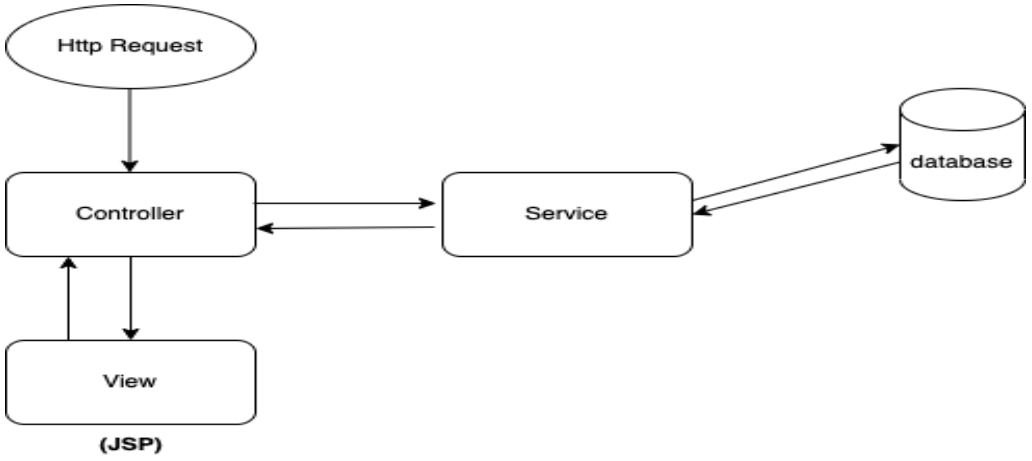
Java Server Pages(JSP)

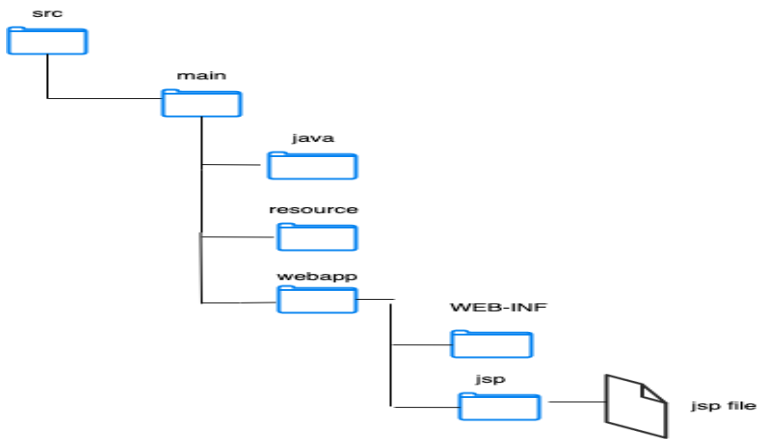
What Is JSP?

JSP (Java Server Pages) is a technology that allows developers to create dynamic web pages. It uses basic HTML tags to display and format the content. It will enable users to use specific JSP tags for Java code inside the HTML.

In a Spring Boot application, JSPs can be used as the view layer to display the content or take input from the user. For example, it can display data from a database or other sources or create forms allowing users to submit data to the server for processing.



Directory Structure of the JSP file:



Code Example:

- Jsp files act the same as HTML pages. You can write your HTML code as you want to do the task. An example is given below for the same:

```
<html>
<div>
   <h1>
       <a href="/signup">Sign me up</a>
   </h1>
</div>
</html>
```

- Here is another example of a ThankYou page in JSP:

```
<html>
<div>
   <h1>Thank You</h1>
</div>
</html>
```

To summarise, **JSP (JavaServer Pages)** is a technology that creates dynamic web pages in Java. It allows developers to embed Java code within HTML markup, enabling dynamic content generation. To use JSP in a Java web application, you must include the necessary dependencies, such as "**tomcat-embed-jasper**" and "**JSTL**", in your project configuration.

These dependencies provide the JSP rendering capabilities and the embedded Tomcat server. With JSP and the required dependencies, you can create JSP files, combine HTML with embedded Java code, and deploy the application to run the JSP pages on a server. Now let's look at the necessary dependencies required to use JSP.

Dependencies for JSP In Spring Boot

A. Jasper

The "**Tomcat Embed Jasper**" Spring Boot dependency supports JSP (JavaServer Pages) rendering within a Spring Boot application. It allows you to use JSP alongside other Spring Boot features as a view technology.

