

Employee Management – Spring Rest Project

Postman screenshots for Admin login:

1. Login page

The screenshot shows a POST request to `http://localhost:8080/api/role`. The Authorization tab is selected, showing 'Basic Auth' selected. The 'Username' field contains `employee1` and the 'Password' field contains `user@123`. The 'Show Password' checkbox is checked. The response status is 401 Unauthorized, with the message: "timestamp": "2022-07-23T06:34:10.256+00:00", "status": 401, "error": "Unauthorized", "message": "Unauthorized", "path": "/api/role".

2. Add new role

The screenshot shows a POST request to `http://localhost:8080/api/role`. The Body tab is selected, showing the JSON input: `{ "name": "USER" }`. The response status is 200 OK, with the message: "id": 2, "name": "USER".

Employee Management – Spring Rest Project

3. Add new user

The screenshot shows the Postman application interface. A POST request is being made to `http://localhost:8080/api/user`. The request body is a JSON object:

```
1 {
2   "username": "employee2",
3   "password": "employee@123",
4   "roles": [
5     {
6       "id": 2,
7       "name": "USER"
8     }
9   ]
10 }
```

The response status is 200 OK, with a response body containing:

```
1 {
2   "id": 2,
3   "username": "employee2",
4   "password": "$2a$10$53U.m5PtFCw86XF5xUtGT.Y0DulV4JsteQsSsIa4b7pjJeJOSVMB1m",
5   "roles": [
6     {
7       "id": 2,
8       "name": "USER"
9     }
10   ]
11 }
```

4. Get all employees

The screenshot shows the Postman application interface. A GET request is being made to `http://localhost:8080/api/employees`. The response status is 200 OK, with a response body containing a list of employees:

```
1 [
2   {
3     "id": 1,
4     "firstName": "Suresh",
5     "lastName": "Kumar",
6     "email": "suresh.kumar@gmail.com"
7   },
8   {
9     "id": 2,
10    "firstName": "Ramesh",
11    "lastName": "Yadav",
12    "email": "ramesh.yadav@gmail.com"
13  },
14  {
15    "id": 3,
16    "firstName": "Keerti",
17    "lastName": "Verma",
18    "email": "keerti.verma@gmail.com"
19  }
20 ]
```

Employee Management – Spring Rest Project

5. Add new employee

The screenshot shows the Postman application interface. A POST request is being made to `http://localhost:8080/api/employees`. The request body contains the following JSON data:

```
1 {"id": 9,
2   "firstName": "Shyam",
3   "lastName": "Malhotra",
4   "email": "Shyam.malhotra@gmail.com"}  
5
```

The response status is 200 OK, with a response body containing the same JSON data as the request body.

6. Get employee by id

The screenshot shows the Postman application interface. A GET request is being made to `http://localhost:8080/api/employees/1`. The response status is 200 OK, with a response body containing the following JSON data:

```
1 {"id": 1,
2   "firstName": "Suresh",
3   "lastName": "Kumar",
4   "email": "suresh.kumar@gmail.com"}  
5
```

The taskbar at the bottom of the screen shows various application icons and system status.

Employee Management – Spring Rest Project

7. Update employee by id

The screenshot shows the Postman application interface. A collection named "http://localhost:8080/api/employees/1" is selected. A PUT request is being prepared to the URL "http://localhost:8080/api/employees/1". The "Body" tab is active, showing a JSON payload:

```
1
2   ...
3     "firstName": "Sunil",
4     "lastName": "kumar",
5     "email": "sunil.kumar@gmail.com"
```

The response status is 200 OK, time 25 ms, size 514 B. The Windows taskbar at the bottom shows the search bar and system tray.

8. Delete employee by id

The screenshot shows the Postman application interface. A DELETE request is being prepared to the URL "http://localhost:8080/api/employees/9". The "Body" tab is active, showing a simple text message:

```
1 Deleted employee id - 9
```

The response status is 200 OK, time 20 ms, size 458 B. The Windows taskbar at the bottom shows the search bar and system tray.

