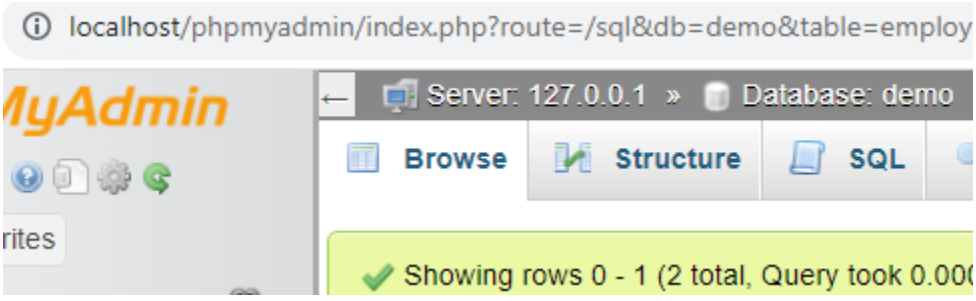
Spring Boot CRUD Web Application with Thymeleaf, Spring Data JPA, Hibernate, MySQL

Step 1:-

Run your xampp and open localhost/phpmyadmin

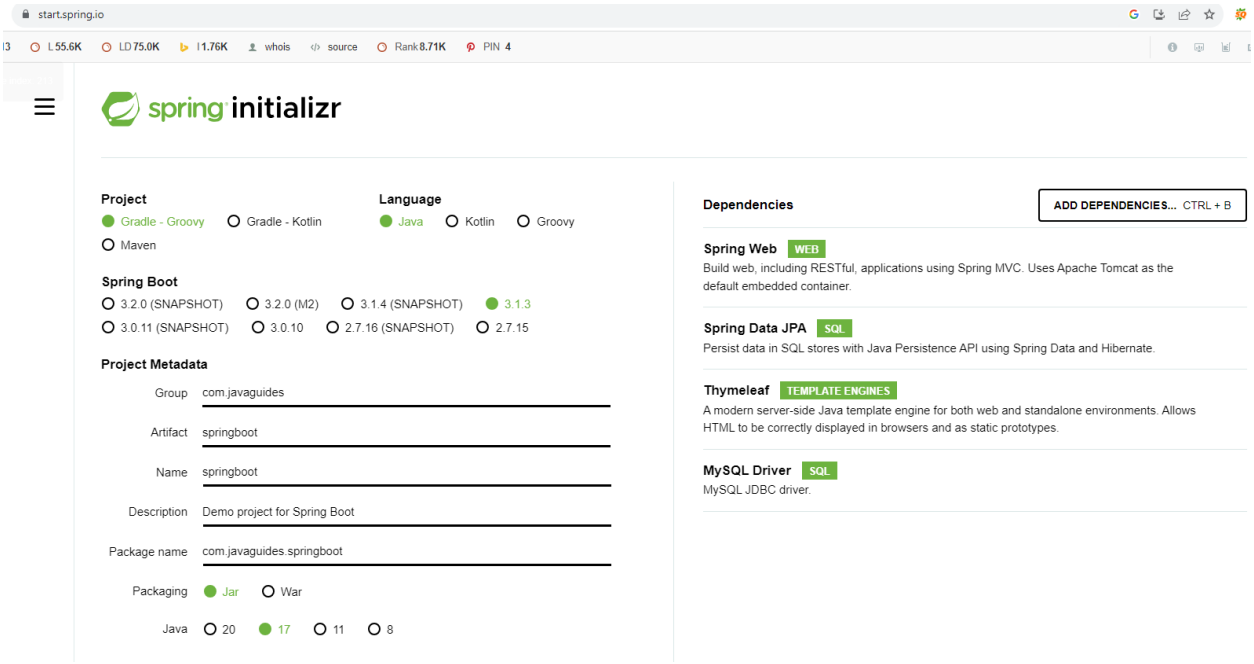
& create database demo as shown below

After it you will see database .

