

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

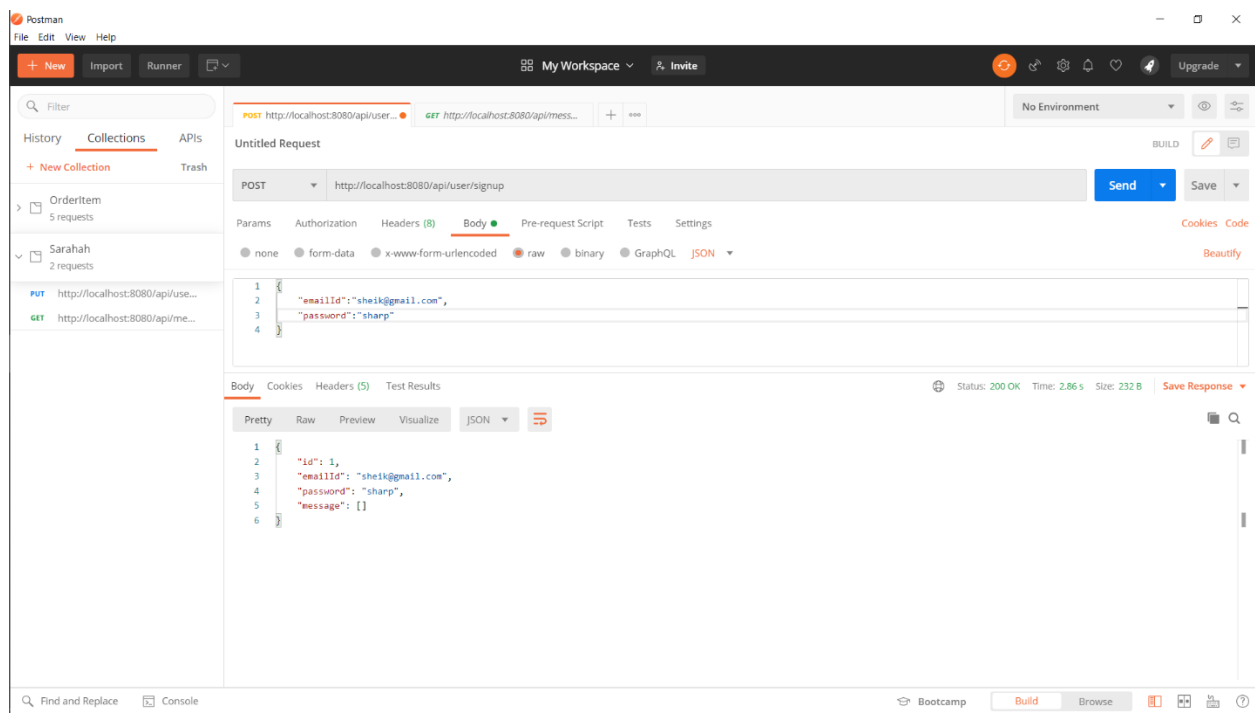
Send message -> <http://localhost:8080/api/message/add/1> - post api with Path variable as userid and json body

