

## Spring AU '21 – Spring Rest Services – Afternoon Session

Name: Sheik Abudhahir K

Date: 22/01/2021

Source code uploaded in Git

Question:

Create a rest service with following endpoints - Get, Put, Post, Delete

Create user -> <http://localhost:8080/api/user/signup> - post api with json Body

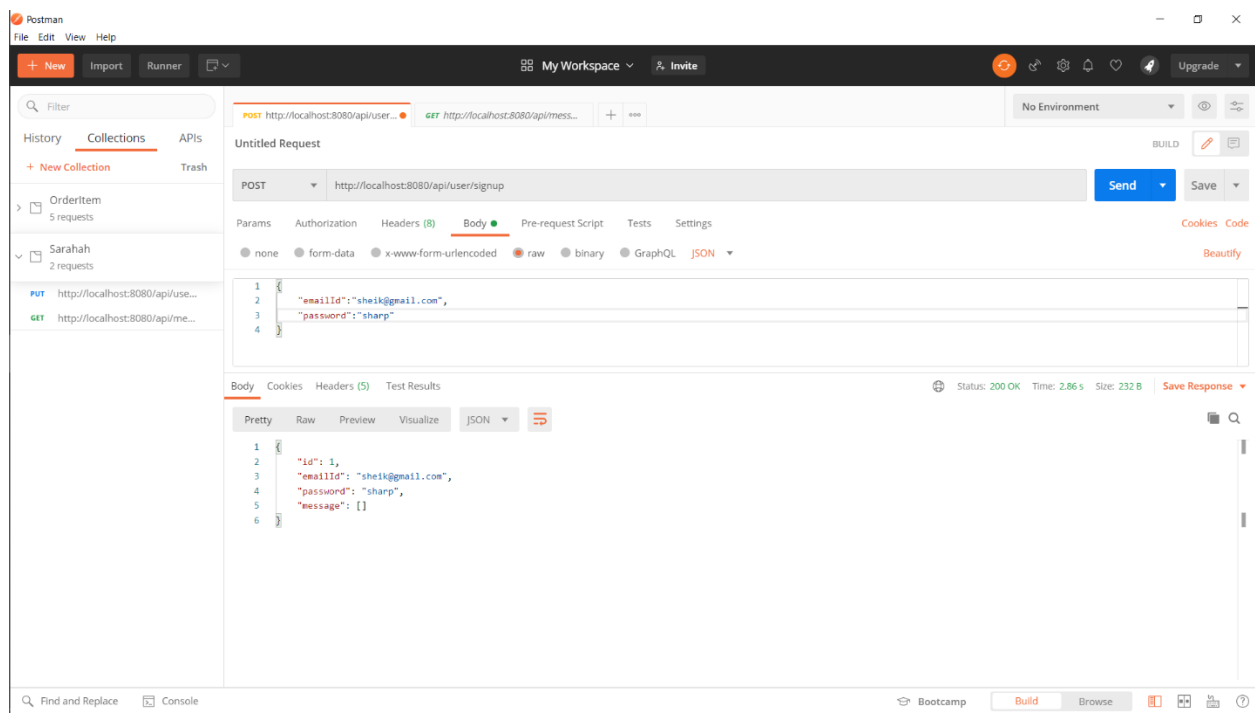
Send message -> <http://localhost:8080/api/message/add/1> - post api with Path variable as userid and json body

View message -> <http://localhost:8080/api/message/get/1> - get api with Path variable as userid

Update user -> <http://localhost:8080/api/user/update/1> - put api with Path variable as userid and json body

Delete message -> <http://localhost:8080/api/message/delete/2> - delete api with Path variable as messageid

Create User api



User created

## Sent message to the user

The screenshot shows the Postman interface with a POST request to `http://localhost:8080/api/message/add/1`. The request body is a JSON object: `{ "message": "Guess who am I?" }`. The response status is 200 OK, and the response body is a JSON object: `{ "id": 2, "message": "Guess who am I?", "createdTime": "2021-01-24T09:02:03.637+00:00" }`.

## Send 2 messages, now view the message of the particular user

The screenshot shows the Postman interface with a GET request to `http://localhost:8080/api/message/get/1`. The response status is 200 OK, and the response body is a JSON array of two objects: `[ { "id": 1, "message": "Hello, Find me", "createdTime": "2021-01-24T09:01:30.171+00:00" }, { "id": 2, "message": "Guess who am I?", "createdTime": "2021-01-24T09:02:03.637+00:00" } ]`.

