

Restful Web Services Software Design Document

08/14/2016

Table of Contents

Content	Page No.
1) Introduction	3
1.1) Purpose	3
2) RESTful Web services	3
3) Dataset	3
4) Architecture	4
4.1) Web page Design	4
4.2) Usage manual	5
4.3) Description of APIs	5
5) Online Hosting	5
6) Testing	5
7) Conclusion	5
8) APPENDIX	6

1) Introduction:

This software design document gives detailed description of the application, use cases and source code can be found in appendix. This application is created as a part of the interview process for the Software Engineer position at Return Path.

1.1) Purpose:

The purpose of this application is to create a RESTful web service with three endpoints and host the data. The data further to be accessed from a single website.

2) RESTful Web services:

Going by the definition of Wikipedia, representational state transfer (REST) is an architectural style consisting of a coordinated set of components, connectors, and data elements within a distributed hypermedia system, where the focus is on component roles and a specific set of interactions between data elements rather than implementation details. Its purpose is to induce performance, scalability, simplicity, modifiability, visibility, portability, and reliability. REST is the software architectural style of the World Wide Web.

Applied to web services, the REST api's gives a robust architectural model to contact the host. HTTP based RESTful APIs are defined with respect to the absolute URL, the media type and HTTP methods. The HTTP methods are supported by different verbs such as GET, HEAD, POST, PUT, DELETE, etc.

3) Dataset:

Data in the JavaScript Object Notation(JSON) has been used as test data. The dataset has been generated from <https://www.mockaroo.com/> by specifying set a set of fields.'

The following fields can be found in the dataset - Id, First Name, Last name, Country, email, gender, job title. There are a total of 100 records in the considered dataset.

JSON format has been chosen because of its efficiency reasons. Parsing of JSON dataset is comparatively easier. Most of modern high level languages support the parsing of JSON datasets. JSON is faster and lightweight compared to XML. JSON also gives a compact representation. Also, looking for a better data exchange format makes JSON a preferred choice when compared to XML.

4) Architecture:

The architecture of the current project is designed as follows. Python along with Flask and Flask-restful is used as the programming language. The application developed will support “GET” request from the host and response will be given based on the requested resource. The resources are hosted in different endpoints. The endpoints for this web service are defined as follows.

The root “/” contains the main webpage. A separate HTML file supports this for browsers. From the landing page, based upon the request, the data is delivered to the user. The endpoint defined by “/emp” contains the complete dataset. To get the record by its id the endpoint “/emp/emp-id/<id>” is used. The webpage is designed in such a way to enter the <id> and retrieve the record. To know the employee IDs an endpoint has been created. “/emp/emp-ids” endpoint returns the list of employee IDs found in the dataset. As an added feature, the employee can also be filtered based on the country. The name of country is sent as query in the http request. This is supported by this endpoint - “/emp/emp-country?country=<name>”. The figure (1) describes the hierarchy of the endpoints of the server.

