Package: golem (via r-universe)

March 25, 2025

Title A Framework for Robust Shiny Applications

Version 0.5.1

Description An opinionated framework for building a production-ready 'Shiny' application. This package contains a series of tools for building a robust 'Shiny' application from start to finish.

License MIT + file LICENSE

URL https://thinkr-open.github.io/golem/, https://github.com/ThinkR-open/golem

BugReports https://github.com/ThinkR-open/golem/issues

Depends R (>= 3.0)

Imports attempt (>= 0.3.0), config, here, htmltools, rlang (>= 1.0.0), shiny (>= 1.5.0), utils, yaml

Suggests attachment (>= 0.3.2), callr, cli (>= 2.0.0), covr, crayon, desc, devtools, dockerfiler (>= 0.2.0), fs, httpuv, knitr, mockery, pkgbuild, pkgdown, pkgload (>= 1.3.0), processx, purrr, rcmdcheck, remotes, renv, rmarkdown, roxygen2, rsconnect, rstudioapi, sass, spelling, stringr, testthat (>= 3.0.0), tools, usethis (>= 1.6.0), withr

VignetteBuilder knitr

Config/testthat/edition 3

Config/testthat/parallel true

Encoding UTF-8

Language en-US

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Config/pak/sysreqs make zlib1g-dev

Repository https://thinkr-open.r-universe.dev

RemoteUrl https://github.com/ThinkR-open/golem

RemoteRef HEAD

RemoteSha 629179665d933deaaec2df8c34ddd5b5e21235d5

Contents

activate_js *Interact with JavaScript built-in Functions*

Description

activate_js is used to insert directly some JavaScript functions in your golem. By default bundle_ressources() load these function automatically for you.

Usage

```
activate_js()

invoke_js(fun, ..., session = shiny::getDefaultReactiveDomain())
```

Arguments

fun	JS function to be invoked.
...	JSON-like messages to be sent to the triggered JS function
session	The shiny session within which to call sendCustomMessage.

show Show an element with the jQuery selector provided. Example: golem::invoke_js("show", "#id");.

hide Hide an element with the jQuery selector provided. Example: golem::invoke_js("hide", "#id").

showid Show an element with the id provided. Example: golem::invoke_js("showid", "id").

hideid Hide an element with the id provided. Example: golem::invoke_js("hideid", "id").

showclass Same as showid, but with class. Example: golem::invoke_js("showclass", "someClass").

hideclass Same as hideid, but with class. Example: golem::invoke_js("hideclass", "someClass").

showhref Same as showid, but with a[href*=. Example: golem::invoke_js("showhref", "thinkr.fr").

hidehref Same as hideid, but with a[href*=. Example: golem::invoke_js("hidehref", "thinkr.fr").

clickon Click on an element. The full jQuery selector has to be used. Example: golem::invoke_js("clickon", "#id").

disable Add "disabled" to an element. The full jQuery selector has to be used. Example: golem::invoke_js("disable", ".someClass").

reable Remove "disabled" from an element. The full jQuery selector has to be used. Example: golem::invoke_js("reable", ".someClass").

alert Open an alert box with the message(s) provided. Example: golem::invoke_js("alert", "WELCOME TO MY APP");.

prompt Open a prompt box with the message(s) provided. This function takes a list with message and id `list(message = "", id = "")`. The output of the prompt will be sent to `input$id`. Example: `golem::invoke_js("prompt", list(message ="what's your name?", id = "name") )`.

confirm Open a confirm box with the message provided. This function takes a list with message and id `list(message = "", id = "")`. The output of the prompt will be sent to `input$id`. Example: `golem::invoke_js("confirm", list(message ="Are you sure you want to do that?", id = "name") )`.

Details

These JavaScript functions can be called from the server with `invoke_js`. `invoke_js` can also be used to launch any JS function created inside a Shiny JavaScript handler.

Value

Used for side-effects.

Examples

```
if (interactive()) {
  library(shiny)
  ui <- fluidPage(
    golem::activate_js(), # already loaded in your golem by `bundle_resources()`
    fluidRow(
      actionButton(inputId = "hidebutton1", label = "hide button1"),
      actionButton(inputId = "showbutton1", label = "show button1"),
      actionButton(inputId = "button1", label = "button1")
    ),
    fluidRow(
      actionButton(inputId = "hideclassA", label = "hide class A"),
      actionButton(inputId = "showclassA", label = "show class A"),
      actionButton(inputId = "buttonA1", label = "button A1", class = "A"),
      actionButton(inputId = "buttonA2", label = "button A2", class = "A"),
      actionButton(inputId = "buttonA3", label = "button A3", class = "A")
    ),
    fluidRow(
    actionButton(inputId = "clickhide", label = "click on 'hide button1' and 'hide class A'"),
    actionButton(inputId = "clickshow", label = "click on 'show button1' and 'show class A'")
    ),
    fluidRow(
      actionButton(inputId = "disableA", label = "disable class A"),
      actionButton(inputId = "reableA", label = "reable class A")
    ),
    fluidRow(
      actionButton(inputId = "alertbutton", label = "alert button"),
      actionButton(inputId = "promptbutton", label = "prompt button"),
      actionButton(inputId = "confirmbutton", label = "confirm button")
    )
  )

  server <- function(input, output, session) {
```

```
    observeEvent(input$hidebutton1, {
      golem::invoke_js("hideid", "button1")
    })

    observeEvent(input$showbutton1, {
      golem::invoke_js("showid", "button1")
    })

    observeEvent(input$hideclassA, {
      golem::invoke_js("hideclass", "A")
    })
    observeEvent(input$showclassA, {
      golem::invoke_js("showclass", "A")
    })

    observeEvent(input$clickhide, {
      golem::invoke_js("clickon", "#hidebutton1")
      golem::invoke_js("clickon", "#hideclassA")
    })

    observeEvent(input$clickshow, {
      golem::invoke_js("clickon", "#showbutton1")
      golem::invoke_js("clickon", "#showclassA")
    })

    observeEvent(input$disableA, {
      golem::invoke_js("disable", ".A")
    })
    observeEvent(input$reableA, {
      golem::invoke_js("reable", ".A")
    })

    observeEvent(input$alertbutton, {
      golem::invoke_js("alert", "ALERT!!")
    })

    observeEvent(input$promptbutton, {
      golem::invoke_js("prompt", list(message = "what's your name?", id = "name"))
    })
    observeEvent(input$name, {
      message(paste("input$name", input$name))
    })

    observeEvent(input$confirmbutton, {
      golem::invoke_js("confirm", list(message = "Are you sure?", id = "sure"))
    })
    observeEvent(input$sure, {
      message(paste("input$sure", input$sure))
    })
  }
  shinyApp(ui, server)
}
```

addins {golem} *addins*

Description

insert_ns() takes a selected character vector and wrap it in ns() The series of go_to_*() addins help you go to common files used in developing a {golem} application.

