

COP 5615 : Fall 2015

FACEBOOK API PROJECT

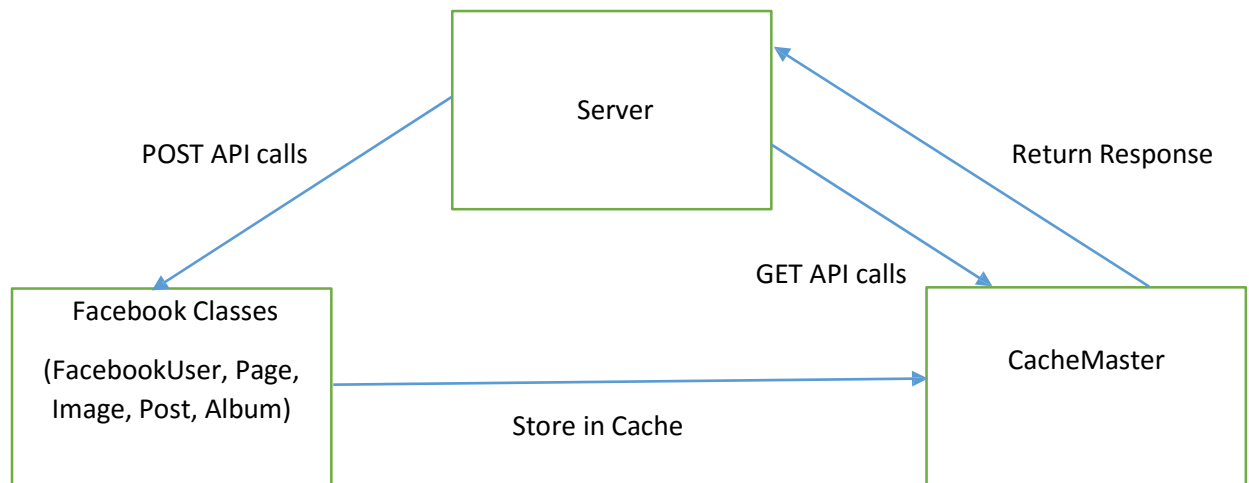
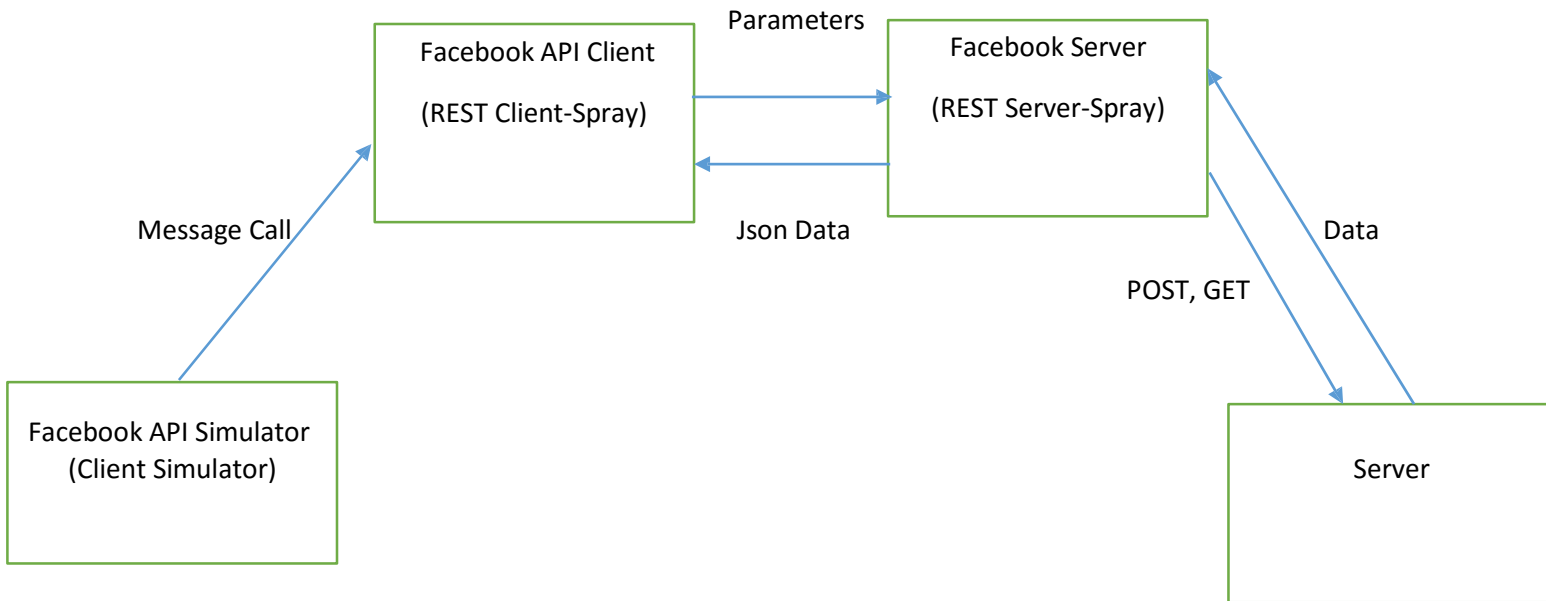
REPORT

