Slumber Documentation

Release 0.4.0.dev.1

Donald Stufft

March 29, 2013

CONTENTS

Slumber is a python library that provides a convenient yet powerful object orientated interface to ReSTful APIs. It acts as a wrapper around the excellent requests library and abstracts away the handling of urls, serialization, and processing requests.

CHAPTER

ONE

GETTING STARTED WITH SLUMBER

1.1 Installation

Slumber is available on PyPi and the preferred method of install is using Pip.

1. Install Slumber:

   ```
   $ pip install slumber
   ```

2. Install Optional Dependencies

   [OPTIONAL] PyYaml *(Required for the yaml serializer)*:

   ```
   $ pip install pyyaml
   ```

   [OPTIONAL] SimpleJson *(Required for the json serializer on Python2.5, or for speedups)*:

   ```
   $ pip install simplejson
   ```

1.2 Using

Using Slumber is easy. Using an example ReST API made with django-tastypie which you can see at http://slumber.in/api/v1/.

```
>>> import slumber
>>> ## Connect to http://slumber.in/api/v1/ with the Basic Auth user/password of demo/demo
>>> api = slumber.API("http://slumber.in/api/v1/", auth=("demo", "demo"))
>>> ## GET http://slumber.in/api/v1/note/
>>> ##     Note: Any kwargs passed to get(), post(), put(), delete() will be used as url parameters
>>> api.note.get()
>>> ## POST http://slumber.in/api/v1/note/
>>> new = api.note.post({"title": "My Test Note", "content": "This is the content of my Test Note!"})
>>> ## PUT http://slumber.in/api/v1/note/{id}/
>>> api.note(new["id"]).put({"content": "I just changed the content of my Test Note!"})
>>> ## PATCH http://slumber.in/api/v1/note/{id}/
>>> api.note(new["id"]).patch({"content": "Wat!"})
>>> ## GET http://slumber.in/api/v1/note/{id}/
>>> api.note(new["id"]).get()
>>> ## DELETE http://slumber.in/api/v1/note/{id}/
>>> api.note(new["id"]).delete()
```

1.3 Url Parameters

Passing an url parameter to Slumber is easy. If you wanted to say, use Tastypie's ApiKey authentication, you could do so like:

```
>>> api.resource.get(username="example", api_key="1639eb74e86717f410c640d2712557aac0e989c8")
```

If you wanted to filter the Slumber demo api for notes that start with Bacon, you could do:

```
>>> import slumber
>>> api = slumber.API("http://slumber.in/api/v1/", auth=("demo", "demo"))
>>> ## GET http://slumber.in/api/v1/note/?title__startswith=Bacon
>>> api.note.get(title__startswith="Bacon")
```

1.4 Nested Resources

Nested resources are also easy and works just how a single level resource works:

```
>>> ## GET /resource1/resource2/
>>> api.resource1.resource2.get()

>>> ## GET /resource1/1/resource2/
>>> api.resource1(1).resource2.get()
```

CHAPTER TWO

OPTIONS

Slumber comes with only a couple options.

2.1 Authentication

Out of the box Slumber should support any authentication method supported by requests. These include Basic, Digest, OAuth. However only Basic and Digest get tested currently.

2.1.1 Specify Authentication

Specifying authentication credentials is easy. When you create your slumber api instance, instead of doing:

```
api = slumber.API("http://path/to/my/api/")
```

You supply the username and password (for Basic Auth) like:

```
api = slumber.API("http://path/to/my/api/", auth=("myuser", "mypass"))
```

And slumber will attempt to use those credentials with each request.

To Use Digest or OAuth please consult the requests documentation. The auth argument is passed directly to requests and thus works exactly the same way and accepts exactly the same arguments.

2.2 Serializer

Slumber allows you to use any serialization you want. It comes with json and yaml but creating your own is easy. By default it will attempt to use json. You can change the default by specifying a `format` argument to your api class.:

```
# Use Yaml instead of Json
api = slumber.API("http://path/to/my/api/", format="yaml")
```

If you want to override the serializer for a particular request, you can do that as well:

```
# Use Yaml instead of Json for just this request.
api = slumber.API("http://path/to/my/api/") # Serializer defaults to Json
api.resource_name(format="yaml").get() # Serializer will be Yaml
```

If you want to create your own serializer you can do so. A serialize inherits from `slumber.serialize.BaseSerializer` and implements `loads`, `dumps`. It also must have a class member of `key` which will be the string key for this serialization (such as "json").The final requirement is either a class member of `content_type` which is the content type to use for requests (such as "application/json") or define a `get_content_type` method.

An example:

```
class PickleSerializer(slumber.serialize.BaseSerializer):
    key = "pickle"
    content_type = "x-application/pickle"

    def loads(self, data):
        return pickle.loads(data)

    def dumps(self, data):
        return pickle.dumps(data)
```

Once you have a custom serializer you can pass it to slumber like so:

```
from slumber import serialize
import slumber

s = serialize.Serializer(
                default="pickle",
                serializers=[
                    serialize.JsonSerializer(),
                    serialize.YamlSerializer(),
                    PickleSerializer(),
                ]
            )
api = slumber.API("http://example.com/api/v1/", format="pickle", serializer=s)
```

2.3 Slashes

Slumber assumes by default that all urls should end with a slash. If you do not want this behavior you can control it via the append_slash option which can be set by passing append_slash to the `slumber.API` kwargs.

CHAPTER

THREE

HOW SLUMBER WORKS BEHIND THE SCENES

Behind the scenes slumber is really 2 things. It is a translator from python code into urls, and it is a serializer/deserializer helper.

3.1 Python to Url Translation

The Url Translation portion of slumber is fairly simple but powerful. It is basically a list of url fragments that get joined together.

In the call:

```
>>> api.note.get()
```

Slumber translates this by adding the url value for `api` (say http://slumber.in/api/v1/) with the url value for note. The url value for an attribute is equal to it's name, so in this case slumber essentially does `"http://slumber.in/api/v1/" + "note"`,

The same holds true when you pass in an ID to a resource. It just adds another segment to the url list.

This means that in this call:

```
>>> api.note(1).get()
```

Slumber translates it to:

```
>>> "http://slumber.in/api/v1/" + "note" + "/" + str(1)
```

Nested Resources follow the same principle.

The other part of an url that Slumber translates is kwargs to `get()`, `post()`, `put()`, `delete()` into query string params. This again is a fairly simple operation which then gets added to the end of the url.

The Final portion of Slumber's Python to HTTP is that the first arg passed to each of the HTTP functions is serialized, and then passed into the HTTP request as the body of the request.

3.2 (De)Serialization

Slumber also has a built in Serialization framework. This is a fairly simple wrapper around (simple)json and/or pyyaml that controls setting the correct content-type on a request, and will automatically serialize or deserialize any incoming or outgoing request body.

CHAPTER

FOUR

GETTING HELP

There are two primary ways of getting help. I have an IRC channel (#slumber on irc.freenode.net) to get help, want to bounce idea or generally shoot the breeze.

CHAPTER FIVE

QUICKSTART

1. Install Slumber:

   ```
   $ pip install slumber
   ```

2. Install Optional Requirements:

   ```
   pip install simplejson pyyaml
   ```

3. Use Slumber!

CHAPTER

SIX

REQUIREMENTS

Slumber requires the following modules:

- Python 2.5+
- requests
- simplejson (If using Python 2.5, or you desire the speedups for JSON serialization)
- pyyaml (If you are using the optional yaml serialization)

Testing Slumber requires the following modules:

- Mock

CHAPTER

SEVEN

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*