Usage

```
insert_ns()

go_to_start(wd = golem::get_golem_wd())

go_to_dev(wd = golem::get_golem_wd())

go_to_deploy(wd = golem::get_golem_wd())

go_to_run_dev(wd = golem::get_golem_wd())

go_to_app_ui(wd = golem::get_golem_wd())

go_to_app_server(wd = golem::get_golem_wd())

go_to_run_app(wd = golem::get_golem_wd())
```

Arguments

wd	The working directory of the {golem} application.

add_dockerfile *Create a Dockerfile for your App*

Description

Build a container containing your Shiny App. add_dockerfile() and add_dockerfile_with_renv() and add_dockerfile_with_renv() creates a generic Dockerfile, while add_dockerfile_shinyproxy(), add_dockerfile_with_renv_shinyproxy() , add_dockerfile_with_renv_shinyproxy() and add_dockerfile_heroku() creates platform specific Dockerfile.

Usage

```
add_dockerfile(
  path = "DESCRIPTION",
  output = "Dockerfile",
  pkg = get_golem_wd(),
  from = paste0("rocker/verse:", R.Version()$major, ".", R.Version()$minor),
  as = NULL,
  port = 80,
  host = "0.0.0.0",
  sysreqs = TRUE,
  repos = c(CRAN = "https://cran.rstudio.com/"),
  expand = FALSE,
  open = TRUE,
  update_tar_gz = TRUE,
  build_golem_from_source = TRUE,
  extra_sysreqs = NULL
)

add_dockerfile_shinyproxy(
  path = "DESCRIPTION",
  output = "Dockerfile",
  pkg = get_golem_wd(),
  from = paste0("rocker/verse:", R.Version()$major, ".", R.Version()$minor),
  as = NULL,
  sysreqs = TRUE,
  repos = c(CRAN = "https://cran.rstudio.com/"),
  expand = FALSE,
  open = TRUE,
  update_tar_gz = TRUE,
  build_golem_from_source = TRUE,
  extra_sysreqs = NULL
)

add_dockerfile_heroku(
  path = "DESCRIPTION",
  output = "Dockerfile",
  pkg = get_golem_wd(),
  from = paste0("rocker/verse:", R.Version()$major, ".", R.Version()$minor),
  as = NULL,
  sysreqs = TRUE,
  repos = c(CRAN = "https://cran.rstudio.com/"),
  expand = FALSE,
  open = TRUE,
  update_tar_gz = TRUE,
  build_golem_from_source = TRUE,
  extra_sysreqs = NULL
)
```

```
add_dockerfile_with_renv(
  source_folder = get_golem_wd(),
  lockfile = NULL,
  output_dir = fs::path(tempdir(), "deploy"),
  distro = "focal",
  from = "rocker/verse",
  as = NULL,
  sysreqs = TRUE,
  port = 80,
  host = "0.0.0.0",
  repos = c(CRAN = "https://cran.rstudio.com/"),
  expand = FALSE,
  open = TRUE,
  document = TRUE,
  extra_sysreqs = NULL,
  update_tar_gz = TRUE,
  dockerfile_cmd = NULL,
  user = "rstudio",
  ...
)

add_dockerfile_with_renv_shinyproxy(
  source_folder = get_golem_wd(),
  lockfile = NULL,
  output_dir = fs::path(tempdir(), "deploy"),
  distro = "focal",
  from = "rocker/verse",
  as = NULL,
  sysreqs = TRUE,
  repos = c(CRAN = "https://cran.rstudio.com/"),
  expand = FALSE,
  extra_sysreqs = NULL,
  open = TRUE,
  document = TRUE,
  update_tar_gz = TRUE,
  user = "rstudio",
  ...
)

add_dockerfile_with_renv_heroku(
  source_folder = get_golem_wd(),
  lockfile = NULL,
  output_dir = fs::path(tempdir(), "deploy"),
  distro = "focal",
  from = "rocker/verse",
  as = NULL,
  sysreqs = TRUE,
  repos = c(CRAN = "https://cran.rstudio.com/"),
```

```
  expand = FALSE,
  extra_sysreqs = NULL,
  open = TRUE,
  document = TRUE,
  user = "rstudio",
  update_tar_gz = TRUE,
  ...
)
```

Arguments

path	path to the DESCRIPTION file to use as an input.
output	name of the Dockerfile output.
pkg	Path to the root of the package. Default is `get_golem_wd()`.

from	The FROM of the Dockerfile. Default is
	`FROM rocker/verse`
	
	`without renv.lock file passed`
	`` `R.Version()$major`.`R.Version()$minor` is used as tag ``

as	The AS of the Dockerfile. Default it NULL.
port	The `options('shiny.port')` on which to run the App. Default is 80.
host	The `options('shiny.host')` on which to run the App. Default is 0.0.0.0.
sysreqs	boolean. If TRUE, RUN statements to install packages system requirements will be included in the Dockerfile.
repos	character. The URL(s) of the repositories to use for `options("repos")`.
expand	boolean. If `TRUE` each system requirement will have its own `RUN` line.
open	boolean. Should the Dockerfile/README/README be open after creation? Default is `TRUE`.
update_tar_gz	boolean. If `TRUE` and `build_golem_from_source` is also `TRUE`, an updated tar.gz is created.
build_golem_from_source	boolean. If `TRUE` no tar.gz is created and the Dockerfile directly mount the source folder.
extra_sysreqs	character vector. Extra debian system requirements.
source_folder	path to the Package/golem source folder to deploy. default is retrieved via `get_golem_wd()`.
lockfile	path to the renv.lock file to use. default is `NULL`.
output_dir	folder to export everything deployment related.
distro	One of "focal", "bionic", "xenial", "centos7", or "centos8". See available distributions at https://hub.docker.com/r/rstudio/r-base/.
document	boolean. If TRUE (by default), DESCRIPTION file is updated using `attachment::att_amend_desc()` before creating the renv.lock file

dockerfile_cmd	What is the CMD to add to the Dockerfile. If NULL, the default, the CMD will be `R -e "options('shiny.port'={port},shiny.host='{host}');library({appname});{appname`
user	Name of the user to specify in the Dockerfile with the USER instruction. Default is `rstudio`, if set to `NULL` no the user from the FROM image is used.
...	Other arguments to pass to `renv::snapshot()`.

