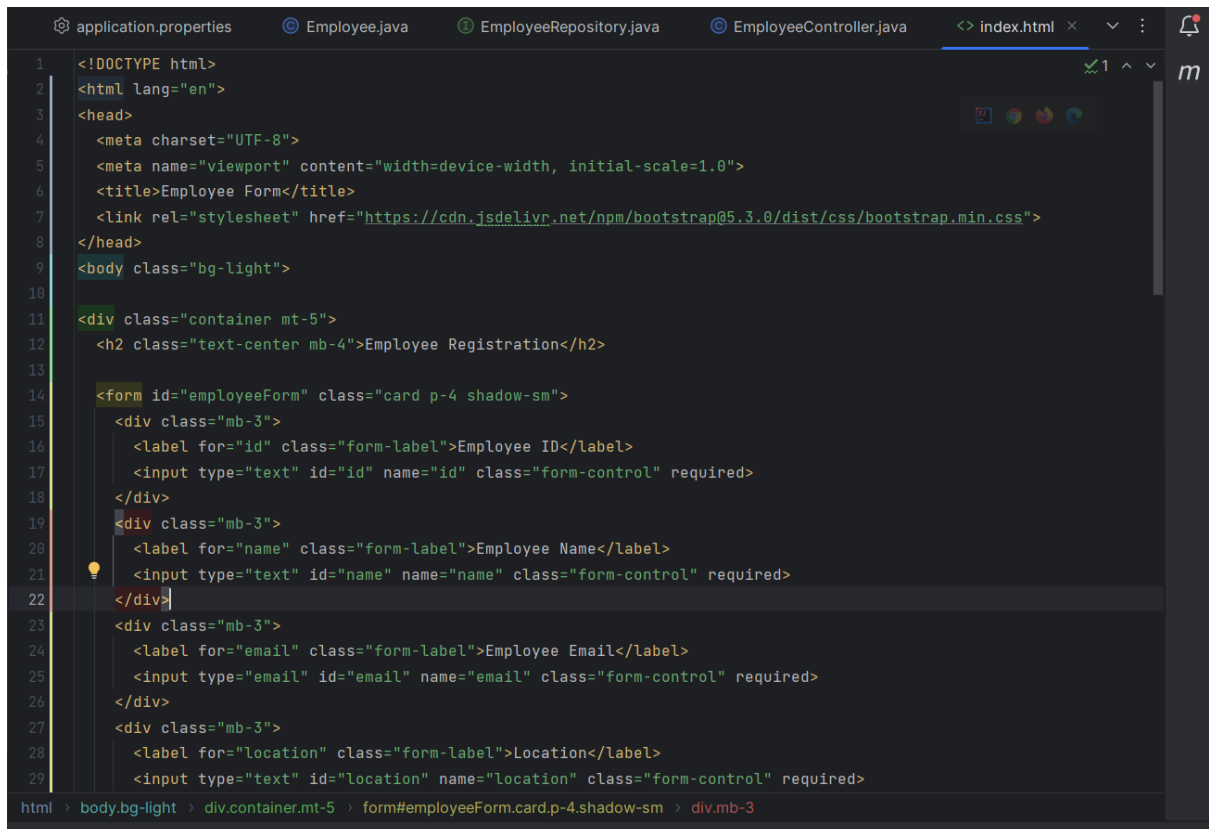


String Data Mongodb

Create a Restful API using spring mvc as per the requirements below: (should use mongodb) I

1. The application should display a Form in the landing page (index.page) where the user can enter the 'Employee' details like 'Employee ID', 'Employee Name', 'Employee Email' and 'Location' and upon submission the data should be saved onto a database table
2. The application should also support a /displayAll' path which shows the list of all the Employees.
3. The application should also support a /display/<user id>' path where upon passing of userid as the parameter the details of only the particular Employee holding that Employee ID will be displayed.
4. Code all the above functionalities required as a RESTful URL.



