Package 'dbWebForms'

May 8, 2026

Type Package

Title Produce R Functions to Create HTML Forms Based on SQL Meta Data

Version 0.1.0

Author Timothy Conwell

Maintainer Timothy Conwell <timconwell@gmail.com>

Description Offers meta programming style tools to generate configurable R functions that produce HTML forms based on table input and SQL meta data.
Also generates functions for collecting the parameters of those HTML forms after they are submitted. Useful for
quickly generating HTML forms based on existing SQL tables. To use the resultant functions, the output files containing those functions must
be read into the R environment (perhaps using base::source()).

License GPL (>= 2)

Encoding UTF-8

LazyData true

Depends DBI, data.table, stringi

Imports html5

RoxygenNote 7.1.1

NeedsCompilation no

Repository CRAN

Date/Publication 2021-01-28 13:00:03 UTC

Contents

createFetchParamsFunction

Based on the columns in a table, produces a R function that returns a data.table of HTTP parameters extracted from a list of HTPP params. You can create such a list easily using serverUtils::paramList.

Description

Based on the columns in a table, produces a R function that returns a data.table of HTTP parameters extracted from a list of HTPP params. You can create such a list easily using serverUtils::paramList.

Usage

```
createFetchParamsFunction(
  filepath,
  function_name,
  x,
  id_col,
  type_col,
  exclude = c()
)
```

Arguments

filepath	A string, the file path including the name and file type extension where the output will be written.
function_name	A string, the name of the function to be written to the file path.
x	A data.table, should have a column of input ids (used for input names as well), and a column of HTML input types.
id_col	A string, the column of x containing the HTML input ids.
type_col	A string, the column of x containing the HTML input types.
exclude	A character vector, ids included here will not be included as parameters for the resultant function.

Value

A data.table, the HTML input type and attributes likely associated with the SQL data type.

Examples

```
createFetchParamsFunction(
filepath = paste0(tempdir(), "/", "testHTMLFormFunction.R"),
function_name = "fetch_example_function",
x = as.data.table(
list(
ids = c("user_id", "user_name", "first_name", "last_name", "state"),
```

```
types = c("number", "text", "text", "text", "text")
)
),
id_col = "ids",
type_col = "types",
exclude = c("user_id")
)
```

`createHTMLFormFunction` *Based on the columns in a table, produces a R function with parameters for each column that produces a HTML form when called.*

Description

Based on the columns in a table, produces a R function with parameters for each column that produces a HTML form when called.

Usage

```
createHTMLFormFunction(
  filepath,
  function_name,
  form_method = "POST",
  add_csrf_param = TRUE,
  form_class = NULL,
  input_class = NULL,
  submit_class = NULL,
  submit_label = "Save",
  required_span_class = NULL,
  required_span_label = NULL,
  x,
  id_col,
  type_col,
  labels = c(),
  select = c(),
  exclude = c(),
  optional = c(),
  custom_input_types = list()
)
```

Arguments

`filepath`	A string, the file path including the name and file type extension where the output will be written.
`function_name`	A string, the name of the function to be written to the file path.
`form_method`	A string, the method attribute for the HTML form tag, likely "GET" or "POST".

add_csrf_param TRUE/FALSE, if TRUE, adds a parameter for passing a value to a hidden input with a name of "crsf_token".

form_class A string, the class attribute for the HTML form tag.

input_class A string, the class attribute for the div tag wrapping the HTML inputs.

submit_class A string, the class attribute for the HTML button tag used to submit the form.

submit_label A string, the label attribute for the HTML button tag used to submit the form.

required_span_class
A string, the class attribute for the HTML span tag optionally added to required fields.

required_span_label
A string, the message for the HTML span tag optionally added to required fields.

x A data.table, should have a column of input ids (used for input names as well), and a column of HTML input types.

id_col A string, the column of x containing the HTML input ids.

type_col A string, the column of x containing the HTML input types (usually created by calling dbTypeToHTMLInputType()) .

labels A named character vector, names are the ids of the inputs and values are the labels to use. If a column is not specified here, the default label is the result of namesToLabels() called for each input id.

select A character vector, ids included here will become select tags rather than input tags and a function parameter will be added to pass options.

exclude A character vector, ids included here will not be included as parameters for the resultant function.

optional A character vector, ids included here will not be marked as required inputs in the relevant HTML tags.

custom_input_types
A named list of character vectors, names are the column ids, character vectors must have an entry named type with a value for the HTML input type to be used and additional attributes can be included as subsequent named values in the character vector.