Value

The `{dockerfiler}` object, invisibly.

Examples

```
# Add a standard Dockerfile
if (interactive() & requireNamespace("dockerfiler")) {
  add_dockerfile()
}
# Crete a 'deploy' folder containing everything needed to deploy
# the golem using docker based on {renv}
if (interactive() & requireNamespace("dockerfiler")) {
  add_dockerfile_with_renv(
    # lockfile = "renv.lock", # uncomment to use existing renv.lock file
    output_dir = "deploy"
  )
}
# Add a Dockerfile for ShinyProxy
if (interactive() & requireNamespace("dockerfiler")) {
  add_dockerfile_shinyproxy()
}

# Crete a 'deploy' folder containing everything needed to deploy
# the golem with ShinyProxy using docker based on {renv}
if (interactive() & requireNamespace("dockerfiler")) {
  add_dockerfile_with_renv(
    # lockfile = "renv.lock",# uncomment to use existing renv.lock file
    output_dir = "deploy"
  )
}

# Add a Dockerfile for Heroku
if (interactive() & requireNamespace("dockerfiler")) {
  add_dockerfile_heroku()
}
```

`add_fct` *Add fct_ and utils_ files*

Description

These functions add files in the R/ folder that starts either with `fct_` (short for function) or with `utils_`.

Usage

```
add_fct(
  name,
  module = NULL,
  pkg = get_golem_wd(),
  open = TRUE,
  dir_create = TRUE,
  with_test = FALSE
)

add_utils(
  name,
  module = NULL,
  pkg = get_golem_wd(),
  open = TRUE,
  dir_create = TRUE,
  with_test = FALSE
)

add_r6(
  name,
  module = NULL,
  pkg = get_golem_wd(),
  open = TRUE,
  dir_create = TRUE,
  with_test = FALSE
)
```

Arguments

`name`	The name of the file
`module`	If not NULL, the file will be module specific in the naming (you don't need to add the leading `mod_`).
`pkg`	Path to the root of the package. Default is `get_golem_wd()`.
`open`	Should the created file be opened?
`dir_create`	Creates the directory if it doesn't exist, default is `TRUE`.
`with_test`	should the module be created with tests?

Value

The path to the file, invisibly.

add_js_file *Create Files*

Description

These functions create files inside the `inst/app` folder.

Usage

```
add_js_file(
  name,
  pkg = get_golem_wd(),
  dir = "inst/app/www",
  open = TRUE,
  dir_create = TRUE,
  with_doc_ready = TRUE,
  template = golem::js_template,
  ...
)

add_js_handler(
  name,
  pkg = get_golem_wd(),
  dir = "inst/app/www",
  open = TRUE,
  dir_create = TRUE,
  template = golem::js_handler_template,
  ...
)

add_js_input_binding(
  name,
  pkg = get_golem_wd(),
  dir = "inst/app/www",
  open = TRUE,
  dir_create = TRUE,
  initialize = FALSE,
  dev = FALSE,
  events = list(name = "click", rate_policy = FALSE)
)

add_js_output_binding(
  name,
  pkg = get_golem_wd(),
  dir = "inst/app/www",
  open = TRUE,
  dir_create = TRUE
```

```
)

add_css_file(
  name,
  pkg = get_golem_wd(),
  dir = "inst/app/www",
  open = TRUE,
  dir_create = TRUE,
  template = golem::css_template,
  ...
)

add_sass_file(
  name,
  pkg = get_golem_wd(),
  dir = "inst/app/www",
  open = TRUE,
  dir_create = TRUE,
  template = golem::sass_template,
  ...
)

add_empty_file(
  name,
  pkg = get_golem_wd(),
  dir = "inst/app/www",
  open = TRUE,
  dir_create = TRUE,
  template = golem::empty_template,
  ...
)

add_html_template(
  name = "template.html",
  pkg = get_golem_wd(),
  dir = "inst/app/www",
  open = TRUE,
  dir_create = TRUE
)

add_partial_html_template(
  name = "partial_template.html",
  pkg = get_golem_wd(),
  dir = "inst/app/www",
  open = TRUE,
  dir_create = TRUE
)
```

```
add_ui_server_files(pkg = get_golem_wd(), dir = "inst/app", dir_create = TRUE)
```

Arguments

name	The name of the module.
pkg	Path to the root of the package. Default is get_golem_wd().
dir	Path to the dir where the file while be created.
open	Should the created file be opened?
dir_create	Creates the directory if it doesn't exist, default is TRUE.
with_doc_ready	For JS file - Should the default file include $(document).ready()?
template	Function writing in the created file. You may overwrite this with your own template function.
...	Arguments to be passed to the template function.
initialize	For JS file - Whether to add the initialize method. Default to FALSE. Some JavaScript API require to initialize components before using them.
dev	Whether to insert console.log calls in the most important methods of the binding. This is only to help building the input binding. Default is FALSE.
events	List of events to generate event listeners in the subscribe method. For instance, list(name = c("click", "keyup"), rate_policy = c(FALSE, TRUE)). The list contain names and rate policies to apply to each event. If a rate policy is found, the debounce method with a default delay of 250 ms is applied. You may edit manually according to https://shiny.rstudio.com/articles/building-inputs.html

Value

The path to the file, invisibly.

