THINKR

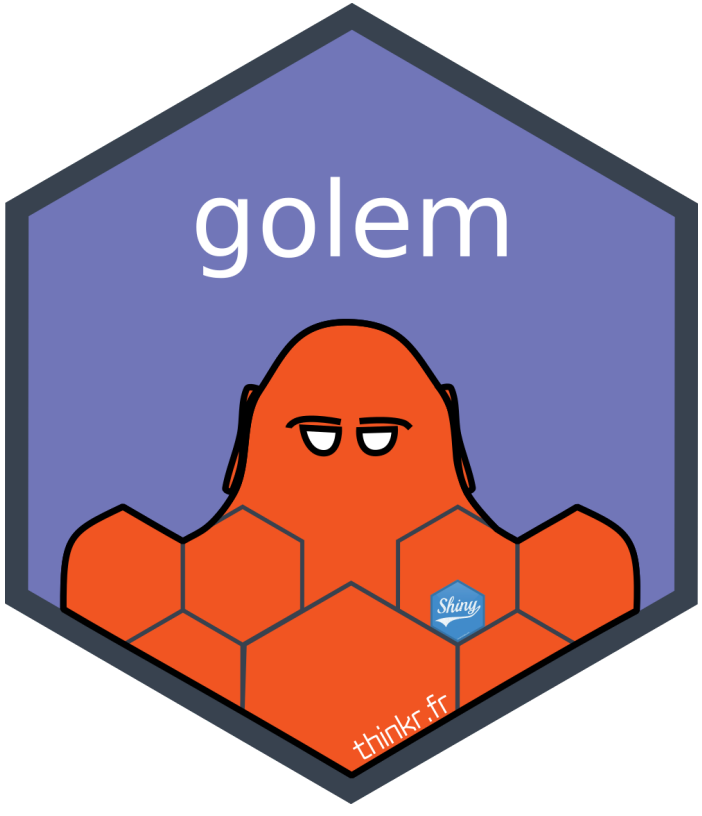
{golem}

A Framework for Building Robust & Production Ready Shiny Apps

Vincent Guyader

Data Scientist, R expert.

- https://rtask.thinkr.fr
- https://github.com/ThinkR-open
- https://twitter.com/thinkr_fr



> create, maintain and deploy easily a shiny app.

Why using Golem ?

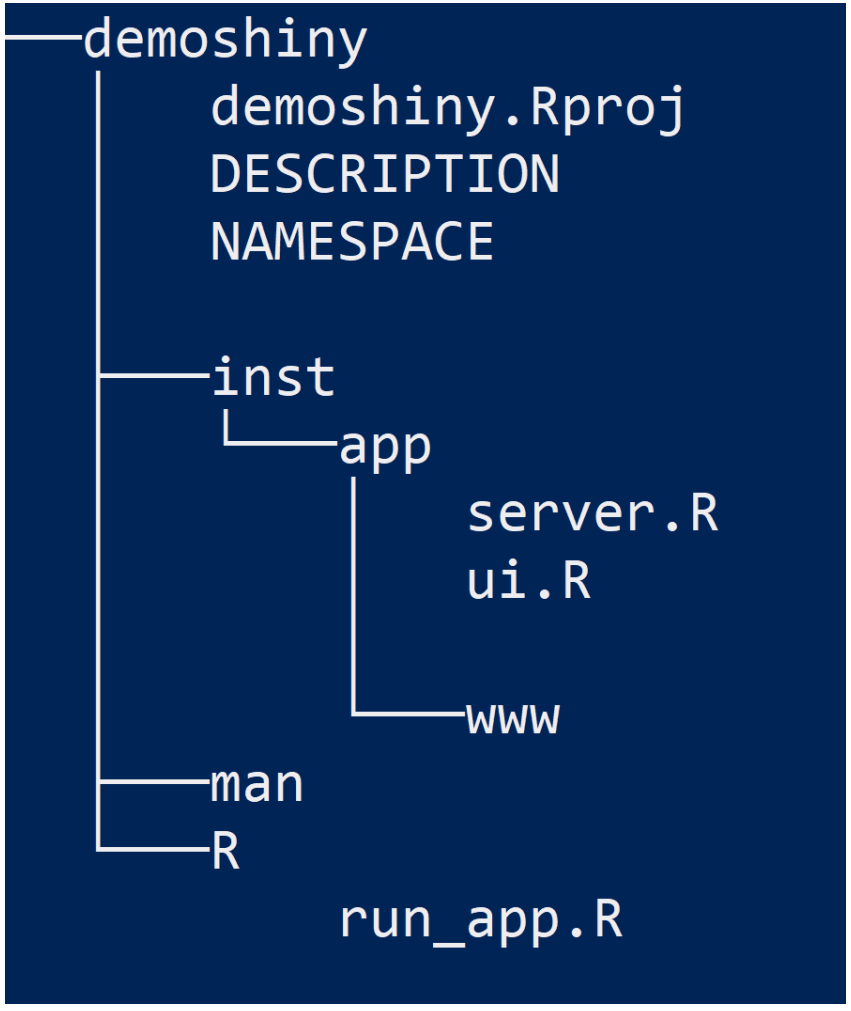
- Saving time
- Working cleanly
- Working with others
- Simplify application deployment
- Facilitate its maintenance
- Having a nice documentation

