bind Documentation

Release 0.1

Rafe Kettler

Jun 14, 2017

Contents

Contents:

CHAPTER 1

bind

bind is a framework for writing Python bindings for web services. The aim of bind is to make writing web service bindings fast and simple, and to make it easy for maintainers of bindings to keep up with constantly changing and growing APIs.

Quickstart

Installation

To install `bind`, download a tarball or zip archive from **'our GitHub page'_**, extract it, change to the extracted directory, and run `python setup.py install`.

Usage

Defining an API in bind is very similar to defining a database table in many popular Python ORMs like SQLAlchemy or the Django ORM. Here's a sample binding:

```
from bind import Request, API

class MyAPI(API):
    BASE_URL = "http://mysite.com/api"

    get_user_info = Request("/users/:user", "GET")
    set_user_info = Request("/users/:user", "POST", requires_auth=True)
    ...
```

Usage is equally simple:

```
# Create a client for our API
api = MyAPI()
# Make an API call
```

```
user_info = api.get_user_info(user="Jane")
# Make some changes to our user info for Jane
...
# Now, let's authenticate and change some info
api.authenticate("<Jane's username>", "<Jane's password>")
api.set_user_info(changed_user_info, user="Jane")
```

Tutorial

Subclassing `bind.API`

Subclassing `API` is how you write your own API bindings with bind. You do so by defining class attributes for your subclass.

There are two kinds of class attributes that you can define in your subclass that are meaningful to bind. The first are constants that determine the general behavior of requests. Right now, there are three constants that you can define:

- `BASE_URL` determines the base URL for API requests
- `REQUEST_CALLBACK` is the default request callback for requests in your API
- `RESPONSE_CALLBACK` is the default response callback for requests in your API

The other meaningful type of class attribute is an instance of `bind.Request`. Each instance represents a particular API call. The instances are callable, so use the desired method name for that particular API call as the attribute name.

Finally, you can define methods and other attributes to your heart's content to provide conveniences to the programmer or the user. bind doesn't treat these attributes specially (or at all), so feel free to define any kind of attribute.

Defining requests in your subclass of `bind.API`

For each API call that you'd like to bind to (or for each request that you'd like the client to be able to make), you'll define a class attribute in your `API` subclass that is an instance of `bind.Request`. For example:

```
from bind import API, Request

class MyAPI(API):
    BASE_URL = "http://mysite.com/api"
    get_user_info = Request("/users/:user", "GET")
```

You'll notice a few important parameters that you've passed when instantiating `bind.Request`. The first is a *URL pattern*. URL patterns are of the form "/path/:param/otherpath/day:param2/". A section of a path can begin with a ':' to indicate a parameter in the URL. A parameter consumes characters until the next '/'. Patterns cannot be absolute paths, only relative. Then, the next argument is the HTTP request method for that particular request. There are also a few other optional arguments: `base_url`, which can override the class-level `BASE_URL`, as well as `request_callback` and `response_callback`, which can override the class level `REQUEST_CALLBACK` and `RESPONSE_CALLBACK`, respectively.

Defining your own callbacks

For request callbacks, a callback should take two arguments `headers` and `body` and return a tuple `(headers, body)` containing the transformed request.

For response callbacks, a callback should take two arguments `response` and `content` and return whatever data you like.

API

Callbacks

`bind` comes with a small library of commonly-used callbacks.

CHAPTER 2

Indices and tables

- genindex
- modindex
- search