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8">
5 <meta name="viewport" content="width=device-width, initial-scale=1.0">
6 <title>Employee Form</title>
7 <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">
8 </head>
9 <body class="bg-light">
10
11 <div class="container mt-5">
12 <h2 class="text-center mb-4">Employee Registration</h2>
13
14 <form id="employeeForm" class="card p-4 shadow-sm">
15 <div class="mb-3">
16 <label for="id" class="form-label">Employee ID</label>
17 <input type="text" id="id" name="id" class="form-control" required>
18 </div>
19 <div class="mb-3">
20 <label for="name" class="form-label">Employee Name</label>
21 <input type="text" id="name" name="name" class="form-control" required>
22 </div>
23 <div class="mb-3">
24 <label for="email" class="form-label">Employee Email</label>
25 <input type="email" id="email" name="email" class="form-control" required>
26 </div>
27 <div class="mb-3">
28 <label for="location" class="form-label">Location</label>
29 <input type="text" id="location" name="location" class="form-control" required>
```

```

<h3 class="mb-3">All Employees</h3>
<button class="btn btn-success mb-3" onclick="loadEmployees()">Load Employees</button>
<ul id="employeeList" class="list-group"></ul>
</div>

<script>
const apiUrl = "http://localhost:8080/employees"; // change if needed

// Submit form
document.getElementById("employeeForm").addEventListener("submit", function (event) {
    event.preventDefault();

    const employee = {
        id: document.getElementById("id").value,
        name: document.getElementById("name").value,
        email: document.getElementById("email").value,
        location: document.getElementById("location").value
    };

    fetch(apiUrl + "/add", {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify(employee)
    })
    .then(response => response.json())
    .then(data => {
        document.getElementById("message").innerHTML =
            <div class='alert alert-success'>Employee saved successfully!</div>;
    });

```

body.bg-light > div.container.mt-5 > form#employeeForm.card.p-4.shadow-sm > div.mb-3

```

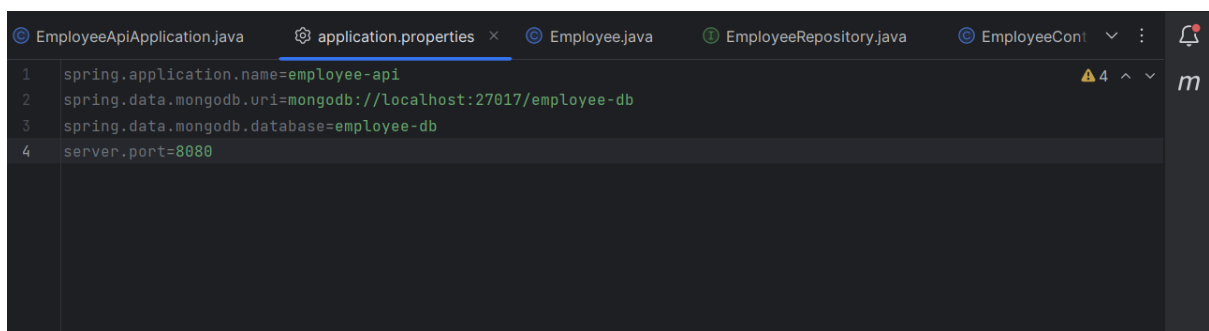
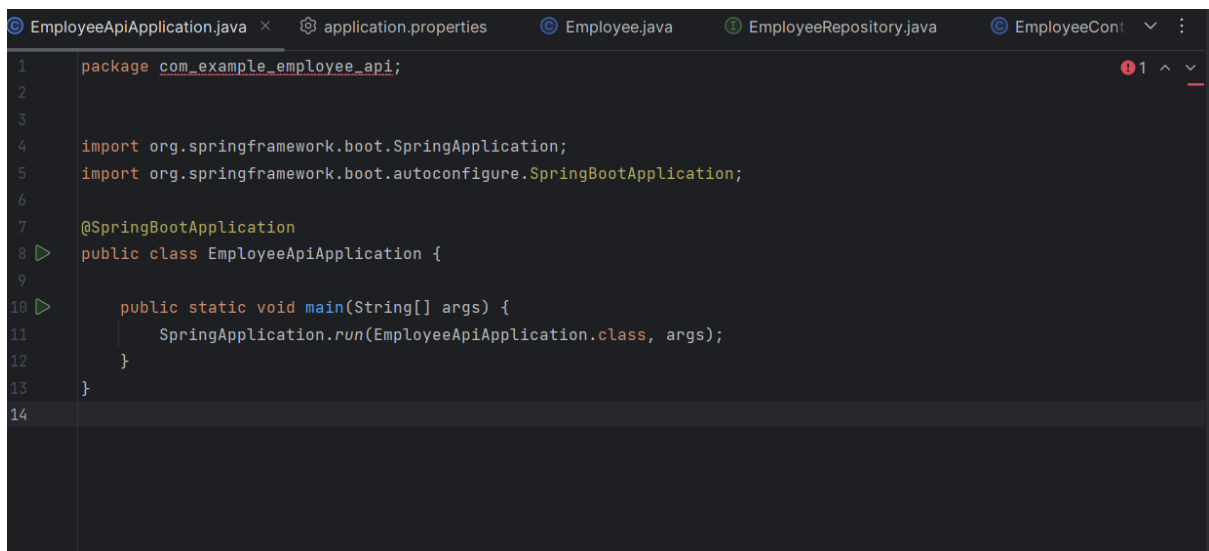
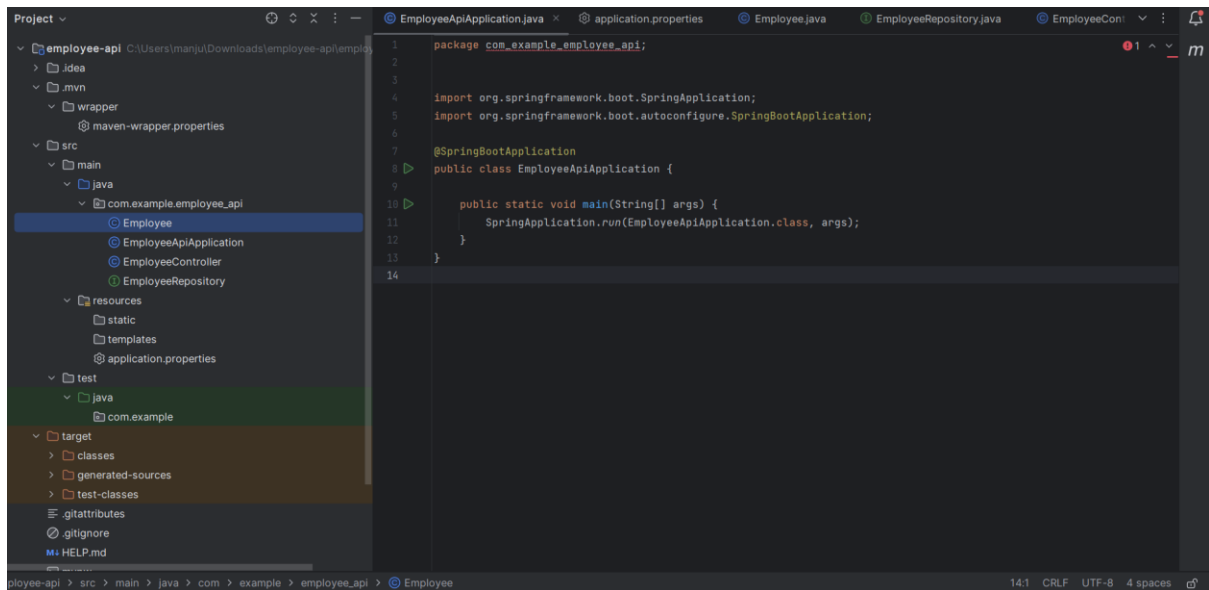
    .then(response => response.json())
    .then(data => {
        document.getElementById("message").innerHTML =
            <div class='alert alert-success'>Employee saved successfully!</div>;
        document.getElementById("employeeForm").reset();
    })
    .catch(error => {
        document.getElementById("message").innerHTML =
            <div class='alert alert-danger'>Error saving employee!</div>;
    });
});