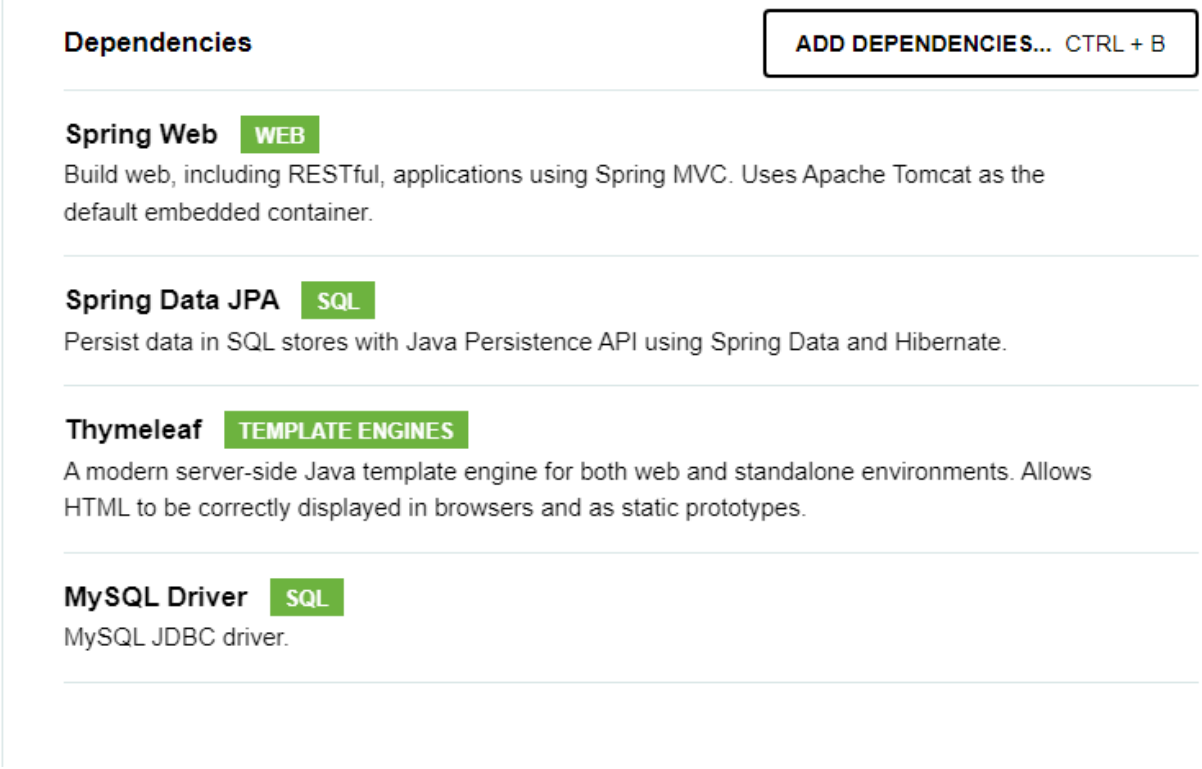


Step 2:- go to website https://start.spring.io/

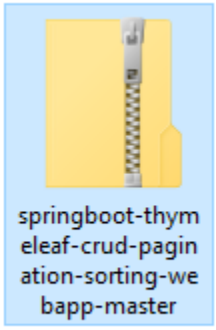
& create project and add dependencies as shown below



