Package 'DQAgui'

July 4, 2022

Title Graphical User Interface for Data Quality Assessment

Version 0.2.1

Description
A graphical user interface (GUI) to the functions implemented in the R package 'DQAstats'.

License GPL-3

URL https://github.com/miracum/dqa-dqagui

BugReports https://github.com/miracum/dqa-dqagui/issues

Encoding UTF-8

Date 2022-07-04

Imports data.table, daterangepicker, DIZtools (>= 0.0.5), DIZutils (>= 0.1.1), DQAstats (>= 0.3.1), DT, jsonlite, knitr, lubridate, magrittr, parsedate, shiny, shinyalert, shinydashboard, shinyFiles, shinyjs, shinyWidgets, waiter

Suggests lintr, testthat (>= 3.0.0)

RoxygenNote 7.2.0

NeedsCompilation no

Author Lorenz A. Kapsner [cre, aut] (<https://orcid.org/0000-0003-1866-860X>), Jonathan M. Mang [aut] (<https://orcid.org/0000-0003-0518-4710>), MIRACUM - Medical Informatics in Research and Care in University Medicine [fnd], Universitätsklinikum Erlangen [cph]

Maintainer Lorenz A. Kapsner <lorenz.kapsner@uk-erlangen.de>

Repository CRAN

Date/Publication 2022-07-04 11:30:02 UTC

R topics documented:

`button_send_datamap` *button_send_datamap*

Description

This function is an exported wrapper around the actual function to send the datamap. This actual function can be customized by the user.

Usage

```
button_send_datamap(rv)
```

Arguments

`rv`	The global rv object. rv$datamap needs to be valid.

Value

This functions is used to trigger logic when clicking the "Send Datamap" button on the dashboard (default: triggers the composing of an email by making use of the java-script command ‘window.open('mailto: ...')‘). When customizing ‘DQAgui‘, the function ‘button_send_datamap‘ can be overwritten in the namespace to trigger any other logic, wanted by the user.

Examples

```
if (interactive()) {
  button_send_datamap(rv=rv)
}
```

`check_load_data_button` *Evaluates whether the load-data button needs to be shown or not.*

Description

If there is a valid source system and a valid target system (this is also the case if the user set target == source), the result of this function will be TRUE and the button will be displayed via shinyjs. Otherwise the result is FALSE and the button will be hidden. This function also displays (or hides) the variables which can be assessed.

Usage

```
check_load_data_button(rv, session)
```

Arguments

`rv`	The global 'reactiveValues()' object, defined in server.R
`session`	Shiny session object

`feedback_txt` *This function is used in the config-tab and displays the selected system to the user.*

Description

This function is used in the config-tab and displays the selected system to the user.

Usage

```
feedback_txt(system, type)
```

Arguments

`system`	(String) e.g. "i2b2", "OMOP" or "CSV"
`type`	(String) "source" or "target"

Value

String containing the input params in a propper manner

get_db_settings *get_db_settings*

Description

get_db_settings

Usage

```
get_db_settings(input, target = TRUE, db_type, displayname_gui, rv)
```

Arguments

input	Shiny server input object.
target	A boolean (default: TRUE).
db_type	(String) "postgres" or "oracle".
displayname_gui	(String) "i2b2 (Prod)"
rv	The global 'reactiveValues()' object, defined in server.R

Value

This functions returns a data table of key-value pairs for the database settings, which are parsed from the respective config tab from the shiny application.

Examples

```
if (interactive()) {
 get_db_settings(
  input = input,
  target = TRUE,
  db_type = "postgres"
 )
}
```

launch_app *Launch the DQA graphical user interface (GUI)*

Description

Launch the DQA graphical user interface (GUI)

Usage

```
launch_app(
  port = 3838,
  utils_path = system.file("demo_data/utilities", package = "DQAstats"),
  mdr_filename = "mdr_example_data.csv",
  logfile_dir = tempdir(),
  parallel = FALSE,
  ncores = 2,
  demo_usage = FALSE
)
```

Arguments

port	The port, the MIRACUM DQA Tool is running on (default: 3838)
utils_path	The path to the utilities-folder, containing the metadata repository files (‘mdr.csv‘ inside the folder ‘MDR‘), JSON files with SQL statements (inside the folder ‘SQL‘), config files for the database connection (‘settings_default.yml‘) and the email address used for the data map (‘email.yml‘), a JSON file containing site names (inside the folder ‘MISC‘) and a markdown template to create the PDF report (‘DQA_report.Rmd‘ inside the folder ‘RMD‘).
mdr_filename	The filename of the mdr (e.g. "mdr_example_data.csv").
logfile_dir	Is the absolute path to the directory where the logfile will be stored. If not path is provided the tempdir() will be used.
parallel	A boolean. If ‘TRUE‘, initializing a ‘future::plan()‘ for running the code (default: ‘FALSE‘).
ncores	A integer. The number of cores to use. Caution: you would probably like to choose a low number when operating on large datasets. Default: 2.
demo_usage	A boolean. If ‘TRUE‘, a box is shown on the dashboard with further instructions on how to use / configure the tool.

