Package 'shinymgr'

May 10, 2024

Type Package

Title A Framework for Building, Managing, and Stitching 'shiny' Modules into Reproducible Workflows

Version 1.1.0

Maintainer Laurence Clarfeld <laurence.clarfeld@uvm.edu>

Description A unifying framework for managing and deploying 'shiny' applications that consist of modules, where an ``app'' is a tab-based workflow that guides a user step-by-step through an analysis. The 'shinymgr' app builder ``stitches'' 'shiny' modules together so that outputs from one module serve as inputs to the next, creating an analysis pipeline that is easy to implement and maintain. Users of 'shinymgr' apps can save analyses as an RDS file that fully reproduces the analytic steps and can be ingested into an R Markdown report for rapid reporting. In short, developers use the 'shinymgr' framework to write modules and seamlessly combine them into 'shiny' apps, and users of these apps can execute reproducible analyses that can be incorporated into reports for rapid dissemination.

License GPL-3

Encoding UTF-8

URL https://code.usgs.gov/vtcfwru/shinymgr

BugReports https://code.usgs.gov/vtcfwru/shinymgr/-/issues

LazyData true

RoxygenNote 7.3.1

expressed or implied, is made by the USGS or the U.S. Government as to the accuracy and functioning of the program and related program material nor shall the fact of distribution constitute any such warranty, and no responsibility is assumed by the USGS in connection therewith. This software is provided ``AS IS.''

Depends R (>= 3.5.0), shiny

Imports DBI, reactable, renv, RSQLite, shinyjs, shinydashboard

Suggests fs, learnr, shinytest, testthat

Config/testthat/edition 3

NeedsCompilation no

Author Laurence Clarfeld [aut, cre] (<https://orcid.org/0000-0002-3927-9411>), Caroline Tang [aut] (<https://orcid.org/0000-0001-7966-5854>), Therese Donovan [aut, org, rth] (<https://orcid.org/0000-0001-8124-9251>)

Repository CRAN

Date/Publication 2024-05-10 17:50:03 UTC

R topics documented:

`appReports` *Sample data for the shinymgr.sqlite table, "appReports"*

Description

Sample data imported to the shinymgr SQLite database by the function `shiny_db_populate`.

Usage

```
data(demo_data)
```

Format

A data frame with 3 observations on the following 3 variables.

Details

Reports are added to the shinymgr.sqlite database via the "Add "Reports" menu in shinymgr's Developer section.

Fields

`fkAppName` References an appName from the "apps" table.

`fkReportName` Reference a report from the "reports" table.

`notes` Notes about a report for a given app.

See Also

Other data: `appStitching`, `appTabs`, `apps`, `modFunctionArguments`, `modFunctionReturns`, `modPackages`, `modules`, `reports`, `tabModules`, `tabs`

Examples

```
# read in the demo data
data("demo_data")

# look at the structure
str(appReports)
```

apps *Sample data for the shinymgr.sqlite table, "apps"*

Description

Sample data imported to the shinymgr SQLite database by the function `shiny_db_populate`.

Usage

```
data(demo_data)
```

Format

A data frame with 3 observations on the following 10 variables.

Details

New records to the "apps" table are added to the shinymgr.sqlite database via the "App Builder" interface within shinymgr's Developer section.

Fields

`pkAppName` Name of the app; primary key.

`appDisplayName` Name that is displayed on app.

`appDescription` A description of the app.

`appVideoURL` A link to a video if desired.

`appCSS` The css file to style the app.

`appNotes` Developer notes about the app.

`appActive` Logical. Is the app active?

`fkParentAppName` References previous version of app.

`appCitation` The official citation of the app.

See Also

Other data: `appReports`, `appStitching`, `appTabs`, `modFunctionArguments`, `modFunctionReturns`, `modPackages`, `modules`, `reports`, `tabModules`, `tabs`

Examples

```
# read in the demo data
data(demo_data)

# look at the structure
str(apps)
```

appStitching *Sample data for the shinymgr.sqlite table, "appStitching"*

Description

Sample data imported to the shinymgr SQLite database by the function shiny_db_populate.

Usage

```
data(demo_data)
```

Format

A data frame with 30 observations on the following 6 variables.

Details

appSTitching records are added to the shinymgr.sqlite database via the "App Builder" interface within shinymgr's Developer section.

Fields

pkStitchID Auto-number primary key.

fkAppName References pkAppName in the "apps" table.

fkInstanceID References pkInstanceID in the "tabModules" table.

fkModArgID Reference pkModArgID in the "modFunctionArguments" table.

fkModReturnID Reference pkModReturnID in the "modFunctionReturns" table.

fkStitchID References the stitch that preceeds a given stitch (record).

See Also

Other data: appReports, appTabs, apps, modFunctionArguments, modFunctionReturns, modPackages, modules, reports, tabModules, tabs

Examples

```
# read in the demo data
data("demo_data")

# look at the structure
str(appStitching)
```

appTabs *Sample data for the shinymgr.sqlite table, "appTabs"*

Description

Sample data imported to the shinymgr SQLite database by the function shiny_db_populate.

Usage

```
data(demo_data)
```

Format

A data frame with 9 observations on the following 3 variables.

Details

Records to the "appTabs" table are added to the shinymgr.sqlite database via the "App Builder" interface within shinymgr's Developer section.

Fields

`fkTabName` References pkTabName in the "tabs" table.

`fkAppName` References pkAppName in the "apps" table.

`tabOrder` Specifies the order of tabs as presented to user.

See Also

Other data: appReports, appStitching, apps, modFunctionArguments, modFunctionReturns, modPackages, modules, reports, tabModules, tabs

Examples

```
# read in the demo data
data(demo_data)

# look at the structure
str(appTabs)
```

check_mod_info *Compares mod header information to the database*

Description

This function checks that mod header information matches what's in the database to ensure that modules will be called and stitched correctly.

Usage

```
check_mod_info(modName, shinyMgrPath, verbose = TRUE)
```

Arguments

modName	The name of the module
shinyMgrPath	The path to the shinymgr folder.
verbose	Whether to print updates to the console (default = TRUE)

Value

A list containing dataframes of logicals indicating whether fields are consistent between the module script header and the database. These include: 1. Data for the modules table 2. Data for the modFunctionArguments table 3. Data for the modFunctionReturns table A value of TRUE indicates that the fields match, and FALSE indicates a mismatch.

More Info

The check_mod_info() function is described in the "shinymgr_modules" tutorial.

