link.middleware Documentation

Release 1.5

David Delassus

September 02, 2016

Contents

1 Installation **3**

2 Contents **5**

2.1 Tutorial 5

2.2 API documentation 7

3 Features **13**

4 Donating **15**

Python Module Index **17**

link.middleware provides some utilities to create middlewares that connect to a remote service.

Check out the source code on Github.

CHAPTER 1

Installation

```
pip install link.middleware
```

CHAPTER 2

Contents

2.1 Tutorial

2.1.1 Introduction: Middlewares and URLs

A *middleware* is a class which can be instantiated with an **URL**.

The middleware is used to provide a Python API for different protocols, those protocol are used as scheme of the middleware URL to identify which one to instantiate.

```
http://user:pass@localhost:81/path/sub#fragment?foo=bar
```

The above URL will instantiate the class `HTTPMiddleware`:

```
HTTPMiddleware(
    username='user',
    password='pass',
    hosts=[('localhost', 81)],
    path=['path', 'sub'],
    fragment='fragment',
    foo='bar'
)
```

NB: As you can see, `hosts` is an array, middleware URLs can have multiple hosts:

```
http://user:pass@host1:80,host2:80,host3/path/sub#fragment?foo=bar
```

With the following instantiation:

```
HTTPMiddleware(
    username='user',
    password='pass',
    hosts=[
        ('host1', 80),
        ('host2', 80),
        ('host3', None)
    ],
    path=['path', 'sub'],
    fragment='fragment',
    foo='bar'
)
```

2.1.2 Creating a middleware

In order to create a middleware for a protocol (database, messaging, ...), you just need to create a new class inheriting from the `Middleware` class, and register it:

```
from link.middleware.core import Middleware, register_middleware


@register_middleware
class MyMiddleware(Middleware):
    __protocols__ = ['myprotocol']

    def __init__(self, foo=None, *args, **kwargs):
        super(MyMiddleware, self).__init__(*args, **kwargs)

        self.foo = foo
```

Then, you can instantiate your new middleware:

```
mid = Middleware.get_middleware_by_uri('myprotocol://?foo=bar')
assert mid.foo == 'bar'
assert mid.tourl() == 'myprotocol://?foo=bar'
```

NB: In order to be available for the `Middleware.get_middleware_by_uri()` method, the middleware **MUST** have been imported first (so the `register_middleware` decorator can do its job).

2.1.3 Constraints and child middleware

Sometimes, your middleware need another middleware to do its job, especially when your protocol can use multiple backends (i.e.: git via SSH or HTTPS).

Middlewares can have a child middleware specified in the URL:

```
git+ssh://user@remotehost/myproject.git
```

With the following instantiation:

```
child = Middleware.get_middleware_by_uri(
    'ssh://user@remotehost/myproject.git'
)
mid = MyGitMiddleware(
    username='user',
    hosts=[('remotehost', None)],
    path=['myproject.git']
)
mid.set_child_middleware(child)
```

Then, you can get the child middleware to use it:

```
child = mid.set_child_middleware()
```

Of course, not all middlewares are compatible, you can provide a list of accepted classes for child middlewares, and the `set_child_middleware()` method will check if the instance match the constraint:

```
from link.middleware.core import Middleware, register_middleware


@register_middleware
class MyDriver(Middleware):
```

```
    __protocols__ = ['mybackend']


@register_middleware
class MyMiddleware(Middleware):
    __protocols__ = ['myprotocol']
    __constraints__ = [MyDriver]
```

And then, someone can inherit from your driver:

```
@register_middleware
class MyCustomDriver(MyDriver):
    _protocols__ = ['mycustombackend']
```

And instantiate the whole thing:

```
mid = Middleware.get_middleware_by_uri(
    'myprotocol+mycustombackend://'
)
```

NB: There is no limit to how much children you want:

```
proto1+proto2+proto3://
```

Here:

- `proto3` is the child middleware of `proto2`
- `proto2` is the child middleware of `proto1`
- `proto1` will be returned by `Middleware.get_middleware_by_uri()`

Finally, the class method `get_middleware_by_uri()` verify that the new middleware is an instance of the calling class, thus, this code will raise an exception:

```
@register_middleware
class MyMiddleware(Middleware):
    __protocols__ = ['myprotocol']


@register_middleware
class NotMyMiddleware(Middleware):
    __protocols__ = ['notmyprotocol']


mid = MyMiddleware.get_middleware_by_uri('notmyprotocol://')
```

2.2 API documentation

2.2.1 link.middleware package

Submodules

link.middleware.core module

class `link.middleware.core.`**`Middleware`**(*user=None*, *pwd=None*, *hosts=None*, *path=None*, *fragment=''*, ***kwargs*)

Bases: `object`

Basic middleware class, resolvable via URLs.

exception `Error`

Bases: `exceptions.Exception`

classmethod `Middleware.constraints()`

Get all constraints enforced by class. A constraint is used when the middleware is instantiated with a child middleware (`protocol1+protocol2://`). The child middleware must be a subclass of each class specified by the constraint.