Employee Management – Spring Rest Project

9. Search employee by firstname

The screenshot shows the Postman interface with a successful API call. The URL is `http://localhost:8080/api/employees/search/rahul`. The response status is 200 OK, time 40 ms, size 594 B. The response body is a JSON array containing two employees:

```
1
2 [
3     {
4         "id": 5,
5         "firstName": "Rahul",
6         "lastName": "Singh",
7         "email": "rahul.singh@gmail.com"
8     },
9     {
10        "id": 6,
11        "firstName": "Rahul",
12        "lastName": "More",
13        "email": "rahul.more@gmail.com"
14    }
]
```

10. Sort the employees in ascending order by firstname

The screenshot shows the Postman interface with a successful API call. The URL is `http://localhost:8080/api/employees/sort`. The response status is 200 OK, time 21 ms, size 1.07 KB. The response body is a JSON array containing three employees, sorted by first name:

```
1
2 [
3     {
4         "id": 8,
5         "firstName": "Bhuwan",
6         "lastName": "Agarwal",
7         "email": "bhuwan.agarwal@gmail.com"
8     },
9     {
10        "id": 7,
11        "firstName": "Divya",
12        "lastName": "Joshi",
13        "email": "divya.joshi@gmail.com"
14    },
15    {
16        "id": 3,
17        "firstName": "Keerti",
18        "lastName": "Verma",
19        "email": "keerti.verma@gmail.com"
20    }
]
```

Employee Management – Spring Rest Project

Postman screenshots for User login:

11. Login page

The screenshot shows the Postman interface with a successful login request. The URL is `http://localhost:8080/api/role`. The method is `POST`. The `Authorization` tab is selected, showing `Basic Auth` with `Username: employee2` and `Password: employee@123`. The response status is `200 OK` with a size of `532 B`.

```
1
2     "id": 2,
3     "name": "USER"
4
```

12. Add new role

The screenshot shows the Postman interface with an attempt to add a new role. The URL is `http://localhost:8080/api/role`. The method is `POST`. The `Body` tab is selected, showing `raw` JSON input: `{"name": "USER1"}`. The response status is `403 Forbidden` with a size of `561 B`.

```
1
2     "timestamp": "2022-07-23T06:50:07.532+00:00",
3     "status": 403,
4     "error": "Forbidden",
5     "message": "Forbidden",
6     "path": "/api/role"
7
```

Employee Management – Spring Rest Project

13. Add new user

The screenshot shows the Postman application interface. A POST request is being made to `http://localhost:8080/api/user`. The request body contains the following JSON:

```
1 {
2     "username": "employee3",
3     "password": "employee@123",
4     "roles": [
5         {
6             "id": 2,
7             "name": "USER"
8         }
9     ]
}
```

The response status is 403 Forbidden, with the following JSON content:

```
1 {
2     "timestamp": "2022-07-23T06:50:50.637+00:00",
3     "status": 403,
4     "error": "Forbidden",
5     "message": "Forbidden",
6     "path": "/api/user"
7 }
```

14. Get all employees

The screenshot shows the Postman application interface. A GET request is being made to `http://localhost:8080/api/employees`. The response status is 200 OK, and the JSON data returned is:

```
1 [
2     {
3         "id": 1,
4         "firstName": "Sunil",
5         "lastName": "Kumar",
6         "email": "sunil.kumar@gmail.com"
7     },
8     {
9         "id": 2,
10        "firstName": "Ramesh",
11        "lastName": "Yadav",
12        "email": "ramesh.yadav@gmail.com"
13    },
14    {
15        "id": 3,
16        "firstName": "Keerti",
17        "lastName": "Verma",
18        "email": "keerti.verma@gmail.com"
19    }
]
```

Employee Management – Spring Rest Project

15. Add new employee

The screenshot shows the Postman application interface. A POST request is being made to `http://localhost:8080/api/employees`. The request body contains the following JSON:

```
1 "firstName": "Shyam",
2 "lastName": "Malhotra",
3 "email": "Shyam.malhotra@gmail.com"
```

The response status is 403 Forbidden, with the following JSON content:

```
1 {
2   "timestamp": "2022-07-23T06:51:46.568+00:00",
3   "status": 403,
4   "error": "Forbidden",
5   "message": "Forbidden",
6   "path": "/api/employees"
7 }
```

The Windows taskbar at the bottom shows various open applications and the date/time as 23-07-2022.

