**11.3. Spring MVC Configuration – Overview**

**Spring MVC Configuration**:

Spring MVC configuration is probably the hardest part of getting started with Spring MVC. Here we follow step-by-step process to configure Spring MVC.

**Spring MVC Configuration Process - Part 1 (Deployment Descriptor)**:

Add configurations to file: **WEB-INF/web.xml**

1. Configure Spring MVC DispatcherServlet
2. Set up URL mappings to Spring MVC Dispatcher Servlet

**Spring MVC Configuration Process - Part 2 (Spring Bean Configuration File)**:

Add configurations to file: **WEB-INF/spring-mvc-demo-servlet.xml**

1. Add support for Spring component scanning
2. Add support for conversion, formatting and validation
3. Configure Spring MVC ViewResolver

**Step 1: Configure Spring DispatcherServlet**:

In our **web.xml** file, we need to add an entry for the Spring DispatcherServlet, or the Front Controller, so if we put in a servlet reference, we have to give the name and class of the servlet.

Again, the DispatcherServlet is part of the core Spring framework, so we get it for free, in the Spring .jar files. We don’t have to create.

Once we have the servlet reference, then we have set up initial parameter. We basically tell it where our Spring context configuration file is located. In this example we will use of

/WEB-INF/spring-mvc-demo-servlet.xml

**File: web.xml**:

<web-app>

<servlet>

<!-- Dispatcher Servlet -->

<servlet-name>dispatcher</servlet-name>

<servlet-class>

org.springframework.web.servlet.DispatcherServlet

</servlet-class>

<!-- Servlet reference-->

<init-param>

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

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

</init-param>

<!-- loads the servlet at the time of deployment or server start if

value is positive if negative loaded at request time, at first request

-->

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

</servlet>

</web-app>

**Step 2: Set up URL mappings to Spring MVC DispatcherServlet**:

Basically, we want to do is tell the system, “Hey, for any URL pattern coming in, I’d like for you to pass it off to the DispatcherServlet.”

So, in this case our URL pattern’s going to be slash, meaning all web request coming in should be handled by the DispatcherServlet.

**File: web.xml**:

<web-app>

<servlet>

<servlet-name>dispatcher</servlet-name>

<servlet-class>

org.springframework.web.servlet.DispatcherServlet

</servlet-class>

...

</servlet>

<servlet-mapping>

<servlet-name>dispatcher</servlet-name>

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

</servlet-mapping>

</web-app>

Now one thing that, is important here, is to know here the servlet name matches with the servlet reference.



**Step 3: Add support for Spring component scanning**:

In our file that is our Spring file “**spring-mvc-demo-servlet.xm**” we simply add “**context:component-scan**”. This is basically scanning this package for any special beans, and make them available. Any @Component items out there, will make them available.

**File: spring-mvc-demo-servlet.xm**:

<beans> <!-- Step 3: Add support for component scanning --> <context:component-scan base-package="com.ruhul.odduu" /></beans>

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

We make uses Spring-MVC, it can perform conversions of form data. It can also format form data, for developer, and we can also perform form validation. In order to get the support we make use of this "mvc:annotation-driven"

**File: spring-mvc-demo-servlet.xml**:

<beans>  
 <!-- Step 3: Add support for component scanning -->  
 <context:component-scan base-package="com.ruhul.odduu" />

<!-- Step 4: Add support for conversion, formatting and validation

support --> <mvc:annotation-driven/></beans>

**Step 5: Configure Spring MVC ViewResolver**:

The final step here is "**configure Spring MVC ViewResolver**". This part help developer "How to display the pages" or where the pages is located.

**File: spring-mvc-demo-servlet.xml**:

<beans> <!-- Step 3: Add support for component scanning -->  
 <context:component-scan base-package="com.luv2code.springdemo" />

<!-- Step 4: Add support for conversion, formatting and validation

support -->  
 <mvc:annotation-driven/>

<!-- Step 5: Define Spring MVC view resolver --> <bean class="org.springframework.web.servlet.view.

InternalResourceViewResolver"> <property name="prefix" value="/WEB-INF/view/" /> <property name="suffix" value=".jsp" /> </bean></beans>

**View Resolver Configs – Explained**:

When our app provides a “view” name, Spring MVC will

* prepend the prefix
* append the suffix

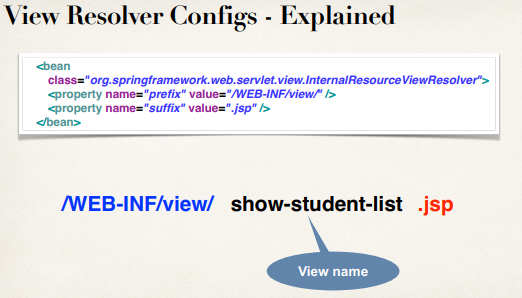
<bean

class="org.springframework.web.servlet.view.

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

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

</bean>



11.3. Spring MVC Configuration - Overview