In R, everything is (must be) a package!

Create a package in 6 minutes and Dockerise your shiny application

- Dependency management.
- Version management.
- Easy installation and deployment.
- Documentation management

In R, everything is (must be) a package!

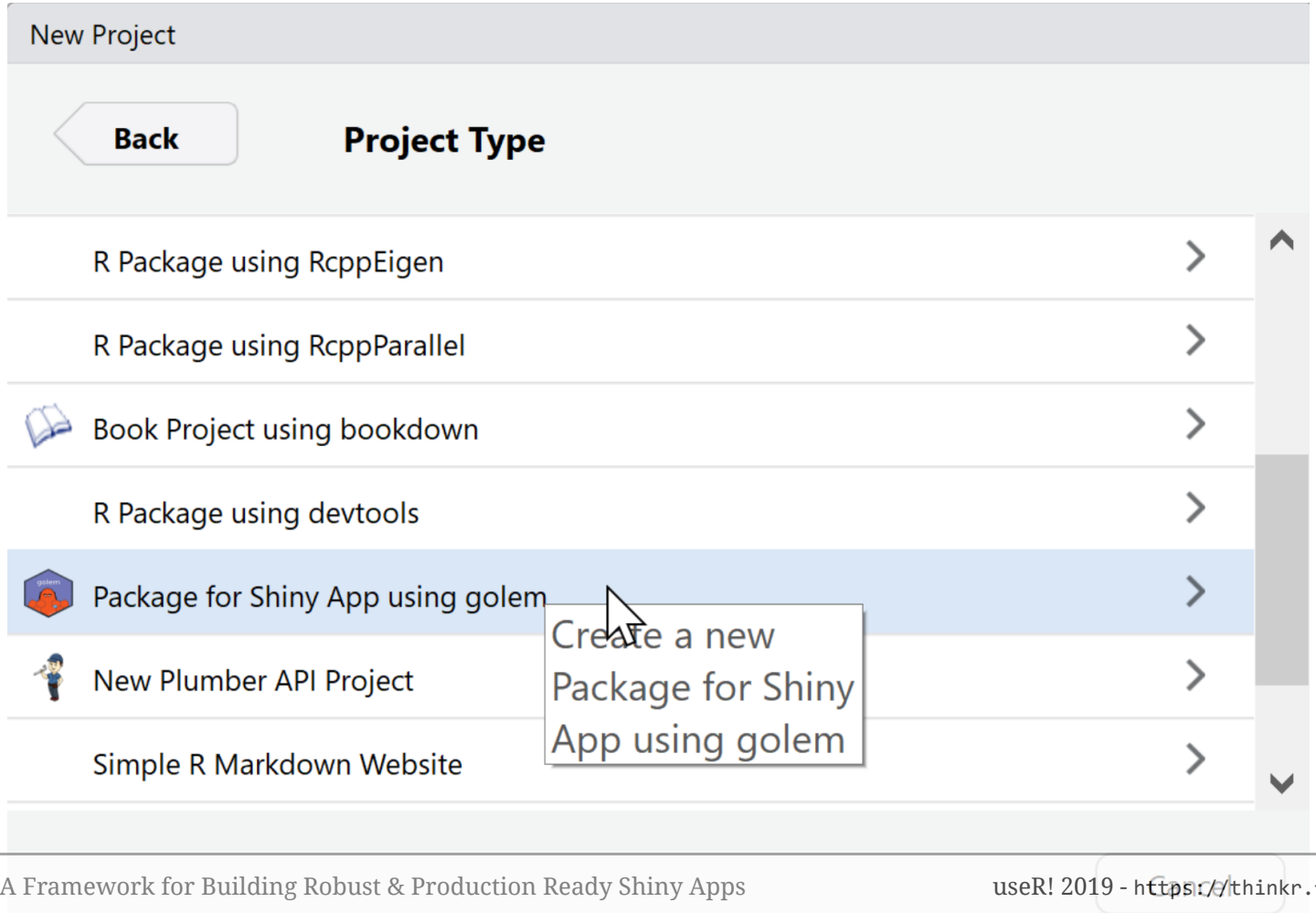