// Load employees
function loadEmployees() {
    fetch(apiUrl + "/displayAll")
    .then(response => response.json())
    .then(data => {
        const list = document.getElementById("employeeList");
        list.innerHTML = "";
        data.forEach(emp => {
            list.innerHTML += `<li class="list-group-item">
                <strong>${emp.name}</strong> (${emp.email}) - ${emp.location}
            </li>`;
        });
    });
}
</script>

</body>

```

body.bg-light > div.container.mt-5 > form#employeeForm.card.p-4.shadow-sm > div.mb-3



```
EmployeeApiApplication.java application.properties Employee.java EmployeeRepository.java EmployeeConti
1 package com.example.employee_api.model;
2
3 import org.springframework.data.annotation.Id;
4 import org.springframework.data.mongodb.core.mapping.Document;
5
6 9 usages
7 @Document(collection = "employee") // matches your MongoDB collection
8 public class Employee {
9
10 3 usages
11 @Id
12 private String id;
13
14 3 usages
15 private String name;
16 3 usages
17 private String email;
18 3 usages
19 private String location;
20
21 no usages
22 public Employee() {}
23
24 no usages
25 public Employee(String id, String name, String email, String location) {
26     this.id = id;
27     this.name = name;
28     this.email = email;
29     this.location = location;
30 }
31
32 10:26 CRLF UTF-8 4 spaces
```

