|  |  |
| --- | --- |
|  | **Spring MVC** |
| Author: PHD | **Self-training roadmap** |
| Review: COH | Imputation number: 221-2007 |
| Date: 07.12.2015 | Total scheduled time: 36 |
| Version: 1.6 | Total time for coach/trainer: 4 |

Self-Training for Spring MVC

**Table of contents**

1. Introduction 1

2. Prerequisites 3

3. References 3

4. Glossary 3

5. Training plan 5

5.1. Basic 5

5.2. Advanced (optional) 12

# Introduction

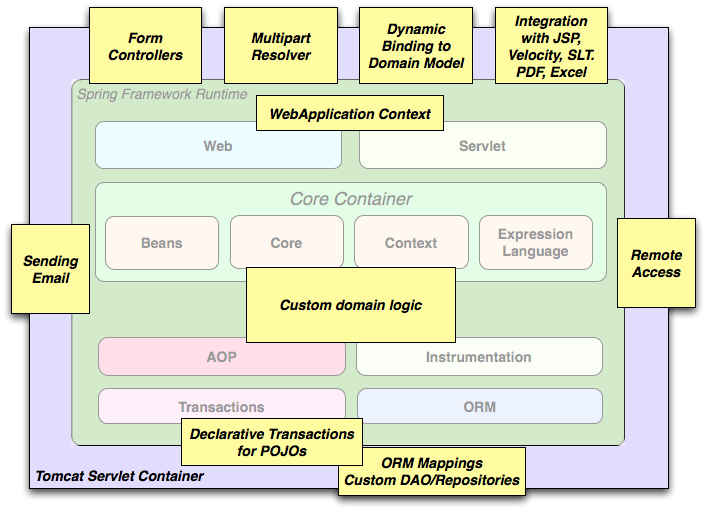
Spring Framework provides a comprehensive programming and configuration model for modern Java-based enterprise applications - on any kind of deployment platform. A key element of Spring is infrastructural support at the application level: Spring focuses on the "plumbing" of enterprise applications so that teams can focus on application-level business logic, without unnecessary ties to specific deployment environments.

Spring Framework consists of features organized into about 20 modules. These modules are grouped into Core Container, Data Access/Integration, Web, AOP (Aspect Oriented Programming), Instrumentation, and Test, as shown in the following diagram.



Among these module, Web MVC is a framework written by Spring as a reaction to what they perceived as poor design of the popular Jakarta Struts web framework, as well as deficiencies in other available frameworks [SPRING-MVC-WIKI].

Spring Framework and Spring MVC is currently used widely in many Java-based projects in ELCA (typically in a full-fledged scenario as below). Therefore, having good knowledge about them is a must to do your job effectively.



# Prerequisites

Having basic knowledge about:

* Java
* Request / Response paradigm
* MVC pattern

Having the following software installed:

* Java SDK 7
* Maven (3.2.5+)
* Eclipse Kepler+

# References

|  |  |
| --- | --- |
| **Ref** | **Source** |
| [SPRING-IN-ACTION] | Craig Walls. Spring In Action, 3rd Edition, Manning Publications 2011 |
| [JSR-000315] | <http://download.oracle.com/otndocs/jcp/servlet-3.0-fr-oth-JSpec/> |
| [SPRING-MVC-WIKI] | <http://en.wikipedia.org/wiki/Spring_MVC#Model-view-controller_framework> |
| [SPRING-WIKI] | <http://en.wikipedia.org/wiki/Spring_framework> |
| [WEBAPP-FW-WIKI] | <http://en.wikipedia.org/wiki/Web_application_framework> |
| [CASESTUDY] | Appendix A |
| [REST-WIKI] | <https://en.wikipedia.org/wiki/Representational_state_transfer> |

