Spring MVC with Bootstrap CSS

Configuration of Bootstrap in Spring Application

- Bootstrap is a *free, popular and open-source CSS framework* directed at responsive, mobile-first front-end web development.
- It contains HTML and CSS based design templates for various interface components such as typography, forms, buttons, modals, image carousels, navigation etc., as well as optional JavaScript extensions.

Features of Bootstrap

- **Browser Support**:- Bootstrap supports all popular web browsers such as Chrome, Safari, Firefox, Opera, and Internet edge.
- **Responsive Design**:- Bootstrap enables us to write responsive websites very easily. Its responsive CSS adjusts automatically to Desktops, Tablets and Mobiles.
- **Uniform solution to build interface**: Bootstrap provides a clean and uniform solution to build an interface for the developers.
- **Customization:** Bootstrap provides a simple and easy way to customize the CSS.
- **Functional built-in components**: Bootstrap contains beautiful and functional built-in components which can be customized very easily.

Different ways to configure bootstrap

1. Bootstrap through Content Delivery Network(CDN)

- CSS: Copy-paste the below Stylesheet link to the `<head>` tag of your desired HTML/JSP file.

```
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet"
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"crossorigin="anonymous">
```

- JS: Place the following `<script>` s near the end of your pages, right before the closing `</body>` tag.

```
<script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.2/dist/umd/popper.min.js" integrity="sha3

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.min.js" integrity="sha384-
```

2. Custom configuration of Bootstrap with Spring MVC (Offline by downloading files locally)

- Download the bootstrap css and js files from Bootstrap website.
- Put static resources CSS and JS files into `webapp/resources/static` directory. To segregate CSS and JS, keep them into respective folders `webapp/resources/static/css` and `webapp/resources/static/js` .
- Create the resource mapping into Spring using either **XML configuration** or **Java Configuration**
  - **XML configuration**

    ```
    <mvc:resources mapping="/js/**" location="/resources/static/js/" />
    <mvc:resources mapping="/css/**" location="/resources/static/j=css/" />
    ```

  - **Java Configuration**

    ```
    @Configuration
    @EnableWebMvc
    public class MyApplication implements WebMvcConfigurer
    {
        public void addResourceHandlers(final ResourceHandlerRegistry registry)
        {
            registry.addResourceHandler("/js/**").addResourceLocations("/resources/static/js/");
            registry.addResourceHandler("/css/**").addResourceLocations("/resources/static/css/");
        }
    }
    ```

- Include the Bootstrap CSS and JS into JSP page via **JSTL tag *c:url*** or **Spring tag *spring:url***.
  - **JSTL tag *c:url***

```
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<!DOCTYPE html>
<html lang="en">
    <head>
        <link href="<c:url value="/css/bootstrap.css" />" rel="stylesheet">
        <script src="<c:url value="/js/jquery.1.10.2.min.js" />"></script>
        <script src="<c:url value="/js/bootstrap.js" />"></script>
    </head>
    <body>
    </body>
</html>
```

- **Spring tag *spring:url***

```
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags"%>
<!DOCTYPE html>
<html lang="en">
    <head>
        <spring:url value="/css/bootstrap.css" var="bootstrapCss" />
        <spring:url value="/js/jquery.1.10.2.min.js" var="jqueryJs" />
        <spring:url value="/js/bootstrap.js" var="bootstrapJs" />
        <link href="${bootstrapCss}" rel="stylesheet" />
        <script src="${jqueryJs}"></script>
        <script src="${bootstrapJs}"></script>
    </head>
    <body>
    </body>
</html>
```

Apply custom CSS in Pages

- **Css for just one page**

```
<head>
    <style type="text/css">
        p {
            color: green;
        }
    </style>
</head>
<body>
    <p>
    This is Green Color Text.
    </p>
    <p>
    This is another paragraph with a green color.
    </p>
</body>
```

- **custom css**

```
<link rel="stylesheet" type="text/css" href="css/custom.css">
```

SETTING UP DATABASE USING HIBERNATE(Database configuration using Hibernate)