Value

Executing this function returns a DQAgui shiny application.

Examples

```
if (interactive()) {
  launch_app()
}
```

module_atemp_pl_server *module_atemp_pl_server*

Description

module_atemp_pl_server

Usage

```
module_atemp_pl_server(input, output, session, rv, input_re)
```

Arguments

input	Shiny server input object
output	Shiny server output object
session	Shiny session object
rv	The global 'reactiveValues()' object, defined in server.R
input_re	The Shiny server input object, wrapped into a reactive expression: input_re = reactive(input)

Value

The function returns a shiny server module.

See Also

https://shiny.rstudio.com/articles/modules.html

Examples

```
if (interactive()) {
rv <- list()
shiny::callModule(
  module_atemp_pl_server,
  "moduleAtemporalPlausibilities",
  rv = rv,
  input_re = reactive(input)
)
}
```

module_atemp_pl_ui *module_atemp_pl_ui*

Description

module_atemp_pl_ui

Usage

```
module_atemp_pl_ui(id)
```

Arguments

id	A character. The identifier of the shiny object

Value

The function returns a shiny ui module.

See Also

https://shiny.rstudio.com/articles/modules.html

Examples

```
if (interactive()) {
shinydashboard::tabItems(
  shinydashboard::tabItem(
    tabName = "atemp_plausi",
    module_atemp_pl_ui(
      "moduleAtemporalPlausibilities"
    )
  )
)
}
```

module_completeness_server *module_completeness_server*

Description

module_completeness_server

Usage

```
module_completeness_server(input, output, session, rv, input_re)
```

Arguments

input	Shiny server input object
output	Shiny server output object
session	Shiny session object
rv	The global 'reactiveValues()' object, defined in server.R
input_re	The Shiny server input object, wrapped into a reactive expression: input_re = reactive(input)

Value

The function returns a shiny server module.

See Also

https://shiny.rstudio.com/articles/modules.html

Examples

```
if (interactive()) {
rv <- list()
shiny::callModule(
  module_completeness_server,
  "moduleCompleteness",
  rv = rv,
  input_re = reactive(input)
)
}
```

---

module_completeness_ui

module_completeness_ui

Description

module_completeness_ui

Usage

```
module_completeness_ui(id)
```

Arguments

id	A character. The identifier of the shiny object

Value

The function returns a shiny ui module.

See Also

https://shiny.rstudio.com/articles/modules.html

Examples

```
if (interactive()) {
shinydashboard::tabItems(
  shinydashboard::tabItem(
    tabName = "completeness",
    module_completeness_ui(
      "moduleCompleteness"
    )
  )
)
}
```

`module_config_server` *module_config_server*

Description

module_config_server

Usage

```
module_config_server(input, output, session, rv, input_re)
```

Arguments

`input`	Shiny server input object
`output`	Shiny server output object
`session`	Shiny session object
`rv`	The global 'reactiveValues()' object, defined in server.R
`input_re`	The Shiny server input object, wrapped into a reactive expression: input_re = reactive(input)

Value

The function returns a shiny server module.

See Also

https://shiny.rstudio.com/articles/modules.html

Examples

```
if (interactive()) {
rv <- list()
shiny::callModule(
  module_config_server,
  "moduleConfig",
  rv = rv,
  input_re = reactive(input)
)
}
```

`module_config_ui` *module_config_ui*

Description

module_config_ui

Usage

```
module_config_ui(id)
```

Arguments

`id`	A character. The identifier of the shiny object

Value

The function returns a shiny ui module.