Returns list of constraints

Return type list

`Middleware.get_child_middleware()`

Get child middleware.

Returns Child middleware or None

Return type *Middleware*

classmethod `Middleware.get_middleware_by_uri(basecls, uri, cache=True)`

Resolve URI to instantiate a middleware.

Parameters

- **uri** (*str*) – URI pointing to middleware
- **cache** (*bool*) – Cache the instantiated middleware (default: True)

Returns Pointed middleware

Return type *Middleware*

Raises **basecls.Error** – if middleware is not an instance of `basecls`

static `Middleware.get_middlewares_by_protocols(protocols)`

Get list of middlewares implementing every listed protocol.

Parameters **protocols** (*str or list*) – list of protocols

Returns list of middleware

Return type list

classmethod `Middleware.protocols()`

Get all protocols supported by class.

Returns list of protocols

Return type list

`Middleware.set_child_middleware(middleware)`

Set child middleware (make sure the child middleware validates the middleware constraints).

Parameters **middleware** (`Middleware`) – Child middleware

`Middleware.tourl()`

Get URL from current middleware.

Returns URL pointing to this middleware.

Return type str

`link.middleware.core.register_middleware(cls)`

Register middleware's protocols.

link.middleware.connectable module

class `link.middleware.connectable.`**`ConnectableMiddleware`**(**args*, ***kwargs*)

Bases: `link.middleware.core.Middleware`

Middleware class connecting to a remote service.

`conn`

Returns internal connection (make sure the middleware is connected).

`connect`()

If not already connected, connect the middleware to the remote service.

`disconnect`()

If connected, disconnect the middleware.

`isconnected`()

Check if middleware is connected.

link.middleware.socket module

class `link.middleware.socket.`**`SocketMiddleware`**(**args*, ***kwargs*)

Bases: `link.middleware.connectable.ConnectableMiddleware`

Socket middleware.

`new_socket`(*host*, *port*)

Create a new socket (must be overriden).

Parameters

- **`host`** (`str`) – host to connect to
- **`port`** (`int`) – port to connect to

Returns socket object

`receive`(*bufsize*)

Fetch data from middleware.

Parameters **`bufsize`** – Size of data to fetch

Returns data read from middleware

`send`(*data*)

Send data to the middleware.

Parameters **`data`** – data to send

link.middleware.tcp module

class `link.middleware.tcp.`**`TCPMiddleware`**(**args*, ***kwargs*)

Bases: `link.middleware.socket.SocketMiddleware`

TCP socket middleware.

`new_socket`(*host*, *port*)

link.middleware.udp module

class `link.middleware.udp.`**`UDPMiddleware`**(**args*, ***kwargs*)

Bases: `link.middleware.socket.SocketMiddleware`

UDP Socket middleware.

`new_socket`(*host*, *port*)

`send`(*data*, *host*, *port*)

link.middleware.http module

class `link.middleware.http.`**`HTTPMiddleware`**(*user=None*, *pwd=None*, *hosts=None*, *path=None*, *fragment=''*, ***kwargs*)

Bases: `link.middleware.core.Middleware`

HTTP middleware.

exception `Error`

Bases: `link.middleware.core.Error`

Error class raised by middleware methods.

`HTTPMiddleware.`**`delete`**(*data*)

DELETE document pointed by middleware.

Parameters **`data`** (`dict`) – data used for DELETE request

Returns request response's content

Return type str

`HTTPMiddleware.`**`get`**()

GET document pointed by middleware.

Returns document's content

Return type str

`HTTPMiddleware.`**`head`**()

Get headers of document pointed by middleware.

Returns headers

Return type dict

`HTTPMiddleware.`**`options`**()

Check allowed requests on document pointed by middleware.

Returns list of allowed requests

Return type list

`HTTPMiddleware.`**`patch`**(*data*)

PATCH document pointed by middleware.

