

Bộ môn: Kỹ Thuật Phần Mềm

Lập trình WWW *Java*



Spring MVC - Overview

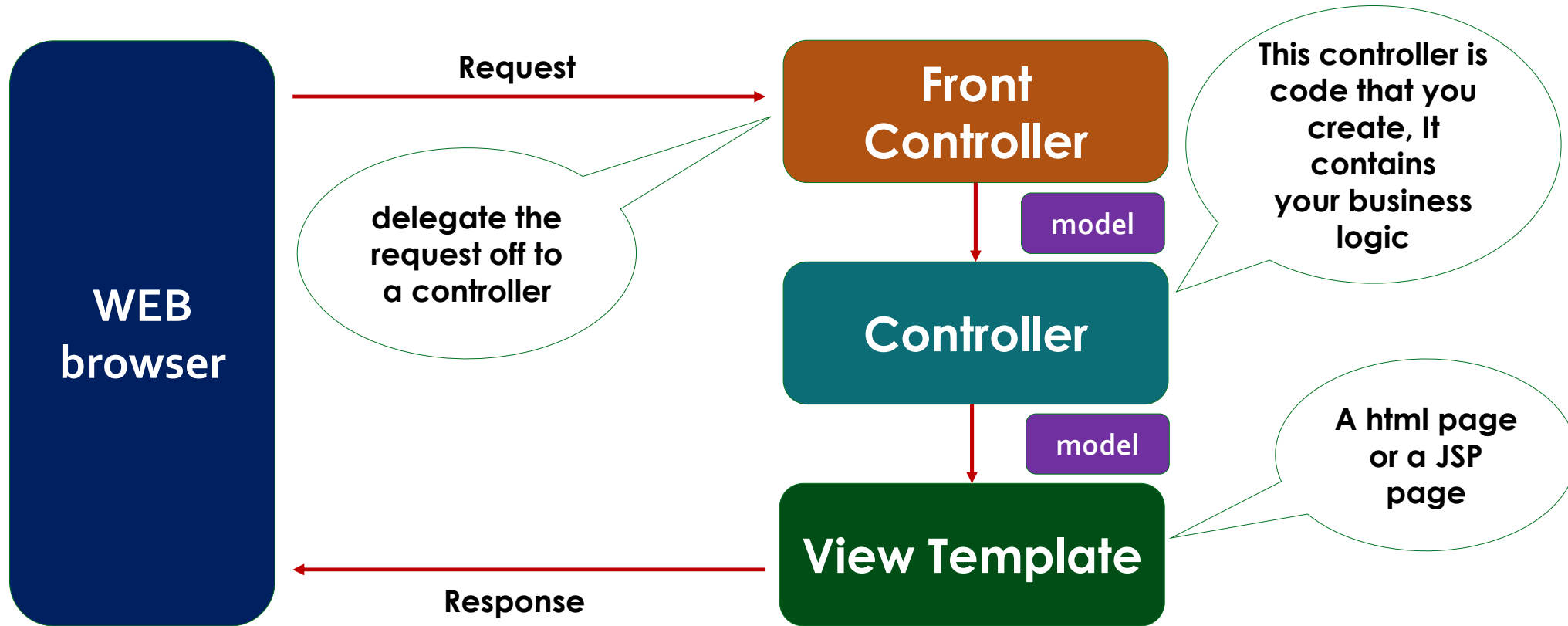


What is Spring MVC



- » Framework for **building web applications** in Java
- » Based on the **Model-View-Controller** design pattern
- » Leverages the **features** of the **Core Spring Framework (IoC, DI)**

Model – View - Controller



- » The **Spring way** of **building web app UIs** in Java
- » **Leverage** a set of reusable **UI components**
- » **Help** manage your **application state** for web requests, **session tracking, application tracking**
- » **Flexible configuration** for the **view layers**

