VirginiaTech
Invent the Future

RESTful Tools

ECE 4564 - Network Application Design

Dr. William O. Plymale

ece
BRADLEY DEPARTMENT
of ELECTRICAL & COMPUTER ENGINEERING

Topics

- cURL
- Python Requests
- Web Microframeworks

cURL

- The name "cURL" stands for "client URL“
- A tool that allows you to interact with URLs to get a variety of work tasks done

$ curl http://www.vt.edu

- show the web page at this address
  - makes a GET request, fetch the page and displays the corresponding HTML code

cURL

- Download web pages and upload files
- Post data to web sites
- Converse with web sites using wide range of protocols
  - HTTP
  - DICT
  - FTP
  - SMTP

cURL

- Interacts with web-based APIs like REST
- Ported to many operating systems including Linux, Mac OS and Windows.

cURL Tutorial

cURL Examples

1. curl http://wttr.in/LOCATION
2. curl wttr.in/Moon
3. curl http://artscene.textfiles.com/asciiart/panda

Get cURLy

Requests

- Requests is an Apache2 Licensed HTTP library, written in Python,
- Requests allow you to send HTTP/1.1 requests.
- You can add headers, form data, multipart files, and parameters with simple Python dictionaries, and access the response data in the same way.

Requests:HTTP for Humans

HTTP Requests Types

```
r = requests.get('https://github.com/timeline.json')

r = requests.post("http://httpbin.org/post")

r = requests.put("http://httpbin.org/put")

r = requests.delete("http://httpbin.org/delete")

r = requests.head("http://httpbin.org/get")

r = requests.options("http://httpbin.org/get")
```

Requests Example

```
>>> import requests
>>> r = requests.get('https://github.com/timeline.json')
>>> r.text
```

URL Parameters

```
>>> import requests
>>> payload = {'key1': 'value1', 'key2': 'value2'}
>>> r = requests.get('http://httpbin.org/get', params=payload)

>>> print(r.url)
```

Status Codes and Headers

```
>>> import requests
>>> r = requests.get('http://httpbin.org/get')
>>> r.status_code

>>>r.headers
>>> r.headers['Content-Type']
```

Response Content

```
>>> import requests
>>> r = requests.get('https://api.github.com/events')
>>> r.text


>>>r.content       //for non-text requests

>>>r.json()        //builtin JSON decoder
```

POST Requests

```
>>> import requests
>>> payload = {'key1': 'value1', 'key2': 'value2'}

>>> r = requests.post("http://httpbin.org/post", data=payload)
>>> print(r.text)
```

Python Requests Tutorial

Python API Tutorial

How to Call a Weather API

Web Framework

- Web Framework represents a collection of libraries and modules that enables a web application developer to write applications without having to bother about low-level details such as protocols, thread management etc.

- Common Web Framework Functionality:
  - URL routing
  - HTML, XML, JSON, and other output format templating
  - Database manipulation
  - Security Support
  - Session Storage and Retrieval

Full-Stack vs Micro

Frameworks fall on the spectrum from executing a single use case to providing every known web framework feature to every developer.

Full-Stack

- An enterprise grade business application
- Django

Micro

- An API focused application
- Flask

Best Python Microframeworks

Flask

- Flask is a micro web development framework for Python.
- Developed by Armin Ronacher, who leads an international group of Python enthusiasts named Pocco.
- Based on
  - Werkzeug WSGI toolkit
    - Web Server Gateway Interface
    - Implements requests, response objects, and other utility functions
  - Jinja2 template engine
    - web templating system combines a template with a certain data source to render dynamic web pages

Flask and REST

“The task of designing a web service or API that adheres to the REST guidelines is an exercise in identifying the resources that will be exposed and how they will be affected by the different request methods.”

Designing a RESTful API with Python and Flask

Flask and MongoDB

Flask Rest API with MongoDB

Flask and Authentication

HTTP Basic Authentication

Closing