Project Members:

Name: Jyotsana Walia

Name: Prerna Prerna

Facebook Architecture



1. **Facebook API Simulator** – It acts as a Client Simulator which triggers messages to REST Client. Simulator has been modelled according to the real Facebook statistics as explained below under heading “Client Simulator”.
2. **Facebook API Client** – It acts as a REST Client which makes REST API calls (POST and GET) to the REST Server. It contains all the APIs for various functionalities of Facebook, such as, user creation, creating a post, creating images, sharing and liking posts, etc.
3. **Facebook Server** – It acts as a REST Server which handles the incoming API calls (POST and GET) and then redirects them to the main Server for processing.
4. **Server** – It is responsible for executing all the functionalities of Facebook. It is composed of –
 - a. FacebookUser – It is the Facebook profile of the user.
 - b. Page – It is a Facebook page.
 - c. Post – It stores all the posts of a Facebook user.
 - d. Image - It stores all the images of a Facebook user.
 - e. Album - It stores all the images compiled in an album of a Facebook user.
5. **CacheMaster** – It acts as an in-memory Cache for all the content present on Facebook. It enables fast retrieval of the content and eliminates the need of hitting the Main server for incoming GET requests. It also manages to refresh the data on updation.

Client Simulator

Client Simulator has been implemented based on real Facebook statistics. Following are the statistics used to simulate if not exactly –

- Facebook has more than 500 million users out of which only 50% are active on any given day. So in our project which runs for maximum 100000 users, we deal with only 50% of users for API implementations.
- Average Facebook user makes 8 requests per day. So in our project, friend list of a user has 8% of all the active Facebook users.
- Facebook has 33% male and rest female. This has been implemented as it is on our project.
- 30% Facebook users have their country as USA, same has been implemented in our project.
- 55 million status updates are made on Facebook on any given day which makes 10% of the total users (500 million). So in our simulator only 10% of total users make status updates.
- Out of total status updates made 10% are in the form of posts. So our simulator has only 10% Post related API requests.
- According to the statistic, maximum posts are made in the form of image posts, so the 10% Post related API requests has 5% image post requests, 2% text post requests and 3% share post requests.
- Out of total active users, only 40-50% read posts. So the API request for reading posts is made for only 40% of the total active users.

Features

1. Data Validation

- a. A Facebook user1 can read the posts of Facebook user2 after verifying that user1 is present in the friend list of user2.

- b. A Facebook user1 can share the post of Facebook user2 after verifying that user1 is present in the friend list of user2 and the given post which user1 wants to share is made by user2.
 - c. A Facebook user1 can like the post of Facebook user2 after verifying that user1 is present in the friend list of user2 and the given post which user1 wants to like is an made by user2.
- 2. **Album Creation** - Whenever the createAlbum API call is made from the client, a folder with the name "username-albumId" is made inside the images folder in the facebookServer. This folder contains the image which client sends to the server.
- 3. **Logging** – The responses of all the API calls are stored in the log files, server_output present in facebookServer folder and client_output present in facebookClient folder.
- 4. **Image Compression** - The image is stored after compressing to enable faster processing.
- 5. **Caching** – Caching has been implemented for faster retrieval of information from the server.

