nukedatastore Documentation

Release 0.2.0

Florian Einfalt

Jun 28, 2018

Contents

Contents:

CHAPTER 1

Installation

To install `nukedatastore`, type:

```
$ pip install nukedatastore
```

Open Nuke's `init.py` file and add:

```
nuke.pluginAddPath('/path/to/your/local/python/site-packages')
```

CHAPTER 2

Getting Started

To get started with `nukedatastore`, type in the Nuke Script Editor:

```
import nukedatastore
```

2.1 NukeDataStore

To initialise a `NukeDataStore`, type:

```
ds = nukedatastore.NukeDataStore('data_store')
```

Note: `nukedatastore` will try to find an existing data store with the same name first, if that does not succeed, a new data store is created.

To store data in the `NukeDataStore`, type:

```
ds['project_data'] = {'id': 1234, 'name': 'project name'}
```

Note: If the key (i.e. `project_data`) does not exists, then `nukedatastore` will automatically create it.

Warning: All data stored in `NukeDataStore` must be JSON serialisable.

To list all available keys in the `NukeDataStore`, type:

```
ds.list()
# ['project_data']
```

To retrieve stored data from the `NukeDataStore`, type:

```
ds['project_data']
# {'id': 1234, 'name': 'project name'}
```

A `NukeDataStore` can be frozen, to freeze, type:

```
ds.freeze()
```

Any further attempt to set data on the `NukeDataStore` will result in an error:

```
ds['color_data'] = {'id': 'AB-123', 'name': 'White'}
# nukedatastore.NukeDataStoreError: Cannot mutate frozen NukeDataStore
```

To un-freeze, type:

```
ds.unfreeze()
```

2.2 NukeAPICache

Working with the `NukeAPICache` is very similar. To register an API, type:

```
api_cache = nukedatastore.NukeAPICache('api_cache')
api_cache.register('project_data', 'https://project.your.domain.com/api')
```

To read the cached API data, type:

```
api_cache['project_data']
```

To update the API data, type:

```
api_cache.update('project_data')
```

Note: `NukeAPICache` supports freezing and unfreezing just like `NukeDataStore`.

To diff existing API data with new API data, type:

```
api_cache.diff('project_data')
# {'project_data': {'values_changed': {"root['headers']['X-Request-Id']": {'new_value
↪': u'f5800c5e-4edb-4509-8339-4bcdf0b32732', 'old_value': u'd8ed6737-e5c8-49aa-b42e-
↪58eb2ba472b9'}}}}
```

CHAPTER 3

API Documentation

class nukedatastore.**NukeDataStoreError**(*message*)
Exception indicating an error related to the Nuke Data Store, inherits from ValueError.

class nukedatastore.**NukeDataStore**(*name*)
NukeDataStore class, wrapper around Nuke's NoOp node.

Parameters **name** (*str*) – Data store name

Usage:

```
>>> from nukedatastore import NukeDataStore
>>> ds = NukeDataStore('data_store')
>>> ds['project_data'] = {'id': 1234, 'name': 'project name'}
>>> print ds['project_data']
>>> {'id': 1234, 'name': 'project name'}
```

freeze()
Freeze the data in the *NukeDataStore* and make it unchangeable.

is_frozen()
Return whether the data in the *NukeDataStore* is frozen and therefore unchangeable.

list()
List all available keys in the *NukeDataStore*.

store
Return the data store's Nuke node

Returns Data store node

Return type Node

unfreeze()
Unfreeze the data in the *NukeDataStore* and make it changeable.

class nukedatastore.**NukeAPICache**(*name*)
NukeAPICache class, inherits from *NukeDataStore*.

Parameters **name** (*str*) – Data store name

Usage:

```
>>> from nukedatastore import NukeAPICache
>>> api_cache = NukeAPICache('api_cache')
>>> api_cache.register('project_data', 'https://project.your.domain.com/api')
>>> print api_cache['project_data']
>>> {'id': 1234, 'name': 'project name'}
```

diff(**args*)

Given *args, diff specified APIs, if no APIs are specified, diff all registered APIs.

Parameters ***args** (*str*) – API names

Returns Diff

Return type dict

register(*name*, *url*, *update=True*, *ignore_exists=True*)

Given a `name` and a `url`, register a new API in the cache.

Parameters

- **name** (*str*) – API name
- **url** (*str*) – API URL
- **update** (*bool*) – Update API data after registering, default: `True`
- **ignore_exists** (*bool*) – Ignore API is already registered, default: `True`

timestamp(**args*)

Given *args, return timestamps for specified APIs, if no APIs are specified, return timestamps for all registered APIs.

Parameters ***args** (*str*) – API names

Returns List of timestamp tuples (api_name, timestamp)

Return type list

update(**args*)

Given *args, update specified APIs, if no APIs are specified, update all registered APIs.

Parameters ***args** (*str*) – API names

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Index