Package ‘dstack’

Type Package
Title Publishing Interactive Plots
Version 0.1.0
Description A native R package that allows to publish, share and track revisions of plots using your favorite plotting package, e.g. ‘ggplot2’. It also provides a kind of interactivity for such plots by specifying certain parameters for any specific plot view. See <https://docs.dstack.ai> for more information.
License Apache License (>= 2.0)
URL https://dstack.ai
BugReports https://github.com/dstackai/dstack-r/issues
Encoding UTF-8
LazyData true
RoxygenNote 7.0.2
Imports uuid, nanotime, rjson, httr, rlist, yaml, base64enc
Suggests ggplot2
NeedsCompilation no
Author Vitaly Khudobakhshov [aut, cre]
Maintainer Vitaly Khudobakhshov <vitaly@dstack.ai>
Repository CRAN
Date/Publication 2020-03-24 16:30:08 UTC

R topics documented:

commit .................................................. 2
create_frame ......................................... 3
ggplot_handler ....................................... 4
json_protocol ......................................... 5
push .................................................... 5
push_frame ........................................... 6
yaml_config .......................................... 7

Index 8
commit  

Commit Data to Stack Frame

Description

Function adds a new view to the stack frame. Multiple views can be added to one frame, but in this case every plot must be supplied with certain parameters to distinguish one view from another. In the case of single plot parameters are not necessary. For multiple views parameters will be automatically converted to UI controls like sliders and drop down lists.

Usage

commit(frame, obj, description = NULL, params = NULL)

Arguments

frame  
A frame you want to commit.

obj  
A data to commit. Data will be preprocessed by the handler but dependently on auto_push mode will be sent to server or not. If auto_push is False then the data won’t be sent. Explicit push call need anyway to process committed data. auto_push is useful only in the case of multiple data objects in the stack frame, e.g. set of plots with settings.

description  
Description of the data.

params  
Parameters associated with this data, e.g. plot settings.

Value

Changed frame.

Examples

library(ggplot2)
library(dstack)
image <- qplot(clarity, data = diamonds, fill = cut, geom = "bar")
frame <- create_frame(stack = "diamonds")
frame <- commit(frame, image, "Diamonds bar chart")
print(push(frame))  # print actual stack URL
create_frame

Description

Frame is kind of revision of data user is going to publish. It consists of one or more views. Views are usually plots with some parameters to distinguish one plot from another. This function creates a frame and by default it checks permission to publish to this stack.

Usage

create_frame(
    stack,
    profile = "default",
    handler = ggplot_handler(),
    auto_push = FALSE,
    protocol = NULL,
    config = yaml_config(),
    encryption = NULL,
    check_access = TRUE
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stack</td>
<td>A name of stack to use.</td>
</tr>
<tr>
<td>profile</td>
<td>A profile refers to credentials, i.e. username and token. Default profile is named 'default'.</td>
</tr>
<tr>
<td>handler</td>
<td>A handler which can be specified in the case of custom content, but by default it is json_handler. The system is looking for specified profile as follows: it looks into working directory to find a configuration file (local configuration), if the file doesn’t exist it looks into user directory to find it (global configuration). The best way to manage profiles is to have dstack CLI tools installed. These tools are written in Python 3, so you have to install dstack support. In the case of PyPI you should type $ pip install dstack or $ conda install -c dstack.ai dstack. We recommend to use local (virtual) environment to install the package. You can use this command in console: $ dstack config --list to list existing profiles or add or replace token by following command $ dstack config --profile &lt;PROFILE&gt; or simply $ dstack config</td>
</tr>
</tbody>
</table>
if profile is not specified 'default' profile will be created. The system asks you about token from command line, make sure that you have already obtained token from the site.

- **auto_push**
  - Tells the system to push frame just after commit. It may be useful if you want to see result immediately. Default is FALSE.

- **protocol**
  - Protocol to use, usually it is NULL it means that json_protocol will be used.

- **config**
  - A configuration, by default it will be obtained from YAML configuration files, so yaml_config will be used.