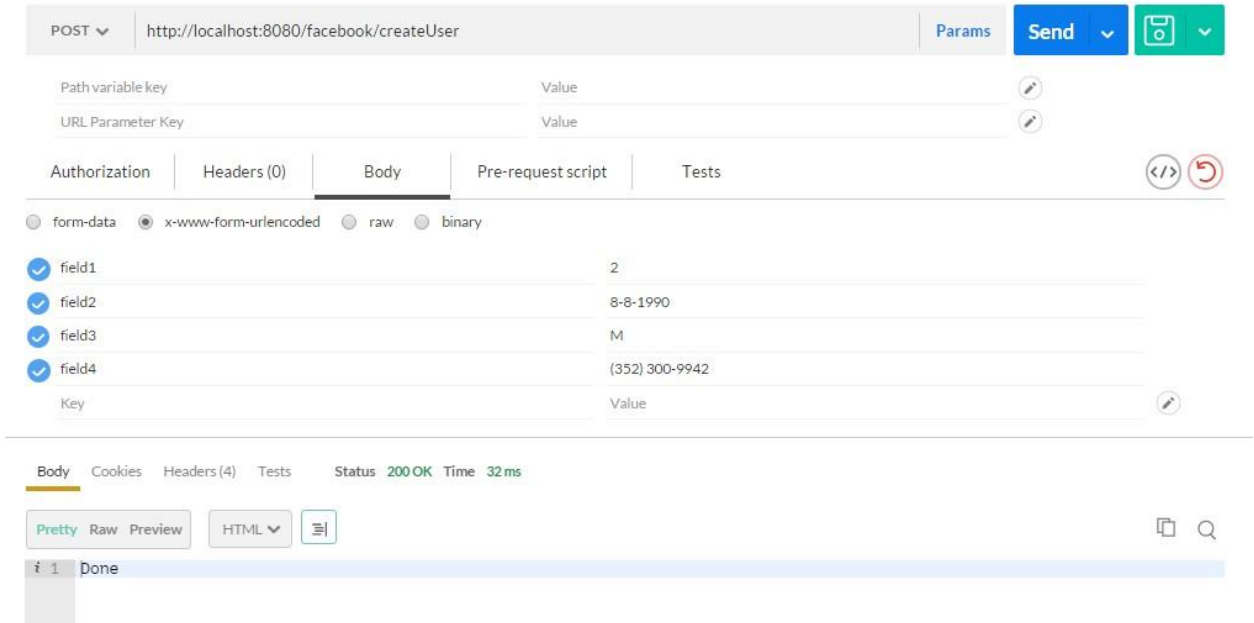
APIs Implemented

Following APIs have been implemented in the project –

- ***To create Facebook User***

```
http://localhost:8080/facebook/createUser",FormData(Seq("field1"->userCount,  
"field2"->dob, "field3"->gender, "field4"->phoneNumber))
```

Postman sample for testing



POST `http://localhost:8080/facebook/createUser` Params Send

Path variable key Value

URL Parameter Key Value

Authorization Headers (0) Body Pre-request script Tests

☐ form-data ☒ x-www-form-urlencoded ☐ raw ☐ binary

field1	2
field2	8-8-1990
field3	M
field4	(352) 300-9942

Key Value

Body Cookies Headers (4) Tests Status 200 OK Time 32 ms

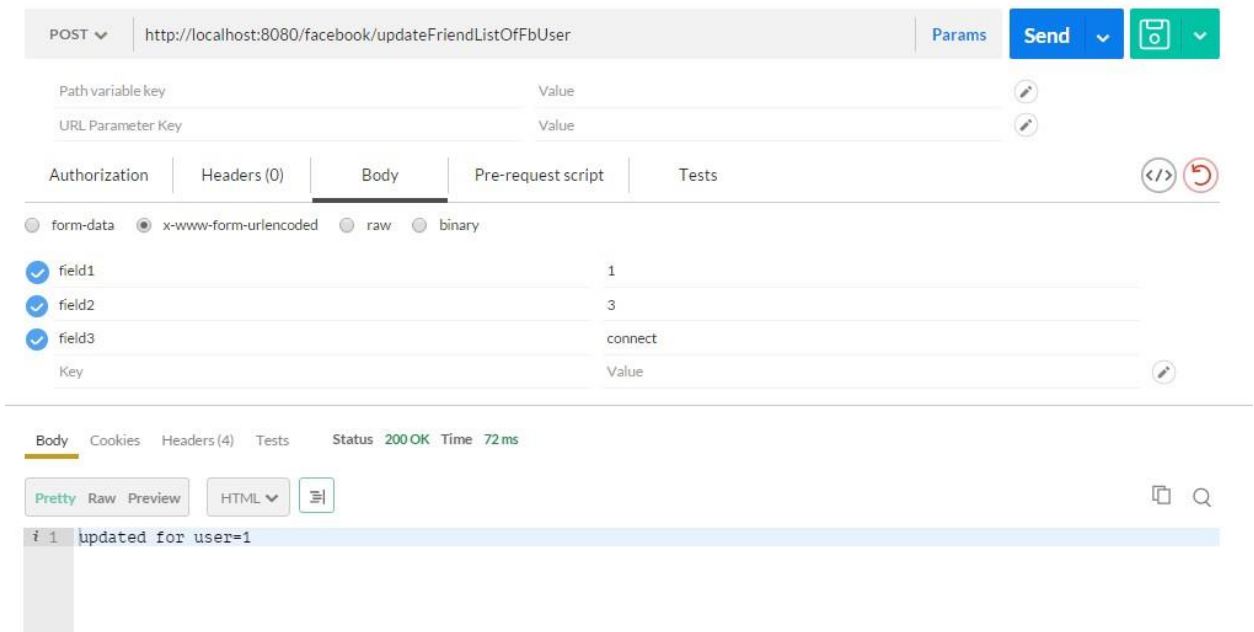
Pretty Raw Preview HTML

1 Done

- ***To update friend list of the user***

`http://localhost:8080/facebook/updateFriendListOfFbUser",FormData(Seq("field1"->userName, "field2"->friendUserName,"field3"->action))`