Tutorials

The shinymgr learnr tutorials include, in order:

1. `learnr::run_tutorial(name = "intro", package = "shinymgr")`
2. `learnr::run_tutorial(name = "shiny", package = "shinymgr")`
3. `learnr::run_tutorial(name = "modules", package = "shinymgr")`
4. `learnr::run_tutorial(name = "app_modules", package = "shinymgr")`
5. `learnr::run_tutorial(name = "tests", package = "shinymgr")`
6. `learnr::run_tutorial(name = "shinymgr", package = "shinymgr")`
7. `learnr::run_tutorial(name = "database", package = "shinymgr")`
8. `learnr::run_tutorial(name = "shinymgr_modules", package = "shinymgr")`
9. `learnr::run_tutorial(name = "apps", package = "shinymgr")`
10. `learnr::run_tutorial(name = "analyses", package = "shinymgr")`
11. `learnr::run_tutorial(name = "reports", package = "shinymgr")`
12. `learnr::run_tutorial(name = "deployment", package = "shinymgr")`

See Also

Other module: mod_header_parser(), mod_init(), mod_register()

Examples

```
# establish shinyMgrPath
parentPath <- tempdir()
shinyMgrPath <- paste0(parentPath, '/shinymgr')

# Create a demo database
shinymgr_setup(parentPath = parentPath, demo = TRUE)

#check info for different modules
check_mod_info(modName = "subset_rows", shinyMgrPath = shinyMgrPath)

check_mod_info(modName = "add_noise", shinyMgrPath = shinyMgrPath)

# Remove demo database
unlink(shinyMgrPath, recursive = TRUE)
```

---

delete_app *Deletes an app from the database*

Description

Deletes an app (and associated files if requested) from the shinymgr.sqlite database

Usage

```
delete_app(appName, shinyMgrPath, fileDelete = FALSE)
```

Arguments

appName	The name of the app to be deleted
shinyMgrPath	The path to the shinymgr project. Typically the working directory.
fileDelete	TRUE/FALSE, whether the app script should also be deleted - defaults to FALSE.

Value

An integer value with the total number of rows deleted (including cascades)

More Info

The delete_app() function is described in the "app_modules" tutorial.

Tutorials

The shinymgr learnr tutorials include, in order:

1. `learnr::run_tutorial(name = "intro", package = "shinymgr")`
2. `learnr::run_tutorial(name = "shiny", package = "shinymgr")`
3. `learnr::run_tutorial(name = "modules", package = "shinymgr")`
4. `learnr::run_tutorial(name = "app_modules", package = "shinymgr")`
5. `learnr::run_tutorial(name = "tests", package = "shinymgr")`
6. `learnr::run_tutorial(name = "shinymgr", package = "shinymgr")`
7. `learnr::run_tutorial(name = "database", package = "shinymgr")`
8. `learnr::run_tutorial(name = "shinymgr_modules", package = "shinymgr")`
9. `learnr::run_tutorial(name = "apps", package = "shinymgr")`
10. `learnr::run_tutorial(name = "analyses", package = "shinymgr")`
11. `learnr::run_tutorial(name = "reports", package = "shinymgr")`
12. `learnr::run_tutorial(name = "deployment", package = "shinymgr")`

References

https://code.usgs.gov/vtcfwru/shinymgr

See Also

Other delete: `delete_mod()`, `delete_report()`

`delete_mod` *Deletes a module from the database*

Description

Deletes a module (and associated files if requested) from the shinymgr.sqlite database

Usage

```
delete_mod(modName, shinyMgrPath, fileDelete = FALSE,
verbose = TRUE)
```

Arguments

`modName`	The name of the module to be deleted, character string.
`shinyMgrPath`	The path to the shinymgr project. Typically the working directory.
`fileDelete`	TRUE/FALSE, whether the module script should also be deleted - defaults to FALSE.
`verbose`	Whether to print updates to the console (default = TRUE)

Value

An integer value with the total number of rows deleted (including cascades)

More Info

The delete_mod() function is described in the "shinymgr_modules" tutorial.

Tutorials

The shinymgr learnr tutorials include, in order:

1. `learnr::run_tutorial(name = "intro", package = "shinymgr")`
2. `learnr::run_tutorial(name = "shiny", package = "shinymgr")`
3. `learnr::run_tutorial(name = "modules", package = "shinymgr")`
4. `learnr::run_tutorial(name = "app_modules", package = "shinymgr")`
5. `learnr::run_tutorial(name = "tests", package = "shinymgr")`
6. `learnr::run_tutorial(name = "shinymgr", package = "shinymgr")`
7. `learnr::run_tutorial(name = "database", package = "shinymgr")`
8. `learnr::run_tutorial(name = "shinymgr_modules", package = "shinymgr")`
9. `learnr::run_tutorial(name = "apps", package = "shinymgr")`
10. `learnr::run_tutorial(name = "analyses", package = "shinymgr")`
11. `learnr::run_tutorial(name = "reports", package = "shinymgr")`
12. `learnr::run_tutorial(name = "deployment", package = "shinymgr")`

References

https://code.usgs.gov/vtcfwru/shinymgr

See Also

Other delete: `delete_app()`, `delete_report()`

`delete_report` *Deletes a report from the database*

Description

Deletes a report (and associated file if requested) from the shinymgr.sqlite database

Usage

```
delete_report(reportName, shinyMgrPath, fileDelete = FALSE)
```

Arguments

reportName	The name of the report to be deleted, character string.
shinyMgrPath	The path to the shinymgr project. Typically the working directory.
fileDelete	TRUE/FALSE, whether the report .Rmd file should also be deleted - defaults to FALSE.

Value

An integer value with the total number of rows deleted (including cascades)

More Info

The delete_report() function is described in the "reports" tutorial.

Tutorials

The shinymgr learnr tutorials include, in order:

1. `learnr::run_tutorial(name = "intro", package = "shinymgr")`
2. `learnr::run_tutorial(name = "shiny", package = "shinymgr")`
3. `learnr::run_tutorial(name = "modules", package = "shinymgr")`
4. `learnr::run_tutorial(name = "app_modules", package = "shinymgr")`
5. `learnr::run_tutorial(name = "tests", package = "shinymgr")`
6. `learnr::run_tutorial(name = "shinymgr", package = "shinymgr")`
7. `learnr::run_tutorial(name = "database", package = "shinymgr")`
8. `learnr::run_tutorial(name = "shinymgr_modules", package = "shinymgr")`
9. `learnr::run_tutorial(name = "apps", package = "shinymgr")`
10. `learnr::run_tutorial(name = "analyses", package = "shinymgr")`
11. `learnr::run_tutorial(name = "reports", package = "shinymgr")`
12. `learnr::run_tutorial(name = "deployment", package = "shinymgr")`

References

https://code.usgs.gov/vtcfwru/shinymgr

See Also

Other delete: `delete_app()`, `delete_mod()`

launch_shinymgr *Launch the master app for shinymgr*

Description

Launches the master app for shinymgr

Usage

```
launch_shinymgr(shinyMgrPath, ...)
```

Arguments

shinyMgrPath	Filepath to the main shinymgr folder.
...	Additional arguments to be passed to the app.