In R, everything is (must be) a package!


```
run_app <- function() {
  appdir <- system.file("app", package = "demoshiny")
  shiny::runApp(appdir, display.mode = "normal")
}
```

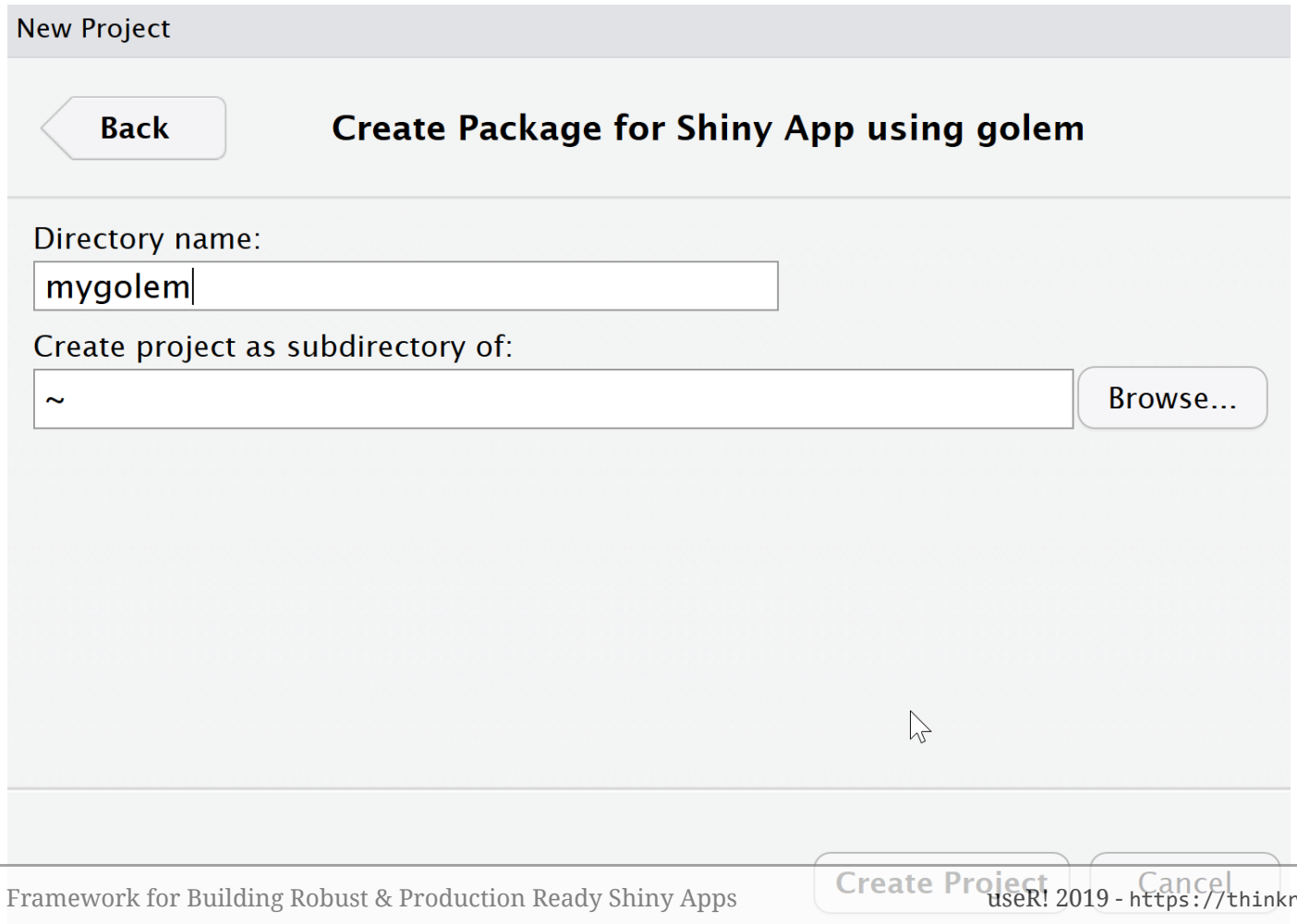
Let's start

```
remotes::install_github("thinkr-open/golem")
```

