Package ‘servr’

October 23, 2018

Type Package
Title A Simple HTTP Server to Serve Static Files or Dynamic Documents
Version 0.11
Maintainer Yihui Xie <xie@yihui.name>
Description Start an HTTP server in R to serve static files, or dynamic
documents that can be converted to HTML files (e.g., R Markdown) under a
given directory.
Depends R (>= 3.0.0)
Imports mime (>= 0.2), httpuv (>= 1.4.0), xfun, jsonlite
Suggests tools, later, rstudioapi, knitr (>= 1.9), rmarkdown
License GPL
URL https://github.com/yihui/servr
BugReports https://github.com/yihui/servr/issues
RoxygenNote 6.1.0
NeedsCompilation no
Author Yihui Xie [aut, cre] (<https://orcid.org/0000-0003-0645-5666>),
Carson Sievert [ctb],
Jesse Anderson [ctb],
Ramnath Vaidyanathan [ctb]
Repository CRAN
Date/Publication 2018-10-23 17:00:02 UTC

R topics documented:

browse_last ................................................................. 2
daemon_stop ............................................................... 2
http ................................................................. 3
jekyll ............................................................... 4
make ............................................................. 6
server_config ............................................................ 7
serve_example ............................................................ 8
vign ............................................................. 8
**daemon_stop**

Utilities for daemonized servers

Description

The server functions in this package will return server handles. You can pass the handles to `daemon_stop()` to stop the daemonized servers.

Usage

```r
daemon_stop(which = daemon_list())
daemon_list()
```

Arguments

- `which` the server handles returned by server functions; by default, all existing handles in the current R session obtained from `daemon_list()`, i.e., all daemon servers will be stopped by default

Value

The function `daemon_list()` returns a list of existing server handles, and `daemon_stop()` returns an invisible NULL.

**browse_last**

Reopen the last browsed page

Description

If you have launched a page in the browser via `servr` but closed it later, you may call this function to reopen it.

Usage

```r
browse_last()
```

Examples

```r
servr::browse_last()
```
httd

Serve static files under a directory

Description

If there is an `index.html` under this directory, it will be displayed; otherwise the list of files is displayed, with links on their names. After we run this function, we can go to `http://localhost:port` to browse the web pages either created from R or read from HTML files.

Usage

```r
httd(dir = ".", ...) 
httw(dir = ".", watch = ".", pattern = NULL, all_files = FALSE, handler = NULL, ...)
```

Arguments

dir
The root directory to serve.

... 
server configurations passed to `server_config()`

watch 
a directory under which `httw()` is to watch for changes; if it is a relative path, it is relative to the `dir` argument

pattern 
a regular expression passed to `list.files()` to determine the files to watch

all_files 
whether to watch all files including the hidden files

handler 
a function to be called every time any files are changed or added under the directory; its argument is a character vector of the filenames of the files modified or added

Details

`httd()` is a pure static server, and `httw()` is similar but watches for changes under the directory: if an HTML file is being viewed in the browser, and any files are modified under the directory, the HTML page will be automatically refreshed.

References

https://github.com/yihui/servr

Examples

```r
# see https://github.com/yihui/servr for command line usage
# or run inside an R session
if (interactive()) servr::httd()
```
jekyll  Serve R Markdown based websites

Description

R Markdown documents (with the filename extension `.Rmd`) are re-compiled using knitr or rmarkdown when necessary (source files are newer than output files), and the HTML pages will be automatically refreshed in the web browser accordingly.

Usage

jekyll(dir = ".", input = c("." , "_source", "_posts"), output = c("." , "_posts", "_posts"), script = c("Makefile", "build.R"), serve = TRUE, command = "jekyll build", ...)

rmdv2(dir = ".", script = c("Makefile", "build.R"), in_session = FALSE, ...)

rmdv1(dir = ".", script = c("Makefile", "build.R"), in_session = FALSE, ...)

Arguments

dir the root directory of the website

input the input directories that contain R Markdown documents (the directories must be relative instead of absolute; same for output directories)

output the output directories corresponding to input; for an input document ‘foo.Rmd’ under the directory input[i], its output document ‘foo.md’ (or ‘foo.html’) is generated under output[i] if the output document is older than the input document

script a Makefile (see make), or (if Makefile not found) the name of an R script to re-build R Markdown documents, which will be executed via command line of the form Rscript build.R arg1 arg2 where build.R is the script specified by this argument, arg1 is the input filename, and arg2 is the output filename; inside the R script, you can use commandArgs(TRUE) to capture c(arg1, arg2), e.g. knitr::knit(commandArgs(TRUE)[1], commandArgs(TRUE)[2]); if this R script is not found, either, internal compiling methods will be used, which are basically knit(), knitr2html(), or render()

serve whether to serve the website; if FALSE, the R Markdown documents and the website will be compiled but not served

command a command to build the Jekyll website; by default, it is jekyll build, and you can use alternative commands, such as bundle exec jekyll build

... server configurations passed to server_config()
in_session whether to render the R Markdown documents in the current R session (TRUE) or in a separate new R session (FALSE); if the former, the argument script can be a function with two arguments, the filenames of the source document and the output document, respectively; an internal function (basically `rmarkdown::render()` or `knitr::knit2html()`) will be used if the script argument is not a function and `in_session = TRUE`

Details

The function `jekyll()` sets up a web server to serve a Jekyll-based website. A connection is established between R and the HTML pages through WebSockets so that R can notify the HTML pages to refresh themselves if any R Markdown documents have been re-compiled.