Value

No return value, function launches shiny app

More Info

The launch_shinymgr() function is described in the "shinymgr" tutorial.

Tutorials

The shinymgr learnr tutorials include, in order:

1. `learnr::run_tutorial(name = "intro", package = "shinymgr")`
2. `learnr::run_tutorial(name = "shiny", package = "shinymgr")`
3. `learnr::run_tutorial(name = "modules", package = "shinymgr")`
4. `learnr::run_tutorial(name = "app_modules", package = "shinymgr")`
5. `learnr::run_tutorial(name = "tests", package = "shinymgr")`
6. `learnr::run_tutorial(name = "shinymgr", package = "shinymgr")`
7. `learnr::run_tutorial(name = "database", package = "shinymgr")`
8. `learnr::run_tutorial(name = "shinymgr_modules", package = "shinymgr")`
9. `learnr::run_tutorial(name = "apps", package = "shinymgr")`
10. `learnr::run_tutorial(name = "analyses", package = "shinymgr")`
11. `learnr::run_tutorial(name = "reports", package = "shinymgr")`
12. `learnr::run_tutorial(name = "deployment", package = "shinymgr")`

References

https://code.usgs.gov/vtcfwru/shinymgr

Examples

```
## Only run this example in interactive R sessions
if (interactive()) {

  # set the directory path that will house the shinymgr project
  parentPath <- tempdir()
  shinyMgrPath <- paste0(parentPath, '/shinymgr')

  # set up raw directories and fresh database
  shinymgr_setup(parentPath, demo = TRUE)

  # The shiny app
  launch_shinymgr(shinyMgrPath)

  # Accepts args to shiny::runApp
  launch_shinymgr(shinyMgrPath, quiet = TRUE)

  # remove demo
  unlink(shinyMgrPath, recursive = TRUE)

}
```

---

modFunctionArguments *Sample data for the shinymgr.sqlite table, "modFunctionArguments"*

Description

Sample data imported to the shinymgr SQLite database by the function `shiny_db_populate`.

Usage

```
data(demo_data)
```

Format

A data frame with 6 observations on the following 5 variables.

Details

New records to "modFunctionArguments" table are added to the shinymgr.sqlite database via the `mod_register` function.

Fields

`pkModArgID` Auto-number primary key.

`fkModuleName` References pkModuleName in the "modules" table.

`functionArgName` Name of the argument.

`functionArgClass` Class of the argument (e.g., data.frame)

`description` Description of the argument.

See Also

Other data: `appReports`, `appStitching`, `appTabs`, `apps`, `modFunctionReturns`, `modPackages`, `modules`, `reports`, `tabModules`, `tabs`

Examples

```
# read in the demo data
data(demo_data)

# look at the structure
str(modFunctionArguments)
```

---

modFunctionReturns *Sample data for the shinymgr.sqlite table, "modFunctionReturns"*

Description

Sample data imported to the shinymgr SQLite database by the function `shiny_db_populate`.

Usage

```
data(demo_data)
```

Format

A data frame with 6 observations on the following 5 variables.

Details

New records to "modFunctionReturns" table are added to the shinymgr.sqlite database via the `mod_register` function.

Fields

`pkModReturnID` References pkModuleName in the "modules" table.

`fkModuleName` References pkModuleName in the "modules" table.

`functionReturnName` Name of the return.

`functionReturnClass` Class of the return (e.g., data.frame)

`description` Description of the return.

See Also

Other data: `appReports`, `appStitching`, `appTabs`, `apps`, `modFunctionArguments`, `modPackages`, `modules`, `reports`, `tabModules`, `tabs`

Examples

```
# read in the demo data
data(demo_data)

# look at the structure
str(modFunctionReturns)
```

---

modPackages *Sample data for the shinymgr.sqlite table, "modPackages"*

Description

Sample data imported to the shinymgr SQLite database by the function `shiny_db_populate`.

Usage

```
data(demo_data)
```

Format

A data frame with 2 observations on the following 4 variables.

Details

New records to "modPackages" table are added to the shinymgr.sqlite database via the `mod_register` function.

Fields

`fkModuleName` References pkModuleName in the "modules" table.

`packageName` Name of the R package

`version` Version of the package.

`notes` Notes about the package.

See Also

Other data: `appReports`, `appStitching`, `appTabs`, `apps`, `modFunctionArguments`, `modFunctionReturns`, `modules`, `reports`, `tabModules`, `tabs`

Examples

```
# read in the demo data
data(demo_data)

# look at the structure
str(modPackages)
```

---

modules *Sample data for the shinymgr.sqlite table, "modules"*

Description

Sample data imported to the shinymgr SQLite database by the function `shiny_db_populate`.

Usage

```
data(demo_data)
```

Format

A data frame with 8 observations on the following 7 variables.

Details

New records to "modules" table are added to the shinymgr.sqlite database via the `mod_register` function.

Fields

`pkModuleName` Name of a module; primary key.

`modDisplayName` Name displayed on the module.

`modDescription` Description of the module.

`modCitation` Citation of the module.

`modNotes` Notes on the module.

`modActive` Logical. Is the module still active?

`dateCreated` Date the module was created.

See Also

Other data: `appReports`, `appStitching`, `appTabs`, `apps`, `modFunctionArguments`, `modFunctionReturns`, `modPackages`, `reports`, `tabModules`, `tabs`

Examples

```
# read in the demo data
data(demo_data)

# look at the structure
str(modules)
```

`mod_header_parser` *Parse the header of module modules to add to the database*

Description

This is a helper function that parses the header of modules to pending addition to the shinymgr.sqlite database. This is used as a helper function by `mod_register` and `check_mod_info` to convert the data in headers into dataframes.

Usage

```
mod_header_parser(filePath)
```

Arguments

`filePath`	The file path to the R module script to be added.

Value

A list containing dataframes that can be used to update the shinyMgr database. These include: 1. Data for the modules table 2. Data for updating the modFunctionArguments table 3. Data for updating the modFunctionReturns table

See Also

Other module: check_mod_info(), mod_init(), mod_register()

Examples

```
# establish the path to a built-in shinymgr module
filePath <- file.path(find.package('shinymgr'), 'shinymgr/modules/poly_fit.R')

# Parse the header and return associated data as a list of dataframes.
data_to_add <- mod_header_parser(filePath)

# look at the result
str(data_to_add)
```

---

mod_init *Creates an R script that contains a framework for developing a new module*

Description

Creates an R script that contains a framework for developing a new module

Usage

```
mod_init(modName, author, shinyMgrPath)
```

Arguments

modName	The name of the module to be added to the modules table of the shinymgr.sqlite database. The function will write an R script as modName.R
author	A string with the author's name, formatted as "Lastname, Firstname".
shinyMgrPath	The path to the shinymgr project. Typically the working directory.