16. Get employee by id

The screenshot shows the Postman application interface. A GET request is being made to `http://localhost:8080/api/employees/3`. The response status is 200 OK, with the following JSON content:

```
1 {
2   "id": 3,
3   "firstName": "Keerti",
4   "lastName": "Verma",
5   "email": "keerti.verma@gmail.com"
6 }
```

The Windows taskbar at the bottom shows various open applications and the date/time as 23-07-2022.

Employee Management – Spring Rest Project

17. Update employee by id

The screenshot shows the Postman application interface. A PUT request is being made to `http://localhost:8080/api/employees/3`. The request body is a JSON object with fields `firstName`, `lastName`, and `email`, all set to "Sonu", "Verma", and "sonu.verma@gmail.com" respectively. The response status is 403 Forbidden, with a timestamp of 2022-07-23T06:53:11.965+00:00, status 403, error "Forbidden", message "Forbidden", and path "/api/employees/3".

18. Delete employee by id

The screenshot shows the Postman application interface. A DELETE request is being made to `http://localhost:8080/api/employees/5`. The response status is 403 Forbidden, with a timestamp of 2022-07-23T06:53:30.166+00:00, status 403, error "Forbidden", message "Forbidden", and path "/api/employees/5".

Employee Management – Spring Rest Project

19. Search employee by firstname

The screenshot shows the Postman application interface. A GET request is made to `http://localhost:8080/api/employees/search/Keerti`. The response status is 200 OK, and the response body is a JSON object:

```
1
2
3
4   "id": 3,
5   "firstName": "Keerti",
6   "lastName": "Verma",
7   "email": "keerti.verma@gmail.com"
8 }
```

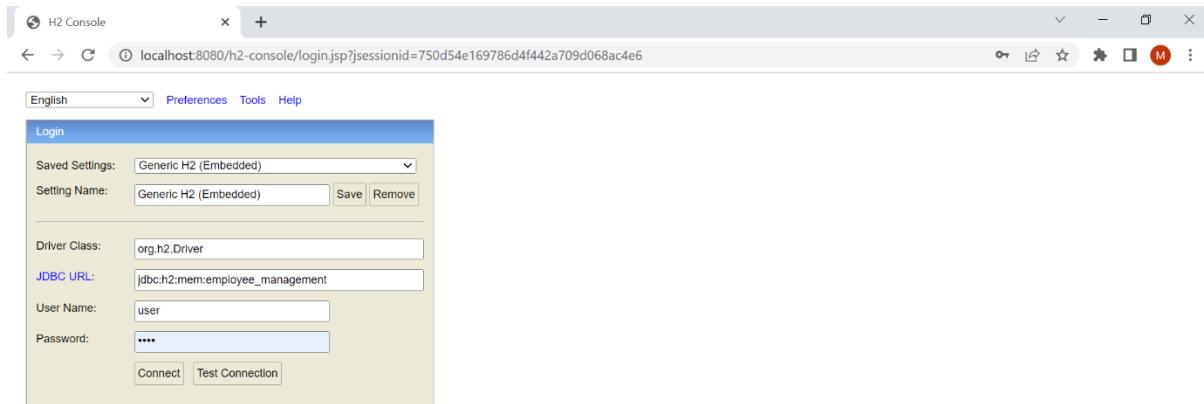
20. Sort the employees in ascending order by firstname

The screenshot shows the Postman application interface. A GET request is made to `http://localhost:8080/api/employees/sort`. The response status is 200 OK, and the response body is a JSON array of three employee objects:

```
1
2
3
4   {
5     "id": 8,
6     "firstName": "Bhuwan",
7     "lastName": "Agarwal",
8     "email": "bhuwan.agarwal@gmail.com"
9   },
10
11   {
12     "id": 7,
13     "firstName": "Divya",
14     "lastName": "Joshi",
15     "email": "divya.joshi@gmail.com"
16   },
17
18   {
19     "id": 3,
20     "firstName": "Keerti",
21     "lastName": "Verma",
22     "email": "keerti.verma@gmail.com"
23 }
```

Employee Management – Spring Rest Project

H2 Database screenshots:



The screenshot shows the H2 Console interface after running a query. The left sidebar lists database objects: EMPLOYEE, ROLES, USERS, USERS_ROLES, INFORMATION_SCHEMA, Sequences, and Users. The main area displays the result of the query 'SELECT * FROM EMPLOYEE'.