Components of a Spring MVC Application



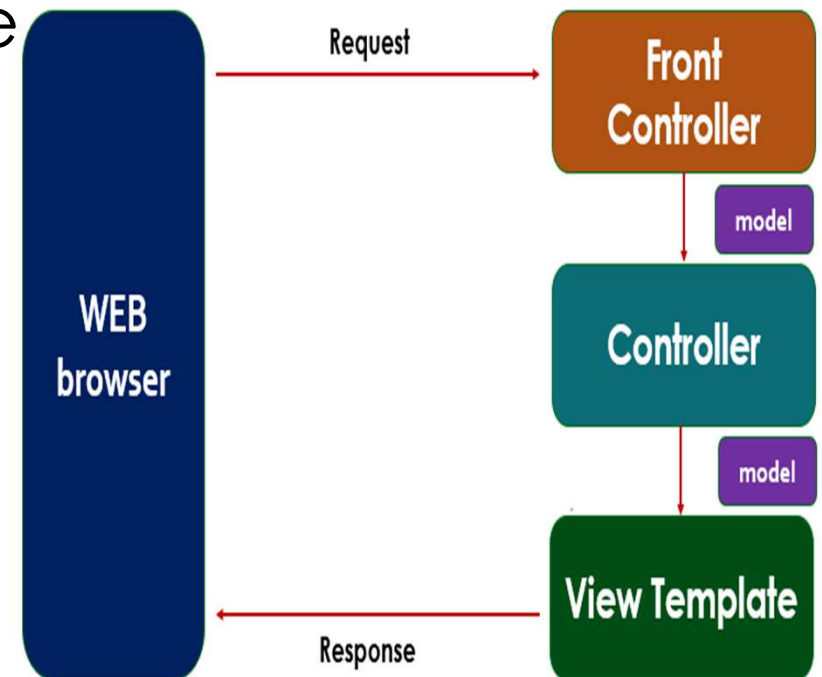
- » A set of **web pages** to **layout UI** components
- » A collection of **Spring beans** for controlling, handling services...
- » **Spring configuration** (XML, Annotations, pure Java)

