Lab 1 : RESTful Web Services Part 1

E222 : Intelligent Systems II

18, January, 2018

1 Introduction

In this lab session we will focus on how to make a RESTful web service with Python Eve. In this tutorial we will focus on installing the required packages for python and we will build a simple web service to retrieve information.

2 Prerequisites

Ubuntu 16.04 is the recommended OS for the labs. We recommend you to use a virtual box to install the OS in case you don't have access to the OS being installed in your workstation.

3 Installation

Initially we need to install some packages as follow :
Installing pip :

```
$ sudo apt-get install python-pip
```

Installing eve :

```
$ pip install eve
```

Let's install MongoDB. The purpose of installing MongoDB is to store data. MongoDB is a Non-SQL database which helps to store light weight data very quickly.
Installing MongoDB :

```
$ sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv
2930ADAE8CAF5059EE73BB4B58712A2291FA4AD5

$ echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu xenial/
mongodb-org/3.6 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-3.6.list

$ sudo apt-get update
```

```
$ sudo apt-get install -y mongodb-org
```

In order to check the MongoDB installation, please run the following commands

```
$ sudo service mongod start
$ mongo --host 127.0.0.1:27017
$ show databases
```

If they execute without errors, you have successfully installed MongoDB. In order to stop the running database instance run the following command.

```
$ sudo service mongod stop
```

4 Building REST API

In this section we'll focus on building up the infrastructure for the REST API.

4.1 Creating Working Directory

```
$ mkdir ~/eve-tutorial
$ cd ~/eve-tutorial
```

First create **settings.py** file in the working directory.

```
DOMAIN = {
    'student': {
        'schema': {
            'firstname': {
                'type': 'string'
            },
            'lastname': {
                'type': 'string'
            },
            'school': {
                'type': 'string'
            },
            'university': {
                'type': 'string'
            },
            'email': {
                'type': 'string',
                 'unique': True
            }
        }
    }
}
RESOURCE_METHODS = ['GET', 'POST']
```

This is the format of the Student object that we are trying to call from REST API. In this format we can use GET protocol to retrieve a particular student by any of the attributes in the Schema and also we can use the above schema to save a student object.

Note : In this tutorial, the MongoDB processes and how it works won't be explained in detail. We will focus on how REST can be used to retrieve and save information in a database. MongoDB is not the only database that can be used to save data, but it is an easy installation process and a quicker way to access data using terminal.

Initially we need to create the python application to host the REST API and we will use CURL to do the GET and POST requests in this tutorial.

4.2 Creating REST Service

Create a file called **run.py** in the same directory where you placed the **settings.py** and add the following code.

```
from eve import Eve
app = Eve()

if __name__ == '__main__':
    app.run()
```

This is the minimal application which can host a REST application. Let's start our database server and the REST service.

4.3 Running REST Service

Let's say we'll use terminal 1 to run the python REST API.

To start the rest service :

```
$python run.py
```

4.4 Running Database Server

Next in terminal 2, run the following command to start the database server

```
$ mongod --dbpath=<BASE-PATH>/data/db/
```

For example : BASE-PATH can be replaced by any path you want.

```
$ mongod --dbpath=/home/john/e222/labs/lab1/data/db/
```

4.5 Saving Data Using Curl

We can use the commandline to save the data in the database using the REST api. Here we use the endpoint we created as student in the schema as the entry point to the rest service. The data is being saved using JSON format.

Run the following command in the terminal 3, this will save data in the database server.

```
$ curl -H "Content-Type: application/json" -X POST -d
'{"firtname":"John","lastname":"Doe", "school" : "ISE",
"university":"IUB", "email":"johndoe@iu.edu"}' http://127.0.0.1:5000/student/
```

In order to check the database please run the following commands in Terminal 4,

```
Run mongo client
$ mongo
Query the databse list
$>show databases; (will display eve)
Select database
$use eve
$ show tables # query the table names
$ db.student.find().pretty()  # pretty will show the json in a clear way
```

Here we specify the header using **H** tag saying we need the data to be saved using json format. And **X** tag says the HTTP protocol and here we use POST method. And the tag **d** specifies the data and make sure you use json format to enter the data. Finally, the REST api endpoint to which we must save data. This allows us to save the data in a table called **student** in MongoDB within a database called **eve**.

4.6 Retrieving Data Using Curl

Run the following command to retrieve the saved data :

```
$ curl http://127.0.0.1:5000/student?firstname=John
```

Here you can use any of the attributes in the saved object to call the value. If you just used the endpoint itself without specifying the attribute name, it will load all the saved objects.

5 Exercises

5.1 Payloads

For this section we will ensure that the proper payloads are received upon a GET and DELETE request. For this to work ensure that you have a MongoDB up and running.

```
$ sudo service mongod start
```

We will now create the proper script to launch the API and an appropriate configuration file.
The configuration file (settings.py) should have the following line.

```
DOMAIN = {'people': {}}
```

In the same directory as the the configuration file you need to create a launch script i.e. run.py with the following lines.

```
from eve import Eve
app = Eve()

if __name__ == '__main__':
    app.run()
```

Now that you have an instance of MongoDB running and the two newly created .py files located in the same directory we can launch the API. In a new terminal navigate to the directory containing the configuration and launch files and execute the launch script. Again make sure this is in a seperate terminal so you can see what's happening as you make requests.

```
$ python run.py
* Running on http://127.0.0.1:5000/
```

Now consume the API with the following command and inspect the payload and identify what request was made.

```
$ curl -i http://127.0.0.1:5000
```

Your payload should look like the one below, if your output isn't formatted like below try adding ?pretty=1 (curl -i http://127.0.0.1:5000?pretty=1) to the end of the curl command.

```
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 150
Server: Eve/0.7.6 Werkzeug/0.11.15 Python/2.7.5
Date: Wed, 17 Jan 2018 18:34:07 GMT

{
    "_links": {
        "child": [
            {
                "href": "people",
                "title": "people"
            }
        ]
    }
```

Remember that the API entry points adhere to the HATEOAS principle, thus provide information regarding the resources accessible through the API. In this case how many child resources are available through our API?

If you said (1) you're correct! It's the one resource we defined in the configuration file 'people'. Now lets try to request people.

```
$ curl -i http://127.0.0.1:5000/people
```

How does the payload change? What does the _links section describe? Notice the difference between the payload when people is requested.


```
{
    "_items": [],
    "_links": {
        "self": {
            "href": "people",
            "title": "people"
        },
        "parent": {
            "href": "/",
            "title": "home"
        }
    },
    "_meta": {
        "max_results": 25,
        "total": 0,
        "page": 1
    }
}
```


6 Homework

Write a RESTful service to determine a useful piece of information off of your computer i.e. disk space, memory, RAM, etc.