Note

add_ui_server_files will be deprecated in future version of {golem}

See Also

js_template, js_handler_template, and css_template

---

add_module *Create a module*

Description

This function creates a module inside the R/ folder, based on a specific module structure. This function can be used outside of a {golem} project.

Usage

```
add_module(
  name,
  pkg = get_golem_wd(),
  open = TRUE,
  dir_create = TRUE,
  fct = NULL,
  utils = NULL,
  r6 = NULL,
  js = NULL,
  js_handler = NULL,
  export = FALSE,
  module_template = golem::module_template,
  with_test = FALSE,
  ...
)
```

Arguments

name	The name of the module.
pkg	Path to the root of the package. Default is get_golem_wd().
open	Should the created file be opened?
dir_create	Creates the directory if it doesn't exist, default is TRUE.
fct	If specified, creates a mod_fct file.
utils	If specified, creates a mod_utils file.
r6	If specified, creates a mod_class file.
js, js_handler	If specified, creates a module related JavaScript file.
export	Should the module be exported? Default is FALSE.
module_template	Function that serves as a module template.
with_test	should the module be created with tests?
...	Arguments to be passed to the module_template function.

Value

The path to the file, invisibly.

Note

This function will prefix the name argument with mod_.

See Also

module_template()

add_positconnect_file *Add an* app.R *at the root of your package to deploy on RStudio Connect*

Description

Additionally, adds a .rscignore at the root of the {golem} project if the rsconnect package version is >= 0.8.25.

Usage

```
add_positconnect_file(pkg = get_golem_wd(), open = TRUE)

add_rstudioconnect_file(pkg = get_golem_wd(), open = TRUE)

add_shinyappsio_file(pkg = get_golem_wd(), open = TRUE)

add_shinyserver_file(pkg = get_golem_wd(), open = TRUE)

add_rscignore_file(pkg = get_golem_wd(), open = TRUE)
```

Arguments

pkg	Path to the root of the package. Default is get_golem_wd().
open	Should the created file be opened?

Value

Side-effect functions for file creation returning the path to the file, invisibly.

List of excluded files in .rscignore

- .here
- CODE_OF_CONDUCT.md
- LICENSE{.md}
- LICENCE{.md}
- NEWS{.md}
- README{.md,.Rmd,.HTML}
- dev
- man
- tests
- vignettes

Note

In previous versions, this function was called add_rconnect_file.

`add_rstudioconnect_file` is now deprecated; replace by `add_positconnect_file()`.

Examples

```
# Add a file for Connect
if (interactive()) {
  add_positconnect_file()
}
# Add a file for Shiny Server
if (interactive()) {
  add_shinyserver_file()
}
# Add a file for Shinyapps.io
if (interactive()) {
  add_shinyappsio_file()
}
```

`add_resource_path` *Add resource path*

Description

Add resource path

Usage

```
add_resource_path(prefix, directoryPath, warn_empty = FALSE)
```

Arguments

`prefix`	The URL prefix (without slashes). Valid characters are a-z, A-Z, 0-9, hyphen, period, and underscore. For example, a value of 'foo' means that any request paths that begin with '/foo' will be mapped to the given directory.
`directoryPath`	The directory that contains the static resources to be served.
`warn_empty`	Boolean. Default is `FALSE`. If TRUE display message if directory is empty.

Value

Used for side effects.

amend_golem_config *Amend golem config file*

Description

Amend golem config file

Usage

```
amend_golem_config(
  key,
  value,
  config = "default",
  pkg = golem::pkg_path(),
  talkative = TRUE
)
```

Arguments

key	key of the value to add in config
value	Name of value (NULL to read all values)
config	Name of configuration to read from. Defaults to the value of the R_CONFIG_ACTIVE environment variable ("default" if the variable does not exist).
pkg	Path to the root of the package. Default is get_golem_wd().
talkative	Should the messages be printed to the console?

Value

Used for side effects.

app_prod *Is the app in dev mode or prod mode?*

Description

Is the app in dev mode or prod mode?

Usage

```
app_prod()

app_dev()
```

Value

TRUE or FALSE depending on the status of getOption("golem.app.prod")

browser_button *Insert an hidden browser button*

Description

See https://rtask.thinkr.fr/a-little-trick-for-debugging-shiny/ for more context.

Usage

```
browser_button()
```

Value

Used for side effects. Prints the code to the console.

bundle_resources *Automatically serve golem external resources*

Description

This function is a wrapper around `htmltools::htmlDependency` that automatically bundles the CSS and JavaScript files in `inst/app/www` and which are created by `golem::add_css_file()` , `golem::add_js_file()` and `golem::add_js_handler()`.

Usage

```
bundle_resources(
  path,
  app_title,
  name = "golem_resources",
  version = "0.0.1",
  meta = NULL,
  head = NULL,
  attachment = NULL,
  package = NULL,
  all_files = TRUE,
  app_builder = "golem",
  with_sparkles = FALSE
)
```

Arguments

path	The path to the folder where the external files are located.
app_title	The title of the app, to be used as an application title.
name	Library name
version	Library version
meta	Named list of meta tags to insert into document head
head	Arbitrary lines of HTML to insert into the document head
attachment	Attachment(s) to include within the document head. See Details.
package	An R package name to indicate where to find the src directory when src is a relative path (see resolveDependencies()).
all_files	Whether all files under the src directory are dependency files. If FALSE, only the files specified in script, stylesheet, and attachment are treated as dependency files.
app_builder	The name of the app builder to add as a meta tag. Turn to NULL if you don't want this meta tag to be included.
with_sparkles	C'est quand que tu vas mettre des paillettes dans ma vie Kevin?

Details

This function also preload activate_js() which allows to use preconfigured JavaScript functions via invoke_js().

Value

an htmlDependency

---

cat_dev *Functions already made dev dependent*

Description

This functions will be run only if golem::app_dev() returns TRUE.

Usage

```
cat_dev(...)

print_dev(...)

message_dev(...)

warning_dev(...)

browser_dev(...)
```

Arguments

...	R objects (see 'Details' for the types of objects allowed).

Value

A modified function.