- Spring supports bootstrapping the Hibernate SessionFactory in order to set up the database using Hibernate. The database setup can be done with a few lines of Java code or XML configuration.
- Spring provides two key beans to support the Hibernate integration, available in the **org.springframework.orm.hibernate5** package:
    - ***LocalSessionFactoryBean***: Creates a Hibernate's SessionFactory which is injected into Hibernate-based DAO classes.
    - ***PlatformTransactionManager***: Provides transaction support code for a SessionFactory. Programmers can use **@Transactional** annotation in DAO methods to avoid writing boiler-plate transaction code explicitly.

```
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-orm</artifactId>
    <version>5.1.0.RELEASE</version>
 </dependency>

 <!--Hibernate -->
 <dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>5.3.5.Final</version>
 </dependency>

 <!-- MySQL -->
 <dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.16</version>
 </dependency>
```

```
packagecom.ignou.mvcapp;
importjava.util.Properties;
importjavax.sql.DataSource;
importorg.springframework.beans.factory.annotation.Autowired;
importorg.springframework.context.annotation.Bean;
importorg.springframework.context.annotation.Configuration;
importorg.springframework.context.annotation.PropertySource;
importorg.springframework.core.env.Environment;
importorg.springframework.jdbc.datasource.DriverManagerDataSource;
importorg.springframework.orm.hibernate5.HibernateTransactionManager;
importorg.springframework.orm.hibernate5.LocalSessionFactoryBean;
importorg.springframework.transaction.PlatformTransactionManager;
importorg.springframework.transaction.annotation.EnableTransactionManagement;

@Configuration
@EnableTransactionManagement
@PropertySource(value = { "classpath:application.properties" })
```

```
publicclassHibernateConfig
{
    @Autowired
    private Environment environment;
    @Bean
    publicLocalSessionFactoryBeansessionFactory()
    {
        LocalSessionFactoryBeansessionFactory = newLocalSessionFactoryBean();
        sessionFactory.setDataSource(dataSource());
        sessionFactory.setPackagesToScan(newString[] { "com.ignou.mvcapp.model" });
        sessionFactory.setHibernateProperties(hibernateProperties());
        returnsessionFactory;
    }

    @Bean
    publicDataSourcedataSource()
    {
        DriverManagerDataSourcedataSource = newDriverManagerDataSource();
        dataSource.setDriverClassName(environment.getRequiredProperty("jdbc.driverClassName"));
        dataSource.setUrl(environment.getRequiredProperty("jdbc.url"));
        dataSource.setUsername(environment.getRequiredProperty("jdbc.username"));
        dataSource.setPassword(environment.getRequiredProperty("jdbc.password"));
        returndataSource;
    }
    private Properties hibernateProperties()
    {
        Properties properties = newProperties();
        properties.put("hibernate.dialect", environment.getProperty("hibernate.dialect"));
        properties.put("hibernate.show_sql", environment.getProperty("hibernate.show_sql"));
        properties.put("hibernate.format_sql", environment.getProperty("hibernate.format_sql"));
        properties.put("hibernate.hbm2ddl.auto", environment.getProperty("hibernate.hbm2ddl.auto"));
        returnproperties;
    }
    @Bean
    publicPlatformTransactionManagergetTransactionManager()
    {
        HibernateTransactionManagerhtm = newHibernateTransactionManager();
        htm.setSessionFactory(sessionFactory().getObject());
        return htm;
    }
}
```

Create, Read, Update, Delete (CRUD)

- The acronym CRUD stands for CREATE, READ, UPDATE and DELETE. The CRUD paradigm is common in constructing web applications. While constructing the APIs, the model should provide four basic types of functionality such as Create, Read, Update and Delete the resources. CRUD operations are also often used with SQL.

EXample of CRUD

- **Create**: A new student is entered into the database.
- **Read**: The student's information is displayed to the users.
- **Update**: Already, existing student’s attributes are being updated with new values.

- **Delete**: If the student is not part of the institute, the student can be deleted from the records.