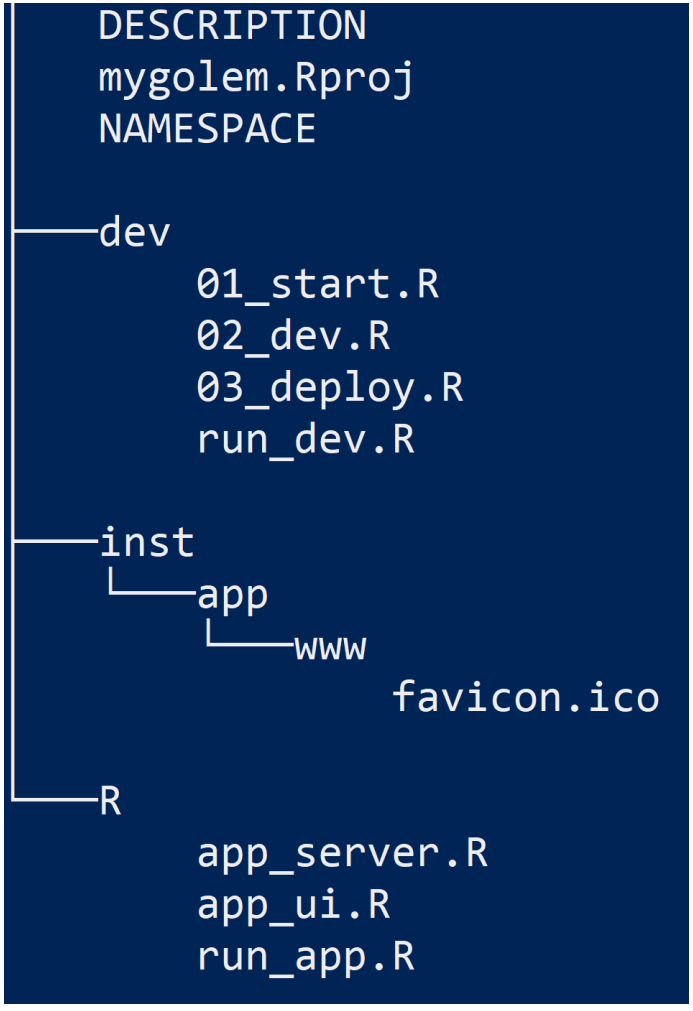


Let's start

```
remotes::install_github("thinkr-open/golem")
```



Let's start


DESCRIPTION
mygolem.Rproj
NAMESPACE
dev
01_start.R
02_dev.R
03_deploy.R
run_dev.R
inst
app
www
favicon.ico
R
app_server.R
app_ui.R
run_app.R