Postman sample for testing



POST `http://localhost:8080/facebook/updateFriendListOfFbUser` Params Send

Path variable key Value

URL Parameter Key Value

Authorization Headers (0) Body Pre-request script Tests

☐ form-data ☒ x-www-form-urlencoded ☐ raw ☐ binary

field1	1
field2	3
field3	connect

Key Value

Body Cookies Headers (4) Tests Status 200 OK Time 72 ms

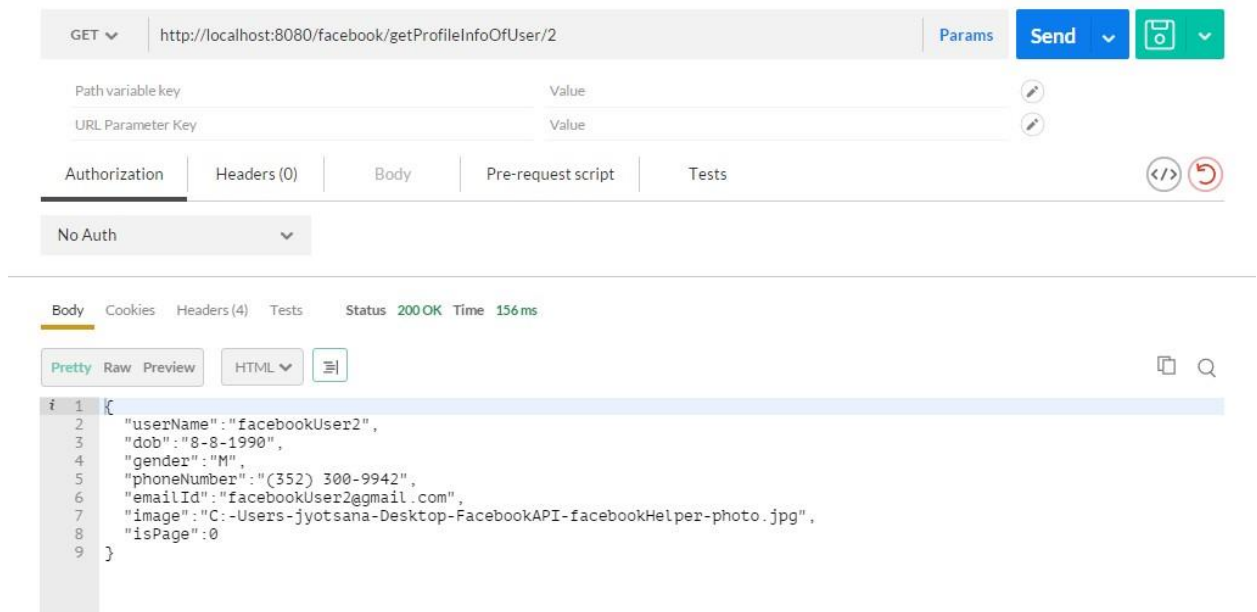
Pretty Raw Preview HTML

1 updated for user=1

- **To get profile information of a particular Facebook user**

[http://localhost:8080/facebook/getProfileInfoOfUser/"+userCount](http://localhost:8080/facebook/getProfileInfoOfUser/)

Postman sample for testing



- **To create Facebook page**

`http://localhost:8080/facebook/createPage",FormData(Seq("field1"->userCount, "field2"->dob, "field3"->gender, "field4"->phoneNumber))`