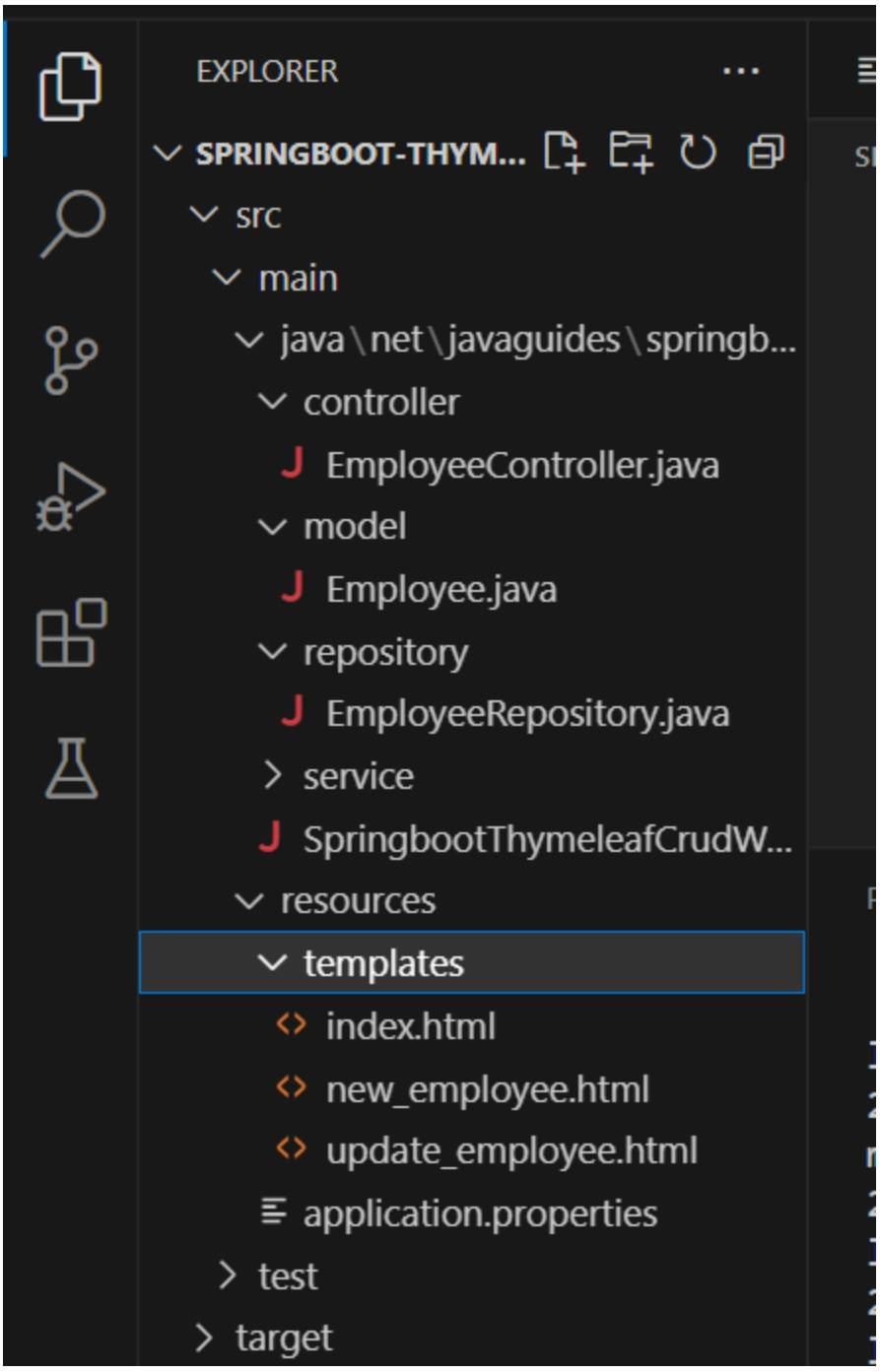
Note:- Following dependencies are important



Finally click on Generate and you will get zip file downloaded

:- 

And extract it open with visual code here you will create following package and files as shown below :-



First of all open application.properties file and add following :-

```
# DATASOURCE (DataSourceAutoConfiguration & DataSourceProperties)
spring.datasource.url=jdbc:mysql://localhost:3306/demo
spring.datasource.username=root
spring.datasource.password=

# Hibernate
```

```
# The SQL dialect makes Hibernate generate better SQL for the chosen database
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQLDialect

# Hibernate ddl auto (create, create-drop, validate, update)
spring.jpa.hibernate.ddl-auto = update

logging.level.org.hibernate.SQL=DEBUG
logging.level.org.hibernate.type=TRACE
```

Step 1:- create model package folder inside your springboot(main folder):-

Employee.java file code :-

```
package net.javaguides.springboot.model;

import jakarta.persistence.*;

@Entity
@Table(name = "employees")
public class Employee {

    @Id
    @GeneratedValue(strategy =  GenerationType.IDENTITY)
    private long id;

    @Column(name = "first_name")
    private String firstName;

    @Column(name = "last_name")
    private String lastName;

    @Column(name = "email")
    private String email;
    public long getId() {
        return id;
    }
    public void setId(long id) {
        this.id = id;
    }
    public String getFirstName() {
        return firstName;
    }
```

```
    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }
    public String getLastName() {
        return lastName;
    }
    public void setLastName(String lastName) {
        this.lastName = lastName;
    }
    public String getEmail() {
        return email;
    }
    public void setEmail(String email) {
        this.email = email;
    }
}
```

Now create repository package folder and inside it

`EmployeeRepository.java file code :-`

```
package net.javaguides.springboot.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import net.javaguides.springboot.model.Employee;

@Repository
public interface EmployeeRepository extends JpaRepository<Employee, Long>{

}
```

Now create service package folder inside it create following files

`EmployeeService.java file code :-`

```
package net.javaguides.springboot.service;

import java.util.List;
```

```
import org.springframework.data.domain.Page;

import net.javaguides.springboot.model.Employee;

public interface EmployeeService {
    List<Employee> getAllEmployees();
    void saveEmployee(Employee employee);
    Employee getEmployeeById(long id);
    void deleteEmployeeById(long id);
    Page<Employee> findPaginated(int pageNo, int pageSize, String sortField,
String sortDirection);
}
```

