Package: shinycoreci (via r-universe)

September 10, 2024

Title CI Testing for shiny

Version 0.0.0.9009

Description Core shiny team tools to facilitate testing of the shiny-verse.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Imports jsonlite, progress, callr, sessioninfo, rprojroot, rstudioapi (>= 0.11), withr, rappdirs, cli

Suggests renv, httpuv, rmarkdown (>= 2.9), rsconnect (>= 1.0.1), testthat, httr, tibble, shiny, english, dplyr, tidyr, later, shinytest, waldo

Config/Needs/website tidyverse/tidytemplate

URL https://github.com/rstudio/shinycoreci

BugReports https://github.com/rstudio/shinycoreci/issues

Repository https://posit-dev-shinycoreci.r-universe.dev

RemoteUrl https://github.com/rstudio/shinycoreci

RemoteRef HEAD

RemoteSha 5f640dd5d11d06cf17e64587666f27319d2da8bc

Contents

`accept_snaps` *Accept all snapshot changes*

Description

Accepts **all** snapshot changes for every shiny application in `inst/apps`.

Usage

```
accept_snaps(repo_dir = rprojroot::find_package_root_file())
```

Arguments

`repo_dir`	Root repo folder path

`assert_test_output` *Assert* `test_in_local()` *output*

Description

Method called when `test_in_local()` is called with `assert = TRUE`.

Usage

```
assert_test_output(output)
```

Arguments

`output`	value received from `test_in_local()`

clear_cache *Clear cache for test chart and package info*

Description

Clear cache for test chart and package info

Usage

```
clear_cache()
```

connect_set_public *Make Connect Shiny applications publically available*

Description

The corresponding `rsconnect` account should already exist before calling `connect_set_public`. This can be done by calling `rsconnect::connectApiUser` to add the appropriate account information.

Usage

```
connect_set_public(
  apps = apps_shiny,
  account = "barret",
  server = "beta.rstudioconnect.com"
)

connect_urls(
  apps = apps_deploy,
  account = "barret",
  server = "beta.rstudioconnect.com"
)
```

Arguments

`apps`	A character vector of fully defined shiny application folders
`account, server`	args supplied to `[rsconnect::deployApp]`

Functions

- `connect_set_public()`: Set all the Shiny apps to be public on a Connect server
- `connect_urls()`: Retrieve the urls from a Connect server using the Shiny applications provided in `dir`

Examples

```
## Not run:
  rsconnect::addConnectServer(url = 'https://SERVER.com/API', name = 'CustomName')
  rsconnect::connectApiUser('barret', 'CustomName', apiKey = 'SuperSecretKey')
  deploy_apps(account = 'barret', server = 'CustomName')
  connect_set_public(account = 'barret', server = 'CustomName')
  urls <- connect_urls(account = 'barret', server = 'CustomName')

## End(Not run)
```

`deploy_apps` *Deploy apps to a server*

Description

Run this in the terminal (not RStudio IDE) as it has issues when installing some packages.

Usage

```
deploy_apps(
  apps = apps_deploy,
  account = "testing-apps",
  server = "shinyapps.io",
  ...,
  local_pkgs = FALSE,
  extra_packages = NULL,
  cores = 1,
  retry = 2,
  retrying_ = FALSE
)
```

Arguments

`apps`	A character vector of fully defined shiny application folders
`account, server`	args supplied to `[rsconnect::deployApp]`
`...`	ignored
`local_pkgs`	If `TRUE`, local packages will be used instead of the isolated shinyverse installation.
`extra_packages`	A character vector of extra packages to install
`cores`	number of cores to use when deploying
`retry`	If `TRUE`, try failure apps again. (Only happens once.)
`retrying_`	For internal use only

Details

Installation will use default libpaths.

docker_clean *Clean up docker files*

Description

Clean up docker files

Usage

```
docker_clean(stopped_containers = TRUE, untagged_images = TRUE)
```

Arguments

stopped_containers	boolean that determines if all stopped containers should be removed
untagged_images	boolean that determines of all untagged images should be removed

fix_snaps *Fix all* _snaps *files and merge all* gha- *branches*

Description

This method will apply patches from corresponding GitHub branches for each R and Operating System combination. Changes will not be committed or pushed back to GitHub. The user will need to perform this action manually.

Usage

```
fix_snaps(
  sha = git_sha(repo_dir),
  ...,
  ask_apps = FALSE,
  ask_branches = TRUE,
  ask_if_not_main = TRUE,
  repo_dir = rprojroot::find_package_root_file()
)
```

Arguments

sha	git sha of base branch to look for
...	Extra arguments passed to shinytest::viewTestDiff
ask_apps, ask_branches	Logical which allows for particular apps or branches are to be inspected
ask_if_not_main	Logical which will check if main is the base branch
repo_dir	Root repo folder path

Details

Note: This function will NOT fix `shinyjster` failures.