Postman sample for testing

POST `http://localhost:8080/facebook/createPage` Params Send

Path variable key Value

URL Parameter Key Value

Authorization Headers (0) Body Pre-request script Tests

☐ form-data ☒ x-www-form-urlencoded ☐ raw ☐ binary

✓ field1	3
✓ field2	8-8-1990
✓ field3	M
✓ field4	(352) 300-9942
Key	Value

Body Cookies Headers (4) Tests Status 200 OK Time 36 ms

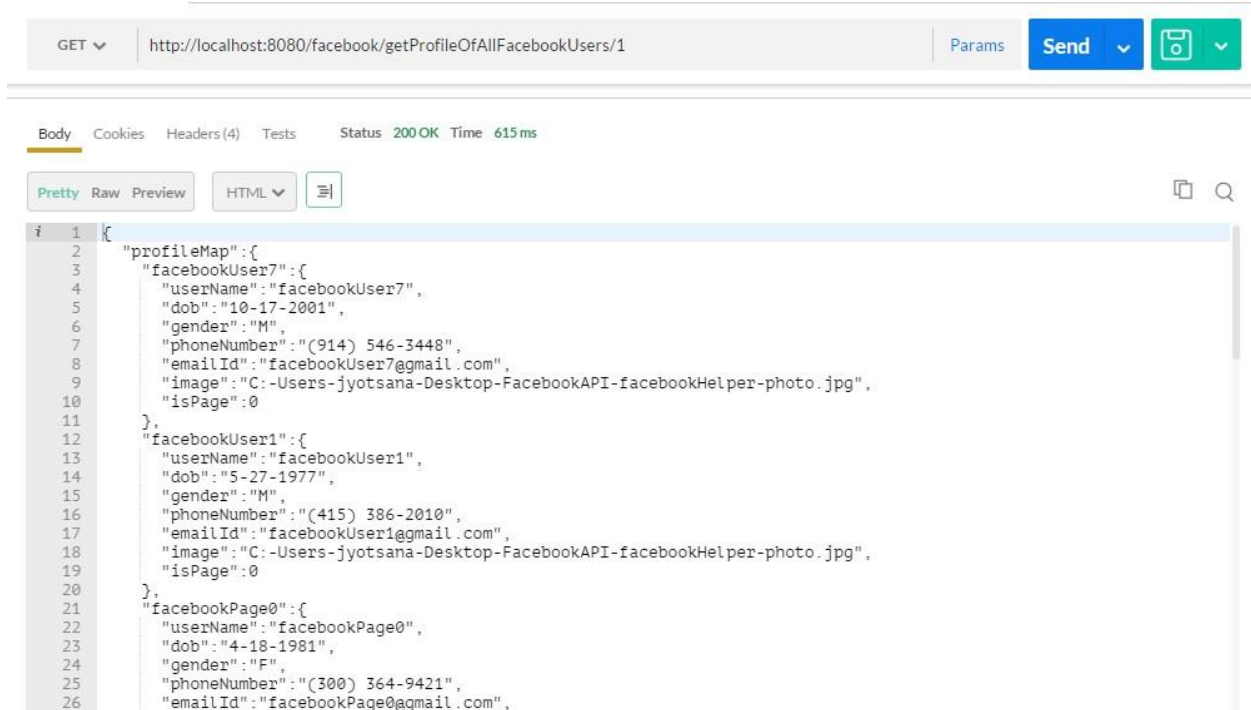
Pretty Raw Preview HTML

i 1 Done

- ***To get profiles of all the Facebook users***

[http://localhost:8080/facebook/getProfileOfAllFacebookUsers/"+userCount](http://localhost:8080/facebook/getProfileOfAllFacebookUsers/)

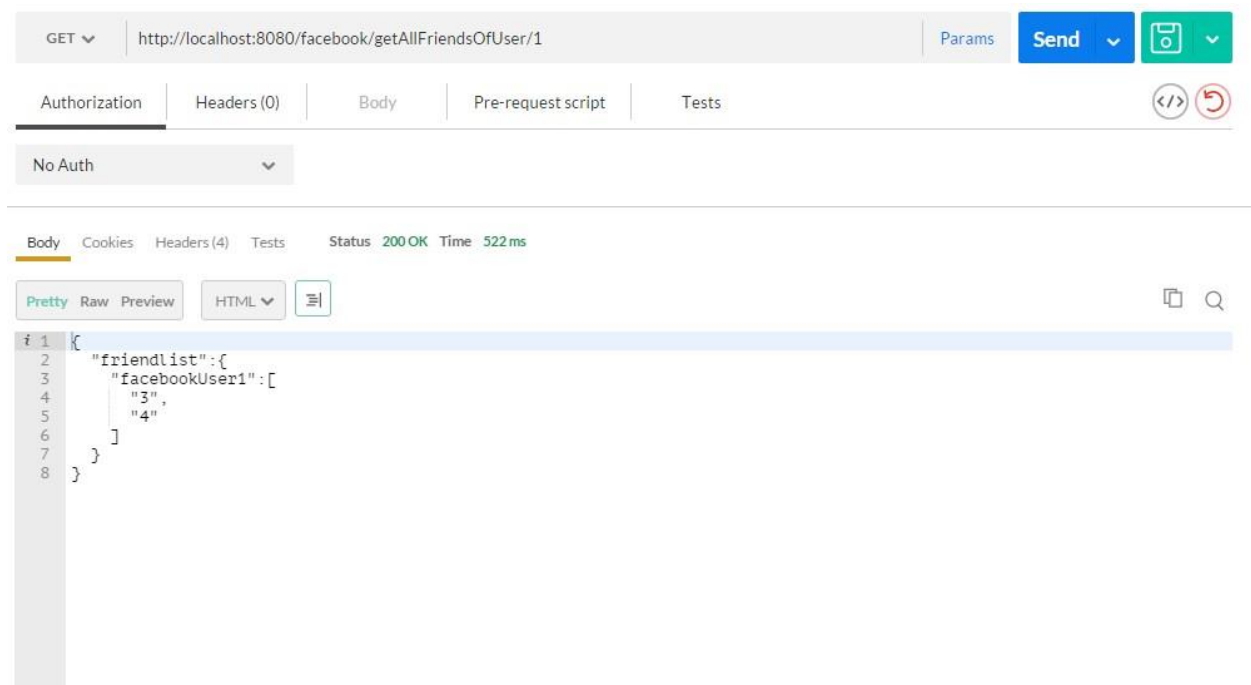
Postman sample for testing



- **To get the friend list of a Facebook user**

[http://localhost:8080/facebook/getAllFriendsOfUser/"+userCount](http://localhost:8080/facebook/getAllFriendsOfUser/)