Value

Invisible. The function will write an R script with the name modName.R and store the file in the shinymgr project's modules folder.

More Info

The mod_init() function is described in the "shinymgr_modules" tutorial.

Tutorials

The shinymgr learnr tutorials include, in order:

1. `learnr::run_tutorial(name = "intro", package = "shinymgr")`
2. `learnr::run_tutorial(name = "shiny", package = "shinymgr")`
3. `learnr::run_tutorial(name = "modules", package = "shinymgr")`
4. `learnr::run_tutorial(name = "app_modules", package = "shinymgr")`
5. `learnr::run_tutorial(name = "tests", package = "shinymgr")`
6. `learnr::run_tutorial(name = "shinymgr", package = "shinymgr")`
7. `learnr::run_tutorial(name = "database", package = "shinymgr")`
8. `learnr::run_tutorial(name = "shinymgr_modules", package = "shinymgr")`
9. `learnr::run_tutorial(name = "apps", package = "shinymgr")`
10. `learnr::run_tutorial(name = "analyses", package = "shinymgr")`
11. `learnr::run_tutorial(name = "reports", package = "shinymgr")`
12. `learnr::run_tutorial(name = "deployment", package = "shinymgr")`

References

https://code.usgs.gov/vtcfwru/shinymgr

See Also

Other module: `check_mod_info()`, `mod_header_parser()`, `mod_register()`

Examples

```
## Only run this example in interactive R sessions
if (interactive()) {

# set the file path to the main shinymgr directory
parentPath <- tempdir()
shinyMgrPath <- paste0(parentPath, '/shinymgr')

shinymgr_setup(parentPath = parentPath, demo = FALSE)

mod_init(
  modName = "my_test_mod",
  author = "Baggins, Bilbo",
  shinyMgrPath = shinyMgrPath
)

# the file should be located in the shinymgr/modules directory
fp <- paste0(shinyMgrPath, "/modules/my_test_mod.R")

# determine if the file exists
file.exists(fp)
```

```
# show the file info
file.info(fp)

# show the file
file.show(fp)

# remove demo
unlink(shinyMgrPath, recursive = TRUE)

}
```

`mod_register` *Register (inserts) a new module into the shinymgr project*

Description

Insert a new record into the shinymgr.sqlite database table "modules" and accompanying tables ("modFunctionArguments", "modFunctionReturns", "modPackages")

Usage

```
mod_register(modName, shinyMgrPath)
```

Arguments

`modName`	Name of the new module
`shinyMgrPath`	Directory that holds the main shinymgr project

Details

This function reads in a module file created by `mod_init` and parses the header using `mod_header_parser` to populate the modules, modFunctionArguments, modFunctionReturns, and modPackages tables of the shinymgr.sqlite database. These tables are referenced in the app builder, so module headers must match the module functions exactly.

Value

Nothing. Records are inserted into shinymgr.sqlite.

More Info

The mod_register() function is described in the "shinymgr_modules" tutorial.

Tutorials

The shinymgr learnr tutorials include, in order:

1. `learnr::run_tutorial(name = "intro", package = "shinymgr")`
2. `learnr::run_tutorial(name = "shiny", package = "shinymgr")`
3. `learnr::run_tutorial(name = "modules", package = "shinymgr")`
4. `learnr::run_tutorial(name = "app_modules", package = "shinymgr")`
5. `learnr::run_tutorial(name = "tests", package = "shinymgr")`
6. `learnr::run_tutorial(name = "shinymgr", package = "shinymgr")`
7. `learnr::run_tutorial(name = "database", package = "shinymgr")`
8. `learnr::run_tutorial(name = "shinymgr_modules", package = "shinymgr")`
9. `learnr::run_tutorial(name = "apps", package = "shinymgr")`
10. `learnr::run_tutorial(name = "analyses", package = "shinymgr")`
11. `learnr::run_tutorial(name = "reports", package = "shinymgr")`
12. `learnr::run_tutorial(name = "deployment", package = "shinymgr")`

References

https://code.usgs.gov/vtcfwru/shinymgr

See Also

mod_init

Other module: check_mod_info(), mod_header_parser(), mod_init()

---

qry_app_flow *Retrieve structure of an app module*

Description

Returns a dataframe showing the ordered layout of a shinymgr app (e.g., tabs, modules, and the order of presentation).

Usage

```
qry_app_flow(appName, shinyMgrPath)
```

Arguments

appName	The name of the app in the shinymgr database (e.g. iris_explorer)
shinyMgrPath	File path to the main shiny manager project directory

Value

Dataframe consisting of the specified rows and columns

More Info

The qry_app_flow() function is described in the "app_modules" tutorial.

Tutorials

The shinymgr learnr tutorials include, in order:

1. `learnr::run_tutorial(name = "intro", package = "shinymgr")`
2. `learnr::run_tutorial(name = "shiny", package = "shinymgr")`
3. `learnr::run_tutorial(name = "modules", package = "shinymgr")`
4. `learnr::run_tutorial(name = "app_modules", package = "shinymgr")`
5. `learnr::run_tutorial(name = "tests", package = "shinymgr")`
6. `learnr::run_tutorial(name = "shinymgr", package = "shinymgr")`
7. `learnr::run_tutorial(name = "database", package = "shinymgr")`
8. `learnr::run_tutorial(name = "shinymgr_modules", package = "shinymgr")`
9. `learnr::run_tutorial(name = "apps", package = "shinymgr")`
10. `learnr::run_tutorial(name = "analyses", package = "shinymgr")`
11. `learnr::run_tutorial(name = "reports", package = "shinymgr")`
12. `learnr::run_tutorial(name = "deployment", package = "shinymgr")`

References

https://code.usgs.gov/vtcfwru/shinymgr

See Also

Other qry: `qry_app_stitching()`, `qry_insert()`, `qry_mod_info()`, `qry_row()`

Examples

```
# set the file path to the main shinymgr directory
parentPath <- tempdir()
shinyMgrPath <- paste0(parentPath, '/shinymgr')

shinymgr_setup(parentPath = parentPath, demo = TRUE)

# get the structure of the iris_explorer app
qry_app_flow(appName = "iris_explorer", shinyMgrPath = shinyMgrPath)

# remove demo
unlink(shinyMgrPath, recursive = TRUE)
```

qry_app_stitching *Retrieve structure of an app module*

Description

Returns a dataframe showing how outputs from one module are "stitched" as inputs to downstream modules in a shinymgr app

Usage

```
qry_app_stitching(appName, shinyMgrPath)
```

Arguments

appName	The name of the app in the shinymgr database (e.g. iris_explorer)
shinyMgrPath	File path to the main shiny manager project directory

Value

Dataframe consisting of the specified rows and columns

Tutorials

The shinymgr learnr tutorials include, in order:

1. `learnr::run_tutorial(name = "intro", package = "shinymgr")`
2. `learnr::run_tutorial(name = "shiny", package = "shinymgr")`
3. `learnr::run_tutorial(name = "modules", package = "shinymgr")`
4. `learnr::run_tutorial(name = "app_modules", package = "shinymgr")`
5. `learnr::run_tutorial(name = "tests", package = "shinymgr")`
6. `learnr::run_tutorial(name = "shinymgr", package = "shinymgr")`
7. `learnr::run_tutorial(name = "database", package = "shinymgr")`
8. `learnr::run_tutorial(name = "shinymgr_modules", package = "shinymgr")`
9. `learnr::run_tutorial(name = "apps", package = "shinymgr")`
10. `learnr::run_tutorial(name = "analyses", package = "shinymgr")`
11. `learnr::run_tutorial(name = "reports", package = "shinymgr")`
12. `learnr::run_tutorial(name = "deployment", package = "shinymgr")`

References

https://code.usgs.gov/vtcfwru/shinymgr

See Also

Other qry: `qry_app_flow()`, `qry_insert()`, `qry_mod_info()`, `qry_row()`

Examples

```
# set the file path to the main shinymgr directory
parentPath <- tempdir()
shinyMgrPath <- paste0(parentPath, '/shinymgr')

shinymgr_setup(parentPath = parentPath, demo = TRUE)

# get the structure of the iris_explorer app
qry_app_stitching(appName = "iris_explorer", shinyMgrPath = shinyMgrPath)

# remove demo
unlink(shinyMgrPath, recursive = TRUE)
```

---

qry_mod_info *Retrieve general information about a module*

Description

Returns a dataframe showing a given module's arguments, returns, and package dependencies.

Usage

```
qry_mod_info(modName, shinyMgrPath)
```

Arguments

modName	The name of the mod in the shinymgr database (e.g. subset_rows)
shinyMgrPath	File path to the main shiny manager project directory

Value

Dataframe consisting of the specified rows and columns

Tutorials

The shinymgr learnr tutorials include, in order:

1. `learnr::run_tutorial(name = "intro", package = "shinymgr")`
2. `learnr::run_tutorial(name = "shiny", package = "shinymgr")`
3. `learnr::run_tutorial(name = "modules", package = "shinymgr")`
4. `learnr::run_tutorial(name = "app_modules", package = "shinymgr")`
5. `learnr::run_tutorial(name = "tests", package = "shinymgr")`
6. `learnr::run_tutorial(name = "shinymgr", package = "shinymgr")`
7. `learnr::run_tutorial(name = "database", package = "shinymgr")`

8. `learnr::run_tutorial(name = "shinymgr_modules", package = "shinymgr")`
9. `learnr::run_tutorial(name = "apps", package = "shinymgr")`
10. `learnr::run_tutorial(name = "analyses", package = "shinymgr")`
11. `learnr::run_tutorial(name = "reports", package = "shinymgr")`
12. `learnr::run_tutorial(name = "deployment", package = "shinymgr")`

References

https://code.usgs.gov/vtcfwru/shinymgr

See Also

Other qry: `qry_app_flow()`, `qry_app_stitching()`, `qry_insert()`, `qry_row()`

Examples

```
# set the file path to the main shinymgr directory
parentPath <- tempdir()
shinyMgrPath <- paste0(parentPath, '/shinymgr')

shinymgr_setup(parentPath = parentPath, demo = TRUE)

# get the details of the "subset_rows" modules
qry_mod_info(modName = "subset_rows", shinyMgrPath = shinyMgrPath)

#' # get the details of the "add_noise" modules
qry_mod_info(modName = "add_noise", shinyMgrPath = shinyMgrPath)

# remove demo
unlink(shinyMgrPath, recursive = TRUE)
```

`qry_row` *Retrieve one or more rows from a specified table from the shinymgr.sqlite. Used internally. database given a set of conditions on one or more columns.*

Description

Returns dataframe containing specified columns and rows from the shinymgr database based on specified conditions.

Usage

```
qry_row(tableName, rowConditions, colConditions, shinyMgrPath)
```

Arguments

tableName	The name of the table of the shinymgr database (e.g. people, apps, etc.).
rowConditions	A dataframe where the keys correspond to columns of the specified dataframe and key values correspond to the equality condition that must be satisfied by any returning rows, else returns all rows (default returns all rows).
colConditions	A vector specifying the names of columns to be returned from the query (default returns all columns).
shinyMgrPath	File path to the main shiny manager project directory

Value

Dataframe consisting of the specified rows and columns

Tutorials

The shinymgr learnr tutorials include, in order:

1. `learnr::run_tutorial(name = "intro", package = "shinymgr")`
2. `learnr::run_tutorial(name = "shiny", package = "shinymgr")`
3. `learnr::run_tutorial(name = "modules", package = "shinymgr")`
4. `learnr::run_tutorial(name = "app_modules", package = "shinymgr")`
5. `learnr::run_tutorial(name = "tests", package = "shinymgr")`
6. `learnr::run_tutorial(name = "shinymgr", package = "shinymgr")`
7. `learnr::run_tutorial(name = "database", package = "shinymgr")`
8. `learnr::run_tutorial(name = "shinymgr_modules", package = "shinymgr")`
9. `learnr::run_tutorial(name = "apps", package = "shinymgr")`
10. `learnr::run_tutorial(name = "analyses", package = "shinymgr")`
11. `learnr::run_tutorial(name = "reports", package = "shinymgr")`
12. `learnr::run_tutorial(name = "deployment", package = "shinymgr")`

References

https://code.usgs.gov/vtcfwru/shinymgr

See Also

Other qry: `qry_app_flow()`, `qry_app_stitching()`, `qry_insert()`, `qry_mod_info()`

Examples

```
# set the file path to the main shinymgr directory
parentPath <- tempdir()
shinyMgrPath <- paste0(parentPath, '/shinymgr')

shinymgr_setup(parentPath = parentPath, demo = TRUE)
```

```
# use the default database path
qry_row(
  tableName = 'apps',
  rowConditions = data.frame(pkAppName = 'iris_explorer'),
  colConditions = c('appDisplayName', 'appDescription'),
  shinyMgrPath = shinyMgrPath
)

# remove demo
unlink(shinyMgrPath, recursive = TRUE)
```

`reports` *Sample data for the shinymgr.sqlite table, "reports"*

Description

Sample data imported to the shinymgr SQLite database by the function `shiny_db_populate`.

Usage

```
data(demo_data)
```

Format

A data frame with 3 observations on the following 3 variables.

Details

Reports are added to the shinymgr.sqlite database via the "Add "Reports" menu in shinymgr's Developer section.

Fields

`pkReportName` Name of the report; primary key.

`displayName` Display name of the report.

`reportDescription` Description of the report.