See Also

https://shiny.rstudio.com/articles/modules.html

Examples

```
if (interactive()) {
shinydashboard::tabItems(
  shinydashboard::tabItem(
    tabName = "config",
    module_config_ui(
      "moduleConfig"
    )
  )
)
}
```

module_dashboard_server *module_dashboard_server*

Description

module_dashboard_server

Usage

```
module_dashboard_server(input, output, session, rv, input_re)
```

Arguments

input	Shiny server input object
output	Shiny server output object
session	Shiny session object
rv	The global 'reactiveValues()' object, defined in server.R
input_re	The Shiny server input object, wrapped into a reactive expression: input_re = reactive(input)

Value

The function returns a shiny server module.

See Also

https://shiny.rstudio.com/articles/modules.html

Examples

```
if (interactive()) {
rv <- list()
shiny::callModule(
  module_dashboard_server,
  "moduleDashboard",
  rv = rv,
  input_re = reactive(input)
)
}
```

module_dashboard_ui *module_dashboard_ui*

Description

module_dashboard_ui

Usage

```
module_dashboard_ui(id)
```

Arguments

id	A character. The identifier of the shiny object

Value

The function returns a shiny ui module.

See Also

https://shiny.rstudio.com/articles/modules.html

Examples

```
if (interactive()) {
shinydashboard::tabItems(
  shinydashboard::tabItem(
    tabName = "dashboard",
    module_dashboard_ui(
      "moduleDashboard"
    )
  )
)
}
```

module_descriptive_server *module_descriptive_server*

Description

module_descriptive_server

Usage

```
module_descriptive_server(input, output, session, rv, input_re)
```

Arguments

input	Shiny server input object
output	Shiny server output object
session	Shiny session object
rv	The global 'reactiveValues()' object, defined in server.R
input_re	The Shiny server input object, wrapped into a reactive expression: input_re = reactive(input)

Value

The function returns a shiny server module.

See Also

https://shiny.rstudio.com/articles/modules.html

Examples

```
if (interactive()) {
rv <- list()
shiny::callModule(
  module_descriptive_server,
  "moduleDescriptive",
  rv = rv,
  input_re = reactive(input)
)
}
```

---

module_descriptive_ui *module_descriptive_ui*

Description

module_descriptive_ui

Usage

```
module_descriptive_ui(id)
```

Arguments

id	A character. The identifier of the shiny object

Value

The function returns a shiny ui module.

See Also

https://shiny.rstudio.com/articles/modules.html

Examples

```
if (interactive()) {
shinydashboard::tabItems(
  shinydashboard::tabItem(
    tabName = "descriptive",
    module_descriptive_ui(
      "moduleDescriptive"
    )
  )
)
}
```

`module_log_server` *module_log_server*

Description

module_log_server

Usage

```
module_log_server(input, output, session, rv, input_re)
```

Arguments

`input`	Shiny server input object
`output`	Shiny server output object
`session`	Shiny session object
`rv`	The global 'reactiveValues()' object, defined in server.R
`input_re`	The Shiny server input object, wrapped into a reactive expression: input_re = reactive(input)

Value

The function returns a shiny server module.

See Also

https://shiny.rstudio.com/articles/modules.html

Examples

```
if (interactive()) {
rv <- list()
shiny::callModule(
  module_log_server,
  "moduleLogging",
  rv = rv,
  input_re = reactive(input)
)
}
```

---

module_log_ui *module_log_ui*

Description

module_log_ui

Usage

```
module_log_ui(id)
```

Arguments

id	A character. The identifier of the shiny object

Value

The function returns a shiny ui module.

See Also

https://shiny.rstudio.com/articles/modules.html

Examples

```
if (interactive()) {
shinydashboard::tabItems(
  shinydashboard::tabItem(
    tabName = "logging",
    module_log_ui(
      "moduleLogging"
    )
  )
)
}
```

module_mdr_server *module_mdr_server*

Description

module_mdr_server

Usage

```
module_mdr_server(input, output, session, rv, input_re)
```

Arguments

input	Shiny server input object
output	Shiny server output object
session	Shiny session object
rv	The global 'reactiveValues()' object, defined in server.R
input_re	The Shiny server input object, wrapped into a reactive expression: input_re = reactive(input)

Value

The function returns a shiny server module.

See Also

https://shiny.rstudio.com/articles/modules.html

Examples

```
if (interactive()) {
rv <- list()
shiny::callModule(
  module_mdr_server,
  "moduleMDR",
  rv = rv,
  input_re = reactive(input)
)
}
```

module_mdr_ui *module_mdr_ui*

Description

module_mdr_ui

Usage

```
module_mdr_ui(id)
```

Arguments

id	A character. The identifier of the shiny object

Value

The function returns a shiny ui module.

