Package 'wdman'

September 1, 2022

Type Package

Title 'Webdriver'/'Selenium' Binary Manager

Version 0.2.6

Description There are a number of binary files associated with the 'Webdriver'/'Selenium' project. This package provides functions to download these binaries and to manage processes involving them.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 3.2)

Suggests testthat, covr, knitr, rmarkdown

Imports binman, assertthat, processx, yaml, semver(>= 0.2.0), utils

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

URLNote https://github.com/ropensci/wdman

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

RoxygenNote 7.2.1

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 2022-09-01 05:10:03 UTC

R topics documented:

`chrome` *Start chrome driver*

Description

Start chrome driver

Usage

```
chrome(
  port = 4567L,
  version = "latest",
  path = "wd/hub",
  check = TRUE,
  verbose = TRUE,
  retcommand = FALSE,
  ...
)
```

Arguments

`port`	Port to run on
`version`	what version of chromedriver to run. Default = "latest" which runs the most recent version. To see other version currently sourced run binman::list_versions("chromedriver")
`path`	base URL path prefix for commands, e.g. wd/hub
`check`	If TRUE check the versions of chromedriver available. If new versions are available they will be downloaded.
`verbose`	If TRUE, include status messages (if any)
`retcommand`	If TRUE return only the command that would be passed to `process`
`...`	pass additional options to the driver

Value

Returns a list with named elements `process`, `output`, `error`, `stop`, and `log`. `process` is the object from calling `process`. `output` and `error` are the functions reading the latest messages from "stdout" and "stderr" since the last call whereas `log` is the function that reads all messages. Lastly, `stop` call the `kill` method in `process` to the kill the `process`.

Examples

```
## Not run:
cDrv <- chrome()
cDrv$output()
cDrv$stop()

## End(Not run)
```

---

gecko *Start gecko driver*

Description

Start gecko driver

Usage

```
gecko(
  port = 4567L,
  version = "latest",
  check = TRUE,
  loglevel = c("info", "fatal", "error", "warn", "config", "debug", "trace"),
  verbose = TRUE,
  retcommand = FALSE,
  ...
)
```

Arguments

port	Port to run on
version	what version of geckodriver to run. Default = "latest" which runs the most recent version. To see other version currently sourced run binman::list_versions("geckodriver")
check	If TRUE check the versions of geckodriver available. If new versions are available they will be downloaded.
loglevel	Set Gecko log level [values: fatal, error, warn, info, config, debug, trace]
verbose	If TRUE, include status messages (if any)
retcommand	If TRUE return only the command that would be passed to process
...	pass additional options to the driver

Value

Returns a list with named elements process, output, error, stop, and log. process is the object from calling process. output and error are the functions reading the latest messages from "stdout" and "stderr" since the last call whereas log is the function that reads all messages. Lastly, stop call the kill method in process to the kill the process.

Examples

```
## Not run:
gDrv <- gecko()
gDrv$output()
gDrv$stop()

## End(Not run)
```

`iedriver` *Start IE driver server*

Description

Start IE driver server

Usage

```
iedriver(
  port = 4567L,
  version = "latest",
  check = TRUE,
  loglevel = c("FATAL", "TRACE", "DEBUG", "INFO", "WARN", "ERROR"),
  verbose = TRUE,
  retcommand = FALSE,
  ...
)
```

Arguments

`port`	Port to run on
`version`	what version of IE driver server to run. Default = "latest" which runs the most recent version. To see other version currently sourced run binman::list_versions("iedriverserver")
`check`	If TRUE check the versions of IE driver available. If new versions are available they will be downloaded.
`loglevel`	Specifies the log level used by the server. Valid values are: TRACE, DEBUG, INFO, WARN, ERROR, and FATAL. Defaults to FATAL if not specified.
`verbose`	If TRUE, include status messages (if any)
`retcommand`	If TRUE return only the command that would be passed to `process`
`...`	pass additional options to the driver

Value

Returns a list with named elements `process`, `output`, `error`, `stop`, and `log`. `process` is the object from calling `process`. `output` and `error` are the functions reading the latest messages from "stdout" and "stderr" since the last call whereas `log` is the function that reads all messages. Lastly, `stop` call the `kill` method in `process` to the kill the `process`.

