

Add a role- ADMIN

The screenshot shows two applications side-by-side. On the left is the H2 Console, and on the right is Postman.

H2 Console: The SQL statement `SELECT * FROM ROLES;` is entered. The result set shows one row with `ROLE_ID` 1 and `NAME` ADMIN.

Postman: A POST request is made to `http://localhost:8080/api/role` with a JSON body: `{ "name": "ADMIN" }`. The response is a JSON object: `{ "id": 1, "name": "ADMIN" }`.

Add an ADMIN user

The screenshot shows two applications side-by-side. On the left is the H2 Console, and on the right is Postman.

H2 Console: The SQL statement `SELECT * FROM USERS;` is entered. The result set shows one row with `USER_ID` 2, `PASSWORD` `$2a$10$Cdk.3UaLLInMQqUMXPsv2.LuPaf91QZ3t2a/txhSqp87zMeRqRXq`, and `USERNAME` `adminUser1`.

Postman: A POST request is made to `http://localhost:8080/api/user` with a JSON body: `{ "id": 1, "username": "adminUser1", "password": "pass123", "roles": [{ "id": 1, "name": "ADMIN" }] }`. The response is a JSON object: `{ "id": 2, "username": "adminUser1", "password": "$2a$10$Cdk.3UaLLInMQqUMXPsv2.LuPaf91QZ3t2a/txhSqp87zMeRqRXq", "roles": [{ "id": 1, "name": "ADMIN" }] }`.

Join Table Auto updation

The screenshot shows the H2 Console interface. The left sidebar displays the database schema: jdbc:h2:mem:employee_system, with tables EMPLOYEE, ROLES, USERS, USERS_ROLES, and INFORMATION_SCHEMA. The main area shows the SQL statement: `SELECT * FROM USERS_ROLES`. The result is a single row with columns USER_ID and ROLE_ID, showing values 2 and 1 respectively. The status bar indicates (1 row, 1 ms).

USER_ID	ROLE_ID
2	1

Add an Employee

The screenshot shows two applications side-by-side. On the left is the H2 Console, and on the right is Postman.

H2 Console: The SQL statement `SELECT * FROM EMPLOYEE` is entered. The result shows three rows of employee data:

ID	EMAIL	FIRST_NAME	LAST_NAME
1	sudha@infosys.com	Sudha	Murthy
2	sudha@infosys.com	Narayan	Murthy
3	satyam@microsoft.com	Satyam	Nadella

Postman: A POST request is sent to `http://localhost:8080/api/employees`. The body is a JSON object:

```
{
  "firstName": "Satyam",
  "lastName": "Nadella",
  "email": "satyam@microsoft.com"
}
```

The response body is also shown as a JSON object:

```
{
  "id": 3,
  "firstName": "Satyam",
  "lastName": "Nadella",
  "email": "satyam@microsoft.com"
}
```

Get employee by id-2

The screenshot shows a web application interface with a REST client on the right and a database query result on the left. The REST client is configured with the URL `http://localhost:8080/api/employees/2` and the method `GET`. The response body is displayed in JSON format:

```
1 {
2   "id": 2,
3   "firstName": "Narayan",
4   "lastName": "Murthy",
5   "email": "sudha@infosys.com"
6 }
```

The database query result on the left shows the SQL statement `SELECT * FROM EMPLOYEE;` and the resulting table:

ID	EMAIL	FIRST_NAME	LAST_NAME
1	sudha@infosys.com	Sudha	Murthy
2	sudha@infosys.com	Narayan	Murthy
3	satyam@microsoft.com	Satyam	Nadella

(3 rows, 0 ms)

Edit Employee Record with id 2(edited email)

The screenshot shows a web application interface with a REST client on the right and a database query result on the left. The REST client is configured with the URL `http://localhost:8080/api/employees` and the method `PUT`. The request body is displayed in JSON format:

```
1 {
2   "id": 2,
3   "firstName": "Narayan",
4   "lastName": "Murthy",
5   "email": "narayan@infosys.com"
6 }
```

The database query result on the left shows the SQL statement `SELECT * FROM EMPLOYEE;` and the resulting table:

ID	EMAIL	FIRST_NAME	LAST_NAME
1	sudha@infosys.com	Sudha	Murthy
2	narayan@infosys.com	Narayan	Murthy
3	satyam@microsoft.com	Satyam	Nadella

(3 rows, 0 ms)

Delete an employee with id-1

The screenshot shows a web application interface with a browser window and a sidebar. The browser window displays a DELETE request to `http://localhost:8080/api/employees/1`. The sidebar shows a SQL query `SELECT * FROM EMPLOYEE` and its result, which is a table with 2 rows and 4 columns: ID, EMAIL, FIRST_NAME, and LAST_NAME. The first row is for employee ID 3, Satyam Nadella, and the second row is for employee ID 4, Narayan Murthy.

