Package: plumbertableau (via r-universe)

June 17, 2024

Type Package

Title Turn 'Plumber' APIs into 'Tableau' Extensions

Version 0.1.1

Description Build 'Plumber' APIs that can be used in 'Tableau' workbooks. Annotations in R comments allow APIs to conform to the 'Tableau Analytics Extension' specification, so that R code can be used to power 'Tableau' workbooks.

License MIT + file LICENSE

URL https://rstudio.github.io/plumbertableau/, https://github.com/rstudio/plumbertableau

BugReports https://github.com/rstudio/plumbertableau/issues

Encoding UTF-8

Depends R (>= 3.0.0)

Imports plumber (>= 1.1.0), magrittr, curl, httpuv, jsonlite, later, promises, rlang, htmltools, debugme, stringi, markdown, urltools, utils, httr, knitr

RoxygenNote 7.1.1

Suggests testthat (>= 3.0.0), rmarkdown, covr

Config/testthat/edition 3

VignetteBuilder knitr

Roxygen list(markdown = TRUE)

Repository https://rstudio.r-universe.dev

RemoteUrl https://github.com/rstudio/plumbertableau

RemoteRef HEAD

RemoteSha 00b19a77ef17273ba0e237ee9d7f2fbd911e22f9

Contents

`arg_spec` *Describe expected args and return values*

Description

`arg_spec()` and `return_spec()` are used to create arguments for `tableau_handler()`. They describe the data type of the arg or return value, and can return a human-readable description that can be used to generate documentation.

Usage

```
arg_spec(
  type = c("character", "integer", "logical", "numeric"),
  desc = "",
  optional = grepl("\\?$", type)
)

return_spec(type = c("character", "integer", "logical", "numeric"), desc = "")
```

Arguments

`type`	A string indicating the data type that is required for this argument.
`desc`	A human-readable description of the argument. Used to generate documentation.
`optional`	If `TRUE`, then this argument need not be present in a request. Defaults to `TRUE` if `type` ends with a `"?"` character.

Value

A `tableau_arg_spec` object, which is a list containing details about the Tableau argument expectations

A `tableau_return_spec` object, which is a list containing details about the values expected to be returned to Tableau

mock_tableau_request *Create a mock JSON request that mimics the request structure of Tableau*

Description

`mock_tableau_request()` creates a JSON object formatted like a request from Tableau. The JSON object it returns can be pasted directly into the "Try it out" field in the Swagger documentation for an endpoint to test its functionality.

Usage

```
mock_tableau_request(script, data, ...)
```

Arguments

`script`	String indicating the path to the endpoint to be called
`data`	A list or dataframe that is serialized to JSON
`...`	Additional arguments passed to `jsonlite::toJSON()`

Details

Behind the scenes, Tableau sends all requests to the `/evaluate` endpoint. Each request is a JSON object containing two items: `script` and `data`. plumbertableau uses `script` to specify an individual endpoint to call, and passes the arguments in `data` on to the function at that endpoint.

Value

A JSON object that can be passed to a Tableau endpoint

Examples

```
mock_tableau_request("/loess/predict", mtcars[,c("hp", "mpg")])
```

tableau_extension *Modify a Plumber router to function as a Tableau Analytics Extension*

Description

Most of the time, you won't call this function directly. Instead, you'll place it at the end of a Plumber router, under a `#* @plumber` annotation, with no trailing parentheses or arguments. This tells Plumber to use the function to modify the router object.

Usage

```
tableau_extension

tableau_extension(pr)
```

Arguments

pr	A plumber router

Value

A modified plumber router that functions as a Tableau Analytics Extension

Examples

```
## Not run:
library(plumber)
library(plumbertableau)

#* Capitalize incoming text
#* @post /capitalize
function(req, res) {
  dat <- req$body$data
  toupper(dat)
}

#* @plumber
tableau_extension

## End(Not run)
```

---

tableau_handler *Create a Tableau-compliant handler for a function*

Description

Creates an object that can translate arguments from Tableau to R, and return values from R to Tableau.

Usage

```
tableau_handler(args, return, func)
```

Arguments

args	A named list describing the arguments that are expected from valid Tableau requests. The names in the named list can be any unique variable names. The values in the named list must each be either a string indicating the expected data type for that argument ("character", "logical", "numeric", or "integer"); or better yet, a specification object created by arg_spec(). If an argument should be considered optional, then its data type should be followed by ?, like "numeric?".
return	A string indicating the data type that will be returned from func ("character", "logical", "numeric", or "integer"); or, a specification object created by return_spec().
func	A function to be used as the handler function. Code in the body of the function will automatically be able to access Tableau request args simply by referring to their names in args; see the example below.

Value

A tableau_handler object that is a validated version of the provided func with additional attributes describing the expected arguments and return values

---

tableau_invoke *Programatically invoke a Tableau extension function*

Description

Simulates invoking a Tableau extension function from a Tableau calculated field SCRIPT_* call. Intended for unit testing of plumbertableau extensions.

Usage

```
tableau_invoke(pr, script, ..., .toJSON_args = NULL, .quiet = FALSE)
```

Arguments

pr	Either a tableau_extension style Plumber router object, or, the filename of a plumber.R that implements a Tableau extension.
script	The script string that identifies the plumber route to invoke. (Equivalent to the first argument to SCRIPT_STR, et al., in Tableau.) URL query parameters are allowed.
...	Zero or more unnamed arguments to be passed to the script.
.toJSON_args	Additional options that should be passed to jsonlite::toJSON() when the ... arguments are serialized; for example, pretty = TRUE or digits = 8.
.quiet	If TRUE, do not print response bodies when errors occur.

Value

The object that was returned from the request, JSON-decoded using `jsonlite::parse_json`.

Examples

```
pr_path <- system.file("plumber/stringutils/plumber.R",
  package = "plumbertableau")

tableau_invoke(pr_path, "/lowercase", LETTERS[1:5])
```

Index