Now create `EmployeeServiceImpl.java file code :-`

```
package net.javaguides.springboot.service;

import java.util.List;
import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Sort;
import org.springframework.stereotype.Service;

import net.javaguides.springboot.model.Employee;
import net.javaguides.springboot.repository.EmployeeRepository;

@Service
public class EmployeeServiceImpl implements EmployeeService {

    @Autowired
    private EmployeeRepository employeeRepository;

    @Override
    public List<Employee> getAllEmployees() {
        return employeeRepository.findAll();
    }

    @Override
    public void saveEmployee(Employee employee) {
```

```
        this.employeeRepository.save(employee);
    }

    @Override
    public Employee getEmployeeById(long id) {
        Optional<Employee> optional = employeeRepository.findById(id);
        Employee employee = null;
        if (optional.isPresent()) {
            employee = optional.get();
        } else {
            throw new RuntimeException(" Employee not found for id :: " + id);
        }
        return employee;
    }

    @Override
    public void deleteEmployeeById(long id) {
        this.employeeRepository.deleteById(id);
    }

    @Override
    public Page<Employee> findPaginated(int pageNo, int pageSize, String
sortField, String sortDirection) {
        Sort sort = sortDirection.equalsIgnoreCase(Sort.Direction.ASC.name()) ?
Sort.by(sortField).ascending() :
            Sort.by(sortField).descending();

        Pageable pageable = PageRequest.of(pageNo - 1, pageSize, sort);
        return this.employeeRepository.findAll(pageable);
    }
}
```

After it

Create controller package folder and inside it create

`EmployeeController.java file code :-`

```
package net.javaguides.springboot.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
```

```java
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;

import net.javaguides.springboot.model.Employee;
import net.javaguides.springboot.service.EmployeeService;

@Controller
public class EmployeeController {

    @Autowired
    private EmployeeService employeeService;

    // display list of employees
    @GetMapping("/")
    public String viewHomePage(Model model) {
        return findPaginated(1, "firstName", "asc", model);
    }

    @GetMapping("/showNewEmployeeForm")
    public String showNewEmployeeForm(Model model) {
        // create model attribute to bind form data
        Employee employee = new Employee();
        model.addAttribute("employee", employee);
        return "new_employee";
    }

    @PostMapping("/saveEmployee")
    public String saveEmployee(@ModelAttribute("employee") Employee employee) {
        // save employee to database
        employeeService.saveEmployee(employee);
        return "redirect:/";
    }

    @GetMapping("/showFormForUpdate/{id}")
    public String showFormForUpdate(@PathVariable ( value = "id") long id, Model
model) {

        // get employee from the service
        Employee employee = employeeService.getEmployeeById(id);
```

```
        // set employee as a model attribute to pre-populate the form
        model.addAttribute("employee", employee);
        return "update_employee";
    }

    @GetMapping("/deleteEmployee/{id}")
    public String deleteEmployee(@PathVariable (value = "id") long id) {

        // call delete employee method
        this.employeeService.deleteEmployeeById(id);
        return "redirect:/";
    }


    @GetMapping("/page/{pageNo}")
    public String findPaginated(@PathVariable (value = "pageNo") int pageNo,
            @RequestParam("sortField") String sortField,
            @RequestParam("sortDir") String sortDir,
            Model model) {
        int pageSize = 5;

        Page<Employee> page = employeeService.findPaginated(pageNo, pageSize,
sortField, sortDir);
        List<Employee> listEmployees = page.getContent();

        model.addAttribute("currentPage", pageNo);
        model.addAttribute("totalPages", page.getTotalPages());
        model.addAttribute("totalItems", page.getTotalElements());

        model.addAttribute("sortField", sortField);
        model.addAttribute("sortDir", sortDir);
        model.addAttribute("reverseSortDir", sortDir.equals("asc") ? "desc" :
"asc");

        model.addAttribute("listEmployees", listEmployees);
        return "index";
    }
}
```

Create html files inside templates folder which you will see inside resources folder :-

Index.html file code :-

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="ISO-8859-1">
<title>Employee Management System</title>

<link rel="stylesheet"
    href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.cs
s"
    integrity="sha384-
MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO"
    crossorigin="anonymous">

</head>
<body>

    <div class="container my-2">
    <h1>Employees List</h1>

    <a th:href = "@{/showNewEmployeeForm}" class="btn btn-primary btn-sm mb-3">
Add Employee </a>

