Spring MVC

Pour utiliser **Spring MVC** sans **Spring Boot**, vous devez configurer manuellement le framework Spring, car Spring Boot automatise généralement une grande partie de la configuration. Voici un guide étape par étape pour créer une application web Spring MVC simple en utilisant une configuration basée sur XML. Cet exemple configurera une application de base avec un seul contrôleur qui affiche un message de "Bienvenue" sur une page JSP.

Guide étape par étape

1. Configurer la structure du projet

Créez une structure d'application web standard. Vous pouvez utiliser un IDE comme IntelliJ ou Eclipse, ou créer manuellement les dossiers. La structure devrait ressembler à ceci :

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

- **`src/main/java`** : Contient votre code source Java (par exemple, les contrôleurs).
- **`src/main/webapp/WEB-INF`** : Contient les fichiers de configuration (`web.xml`, `spring-mvc-config.xml`) et les vues JSP.

2. Ajouter les dépendances

Si vous utilisez Maven, incluez les dépendances requises dans votre `pom.xml`. Pour une application Spring MVC simple, vous avez besoin de la bibliothèque Spring Web MVC et de l'API Servlet (fournie par le conteneur).

Créez ou modifiez `pom.xml` avec le contenu suivant :

```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 (fournie par le conteneur) -->
        <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>
```

- **Notes** :
  - `<packaging>war</packaging>` : Assure que le projet est empaqueté en un fichier WAR pour le déploiement sur un conteneur de servlets.

- Si vous n'utilisez pas Maven, téléchargez manuellement les JARs Spring MVC et les JARs Servlet API et ajoutez-les au classpath de votre projet.

3. Configurer le DispatcherServlet dans `web.xml`

Le fichier `web.xml` est le descripteur de déploiement de votre application web. Il configure le `DispatcherServlet`, le contrôleur frontal de Spring MVC, pour gérer les requêtes entrantes.

Créez `src/main/webapp/WEB-INF/web.xml` avec le contenu suivant :

```xml
<?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">

    <!-- Définir le 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>

    <!-- Mapper le DispatcherServlet pour gérer toutes les requêtes -->
    <servlet-mapping>
        <servlet-name>spring-mvc</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>
</web-app>
```

- **Explication** :
  - `<servlet-class>` : Spécifie le `DispatcherServlet`, qui route les requêtes vers les contrôleurs.
  - `<init-param>` : Pointe vers le fichier de configuration Spring (`spring-mvc-config.xml`).
  - `<url-pattern>/</url-pattern>` : Mappe le servlet pour gérer toutes les requêtes vers l'application.

4. Créer le fichier de configuration Spring

Créez `src/main/webapp/WEB-INF/spring-mvc-config.xml` pour définir les beans Spring MVC, tels que les contrôleurs et les résolveurs de vues.

```
<?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"

    <!-- Activer le scan de composants pour les contrôleurs -->
    <context:component-scan base-package="com.example.controllers" />

    <!-- Activer le MVC basé sur les annotations -->
    <mvc:annotation-driven />

    <!-- Configurer le résolveur de vues pour les fichiers JSP -->
    <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix" value="/WEB-INF/views/" />
        <property name="suffix" value=".jsp" />
    </bean>
</beans>
```

- **Explication** :
  - `<context:component-scan>` : Scanne le package `com.example.controllers` pour les composants annotés (par exemple, `@Controller`).
  - `<mvc:annotation-driven>` : Active les fonctionnalités MVC basées sur les annotations (par exemple, `@GetMapping`).
  - `InternalResourceViewResolver` : Mappe les noms de vues aux fichiers JSP dans `/WEB-INF/views/` avec un suffixe `.jsp`.

5. Créer un contrôleur simple

Créez un contrôleur pour gérer les requêtes HTTP. Ajoutez `HomeController.java` dans `src/main/java/com/example/contr`

```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";
    }
}
```

- **Explication** :
  - `@Controller` : Marque cette classe comme un contrôleur Spring MVC.
  - `@GetMapping("/")` : Mappe les requêtes GET à l'URL racine (`/`) à la méthode `home()`.
  - `return "home"` : Retourne le nom de la vue `"home"`, qui se résout en `/WEB-INF/views/home.jsp`.

6. Créer une vue JSP

Créez un fichier JSP simple pour afficher la sortie. Ajoutez `home.jsp` dans `src/main/webapp/WEB-INF/views/`:

```jsp
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Accueil</title>
</head>
<body>
    <h1>Bienvenue sur Spring MVC sans Spring Boot</h1>
</body>
</html>
```

7. Construire et empaqueter l'application

Si vous utilisez Maven, exécutez la commande suivante à partir de la racine du projet pour construire le fichier WAR :

```
mvn clean package
```

Cela génère `SimpleSpringMVCApp-1.0-SNAPSHOT.war` dans le répertoire `target`.

- **Note** : Si vous n'utilisez pas Maven, compilez les fichiers Java et empaquetez le projet en un fichier WAR manuellement, en maintenant la structure des dossiers.

8. Déployer l'application

Déployez le fichier WAR sur un conteneur de servlets comme Apache Tomcat : - Copiez `SimpleSpringMVCApp-1.0-SNAPSH` dans le répertoire `webapps` de votre installation Tomcat. - Démarrez Tomcat (par exemple, exécutez `startup.sh` ou `startup.bat` à partir du répertoire `bin` de Tomcat).

9. Accéder à l'application

Ouvrez un navigateur et accédez à :

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

Vous devriez voir :

```
Bienvenue sur Spring MVC sans Spring Boot
```

- **Note** : Le chemin de contexte (`/SimpleSpringMVCApp-1.0-SNAPSHOT`) dépend du nom du fichier WAR. Si vous renommez le WAR en `myapp.war`, l'URL devient `http://localhost:8080/myapp/`.

Résumé

Pour utiliser **Spring MVC sans Spring Boot** : 1. **Configurer un projet web** avec une structure standard. 2. **Ajouter les dépendances** pour Spring MVC et l'API Servlet. 3. **Configurer le DispatcherServlet** dans `web.xml`. 4. **Définir la configuration Spring** dans un fichier XML (par exemple, `spring-mvc-config.xml`). 5. **Créer un contrôleur** pour gérer les requêtes. 6. **Ajouter une vue JSP** pour la réponse. 7. **Construire et déployer** le fichier WAR sur un conteneur de servlets.

Cette approche vous donne un contrôle total sur la configuration, ce qui en fait une excellente manière d'apprendre les fondamentaux de Spring MVC !