# Instructions on importing DCHC & DSHC:

Prerequisites:

1. Knowledge of how spring boot works.
2. knowledge of Maven.
3. An IDE with Maven configured.

## Instructions on importing spring boot project:

1. **Import the Maven Project:**
   * In STS/Eclipse IDE, go to **File** > **Import**.
   * In the "Import" dialog, select **Existing Maven Projects** under the **Maven** category.
   * Click **Next**.
   * Browse to the location of your Spring Boot project (where the **pom.xml** file is located) and select the project folder.
   * Click **Finish** to import the project.
2. **Wait for Maven to Download Dependencies:**
   * STS will automatically start downloading the necessary dependencies specified in your **pom.xml** file. This may take some time, depending on your internet connection.
3. **Set up the Run Configuration:**
   * After the project is imported, right-click on the project in the "Project Explorer" pane.
   * Select **Run As** > **Spring Boot App**.
   * STS will automatically create a run configuration for your Spring Boot application.
4. **Run the Application:**
   * Right-click on your project again.
   * Select **Run As** > **Spring Boot App**.
   * Alternatively, you can run it using the "Run" button in the toolbar.
5. **Access the Application:**
   * Once the application is successfully started, you can access it by opening a web browser and navigating to **http://localhost:8080** (or another port if configured differently).

## Information on Spring boot controllers and Thyme leaf templates:

### Controllers:

Controllers in Spring Boot handle incoming requests and define the flow of the application. They are responsible for processing user input, interacting with the business logic (services), and returning an appropriate response, often in the form of a view. Here's a simple example of a controller:

@Controller

public class MyController {

@GetMapping("/hello")

public String hello(Model model) {

model.addAttribute("message", "Hello, World!");

return "hello"; // This corresponds to the Thymeleaf template name

}

}

In this example, the hello method is mapped to the URL path "/hello". It adds a message attribute to the model, and then returns the name of a Thymeleaf template ("hello").

### Services:

Services in Spring Boot contain the business logic of your application. They encapsulate functionality that is not directly related to handling HTTP requests. For example:

@Service

public class MyService {

public String generateMessage() {

return "This message is generated by the service.";

}

}

### Thymeleaf Templates:

Thymeleaf is a templating engine used for server-side rendering in Spring Boot applications. It allows you to create dynamic HTML templates that can be rendered on the server side before being sent to the client. Here's a simple Thymeleaf template (hello.html) that corresponds to the controller example above:

html

Copy code

<!DOCTYPE html>

<html lang="en" xmlns:th="http://www.thymeleaf.org">

<head>

<meta charset="UTF-8">

<title>Hello Page</title>

</head>

<body>

<h1 th:text="${message}"></h1>

</body>

</html>

In this template, ${message} is a Thymeleaf expression that evaluates to the value of the "message" attribute added to the model in the controller.

### Connecting Controller, Service, and Template:

The controller can use the service to fetch data or perform business logic and then pass the results to the Thymeleaf template for rendering. Here's an updated controller example:

@Controller

public class MyController {

private final MyService myService;

public MyController(MyService myService) {

this.myService = myService;

}

@GetMapping("/hello")

public String hello(Model model) {

String message = myService.generateMessage();

model.addAttribute("message", message);

return "hello";

}

}

In this example, the MyController class has a dependency on MyService, and it uses this service to generate the message for the Thymeleaf template.