

WEB FORMS- REST API WITH FLASK, DOCKER, PyCharm, and MySQL

Name: Jasneek Singh Chugh

Link to Git-hub repository: https://github.com/jasneekchugh/Web_Forms_REST-API_using_Flask

A. YOUR project showing postman listing all records.

The screenshot shows a Postman interface with a collection named 'mlbPlayersData' and a sub-collection 'mlbPlayers-List'. A GET request is configured to 'http://localhost:5000/api/v1/Names'. The response is a JSON array of two player objects. The status is 200 OK, and the response time is 40 ms.

```
1 {
2   {
3     "Name": "Adam Donachie",
4     "Team": " \BAL\"",
5     "Position": " \Catcher\"",
6     "Height_inches": 74,
7     "Weight_lbs": 180,
8     "Age": 22.99
9   },
10  {
11    "Name": "Paul Bako",
12    "Team": " \BAL\"",
13    "Position": " \Catcher\"",
14    "Height_inches": 74,
15    "Weight_lbs": 215,
16    "Age": 34.69
17  },
18 }
```

B. YOUR project showing postman showing one record with a get request

Showing record with name “Brian Roberts” using GET request

The screenshot shows a Postman interface with a collection named 'mlbPlayersData' and a sub-collection 'mlbPlayers-Show'. A GET request is configured to 'http://localhost:5000/api/v1/Names/Brian Roberts'. The response is a JSON object for Brian Roberts. The status is 200 OK, and the response time is 34 ms.

```
1 {
2   {
3     "Name": "Brian Roberts",
4     "Team": " \BAL\"",
5     "Position": " \Second Baseman\"",
6     "Height_inches": 69,
7     "Weight_lbs": 176,
8     "Age": 29.39
9   }
10 }
```

C. YOUR project showing postman creating a record by showing the post request and the results in the record listing.

The screenshot shows the Postman interface with a collection named 'mlbPlayersData' and a sub-collection 'mlbPlayers-List'. A POST request is configured with the URL 'http://localhost:5000/api/v1/Names/'. The request body is a JSON object representing a new player record. The response is a 201 Created status with a 41ms response time and 150B of data.

```
POST http://localhost:5000/api/v1/Names/

{
  "Name": "Jasneek Singh",
  "Team": "BAL",
  "Position": "First Baseman",
  "Height_inches": 78,
  "Weight_lbs": 180,
  "Age": 25
}
```

Body: 201 CREATED 41 ms 150 B

Result showing a new record with name “Jasneek Singh” added

The screenshot shows the Postman interface with the same collection and sub-collection. A GET request is configured with the URL 'http://localhost:5000/api/v1/Names'. The response is a 200 OK status with a 44ms response time and 14.55 KB of data. The response body shows a JSON array of player records, including the newly added 'Jasneek Singh'.

```
GET http://localhost:5000/api/v1/Names

This request does not have a body
```

Body: 200 OK 44 ms 14.55 KB

```
[
  {
    "Name": "Alex Cora",
    "Team": "BOS",
    "Position": "Shortstop",
    "Height_inches": 72,
    "Weight_lbs": 180,
    "Age": 31.37
  },
  {
    "Name": "Jasneek Singh",
    "Team": "BAL",
    "Position": "First Baseman",
    "Height_inches": 78,
    "Weight_lbs": 180,
    "Age": 25.00
  }
]
```

D. YOUR project showing postman edit a record by showing the put request and the results in the record listing.

Editing the Age and Weight of “Brian Roberts” using PUT request

The screenshot shows a Postman interface for a PUT request. The left sidebar lists a collection named 'mlbPlayersData' with three items: 'mlbPlayers-List', 'mlbPlayers-Show', and 'mlbPlayers-Edit'. The 'mlbPlayers-Edit' item is selected. The main panel shows the request details for 'http://localhost:5000/api/v1/Names/Brian Roberts'. The request method is 'PUT'. The request body is in JSON format, containing the following data:

```
{  "Name": "Brian Roberts",  "Team": "BAL",  "Position": "Second Baseman",  "Height_inches": 69,  "Weight_lbs": 178,  "Age": 35}
```

 The response status is '200 OK' with a response time of '43 ms' and a size of '145 B'. The response body is in JSON format, containing the same data as the request body.

Result of Edit

The screenshot shows a Postman interface for a GET request. The left sidebar lists the same collection 'mlbPlayersData' with the same three items. The 'mlbPlayers-Edit' item is selected. The main panel shows the request details for 'http://localhost:5000/api/v1/Names/Brian Roberts'. The request method is 'GET'. The response status is '200 OK' with a response time of '153 ms' and a size of '283 B'. The response body is in JSON format, containing the following data:

```
{  "Name": "Brian Roberts",  "Team": "BAL",  "Position": "Second Baseman",  "Height_inches": 69,  "Weight_lbs": 178,  "Age": 35.00}
```

 The 'Weight_lbs' and 'Age' fields are highlighted with a red box, indicating the successful update.

E. YOUR project showing postman delete a record by showing the delete request and the results in the record listing

The screenshot shows the Postman interface with a collection named 'mlbPlayersData'. The selected item is 'mlbPlayers-Delete'. The request method is 'DELETE' and the URL is 'http://localhost:5000/api/v1/Names/Brian Roberts'. The 'Body' tab is active, showing a JSON payload:

```
{  "Name": "Brian Roberts",  "Team": "BAL",  "Position": "Second Baseman",  "Height_inches": 69,  "Weight_lbs": 178,  "Age": 35}
```

. The status bar at the bottom indicates a 200 OK response with 35 ms latency and 145 B of data.

Result- showing no record since it's deleted

The screenshot shows the Postman interface with the same collection 'mlbPlayersData'. The selected item is 'mlbPlayers-Delete'. The request method is 'GET' and the URL is 'http://localhost:5000/api/v1/Names/Brian Roberts'. The 'Body' tab is active, showing an empty JSON object:

```
{}
```

. The status bar at the bottom indicates a 200 OK response with 85 ms latency and 147 B of data.