

Package: scaffolder (via r-universe)

August 30, 2024

Type Package

Title Scaffolding Interfaces to Packages in Other Programming Languages

Version 0.0.1

Description Comprehensive set of tools for scaffolding R interfaces to modules, classes, functions, and documentations written in other programming languages, such as 'Python'.

License Apache License 2.0

URL <https://github.com/terrytangyuan/scaffolder>

BugReports <https://github.com/terrytangyuan/scaffolder/issues>

SystemRequirements Python (>= 2.7.0)

Encoding UTF-8

LazyData true

Depends R (>= 3.0)

Imports reticulate (>= 1.17), utils

Suggests knitr, rmarkdown, testthat, stringr, tensorflow

RoxygenNote 7.0.2

Roxygen list(markdown = TRUE)

VignetteBuilder knitr

Repository <https://terrytangyuan.r-universe.dev>

RemoteUrl <https://github.com/terrytangyuan/scaffolder>

RemoteRef HEAD

RemoteSha 58b2434f2bbbcf69407b717554612b482875e7f8

Contents

custom_scaffold_py_function_wrapper	2
scaffolder	3
scaffold_py_function_wrapper	3

Index	5
--------------	----------

 custom_scaffold_py_function_wrapper

Custom Scaffolding of R Wrappers for Python Functions

Description

This function can be used to generate R wrapper for a specified Python function while allowing to inject custom code for critical parts of the wrapper generation, such as process the any part of the obtained docs and append additional roxygen fields. The result from execution of `python_function` is assigned to a variable called `python_function_result` that can also be processed by `postprocess_fn` before writing the closing curly braces for the generated wrapper function.

Usage

```
custom_scaffold_py_function_wrapper(
  python_function,
  r_function = NULL,
  additional_roxygen_fields = NULL,
  process_docs_fn = function(docs) docs,
  process_param_fn = function(param, docs) param,
  process_param_doc_fn = function(param_doc, docs) param_doc,
  postprocess_fn = NULL,
  file_name = NULL
)
```

Arguments

<code>python_function</code>	Fully qualified name of Python function or class constructor (e.g. <code>tf.nn.top_k</code>)
<code>r_function</code>	Name of R function to generate (defaults to name of Python function if not specified)
<code>additional_roxygen_fields</code>	A list of additional roxygen fields to write to the roxygen docs, e.g. <code>list(export = "", rdname = "generated-wrappers")</code> .
<code>process_docs_fn</code>	A function to process the obtained docs.
<code>process_param_fn</code>	A function to process each parameter needed for <code>python_function</code> before executing <code>python_function</code> .
<code>process_param_doc_fn</code>	A function to process the roxygen docstring for each parameter.
<code>postprocess_fn</code>	A function to inject any custom code in the form of a string before writing the closing curly braces for the generated wrapper function.
<code>file_name</code>	The file name to write the generated wrapper function to. If <code>NULL</code> , the generated wrapper will only be printed out in the console.

Examples

```

library(tensorflow)
library(stringr)

# Example of a `process_param_fn` to cast parameters with default values
# that contains "L" to integers
process_int_param_fn <- function(param, docs) {
  # Extract the list of parameters that have integer values as default
  int_params <- gsub(
    " = [-]?[0-9]+L",
    "",
    str_extract_all(docs$signature, "[A-z]+ = [-]?[0-9]+L")[[1]])
  # Explicitly cast parameter in the list obtained above to integer
  if (param %in% int_params) {
    param <- paste0("as.integer(", param, ")")
  }
  param
}

# Note that since the default value of parameter `k` is `1L`. It is wrapped
# by `as.integer()` to ensure it's casted to integer before sending it to `tf$nn$top_k`
# for execution. We then print out the python function result.
custom_scaffold_py_function_wrapper(
  "tf$nn$top_k",
  r_function = "top_k",
  process_param_fn = process_int_param_fn,
  postprocess_fn = function() { "print(python_function_result)" })

```

scaffolder

Scaffolding Interfaces to Packages in Other Programming Languages

Description

This package provides a comprehensive set of tools to scaffold interfaces to modules, classes, functions, and documentations written in other programming languages.

scaffold_py_function_wrapper

Scaffold R wrappers for Python functions

Description

Scaffold R wrappers for Python functions

Usage

```
scaffold_py_function_wrapper(  
  python_function,  
  r_function = NULL,  
  file_name = NULL  
)
```

Arguments

<code>python_function</code>	Fully qualified name of Python function or class constructor (e.g. <code>tf.nn.top_k</code>)
<code>r_function</code>	Name of R function to generate (defaults to name of Python function if not specified)
<code>file_name</code>	The file name to write the generated wrapper function to. If NULL, the generated wrapper will only be printed out in the console.

Note

The generated wrapper will often require additional editing (e.g. to convert Python list literals in the docs to R lists, to massage R numeric values to Python integers via `as.integer` where required, etc.) so is really intended as an starting point for an R wrapper rather than a wrapper that can be used without modification.

Examples

```
library(scaffolder)  
library(tensorflow)  
  
scaffold_py_function_wrapper("tf.nn.top_k")
```

Index

`custom_scaffold_py_function_wrapper`, [2](#)

`scaffold_py_function_wrapper`, [3](#)

`scaffolder`, [3](#)