```
EmployeeApiApplication.java application.properties Employee.java EmployeeRepository.java EmployeeConti
21     this.email = email;
22     this.location = location;
23 }
24
25 // Getters and Setters
26 no usages
27 public String getId() { return id; }
28 no usages
29 public void setId(String id) { this.id = id; }
30
31 1 usage
32 public String getName() { return name; }
33 1 usage
34 public void setName(String name) { this.name = name; }
35
36 1 usage
37 public String getEmail() { return email; }
38 1 usage
39 public void setEmail(String email) { this.email = email; }
40
41 1 usage
42 public String getLocation() { return location; }
43 1 usage
44 public void setLocation(String location) { this.location = location; }
45 }
46
47 10:26 CRLF UTF-8 4 spaces
```

```
EmployeeApiApplication.java  application.properties  Employee.java  EmployeeRepository.java  EmployeeConti
1 package com.example.employee_api.repository;
2
3 import org.springframework.data.mongodb.repository.MongoRepository;
4 import com.example.employee_api.model.Employee;
5
6 2 usages
7 public interface EmployeeRepository extends MongoRepository<Employee, String> {
8 }
```

```
ApiApplication.java  application.properties  Employee.java  EmployeeRepository.java  EmployeeController.java
31 @GetMapping("/display/{id}")
32 public Optional<Employee> getEmployeeById(@PathVariable String id) {
33     return employeeRepository.findById(id);
34 }
35
36 // Update employee by ID
37 no usages
38 @PutMapping("/update/{id}")
39 public Employee updateEmployee(@PathVariable String id, @RequestBody Employee employeeDetails) {
40     return employeeRepository.findById(id).map(employee -> {
41         employee.setName(employeeDetails.getName());