## Pass path variable as invalid id

The screenshot shows the Postman application interface. The left sidebar displays a collection named 'Sarahah' with 4 requests. The main panel shows a GET request to `http://localhost:8080/api/message/get/1`. The 'Params' tab is active, showing a single query parameter: `Key` with `Value`. The 'Body' tab is also visible. The response status is `404 Not Found` with a message `1 User Not Found`.

## Update the user details

The screenshot shows the Postman application interface. The left sidebar displays a collection named 'Sarahah' with 4 requests. The main panel shows a PUT request to `http://localhost:8080/api/user/update/1`. The 'Body' tab is active, showing a JSON body with the following content:

```
1 {
2   "emailId": "abul23@gmail.com",
3   "password": "sharp"
4 }
```

The response status is `200 OK` with a message `1 User Not Found`. The response body is a JSON object:

```
1 {
2   "id": 1,
3   "emailId": "abul23@gmail.com",
4   "password": "sharp",
5   "message": [
6     {
7       "id": 1,
8       "message": "Hello, Find me",
9       "createdTime": "2021-01-24T09:01:38.171+00:00"
10    },
11    {
12      "id": 2,
13      "message": "Guess who am I??",
14      "createdTime": "2021-01-24T09:02:03.637+00:00"
15    }
16  ]
17 }
```

## Delete the message using messageId

The screenshot shows the Postman application interface. The top bar includes the Postman logo, menu items (File, Edit, View, Help), and workspace controls (My Workspace, Invite). The left sidebar shows a 'Collections' tab with a collection named 'Sarahah' containing 4 requests. The main panel displays an 'Untitled Request' for a DELETE method at the URL 'http://localhost:8080/api/message/delete/2'. The 'Params' tab is active, showing a table with one row: 'Key' and 'Value'. The 'Body' tab is also visible, showing a status of 200 OK, time of 220 ms, and size of 192 B. The response body is displayed in the 'Body' tab, showing the text '1 Message deleted successfully'.

Postman interface showing a DELETE request to delete a message. The request is configured with the URL `http://localhost:8080/api/message/delete/2` and the method `DELETE`. The response status is 200 OK, and the response body is '1 Message deleted successfully'.

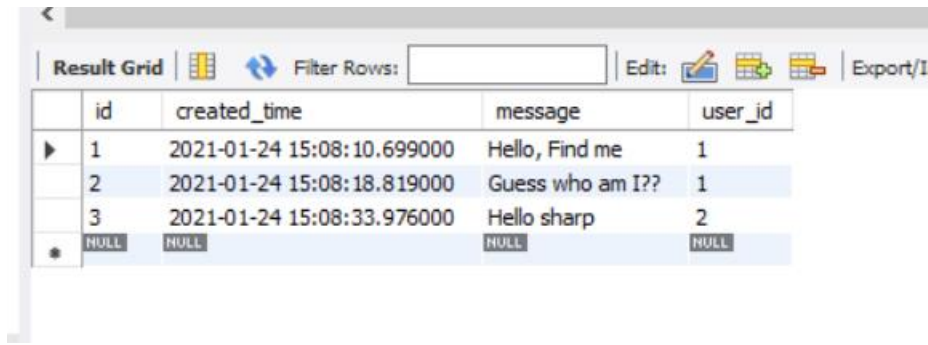
## View the messages after delete

The screenshot shows the Postman application interface. The top bar includes the Postman logo, menu items (File, Edit, View, Help), and workspace controls (My Workspace, Invite). The left sidebar shows a 'Collections' tab with a collection named 'Sarahah' containing 4 requests. The main panel displays an 'Untitled Request' for a GET method at the URL 'http://localhost:8080/api/message/get/1'. The 'Params' tab is active, showing a table with one row: 'Key' and 'Value'. The 'Body' tab is also visible, showing a status of 200 OK, time of 65 ms, and size of 247 B. The response body is displayed in the 'Body' tab, showing a JSON object: 

```
{  "id": 1,  "message": "Hello, Find me",  "createdTime": "2021-01-24T09:01:38.171+00:00"}
```

Postman interface showing a GET request to view messages. The request is configured with the URL `http://localhost:8080/api/message/get/1` and the method `GET`. The response status is 200 OK, and the response body is a JSON object representing a message.