The functions `rmdv1()` and `rmdv2()` are similar to `jekyll()`, and the only difference is the way to compile R Markdown documents: `rmdv1()` uses the `markdown` package (a.k.a R Markdown v1) via `knit2html()`, and `rmdv2()` calls `render()` in the `rmarkdown` package (a.k.a R Markdown v2).

Note

Apparently `jekyll()` and `rmdv1()` require the `knitr` package, and `rmdv2()` requires `rmarkdown`. You have to install them before calling the server functions here.

All R Markdown documents are compiled in separate R sessions by default. If you have any R Markdown documents that should not be compiled as standalone documents (e.g. child documents), you can use different filename extensions, such as `.Rmarkdown`.

The `baseurl` argument does not work in `jekyll()`, and the base URL setting will be read from `_config.yml` (the ‘baseurl’ field) of the website if present. You should not pass `baseurl` to the function `jekyll()` directly.

For the sake of reproducibility, you are recommended to compile each source document in a separate R session (i.e., use the default `in_session = FALSE`) to make sure they can compile on their own, otherwise the current workspace may affect the evaluation of the code chunks in these source documents. Sometimes it might be useful to compile a document in the current R session. For example, if reading data is time-consuming and it is not convenient to cache it (using the `knitr` chunk option `cache = TRUE`), you may read the data once, temporarily turn off the evaluation of that code chunk, and keep on working on the rest of code chunks so that data will not be read over and over again.

References


See Also

The `blogdown` package (based on Hugo and R Markdown v2) is a better alternative to Jekyll: [https://github.com/rstudio/blogdown/](https://github.com/rstudio/blogdown/). I strongly recommend you to try it.
make

Serve files under a directory based on GNU Make

Description

You can define how and when to rebuild files (such as R Markdown files) using Make rules, e.g. a rule _posts/%.md: _source/%.Rmd with a command to build `Rmd` to `md` will be executed if and only if `foo.Rmd` is newer than `foo.md`. The exit status of the command `make -q` will decide whether to rebuild files: rebuilding occurs only when the exit code is not 0. When an HTML file has been rebuilt, it will be automatically refreshed in the web browser.

Usage

make(dir = ".", ...)

Arguments

dir The root directory to serve.

... server configurations passed to server_config()

Note

You must have installed GNU Make to use this function. This is normally not a problem for Linux and OS X users (it should be available by default). For Windows users, you can either install GNU Make, or just install Rtools, which also contains GNU Make.

Examples

# some built-in examples (if you are not familiar with make, you can take a look  # at the Makefile of each example)
servr::serve_example("make1", servr::make)
servr::serve_example("make2", servr::make)
server_config

Description

The server functions in this package are configured through this function.

Usage

server_config(dir = ".", host = "127.0.0.1", port, browser, daemon, interval = 1,
baseurl = "", initpath = "")

Arguments

dir
 host
 port
 browser
 daemon
 interval
 baseurl
 initpath

Value

A list of configuration information of the form list(host, port, start_server = function(app) {}, ...).
serve_example

A convenience function to serve examples in this package

Description

Use server functions to serve built-in examples of this package.

Usage

serve_example(name, FUN, ..., run = interactive())

Arguments

name
the directory name of the example under the directory system.file('examples', package = 'servr')

FUN
a server function that takes the example path as its first argument, e.g. htd, or rmdv1

... other arguments passed to FUN

run whether to run the example (this is mainly for R CMD check purposes: the examples will not be really served when the R session is not interactive, so they will not block R CMD check)

Value

NULL if run = FALSE, otherwise the value returned from FUN().

Examples

# R Markdown v1 or v2
servr::serve_example("rmd", servr::rmdv1)
servr::serve_example("rmd", servr::rmdv2)

# GNU Make
servr::serve_example("make1", servr::make)
servr::serve_example("make2", servr::make)

vign

Serve R Markdown/HTML package vignettes

Description

Serve package vignettes under the 'vignettes/' directory. Because the HTML output files should not be included in the source package, this function renders R Markdown/HTML vignettes, displays them in the web browser, and deletes the HTML output files. You will see the HTML output when you click the links on the ".Rmd" or ".Rhtml" files (unlike the static HTTP server, the compiled output instead of the source document is displayed).
vign

Usage

vign(dir = "vignettes", ...)

Arguments

dir The root directory to serve.
...

server configurations passed to server_config()

Details

When developing R packages, you may want to preview your vignettes once in a while. You can
certainly click the button in RStudio to do it, but that requires you to install the package and rebuild
the vignettes. With this function, your vignette will be rebuilt automatically when you update the
source document. Moreover, because the compilation takes place in the current R session, you can
take advantage of devtools::load_all() (which has a keyboard shortcut in the RStudio IDE) to
reload your package and see the updated vignette in the web browser.

Note

You are supposed to call this function from the root directory of your package. If that is not the
case, you should provide the correct path to the ‘vignettes/’ directory of your package to the dir
argument.
Index

browse_last, 2
commandargs, 4, 7
daemon_list (daemon_stop), 2
daemon_stop, 2
httd, 3, 8
httw (httd), 3
interactive, 7
jekyll, 4
knit, 4
knit2html, 4, 5
list.files, 3
make, 4, 6
render, 4, 5
rmdv1, 8
rmdv1 (jekyll), 4
rmdv2 (jekyll), 4
serve_example, 8
server_config, 3, 4, 6, 7, 9
vign, 8