Spring MVC

要在沒有 **Spring Boot** 的情況下使用 **Spring MVC**，你需要手動配置 Spring 框架，因為 Spring Boot 通常會自動化大部分的設置。以下是使用基於 XML 的配置創建簡單 Spring MVC Web 應用程序的分步指南。這個示例將設置一個基本應用程序，其中包含一個顯示“歡迎”消息的控制器，並在 JSP 頁面上顯示。

分步指南

1. 設置項目結構

創建標準的 Web 應用程序結構。你可以使用像 IntelliJ 或 Eclipse 這樣的 IDE，或者手動創建文件夾。結構應該如下所示：

```
SimpleSpringMVCApp/
  src/
     main/
         java/
             com/
                 example/
                     controllers/
                         HomeController.java
         webapp/
             WEB-INF/
                 web.xml
                 spring-mvc-config.xml
                 views/
                     home.jsp
  pom.xml （如果使用 Maven）
```

- **`src/main/java`**：包含你的 Java 源代碼（例如，控制器）。
- **`src/main/webapp/WEB-INF`**：包含配置文件（`web.xml`，`spring-mvc-config.xml`）和 JSP 視圖。

2. 添加依賴

如果你使用 Maven，在 `pom.xml` 中包含所需的依賴。對於簡單的 Spring MVC 應用程序，你需要 Spring Web MVC 庫和 Servlet API（由容器提供）。

創建或編輯 `pom.xml`，內容如下：

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.example</groupId>
    <artifactId>SimpleSpringMVCApp</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>war</packaging>

    <dependencies>
        <!-- Spring Web MVC -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-webmvc</artifactId>
            <version>5.3.10</version>
        </dependency>
        <!-- Servlet API (provided by the container) -->
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>javax.servlet-api</artifactId>
            <version>4.0.1</version>
            <scope>provided</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-war-plugin</artifactId>
                <version>3.3.1</version>
            </plugin>
        </plugins>
    </build>
</project>
```

- **備註**：
  - `<packaging>war</packaging>`：確保項目被打包為 WAR 文件以部署到 Servlet 容器。
  - 如果你不使用 Maven，手動下載 Spring MVC JAR 和 Servlet API JAR，並將其添加到項目的類路徑。

3. 在 `web.xml` 中配置 DispatcherServlet

`web.xml` 文件是你的 Web 應用程序的部署描述符。它配置 `DispatcherServlet`，Spring MVC 的前端控制器，來處理傳入的請求。

創建 `src/main/webapp/WEB-INF/web.xml`，內容如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0
         version="4.0">

    <!-- 定義 DispatcherServlet -->
    <servlet>
        <servlet-name>spring-mvc</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <init-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>/WEB-INF/spring-mvc-config.xml</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>

    <!-- 將 DispatcherServlet 映射到處理所有請求 -->
    <servlet-mapping>
        <servlet-name>spring-mvc</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>
</web-app>
```

- **說明**：
  - `<servlet-class>`：指定 `DispatcherServlet`，它將請求路由到控制器。
  - `<init-param>`：指向 Spring 配置文件（`spring-mvc-config.xml`）。
  - `<url-pattern>/</url-pattern>`：將 Servlet 映射到處理應用程序的所有請求。

4. 創建 Spring 配置文件

創建 `src/main/webapp/WEB-INF/spring-mvc-config.xml` 來定義 Spring MVC 豆，例如控制器和視圖解析器。

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
```

```
        xmlns:context="http://www.springframework.org/schema/context"
        xmlns:mvc="http://www.springframework.org/schema/mvc"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="
            http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-bean
            http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-
            http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc.xsd">

    <!-- 啟用控制器的組件掃描 -->
    <context:component-scan base-package="com.example.controllers" />

    <!-- 啟用基於註解的 MVC -->
    <mvc:annotation-driven />

    <!-- 配置 JSP 文件的視圖解析器 -->
    <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix" value="/WEB-INF/views/" />
        <property name="suffix" value=".jsp" />
    </bean>
</beans>
```

- **說明**：
  - `<context:component-scan>`：掃描 `com.example.controllers` 包中的註解組件（例如，`@Controller`）。
  - `<mvc:annotation-driven>`：啟用基於註解的 MVC 功能（例如，`@GetMapping`）。
  - `InternalResourceViewResolver`：將視圖名稱映射到 `/WEB-INF/views/` 中的 JSP 文件，並附加 `.jsp` 後綴。

5. 創建簡單控制器

創建控制器來處理 HTTP 請求。在 `src/main/java/com/example/controllers/` 中添加 `HomeController.java`：

```
package com.example.controllers;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class HomeController {

    @GetMapping("/")
```

```
    public String home() {
        return "home";
    }
}
```

- **說明**：
  - `@Controller`：將此類標記為 Spring MVC 控制器。
  - `@GetMapping("/")`：將 GET 請求映射到根 URL（`/`）到 `home()` 方法。
  - `return "home"`：返回視圖名稱 `"home"`，這將解析為 `/WEB-INF/views/home.jsp`。

6. 創建 JSP 視圖

創建簡單的 JSP 文件來顯示輸出。在 `src/main/webapp/WEB-INF/views/` 中添加 `home.jsp`：

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Home</title>
</head>
<body>
    <h1> 歡迎來到 Spring MVC 而不使用 Spring Boot</h1>
</body>
</html>
```

7. 构建和打包應用程序

如果使用 Maven，從項目根目錄運行以下命令來構建 WAR 文件：

```
mvn clean package
```

這將在 `target` 目錄中生成 `SimpleSpringMVCApp-1.0-SNAPSHOT.war`。

- **備註**：如果不使用 Maven，手動編譯 Java 文件並將項目打包為 WAR 文件，保持文件夾結構。

8. 部署應用程序

將 WAR 文件部署到像 Apache Tomcat 這樣的 Servlet 容器：- 將 `SimpleSpringMVCApp-1.0-SNAPSHOT.war` 複製到 Tomcat 安裝的 `webapps` 目錄。- 啟動 Tomcat（例如，從 Tomcat 的 `bin` 目錄運行 `startup.sh` 或 `startup.bat`）。

9. 訪問應用程序

打開瀏覽器並導航到：

```
http://localhost:8080/SimpleSpringMVCApp-1.0-SNAPSHOT/
```

你應該會看到：

```
歡迎來到 Spring MVC 而不使用 Spring Boot
```

- **備註**：上下文路徑（`/SimpleSpringMVCApp-1.0-SNAPSHOT`）取決於 WAR 文件名稱。如果將 WAR 重命名為 `myapp.war`，URL 將變為 `http://localhost:8080/myapp/`。

摘要

要在沒有 Spring Boot 的情況下使用 Spring MVC：1. **設置 Web 項目** 並使用標準結構。2. **添加依賴** 以獲取 Spring MVC 和 Servlet API。3. **在 `web.xml` 中配置 DispatcherServlet**。4. **在 XML 文件中定義 Spring 配置**（例如，`spring-mvc-config.xml`）。5. **創建控制器** 來處理請求。6. **添加 JSP 視圖** 以顯示響應。7. **構建和部署** WAR 文件到 Servlet 容器。

這種方法給你對配置的完全控制，是學習 Spring MVC 基礎的絕佳方法！