Examples

```
## Not run:
ieDrv <- iedriver()
ieDrv$output()
ieDrv$stop()

## End(Not run)
```

`phantomjs` *Start phantomjs*

Description

Start phantomjs in webdriver mode

Usage

```
phantomjs(
  port = 4567L,
  version = "2.1.1",
  check = TRUE,
  loglevel = c("INFO", "ERROR", "WARN", "DEBUG"),
  verbose = TRUE,
  retcommand = FALSE,
  ...
)
```

Arguments

`port`	Port to run on
`version`	what version of phantomjs to run. Default = "2.2.1" which runs the most recent stable version. To see other version currently sourced run binman::list_versions("phantomjs")
`check`	If TRUE check the versions of phantomjs available. If new versions are available they will be downloaded.
`loglevel`	Set phantomjs log level [values: fatal, error, warn, info, config, debug, trace]
`verbose`	If TRUE, include status messages (if any)
`retcommand`	If TRUE return only the command that would be passed to `process`
`...`	pass additional options to the driver

Value

Returns a list with named elements `process`, `output`, `error`, `stop`, and `log`. `process` is the object from calling `process`. `output` and `error` are the functions reading the latest messages from "stdout" and "stderr" since the last call whereas `log` is the function that reads all messages. Lastly, `stop` call the `kill` method in `process` to the kill the `process`.

Examples

```
## Not run:
pjs <- phantomjs()
pjs$output()
pjs$stop()

## End(Not run)
```

`selenium` *Start Selenium Server*

Description

Start Selenium Server

Usage

```
selenium(
  port = 4567L,
  version = "latest",
  chromever = "latest",
  geckover = "latest",
  iedrver = NULL,
  phantomver = "2.1.1",
  check = TRUE,
  verbose = TRUE,
  retcommand = FALSE,
  ...
)
```

Arguments

`port`	Port to run on
`version`	what version of Selenium Server to run. Default = "latest" which runs the most recent version. To see other version currently sourced run binman::list_versions("seleniumserver")
`chromever`	what version of Chrome driver to run. Default = "latest" which runs the most recent version. To see other version currently sourced run binman::list_versions("chromedriver"), A value of NULL excludes adding the chrome browser to Selenium Server.
`geckover`	what version of Gecko driver to run. Default = "latest" which runs the most recent version. To see other version currently sourced run binman::list_versions("geckodriver"), A value of NULL excludes adding the firefox browser to Selenium Server.
`iedrver`	what version of IEDriverServer to run. Default = "latest" which runs the most recent version. To see other version currently sourced run binman::list_versions("iedriverserver"), A value of NULL excludes adding the internet explorer browser to Selenium Server. NOTE this functionality is Windows OS only.

phantomver	what version of PhantomJS to run. Default = "2.2.1" which runs the most recent stable version. To see other version currently sourced run binman::list_versions("phantomjs"), A value of NULL excludes adding the PhantomJS headless browser to Selenium Server.
check	If TRUE check the versions of selenium available and the versions of associated drivers (chromever, geckover, phantomver, iedrver). If new versions are available they will be downloaded.
verbose	If TRUE, include status messages (if any)
retcommand	If TRUE return only the command that would be passed to process
...	pass additional options to the driver

Value

Returns a list with named elements process, output, error, stop, and log. process is the object from calling process. output and error are the functions reading the latest messages from "stdout" and "stderr" since the last call whereas log is the function that reads all messages. Lastly, stop call the kill method in process to the kill the process.

Examples

```
## Not run:
selServ <- selenium()
selServ$output()
selServ$stop()

## End(Not run)
```

---

wdman *wdman*

Description

Webdriver/Selenium Binary Manager

Details

There are a number of binary files associated with the Webdriver/Selenium project. This package provides functions to download these binaries and to manage processes involving them.

Index