# Glossary

|  |  |
| --- | --- |
| **Term** | **Description** |
| Servlet | [JSR-000315] A servlet is a JavaTM technology-based Web component, managed by a container, that generates dynamic content. Like other Java technology-based components, servlets are platform-independent Java classes that are compiled to platform-neutral byte code that can be loaded dynamically into and run by a Java technology-enabled Web server. Containers, sometimes called servlet engines, are Web server extensions that provide servlet functionality. Servlets interact with Web clients via a request/response paradigm implemented by the servlet container. |
| Servlet container | [JSR-000315] The servlet container is a part of aWeb server or application server that provides the network services over which requests and responses are sent, decodes MIME-based requests, and formats MIME-based responses. A servlet container also contains and manages servlets through their lifecycle. |
| REST | In computing, Representational State Transfer (REST) is the software architectural style of the World Wide Web. REST gives a coordinated set of constraints to the design of components in a distributed hypermedia system that can lead to a higher performing and more maintainable architecture. |

# Training plan

## Basic

These steps are mandatory and should be completed by all trainees.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Type[[1]](#footnote-1)** | **Task** | **Description** | **Estimated Time (h)** |
|  | R | Acquire general concept about Spring Framework | Get an “Overview of Spring Framework” here:  <http://docs.spring.io/spring/docs/current/spring-framework-reference/htmlsingle/#spring-introduction>  *Notes: The reading can stop at section 2.3.1 (Dependency Management and Naming Conventions) which is not necessary for this training.* | 2 |
|  | R | Acquire basic understanding of Spring framework | Read the following chapters of the book [SPRING-IN-ACTION]:   * Chapter 1: Springing into action * Chapter 2: Wiring beans * Chapter 3: Minimizing XML configuration in Spring   Read the following link to understand how Java based configuration can be used to configure a Spring application without XML: <http://www.tutorialspoint.com/spring/spring_java_based_configuration.htm>.  You are expected to know at least:   * Spring’s core concepts:   + Dependency injection   + Spring bean life cycle * How to configure and wire Spring beans? * How autowiring and autodiscovery help minimizing XML configuration? * How to use both XML and Java based configuration in a Spring application? | 4 |
|  | E | Setup development environment | Check out the project on-boarding-exercise from SVN (link should be given by your coach).  Configure file etc/env/setenv.bat   * Set **JAVA\_HOME** to the directory where Java is installed * Set **M2\_HOME** to the directory where Maven is installed   Copy etc/maven/settings.xml to your local M2\_HOME\conf.  Bootstrap the project with Maven commands:   * From the directory on-boarding-exercise, open command line, type ..\etc\env\setenv.bat. * Verify versions of Java and Maven by executing the following commands:   + java –version   + mvn -version * Type mvn clean install to clean and build the whole project. In this step, missing dependencies will be downloaded to the local repository and you should see “BUILD SUCCESS” in the command line at the end. * Type mvn eclipse:eclipse to generate Eclipse project descriptor. You should see 2 files .classpath and .project appear next to the pom.xml. * Type mvn eclipse:configure-workspace to generate the Eclipse workspace. A directory named eclipse-workspace should appear at the same level with the on-boarding-exercise directory.   Import the project into Eclipse:   * Open Eclipse and choose the previously generated eclipse-workspace directory as your workspace. * Open menu File –> Import –> General -> Existing Projects into Workspace –> choose the on-boarding-exercise directory. After this, the project on-boarding-exercise should appear in Eclipse   Setup JDK, coding standards and preferences:   * Open menu Windows -> Preferences * On Preferences dialog:   + Java -> Installed JREs: Add (if not exist) and choose JDK7 as the default JRE.   + Java -> Code Style -> Code Templates: Import the java-codeStyle-codeTemplate.xml from etc/eclipse.   + Java -> Code Style -> Formatter : Import the java-codeStyle-formatter.xml from etc/eclipse.   + Java -> Editor -> Templates : Import the java-editor-templates.xml from etc/eclipse. * Open menu File –> Import –> General -> Preferences: Import the preferences.epf from etc/eclipse. | 2 |
|  | E | Working with the Spring configuration in the project | In this step you are expected to understand clearly how Spring was applied.  **Problem 1:** From Eclipse:   * Open ApplicationLauncher and run it as a normal Java application * Open the application from a browser at <http://localhost:8080/> and you will see NullPointerException (actually 2 NullPointerException(s) respectively). * *Can you find the causes?* * *And then please fix the mistakes.*   **Problem 2:** After fixing the 1st problem, you should now be able to view the homepage of the application. However, there is no data displayed when button “Query” is clicked.   * *Can you explain why?* * *Can you fix the problem by introducing an implementation for IProjectService that provides some dummy data and configure the application to use it so that the dummy data can be displayed on the homepage?* | 4 |