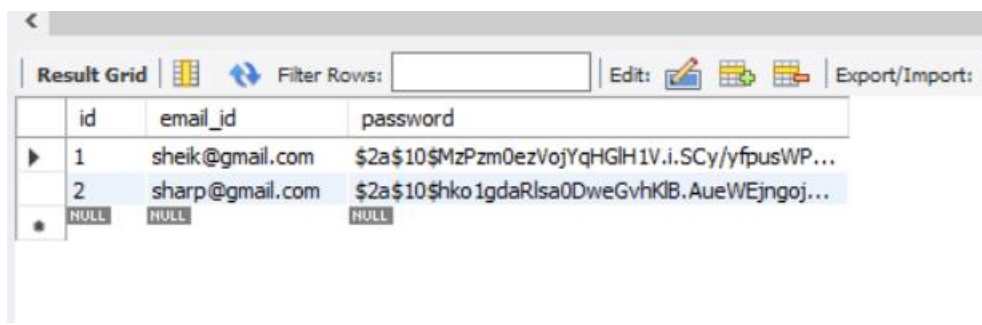
## Using DB to persist data – MySQL - message table data



A screenshot of a database client interface showing a table named 'message'. The table has five columns: 'id', 'created\_time', 'message', and 'user\_id'. There are three rows of data. The first row has id 1, created\_time 2021-01-24 15:08:10.699000, message 'Hello, Find me', and user\_id 1. The second row has id 2, created\_time 2021-01-24 15:08:18.819000, message 'Guess who am I??', and user\_id 1. The third row has id 3, created\_time 2021-01-24 15:08:33.976000, message 'Hello sharp', and user\_id 2. Below the data rows, there is a row with NULL values for all columns.

	id	created_time	message	user_id
▶	1	2021-01-24 15:08:10.699000	Hello, Find me	1
	2	2021-01-24 15:08:18.819000	Guess who am I??	1
	3	2021-01-24 15:08:33.976000	Hello sharp	2
*	NULL	NULL	NULL	NULL

## User table data with encoding password



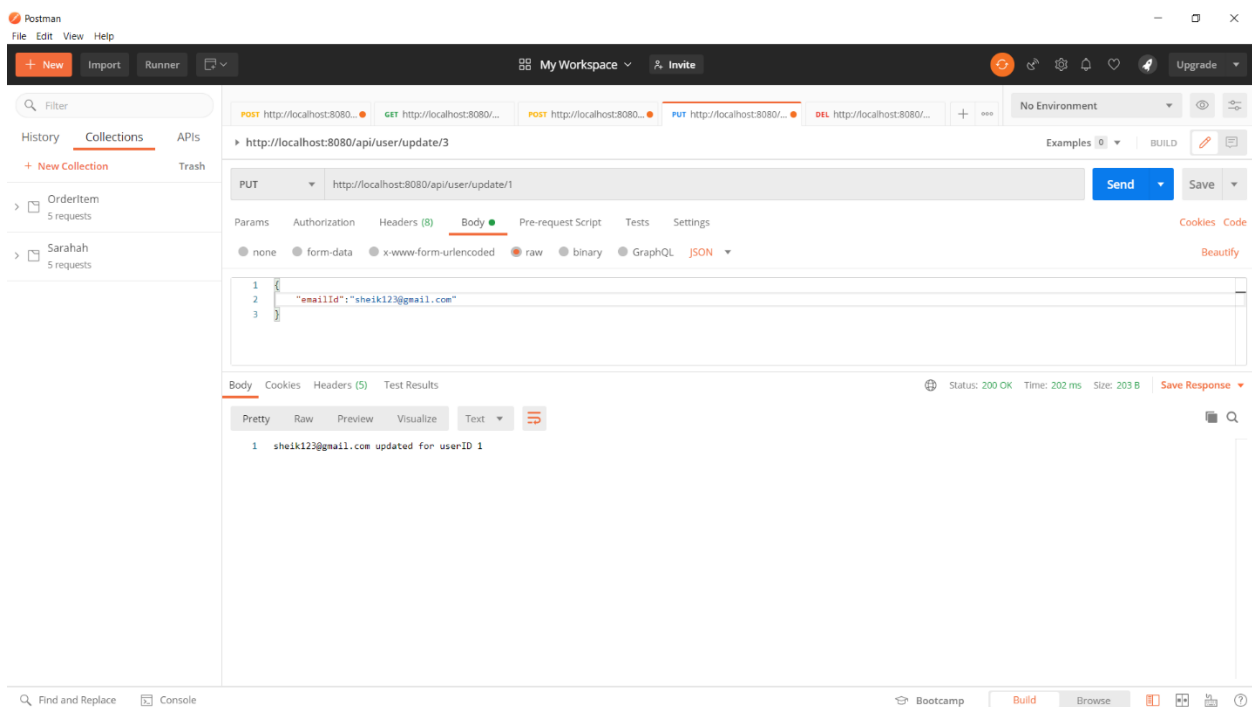
A screenshot of a database client interface showing a table named 'user'. The table has four columns: 'id', 'email\_id', and 'password'. There are two rows of data. The first row has id 1, email\_id 'sheik@gmail.com', and password '\$2a\$10\$MzPzm0ezVojYqHG1H1V.i.SCy/yfpusWP...'. The second row has id 2, email\_id 'sharp@gmail.com', and password '\$2a\$10\$shko1gdaRlsa0DweGvhKIB.AueWEjngoj...'. Below the data rows, there is a row with NULL values for all columns.