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

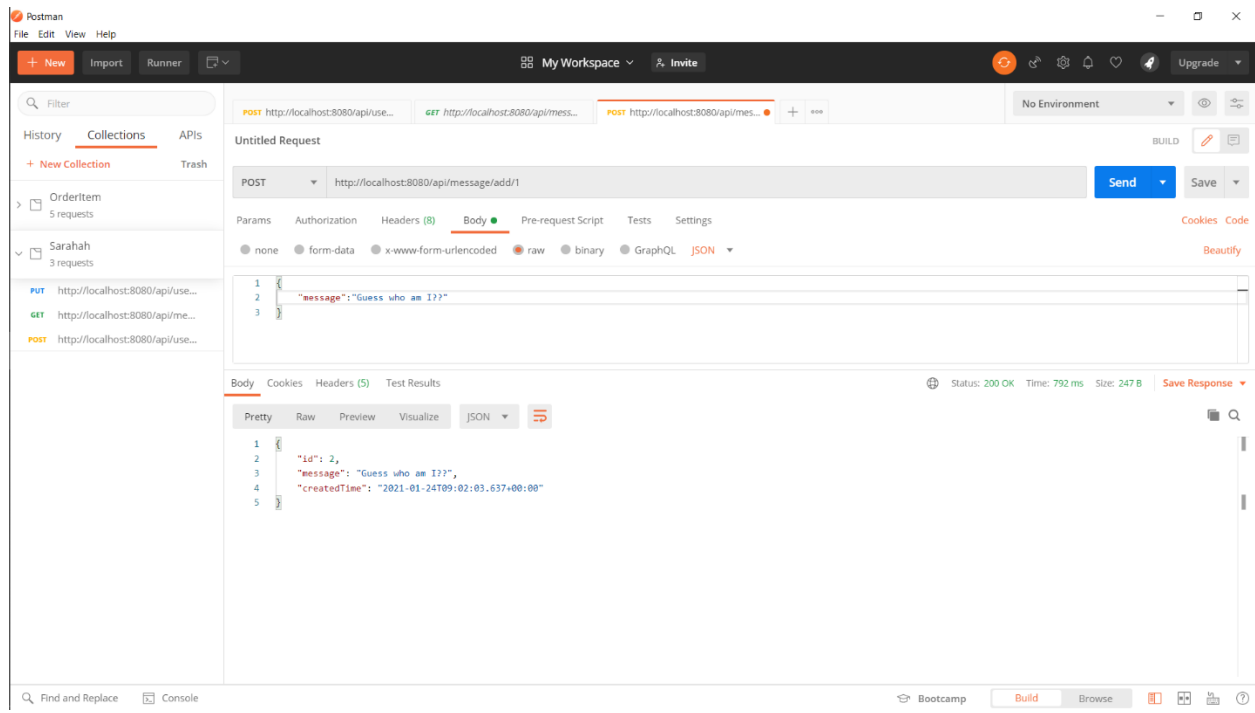
Delete message -> <http://localhost:8080/api/message/delete/2> - delete api with Path variable as messageid