        <table border="1" class = "table table-striped table-responsive-md">
            <thead>
                <tr>
                    <th>
                        <a th:href="@{'/page/' + ${currentPage} +
'?sortField=firstName&sortDir=' + ${reverseSortDir}}">
                            Employee First Name</a>
                    </th>
                    <th>
                    <a th:href="@{'/page/' + ${currentPage} +
'?sortField=lastName&sortDir=' + ${reverseSortDir}}">
                            Employee Last Name</a>
                    </th>
                    <th>
                    <a th:href="@{'/page/' + ${currentPage} +
'?sortField=email&sortDir=' + ${reverseSortDir}}">
                            Employee Email</a>
                    </th>
                    <th> Actions </th>
                </tr>
            </thead>
            <tbody>
                <tr th:each="employee : ${listEmployees}">
```

```
                    <td th:text="${employee.firstName}"></td>
                    <td th:text="${employee.lastName}"></td>
                    <td th:text="${employee.email}"></td>
                    <td> <a
th:href="@{/showFormForUpdate/{id}(id=${employee.id})}" class="btn btn-
primary">Update</a>
                        <a th:href="@{/deleteEmployee/{id}(id=${employee.id})}"
class="btn btn-danger">Delete</a>
                    </td>
                </tr>
            </tbody>
        </table>

        <div th:if = "${totalPages > 1}">
            <div class = "row col-sm-10">
                <div class = "col-sm-2">
                    Total Rows: [[${totalItems}]]
                </div>
                <div class = "col-sm-1">
                    <span th:each="i: ${#numbers.sequence(1, totalPages)}">
                        <a th:if="${currentPage != i}" th:href="@{'/page/' +
${i}+ '?sortField=' + ${sortField} + '&sortDir=' + ${sortDir}}">[[${i}]]</a>
                        <span th:unless="${currentPage !=
i}">[[${i}]]</span>     
                    </span>
                </div>
                <div class = "col-sm-1">
                    <a th:if="${currentPage < totalPages}" th:href="@{'/page/' +
${currentPage + 1}+ '?sortField=' + ${sortField} + '&sortDir=' +
${sortDir}}">Next</a>
                    <span th:unless="${currentPage < totalPages}">Next</span>
                </div>

                 <div class="col-sm-1">
                    <a th:if="${currentPage < totalPages}" th:href="@{'/page/' +
${totalPages}+ '?sortField=' + ${sortField} + '&sortDir=' + ${sortDir}}">Last</a>
                    <span th:unless="${currentPage < totalPages}">Last</span>
                 </div>
            </div>
        </div>
    </div>
</body>
</html>
```

new_employee.html file code :-

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="ISO-8859-1">
<title>Employee Management System</title>
<link rel="stylesheet"
    href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.cs
s"
    integrity="sha384-
MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO"
    crossorigin="anonymous">
</head>
<body>
    <div class="container">
        <h1>Employee Management System</h1>
        <hr>
        <h2>Save Employee</h2>

        <form action="#" th:action="@{/saveEmployee}" th:object="${employee}"
            method="POST">
            <input type="text" th:field="*{firstName}"
                placeholder="Employee First Name" class="form-control mb-4 col-
4">


                <input type="text" th:field="*{lastName}"
                placeholder="Employee Last Name" class="form-control mb-4 col-4">

                <input type="text" th:field="*{email}"
                placeholder="Employee Email" class="form-control mb-4 col-4">

                <button type="submit" class="btn btn-info col-2"> Save
Employee</button>
        </form>

        <hr>

        <a th:href = "@{/}"> Back to Employee List</a>
    </div>
</body>
</html>
```