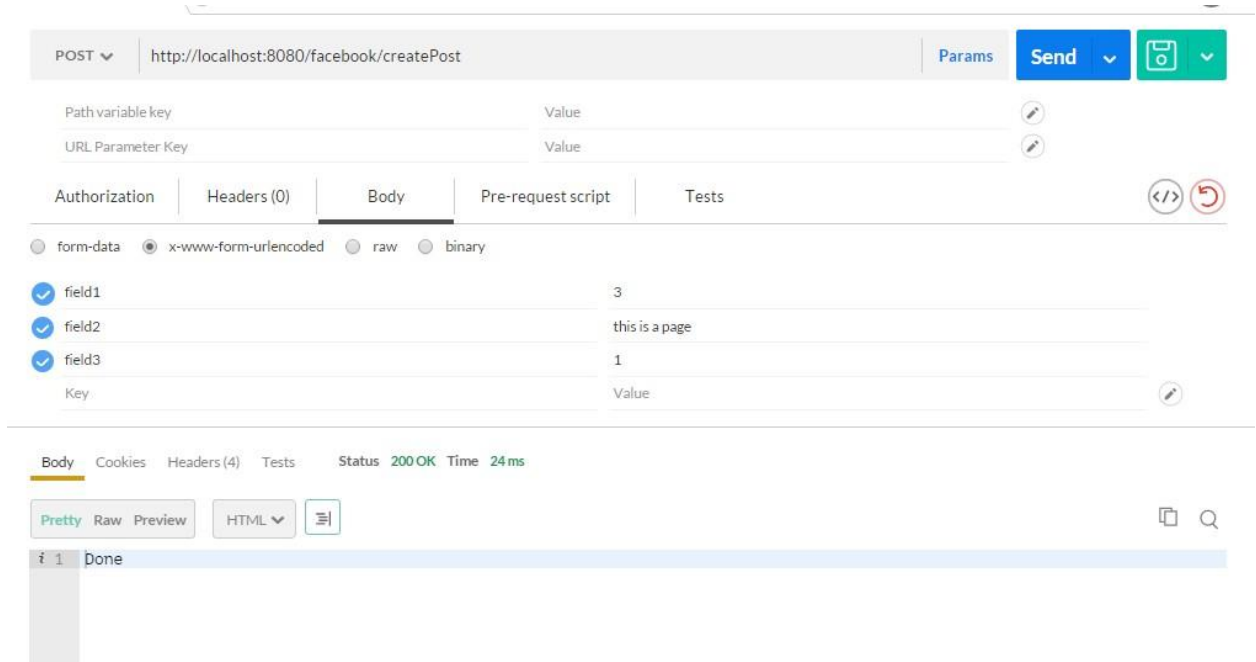
Postman sample for testing



- **To create a post**

`http://localhost:8080/facebook/createPost",FormData(Seq("field1"->userCount,
"field2"->content,"field3"->postId))`

Postman sample for testing



- **To retrieve all the posts of all Facebook users**

[http://localhost:8080/facebook/getPostsOfAllFacebookUsers/"+userCount](http://localhost:8080/facebook/getPostsOfAllFacebookUsers/)

Postman sample for testing



- ***To read the post of a Facebook user***

`http://localhost:8080/facebook/getPostOfUser",FormData(Seq("field1"->authorId,
"field2"->actionUserId))`

Sample Input -

Given authorId = 3 and actionUserId = 1

Sample Output

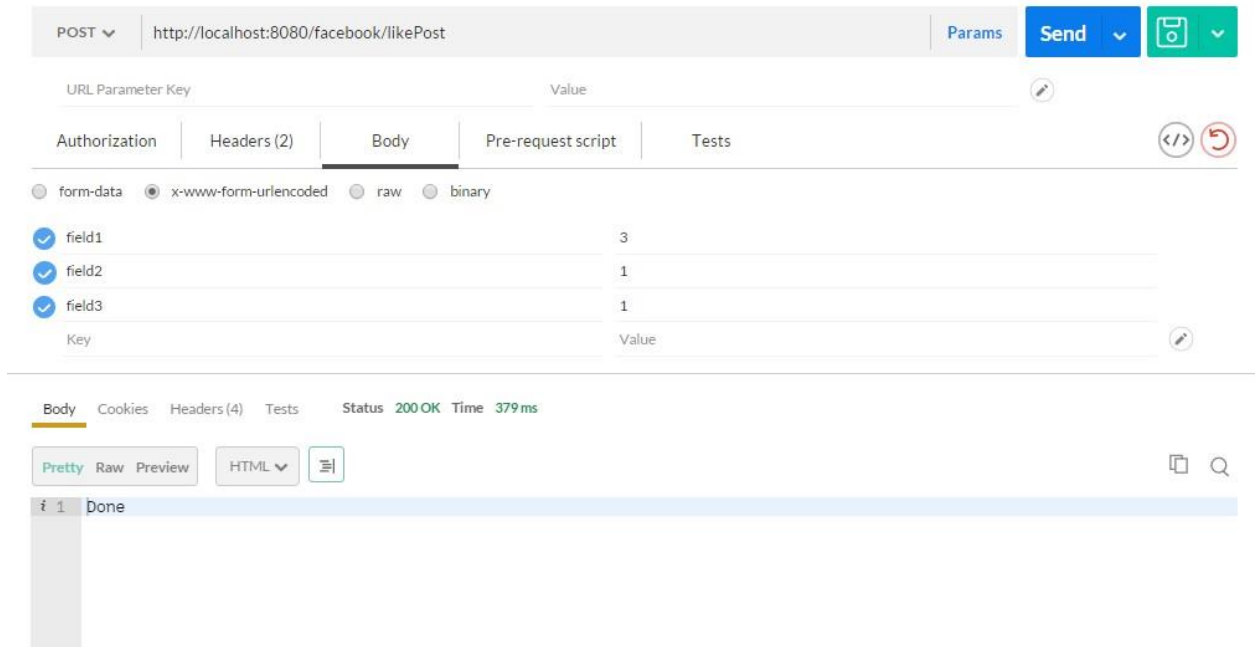
Request completed with status 200 OK and content:

```
{
  "postMap":{
    "2":{
      "author":"facebookUser3",
      "content":"second post of the User",
      "likeCount":0,
      "shareCount":0
    },
    "1":{
      "author":"facebookUser3",
      "content":"First post of the User",
      "likeCount":0,
      "shareCount":0
    }
  }
}
```

