**Bài Giảng về Spring MVC**

**1) Giới thiệu Spring, Spring MVC**

**1.1) Giới thiệu Spring**

Spring là một framework mã nguồn mở được phát triển bởi Rod Johnson vào năm 2002, nhằm cung cấp một nền tảng phát triển mạnh mẽ và linh hoạt cho các ứng dụng Java. Spring cung cấp một loạt các mô-đun giúp phát triển ứng dụng nhanh chóng và hiệu quả, bao gồm:

* **Spring Core**: Cung cấp các dịch vụ cơ bản như quản lý đối tượng (IoC), khởi tạo và cấu hình đối tượng.
* **Spring AOP (Aspect Oriented Programming)**: Tích hợp tính năng lập trình hướng khía cạnh để xử lý các vấn đề như logging, transaction management, v.v.
* **Spring Data**: Giúp tương tác với cơ sở dữ liệu, ORM (JPA, Hibernate).
* **Spring Security**: Cung cấp các dịch vụ bảo mật cho ứng dụng.
* **Spring Web**: Xây dựng các ứng dụng web, bao gồm Spring MVC, WebSocket.

**1.2) Luồng xử lý MVC, Spring MVC**

Spring MVC là một mô-đun trong Spring giúp phát triển các ứng dụng web theo mô hình **Model-View-Controller** (MVC). Mô hình này giúp tách biệt các phần của ứng dụng:

* **Model**: Đại diện cho dữ liệu của ứng dụng (thường là các đối tượng Java).
* **View**: Là phần giao diện người dùng, hiển thị dữ liệu từ model.
* **Controller**: Điều phối các yêu cầu từ người dùng và quyết định mô hình và giao diện sẽ được sử dụng như thế nào.

Luồng xử lý trong Spring MVC:

1. Người dùng gửi một yêu cầu HTTP đến ứng dụng.
2. **DispatcherServlet** tiếp nhận yêu cầu.
3. **Controller** xử lý logic yêu cầu và trả về **Model** (dữ liệu) và tên của **View**.
4. **View Resolver** giải quyết tên view và trả về giao diện (JSP, Thymeleaf, v.v.).
5. **View** trả lại cho người dùng.

**1.3) Giới thiệu DispatcherServlet**

* **Vai trò**: DispatcherServlet là một phần trung tâm của Spring MVC, có nhiệm vụ tiếp nhận tất cả các yêu cầu HTTP từ người dùng và chuyển hướng chúng tới các bộ xử lý phù hợp, đồng thời trả về kết quả (thường là một trang JSP hoặc một đối tượng JSON).
* **Luồng xử lý với DispatcherServlet và Spring MVC**:
  1. Người dùng gửi yêu cầu HTTP tới ứng dụng.
  2. DispatcherServlet tiếp nhận yêu cầu và dựa vào các cấu hình trong web.xml và các controller đã được định nghĩa để chuyển yêu cầu tới phương thức xử lý tương ứng trong controller.
  3. Controller xử lý logic và trả về Model và View.
  4. ViewResolver sẽ giải quyết view và trả về kết quả cho người dùng.

**1.4) Config trong file web.xml**

Trong Spring MVC, file web.xml đóng vai trò quan trọng trong việc cấu hình ứng dụng web:

* Định nghĩa DispatcherServlet.
* Cấu hình các tham số khởi tạo cho Spring.
* Cấu hình các filter (như CharacterEncodingFilter, SpringSecurityFilterChain, v.v.)

Ví dụ cấu hình web.xml:

xml

Sao chép

<web-app>

<servlet>

<servlet-name>springDispatcher</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<init-param>

<param-name>contextConfigLocation</param-name>

<param-value>/WEB-INF/spring/dispatcher-servlet.xml</param-value>

</init-param>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>springDispatcher</servlet-name>

<url-pattern>/</url-pattern>

</servlet-mapping>

</web-app>

**2) Bean trong Spring**

**2.1) Định nghĩa bean, cách cấu hình bean trong dispatcher-servlet.xml**

Bean trong Spring là các đối tượng được quản lý bởi **ApplicationContext** (hoặc **BeanFactory**). Các đối tượng này có thể được cấu hình thông qua XML hoặc annotations.

Cấu hình bean trong dispatcher-servlet.xml:

xml

Sao chép

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans.xsd">

<bean id="myController" class="com.example.MyController"/>

</beans>

Trong ví dụ trên, bean myController được khai báo và Spring sẽ tạo đối tượng của lớp MyController khi ứng dụng khởi động.

**2.2) context:component-scan**

Dùng để tự động phát hiện các bean trong ứng dụng Spring mà không cần khai báo chúng trong file XML. Điều này thường được sử dụng với các annotation như @Component, @Controller, @Service, @Repository, v.v.

Ví dụ cấu hình:

xml

Sao chép

<context:component-scan base-package="com.example"/>

Điều này sẽ tự động quét các lớp trong package com.example và khởi tạo các bean cho các lớp được đánh dấu bằng các annotation như @Controller, @Service, v.v.

**2.3) Thẻ bean**

Thẻ <bean> trong Spring XML dùng để khai báo các đối tượng mà Spring sẽ quản lý. Một số thuộc tính của thẻ <bean>:

* **id**: Tên của bean.
* **class**: Lớp Java của bean.
* **scope**: Phạm vi sống của bean (singleton, prototype, v.v.).

Ví dụ cấu hình bean với thuộc tính scope:

xml

Sao chép

<bean id="myService" class="com.example.MyService" scope="singleton"/>

**3) Resolving Views trong Spring MVC**

Spring MVC sử dụng ViewResolver để xác định và trả về giao diện người dùng dựa trên tên của view. Có nhiều loại ViewResolver khác nhau, ví dụ:

* **InternalResourceViewResolver**: Dùng để xử lý các view JSP.
* **ThymeleafViewResolver**: Dùng để xử lý các view với Thymeleaf.

Cấu hình ViewResolver trong Spring:

xml

Sao chép

<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">

<property name="prefix" value="/WEB-INF/views/"/>

<property name="suffix" value=".jsp"/>

</bean>

Khi controller trả về tên view, InternalResourceViewResolver sẽ thêm tiền tố và hậu tố để tìm và hiển thị view tương ứng.

**4) Controller trong Spring MVC**

Controller là lớp xử lý các yêu cầu từ người dùng và trả về kết quả. Trong Spring MVC, controller thường được đánh dấu với annotation @Controller.

Ví dụ về controller trong Spring:

java

Sao chép

@Controller

public class MyController {

@RequestMapping("/home")

public String showHomePage(Model model) {

model.addAttribute("message", "Hello, Spring MVC!");

return "home";

}

}

* @RequestMapping("/home"): ánh xạ yêu cầu tới phương thức showHomePage.
* Phương thức showHomePage xử lý yêu cầu, thêm dữ liệu vào model và trả về tên view "home", Spring sẽ sử dụng ViewResolver để tìm view và trả về cho người dùng.

**Tổng kết**

Spring MVC giúp tách biệt các phần trong ứng dụng web và dễ dàng mở rộng, bảo trì. Bạn đã học về cách cấu hình Spring MVC, cách quản lý các đối tượng (bean), cách cấu hình view, và cách tạo controller để xử lý yêu cầu của người dùng. Spring MVC là một công cụ mạnh mẽ giúp phát triển các ứng dụng web quy mô lớn và phức tạp.