And update_employee.html file code:-

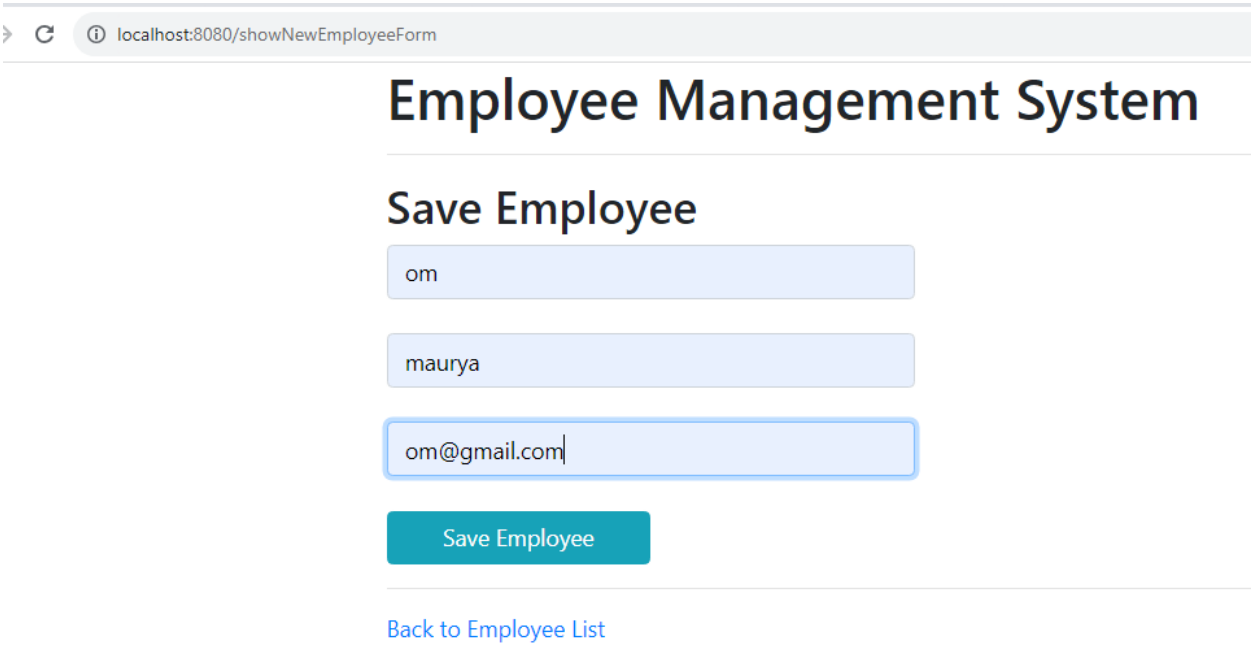
```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="ISO-8859-1">
<title>Employee Management System</title>

<link rel="stylesheet"
    href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.cs
s">
</head>
<body>
    <div class="container">
        <h1>Employee Management System</h1>
        <hr>
        <h2>Update Employee</h2>

        <form action="#" th:action="@{/saveEmployee}" th:object="${employee}"
            method="POST">

            <!-- Add hidden form field to handle update -->
            <input type="hidden" th:field="*{id}" />

            <input type="text" th:field="*{firstName}" class="form-control mb-4
col-4">

                <input type="text" th:field="*{lastName}" class="form-control mb-
4 col-4">

                <input type="text" th:field="*{email}" class="form-control mb-4
col-4">

                <button type="submit" class="btn btn-info col-2"> Update
Employee</button>
        </form>

        <hr>

        <a th:href = "@{/}"> Back to Employee List</a>
    </div>
</body>
</html>
```

Finally your application java file code :-

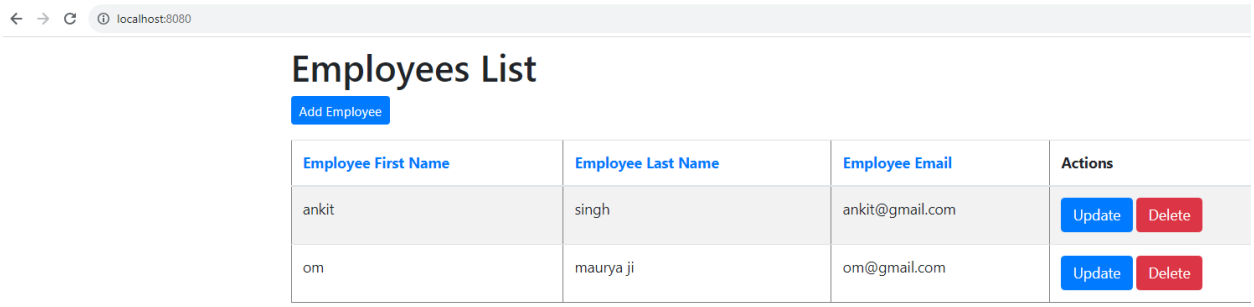
`SpringbootThymeleafCrudWebAppApplication.java` file code:-

```
package net.javaguides.springboot;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class SpringbootThymeleafCrudWebAppApplication {

    public static void main(String[] args) {
        SpringApplication.run(SpringbootThymeleafCrudWebAppApplication.class,
args);
    }

}
```

Run it and you will see output:-



After save you will see



And after click on update you will see