See Also

https://shiny.rstudio.com/articles/modules.html

Examples

```
if (interactive()) {
shinydashboard::tabItems(
  shinydashboard::tabItem(
    tabName = "mdr",
    module_mdr_ui(
      "moduleMDR"
    )
  )
)
}
```

module_report_server *module_report_server*

Description

module_report_server

Usage

```
module_report_server(input, output, session, rv, input_re)
```

Arguments

input	Shiny server input object
output	Shiny server output object
session	Shiny session object
rv	The global 'reactiveValues()' object, defined in server.R
input_re	The Shiny server input object, wrapped into a reactive expression: input_re = reactive(input)

Value

The function returns a shiny server module.

See Also

https://shiny.rstudio.com/articles/modules.html

Examples

```
if (interactive()) {
rv <- list()
shiny::callModule(
  module_report_server,
  "moduleReport",
  rv = rv,
  input_re = reactive(input)
)
}
```

---

module_report_ui *module_report_ui*

Description

module_report_ui

Usage

```
module_report_ui(id)
```

Arguments

id	A character. The identifier of the shiny object

Value

The function returns a shiny ui module.

See Also

https://shiny.rstudio.com/articles/modules.html

Examples

```
if (interactive()) {
shinydashboard::tabItems(
  shinydashboard::tabItem(
    tabName = "report",
    module_report_ui(
      "moduleReport"
    )
  )
)
}
```

`module_uniq_plaus_server` *module_uniq_plaus_server*

Description

module_uniq_plaus_server

Usage

```
module_uniq_plaus_server(input, output, session, rv, input_re)
```

Arguments

`input`	Shiny server input object
`output`	Shiny server output object
`session`	Shiny session object
`rv`	The global 'reactiveValues()' object, defined in server.R
`input_re`	The Shiny server input object, wrapped into a reactive expression: input_re = reactive(input)

Value

The function returns a shiny server module.

See Also

https://shiny.rstudio.com/articles/modules.html

Examples

```
if (interactive()) {
rv <- list()
shiny::callModule(
  module_uniq_plaus_server,
  "moduleUniquenessPlausibilities",
  rv = rv,
  input_re = reactive(input)
)
}
```

`module_uniq_plaus_ui` *module_uniq_plaus_ui*

Description

module_uniq_plaus_ui

Usage

```
module_uniq_plaus_ui(id)
```

Arguments

`id`	A character. The identifier of the shiny object

Value

The function returns a shiny ui module.

See Also

https://shiny.rstudio.com/articles/modules.html

Examples

```
if (interactive()) {
shinydashboard::tabItems(
  shinydashboard::tabItem(
    tabName = "uniq_plausis",
    module_uniq_plaus_ui(
      "moduleUniquenessPlausibilities"
    )
  )
)
}
```

set_target_equal_to_source *This function is called when the user clicks on the button*

Description

"Set target == source". It sets target settings = source settings.

Usage

```
set_target_equal_to_source(rv)
```

Arguments

rv	The global 'reactiveValues()' object, defined in server.R

show_failure_alert *Sjows an error alert modal with the hint to look into the logfile.*

Description

See title.

Usage

```
show_failure_alert(what_failed)
```

Arguments

what_failed	Short description of what failed.Like: "Getting the data failed." '

Value

Nothing - Just shows the alert modal.

`test_connection_button_clicked` *Checks if an connection can be established to the system.*

Description

After the button "Check connection" is pressed in the GUI, this function will be called and tries to connect to this system and feedbacks the result to the user. If the connection is successfully established, the button will be disabled and this connection will be stored as valid for the given source/target system.

Usage

```
test_connection_button_clicked(
  rv,
  source_target,
  db_type,
  input,
  output,
  session
)
```

Arguments

`rv`	The global 'reactiveValues()' object, defined in server.R
`source_target`	(String) "source" or "target"
`db_type`	(String) "postgres" or "oracle"
`input`	Shiny server input object
`output`	Shiny server output object
`session`	Shiny session object

Value

true if the connection could be established and false otherwise (also if an error occurred)

`validate_inputs` *This function checks if all necessary input parameters for source and target exist and are valid.*

Description

This function checks if all necessary input parameters for source and target exist and are valid.

Usage

```
validate_inputs(rv, input, output, session)
```

Arguments

rv	The global 'reactiveValues()' object, defined in server.R
input	Shiny server input object
output	Shiny server output object
session	Shiny session object

Index