Web pages

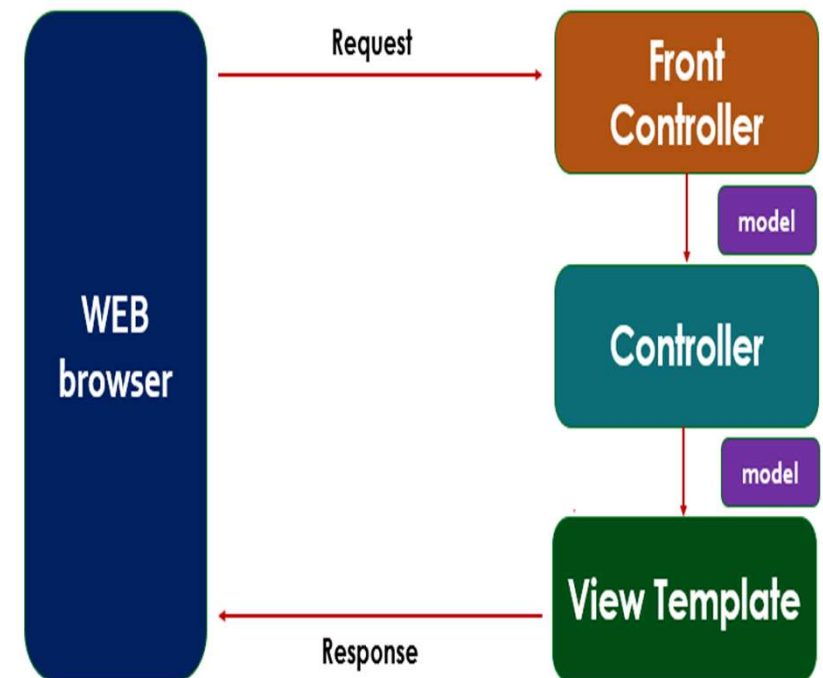
Beans

Spring
configuration

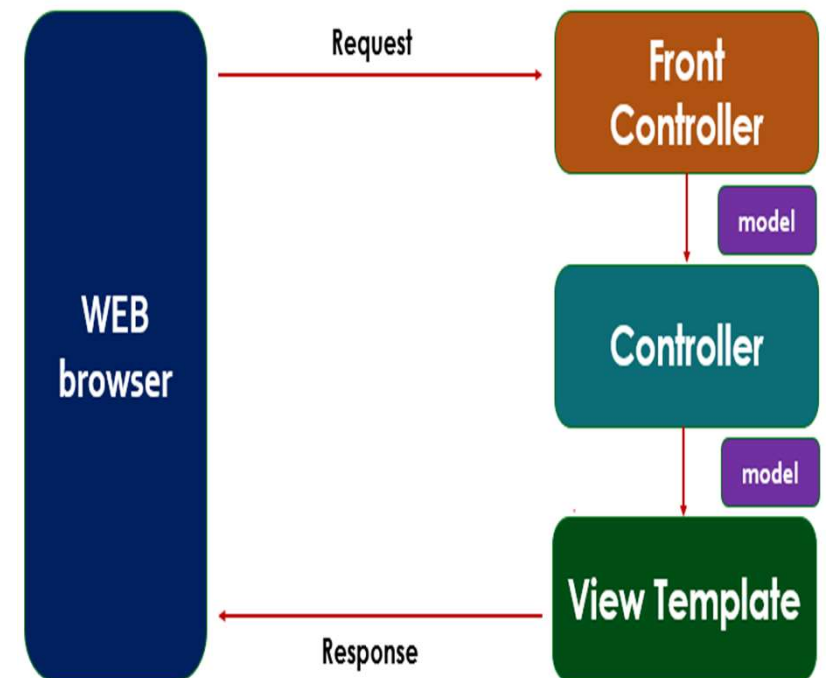
- » **Front controller** is known as the **dispatcher servlet**
 - Part of the Spring framework.
 - It's already developed by the Spring development team
- » You will create **the model**, **the view**, and **the controller**



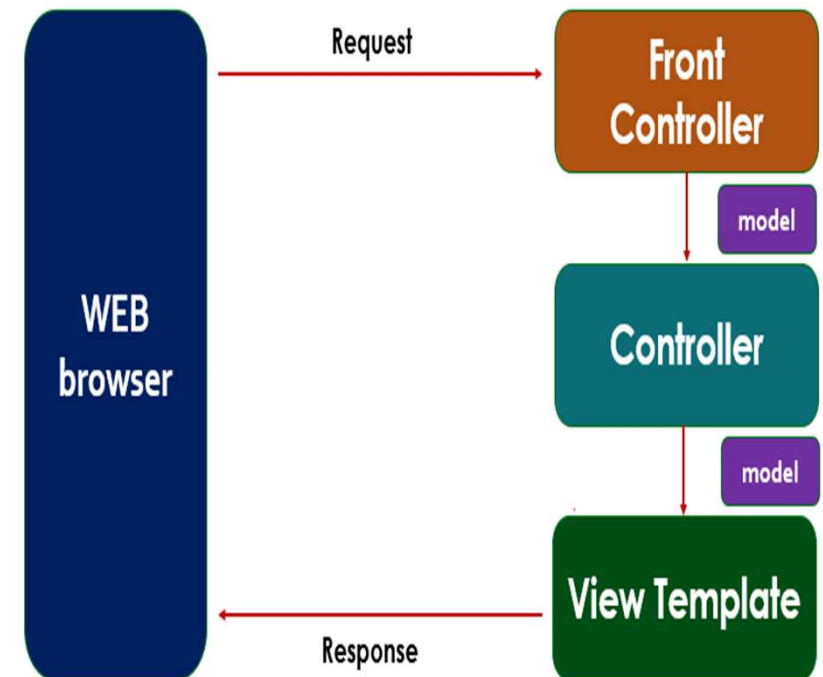
- » **Code** created by **developer**
- » Contains business logic:
 - Handle the **request**
 - Store/retrieve **data** (db, web service..)
 - Place **data** in **model**
- » Pass model to **appropriate view template**



- » **Model:** Contains your data
- » Store/retrieve **data** via **backend system** (db, web service, spring bean..)
- » Place **your data** into **model**



- » **Most common** is JSP + JSTL
- » **Developer** create a page



- » **Apache Tomcat or TomEE**
- » **Eclipse (Java EE version) or Apache NetBeans**
- » **Connect Eclipse/NetBeans to Tomcat or TomEE**

- » **Part 1:** Add **configuration** to file: **WEB-INF/web.xml**
 1. Configure **Spring MVC Dispatcher Servlet**
 2. Setting up a **URL mapping** to **Spring MVC dispatcher servlet**
- » **Part 2:** add some entries into a **Spring context configuration file** (any name you want):
 3. Add **support** for **Spring component scanning**
 4. Add **support** for **conversion, formatting, and validation**
 5. Configure the **Spring MVC view resolver**

