**Add Campaign App Specs**

**Environment**: Spring-Boot, Rest Web services, Maven, Java 8.0, Postman or Browser for testing.

* Download the App from GIT HUB.
* Open the Command (Windows) or Console (Mac) and Navigate to Pom.xml

To Clean : mvn clean

To Compile : mvn compile

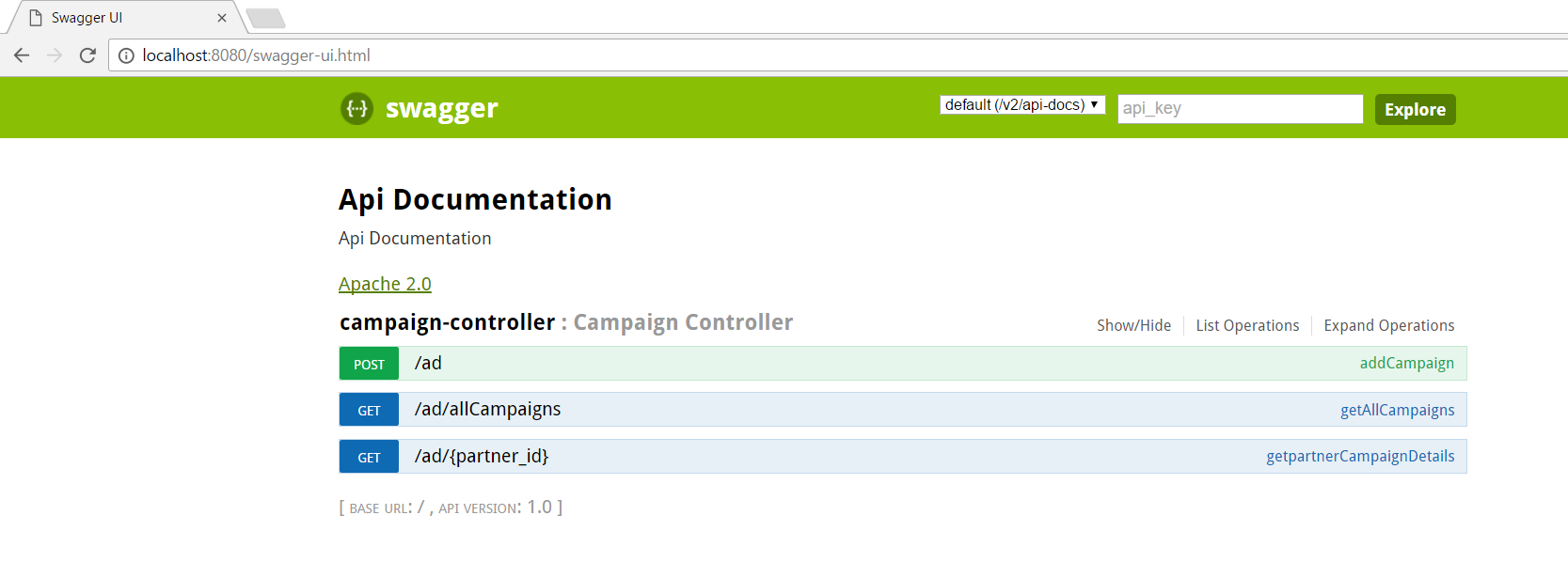
To Run: mvn spring-boot:run

To generate Jar : mvn package

To Launch : To launch the application you can use Rest clients like Postman or use Swagger UI.

This App has in-built Swagger-UI plugin.

Swagger: <http://localhost:8080/swagger-ui.html>



Sample Test Data :

--------- Partner1 Campain 1--------------------

{"partner\_id":"123",

"duration":90,

"ad\_content":"Campaign1\_1"

}

--------- Partner1 Campain 2--------------------

{"partner\_id":"123",

"duration":180,

"ad\_content":"Campaign1\_2"

}

--------- Partner1 Campain 3--------------------

{"partner\_id":"123",

"duration":30,

"ad\_content":"Campaign1\_3"

}

--------- Partner2 Campain 1--------------------

{"partner\_id":"456",

"duration":20,

"ad\_content":"Campaign2\_1"

}

--------- Partner2 Campain 2--------------------

{"partner\_id":"456",

"duration":40,

"ad\_content":"Campaign2\_2"

}

**Requirements 1:**

== Create Ad Campaign via HTTP POST

A user should be able to create an ad campaign by sending a POST request to the ad server at http://<host>/ad. The body of the POST request must be a JSON object containing the following data:

{

"partner\_id": "unique\_string\_representing\_partner',

"duration": "int\_representing\_campaign\_duration\_in\_seconds\_from\_now"

"ad\_content": "string\_of\_content\_to\_display\_as\_ad"

}

The server should enforce the following invariant upon receiving a request to create a new campaign.