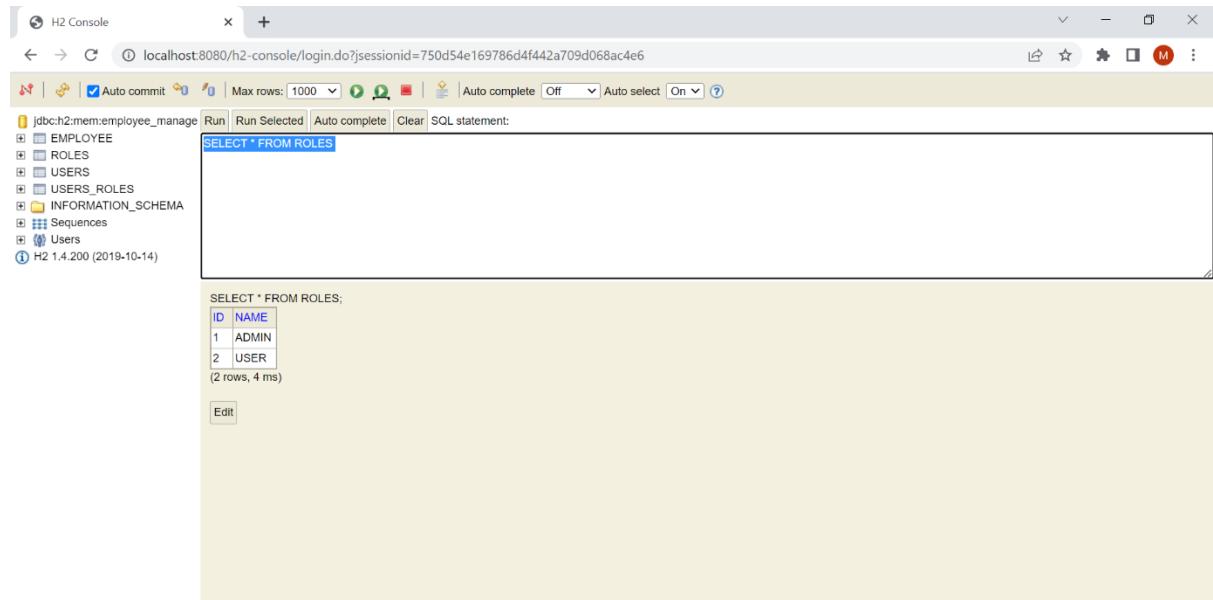
ID	EMAIL	FIRST_NAME	LAST_NAME
1	sunil.kumar@gmail.com	Sunil	kumar
2	ramesh.yadav@gmail.com	Ramesh	Yadav
3	keerti.verma@gmail.com	Keerti	Verma
4	priyanka.sharma@gmail.com	Priyanka	Sharma
5	rahul.singh@gmail.com	Rahul	Singh
6	rahul.more@gmail.com	Rahul	More
7	divya.joshi@gmail.com	Divya	Joshi
8	bhuwan.agarwal@gmail.com	Bhuwan	Agarwal

(8 rows, 4 ms)

Buttons for 'Edit' and 'Run' are present at the bottom of the result table.