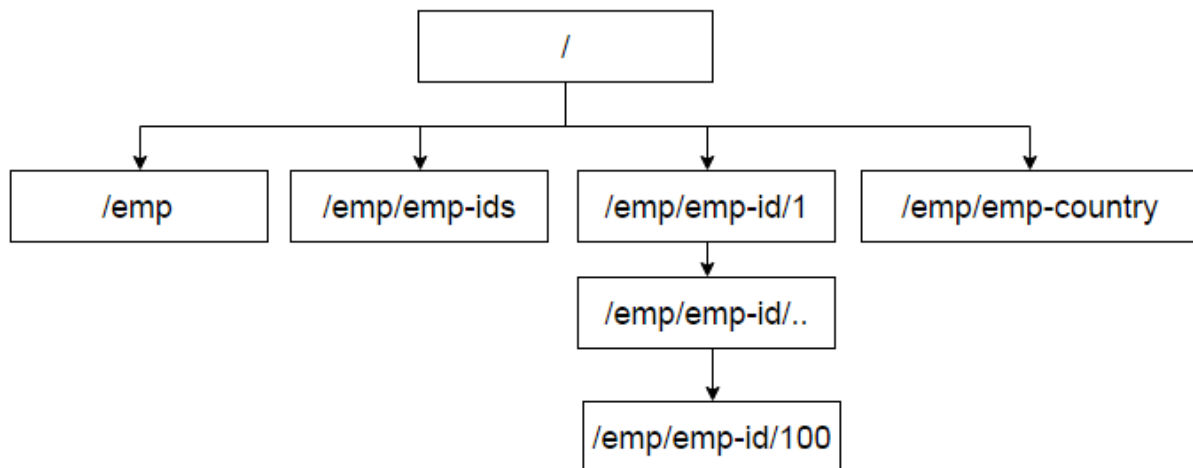


Fig (1)

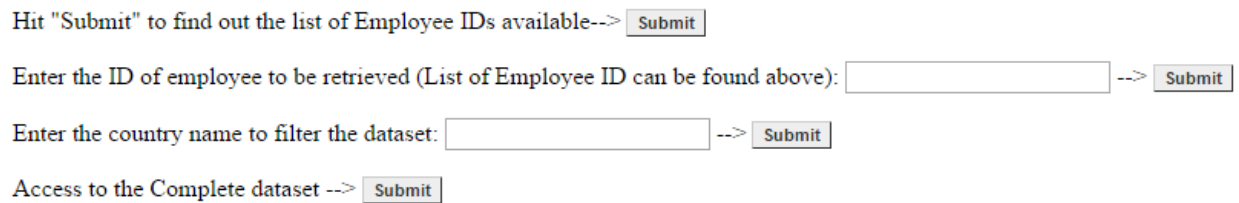
4.1) Web page Design:

The webpage rendered contains the following features. JavaScript has been written to capture and validate the entry in the text box and the request is forwarded to server. Description of the features is given below.

4.2) Usage manual:

- 1) The first option gives the list of employee IDs available.
- 2) The second feature retrieves the record of the given employee ID.
- 3) The third feature retrieves the records for a particular country.
- 4) The final feature returns the complete dataset.

The figure (2) gives the snapshot from the webpage



Hit "Submit" to find out the list of Employee IDs available-->

Enter the ID of employee to be retrieved (List of Employee ID can be found above): -->

Enter the country name to filter the dataset: -->

Access to the Complete dataset -->

Fig (2)

4.3) Description of APIs

The landing page is handled by the class "root". The class "emp_ids" responds to "/emp/emp-ids" returning the complete list of employee IDs available. The class "employee_by_id" responds to "/emp/emp-id/.." to return the record of given employee. The class "employee_by_country" responds for "/emp/emp-country" to respond for the query of country. The class "emp" responds to "/emp" to return the complete records.

5) Online Hosting:

This application has been hosted online and the link for the same is <http://webhost.pythonanywhere.com/>. The service is hosted with help of pythonanywhere(<https://www.pythonanywhere.com>) service. The website can be hosted as a stand-alone application along with the dataset "dataset.json" that is attached with this. As described in section (4.2) the data can be accessed from the given website.

6) Testing:

The smaller modules of code have been tested separately. The code also has been validated for different types of inputs such as int, string. For example the string country has been checked by converting the entire text into lower case. The employee id has been checked for integer type or else an appropriate response has been given for that.

7) Conclusion:

The application created has been designed as specified in the design document. The requirements have been met and application functions well.

APPENDIX

Source Code:

A1)

