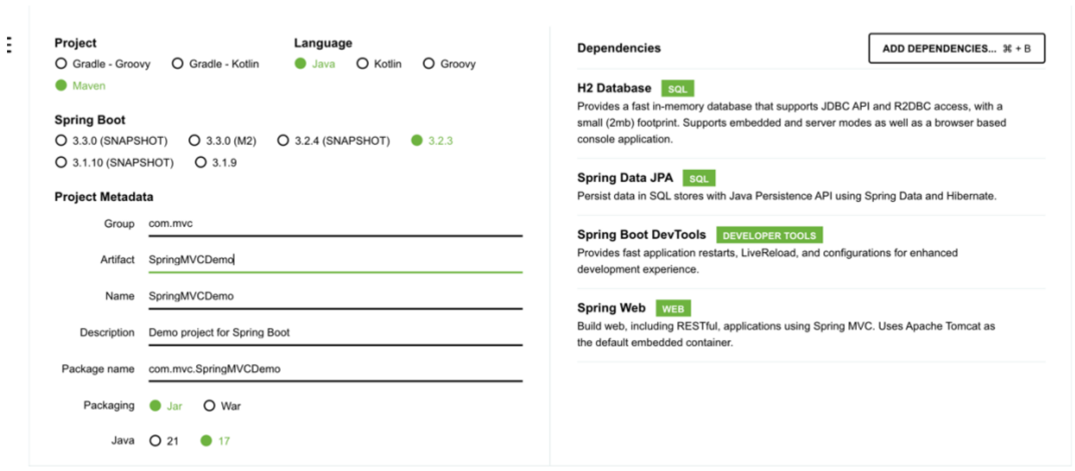
Table of Contents

1. Spring Boot Web Maven Project

1.1. Create a Spring boot maven project with the following dependencies from start.spring.io



Project
Gradle - Groovy Gradle - Kotlin [✓ Maven]

Language
[✓ Java] Kotlin Groovy

Spring Boot
3.3.0 (SNAPSHOT) 3.3.0 (M2) 3.2.4 (SNAPSHOT) [✓ 3.2.3] 3.1.10 (SNAPSHOT) 3.1.9

Project Metadata

Group	com.mvc
Artifact	SpringMVCDemo
Name	SpringMVCDemo
Description	Demo project for Spring Boot
Package name	com.mvc.SpringMVCDemo
Packaging	[✓ Jar] War
Java	21 [✓ 17]

Dependencies

H2 Database SQL
Provides a fast in-memory database that supports JDBC API and R2DBC access, with a small (2mb) footprint. Supports embedded and server modes as well as a browser based console application.

Spring Data JPA SQL
Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

Spring Boot DevTools DEVELOPER TOOLS
Provides fast application restarts, LiveReload, and configurations for enhanced development experience.

Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

1.2. Unzip the project and open in the editor

1.3. The embedded tomcat with spring boot web only includes a light weight server which is the tomcat core. Add the below dependency in pom.xml to be able to compile jsp-files.

```
<dependency>
        <groupId>org.apache.tomcat.embed</groupId>
        <artifactId>tomcat-embed-jasper</artifactId>
</dependency>
```

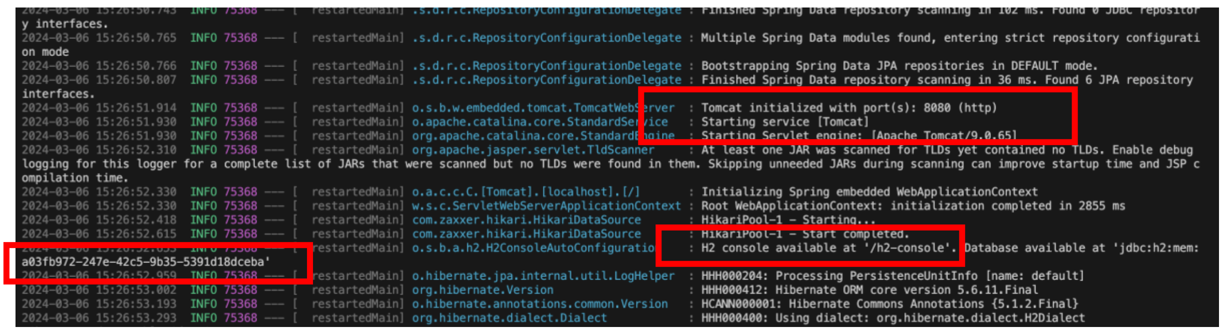
1.4. Also add below dependencies that we will use later:

```
<!-- JSTL Dependency -->
   <dependency>
     <groupId>javax.servlet.jsp.jstl</groupId>
     <artifactId>javax.servlet.jsp.jstl-api</artifactId>
     <version>1.2.1</version>
   </dependency>
   <dependency>
     <groupId>taglibs</groupId>
     <artifactId>standard</artifactId>
     <version>1.1.2</version>
   </dependency>
```