|  | C | Check common Spring’s annotations | Coach checks the understanding of trainee about the way the following annotations work:   * + @Configuration   + @Bean   + @Component   + @Controller   + @Service   + @Repository   + @Autowired (with or without @Qualifier)   + @Value | 1 |
|  | R | Introduction to web application framework | Because Spring MVC is a web application framework, it is important to trainee to read [WEBAPP-FW-WIKI] to acquire some knowledge about web application framework in general. | 1 |
|  | R | Understand Spring MVC | Read the the following sections from the book [SPRING-IN-ACTION]:   * Section 7.1: Getting started with Spring MVC * Section 7.2: Writing a basic controller * Section 7.3: Handling controller input * Section 7.4: Processing forms   Please also read the following sections from Spring’s reference to understand some commonly used annotations that are not covered in the book:   * @ModelAttribute   + <http://docs.spring.io/spring/docs/current/spring-framework-reference/html/mvc.html#mvc-ann-modelattrib-methods>   + <http://docs.spring.io/spring/docs/current/spring-framework-reference/html/mvc.html#mvc-ann-modelattrib-method-args> * @SessionAttribute   + <http://docs.spring.io/spring/docs/current/spring-framework-reference/html/mvc.html#mvc-ann-sessionattrib> * @InitBinder   + <http://docs.spring.io/spring/docs/current/spring-framework-reference/html/mvc.html#mvc-ann-initbinder> | 4 |
|  | R | Understand Spring MVC REST | In recent years, REST has emerged as a popular information-centric alternative to traditional SOAP-based web services. Read the following sections from the book [SPRING-IN-ACTION] to understand the REST support from Spring MVC:   * Section 11.1: Getting REST * Section 11.2: Writing resource-oriented controllers * Section 11.3: Representing resources | 2 |
|  | C | Check & explain basic knowledge about Spring MVC | After reading, the trainee is expected to understand:   * The functionality of Dispatcher Servlet, Handler Mapping, Controller, ModelAndView, ViewResolver and View * How the above components interact with each other to process a Request to generate a Response (a work-flow diagram is required) * Explain in depth the most commonly used components:   + @RequestMapping / @RequestParam   + @RequestBody / @ResponseBody   + @ModelAttribute / @SessionAttributes   + @InitBinder * Explain why a controller method can return either a String or a ModelAndView? When each alternative should be used? * Explain how the validation input with JSR-303 (Bean Validation) is done with Spring? And how errors can be communicated back to users? * Should we store editing data as instance member of the controller? | 2 |
|  | R | Study JSP | JSP is one of the view technology supported by Spring MVC. Understand how JSP, JSTL and Spring tags work by reading the “JSP & JSTL” section here: <http://docs.spring.io/autorepo/docs/spring/current/spring-framework-reference/html/view.html>. | 1 |
|  | E | Complete the query-by-name on “search” screen of the on-boarding-exercise | Complete “search” screen of the on-boarding-exercise to have the query-by-name feature working as expected (the criterion is processed correctly on the backend).  *Notes:*   * *The dummy implementation of IProjectService should be used.* * *If you have problem reading the client code with Knockout, please consult the following resources to get a basic knowledge about this very useful JavaScript library:*   + [*http://knockoutjs.com/documentation/introduction.html*](http://knockoutjs.com/documentation/introduction.html)*: Introduction to Knockout*   + [*http://learn.knockoutjs.com/#/?tutorial=intro*](http://learn.knockoutjs.com/#/?tutorial=intro)*: A tutorial to demonstrate the MVVM pattern using Knockout* * *If you’re asking yourself how the application works without any “explicit” Spring MVC configuration (i.e. DispatcherServlet, HandlerMapping, ViewResolver, …), don’t worry. We’re using Spring Boot for convenience and the framework is capable of configuring a Spring MVC Web application automatically for you. More information can be found here:* [*http://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-developing-web-applications.html*](http://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-developing-web-applications.html)*.* | 3 |
|  |  | Initialize “detail” screen of the on-boarding-exercise | Initialize “detail” screen of the on-boarding-exercise. It should cover the following features:   * Load the corresponding project and display it on browser when navigating from “search” screen. * The loaded project can be modified and saved successfully. * After successfully saving, the application returns back to “search” screen. * Previously entered criterion must be preserved after navigating to “detail” screen and back.   *Notes*:   * *The dummy implementation of IProjectService should be used.* * *No Knockout implementaton is required here. The whole screen should be implemented purely with Spring MVC and JSP.* * *@InitBinder can be used to register custom date editor/formatter to serialize/deserialize date time data with different formats* | 8 |
|  | C | Check the previous exercise | Verify if the trainees can do what they are expected | 1 |
|  | C | Expert validation | Validate the training result with expert:   * Expert’s visa: MMN * Validation date: * Remark: Check if all business requirements are fulfilled. | 1 |