- **encryption**
  - This is an encryption method. By default is NULL and no encryption will be used.

- **check_access**
  - Check access to specified stack, default is TRUE.

**Value**

New frame.

**Examples**

```r
library(ggplot2)
library(dstack)
image <- qplot(clarity, data = diamonds, fill = cut, geom = "bar")
frame <- create_frame(stack = "diamonds")
frame <- commit(frame, image, "Diamonds bar chart")
print(push(frame)) # print actual stack URL
```

---

### ggplot_handler

**Handles 'ggplot2' Objects**

**Description**

Returns a function that converts 'ggplot2' plots to appropriate format and format itself. PNG and SVG are supported.

**Usage**

```r
ggplot_handler(dpi = 300, width = NA, height = NA, format = "png")
```

**Arguments**

- **dpi**
  - DPI, default is 300.

- **width**
  - Image width.

- **height**
  - Image height.

- **format**
  - Image format to use, can be "png" or "svg", by default PNG will be used.

**Value**

A list which contains conversion function and format itself.
json_protocol

JSON Protocol Implementation to Connect API Server

Description
Protocol is an abstraction which allows to send data to server. This function implements JSON-based protocol. It provides token in 'Authorization' header.

Usage
json_protocol(server, error = .error)

Arguments
server A server to connect.
error An error handling function.

Value
A function that implements JSON protocol.

push

Push All Commits to Server

Description
This function is used to send a branch of committed plots to server or say server that operation with this frame is done. In the case of 'auto_push' mode it sends only a total number of views in the frame. So call this method is obligatory to close the frame anyway.

Usage
push(frame)

Arguments
frame A frame to push.

Value
Stack URL.
Examples

```r
library(ggplot2)
library(dstack)
image <- qplot(clarity, data = diamonds, fill = cut, geom = "bar")
frame <- create_frame(stack = "diamonds")
frame <- commit(frame, image, "Diamonds bar chart")
print(push(frame)) # print actual stack URL
```

push_frame

**push_frame**

*Creates a Frame, Commits and Pushes Data in a Single Operation*

Description

In the case of one plot per push you can use do all operations in one call. This function creates a frame, commits view and pushes all data to server. The main difference in behaviour in this case is the function creates frame without permission check, so be sure that you have certain permission to push in the stack.

Usage

```r
push_frame(
  stack,
  obj,
  description = NULL,
  params = NULL,
  profile = "default",
  handler = ggplot_handler(),
  protocol = NULL,
  config = yaml_config(),
  encryption = .no_encryption
)
```

Arguments

- **stack**
  A name of stack to use.
- **obj**
  Object to commit and push, e.g. plot.
- **description**
  Optional description of the object.
- **params**
  Optional parameters.
- **profile**
  Profile you want to use, i.e. username and token. Default profile is 'default'.
- **handler**
  Specify handler to handle the object, if it's None then ggplot_handler will be used.
- **protocol**
  Protocol to use, usually it is NULL it means that json_protocol will be used.
- **config**
  Configuration to manage profiles. If it is unspecified yaml_config will be used.
- **encryption**
  Encryption method by default no_encryption will be used.


**yaml_config**

**Value**

Stack URL.

**Examples**

```r
library(ggplot2)
library(dstack)
image <- qplot(clarity, data = diamonds, fill = cut, geom = "bar")
push_frame("diamonds", image, "Diamonds bar chart")
```

**yaml_config**

**YAML-based Configuration**

**Description**

It tries to find YAML file in working directory looking for `.dstack/config.yaml` by default. If it’s failed it tries to use global setting in home directory in the same relative path.

**Usage**

```r
yaml_config(dstack_dir = ".dstack")
```

**Arguments**

- `dstack_dir` Directory which contains config.yaml, locally or globally, by default it is `.dstack`.

**Value**

A function that returns a list that contains user, token and server for specified profile.
Index

commit, 2
create_frame, 3

ggplot_handler, 4
json_protocol, 5
push, 5
push_frame, 6
yaml_config, 7