Outline of steps performed:

1. Validate the current git branch is `main`
2. Validate there are no git changes or untracked files in the current base branch
3. Validate there are `.new` _snaps files
4. Create patch files for each corresponding `gha-` branch in `./patches`
5. Ask which branches should be applied. Filter patch files accordingly
6. Ask which app changes should be kept
7. Apply patch files
8. Call `accept_snaps()` on all app folders
9. Undo changes to app that were not selected
10. Prompt user to commit and push changes back to GitHub

`github_user` *Retrieve default GitHub username*

Description

Equivalent to the terminal code: `git config github.user`

Usage

```
github_user()
```

`platform` *Execution platform*

Description

Execution platform

Usage

```
platform()
```

Value

one of `c("win", "mac", "linux")`

platform_rversion *Platform and R Version*

Description

Platform and R Version

Usage

```
platform_rversion(platform_val = platform(), r_version = r_version_short())
```

Arguments

platform_val	See platform()
r_version	See r_version_short()

r_version_short *R Version*

Description

R Version

Usage

```
r_version_short()
```

Value

Major and minor number only. Ex: `"4.0"`

save_test_results *View (and save)* test_in_local() *output*

Description

Use view_test_results() to preview test results generated by save_test_results() (saving should only ever be done in a GHA workflow).

Usage

```
save_test_results(
  output,
  gha_branch_name,
  pr_number,
  username,
  repo_dir = rprojroot::find_package_root_file()
)
```

Arguments

output	output from test_in_local().
gha_branch_name	an identifier determined by the GHA workflow
pr_number	the pull request number.
username	the username of the github profile.
repo_dir	Location of local shinycoreci repo

---

shinycoreci_libpath *Shinyverse libpath*

Description

Methods to get and reset the shinyverse libpath.

Usage

```
shinycoreci_libpath()

shinycoreci_clean_libpaths()
```

Functions

- shinycoreci_libpath(): Library path that will persist across installations. But will have a different path for different R versions
- shinycoreci_clean_libpaths(): Removes the cached R library

`test_in_browser` *Test apps within the terminal*

Description

Automatically runs the next app in a fresh callr::r_bg session. To stop, close the shiny application window.

Usage

```
test_in_browser(
  app_name = apps[1],
  apps = apps_manual,
  ...,
  port = 8080,
  port_background = NULL,
  host = "127.0.0.1",
  local_pkgs = FALSE
)
```

Arguments

`app_name`	app number or name to start with. If numeric, it will match the leading number in the testing application
`apps`	List of apps to test
`...`	ignored
`port`	`port` for the foreground app process
`port_background`	`port` for the background app process
`host`	`host` for the foreground and background app processes
`local_pkgs`	If `TRUE`, local packages will be used instead of the isolated shinyverse installation.

Examples

```
## Not run:
test_in_browser()

## End(Not run)
```

test_in_connect *Test deployed apps*

Description

Opens an app on the hosted server and runs silbing apps in an iframe.

Usage

```
test_in_connect(type = c("manual", "all"))

test_in_shinyappsio(type = c("manual", "all"))
```

Arguments

type	Type of apps to test. "manual" (default) will only contain apps that should be manually tested. "all" will contain all apps that have been deployed. This is every app except for 141-radiant.

Functions

- test_in_connect(): Test deployed applications on RStudio Connect
- test_in_shinyappsio(): Test connect applications given the server and account

Examples

```
## Not run: test_in_connect()
## Not run: test_in_test_in_shinyapps_io()
```

test_in_ide *Test apps within RStudio IDE*

Description

Automatically runs the next app in a fresh RStudio session after closing the current app. To stop, send an interrupt signal (esc or ctrl+c) to the app twice in rapid succession.

Usage

```
test_in_ide(
  app_name = apps[1],
  apps = apps_manual,
  ...,
  port = 8000,
  host = "127.0.0.1",
  delay = 1,
```

```
  local_pkgs = FALSE,
  viewer = NULL,
  refresh_ = FALSE
)
```

Arguments

app_name	app number or name to start with. If numeric, it will match the leading number in the testing application
apps	List of apps to test
...	ignored
port	port for the foreground app process
host	host for the foreground and background app processes
delay	Time to wait between applications. [1]
local_pkgs	If TRUE, local packages will be used instead of the isolated shinyverse installation.
viewer	RStudio IDE viewer to use. ["pane"]
refresh_	For internal use. If TRUE, packages will not be reinstalled.

Details

Kill testing by hitting esc in RStudio.