`create_golem` *Create a package for a Shiny App using* `{golem}`

Description

Create a package for a Shiny App using `{golem}`

Usage

```
create_golem(
  path,
  check_name = TRUE,
  open = TRUE,
  overwrite = FALSE,
  package_name = basename(normalizePath(path, mustWork = FALSE)),
  without_comments = FALSE,
  project_hook = golem::project_hook,
  with_git = FALSE,
  ...
)
```

Arguments

`path`	Name of the folder to create the package in. This will also be used as the package name.
`check_name`	Should we check that the package name is correct according to CRAN requirements.
`open`	Boolean. Open the created project?
`overwrite`	Boolean. Should the already existing project be overwritten ?
`package_name`	Package name to use. By default, `{golem}` uses `basename(path)`. If `path == '.' & package_name` is not explicitly set, then `basename(getwd())` will be used.
`without_comments`	Boolean. Start project without `{golem}` comments
`project_hook`	A function executed as a hook after project creation. Can be used to change the default `{golem}` structure. to override the files and content. This function is executed just after the project is created.
`with_git`	Boolean. Initialize git repository
`...`	Arguments passed to the `project_hook()` function.

Value

The path, invisibly.

Note

For compatibility issue, this function turns `options(shiny.autoload.r)` to `FALSE`. See https://github.com/ThinkR-open/golem/issues/468 for more background.

---

detach_all_attached *Detach all attached package*

Description

Detach all attached package

Usage

```
detach_all_attached()
```

Value

TRUE, invisibly.

---

disable_autoload *Disabling Shiny Autoload of R Scripts*

Description

Disabling Shiny Autoload of R Scripts

Usage

```
disable_autoload(pkg = get_golem_wd())
```

Arguments

pkg	Path to the root of the package. Default is `get_golem_wd()`.

Value

The path to the file, invisibly.

Examples

```
if (interactive()) {
  disable_autoload()
}
```

document_and_reload *Document and reload your package*

Description

This function calls `rstudioapi::documentSaveAll()`, `roxygen2::roxygenise()` and `pkgload::load_all()`.

Usage

```
document_and_reload(
  pkg = get_golem_wd(),
  roclets = NULL,
  load_code = NULL,
  clean = FALSE,
  export_all = FALSE,
  helpers = FALSE,
  attach_testthat = FALSE,
  ...
)
```

Arguments

pkg	Path to the root of the package. Default is `get_golem_wd()`.
roclets	Character vector of roclet names to use with package. The default, `NULL`, uses the roxygen `roclets` option, which defaults to `c("collate", "namespace", "rd")`.
load_code	A function used to load all the R code in the package directory. The default, `NULL`, uses the strategy defined by the `load` roxygen option, which defaults to `load_pkgload()`. See load for more details.
clean	If `TRUE`, roxygen will delete all files previously created by roxygen before running each roclet.
export_all	If `TRUE` (the default), export all objects. If `FALSE`, export only the objects that are listed as exports in the NAMESPACE file.
helpers	if `TRUE` loads **testthat** test helpers.
attach_testthat	If `TRUE`, attach **testthat** to the search path, which more closely mimics the environment within test files.
...	Other arguments passed to `pkgload::load_all()`

Value

Used for side-effects

expect_shinytag *Test helpers*

Description

These functions are designed to be used inside the tests in your Shiny app package.

Usage

```
expect_shinytag(object)

expect_shinytaglist(object)

expect_html_equal(ui, html, ...)

expect_running(sleep, R_path = NULL)
```

Arguments

object	the object to test
ui	output of an UI function
html	deprecated
...	arguments passed to testthat::expect_snapshot()
sleep	number of seconds
R_path	path to R. If NULL, the function will try to guess where R is.

Value

A testthat result.

fill_desc *Fill your* DESCRIPTION *file*

Description

Generates a standard DESCRIPTION file as used in R packages. Also sets a series of global options inside golem-config.yml that will be reused inside {golem} (see set_options and set_golem_options() for details).

Usage

```
fill_desc(
  pkg_name,
  pkg_title,
  pkg_description,
  authors = person(given = NULL, family = NULL, email = NULL, role = NULL, comment =
    NULL),
  repo_url = NULL,
  pkg_version = "0.0.0.9000",
  pkg = get_golem_wd(),
  author_first_name = NULL,
  author_last_name = NULL,
  author_email = NULL,
  author_orcid = NULL,
  set_options = TRUE
)
```

Arguments

pkg_name	The name of the package
pkg_title	The title of the package
pkg_description	Description of the package
authors	a character string (or vector) of class person (see person() for details)
repo_url	URL (if needed)
pkg_version	The version of the package. Default is 0.0.0.9000
pkg	Path to look for the DESCRIPTION. Default is get_golem_wd().
author_first_name	to be deprecated: use character for first name via authors = person(given = "authors_first_name") instead
author_last_name	to be deprecated: use character for last name via authors = person(given = "authors_last_name") instead
author_email	to be deprecated: use character for first name via authors = person(email = "author_email") instead
author_orcid	to be deprecated
set_options	logical; the default TRUE sets all recommended options but this can be suppressed with FALSE. For details on the exact behaviour see the help set_golem_options().

Value

The {desc} object, invisibly.

`get_current_config` *Return path to the* {golem} *config-file*

Description

This function tries to find the path to the {golem} config-file currently used. If the config-file is not found, the user is asked if they want to set parts of the {golem} skeleton. This includes default versions of "R/app_config" and "inst/golem-config.yml" (assuming that the user tries to convert the directory to a {golem} based shiny App).

Usage

```
get_current_config(path = getwd())
```

Arguments

path	character string giving the path to start looking for the config; the usual value is the {golem}-package top-level directory but a user supplied config is now supported (see **Details** for how to use this feature).

Details

In most cases this function simply returns the path to the default golem-config file located under "inst/golem-config.yml". That config comes in yml-format, see the Engineering Production-Grade Shiny Apps for further details on its format and how to set options therein.