Create User api



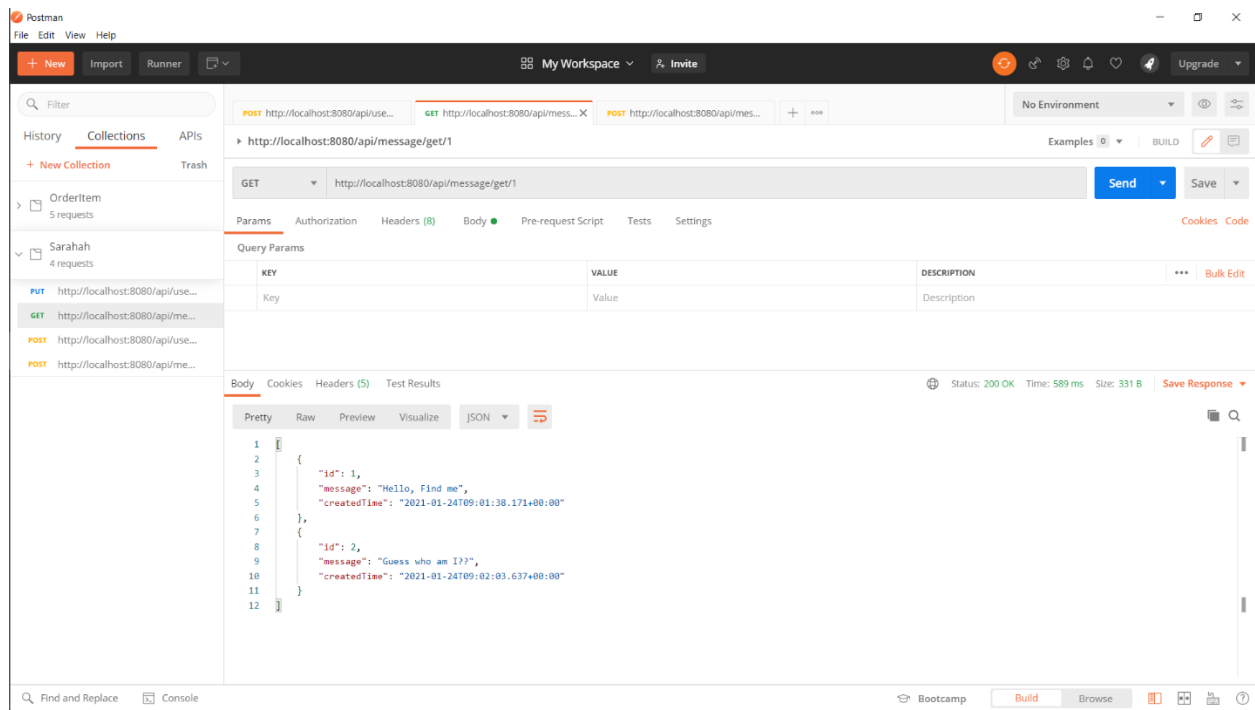
User created

Sent message to the user



Postman interface showing 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 is a JSON object: `{ "id": 2, "message": "Guess who am I?", "createTime": "2021-01-24T09:02:03.637+00:00" }`.

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



Postman interface showing a GET request to `http://localhost:8080/api/message/get/1`. The response is a JSON array containing two message objects:

```
1 {
2   {
3     "id": 1,
4     "message": "Hello, Find me",
5     "createTime": "2021-01-24T09:01:38.171+00:00"
6   },
7   {
8     "id": 2,
9     "message": "Guess who am I?",
10    "createTime": "2021-01-24T09:02:03.637+00:00"
11  }
12 }
```

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 'Query Params' section is empty. The 'Body' tab is selected, showing a status of 404 Not Found. The response body is 'User Not Found'.

| KEY | VALUE | DESCRIPTION |
|-----|-------|-------------|
| Key | Value | Description |

Status: 404 Not Found Time: 15 ms Size: 185 B Save Response

1 User Not Found

Update the user details

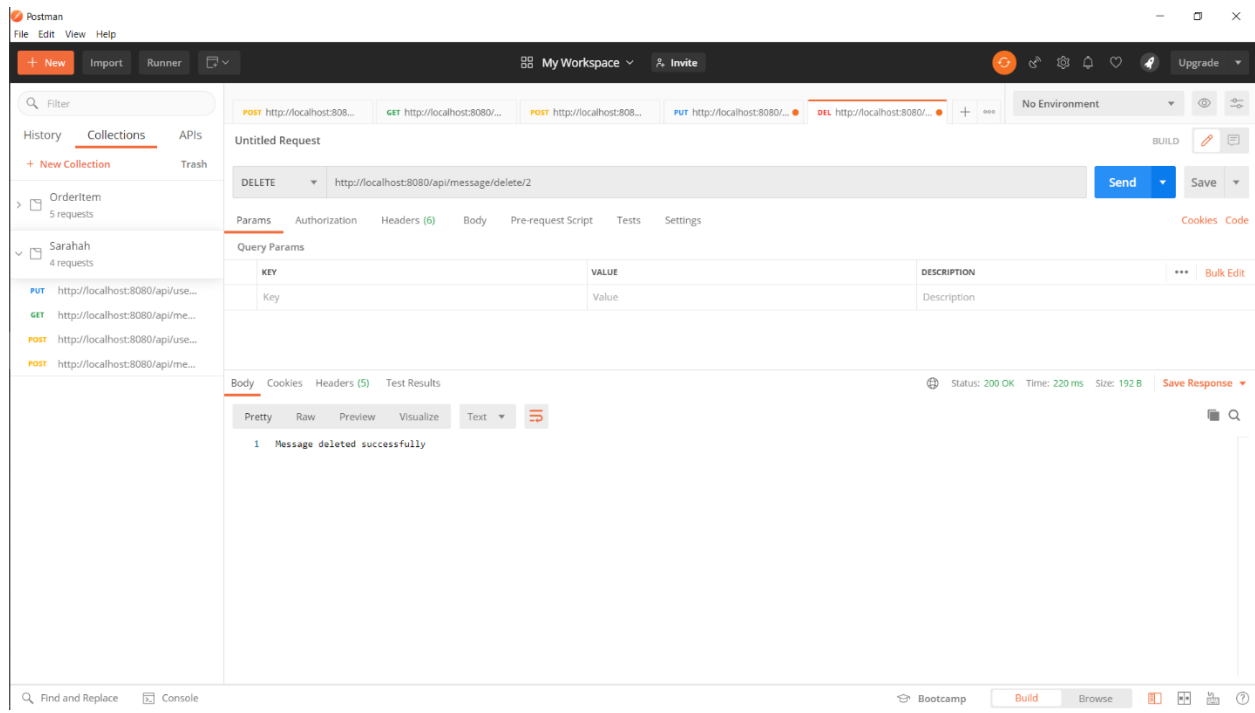
The screenshot shows the Postman application interface. The left sidebar displays a collection named 'Sarahah' with 4 requests. The main panel shows a PUT request to `http://localhost:8080/api/user/update/1`. The 'Body' tab is selected, showing a status of 200 OK. The response body is a JSON object.