R/app_ui.R


```r
#' @import shiny
app_ui <- function() {
  tagList(
    golem_add_external_resources(),
          fluidPage(
            h1("demogolem")
            )
    )
}
```

```r
#' @import shiny
golem_add_external_resources <- function() {
  addResourcePath('www', system.file('app/www', package = 'demogolem'))
  tags$head(
    golem::activate_js(),
    golem::favicon()
    #tags$link(rel="stylesheet", type="text/css", href="www/custom.css")
    )
}
```

R/app_server.R


```r
#' @import shiny
app_server <- function(input, output,session) {
# List the first level callModules here
}
```

R/run_app.R

```
#' Run the Shiny Application
#'
#' @export
#' @importFrom shiny shinyApp
#' @importFrom golem with_golem_options
run_app <- function(...) {
  with_golem_options(
    app = shinyApp(ui = app_ui, server = app_server),
    golem_opts = list(...)
  )
}
```

To lauch your application :

```
remotes::install_local() # install your golem from sources
mygolem::run_app() # Launch your golem
```

dev/run_dev.R

This script allows you to quickly rebuild and display your golem.

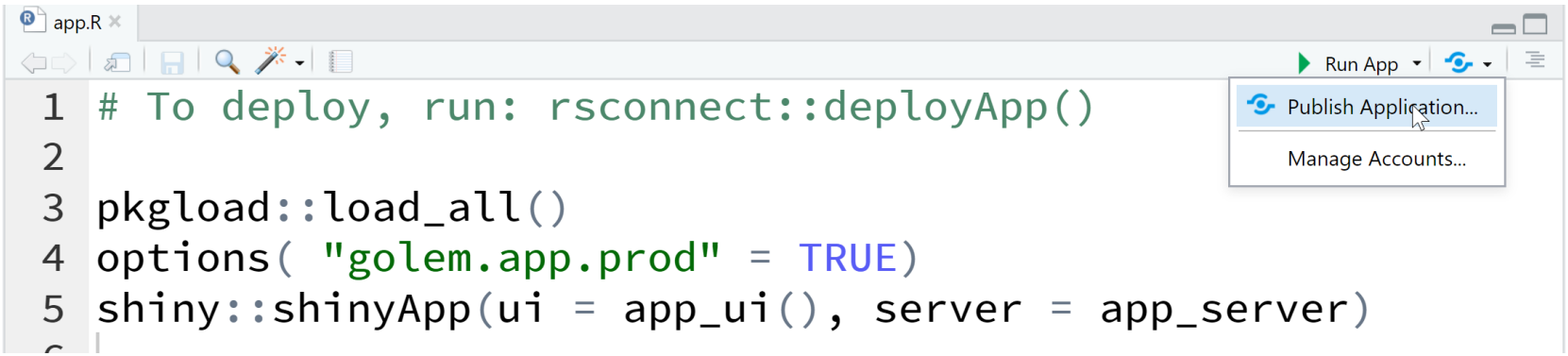
```
# Detach all loaded packages and clean your environment
golem::detach_all_attached()
# rm(list=ls(all.names = TRUE))

# Document and reload your package
golem::document_and_reload()

# Run the application
mygolem::run_app()
```

Deployment

{golem} contains a set of tools that make it easy to deploy.

to : Rstudio Connect, shinyproxy, shiny server, heroku, ...

```
golem::add_dockerfile()
golem::add_dockerfile_heroku()
golem::add_dockerfile_shinyproxy()

golem::add_rstudioconnect_file()
golem::add_shinyappsio_file()
golem::add_shinyserveur_file()
```



dev/01_start.R : fill_desc()

To initialize your golem file 01_start.R contains a succession of instructions to be launched

- fill the DESCRIPTION file

```
golem::fill_desc(
pkg_name = , # The Name of the package containing the App
pkg_title = , # The Title of the package containing the App
pkg_description = , # The Description of the package containing the App
author_first_name = , # Your First Name
author_last_name = , # Your Last Name
author_email = , # Your Email
repo_url = NULL) # The (optional) URL of the GitHub Repo
```

- add dependencies

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

dev/01_start.R : usethis::

It is pre-filled with the most common {usethis} calls

```
usethis::use_mit_license(name = "Your Name")
usethis::use_readme_rmd()
usethis::use_code_of_conduct()
usethis::use_lifecycle_badge("Experimental")
usethis::use_news_md()
```

dev/01_start.R : use_utils_ui()