\* Only one active campaign can exist for a given partner.

If an error is encountered, the ad server must return an appropriate response and indicate the problem to the user. The response format is left up to the discretion of the implementer.

Storing campaign data in memory or a cookie is totally fine.

**Solution:**

1. Storing Campaign Data : Used HashMap and LinkedList
2. Create an Ad Campaign by Posting a request using below URL:

<http://localhost:8080/ad/>

Content-Type - application/json

Test data: Refer above section

**Requirements 2:**

== Fetch Ad Campaign for a Partner

A partner should be able to get their ad data by sending a GET request to the ad server at http://<host>/ad/<partner\_id>. Response can be delivered as a JSON object representing the active ad

If the current time is greater than a campaign's creation time + duration, then the server's response should be an error indicating that no active ad campaigns exist for the specified partner.

**Solution :**

The Campaign List for a Partner can be retrieved using

[http://localhost:8080/ad//<partner\_id](http://localhost:8080/ad//%3cpartner_id)>

In the response you will find partner\_id, duration, ad\_Content, startTime,endTime.

StartTime = Start Time of Campaign

EndTime= Start Time + Duration

**Test Data 1:**

<http://localhost:8080/ad/123>

**Response:**

[

{

"partner\_id": "123",

"duration": 90,

"ad\_content": "Campaign1\_1",

"startTime": "05/02/2017 23:09:29",

"endTime": "05/02/2017 23:10:59"

},

{

"partner\_id": "123",

"duration": 180,

"ad\_content": "Campaign1\_2",

"startTime": "05/02/2017 23:09:42",

"endTime": "05/02/2017 23:12:42"

},

{

"partner\_id": "123",

"duration": 180,

"ad\_content": "Campaign1\_2",

"startTime": "05/02/2017 23:09:46",

"endTime": "05/02/2017 23:12:46"

}

]

**Test Data2:**

<http://localhost:8080/ad/456>

**Response** :

[

{

"partner\_id": "456",

"duration": 60,

"ad\_content": "Campaign2\_1",

"startTime": "05/02/2017 23:09:54",

"endTime": "05/02/2017 23:10:54"

},

{

"partner\_id": "456",

"duration": 140,

"ad\_content": "Campaign2\_2",

"startTime": "05/02/2017 23:10:00",

"endTime": "05/02/2017 23:12:20"

}

]

**Requirements 3:**

\* Add a URL to return a list of all campaigns as JSON.

**Solution:**

<http://localhost:8080/ad/allCampaigns>

Response:

{

"123": [

{

"partner\_id": "123",

"duration": 90,

"ad\_content": "Campaign1\_1",

"startTime": "05/02/2017 23:09:29",

"endTime": "05/02/2017 23:10:59"

},

{

"partner\_id": "123",

"duration": 180,

"ad\_content": "Campaign1\_2",

"startTime": "05/02/2017 23:09:42",

"endTime": "05/02/2017 23:12:42"

},

{

"partner\_id": "123",

"duration": 180,

"ad\_content": "Campaign1\_2",

"startTime": "05/02/2017 23:09:46",

"endTime": "05/02/2017 23:12:46"

}

],

"456": [

{

"partner\_id": "456",

"duration": 60,

"ad\_content": "Campaign2\_1",

"startTime": "05/02/2017 23:09:54",

"endTime": "05/02/2017 23:10:54"

},

{

"partner\_id": "456",

"duration": 140,

"ad\_content": "Campaign2\_2",

"startTime": "05/02/2017 23:10:00",

"endTime": "05/02/2017 23:12:20"

}

]

}

**Requirements 4:**

\* Add support for multiple ad campaigns per partner.

**Solution:**

Please find the above test cases for this scenario.

**Requirements 5:**

\* Describe a fault tolerant deployment topology for your application, and the types of failures it would and would not be resilient to.

**Solution:**

Case 1: If the PartnerId is worng or if user trying to retrieve invalid PartnerID then a Error message is thrown back to the user

<http://localhost:8080/ad/9101123>

Response: [

"No Partner is availalbe"

]

Case 2: If the App will return only Active campaigns for the user. If current time is greater than a campaign's creation time + duration then the Campaign will be removed from the response. If No active campaigns available for the user then a Error message will be thrown back :

[

"No Active Cmpaigns for this Partner"

]

**Requirements 6:**

\* Discuss the advantages and disadvantages of your persistence mechanism.

**Solution:**

We can use Hibernate -- but since it a sample test case, No DB Connections are made. Instead Collections are used to store Campaign data.

LinkedList : Data Manipulation is faster

HashMap: Not synchronized and hence performance is not impacted