1.5. Applications that use devtools will automatically restart whenever files on the classpath change

2. Understand web environment and H2 database

2.1. Run the main method and observe the console.

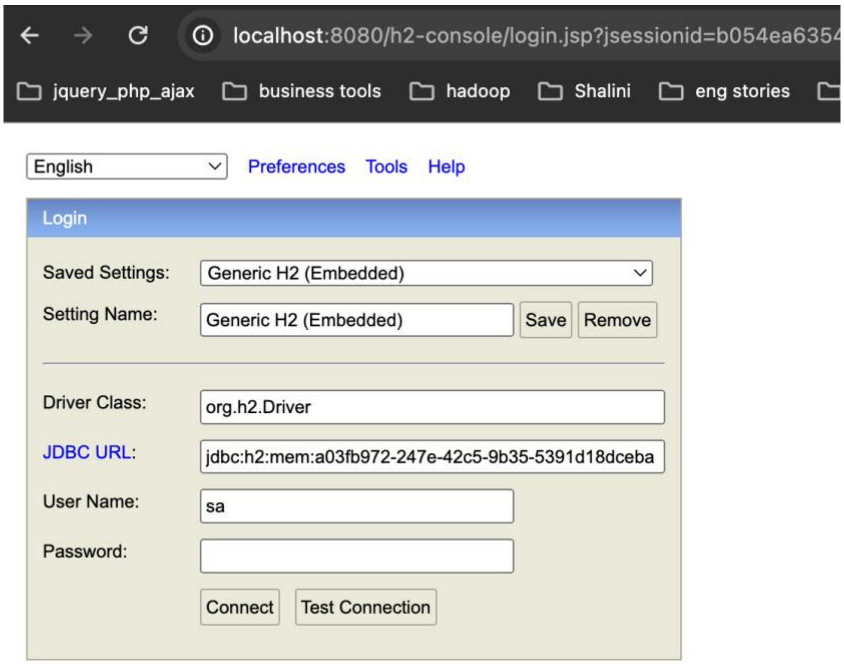


```
2024-03-06 15:26:50.743  INFO 75368 --- [  restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 102 ms. Found 0 JDBC repositor
y interfaces.
2024-03-06 15:26:50.765  INFO 75368 --- [  restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Multiple Spring Data modules found, entering strict repository configurati
on mode
2024-03-06 15:26:50.766  INFO 75368 --- [  restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2024-03-06 15:26:50.807  INFO 75368 --- [  restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 36 ms. Found 6 JPA repository
interfaces.
2024-03-06 15:26:51.914  INFO 75368 --- [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080 (http)
2024-03-06 15:26:51.930  INFO 75368 --- [  restartedMain] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2024-03-06 15:26:51.930  INFO 75368 --- [  restartedMain] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.65]
2024-03-06 15:26:52.310  INFO 75368 --- [  restartedMain] org.apache.jasper.servlet.TldScanner     : At least one JAR was scanned for TLDs yet contained no TLDs. Enable debug
logging for this logger for a complete list of JARs that were scanned but no TLDs were found in them. Skipping unneeded JARs during scanning can improve startup time and JSP c
ompilation time.
2024-03-06 15:26:52.330  INFO 75368 --- [  restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2024-03-06 15:26:52.330  INFO 75368 --- [  restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 2855 ms
2024-03-06 15:26:52.418  INFO 75368 --- [  restartedMain] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
2024-03-06 15:26:52.615  INFO 75368 --- [  restartedMain] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
2024-03-06 15:26:52.633  INFO 75368 --- [  restartedMain] o.s.b.a.h2.H2ConsoleAutoConfiguration    : H2 console available at '/h2-console'. Database available at 'jdbc:h2:mem:
a03fb972-247e-42c5-9b35-5391d18dceba'
2024-03-06 15:26:52.959  INFO 75368 --- [  restartedMain] o.hibernate.jpa.internal.util.LogHelper  : HHH000204: Processing PersistenceUnitInfo [name: default]
2024-03-06 15:26:53.002  INFO 75368 --- [  restartedMain] org.hibernate.Version                    : HHH000412: Hibernate ORM core version 5.6.11.Final
2024-03-06 15:26:53.193  INFO 75368 --- [  restartedMain] o.hibernate.annotations.common.Version   : HCANN000001: Hibernate Commons Annotations {5.1.2.Final}
2024-03-06 15:26:53.293  INFO 75368 --- [  restartedMain] org.hibernate.dialect.Dialect            : HHH000400: Using dialect: org.hibernate.dialect.H2Dialect
```