Step 1: Configure Spring MVC Dispatcher Servlet



File: web.xml

```
...  
<servlet>  
  <servlet-name>dispatcher</servlet-name>  
  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  
  <init-param>  
    <param-name>contextConfigLocation</param-name>  
    <param-value>/WEB-INF/spring-mvc-context.xml</param-value>  
  </init-param>  
  <load-on-startup>2</load-on-startup>  
</servlet>
```

Configure the Spring
MVC
DispatcherServlet

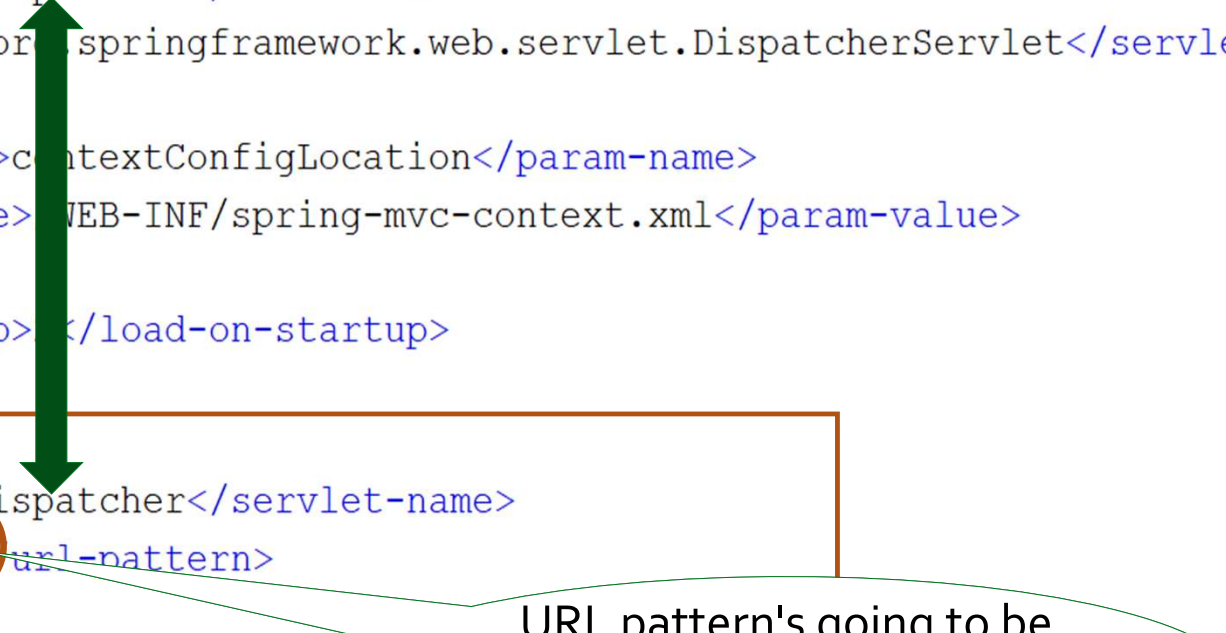
initial parameter:
tell "Dispatcher Servlet" where your
Spring context configuration file is
located.

Step 2: Setting up a URL mapping to Spring MVC dispatcher servlet



File: web.xml

```
<servlet>
  <servlet-name>dispatcher</servlet-name>
  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
  <init-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/spring-mvc-context.xml</param-value>
  </init-param>
  <load-on-startup></load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>dispatcher</servlet-name>
  <url-pattern>/</url-pattern>
</servlet-mapping>
```



URL pattern's going to be slash for handling all the requests

Step 3: Add support for Spring component scanning



File: spring-mvc-context.xml

```
<!-- Step 3: Add support for component scanning -->  
<context:component-scan base-package="com.se.springmvc.demo" />
```

Spring scans this package and all its sub packages for any special Spring beans, and make them available

Step 4: Add support for conversion, formatting, and validation



File: spring-mvc-context.xml

```
<!-- Step 4: Add support for conversion, formatting  
and validation support -->  
<mvc:annotation-driven/>
```

It could perform conversions of form data. It can also format form data, for you, and you can also perform form validation

Step 5: Configure the Spring MVC view resolver



File: spring-mvc-context.xml

```
<bean
    class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name="prefix" value="/WEB-INF/view/" />
    <property name="suffix" value=".jsp" />
</bean>
```

/WEB-INF/view/show-student-list.jsp

If we returned a view name of **show-student-list**.
Spring will prepend **/WEB-INF/view/**, and then they'll
append the suffix **.jsp**



QUESTIONS