Spring MVC

Um **Spring MVC** ohne **Spring Boot** zu verwenden, müssen Sie die Spring-Framework manuell konfigurieren, da Spring Boot normalerweise viel der Einrichtung automatisiert. Hier ist eine Schritt-für-Schritt-Anleitung zur Erstellung einer einfachen Spring MVC-Webanwendung mit XML-basierter Konfiguration. Dieses Beispiel richtet eine grundlegende Anwendung mit einem einzigen Controller ein, der eine "Willkommen"-Nachricht auf einer JSP-Seite anzeigt.

Schritt-für-Schritt-Anleitung

1. Projektstruktur einrichten

Erstellen Sie eine Standard-Webanwendungsstruktur. Sie können eine IDE wie IntelliJ oder Eclipse verwenden oder die Ordner manuell erstellen. Die Struktur sollte wie folgt aussehen:

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

- `src/main/java`: Enthält Ihren Java-Quellcode (z.B. Controller).
- `src/main/webapp/WEB-INF`: Enthält Konfigurationsdateien (`web.xml`, `spring-mvc-config.xml`) und JSP-Ansichten.

2. Abhängigkeiten hinzufügen

Wenn Sie Maven verwenden, fügen Sie die erforderlichen Abhängigkeiten in Ihrer `pom.xml` hinzu. Für eine einfache Spring MVC-Anwendung benötigen Sie die Spring Web MVC-Bibliothek und die Servlet-API (vom Container bereitgestellt).

Erstellen oder bearbeiten Sie `pom.xml` mit folgendem Inhalt:

```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 (vom Container bereitgestellt) -->
        <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>
```

- **Hinweise**:
  - `<packaging>war</packaging>`: Stellt sicher, dass das Projekt als WAR-Datei für die Bereitstellung in einem Servlet-Container gepackt wird.

- Wenn Sie Maven nicht verwenden, laden Sie die Spring MVC JARs und Servlet API JARs manuell herunter und fügen Sie sie zum Klassenspeicher Ihres Projekts hinzu.

3. DispatcherServlet in `web.xml` konfigurieren

Die `web.xml`-Datei ist der Bereitstellungsdeskriptor für Ihre Webanwendung. Sie konfiguriert den `DispatcherServlet`, den Front-Controller von Spring MVC, um eingehende Anfragen zu verarbeiten.

Erstellen Sie `src/main/webapp/WEB-INF/web.xml` mit folgendem Inhalt:

```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">

    <!-- Definieren Sie den 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>

    <!-- Zuordnung des DispatcherServlet zur Verarbeitung aller Anfragen -->
    <servlet-mapping>
        <servlet-name>spring-mvc</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>
</web-app>
```

- **Erklärung**:
  - `<servlet-class>`: Gibt den `DispatcherServlet` an, der Anfragen an Controller weiterleitet.
  - `<init-param>`: Verweist auf die Spring-Konfigurationsdatei (`spring-mvc-config.xml`).
  - `<url-pattern>/</url-pattern>`: Ordnet den Servlet zur Verarbeitung aller Anfragen an die Anwendung zu.

4. Spring-Konfigurationsdatei erstellen

Erstellen Sie `src/main/webapp/WEB-INF/spring-mvc-config.xml`, um Spring MVC-Beans wie Controller und View-Resolver zu definieren.

```
<?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'

    <!-- Aktivieren Sie die Komponentensuche für Controller -->
    <context:component-scan base-package="com.example.controllers" />

    <!-- Aktivieren Sie die annotationsbasierte MVC -->
    <mvc:annotation-driven />

    <!-- Konfigurieren Sie den View-Resolver für JSP-Dateien -->
    <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix" value="/WEB-INF/views/" />
        <property name="suffix" value=".jsp" />
    </bean>
</beans>
```

- **Erklärung**:
  - `<context:component-scan>`: Scannt das `com.example.controllers`-Paket nach annotierten Komponenten (z.B. `@Controller`).
  - `<mvc:annotation-driven>`: Aktiviert annotationsbasierte MVC-Funktionen (z.B. `@GetMapping`).
  - `InternalResourceViewResolver`: Ordnet View-Namen JSP-Dateien in `/WEB-INF/views/` mit einer `.jsp`-Endung zu.

5. Einfacher Controller erstellen

Erstellen Sie einen Controller zur Verarbeitung von HTTP-Anfragen. Fügen Sie `HomeController.java` in `src/main/java/com/example/controllers/` hinzu:

```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";
    }
}
```

- **Erklärung**:
  - `@Controller`: Markiert diese Klasse als Spring MVC-Controller.
  - `@GetMapping("/")`: Ordnet GET-Anfragen an die Stamm-URL (`/`) der `home()`-Methode zu.
  - `return "home"`: Gibt den View-Namen `"home"` zurück, der auf `/WEB-INF/views/home.jsp` aufgelöst wird.

6. JSP-Ansicht erstellen

Erstellen Sie eine einfache JSP-Datei zur Anzeige der Ausgabe. Fügen Sie `home.jsp` in `src/main/webapp/WEB-INF/views/` hinzu:

```jsp
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Home</title>
</head>
<body>
    <h1>Willkommen zu Spring MVC ohne Spring Boot</h1>
</body>
</html>
```

7. Anwendung erstellen und packen

Wenn Sie Maven verwenden, führen Sie den folgenden Befehl vom Projektverzeichnis aus aus, um die WAR-Datei zu erstellen:

```
mvn clean package
```

Dies erzeugt `SimpleSpringMVCApp-1.0-SNAPSHOT.war` im `target`-Verzeichnis.

- **Hinweis**: Wenn Sie Maven nicht verwenden, kompilieren Sie die Java-Dateien und packen Sie das Projekt manuell in eine WAR-Datei, wobei die Ordnerstruktur beibehalten wird.

8. Anwendung bereitstellen

Stellen Sie die WAR-Datei in einem Servlet-Container wie Apache Tomcat bereit: - Kopieren Sie `SimpleSpringMVCApp-1.0-SNAPSHOT.war` in das `webapps`-Verzeichnis Ihrer Tomcat-Installation. - Starten Sie Tomcat (z.B. führen Sie `startup.sh` oder `startup.bat` aus dem `bin`-Verzeichnis von Tomcat aus).

9. Anwendung aufrufen

Öffnen Sie einen Browser und navigieren Sie zu:

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

Sie sollten sehen:

```
Willkommen zu Spring MVC ohne Spring Boot
```

- **Hinweis**: Der Kontextpfad (`/SimpleSpringMVCApp-1.0-SNAPSHOT`) hängt vom Namen der WAR-Datei ab. Wenn Sie die WAR-Datei in `myapp.war` umbenennen, wird die URL zu `http://localhost:8080/myapp/`.

Zusammenfassung

Um **Spring MVC ohne Spring Boot** zu verwenden: 1. **Richten Sie ein Webprojekt** mit einer Standardstruktur ein. 2. **Fügen Sie Abhängigkeiten** für Spring MVC und die Servlet-API hinzu. 3. **Konfigurieren Sie den DispatcherServlet** in `web.xml`. 4. **Definieren Sie die Spring-Konfiguration** in einer XML-Datei (z.B. `spring-mvc-config.xml`). 5. **Erstellen Sie einen Controller** zur Verarbeitung von Anfragen. 6. **Fügen Sie eine JSP-Ansicht** für die Antwort hinzu. 7. **Erstellen und stellen Sie die WAR-Datei** in einem Servlet-Container bereit.

Dieser Ansatz gibt Ihnen volle Kontrolle über die Konfiguration und ist eine großartige Möglichkeit, die Grundlagen von Spring MVC zu erlernen!