Employee Management – Spring Rest Project

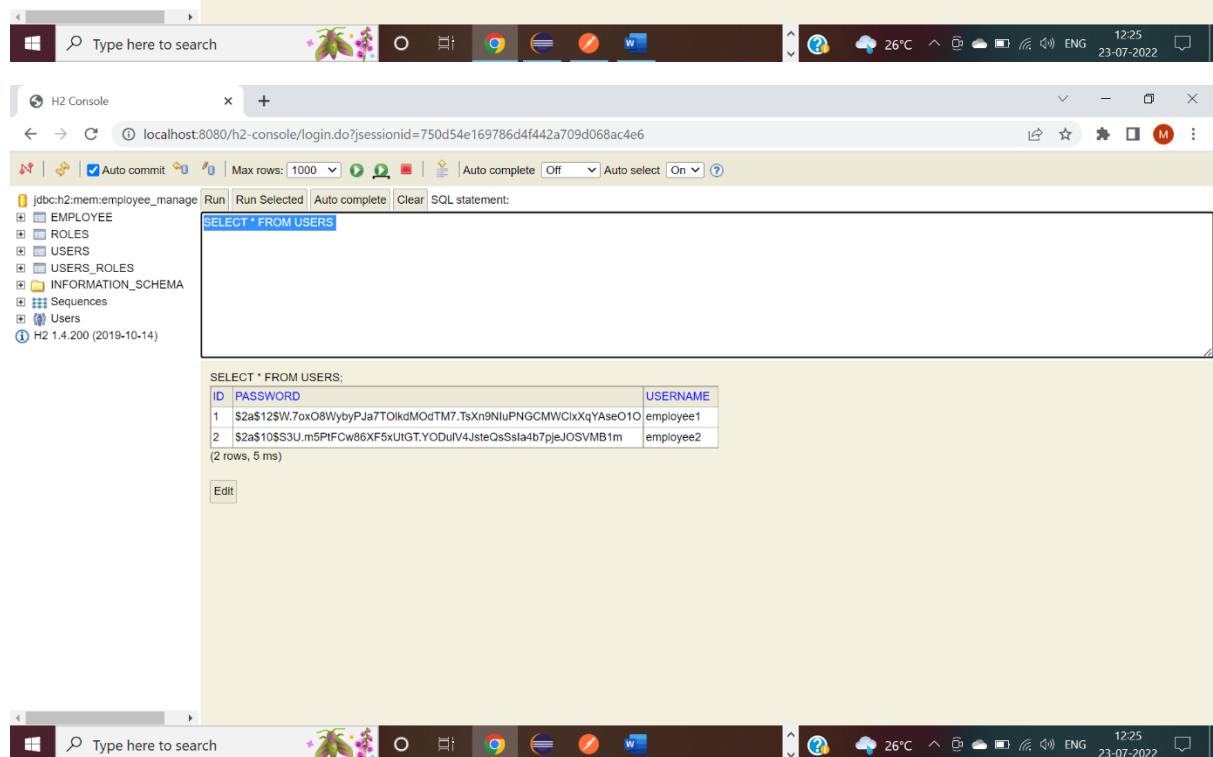


The screenshot shows the H2 Console interface with the URL `localhost:8080/h2-console/login.do?sessionid=750d54e169786d4f442a709d068ac4e6`. The left sidebar lists database schemas: EMPLOYEE, ROLES, USERS, USERS_ROLES, INFORMATION_SCHEMA, Sequences, and Users. The main area displays the result of the SQL query `SELECT * FROM ROLES;`. The output is a table with two rows:

ID	NAME
1	ADMIN
2	USER

(2 rows, 4 ms)

Edit



The screenshot shows the H2 Console interface with the same URL. The left sidebar lists the same database schemas. The main area displays the result of the SQL query `SELECT * FROM USERS;`. The output is a table with two rows:

ID	PASSWORD	USERNAME
1	\$2a\$12\$W.7oxO8WybyPJa7TOlkdModTM7.TsXn9NiuPNGCMWCixXqYAsEo1O	employee1
2	\$2a\$10\$\$3U.m5PiFCw86XF5xUfGT.YODuIV4JsteQsSla4b7pjJOSVMB1m	employee2

(2 rows, 5 ms)

Edit

Employee Management – Spring Rest Project

The screenshot shows the H2 Console interface running on port 8080. The left sidebar lists database objects: EMPLOYEE, ROLES, USERS, USERS_ROLES, INFORMATION_SCHEMA, Sequences, and Users. The main area displays the result of the SQL query `SELECT * FROM USERS_ROLES;`. The results are shown in a table:

USER_ID	ROLE_ID
1	1
2	2

(2 rows, 6 ms)

The system status bar at the bottom indicates: Type here to search, 26°C, ENG, 12:25, 23-07-2022.