Preload useful UI functions into your goelm

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

Example :

```
# @examples
# rep_br(5)
rep_br <- function(times = 1) {
  HTML(rep("<br/>", times = times))
} #@examples
# enurl("https://www.thinkr.fr", "ThinkR")
enurl <- function(url, text) {
  tags$a(href = url, text)
}
```

dev/01_start.R : use_utils_server()

Preload useful server functions into your golem

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

Example :

```
`%not_in%` <- Negate(`%in%`)
not_null <- Negate(is.null)
not_na <- Negate(is.na)

# Removes the null from a vector
drop_nulls <- function(x) {
  x[!sapply(x, is.null)]
}

"%||%" <- function(x, y) {
  if (is.null(x)) { y
  } else { x }
}
```

dev/02_dev.R : add_js() & addcss()

A big ambitious application often requires the use of javascript functions and personalized css

{golem} allow you to easily add such files to your project.

```
golem::add_js_file("script")
golem::add_js_handler("script")
golem::add_css_file("custom")
```

dev/02_dev.R : add_module()

{golem} encourages the use of Shiny modules. The `add_module` function allows you to quickly and efficiently add a module to your golem.

Why modules? Because modules...

- divide your application into small pieces
- save you from NAMESPACE conflict(s)
- are reusable
- keep up from having files which are too long

dev/02_dev.R : add_module()

{golem} encourages the use of Shiny modules. The `add_module` function allow you to quickly and efficiently add a module to your golem.

```
golem::add_module("clock")
```

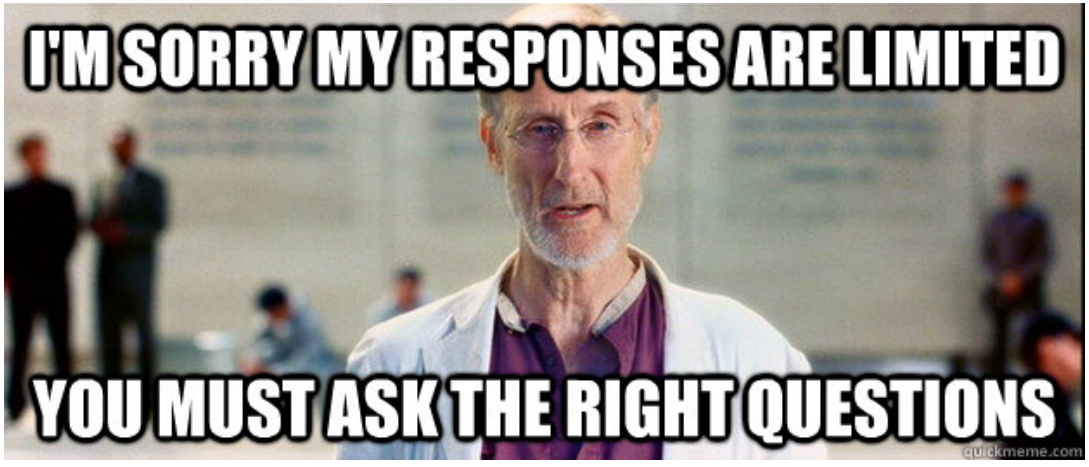
```
mod_clock_ui <- function(id){
  ns <- NS(id)
  tagList(
  )
}

mod_clock_server <- function(input, output, session){

  }

## To be copied in the UI
# mod_clock_ui("clock_ui_1")

## To be copied in the server
# callModule(mod_clock_server, "clock_ui_1")
```

Your Golem is a package

Dont forget that your golem is a package, so you can (need to) use :

- documentation
- unit testing
- continuous integration
- continuous deployment
- ...

Merci !



To find me:

- vincent@thinkr.fr
- http://twitter.com/vincentguyader
- http://twitter.com/thinkr_fr
- https://rtask.thinkr.fr/
- https://thinkr.fr/

To go deeper :

- building-shiny-apps-workflow
- {golem}
- {shinypsum}
- {fakir}
- {shinysnippets}