2.2. Notice tomcat is running on port 8080

2.3. H2 console is available at /h2-console with the url starting with jdbc:h2:mem:<some id>. Default username is sa and no password.
This id is random and keeps changing whenever the server is restarted

2.4. Open browser and type in the url http://localhost:8080/h2-console and copy paste the h2 url from the console as shown in below screen:



Saved Settings: Generic H2 (Embedded)
Setting Name: Generic H2 (Embedded)
Driver Class: org.h2.Driver
JDBC URL: jdbc:h2:mem:a03fb972-247e-42c5-9b35-5391d18dceba
User Name: sa
Password:

2.5. Since url keeps on changing lets override the default url and update the application.properties as follows:

```
spring.h2.console.enabled=true
spring.datasource.url=jdbc:h2:mem:testdb
spring.jpa.show-sql=true
#spring.jpa.generate-ddl=true
```

2.6. Now stop the server and restart to see the output in console for the changed url for h2-console. Login to h2 databse on the browser using **jdbc:h2:mem:testdb**

2.7. Tomcat server bby default runs on port 8080. To change the port add below in application.properties file
server.port=8081

2.8. Stop the server and restart, you will notice now tomcat is available at port 8081. Verify by going on the browser and typing http://localhost:8081/h2-console

3. Controller

3.1. Create a controller class as follows:

```
@Controller
public class GreetController {

        @GetMapping("/welcome")
        public @ResponseBody String welcomeMessage()
        {
                return "<h1>Welcome!!</h1>";
        }
}
```

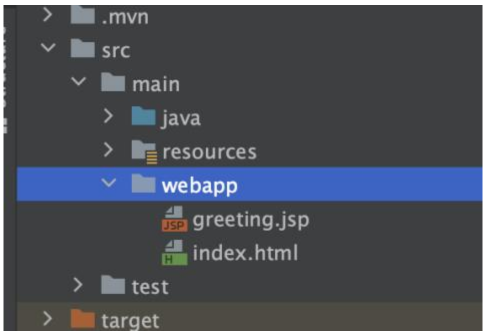
3.2. Restart the server and open browser at
http://localhost:8080/welcome.

It returns HTML heading as a response but writing the entire HTML code as above is messy.

4. Views

4.1. Let's create a view

4.1.1. Create a webapp folder as shown below



4.1.2. Create index.html file [this serves as static content] within the webapp folder as shown above and a heading as follows within the <body> tag
<h1>Welcome to MVC tutorial</h1>

4.1.3. Restart the server and now you should see the above message on the browser at http://localhost:8080/

4.2. Add another view with the name greeting.jsp by saying right click on webapp folder-> New File and add below heading

```
    <%@ page language="java" contentType="text/html; charset=UTF-8"
  pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <h1>Greetings!!</h1>
</body>
</html>
```

4.2.1. To inform spring about the view add below in application.properties file:

```
spring.mvc.view.prefix=/
spring.mvc.view.suffix=.jsp
```