## Advanced (optional)

These steps are for experienced trainees and should be used only when the basic plan is completed under the estimated time.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Type[[2]](#footnote-2)** | **Task** | **Description** | **Estimated Time (h)** |
|  | R, E | Understand Spring Bean scopes | When you create a bean definition, you create a recipe for creating actual instances of the class defined by that bean definition. The idea that a bean definition is a recipe is important, because it means that, as with a class, you can create many object instances from a single recipe.  Please help to answer the following questions:   * Name all bean scopes are supported out-of-the-box in Spring. * For each listed bean scope, describe their purpose and give one example of their application. * Can you describe in detail how session-scope beans work?   Then, configure a session-scope bean for storing user preference in the existing application. How can we verify that the bean is truely session-scope? | NA |
|  | R, E | Understand Aspect Oriented Programming (AOP) with Spring | AOP complements OOP by providing another way of thinking about program structure. Whereas OO decomposes applications into a hierarchy of objects, AOP decomposes programs into aspects or concerns. This enables the modularization of concerns such as transaction management that would otherwise cut across multiple objects (such concerns are often termed crosscutting concerns).  Please firstly help to answer the following questions:   * What are “aspect”, “join point”, “advice” and “pointcut” in Spring AOP? * How many proxying mechanisms supported in Spring? What are the pros/cons between them? * Give 3 applications of AOP in Spring?   Secondly, configure the existing application to make ProjectService transactional and propose a simple way to verify that the configuration is correct.  Thirdly, configure an annotation-based “around advice” to trace every call to ProjectService by logging a simple message to the console out. | NA |
|  | R, E | Understand Spring MVC’s advanced concepts | Please answer the following questions:   * What is the scope of beans annotated with @Controller in Spring MVC? How can we verify the answer using the developing application? * As you’ve studied about @SessionAttribute before, it is time for a bit more tricky question: Do you think it is thread-safe to use @SessionAttribute in every web application? If it is not, can you find or propose a solution to mitigate or better, completely solve the problem? Then, please implement and verify the solution in the developing application. | NA |

1. R = Reading, E = Exercise, C = Checkpoint, S=Setup, P=Presentation [↑](#footnote-ref-1)
2. R = Reading, E = Exercise, C = Checkpoint, S=Setup, P=Presentation [↑](#footnote-ref-2)