Watson - Cache

Release 1.0.1

April 08, 2015

Contents

A collection of cache storage mechanisms that act like a dict.

Currently supporting:

- Memory
- File
- Memcache

Also contains a decorator that can be used within the `watson-framework` package.

CHAPTER 1

Build Status

CHAPTER 2

Dependencies

- watson-common
- watson-di (for test coverage, and decorator usage)
- python3-memcached

CHAPTER 3

Installation

```
pip install watson-cache
```

CHAPTER 4

Testing

Watson can be tested with py.test. Simply activate your virtualenv and run `python setup.py test`.

CHAPTER 5

Contributing

If you would like to contribute to Watson, please feel free to issue a pull request via Github with the associated tests for your code. Your name will be added to the AUTHORS file under contributors.

CHAPTER 6

Table of Contents

6.1 Usage

In order to use the Memcache backend, you must install `python3-memcached` first. This is available via `pip install python3-memcached`.

```
cache = StorageType()
cache['key'] = 'value'
cache.set('key', 'value', timeout=360)
cache['key']  # value
cache.get('missing_key', default='value')  # value
```

6.2 Reference Library

6.2.1 watson.cache.decorators

watson.cache.decorators.**cache**(*func=None*, *timeout=0*, *key=None*, *cache_type=<class 'watson.cache.storage.Memory'>*)

Retrieve a value from the cache

Attempts to retrieve a value from the cache. If the wrapped function does not have an attribute of container (see watson.di.container), from which to retrieve the cache type then it will default to cache.storage.Memory.

Parameters

- **func** (*callable*) – the function that is being wrapped
- **timeout** (*int*) – the number of seconds the item should exist in the cache
- **key** (*string*) – the key to store the data against in the cache, defaults to the qualified name of the decorated function.

Returns The contents of the cache key.

Example:

```
class MyClass(ContainerAware):
    @cache(timeout=3600)
    def expensive_func(self):
        return 'something'

c = MyClass()
```

```
c.expensive_func() # something
c.expensive_func() # something - returned from cache
```

6.2.2 watson.cache.storage

class watson.cache.storage.**BaseStorage**(*config=None*)
Base class for all cache storage classes.

Cache storage classes are designed to act similar to a dict, however get and set methods can be used when a timeout is required on a set, or when a default value is to be specified on a get.

config
dict

The relevant configuration settings for the storage.

__init__(*config=None*)

expired(*key*)
Determine if a key has expired or not.

Parameters **key** (*string*) – The key to find

Returns True/False depending on expiration

Return type Boolean

flush()
Clears all items from the cache.

get(*key*, *default=None*)
Gets a key from the cache, returns the default if not set.

Parameters **key** (*string*) – The key to be retrieved

Returns The value stored within the cache

Example:

```
value = cache['key']
```

set(*key*, *value*, *timeout=0*)
Sets a key in the cache.

Parameters

- **key** (*string*) – The key to be used as a reference
- **value** (*mixed*) – The value to store in the key
- **timeout** (*int*) – The amount of time in seconds a key is valid for.

Example:

```
cache['key'] = 'value'
```

class watson.cache.storage.**File**(*config=None*)
A cache storage mechanism for storing items on the local filesystem.

File cache storage will persist the data to the filesystem in whichever directory has been specified in the configuration options. If no directory is specified then the system temporary folder will be used.

__init__(*config=None*)
Initializes the cache.

Parameters **config** (*dict*) – The config for the cache

Example:

```
cache = File({'dir': '/tmp', 'prefix': 'my-cache'})
# all cached items will be saved to /tmp
# and will be prefixed with my-cache
cache['key'] = 'value' # /tmp/my-cache-key contains a serialized 'value'
```

class watson.cache.storage.**Memcached**(*config=None*)

A cache storage mechanism for storing items in memcached.

Memcached cache storage will utilize python3-memcached to maintain the cache across multiple servers. Python3-memcached documentation can be found at http://pypi.python.org/pypi/python3-memcached/

__init__(*config=None*)

Initializes the cache.

Parameters **config** (*dict*) – The config for the cache

Example:

```
cache = Memcached({'servers': ['127.0.0.1:11211', '192.168.100.1:11211']})
```

class watson.cache.storage.**Memory**

A cache storage mechanism for storing items in memory.

Memory cache storage will maintain the cache while the application is being run. This is usually best used in instances when you don't want to keep the cached items after the application has finished running.

__init__()

W

Symbols

B

C

E

F

G

M

S

W