Advanced app developers may benefit from having an additional user config-file. This is achieved by copying the file to the new location and setting a new path pointing to this file. The path is altered inside the app_sys()-call of the file "R/app_config.R" to point to the (relative to inst/) location of the user-config i.e. change

```
# Modify this if your config file is somewhere else
file = app_sys("golem-config.yml")
```

to

```
# Modify this if your config file is somewhere else
file = app_sys("configs/user-golem-config.yml")
```

NOTE

- the path to the config is changed relative to ***inst/*** from ***inst/golem-config.yml*** to ***inst/configs/user-golem-config.yml***
- if both, the default config ***and*** user config files exists (and the path is set correctly for the latter), an error is thrown due to ambiguity: in this case simply rename/delete the default config or change the entry in "R/app_config.R" back to app_sys("golem-config.yml") to point to the default location

Value

character string giving the path to the {golem} config-file

---

get_golem_options *Get all or one golem options*

Description

This function is to be used inside the server and UI from your app, in order to call the parameters passed to run_app().

Usage

```
get_golem_options(which = NULL)
```

Arguments

which	NULL (default), or the name of an option

Value

The value of the option.

Examples

```
# Define and use golem_options
if (interactive()) {
  # 1. Pass parameters directly to `run_app`

  run_app(
    title = "My Golem App",
    content = "something"
  )

  # 2. Get the values
  # 2.1 from the UI side

  h1(get_golem_options("title"))

  # 2.2 from the server-side

  output$param <- renderPrint({
    paste("param content = ", get_golem_options("content"))
  })

  output$param_full <- renderPrint({
    get_golem_options() # list of all golem options as a list.
  })
```

```
  # 3. If needed, to set default value, edit `run_app` like this :

  run_app <- function(
  title = "this",
  content = "that",
  ...
  ) {
    with_golem_options(
      app = shinyApp(
        ui = app_ui,
        server = app_server
      ),
      golem_opts = list(
        title = title,
        content = content,
        ...
      )
    )
  }
}
```

---

get_golem_wd {golem} *options*

Description

Set and get a series of options to be used with {golem}. These options are found inside the golem-config.yml file, found in most cases inside the inst folder.

Usage

```
get_golem_wd(use_parent = TRUE, pkg = golem::pkg_path())

get_golem_name(
 config = Sys.getenv("GOLEM_CONFIG_ACTIVE", Sys.getenv("R_CONFIG_ACTIVE", "default")),
  use_parent = TRUE,
  pkg = golem::pkg_path()
)

get_golem_version(
 config = Sys.getenv("GOLEM_CONFIG_ACTIVE", Sys.getenv("R_CONFIG_ACTIVE", "default")),
  use_parent = TRUE,
  pkg = golem::pkg_path()
)

set_golem_wd(
  golem_wd = golem::pkg_path(),
  pkg = golem::pkg_path(),
```

```
  talkative = TRUE
)

set_golem_name(
  name = golem::pkg_name(),
  pkg = golem::pkg_path(),
  talkative = TRUE,
  old_name = golem::pkg_name()
)

set_golem_version(
  version = golem::pkg_version(),
  pkg = golem::pkg_path(),
  talkative = TRUE
)

set_golem_options(
  golem_name = golem::pkg_name(),
  golem_version = golem::pkg_version(),
  golem_wd = golem::pkg_path(),
  app_prod = FALSE,
  talkative = TRUE,
  config_file = golem::get_current_config(golem_wd)
)
```

Arguments

use_parent	TRUE to scan parent directories for configuration files if the specified config file isn't found.
pkg	The path to set the golem working directory. Note that it will be passed to normalizePath.
config	Name of configuration to read from. Defaults to the value of the R_CONFIG_ACTIVE environment variable ("default" if the variable does not exist).
golem_wd	Working directory of the current golem package.
talkative	Should the messages be printed to the console?
name	The name of the app
old_name	The old name of the app, used when changing the name
version	The version of the app
golem_name	Name of the current golem.
golem_version	Version of the current golem.
app_prod	Is the {golem} in prod mode?
config_file	path to the {golem} config file

Value

Used for side-effects for the setters, and values from the config in the getters.

Set Functions

- `set_golem_options()` sets all the options, with the defaults from the functions below.
- `set_golem_wd()` defaults to `golem::golem_wd()`, which is the package root when starting a golem.
- `set_golem_name()` defaults `golem::pkg_name()`
- `set_golem_version()` defaults `golem::pkg_version()`

Get Functions

Reads the information from `golem-config.yml`

- `get_golem_wd()`
- `get_golem_name()`
- `get_golem_version()`

`get_sysreqs` *Get system requirements (Deprecated)*

Description

This function is now deprecated, and was moved to `{dockerfiler}`.

Usage

```
get_sysreqs(packages, quiet = TRUE, batch_n = 30)
```

Arguments

`packages`	character vector. Packages names.
`quiet`	Boolean. If `TRUE` the function is quiet.
`batch_n`	numeric. Number of simultaneous packages to ask.

Value

A vector of system requirements.

golem *A package for building Shiny App*

Description

Read more about building big shiny apps at https://engineering-shiny.org/.

Author(s)

Maintainer: Colin Fay <contact@colinfay.me> (ORCID)

Authors:

- Vincent Guyader <vincent@thinkr.fr> (ORCID) (previous maintainer)
- Sébastien Rochette <sebastien@thinkr.fr> (ORCID)
- Cervan Girard <cervan@thinkr.fr> (ORCID)

Other contributors:

- Novica Nakov <nnovica@gmail.com> [contributor]
- David Granjon <dgranjon@ymail.com> [contributor]
- Arthur Bréant <arthur@thinkr.fr> [contributor]
- Antoine Languillaume <antoine@thinkr.fr> [contributor]
- Ilya Zarubin <zarubin@wiso.uni-koeln.de> [contributor]
- ThinkR [copyright holder]

See Also

Useful links:

- https://thinkr-open.github.io/golem/
- https://github.com/ThinkR-open/golem
- Report bugs at https://github.com/ThinkR-open/golem/issues

golem_welcome_page *Welcome Page*

Description

Welcome Page

Usage

```
golem_welcome_page()
```

Value

A welcome page for your {golem} app

`install_dev_deps` *Install* {golem} *dev dependencies*

Description

This function will run rlang::check_installed() on:

- usethis
- pkgload
- dockerfiler
- devtools
- roxygen2
- attachment
- rstudioapi
- here
- fs
- desc
- pkgbuild
- processx
- rsconnect
- testthat
- rstudioapi

Usage

```
install_dev_deps(dev_deps, force_install = FALSE, ...)
```

Arguments

`dev_deps`	optional character vector of packages to install
`force_install`	If force_install is TRUE, then the user is not interactively asked to install them.
`...`	further arguments passed to the install function.