42         employee.setEmail(employeeDetails.getEmail());
43         employee.setLocation(employeeDetails.getLocation());
44         return employeeRepository.save(employee);
45     }).orElseThrow(() -> new RuntimeException("Employee not found with id " + id));
46 }
47
48 // Delete employee by ID
49 no usages
50 @DeleteMapping("/delete/{id}")
51 public String deleteEmployee(@PathVariable String id) {
52     employeeRepository.deleteById(id);
53     return "Employee with ID " + id + " deleted successfully!";
54 }
55 }
```

EmployeeController > updateEmployee > Lambda 41:59 CRLF UTF-8 4 sp

The output:

Add:

Employee Registration

Employee ID

122334

Employee Name

sam

Employee Email

sam@gmail.com

Location

chennai

Save Employee

http://localhost:8080/employee/add

Save

Share

POST

http://localhost:8080/employee/add

Send

Params

Authorization

Headers (9)

Body

Scripts

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

1

{

2

"id": "098877",

3

"name": "alex",

4

"email": "alex@example.com",

5

"location": "Bangalore"

6

}

Body

Cookies

Headers (5)

Test Results

200 OK

414 ms

243 B

{}

JSON

Preview

Visualize

1

{

2

"id": "098877",

3

"name": "alex",

4

"email": "alex@example.com",

5

"location": "Bangalore"

6

}

Postbot

Runner

Start Proxy

Cookies

Vault

Trash

DisplayAll:

Body

Cookies

Headers (5)

Test Results

200 OK

136 ms

421 B

{}

JSON

Preview

Visualize

7

{,

8

{

9

"id": "60ba55adc0059d34d55066e0",

10

"name": "Swetha",

11

"email": "swetha@example.com",

12

"location": "Bangalore"

13

},

14

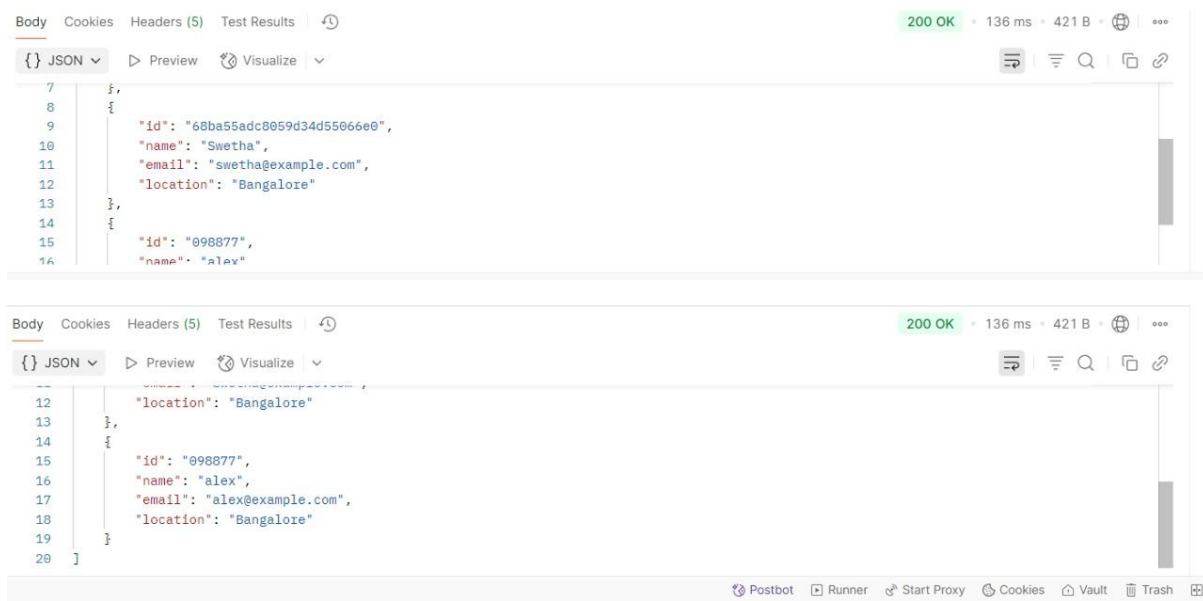
{

15

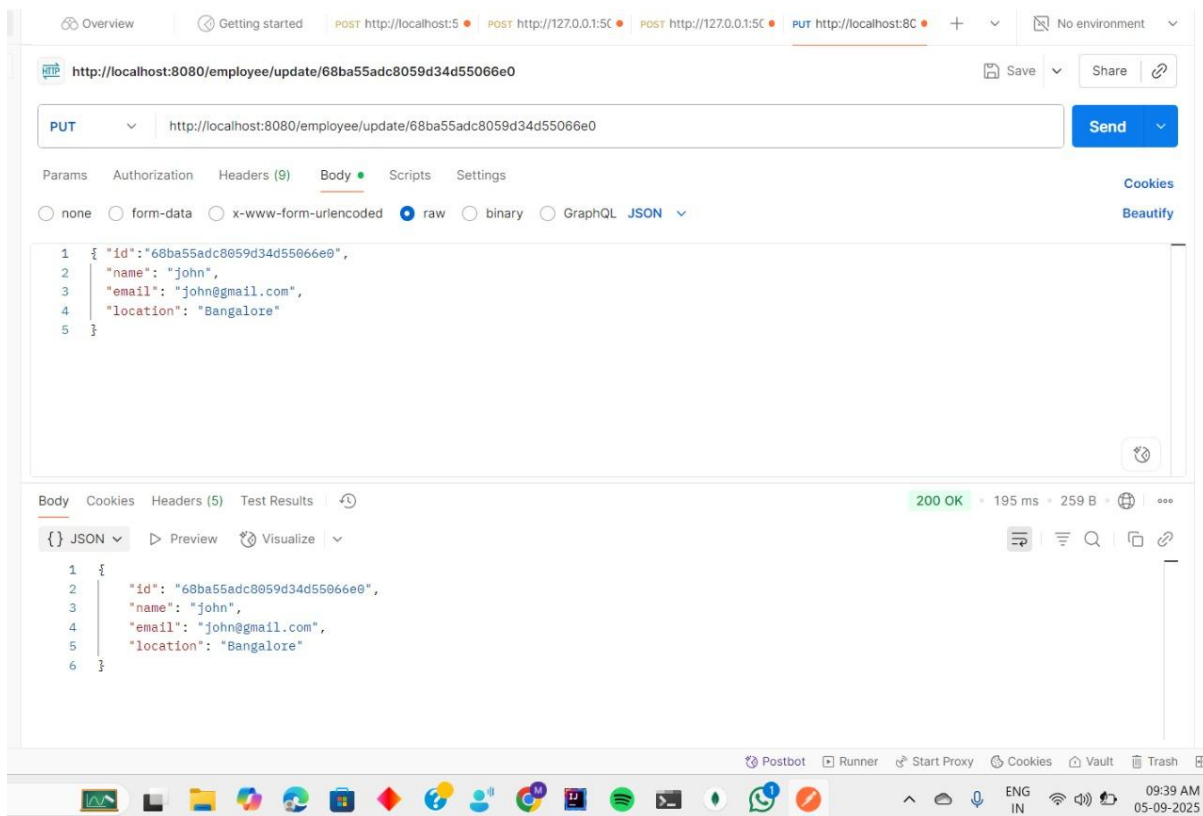
"id": "098877",

16

"name": "alex"



Update:



Delete:

