Java Study Resources

Spring MVC Concepts Study List

The Learning House

427 S 4th Street #300

Louisville KY 40202

Spring MVC Concepts Study List

Overview

The purpose of this document is to give students a guide to the big concepts that should be understood about Spring MVC when heading into an interview. It is unlikely that an interviewer would ask you about specific syntax for Spring configuration files or annotations. However, an interviewer would ask you to explain the big concepts of Spring MVC and expect you to know the purpose of each component and how all of the components interact to function as a web application. It is also expected that you know the basics of HTTP and the evolution of the Java web application.

HTTP Basics

Make sure you understand the basics of HTTP and web application in general. See Unit 09 slide deck “01 - Intro to HTTP and Web Application” slides.

Evolution of the Java Web Application

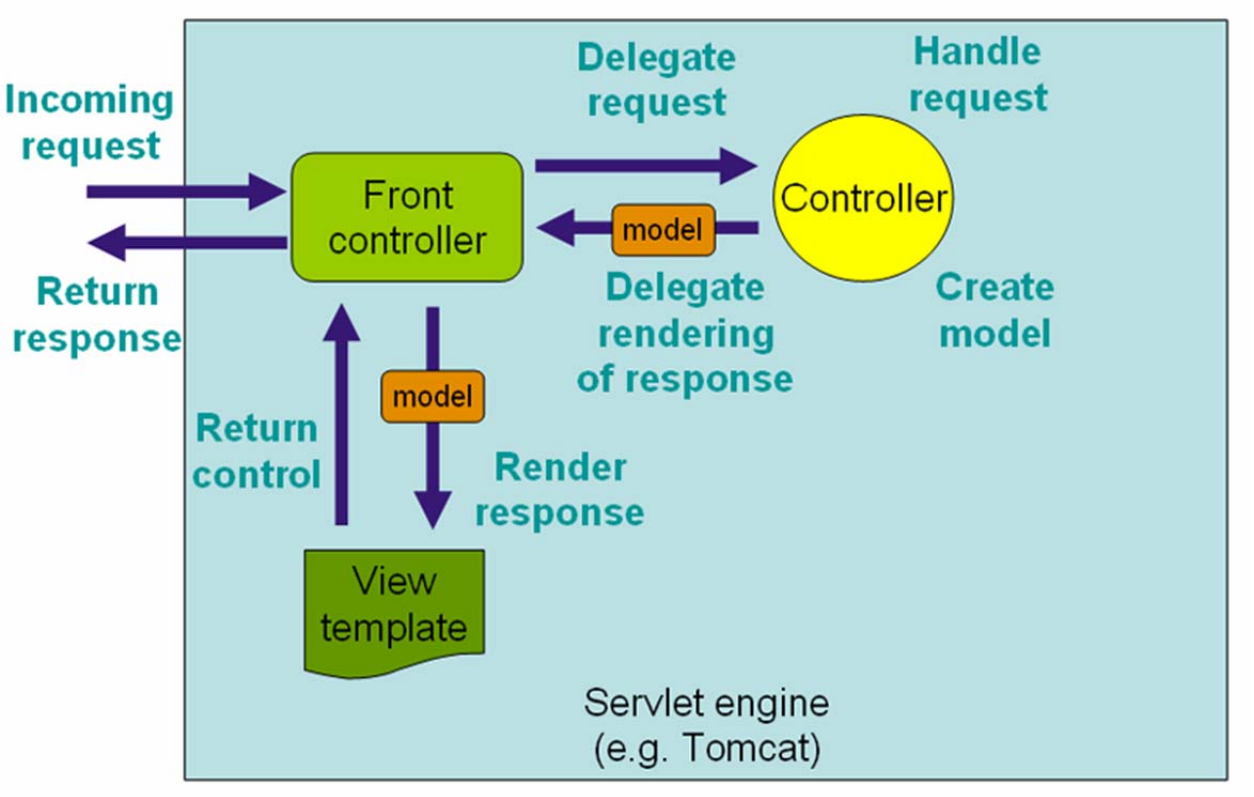
Java web applications have evolved considerably since the late 1990s when JavaServer Pages (JSPs) were introduced. The initial pattern was Model-1 and consisted of JSPs and/or servlets. In Model-1, requests are made to individual JSPs or servlets. The JSP or servlet is completely responsible for handling the incoming request, including request processing, input validation, business logic, and generating the proper response. This model is simple and can work well for very small applications, but it does not scale well and does not lend itself to code reuse and modularization.

The next pattern was Model-2. Model-2 separates the logic for handling the request from the display of the result. The logic necessary to handle the request is contained in a servlet that acts as a Controller. The display of the results is handled by a JSP that acts as the View. The pattern is a form of the Model-View-Controller (MVC) approach.

Spring MVC (as well as other frameworks) represents the next step in this evolution: the Front Controller Pattern, described below.