4.3. Add below in GreetController to load the greeting.jsp:

```
// Here is a general annotation RequestMapping for /greet url which can accept requests
// coming for all Http methods viz: GET POST PUT DELETE etc

@RequestMapping(value = "/greet")
      public String greet() {

        // this method should return the name of the view
        // [ IT SHOULD BE EXACT AS CREATED IN WEBAPP FOLDER ]

          return "greeting";
        }




    // To make it specific to GET, add method attribute

    @RequestMapping(value = "/greet", method = RequestMethod.GET)
      public String greet() {
          return "greeting";
        }

    // OR CAN USE GetMapping for /greet url as a shorthand syntax

    /*
    @GetMapping(value = "/greet")
      public String greet() {
          return "greeting";
      }
    */
```

5. Prepare model, repository and service layer

5.1. Copy Employee class created earlier in SpringBoot project.
REMOVE THE REFERNCE TO ADDRESS CLASS
5.2. Copy respective repos for Employee
5.3. Copy EmployeeService class as well with completed code for the CRUD operations
ONLY FOR EMPLOYEE

Add below code in class with main method to create seed data

```
@Autowired
private EmployeeService employeeService;

@Bean
    public void initialize()
    {
            Employee emp = new Employee();
            emp.setEname("Sia");
            emp.setEmail("sia@test.com");
            emp.setPassword("sia123");
            emp.setPhone("9898989898");

            Employee e = employeeService.insertEmployee(emp);

            Employee em = new Employee();
            em.setEname("John");
            em.setEmail("john@test.com");
            em.setPassword("john1235");
            em.setPhone("7654323456");

            Employee e = employeeService.insertEmployee(em);

    }
```

6. Login/ Logout/ Registration/ Welcome Views

6.1. Add below code in index.html that was created earlier

```
<html>
    <head>
    <link
      href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
      rel="stylesheet"
      integrity="sha384-
    EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"
      crossorigin="anonymous">
    </head>
    <body>
      <nav class="navbar navbar-expand-lg navbar-light bg-light">
            <div class="container-fluid">
                    <a class="navbar-brand" href="#">NorthernTrust</a>
                    <button class="navbar-toggler" type="button"
                            data-bs-toggle="collapse" data-bs-
    target="#navbarSupportedContent"
```

```
                                aria-controls="navbarSupportedContent" aria-expanded="false"
                                aria-label="Toggle navigation">
                                <span class="navbar-toggler-icon"></span>
                        </button>
                        <div class="collapse navbar-collapse" id="navbarSupportedContent">
                                <ul class="navbar-nav me-auto mb-2 mb-lg-0">
                                        <li class="nav-item"><a class="nav-link active"
                                                aria-current="page" href="./">Home</a></li>

                                        <li class="nav-item"><a class="nav-link"
href="login">Login</a>
                                        </li>
                                        <li class="nav-item"><a class="nav-link"
href="customers">Customers</a>
                                        </li>
                                        <li class="nav-item"><a class="nav-link"
href="register">Register</a>
                                        </li>
                                        <li class="nav-item"><a class="nav-link"
href="logout">Logout</a>
                                        </li>
                                </ul>

                        </div>
                </div>
  </nav>
  <div class="container mt-5">
                <div class="row">
                        <div class="col-md-8">
                                <img

  src=https://assets.aboutamazon.com/dims4/default/d13d39a/2147483647/strip/true/crop/
1279x720+0+0/resize/2640x1486!/format/webp/quality/90/?url=https%3A%2F%2Famazon-
blogs-brightspot.s3.amazonaws.com%2Ffb%2F1f%2Fa53279a446ccbf19e4c881fde4f4%2Fecho-
amazon-4-1.jpg class="img-fluid" />
                        </div>
                        <div class="col-md-4">
                                <p style='font-size: 1em'>This is a website that allows you to
                                        Manage employees from home</p>
                                <p style='font-size: 1em'>This is a website that allows you to
                                        Manage employees from home</p>
                                <p style='font-size: 1em'>This is a website that allows you to
                                        Manage employees from home</p>
                        </div>
                </div>
  </div>
</body>
</html>
```

6.2. login.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
        pageEncoding="UTF-8" isELIgnored="false"%>
        <!DOCTYPE html>
        <html>
        <head>
        <meta charset="UTF-8">
```