The main application which is running on the server is as follows,

```
#####app.py#####  
  
from flask import Flask,url_for,request,json,render_template,Markup,Response  
from flask_restful import reqparse, abort, Api, Resource  
import json as JSON  
  
app=Flask(__name__)  
api = Api(app)  
  
global z  
global list_of_id  
global list_of_cry  
  
class employee_by_id(Resource) :  
    def get(self,emp_id):  
        try:  
            emp_id = int(emp_id)  
        except:  
            return "Not an valid employee id"  
  
        if not emp_id in list_of_id:
```

```

        ret = 'Record not found with the given id'
        return ret
    else:
        for record in z:
            if record['id']==emp_id:
                ret_string = JSON.dumps(record)
                return Response(ret_string,mimetype='text/plain')

class emp(Resource):
    def get(self):
        global z
        ret_string = "\n".join(JSON.dumps(item,separators=(',',':')) for item in z)
        return Response(ret_string,mimetype='text/plain')

class root(Resource):
    def get(self):
        return Response(render_template('index.htm'),mimetype='text/html')

class employee_by_country(Resource):
    def get(self):
        parser = reqparse.RequestParser()
        parser.add_argument('country',type=str)
        args = parser.parse_args()
        country = args.get('country')
        country=country.lower()
        records = []
        if not country in list_of_cry:

```

```

        ret = 'No records with country:'+country
        return ret
    else:
        for record in z:
            check = record['country'].encode('ascii','ignore')
            check = check.lower()
            if check==country:
                records.append(record)
        ret_string = "\n".join(JSON.dumps(item,separators=(',',':')) for item in records)
        return Response(ret_string,mimetype='text/plain')

```

```

class emp_ids(Resource):

```

```

    def get(self):
        global list_of_id
        return list_of_id

```

```

api.add_resource(emp, '/emp')
api.add_resource(emp_ids, '/emp/emp-ids')
api.add_resource(employee_by_id, '/emp/emp-id/<emp_id>')
api.add_resource(employee_by_country, '/emp/emp-country', endpoint='emp-country')
api.add_resource(root, '/')

```

```

if __name__ == '__main__':
    global z
    global list_of_id
    z = json.loads(open('dataset.json','r').read())
    list_of_id = [record['id'] for record in z]

```



```
list_of_cry = [record['country'].lower() for record in z]
list_of_cry = [record.encode('ascii','ignore') for record in list_of_cry]
app.run()
```

A2) The HTML web page source code is detailed below,

```
<!DOCTYPE html>
<html>
<head>
<style>
.button {
font: bold 11px Arial;
text-decoration: none;
background-color: #EEEEEE;
color: #333333;
padding: 2px 6px 2px 6px;
border-top: 1px solid #CCCCCC;
border-right: 1px solid #333333;
border-bottom: 1px solid #333333;
border-left: 1px solid #CCCCCC;
}
</style>
</head>
<body>
<h1>Welcome to the Data Hosting Service</h1>
<p><h5>A set of sample employee data has been hosted in this site. There are three ways
to access the data. <br>
1) Get a List of employee ID's <br>
2) Get the record by employee ID <br>
```

3) Records of employees by country (Query)

4) Complete Dataset</h5>

</p>

<p>

Hit "Submit" to find out the list of Employee IDs available-->

<button class="button" onclick="myFunction3()">Submit</button>

<script language="javascript" type="text/javascript">

function myFunction3() {

 var url = '/emp/emp-ids';

 window.location = url;

}

</script>

</p>

<p>

Enter the ID of employee to be retrieved (List of Employee ID can be found above): <input type="text" id="emp-id" autocomplete="off" > -->

<button class="button" onclick="myFunction()">Submit</button>

<script language="javascript" type="text/javascript">

function myFunction() {

 var x = document.getElementById("emp-id").value;

 var url = '/emp/emp-id/'+x;

 if(x.length < 1)

 {window.alert("Employee ID field is blank");}

 else{window.location = url;}

}

</script>

</p>

<p>

Enter the country name to filter the dataset: <input type="text" id="emp-country" autocomplete="off"> -->

<button class="button" onclick="myFunction2()">Submit</button>

<script language="javascript" type="text/javascript">

function myFunction2() {

var x = document.getElementById("emp-country").value;

var url = '/emp/emp-country?'+ 'country='+x;

if(x.length < 1)

{window.alert("Country field is blank");}

else{window.location = url;}

}

</script>

</p>

<p>

Access to the Complete dataset -->

<button class="button" onclick="myFunction4()">Submit</button>

<script language="javascript" type="text/javascript">

function myFunction4() {

var url = '/emp';

window.location = url;

}

</script>

</p>

</body>

</html>