ID	EMAIL	FIRST_NAME	LAST_NAME
3	satyam@microsoft.com	Satyam	Nadella
4	narayan@infosys.com	Narayan	Murthy

Search by firstName as satyam.--2 results

The screenshot shows a web application interface with a browser window and a sidebar. The browser window displays a GET request to `http://localhost:8080/api/employees/search/satyam`. The sidebar shows the response body, which is a JSON array containing 2 results. The first result is for employee ID 3, Satyam Nadella, and the second result is for employee ID 5, Satyam Sundaram.

```
[{"id": 3, "firstName": "Satyam", "lastName": "Nadella", "email": "satyam@microsoft.com"}, {"id": 5, "firstName": "Satyam", "lastName": "Sundaram", "email": "sundar@microsoft.com"}]
```

Sort in Ascending order

http://localhost:8080/ + ...

GET http://localhost:8080/api/employees/sort Params Send

Body Cookies Headers (5) Test Results

Pretty Raw Preview JSON

```
1 [
2   {
3     "id": 6,
4     "firstName": "Gargeyi",
5     "lastName": "Sharma",
6     "email": "gargeyi@microsoft.com"
7   },
8   {
9     "id": 4,
10    "firstName": "Narayan",
11    "lastName": "Murthy",
12    "email": "narayan@infosys.com"
13  },
14  {
15    "id": 3,
16    "firstName": "Satyam",
17    "lastName": "Nadella",
18    "email": "satyam@microsoft.com"
19  },
20  {
21    "id": 5,
22    "firstName": "Satyam",
23    "lastName": "Sundaram",
24    "email": "sundar@microsoft.com"
25  },
26  {
27    "id": 7,
28    "firstName": "Tapasya",
29    "lastName": "Charan",
30    "email": "tapasya@gmail.com"
31  }
32 ]
```

Sort By Descending order

http://localhost:8080/ GET http://localhost:8080/api/employees/desc Params

Body Cookies Headers (5) Test Results

Pretty Raw Preview JSON

```
1 [
2   {
3     "id": 7,
4     "firstName": "Tapasya",
5     "lastName": "Charan",
6     "email": "tapasya@gmail.com"
7   },
8   {
9     "id": 3,
10    "firstName": "Satyam",
11    "lastName": "Nadella",
12    "email": "satyam@microsoft.com"
13  },
14  {
15    "id": 5,
16    "firstName": "Satyam",
17    "lastName": "Sundaram",
18    "email": "sundar@microsoft.com"
19  },
20  {
21    "id": 4,
22    "firstName": "Narayan",
23    "lastName": "Murthy",
24    "email": "narayan@infosys.com"
25  },
26  {
27    "id": 6,
28    "firstName": "Gargeyi",
29    "lastName": "Sharma",
30    "email": "gargeyi@microsoft.com"
31  }
32 ]
```

List All Employees

GET http://localhost:8080/api/employees

Body Cookies Headers (5) Test Results

Pretty Raw Preview JSON

```
1 [
2   {
3     "id": 1,
4     "firstName": "Tapasya",
5     "lastName": "Charan",
6     "email": "tapasya@microsoft.com"
7   },
8   {
9     "id": 2,
10    "firstName": "Satyam",
11    "lastName": "Nadella",
12    "email": "satyam@microsoft.com"
13  },
14  {
15    "id": 3,
16    "firstName": "Satyam",
17    "lastName": "Sundaram",
18    "email": "sundar@microsoft.com"
19  },
20  {
21    "id": 4,
22    "firstName": "Narayan",
23    "lastName": "Murthy",
24    "email": "narayan@infosys.com"
25  },
26  {
27    "id": 5,
28    "firstName": "Gargeyi",
29    "lastName": "Sharma",
30    "email": "gargeyi@gmail.com"
31  }
32 ]
```