Value

Used for side-effects

Examples

```
if (interactive()) {
  install_dev_deps()
}
```

is_golem *Is the directory a golem-based app?*

Description

Trying to guess if `path` is a golem-based app.

Usage

```
is_golem(path = getwd())
```

Arguments

`path`	Path to the directory to check. Defaults to the current working directory.

Value

A boolean, `TRUE` if the directory is a golem-based app, `FALSE` else.

Examples

```
is_golem()
```

is_running *Is the running app a golem app?*

Description

Note that this will return `TRUE` only if the application has been launched with `with_golem_options()`

Usage

```
is_running()
```

Value

TRUE if the running app is a `{golem}` based app, FALSE otherwise.

Examples

```
is_running()
```

js_handler_template *Golem's default custom templates*

Description

These functions do not aim at being called as is by users, but to be passed as an argument to the `add_js_handler()` function.

Usage

```
js_handler_template(path, name = "fun", code = " ")

js_template(path, code = " ")

css_template(path, code = " ")

sass_template(path, code = " ")

empty_template(path, code = " ")
```

Arguments

`path`	The path to the JS script where this template will be written.
`name`	Shiny's custom handler name.
`code`	JavaScript code to be written in the function.

Value

Used for side effect

See Also

`add_js_handler()`

maintenance_page *maintenance_page*

Description

A default html page for maintenance mode

Usage

```
maintenance_page()
```

Details

see the vignette `vignette("f_extending_golem", package = "golem")` for details.

Value

an html_document

`make_dev` *Make a function dependent to dev mode*

Description

The function returned will be run only if `golem::app_dev()` returns TRUE.

Usage

```
make_dev(fun)
```

Arguments

`fun`	A function

Value

Used for side-effects

`module_template` *Golem Module Template Function*

Description

Module template can be used to extend golem module creation mechanism with your own template, so that you can be even more productive when building your `{shiny}` app. Module template functions do not aim at being called as is by users, but to be passed as an argument to the `add_module()` function.

Usage

```
module_template(name, path, export, ph_ui = " ", ph_server = " ", ...)
```

Arguments

name	The name of the module.
path	The path to the R script where the module will be written. Note that this path will not be set by the user but via add_module().
export	Should the module be exported? Default is FALSE.
ph_ui, ph_server	Texts to insert inside the modules UI and server. For advanced use.
...	Arguments to be passed to the module_template function.

Details

Module template functions are a way to define your own template function for module. A template function that can take the following arguments to be passed from add_module():

- name: the name of the module
- path: the path to the file in R/
- export: a TRUE/FALSE set by the export param of add_module()

If you want your function to ignore these parameters, set ... as the last argument of your function, then these will be ignored. See the examples section of this help.

Value

Used for side effect

See Also

add_module()

Examples

```
if (interactive()) {
  my_tmpl <- function(name, path, ...) {
    # Define a template that write to the
    # module file
    write(name, path)
  }
  golem::add_module(name = "custom", module_template = my_tmpl)

  my_other_tmpl <- function(name, path, ...) {
    # Copy and paste a file from somewhere
    file.copy(..., path)
  }
  golem::add_module(name = "custom", module_template = my_other_tmpl)
}
```

pkg_name *Package tools*

Description

These are functions to help you navigate inside your project while developing

Usage

```
pkg_name(path = get_golem_wd())

pkg_version(path = get_golem_wd())

pkg_path()
```

Arguments

path	Path to use to read the DESCRIPTION

Value

The value of the entry in the DESCRIPTION file

project_hook *Project Hook*

Description

Project hooks allow to define a function run just after {golem} project creation.

Usage

```
project_hook(path, package_name, ...)
```

Arguments

path	Name of the folder to create the package in. This will also be used as the package name.
package_name	Package name to use. By default, {golem} uses basename(path). If path == '.' & package_name is not explicitly set, then basename(getwd()) will be used.
...	Arguments passed from create_golem(), unused in the default function.

Value

Used for side effects

Examples

```
if (interactive()) {
  my_proj <- function(...) {
    unlink("dev/", TRUE, TRUE)
  }
  create_golem("ici", project_template = my_proj)
}
```

---

run_dev *Run the* dev/run_dev.R *file*

Description

The default file="dev/run_dev.R" launches your {golem} app with a bunch of useful options. The file content can be customized and file-name and path changed as long as the argument combination of file and pkg are supplied correctly: the file-path is a relative path to a {golem}-package root pkg. An error is thrown if pkg/file cannot be found.

Usage

```
run_dev(
  file = "dev/run_dev.R",
  pkg = get_golem_wd(),
  save_all = TRUE,
  install_required_packages = TRUE
)
```

Arguments

file	String with (relative) file path to a run_dev.R-file
pkg	Path to the root of the package. Default is get_golem_wd().
save_all	Boolean; if TRUE saves all open files before sourcing file
install_required_packages	Boolean; if TRUE install the packages used in run_dev.R-file

Details

The function run_dev() is typically used to launch a shiny app by sourcing the content of an appropriate run_dev-file. Carefully read the content of dev/run_dev.R when creating your custom run_dev-file. It has already many useful settings including a switch between production/development, reloading the package in a clean R environment before running the app etc.

Value

pure side-effect function; returns invisibly

sanity_check *Sanity check for R files in the project*

Description

This function is used check for any 'browser()" or commented #TODO / #TOFIX / #BUG in the code

Usage

```
sanity_check(pkg = get_golem_wd())
```

Arguments

pkg	Path to the root of the package. Default is `get_golem_wd()`.

Value

A DataFrame if any of the words has been found.

use_external_js_file *Use Files*

Description

These functions download files from external sources and put them inside the `inst/app/www` directory. The `use_internal_` functions will copy internal files, while `use_external_` will try to download them from a remote location.

