_platform *Remove application platform*

Description

Remove application files/directories for a given platform

Usage

```
rm_platform(appname, platform = c("ALL"))
```

Arguments

appname	A character string giving the name of the application
platform	A character vector indicating the platform to remove. Defaults to "ALL"

Value

Returns a logical vector indicating whether the removal of platform was successful. Return is invisible.

Examples

```
## Not run:
appdir <- app_dir(appname, FALSE)
platforms <- LETTERS[1:4]
versions <- LETTERS[5:7]
mkdirs <- file.path(appdir, outer(platforms, versions, file.path))
chk <- vapply(mkdirs, dir.create, logical(1), recursive = TRUE)
appver <- list_versions(appname)
rm_platform(appname, platforms[2:3])
unlink(appdir, recursive = TRUE)

## End(Not run)
```

`rm_version` *Remove application version*

Description

Remove application version for a given platform

Usage

```
rm_version(appname, platform, version = c("ALL"))
```

Arguments

`appname`	A character string giving the name of the application
`platform`	A character string indicating the platform.
`version`	A character vector of versions to remove. Defaults to "ALL"

Value

Returns a logical vector indicating whether the removal of version was successful. Return is invisible.

Examples

```
## Not run:
appdir <- app_dir(appname, FALSE)
platforms <- LETTERS[1:4]
versions <- LETTERS[5:7]
mkdirs <- file.path(appdir, outer(platforms, versions, file.path))
chk <- vapply(mkdirs, dir.create, logical(1), recursive = TRUE)
appver <- list_versions(appname)
rm_version(appname, platforms[2], versions[1:2])
unlink(appdir, recursive = TRUE)

## End(Not run)
```

unziptar_dlfiles *Unzip/Untar downloaded files*

Description

Unzip/Untar downloaded files. Keeps the original zip file

Usage

```
unziptar_dlfiles(dlfiles, chmod = FALSE)
```

Arguments

dlfiles	A data.frame of files by platform and indicating whether they were processed
chmod	change the mode of the unarchived file/files to "755" so they are executable on unix like systems.

Value

Returns a list of character vectors indicating files processed

Examples

```
## Not run:
ymlfile <- system.file("exdata", "sampleapp.yml", package = "binman")
trdata <- system.file("testdata", "test_dlres.Rdata", package = "binman")
load(trdata)
testthat::with_mock(
  `httr::GET` = function(...) {
    test_llres
  },
  `base::dir.create` = function(...) {
    TRUE
  },
  `utils::unzip` = function(zipfile, ...) {
    zipfile
  },
  procyml <- process_yaml(ymlfile)
)
procyml

## End(Not run)
```

Index