See Also

Other data: `appReports`, `appStitching`, `appTabs`, `apps`, `modFunctionArguments`, `modFunctionReturns`, `modPackages`, `modules`, `tabModules`, `tabs`

Examples

```
# read in the demo data
data(demo_data)

# look at the structure
str(reports)
```

`rerun_analysis` *Re-run an previously executed shinymgr analysis*

Description

Re-run an previously executed shinymgr analysis given an RDS file input from a previously saved analysis.

Usage

```
rerun_analysis(analysis_path)
```

Arguments

`analysis_path`	File path to the RDS file that stores a previously executed analysis.

Details

The function accepts a single argument that defines the file path to a saved shinymgr analysis (RDS file). This function will launch a shiny app, so can only be run during an interactive R session, in an R session with no other shiny apps running.

The app that is launched contains 2 tabs. The first tab is called "The App" and will be visible when the re-run function is called. It contains a header with the app's name and a subheading of "Analysis Rerun". Below that, a disclaimer appears, indicating the app was produced from a saved analysis. You may need to scroll down using the vertical scroll bar in the rendering betlow to see that below this disclaimer is a fully functioning, identical copy of the shiny app used to generate the saved analysis.

The second tab, called "Analysis Summary", simply displays the structure of the saved analysis, excluding any saved source code. The structure of the analysis gives a high-level summary, including the values that can be entered in the app to reproduce results.

Value

No return value, function launches shiny app

More Info

The rerun_analysis() function is described in the "analyses" tutorial.

Tutorials

The shinymgr learnr tutorials include, in order:

1. `learnr::run_tutorial(name = "intro", package = "shinymgr")`
2. `learnr::run_tutorial(name = "shiny", package = "shinymgr")`
3. `learnr::run_tutorial(name = "modules", package = "shinymgr")`
4. `learnr::run_tutorial(name = "app_modules", package = "shinymgr")`
5. `learnr::run_tutorial(name = "tests", package = "shinymgr")`
6. `learnr::run_tutorial(name = "shinymgr", package = "shinymgr")`
7. `learnr::run_tutorial(name = "database", package = "shinymgr")`
8. `learnr::run_tutorial(name = "shinymgr_modules", package = "shinymgr")`
9. `learnr::run_tutorial(name = "apps", package = "shinymgr")`
10. `learnr::run_tutorial(name = "analyses", package = "shinymgr")`
11. `learnr::run_tutorial(name = "reports", package = "shinymgr")`
12. `learnr::run_tutorial(name = "deployment", package = "shinymgr")`

References

https://code.usgs.gov/vtcfwru/shinymgr

See Also

Other analysis: `restore_analysis()`

Examples

```
## Only run this example in interactive R sessions
if (interactive()) {

  # ----------------------------------------------------------------
  # Load the sample analysis from the shinymgr package and re-run it.
  # ----------------------------------------------------------------

  # Get the path for the sample analysis from shinymgr
  analysis_path <- paste0(
    find.package('shinymgr'),
    '/shinymgr/analyses/iris_explorer_Gandalf_2023_06_05_16_30.RDS'
  )

  # Re-run the sample analysis
  rerun_analysis(analysis_path)

}
```

`restore_analysis` *Re-store a previously executed shinymgr analysis by regenerating an R project from an renv lockfile*

Description

Re-run an previously executed shinymgr analysis given an RDS file input from a previously saved analysis.

Usage

```
restore_analysis(analysis_path)
```

Arguments

`analysis_path`	File path to the RDS file that stores a previously executed analysis.

Details

The function accepts a single argument that defines the file path to a saved shinymgr analysis (RDS file). This function will find the lockfile and use it to create a new renv-enabled R project (a folder), that includes the full R library used by the developer when creating the app. The function creates this new project, copies the original RDS file to it, and copies a script that the user can run in an attempt to restore an old shinymgr analysis utilizing the R version and all package versions that the developer used when creating the app.

Value

No return value, restores an R environment from a saved analysis

More Info

The restore_analysis() function is described in the "analyses" tutorial.

Tutorials

The shinymgr learnr tutorials include, in order.

1. `learnr::run_tutorial(name = "intro", package = "shinymgr")`
2. `learnr::run_tutorial(name = "shiny", package = "shinymgr")`
3. `learnr::run_tutorial(name = "modules", package = "shinymgr")`
4. `learnr::run_tutorial(name = "app_modules", package = "shinymgr")`
5. `learnr::run_tutorial(name = "tests", package = "shinymgr")`
6. `learnr::run_tutorial(name = "shinymgr", package = "shinymgr")`
7. `learnr::run_tutorial(name = "database", package = "shinymgr")`
8. `learnr::run_tutorial(name = "shinymgr_modules", package = "shinymgr")`

9. `learnr::run_tutorial(name = "apps", package = "shinymgr")`
10. `learnr::run_tutorial(name = "analyses", package = "shinymgr")`
11. `learnr::run_tutorial(name = "reports", package = "shinymgr")`
12. `learnr::run_tutorial(name = "deployment", package = "shinymgr")`

References

https://code.usgs.gov/vtcfwru/shinymgr

See Also

Other analysis: `rerun_analysis()`

Examples

```
## Only run this example in interactive R sessions
if (interactive()) {

# ----------------------------------------------------------------
# Load the sample analysis from the shinymgr package and restore it.
# ----------------------------------------------------------------

  # Get the path for the sample analysis from shinymgr
  analysis_path <- paste0(
    find.package('shinymgr'),
    '/shinymgr/analyses/iris_explorer_Gandalf_2023_06_05_16_30.RDS'
  )

  # confirm file exists
  file.exists(analysis_path)

  dir_current <- getwd()

  # Re-run the sample analysis
  restore_analysis(analysis_path)

  # A new project will created in the temporary directory that
  # includes a script to run within the new renv project
  # Rerun the saved analysis from the restored environment:
  rerun_analysis('renv_iris_explorer_Gandalf_2023_06_05_16_30.RDS')

  # Reset directory and clean-up
  setwd(dir_current)
  unlink(
    file.path(tempdir(), paste0('renv_', basename(analysis_path))),
    recursive = TRUE
  )
}
```

shinymgr *A unifying framework for managing and deploying module-based Shiny applications for reproducible analyses and rapid reporting*

Description

'shinymgr' provides a unifying framework for managing and deploying Shiny applications that consist of modules. From the user's perspective, an "app" consists of a series of RShiny tabs that are presented in order, establishing an analysis workflow; results are saved as an RDS file that fully reproduces the analytic steps and can be ingested into an RMarkdown report for rapid reporting. Modules are the basic element in the 'shinymgr' framework; they can be used and re-used across different apps. New "apps" can be created with the 'shinymgr' app builder that "stitches" shiny modules together so that outputs from one module serve as inputs to the next, creating an analysis pipeline that is easy to implement and maintain. In short, developers use the 'shinymgr' framework to write modules and seamlessly combine them into shiny apps, and users of these apps can execute reproducible analyses that can be incorporated into reports for rapid dissemination.