Usage

```
use_external_js_file(
  url,
  name = NULL,
  pkg = get_golem_wd(),
  dir = "inst/app/www",
  open = FALSE,
  dir_create = TRUE
)

use_external_css_file(
  url,
  name = NULL,
  pkg = get_golem_wd(),
  dir = "inst/app/www",
  open = FALSE,
```

```
  dir_create = TRUE
)

use_external_html_template(
  url,
  name = "template.html",
  pkg = get_golem_wd(),
  dir = "inst/app/www",
  open = FALSE,
  dir_create = TRUE
)

use_external_file(
  url,
  name = NULL,
  pkg = get_golem_wd(),
  dir = "inst/app/www",
  open = FALSE,
  dir_create = TRUE
)

use_internal_js_file(
  path,
  name = NULL,
  pkg = get_golem_wd(),
  dir = "inst/app/www",
  open = FALSE,
  dir_create = TRUE
)

use_internal_css_file(
  path,
  name = NULL,
  pkg = get_golem_wd(),
  dir = "inst/app/www",
  open = FALSE,
  dir_create = TRUE
)

use_internal_html_template(
  path,
  name = "template.html",
  pkg = get_golem_wd(),
  dir = "inst/app/www",
  open = FALSE,
  dir_create = TRUE
)
```

```
use_internal_file(
  path,
  name = NULL,
  pkg = get_golem_wd(),
  dir = "inst/app/www",
  open = FALSE,
  dir_create = TRUE
)
```

Arguments

url	String representation of URL for the file to be downloaded
name	The name of the module.
pkg	Path to the root of the package. Default is `get_golem_wd()`.
dir	Path to the dir where the file while be created.
open	Should the created file be opened?
dir_create	Creates the directory if it doesn't exist, default is `TRUE`.
path	String representation of the local path for the file to be implemented (use_file only)

Value

The path to the file, invisibly.

Note

See `?htmltools::htmlTemplate` and `https://shiny.rstudio.com/articles/templates.html` for more information about `htmlTemplate`.

---

use_favicon *Add a favicon to your shinyapp*

Description

This function adds the favicon from `ico` to your shiny app.

Usage

```
use_favicon(path, pkg = get_golem_wd(), method = "curl")

remove_favicon(path = "inst/app/www/favicon.ico")

favicon(
  ico = "favicon",
  rel = "shortcut icon",
  resources_path = "www",
  ext = "ico"
)
```

Arguments

path	Path to your favicon file (.ico or .png)
pkg	Path to the root of the package. Default is get_golem_wd().
method	Method to be used for downloading files, 'curl' is default see utils::download.file().
ico	path to favicon file
rel	rel
resources_path	prefix of the resource path of the app
ext	the extension of the favicon

Value

Used for side-effects.

An HTML tag.

Examples

```
if (interactive()) {
  use_favicon()
  use_favicon(path = "path/to/your/favicon.ico")
}
```

---

use_module_test *Add a test file for a module*

Description

Add a test file for in module, with the new testServer structure.

Usage

```
use_module_test(name, pkg = get_golem_wd(), open = TRUE)
```

Arguments

name	The name of the module.
pkg	Path to the root of the package. Default is get_golem_wd().
open	Should the created file be opened?

Value

Used for side effect. Returns the path invisibly.

use_readme_rmd *Generate a README.Rmd*

Description

Generate a README.Rmd

Usage

```
use_readme_rmd(
  open = rlang::is_interactive(),
  pkg_name = golem::get_golem_name(),
  overwrite = FALSE,
  pkg = golem::get_golem_wd()
)
```

Arguments

open	Open the newly created file for editing? Happens in RStudio, if applicable, or via `utils::file.edit()` otherwise.
pkg_name	The name of the package
overwrite	an optional `logical` flag; if `TRUE`, overwrite existing `README.Rmd`, else throws an error if `README.Rmd` exists
pkg	Path to the root of the package. Default is `get_golem_wd()`.

Value

pure side-effect function that generates template `README.Rmd`

use_recommended_deps *Add recommended elements*

Description

use_recommended_deps Adds `shiny`, `DT`, `attempt`, `glue`, `golem`, `htmltools` to dependencies

use_recommended_tests Adds a test folder and copy the golem tests

Usage

```
use_recommended_deps(
  pkg = get_golem_wd(),
  recommended = c("shiny", "DT", "attempt", "glue", "htmltools", "golem")
)

use_recommended_tests(
  pkg = get_golem_wd(),
  spellcheck = TRUE,
  vignettes = TRUE,
  lang = "en-US",
  error = FALSE
)
```

Arguments

pkg	Path to the root of the package. Default is `get_golem_wd()`.
recommended	A vector of recommended packages.
spellcheck	Whether or not to use a spellcheck test.
vignettes	Logical, `TRUE` to spell check all `rmd` and `rnw` files in the `vignettes/` folder.
lang	Preferred spelling language. Usually either `"en-US"` or `"en-GB"`.
error	Logical, indicating whether the unit test should fail if spelling errors are found. Defaults to `FALSE`, which does not error, but prints potential spelling errors

Value

Used for side-effects.

---

use_utils_ui *Use the utils files*

Description

use_utils_ui Copies the golem_utils_ui.R to the R folder.

use_utils_server Copies the golem_utils_server.R to the R folder.

Usage

```
use_utils_ui(pkg = get_golem_wd(), with_test = FALSE)

use_utils_test_ui(pkg = get_golem_wd())

use_utils_server(pkg = get_golem_wd(), with_test = FALSE)

use_utils_test_ui(pkg = get_golem_wd())

use_utils_test_server(pkg = get_golem_wd())
```

Arguments

pkg	Path to the root of the package. Default is get_golem_wd().
with_test	should the module be created with tests?

Value

Used for side-effects.

---

with_golem_options *Add Golem options to a Shiny App*

Description

You'll probably never have to write this function as it is included in the golem template created on launch.

Usage

```
with_golem_options(
  app,
  golem_opts,
  maintenance_page = golem::maintenance_page,
  print = FALSE
)
```

Arguments

app	the app object.
golem_opts	A list of options to be added to the app
maintenance_page	an html_document or a shiny tag list. Default is golem template.
print	Whether or not to print the app. Default is to FALSE, which should be what you need 99.99% of the time. In case you need to actively print() the app object, you can set it to TRUE.

Value

a shiny.appObj object

Index