Here is a brief description of the relevant dependencies:

- **Tomcat-embed-jasper** is the main dependency providing JSP rendering capabilities within a Spring Boot application. It embeds the Tomcat Jasper JSP engine responsible for compiling and rendering JSP pages.

- **Tomcat-embed-core**: This dependency embeds the Apache Tomcat servlet container, Spring Boot's default container. It provides the necessary runtime environment for running JSP pages.

To use the "**Tomcat Embed Jasper**" dependency in your Spring Boot application, you need to include the following Maven dependency in your project configuration:

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

You may also need to configure the *"**application.yaml**"* file in your Spring Boot application to specify the JSP-related properties, such as the prefix and suffix for your JSP views. Here's an example:

```
spring:
  mvc:
    view:
      prefix: /WEB-INF/views/
      suffix: .jsp
```

You can include this application.yml file in your Spring Boot project's classpath, and Spring Boot will automatically read and apply these configuration properties when running your application.

B. JSTL

JSTL (JavaServer Pages Standard Tag Library) is a standard tag library that provides a set of tags for JavaServer Pages (JSP) to simplify the development of dynamic web applications. It is an integral part of the Java EE (Enterprise Edition) specification and is widely used in web applications built using JSP technology.

The JSTL library consists of several tag libraries, each serving a specific purpose. Here are some of the main JSTL tag libraries:

- **Core Tag Library**: This tag library provides tags for control structures (conditionals and loops), variable manipulation, and internationalisation support. It includes tags such as <c:if>, <c:forEach>, <c:set>, <c:choose>, etc.

- **Formatting Tag Library**: This library offers tags for formatting and presenting data. It includes labels such as <fmt: message> for retrieving localised messages, <fmt:formatNumber> for formatting numbers, <fmt:formatDate> for formatting dates, and more.

- **XML Tag Library:** This tag library provides tags for XML processing, allowing you to parse and transform XML data using JSTL. It includes tags such as <x:parse> for parsing XML, <x:forEach> for iterating over XML elements, and <x:transform> for applying XSLT transformations.

- **SQL Tag Library**: This tag library enables database access within JSP using SQL. It includes tags such as <sql:setDataSource> for defining a connection to a database, <sql:query> for executing SQL queries, <sql:update> for running SQL updates and more.

To use JSTL in your web application, include the JSTL library in your project's classpath and import the appropriate JSTL tag libraries into your JSP pages. You can typically include the JSTL library by adding the following Maven dependency:

```
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>jstl</artifactId>
    <version>1.2</version>
</dependency>
```

Once you have included the JSTL library, you can import the desired JSTL tag libraries in your JSP pages using the <%@ taglib %> directive. For example, to import the Core Tag Library, you would use the following directive:

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
```

JSTL simplifies JSP development by abstracting everyday tasks into reusable tags, promoting separation of concerns and reducing the amount of Java code within JSP pages. It provides a standardised and consistent way to accomplish tasks in JSP-based web applications.

What is DevTools?

Spring Boot DevTools is a development-time module that provides several features to enhance the development experience when working with Spring Boot applications.

Here are some key features and functionalities provided by Spring Boot DevTools:

- **Automatic Restart**: DevTools monitors the classpath for changes and automatically restarts the Spring Boot application whenever a change is detected, eliminating the need to manually stop and start the application during development, significantly reducing turnaround time.

- **Hot Reloading**: DevTools supports hot reloading of specific resources, such as templates and static files, in addition to automatic restarts. Changes to these resources are immediately reflected in the running application without requiring a complete restart, allowing for faster iterative development.

- **Remote-Live Reload**: DevTools includes a remote live reload feature that allows you to trigger a browser refresh whenever changes are made to client-side resources. This feature is handy when working on front-end code and enables a seamless development experience between the backend and front end.

- **Developer-Friendly Error Page**: DevTools replaces the default Spring Boot error page with a more detailed and developer-friendly error page. It provides helpful information about the error, including the stack trace, request details, and application configuration.

- **Additional Developer Tools**: DevTools offers various tools and utilities, such as enhanced logging output, auto-configuration report, and property defaults report. These tools provide insights into the application's configuration and help diagnose potential issues during development.

To use Spring Boot DevTools in your Spring Boot project, you need to include the following Maven dependency in your project configuration:

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <version>3.0.1</version>
</dependency>
```

References:

1. JSP
2. JSTL
3. JSTL II
4. Devtools

Note: We will explore them in detail in the upcoming lecture videos.