Front Controller Pattern

The Front Controller Pattern provides a single entry point for all requests into an application. For Spring MVC, the front controller is the DispatcherServlet, which handles all incoming requests and then routes them to proper Controller component in the application:



Front Controller Pattern (from Spring MVC documentation)

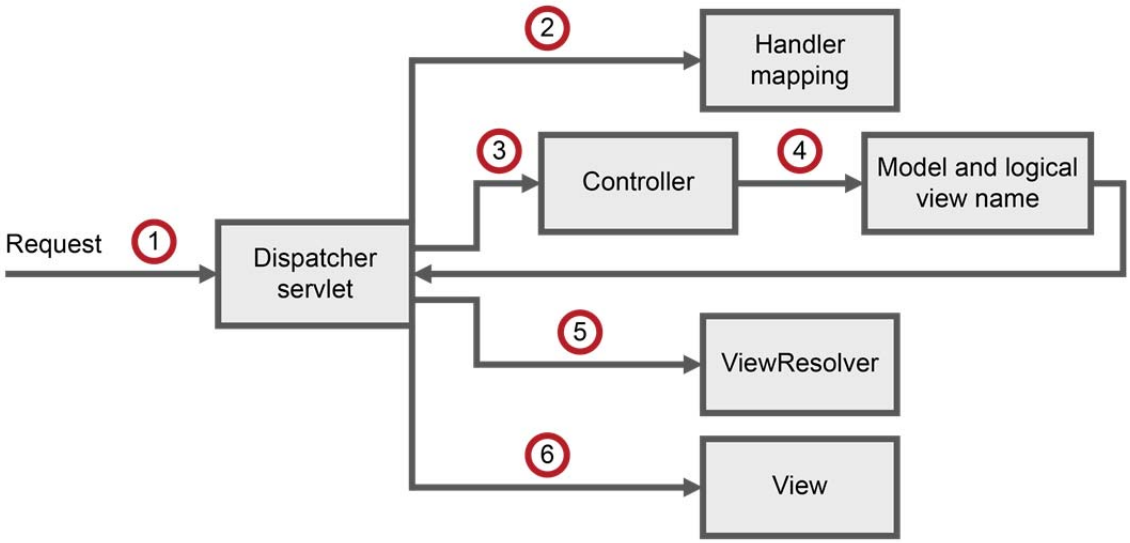
Spring MVC Implementation and Process

Step 1 of the tutorial described how the Dispatcher Servlet, the location of the Controller components, and the View Resolver are configured (see the spring-dispatcher-servlet.xml and web.xml sections). Once those components are configured (the SWCGuild Spring MVC Maven Archetype does this for us), Spring MVC handles requests to the web application in the following manner:

1. Request matching Dispatcher Servlet mapping (see #2-b in the web.xml section of Step 1), which is handled by the Dispatcher Servlet.
2. The Dispatcher Servlet determines which Controller component should handle this request. Controller components are registered with Spring via the @Controller annotation and request to Controller mapping is done via @RequestMapping annotations on the Controller (see #1, #2, and #3 in the HelloController.java section of Step 1).
3. The Dispatcher Servlet routes the request to the Controller.
4. The Controller handles the request and returns the **logical name** of the View component that should be used to render the results of the request (see #5 in the HelloController.java section of Step 1).
5. The Dispatcher Servlet translates the logical name of the View components into the actual View component. This is done according to the View Resolver configuration in

the spring-dispatcher-servlet.xml file (see #4 in the spring-dispatcher-servlet.xml section of Step 1).

6. The Dispatcher Servlet forwards the request to the View component, which renders the result to the client.



Remember the following points:

1. In our applications, the View components are all JSPs
2. In our applications, the Controllers have the following characteristics:
    a. They are POJOs
    b. We use annotations to configure them:
        i. @Controller registers the Controller with Spring as a bean
        ii. @RequestMapping maps URL requests to Controller methods
        iii. @RequestParam maps Request Parameters to Controller method Java parameters

Study Terms/Questions

1. What is HTTP?
2. What is a Servlet?
3. What is a JSP?
4. What is JSP Expression Language?
5. What is JSTL?
6. What does MVC stand for? What is it?

7. What is a Model-1 application?
8. What is a Model-2 application?
9. What is Tomcat?
10. What is Spring MVC?
11. What is a DAO?
12. What is a View? What view technology are we using for our Spring MVC Projects?
13. What are the four HTTP methods?
14. What HTTP method is usually associated with pressing a button on a form?
15. What HTTP method is usually associated with clicking a link?
16. What is a WAR file?
17. What is the Front Controller Pattern?
18. What is a URL?
19. What is the HTTP response status code for success?
20. What is the HTTP response status code for a page that is not found?
21. What is the HTTP response status code for an internal server error?
22. What is the HTTP response status code for forbidden?
23. What are URL parameters?
24. How are URL parameters specified?
25. What are 5 benefits of a servlet container?
26. What role does the Dispatcher Servlet play in a Spring MVC web app?
27. What role does the Controller play in a Spring MVC web app?
28. What role does the View Resolver play in a Spring MVC web app?
29. What is a Web Service?
30. What is AJAX?
31. What is REST?
32. What is SOAP?
33. What is Dependency Injection?
34. What is interface orientation? How does it contribute to loose coupling?
35. What is aspect-oriented programming (AOP)?
36. How is AOP implemented in Spring?
37. What is advice?
38. What is a pointcut?
39. What are joinpoints?
40. In Spring AOP, what/where can joinpoints be?
41. What is an aspect?

42. What is constructor injection?
43. What is setter injection?
44. What does the DispatcherServlet do?
45. How are URL patterns mapped to controller code in Spring MVC?