```
<title>Login</title>
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"
		crossorigin="anonymous">
		<style type="text/css">
		.error{
				color:red;
		}
		</style>
</head>
<body>
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">NorthernTrust</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-
target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-
expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarSupportedContent">
      <ul class="navbar-nav me-auto mb-2 mb-lg-0">
        <li class="nav-item">
          <a class="nav-link active" aria-current="page" href="./">Home</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="login">Login</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="register">Register</a>
        </li>
      </ul>

    </div>
  </div>
</nav>
		<div class="container">

				<h1>Login</h1>
				<div class="error">${error}</div>
				<form action="login" method="POST">
				<div class="mb-3">
						<label for="formGroupExampleInput2" class="form-label">Email
								</label> <input type="email" class="form-control"
								name="email" value=" sia@test.com "
								id="formGroupExampleInput2" placeholder="Email">
				</div>
				<div class="mb-3">
						<label for="formGroupExampleInput2" class="form-
label">Password
								</label> <input type="password" class="form-control"
								name="password" value="sia123"
								id="formGroupExampleInput2" placeholder="Password">
				</div>
				<button type="submit" class="btn btn-primary">Login</button>
				</form>
		</div>
```

```
            </body>
            </html>
```

6.3. register.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
            pageEncoding="UTF-8" isELIgnored="false"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Login</title>
<link
            href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
            rel="stylesheet"
            integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"
            crossorigin="anonymous">
</head>
<body>
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">NorthernTrust</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-
target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false"
aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarSupportedContent">
      <ul class="navbar-nav me-auto mb-2 mb-lg-0">
        <li class="nav-item">
          <a class="nav-link active" aria-current="page" href="./">Home</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="login">Login</a>
        </li>
         <li class="nav-item">
          <a class="nav-link" href="regsiter">Register</a>
        </li>
      </ul>

    </div>
  </div>
</nav>
            <div class="container">
                        <h1>Register Customer</h1>
                        <form action="register" method="post">


                                    <div class="mb-3">
                                                <label for="formGroupExampleInput2" class="form-label">Email
                                                </label> <input type="email" class="form-control" name="email"
                                                            id="formGroupExampleInput2" placeholder="Email">
                                    </div>
                                    <div class="mb-3">
                                                <label for="formGroupExampleInput2" class="form-label">Name
```

```
                                        </label> <input type="text" class="form-control" name="ename"
                                                id="formGroupExampleInput2" placeholder="Name">
                                </div>
                                <div class="mb-3">
                                        <label for="formGroupExampleInput2" class="form-label">Phone
                                        </label> <input type="text" class="form-control" name="phone"
                                                id="formGroupExampleInput2" placeholder="Phone">
                                </div>
                                <div class="mb-3">
                                        <label for="formGroupExampleInput2" class="form-
    label">Password
                                        </label> <input type="password" class="form-control"
    name="password"
                                                id="formGroupExampleInput2" placeholder="Password">
                                </div>

                                <button type="submit" class="btn btn-primary">Register</button>
                        </form>
                </div>
    </body>
    </html>
```

6.4. welcome.jsp

```
     <%@ page language="java" contentType="text/html; charset=UTF-8"
  pageEncoding="UTF-8" isELIgnored="false"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
<link
    href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
    rel="stylesheet"
    integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"
    crossorigin="anonymous">
</head>
<body>
<nav class="navbar navbar-expand-lg navbar-light bg-light">
 <div class="container-fluid">
  <a class="navbar-brand" href="#">NorthernTrust</a>
  <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-
target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-
label="Toggle navigation">
   <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="navbarSupportedContent">
   <ul class="navbar-nav me-auto mb-2 mb-lg-0">
    <li class="nav-item">
     <a class="nav-link active" aria-current="page" href="./">Home</a>
    </li>

    <li class="nav-item">
     <a class="nav-link" href="profile">Profile</a>
    </li>
    <li class="nav-item">
     <a class="nav-link" href="logout">Logout</a>
    </li>
   </ul>

  </div>
 </div>
</nav>
<h1>Welcome Employee!!</h1>
</body>
</html>
```