Details

The 'shinymgr' learnr tutorials include, in order:

1. `learnr::run_tutorial(name = "intro", package = "shinymgr")`
2. `learnr::run_tutorial(name = "shiny", package = "shinymgr")`
3. `learnr::run_tutorial(name = "modules", package = "shinymgr")`
4. `learnr::run_tutorial(name = "app_modules", package = "shinymgr")`
5. `learnr::run_tutorial(name = "tests", package = "shinymgr")`
6. `learnr::run_tutorial(name = "shinymgr", package = "shinymgr")`
7. `learnr::run_tutorial(name = "database", package = "shinymgr")`
8. `learnr::run_tutorial(name = "shinymgr_modules", package = "shinymgr")`
9. `learnr::run_tutorial(name = "apps", package = "shinymgr")`
10. `learnr::run_tutorial(name = "analyses", package = "shinymgr")`
11. `learnr::run_tutorial(name = "reports", package = "shinymgr")`
12. `learnr::run_tutorial(name = "deployment", package = "shinymgr")`

Author(s)

Laurence Clarfeld, Caroline Tang, and Therese Donovan

References

https://code.usgs.gov/vtcfwru/shinymgr

See Also

Useful links:

- https://code.usgs.gov/vtcfwru/shinymgr
- Report bugs at https://code.usgs.gov/vtcfwru/shinymgr/-/issues

Examples

```
## Only run this example in interactive R sessions
if (interactive()) {
# load shinymgr
library(shinymgr)

# set the directory path that will house the shinymgr project
parentPath <- tempdir()
shinyMgrPath <- paste0(parentPath, '/shinymgr')

# set up raw directories and fresh database
shinymgr_setup(parentPath, demo = TRUE)

# look the file structure
list.files(
  path = shinyMgrPath,
  recursive = FALSE
)

# launch the demo project
launch_shinymgr(shinyMgrPath = shinyMgrPath)

# remove demo
unlink(shinyMgrPath, recursive = TRUE)

}
```

`shinymgr_setup` *Sets up a new *shinymanager* directory structure and database*

Description

Create a new *shinymgr* directory structure, database, and master app. If demo == TRUE, the database includes sample data and sample modules are also provided.

Usage

```
shinymgr_setup(parentPath, demo = FALSE)
```

Arguments

parentPath	Path to the parent directory that will house the *shinymgr* file system. A folder called "shinymgr" will be created under this parent directory. If desired, create an RStudio project associated with the "shinymgr" folder, enabling use of the renv package.
demo	TRUE or FALSE. Should the demo modules and demo database be included?

Details

shinymgr_setup is the primary function to use when starting your own *shinymgr* project. The function's has two arguments: parentPath is the path to a folder that will house the *shinymgr* project (a directory called "shinymgr"). The function will create the main directory, plus 9 sub directories ("analyses", "data", "database", "tests", "modules", "modules_app", modules_mgr","reports", "www"). Directory definitions are provided below. If demo = TRUE, these directories will be populated with sample modules and a sample database that can be used to explore the package's functionality. Once you understand the general *shinymgr* framework, you can create as many *shinymgr* projects as you wish by setting demo = FALSE.

The parentPath argument points to a directory that will house the main *shinymgr* directory, plus 9 subdirectories, along with the main *shinymgr* master app.R (or server.R and ui.R) shiny scripts.

Directories of *shinymgr* include:

analyses = stores previously run "app" results as RDS file. data - holds datasets (RData, csv) that are used by "apps". database - holds the shinymgr sqlite database, named "shinymgr.sqlite". modules - holds stand-alone modules that are combined into shinymgr "apps". modules_mgr - holds modules that are used in the shinymgr main app framework. modules_app - stores app modules; i.e., a series of modules that are linked into a tabbed workflow. reports - holds Rmd files that call in previously run analyses to produce an Rmarkdown report. tests - holds unit testing of modules to ensure everything works. www - stores all images and css files that are rendered in shiny.

Value

Returns a file structure, database, and master app called app.R

More Info

The shinymgr_setup() function is described in the "shinymgr" tutorial.

Tutorials

The shinymgr learnr tutorials include, in order:

1. `learnr::run_tutorial(name = "intro", package = "shinymgr")`
2. `learnr::run_tutorial(name = "shiny", package = "shinymgr")`
3. `learnr::run_tutorial(name = "modules", package = "shinymgr")`
4. `learnr::run_tutorial(name = "app_modules", package = "shinymgr")`
5. `learnr::run_tutorial(name = "tests", package = "shinymgr")`
6. `learnr::run_tutorial(name = "shinymgr", package = "shinymgr")`

7. learnr::run_tutorial(name = "database", package = "shinymgr")
8. learnr::run_tutorial(name = "shinymgr_modules", package = "shinymgr")
9. learnr::run_tutorial(name = "apps", package = "shinymgr")
10. learnr::run_tutorial(name = "analyses", package = "shinymgr")
11. learnr::run_tutorial(name = "reports", package = "shinymgr")
12. learnr::run_tutorial(name = "deployment", package = "shinymgr")

References

https://code.usgs.gov/vtcfwru/shinymgr

See Also

shiny_db_create

Examples

```
# ------------------------------------------------------------
# Set up an shinymgr framework in a parent directory
# ------------------------------------------------------------

# set the directory path that will house the shinymgr project
parentPath <- tempdir()
shinyMgrPath <- paste0(parentPath, '/shinymgr')

# set up raw directories and fresh database
shinymgr_setup(parentPath, demo = FALSE)

# verify that the folder structure exists in your specified directory
list.dirs(
  path = shinyMgrPath,
  full.names = FALSE,
  recursive = TRUE)

# look at the files
list.files(
  path = shinyMgrPath,
  full.names = FALSE,
  recursive = TRUE)

# Remove demo database
unlink(shinyMgrPath, recursive = TRUE)
```

`shiny_db_create` *Create an empty *shinymgr* SQLite database, and populate with demo data if desired.*

Description

Create an empty *shinymgr* SQLite database for managing multiple apps and scripts in a common framework. This function is typically not called; instead use `shinymgr_setup`

Usage

```
shiny_db_create(db_path, demo)
```

Arguments

`db_path`	Filepath that will house the sqlite database
`demo`	Logical. Should demo data be included?

Details

The *shinymgr* database is a SQLite database. The function uses the R package, RSQLite, to connect the database with R (the package itself contains SQLite, so no external software is needed. Once the connection is made, the function uses database functions from the package, DBI, which in turn can be used to query the database, add records, etc.) This function is not intended to be used. Rather, users should use `shinymgr_setup` to create the database instance that comes with the package. The function is included here so users can inspect the code used to create the database.

Value

Returns a *shinymgr* SQLite database.