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

Status: 200 OK Time: 309 ms Size: 398 B Save Response

```
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 with a DELETE request configured. The URL is `http://localhost:8080/api/message/delete/2`. The response status is 200 OK, and the body contains the text "1 Message deleted successfully".

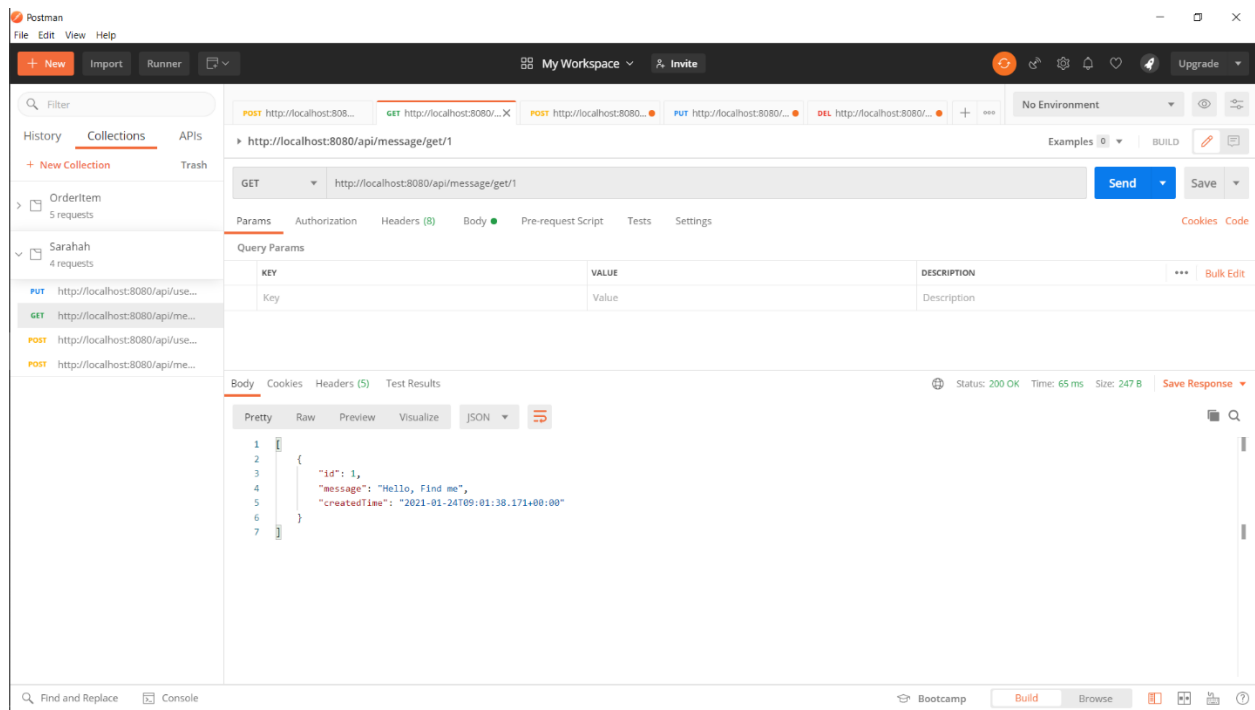
Request Details:

- Method: DELETE
- URL: `http://localhost:8080/api/message/delete/2`
- Headers: 6 headers (not shown)
- Body: Empty

Response Details:

- Status: 200 OK
- Time: 220 ms
- Size: 192 B
- Body: `1 Message deleted successfully`

View the messages after delete



The screenshot shows the Postman application with a GET request configured. The URL is `http://localhost:8080/api/message/get/1`. The response status is 200 OK, and the body contains a JSON object representing a message.

Request Details:

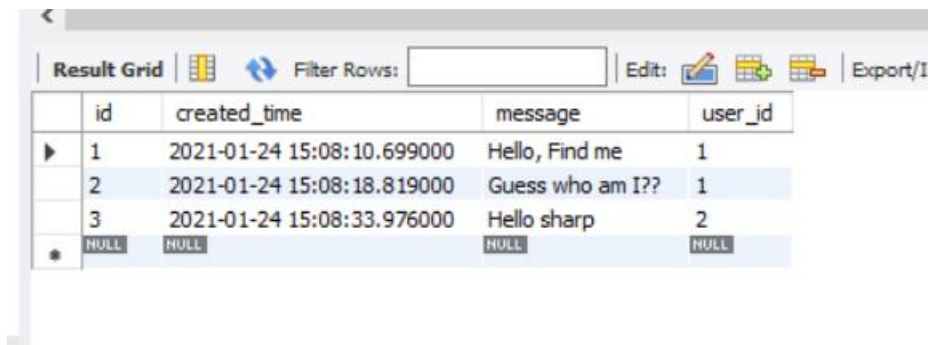
- Method: GET
- URL: `http://localhost:8080/api/message/get/1`
- Headers: 8 headers (not shown)
- Body: Empty

Response Details:

- Status: 200 OK
- Time: 65 ms
- Size: 247 B
- Body:

```
1 {
2   {
3     "id": 1,
4     "message": "Hello, Find me",
5     "createdTime": "2021-01-24T09:01:38.171+00:00"
6   }
7 }
```

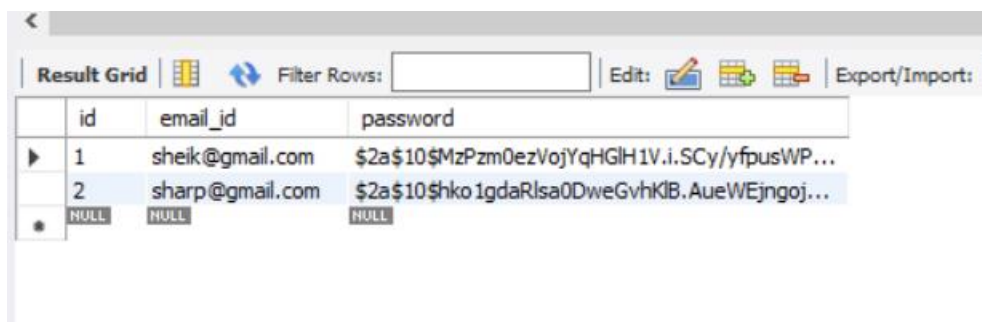
Using DB to persist data – MySQL - message table data



A screenshot of a database management tool's 'Result Grid' window. The window has a toolbar with icons for 'Filter Rows', 'Edit', and 'Export/Import'. Below the toolbar is a table with five columns: 'id', 'created_time', 'message', and 'user_id'. The table contains three rows of data and a row with all 'NULL' values. 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.

| | 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 management tool's 'Result Grid' window. The window has a toolbar with icons for 'Filter Rows', 'Edit', and 'Export/Import'. Below the toolbar is a table with three columns: 'id', 'email_id', and 'password'. The table contains two rows of data and a row with all 'NULL' values. 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.AueWEjngo...'. The password field is truncated with an ellipsis.

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

Used BcryptEncoder – source code also uploaded