7. Controller

7.1. Create class LoginController adding below methods

```
 @Controller
public class LoginController {

        @Autowired
        private EmployeeService employeeService;

        @GetMapping("/login")
        public String loginPage(HttpServletRequest request)
        {
                System.out.println("login request " + request.getMethod() );
                // login is the name of the view(filename) to be returned to the browser
                return "login";
        }

        @GetMapping("/register")
        public String registerPage(Map<String, List<String>> map)
        {
                List<String> countries = Arrays.asList("India","USA","UK");
                map.put("countries", countries);
                return "register";
        }
        @PostMapping("/register")
        public String registerEmployee(Employee emp)
        {
                System.out.println("register "+emp);
                try {
                        if(this.employeeService.insertEmployee(emp) != null) {
                                System.out.println("if");
                                return "redirect:login";
                        }
                } catch (Exception e) {
                        // TODO Auto-generated catch block
                        System.out.println("error ***********"+e.getMessage());
                        return "redirect:register";
                }
                System.out.println("out ***********");
                return "redirect:register";
        }
        @PostMapping("/login")
        public String loginEmployee(HttpServletRequest request,
                        Map<String, String> errorMap,
                        Employee emp)
	{
		System.out.println("login emp request " + request.getMethod() );
		System.out.println(emp);
		try {
			if(this.employeeService. loginEmployee (emp.getEmail(),
			emp.getPassword())){
				return "redirect:welcome";
			}
		} catch (Exception e) {
			System.out.println(e.getMessage());
			// TODO Auto-generated catch block
			errorMap.put("error",e.getMessage());
		}
		return "login";
	}
}
```

7.2. Create class WelcomeController adding below method

```
@Controller
public class WelcomeController {


	@GetMapping("/welcome")
	public String welcomePage()
	{

		return "welcome";
	}
```

8. Test the http requests

8.1. Restart the server and it opens up the default index.html file

8.2. Click on login and register links

8.3. Register link should add an employee record in h2 database. Open h2 and check if employee was added and you should be redirected to login page

8.4. Login link if successful should redirect to welcome page else should display error message within login.jsp file

9. Session management

9.1. The welcome page should display the employee name / email who logged in. Modify the loginEmployee method as follows to send the email of logged in employee using url rewriting:

```
public String loginEmployee(HttpServletRequest request,
				Map<String, String> errorMap,
				Employee emp)
	{
		System.out.println("login emp request " + request.getMethod() );
		System.out.println(emp);
		try {
```

```
                        if(this.employeeService. loginEmployee (emp.getEmail(),
                        emp.getPassword())){
                                return "redirect:welcome?email="+ emp.getEmail();
                        }
                } catch (Exception e) {
                        System.out.println(e.getMessage());
                        // TODO Auto-generated catch block
                        errorMap.put("error",e.getMessage());
                }
                return "login";
        }
```

9.2. Update the welcome.jsp file to display the employee email as follows:

```
<h1>Welcome ${email }</h1>
```

9.3. It is definitely tedious to keep on sending the logged in user id as part of url rewriting. Instead lets use HttpSession as follows:

9.3.1. Update LoginController class loginEmployee() method as follows:

```
public String loginEmployee(HttpServletRequest request,
                        Map<String, String> errorMap, HttpSession session
                        Employee emp)
  {
                System.out.println("login emp request " + request.getMethod() );
                System.out.println(emp);
                try {
                        if(this.employeeService. loginEmployee (emp.getEmail(),
                        emp.getPassword())){
                        // REMOVE THE URL REWRITINIG AND ADD BELOW CODE
                        session.setAttribute("email",emp.getEmail());
                                return "redirect:welcome;
                        }
                } catch (Exception e) {
                        System.out.println(e.getMessage());
                        // TODO Auto-generated catch block
                        errorMap.put("error",e.getMessage());
                }
                return "login";
  }
```

9.3.2. No changes to be made in welcome.jsp file and restart the server. Should see the email id of logged in user

10. Logout

10.1. To logout add below code in LoginController

```
@GetMapping("/logout")
public String logoutPage(HttpSession session)
{
        session.removeAttribute("email");
        session.invalidate();
        return "redirect:login";
}
```

11. Admin Page

11.1. Create a class AdminController as follows :

```
@Controller
public class AdminController(){

	@GetMapping("/admin")
	public String getAdminPage(Map<String, List<Employee>> map)
	{
		map.put("employees", this.employeeService.getEmployees());
		return "customers";
	}
```