Package ‘jqbr’

August 15, 2023

Title 'jQuery QueryBuilder' Input for 'Shiny'

Version 1.0.3

Description A highly configurable 'jQuery' plugin offering a simple interface to create complex queries/filters in 'Shiny'. The outputted rules can easily be parsed into a set of 'R' and/or 'SQL' queries and used to filter data. Custom parsing of the rules is also supported. For more information about 'jQuery QueryBuilder' see <https://querybuilder.js.org/>.

License MIT + file LICENSE


BugReports https://github.com/hfshr/jqbr/issues

Imports htmltools, jsonlite, shiny

Suggests bslib, packer, testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.2.3

NeedsCompilation no

Author Harry Fisher [aut, cre]

Maintainer Harry Fisher <harryfisher21@gmail.com>

Repository CRAN

Date/Publication 2023-08-15 06:50:02 UTC

R topics documented:

filter_table ......................................................... 2
queryBuilderInput .................................................. 3
run_jqbr_demo .................................................... 5
updateQueryBuilder ............................................... 6
useQueryBuilder .................................................. 7

Index 9
filter_table  

Apply query to a dataframe

Description

Filter a dataframe using the output of a queryBuilder. The return_value Should be set to r_rules, and the list of filters should contain column names that are present in the data as their id value.

Usage

filter_table(data = NULL, filters = NULL)

Arguments

data  data.frame to filter.
filters  output from queryBuilder when return_value = "r_rules".

Value

A filtered version of the input data.frame

Examples

library(shiny)
library(jqbr)

filters <- list(
  list(
    id = "cyl",
    type = "integer",
    input = "radio",
    values = list(
      4,
      6,
      8
    )
  )
)

ui <- fluidPage(
  queryBuilderInput(
    inputId = "r_filter",
    filters = filters,
    return_value = "r_rules"
  ),
  tableOutput("cars")
)
server <- function(input, output) {
  output$cars <- renderTable(
    filter_table(mtcars, input$r_filter)
  )
}

if (interactive()) {
  shinyApp(ui, server)
}

queryBuilderInput

---

queryBuilderInput

**Description**

Shiny input bindings for queryBuilder.

**Usage**

```r
queryBuilderInput(
  inputId,
  width = "100%",
  filters,
  plugins = NULL,
  rules = NULL,
  optgroups = NULL,
  default_filter = NULL,
  sort_filters = FALSE,
  allow_groups = TRUE,
  allow_empty = FALSE,
  display_errors = FALSE,
  conditions = c("AND", "OR"),
  default_condition = "AND",
  inputs_separator = ",",
  display_empty_filter = TRUE,
  select_placeholder = "------",
  operators = NULL,
  add_na_filter = FALSE,
  return_value = c("r_rules", "rules", "sql", "all")
)
```

**Arguments**

- **inputId**: string. Input id for the builder.
- **width**: Width of the builder. Default if "100%".
- **filters**: list of list specifying the available filters in the builder. See example for a See https://querybuilder.js.org/#filters for details on the possible options.
plugins  list of optional plugins.
rules  Initial set of rules. By default the builder will contain one empty rule
optgroups  List of groups in the filters and operators dropdows.
default_filter  string. The id of the default filter for any new rule.
sort_filters  boolean \| string. Sort filters alphabetically, or with a custom JS function.
allow_groups  boolean \| integer. Number of allowed nested groups. TRUE for no limit.
allow_empty  boolean. If TRUE, no error will be thrown if the builder is entirely empty.
display_errors  boolean. If TRUE, when an error occurs on a rule, display an icon with a tooltip explaining the error.
conditions  string. Array of available group conditions. Use the lang option to change the label.
default_condition  Default active condition. Default ’AND’.
inputs_separator  string used to separate multiple inputs (for between operator). default is ".".
display_empty_filter  boolean. Default TRUE. Add an empty option with select_placeholder string to the filter dropdowns. If the empty filter is disabled and no default_filter is defined, the first filter will be loaded when adding a rule.
select_placeholder  string. Label of the "no filter" option.
operators  NULL or list. If a list, format should follow that described here: https://querybuilder.js.org/#operators
add_na_filter  bool. Default is FALSE. If TRUE, "is_na" and "is_not_na" are added to the global filter list for testing for NA values. Only works when return_type is "rules" or "r_rules".
return_value  string. On of "r_rules", "rules", "sql_rules" or "all". Default "r_rules". Determines the return value from the builder accessed with input$<builder_id> in shiny server

Value
A htmltools::tagList() containing the queryBuilder dependencies and configuration that can be added to a shiny UI definition.

Examples
library(shiny)
library(jqbr)

ui <- fluidPage(
  useQueryBuilder(),
  queryBuilderInput(
    inputId = "qb",
    filters = list(  
      list(  
        id = "name",
        
As
run_jqbr_demo

```r
run_jqbr_demo

```
Value

A Shiny app

Examples

```r
if (interactive()) {
  run_jqbr_demo()
}
```

---

**Description**

Update a queryBuilder with available methods.

**Usage**

```r
updateQueryBuilder(
  inputId,
  setFilters = NULL,
  addFilter = NULL,
  setRules = NULL,
  destroy = FALSE,
  reset = FALSE,
  session = shiny::getDefaultReactiveDomain()
)
```

**Arguments**

- **inputId**: inputId of builder to update.
- **setFilters**: list of lists container new filters.
- **addFilter**: Named list containing filter and position elements. filter should be a list containing a list which has the new filter to add. position can be a string of either "start" or "end" or a integer specifying the position to insert the rule. If position if omitted, filter will be inserted at the end.
- **setRules**: List of rules apply to the builder.
- **destroy**: bool. TRUE to destroy filter.
- **reset**: bool. TRUE to reset filter.
- **session**: The session object passed to function given to shinyServer. Default is getDefaultReactiveDomain().

**Value**

An updated `queryBuilderInput()`
useQueryBuilder

Examples

```r
library(shiny)
library(jqbr)

# Button to reset the build and remove all rules
ui <- fluidPage(
  useQueryBuilder(),
  queryBuilderInput(
    inputId = "qb",
    filters = list(
      list(
        id = "name",
        type = "string"
      )
    ),
  ),
  actionButton("reset", "Reset")
)

server <- function(input, output) {
  observeEvent(input$reset, {
    updateQueryBuilder(
      inputId = "qb",
      reset = TRUE
    )
  })
}

if (interactive()) {
  shinyApp(ui, server)
}
```

Description

Make a call to `useQueryBuilder` in your ui code to load the required dependencies for the QueryBuilder and optionally specify the bootstrap version to use.

Usage

```r
useQueryBuilder(bs_version = c("3", "4", "5"))
```

Arguments

- `bs_version` The version of bootstrap to use with the builder. Possible values are "3", "4" or "5"
Value

list. html dependency for queryBuilderBinding. See htmltools::htmlDependency() for further information.
Index

filter_table, 2
htmltools::htmlDependency(), 8
htmltools::tagList(), 4
queryBuilderInput, 3
queryBuilderInput(), 6
run_jqbr_demo, 5
shiny, 4
updateQueryBuilder, 6
useQueryBuilder, 7