

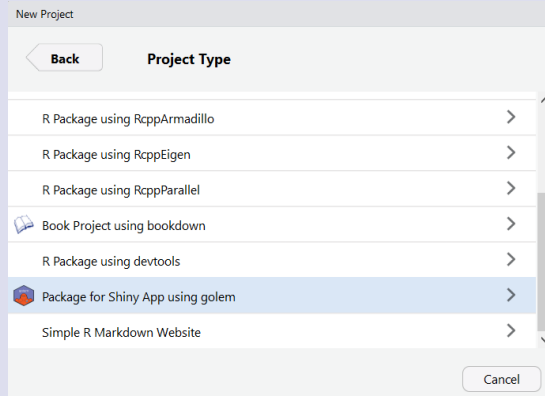
golem :: A Framework for Building Robust Shiny Apps

Create, maintain & deploy a packaged Shiny Application



1. Create a golem

With RStudio:
File → New Project → New Directory → Package for Shiny App using golem



Using the command line:

```
golem::create_golem( path = "~/appdemo" )
```

Creates a golem at '~/appdemo'.

2. Set up your golem with dev/01_start.R

```
golem::fill_desc( pkg_name = "appdemo", ... )
```

Fills the package DESCRIPTION with the author information, the application title & description, links...

```
golem::set_golem_options()
```

Sets {golem} global options.

```
golem::use_recommended_tests()
```

Creates a test template for your app.

```
golem::use_recommended_deps()
```

Adds {shiny}, {DT}, {attempt}, {glue}, {htmltools}, and {golem} as dependencies.

```
golem::use_favicon( path = "path/to/favicon.ico" )
```

Changes the default favicon.

```
golem::use_utils_ui()
```

Creates 'R/golem_utils_ui.R', with UI-related helper functions.

```
golem::use_utils_server()
```

Creates 'R/golem_utils_server.R', with server-related helper functions.

3. Day-to-day dev with golem

A. Look at your golem

- Launch your app with `dev/run_dev.R`:

```
options(golem.app.prod = FALSE)
```

Sets the prod or dev mode. (see ?golem::app_dev)

```
golem::detach_all_attached()
```

Detaches all loaded packages and cleans your environment.

```
golem::document_and_reload()
```

Documents and reloads your package.

```
appdemo::run_app()
```

Launches your application.

B. Customise your golem with dev/02_dev.R

- Edit `R/app_ui.R` & `R/app_server.R`

'R/app_ui.R' & 'R/app_server.R' hold the UI and server logic of your app. You can edit them directly, or add elements created with golem (e.g. modules).

- Add shiny modules

```
golem::add_module( name = "example" )
```

Creates 'R/mod_example.R', with `mod_example_ui` and `mod_example_server` functions inside.

- Add external files

```
golem::add_js_file( "script" )
```

Creates 'inst/app/www/script.js'.

```
golem::add_js_handler( "script" )
```

Creates 'inst/app/www/script.js' with a skeleton for shiny custom handlers.

```
golem::add_css_file( "custom" )
```

Creates 'inst/app/www/custom.css'.

- Use golem built-in JavaScript functions

```
golem::activate_js()
```

Activates the built-in JavaScript functions. To be inserted in the UI.

```
golem::invoke_js("jsfunction", ns("ref_ui"))
```

Invokes from the server any JS function: built-in golem JS functions or custom ones created with `add_js_handler()`

4. Exhibit your golem

Locally

```
remotes::install_local()
```

Installs your golem locally like any other package.

To Rstudio products

```
golem::add_rstudioconnect_file()
```

Creates an app.R file, ready to be deployed to RStudio Connect.

```
golem::add_shinyappsio_file()
```

Creates an app.R file, ready to be deployed to shinyapps.io.

```
golem::add_shinyserver_file()
```

Creates an app.R file, ready to be deployed to Shiny Server.

With Docker

```
golem::add_dockerfile()
```

Creates a Dockerfile that can launch your app.

```
golem::add_dockerfile_shinyproxy()
```

Creates a Dockerfile for ShinyProxy.

```
golem::add_dockerfile_heroku()
```

Creates a Dockerfile for Heroku.

Tips and tricks

```
golem::print_dev( "text" )
```

Prints `text` in your console if `golem::app_dev()` is `TRUE`.

```
golem::make_dev( function )
```

Makes `function` depend on `golem::app_dev()` being `TRUE`.

```
golem::browser_button()
```

Creates a backdoor to your app (see ?golem::browser_button).

- How to make a `run_dev` script for a specific module:

```
golem::detach_all_attached()
golem::document_and_reload()

ui <- mod_example_ui( "my_module" )
server <- function(input,output,session){
  callModule( mod_example_server, "my_module", session )
}
shinyApp(ui, server)
```

Keep in mind that a golem is a package. Everything you know about package development works with your packaged Shiny App created with {golem}! (documentation, tests, CI & CD, ...)