If options() need to be set, set them in your

.Rprofile

file. See usethis::edit_r_profile()

Examples

```
## Not run:
test_in_ide(dir = "apps")

## End(Not run)
```

---

test_in_local *Test apps using* shiny::runTests() *using local libpath*

Description

Test apps using shiny::runTests() using local libpath

Usage

```
test_in_local(
  apps = apps_with_tests(repo_dir),
  ...,
  assert = TRUE,
  timeout = 10 * 60,
  retries = 2,
  repo_dir = rprojroot::find_package_root_file(),
  local_pkgs = FALSE
)
```

Arguments

apps	applications within dir to run
...	ignored
assert	logical value which will determine if assert_test_output() will be called on the result
timeout	Length of time allowed for an application's full test suit can run before determining it is a failure
retries	number of attempts to retry before declaring the test a failure
repo_dir	Location of local shinycoreci repo
local_pkgs	If TRUE, local packages will be used instead of the isolated shinyverse installation.

---

test_in_sso *Test Apps in SSO/SSP*

Description

Automatically launches docker in a background process. Once the docker is ready, a shiny application will be launched to help move through the applications.

Usage

```
test_in_sso(
  app_name = apps[1],
  apps = apps_manual,
  ...,
  user = github_user(),
  release = c("jammy", "focal", "centos7"),
  r_version = c("4.3", "4.2", "4.1", "4.0", "3.6"),
  tag = NULL,
  port = 8080,
  port_background = switch(release, centos7 = 7878, 3838),
  host = "127.0.0.1"
```

```
)

test_in_ssp(
  app_name = apps[1],
  apps = apps_manual,
  ...,
  license_file = NULL,
  user = github_user(),
  release = c("jammy", "focal", "centos7"),
  r_version = c("4.3", "4.2", "4.1", "4.0", "3.6"),
  tag = NULL,
  port = 8080,
  port_background = switch(release, centos7 = 8989, 4949),
  host = "127.0.0.1"
)
```

Arguments

app_name	app number or name to start with. If numeric, it will match the leading number in the testing application
apps	List of apps to test
...	ignored
user	GitHub username. Ex: schloerke. Uses github_user by default
release	Distro release name, such as "focal" for ubuntu or "7" for centos
r_version	R version to use. Ex: "3.6"
tag	Extra tag information for the docker image. This will prepend a - if a value is given.
port	Port for local shiny application
port_background	Port to connect to the Docker container
host	host for the foreground and background app processes
license_file	Path to a SSP license file

Details

The docker application will stop when the shiny application exits.

Functions

- test_in_sso(): Test SSO Shiny applications
- test_in_ssp(): Test SSP Shiny applications

Examples

```
## Not run: test_in_sso()
## Not run: test_in_ssp()
```

`trigger` *Generate a repository event.*

Description

This function uses the GitHub API to create a repository dispatch event that can trigger workflows.

Usage

```
trigger(
  event_type,
  repo = "rstudio/shinycoreci",
  client_payload = list(),
  auth_token = Sys.getenv("GITHUB_PAT")
)

trigger_tests(
  repo = "rstudio/shinycoreci",
  auth_token = Sys.getenv("GITHUB_PAT")
)

trigger_deploy(
  repo = "rstudio/shinycoreci",
  auth_token = Sys.getenv("GITHUB_PAT")
)

trigger_docker(
  repo = "rstudio/shinycoreci",
  auth_token = Sys.getenv("GITHUB_PAT")
)

trigger_results(
  repo = "rstudio/shinycoreci",
  auth_token = Sys.getenv("GITHUB_PAT")
)
```

Arguments

`event_type`	The name of the event to create on the repository
`repo`	The GitHub repo to create the event on; defaults to rstudio/shinycoreci
`client_payload`	The JSON object to make available in the workflow as the `github.event.client_payload` object
`auth_token`	Your GitHub **P**ersonal **A**ccess **T**oken; defaults to `Sys.getenv("GITHUB_PAT")`

use_manual_app *Flag an app to be manually tested*

Description

All apps_* methods inspect each application to determine if if testing is possible.

Usage

```
use_manual_app(app_dir)
```

Arguments

app_dir	Shiny application directory containing an app.R, ui.R, server.R, or index.Rmd

use_shinyjster *Create Shinyjster test file*

Description

This creates a testing file that will test shinyjster on each applicable browser.

Usage

```
use_shinyjster(app_dir)
```

Arguments

app_dir	Location of shiny application to test

view_test_images *View Shinytest Images*

Description

View Shinytest Images

Usage

```
view_test_images(repo_dir = rprojroot::find_package_root_file())
```

Arguments

repo_dir	directory to the shinycoreci repo

`write_sysinfo` *Write system information to a file*

Description

Write system information to a file

Usage

```
write_sysinfo(file = stdout(), libpath = resolve_libpath())
```

Arguments

`file`	Name of file, or file object to write to (defaults to stdout).
`libpath`	Library path to find installed packages.

Index