Tutorials

The shinymgr learnr tutorials include, in order:

1. `learnr::run_tutorial(name = "intro", package = "shinymgr")`
2. `learnr::run_tutorial(name = "shiny", package = "shinymgr")`
3. `learnr::run_tutorial(name = "modules", package = "shinymgr")`
4. `learnr::run_tutorial(name = "app_modules", package = "shinymgr")`
5. `learnr::run_tutorial(name = "tests", package = "shinymgr")`
6. `learnr::run_tutorial(name = "shinymgr", package = "shinymgr")`
7. `learnr::run_tutorial(name = "database", package = "shinymgr")`
8. `learnr::run_tutorial(name = "shinymgr_modules", package = "shinymgr")`
9. `learnr::run_tutorial(name = "apps", package = "shinymgr")`
10. `learnr::run_tutorial(name = "analyses", package = "shinymgr")`
11. `learnr::run_tutorial(name = "reports", package = "shinymgr")`
12. `learnr::run_tutorial(name = "deployment", package = "shinymgr")`

References

https://code.usgs.gov/vtcfwru/shinymgr

See Also

Other database: shiny_db_populate()

Examples

```
# ------------------------------------------------------------
# Set up an empty database for demonstration and then delete it
# ------------------------------------------------------------

# create the database (to be deleted):
db_dir <- tempdir()
db_path <- paste0(db_dir, "/shinymgr.sqlite")
shiny_db_create(
  db_path  = db_path,
  demo = TRUE)

# verify that the database exists in your current working directory
file.exists(db_path)

# to work with a database *outside* of a *shinymgr* function,
# load the DBI package first and use RSQLite to set the driver
conx <- DBI::dbConnect(drv = RSQLite::SQLite(), dbname = db_path)

# look at the overall schema
DBI::dbReadTable(conn = conx, "sqlite_master")

# look at the tables in the database
DBI::dbListTables(conx)

# look at fields in table apps
DBI::dbListFields(conx, "apps")

# get more detailed information with a query
DBI::dbGetQuery(conx,
  statement = "PRAGMA table_info('apps');"
  )

# disconnect from database
DBI::dbDisconnect(conx)

# Delete the test database and remove it from your working directory
unlink(db_path)
```

`shiny_db_populate` *Populates an empty shinymgr.sqlite database with demo data*

Description

Populates empty shinymgr.sqlite database with demo data. The learnr tutorials illustrate the shinymgr approach and utilize the demo data. This function is typically not called; instead use `shinymgr_setup`

Usage

```
shiny_db_populate(conx)
```

Arguments

`conx`	A connection to the shinymgr.sqlite database

Details

The shinymgr database is a SQLite database. The function uses the R package, RSQLite, to connect the database with R (the package itself contains SQLite, so no external software is needed. Once the connection is made, the function uses database functions from the package, DBI, which in turn can be used to query the database, add records, etc.) This function is not intended to be used. Rather, users should use `shinymgr_setup` to copy the database instance that comes with the package. The function is included here so users can inspect the code used to create the database.

Value

Returns invisible, but the shinymgr.sqlite database will be populated.

Tutorials

The shinymgr learnr tutorials include, in order:

1. `learnr::run_tutorial(name = "intro", package = "shinymgr")`
2. `learnr::run_tutorial(name = "shiny", package = "shinymgr")`
3. `learnr::run_tutorial(name = "modules", package = "shinymgr")`
4. `learnr::run_tutorial(name = "app_modules", package = "shinymgr")`
5. `learnr::run_tutorial(name = "tests", package = "shinymgr")`
6. `learnr::run_tutorial(name = "shinymgr", package = "shinymgr")`
7. `learnr::run_tutorial(name = "database", package = "shinymgr")`
8. `learnr::run_tutorial(name = "shinymgr_modules", package = "shinymgr")`
9. `learnr::run_tutorial(name = "apps", package = "shinymgr")`
10. `learnr::run_tutorial(name = "analyses", package = "shinymgr")`
11. `learnr::run_tutorial(name = "reports", package = "shinymgr")`
12. `learnr::run_tutorial(name = "deployment", package = "shinymgr")`

References

https://code.usgs.gov/vtcfwru/shinymgr

See Also

Other database: shiny_db_create()

Examples

```
# ------------------------------------------------------------
# Set up an empty database for demonstration and then delete it
# ------------------------------------------------------------

# Create the database (to be deleted):
db_dir <- tempdir()
db_path <- paste0(db_dir, "/shinymgr.sqlite")
shiny_db_create(db_path = db_path)


# Verify that the database exists in your current working directory
file.exists(db_path)

# function will populate an empty sqlite database with the RData files
# in the package's data folder
conx <- DBI::dbConnect(drv = RSQLite::SQLite(), dbname = db_path)
shiny_db_populate(conx)

# look at some tables with R coding
DBI::dbReadTable(conx, "apps")
DBI::dbReadTable(conx, "modules")

# disconnect from database
DBI::dbDisconnect(conx)

# Remove demo database
unlink(db_path)
```

---

tabModules *Sample data for the shinymgr.sqlite table, "tabModules"*

Description

Sample data imported to the shinymgr SQLite database by the function shiny_db_populate.

Usage

```
data(demo_data)
```

Format

A data frame with 10 observations on the following 4 variables.

Details

New records to the "tabModules" table are added to the shinymgr.sqlite database via the "App Builder" interface within shinymgr's Developer section.

Fields

`pkInstanceID` Auto-number primary key.

`fkTabName` References pkTabID from the "tabs" table.

`fkModuleName` References pkModuleID from the "module" table.

`modOrder` Integer. Gives the order in which a module appears in a tab.

See Also

Other data: `appReports`, `appStitching`, `appTabs`, `apps`, `modFunctionArguments`, `modFunctionReturns`, `modPackages`, `modules`, `reports`, `tabs`

Examples

```
# read in the demo data
data(demo_data)

# look at the structure
str(tabModules)
```

---

tabs *Sample data for the shinymgr.sqlite table, "tabs"*

Description

Sample data imported to the shinymgr SQLite database by the function `shiny_db_populate`.

Usage

```
data(demo_data)
```

Format

A data frame with 10 observations on the following 4 variables.

Details

New records to the "tabs" table are added to the shinymgr.sqlite database via the "App Builder" interface within shinymgr's Developer section.

Fields

`pkTabName` The name of the tab; primary key.

`tabDisplayName` The name displayed on the tab.

`tabInstructions` Instructions for the tab.

`tabNotes` Notes on the tab.

See Also

Other data: `appReports`, `appStitching`, `appTabs`, `apps`, `modFunctionArguments`, `modFunctionReturns`, `modPackages`, `modules`, `reports`, `tabModules`

Examples

```
# read in the demo data
data(demo_data)

# look at the structure
str(tabs)
```

Index