Parameters **`data`** (`dict`) – data used for PATCH request

Returns request response's content

Return type str

`HTTPMiddleware.`**`post`**(*data*)

POST data to document pointed by middleware.

Parameters **data** (*dict*) – data used by POST

Returns request response's content

Return type str

`HTTPMiddleware.put(data)`

PUT data to document pointed by middleware.

Parameters **data** (*dict*) – data used by PUT

Returns request response's content

Return type str

link.middleware.file module

class `link.middleware.file.FileMiddleware(user=None, pwd=None, hosts=None, path=None, fragment='', **kwargs)`

Bases: *link.middleware.core.Middleware*

Middleware with the same API as the *link.middleware.http.HTTPMiddleware* but with ''file://' URI.

delete(*data*)

Delete file pointed by middleware.

Parameters **data** – unused (only for API compatibility)

get()

Get content of file pointed by middleware.

Returns file's content

Return type str

head()

options()

patch(*data*)

post(*data*)

Append data to file pointed by middleware.

Parameters **data** (*str*) – data to append

put(*data*)

Write data to file pointed by middleware.

Parameters **data** (*str*) – data to write

Module contents

CHAPTER 3

Features

- associate URLs to business class (called middleware)
- **support for protocols:**
  - `http://`
  - `file://`
  - `tcp://`
  - `udp://`

CHAPTER 4

Donating

Python Module Index

l

link.middleware, 11
link.middleware.connectable, 9
link.middleware.core, 7
link.middleware.file, 11
link.middleware.http, 10
link.middleware.socket, 9
link.middleware.tcp, 9
link.middleware.udp, 10

C

conn (link.middleware.connectable.ConnectableMiddleware attribute), 9
connect() (link.middleware.connectable.ConnectableMiddleware method), 9
ConnectableMiddleware (class in link.middleware.connectable), 9
constraints() (link.middleware.core.Middleware class method), 8

D

delete() (link.middleware.file.FileMiddleware method), 11
delete() (link.middleware.http.HTTPMiddleware method), 10
disconnect() (link.middleware.connectable.ConnectableMiddleware method), 9

F

FileMiddleware (class in link.middleware.file), 11

G

get() (link.middleware.file.FileMiddleware method), 11
get() (link.middleware.http.HTTPMiddleware method), 10
get_child_middleware() (link.middleware.core.Middleware method), 8
get_middleware_by_uri() (link.middleware.core.Middleware class method), 8
get_middlewares_by_protocols() (link.middleware.core.Middleware static method), 8

H

head() (link.middleware.file.FileMiddleware method), 11
head() (link.middleware.http.HTTPMiddleware method), 10
HTTPMiddleware (class in link.middleware.http), 10
HTTPMiddleware.Error, 10

I

isconnected() (link.middleware.connectable.ConnectableMiddleware method), 9

L

link.middleware (module), 11
link.middleware.connectable (module), 9
link.middleware.core (module), 7
link.middleware.file (module), 11
link.middleware.http (module), 10
link.middleware.socket (module), 9
link.middleware.tcp (module), 9
link.middleware.udp (module), 10

M

Middleware (class in link.middleware.core), 7
Middleware.Error, 8

N

new_socket() (link.middleware.socket.SocketMiddleware method), 9
new_socket() (link.middleware.tcp.TCPMiddleware method), 9
new_socket() (link.middleware.udp.UDPMiddleware method), 10

O

options() (link.middleware.file.FileMiddleware method), 11
options() (link.middleware.http.HTTPMiddleware method), 10

P

patch() (link.middleware.file.FileMiddleware method), 11
patch() (link.middleware.http.HTTPMiddleware method), 10
post() (link.middleware.file.FileMiddleware method), 11
post() (link.middleware.http.HTTPMiddleware method), 10

protocols() (link.middleware.core.Middleware class method), 8
put() (link.middleware.file.FileMiddleware method), 11
put() (link.middleware.http.HTTPMiddleware method), 11

R

receive() (link.middleware.socket.SocketMiddleware method), 9
register_middleware() (in module link.middleware.core), 8

S

send() (link.middleware.socket.SocketMiddleware method), 9
send() (link.middleware.udp.UDPMiddleware method), 10
set_child_middleware() (link.middleware.core.Middleware method), 8
SocketMiddleware (class in link.middleware.socket), 9

T

TCPMiddleware (class in link.middleware.tcp), 9
tourl() (link.middleware.core.Middleware method), 8

U

UDPMiddleware (class in link.middleware.udp), 10