Mapping CRUD to SQL statements

- This table provides examples of SQL statements for performing Create, Read, Update, and Delete (CRUD) operations on a `student` table.

Function	Operation	SQL Statement
Create	Insert	`INSERT INTO student (name, grade) VALUES ('Prasoon', 10);`
Read	Select	`SELECT * FROM student;`
Update	Update	`UPDATE student SET grade = 10 WHERE id = 1;`
Delete	Delete	`DELETE FROM student WHERE id = 1;`

Mapping CRUD to REST

- In REST context, CRUD corresponds to Rest web service POST, GET, PUT and DELETE respectively. The following table maps CRUD to REST with example.

Function	Operation	HTTP Method	Endpoint URL
Create	POST	`POST`	`http://teststudent.com/students/`
Read	GET	`GET`	`http://teststudent.com/students/`
Update	PUT	`PUT`	`http://teststudent.com/students/1`
Delete	DELETE	`DELETE`	`http://teststudent.com/students/1`

CREATE

- To create a resource in a REST environment, HTTP Post method is used. Post method creates a new resource of the specified resource type. To create a new student record for the Student Management System, HTTP POST request and response are shown below.
    - **REQUEST**:POST `http://teststudent.com/students/`
    - **REQUEST BODY**:

```
{
"firstName": "Anurag",
"lastName": "Singh",
"grade": 10
}
```

    - **RESPONSE:** Status Code – 201(Created)

```
{
    "id": 1,
    "firstName": "Anurag",
    "lastName": "Singh",
    "grade": 10
}
```

READ

- To read resources in a REST environment, HTTP GET method is used. Read operation should not change the resource. The Get method returns the same response irrespective of number of times of execution.
- HTTP GET method is **idempotent**. A HTTP method is called idempotent if an identical request can be made once or several times in a row with the same effect while leaving the server in the same state. To retrieve all student records from the Student Management System, HTTP GET method request and response is shown below.
  - **REQUEST**: GET `http://teststudent.com/students/`
  - **RESPONSE**: Status Code – 200 (OK)

```
[
    {"id": 1, "firstName": "Anurag", "lastName": "Singh", "grade": 10},
    {"id": 1, "firstName": "Dinesh", "lastName": "Chaurasia", "grade": 10}
]
```

UPDATE

- To update a resource in a REST environment, HTTP PUT or PATCH method is used.
- This operation is **idempotent**. Request and response to update the grade of student id 1, is shown below.
  - **REQUEST**: PUT `http://teststudent.com/students/1`
  - **REQUEST BODY**:

```
{
    "id": 1,
    "firstName": "Anurag",
    "lastName": "Singh",
    "grade": 8
}
```

  - **RESPONSE**:Status Code – 200 (OK)
    - The response includes a Status Code of 200 (OK) to signify that the operation was successful, but it need not return a response body.

DELETE

- To delete a resource in a REST environment, HTTP DELETE method is used. It is used to remove a resource from the system. This operation is also **idempotent**. Request and response to delete a student with id 1, is shown below.
  - **REQUEST**:DELETE `http://teststudent.com/students/1`

- **RESPONSE**:Status Code – 204 (No Content). Successful delete operation returns a response code of 204 (NO CONTENT), with no response body.

CRUD Examples in Spring MVC and Hibernate

Check your progress - 1

1. What is Bootstrap and what factors make Bootstrap a popular choice to use in web applications?
2. What are the different ways to configure Bootstrap in Spring MVC?
3. What are the obvious reasons to use Bootstrap CDN to configure it with Spring?

Check your progress - 2

1. What is custom CSS and how can these be included into a JSP?
2. Write the custom CSS to make all the paragraphs of a page to red color. Write the sample JSP page for it.
3. What are the key beans provided by Spring to integrate with Hibernate?

Check your progress - 3

1. Define CRUD within computer programming.
2. Define the mapping of the CRUD operations with SQL statements with suitable examples.
3. Define the mapping of CRUD in the Rest context.