- *To like the post of a Facebook user*

`http://localhost:8080/facebook/likePost",FormData(Seq("field1"->authorId, "field2"->postId, "field3"->actionUserId))`

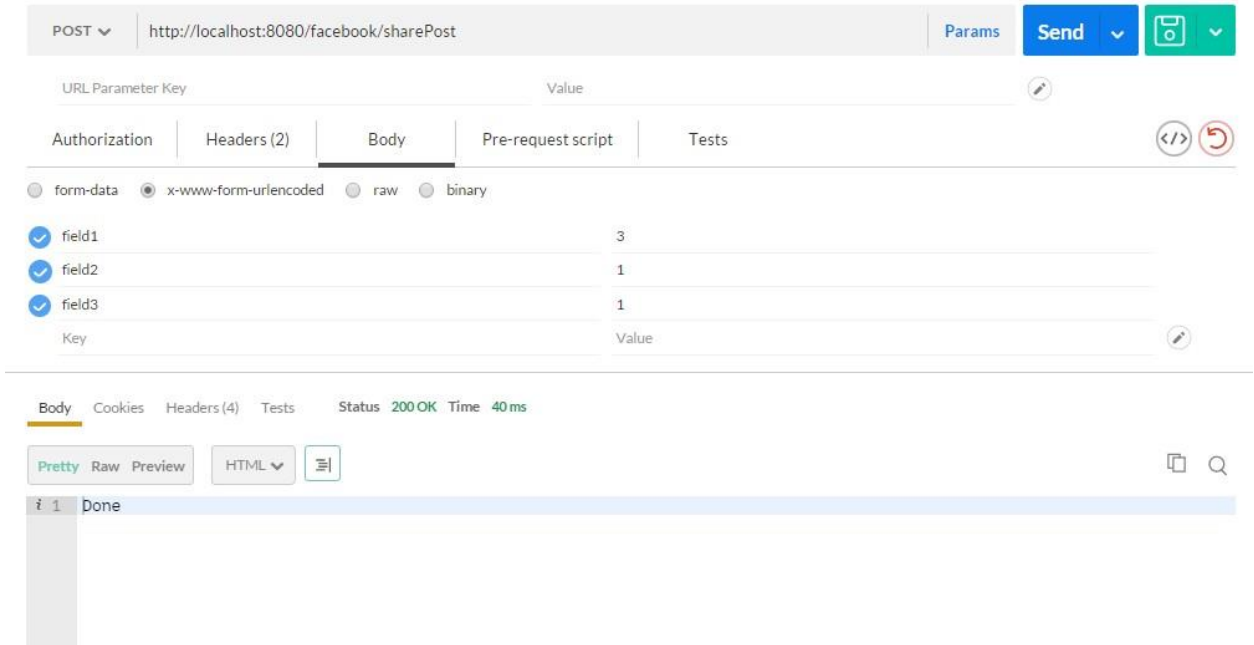
Postman sample for testing



- ***To share the post of a Facebook user***

`http://localhost:8080/facebook/sharePost",FormData(Seq("field1"->authorId, "field2"->postId, "field3"->actionUserId))`