	id	email_id	password
▶	1	sheik@gmail.com	\$2a\$10\$MzPzm0ezVojYqHG1H1V.i.SCy/yfpusWP...
	2	sharp@gmail.com	\$2a\$10\$shko1gdaRlsa0DweGvhKIB.AueWEjngoj...
*	NULL	NULL	NULL

Used BcryptEncoder – source code also uploaded

Corrections:

Update emailId only



A screenshot of the Postman application interface. The top bar shows 'Postman' and 'My Workspace'. The left sidebar has 'History', 'Collections', and 'APIs' tabs. The main area shows a PUT request to 'http://localhost:8080/api/user/update/1'. The request body is a JSON object: { "emailId": "sheik123@gmail.com" }. The status is '200 OK', time is '202 ms', and size is '203 B'. The response body is '1 sheik123@gmail.com updated for userID 1'.

Postman

File Edit View Help

+ New Import Runner My Workspace % Invite

POST http://localhost:8080/... GET http://localhost:8080/... POST http://localhost:8080/... PUT http://localhost:8080/... DELETE http://localhost:8080/...

Examples BUILD

PUT http://localhost:8080/api/user/update/1 Send Save

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

1 {  
2 "emailId": "sheik123@gmail.com"  
3 }

Body Cookies Headers (5) Test Results

Status: 200 OK Time: 202 ms Size: 203 B Save Response

Pretty Raw Preview Visualize Text

1 sheik123@gmail.com updated for userID 1

Find and Replace Console

Bootcamp Build Browse

## Update password only

The screenshot shows the Postman interface with a PUT request to `http://localhost:8080/api/user/update/1`. The request body is a JSON object with the password updated to "sharp007". The response is a JSON array containing two user objects, one of which has the updated password.

```
PUT http://localhost:8080/api/user/update/1
```

```
{
  "password": "sharp007"
}
```

```
[
  {
    "id": 1,
    "emailId": "sheik123@gmail.com",
    "password": "sharp007",
    "message": [
      {
        "id": 1,
        "message": "Hello, Find me",
        "createdTime": "2021-01-24T09:38:10.699+00:00"
      }
    ]
  },
  {
    "id": 2,
    "message": "Guess who am I??",
    "createdTime": "2021-01-24T09:38:18.819+00:00"
  }
]
```

## Update both emailId and password

The screenshot shows the Postman interface with a PUT request to `http://localhost:8080/api/user/update/3`. The request body is a JSON object with both emailId and password updated. The response is a JSON array containing two user objects, one of which has the updated email and password.

```
PUT http://localhost:8080/api/user/update/3
```

```
{
  "emailId": "sheik@gmail.com",
  "password": "sheik123"
}
```

```
[
  {
    "id": 1,
    "emailId": "sheik@gmail.com",
    "password": "sheik123",
    "message": [
      {
        "id": 1,
        "message": "Hello, Find me",
        "createdTime": "2021-01-24T09:38:10.699+00:00"
      }
    ]
  },
  {
    "id": 2,
    "message": "Guess who am I??",
    "createdTime": "2021-01-24T09:38:18.819+00:00"
  }
]
```