Value

A character vector, the HTML input type and attributes likely associated with the SQL data type.

Examples

```
createHTMLFormFunction(
filepath = paste0(tempdir(), "/", "testHTMLFormFunction.R"),
function_name = "example_function",
form_method = "POST",
add_csrf_param = TRUE,
form_class ="pure-form pure-form-aligned",
input_class = "pure-control-group",
submit_class = "pure-button pure-button-primary",
submit_label = "Save",
```

```
required_span_class = "pure-form-message-inline",
required_span_label = "Required field",
x = as.data.table(
list(
ids = c("user_id", "user_name", "first_name", "last_name", "state"),
types = c("number", "text", "text", "text", "text")
)
),
id_col = "ids",
type_col = "types",
labels = c(user_name = "Account Name"),
select = c("state"),
exclude = c("user_id"),
optional = c("first_name", "last_name"),
custom_input_types = list(user_name = c(type = "email", minlength = 5))
)
```

`dbTableInfo` *Query INFORMATION_SCHEMA or equivalent SQL meta data to obtain column names and types for a table.*

Description

Query INFORMATION_SCHEMA or equivalent SQL meta data to obtain column names and types for a table.

Usage

```
dbTableInfo(
  con = NULL,
  sql = c("MariaDB", "Microsoft SQL Server", "MySQL", "PostgreSQL", "SQLite"),
  table_catalog = NULL,
  table_schema = NULL,
  table_name
)
```

Arguments

`con`	A database connection that can be passed to DBI::dbSendQuery/DBI::dbGetQuery.
`sql`	A string, the type of SQL database con is connected to; must be one of c("MariaDB", "Microsoft SQL Server", "MySQL", "PostgreSQL", "SQLite").
`table_catalog`	A string, the catalog (usually a database name) name of the SQL table to return column meta data for. Not used if sql = "SQLite".
`table_schema`	A string, the schema name of the SQL table to return column meta data for.
`table_name`	A string, the name of the SQL table to return column meta data for.

Value

A data.table, two columns, "column_name" has the names of the columns in the specified SQL table and "data_type" has the data types for each column. If con is NULL, returns the SQL string for querying the meta data but does not execute the statement.

Examples

```
dbTableInfo(
con = NULL,
sql = "PostgreSQL",
table_catalog = "db1",
table_schema = "public",
table_name = "table1"
)
```

---

dbTypeToHTMLInputType *Convert SQL data types to likely HTML input types*

Description

Convert SQL data types to likely HTML input types

Usage

```
dbTypeToHTMLInputType(db_type)
```

Arguments

db_type	A string, the SQL data type to convert to HTML input type.

Value

A character vector, the HTML input type and attributes likely associated with the SQL data type.

Examples

```
dbTypeToHTMLInputType("int")
```

namesToLabels — *Convert strings to title case, splitting strings into separate words based on a separator.*

Description

Convert strings to title case, splitting strings into separate words based on a separator.

Usage

```
namesToLabels(x, split = "_")
```

Arguments

x	A string.
split	A string, used to split x into constituent words to be converted to title case.

Value

A string, converted to title case with split words separated with a space character.

Examples

```
namesToLabels("date_of_birth", split = "_")
```

quoteText — *Add single quotes to strings, useful for converting R strings into SQL formatted strings.*

Description

Add single quotes to strings, useful for converting R strings into SQL formatted strings.

Usage

```
quoteText(x, char_only = TRUE)
```

Arguments

x	A string.
char_only	TRUE/FALSE, if TRUE, adds quotes only if is.character(x) is TRUE.

Value

A string, with single quotes added to match postgreSQL string formatting.

Examples

```
quoteText("Sample quotes.")
```

Index