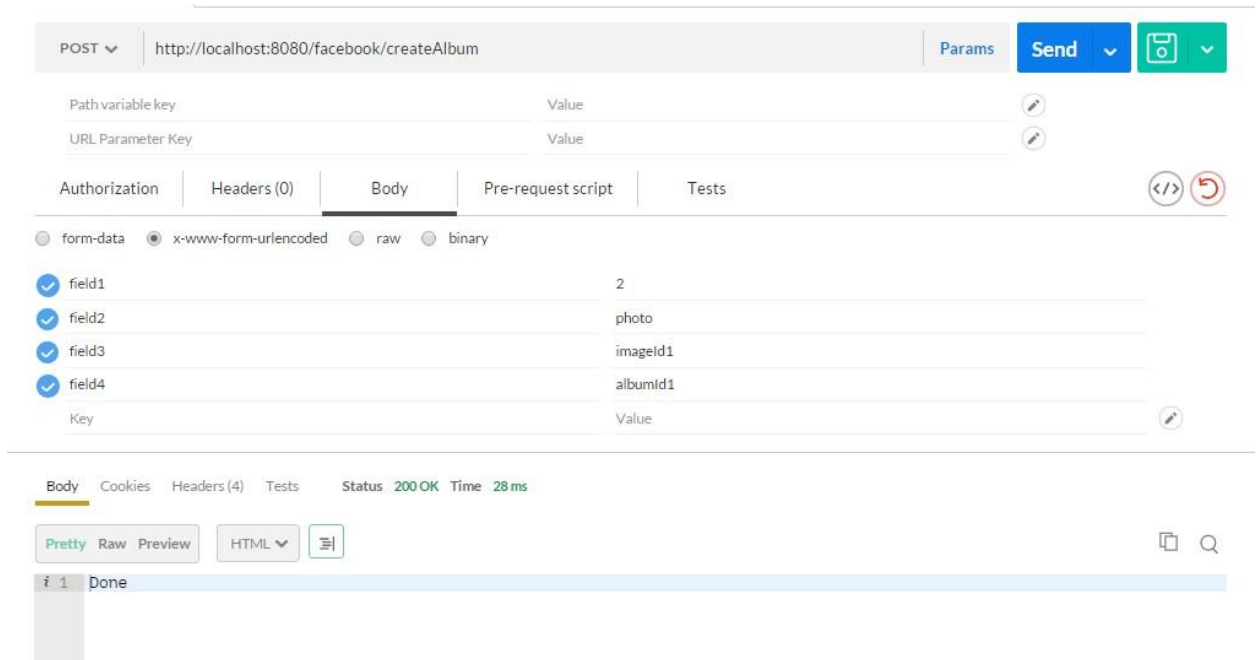
Postman sample for testing



- ***To create an image album***

`http://localhost:8080/facebook/createAlbum",FormData(Seq("field1"->userCount,
"field2"->imageContent,"field3"->imageId,"field4"->albumId))`

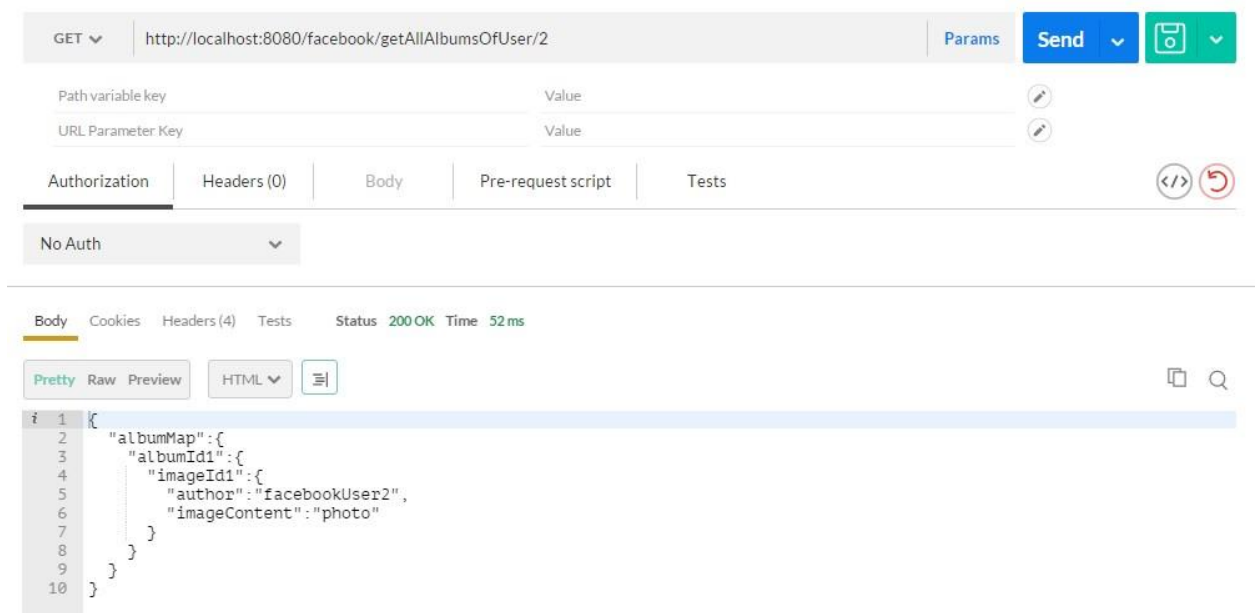
Postman sample for testing



- **To get all albums of a Facebook user**

[http://localhost:8080/facebook/getAllAlbumsOfUser/"+userCount](http://localhost:8080/facebook/getAllAlbumsOfUser/)

Postman sample for testing



Different types of Status Handled

Following are the different types of status returned as output when API calls are executed –

- **HTTP Status Code – 200 OK**

The request has succeeded. The information returned with the response is dependent on the method used in the request.

- **HTTP Status Code – 404 Not Found**

The server has not found anything matching the Request-URI. No indication is given of whether the condition is temporary or permanent.

- **HTTP Status Code – 500 Internal Server Error**

The server encountered an unexpected condition which prevented it from fulfilling the request.

References

1. <http://stackoverflow.com/questions/17008878/sending-a-post-request-in-spray>
2. <https://www.youtube.com/watch?v=XPuOlpWEvmw>
3. <http://engineering.monsanto.com/2015/08/11/simple-spray/>
4. <http://stackoverflow.com/questions/21323842/how-do-i-create-a-post-request-with-form-field-content-with-spray>
5. <http://blog.hubspot.com/blog/tabid/6307/bid/6128/The-Ultimate-List-100-Facebook-Statistics-Infographics.aspx>
6. <https://blog.kissmetrics.com/facebook-statistics/>