

# **Spring Web Developer Certification Study Guide**

Completed: February, 2010

# Table of Contents

Overview.....	4
Topics By Subject Area.....	5
Spring Web Overview.....	5
Spring MVC DispatcherServlet Configuration .....	5
HandlerMapping.....	5
HandlerAdapter.....	5
HandlerInterceptor.....	6
MessageSource.....	6
Spring MVC Programming Model Essentials.....	6
@RequestMapping Annotation.....	6
Request Handling Methods.....	6
@RequestParam Annotation.....	6
@ModelAttribute Annotation.....	6
Spring MVC Views.....	7
View Resolvers.....	7
Spring MVC Form Processing.....	7
@SessionAttributes Annotation.....	7
@InitBinder Annotation and Data Binding Customizations.....	7
Spring Form Tag Library.....	7
Spring JavaScript .....	7
JavaScript.....	8
ResourceServlet.....	8
Spring JavaScript Decorations.....	8
Spring Web Flow Overview.....	8
Spring Web Flow Configuration .....	9
Flow Handler Mapping.....	9
Flow Handler Adapter.....	9
Flow Registry.....	9
Flow Registry.....	9
Flow Builder Services.....	9
Spring Web Flow Essentials .....	9
Spring Web Flow Actions .....	10

Spring Faces .....	10
Securing Web Applications With Spring Security .....	11
Testing Web Applications.....	11
Resources.....	12
Tools.....	12

# Overview

This guide is designed to help you prepare for the Spring Web Developer certification exam.

If you're considering taking the exam you've most likely taken the SpringSource 4-day [Rich Web](#) training. The certification exam is based on this training and the materials provided with it are the ideal source to use for preparation. Of course as with any certification the most valuable part, besides recognition, is the learning process. Hence we encourage you to take time to experiment and follow your curiosity when questions arise.

A 4-day course contains a lot of material. To help you focus your efforts and to know when you're ready we've put together this guide. The guide contains a list of topics and a list of further resources. Topics are organized by subject area where each topic contains a description of what you should make sure you know.

The list of topics can be used as a check-list. The training materials is a point of reference and as a learning ground. The list of resources is where you can go further for getting answers.

*If you're planning to take the exam without having taking the course (i.e. through the process known as "grandfathering"), you'll have to rely primarily on the list of topics as your check-list and use any resources listed here or that you know of in order to prepare.*

One possible way to prepare is to do the following for a given training module:

1. Review the slides making notes of questions
2. Work through the lab (if there is one)
3. Review the list of topics that matches to the module by subject area
4. Use the lab to experiment with anything you need to spend more time on
5. Use the provided list of resources to look for further answers

Of course there are many more ways to organize your efforts. You can pair up with someone else planning to take the exam or review all presentations for a given subject area before going through the their labs. Or maybe you have access to actual applications you can review to test your knowledge.

Last but not least we're always interested to hear your feedback. You can send an email to your instructor and/or to [training@springsource.com](mailto:training@springsource.com) to let us know what you thought.

# Topics By Subject Area

The following is a list of topics each of which is likely to have questions on the exam. The topics are organized by subject area.

## Spring Web Overview

How Spring applications can be loaded in a Servlet container independent of what framework is used (Spring MVC, Struts, etc.) to develop the web layer.

Basic facts about what Spring Web is, what products it consists of and how they relate to each other in terms of dependencies. For example Spring Faces builds on Spring MVC, Spring Web Flow and Spring JavaScript.

*Tip: An easy way to explore project dependencies is to start the SpringSource Tool Suite, open a maven pom.xml file for a project that uses Spring MVC, Spring Web Flow, Spring JavaScript, and/or Spring Faces, and then click on the “Dependency Graph” (or the “Dependency Hierarchy”) tab. Alternatively, browse the SpringSource repository <http://www.springsource.com/repository> (a web application that uses Spring MVC, Spring JavaScript and Dojo).*

## Spring MVC DispatcherServlet Configuration

How to configure the DispatcherServlet in web.xml.

Where Spring-based, DispatcherServlet configuration is expected to be by convention if a location is not provided.

What configuration can be provided in the Spring-based, DispatcherServlet configuration and what configuration is used by default.

### **Tip:**

*An important concept to keep in mind is that the DispatcherServlet looks for certain types of Spring MVC “infrastructure” beans in its Spring configuration (see mvc-essentials-2). Examples of such types of beans include MessageSource, HandlerMapping, ViewResolver, and others. The exam will test your understanding of the purpose of these Spring MVC beans including what their purpose is, how they are discovered (by bean class type or by bean id), what implementations are configured by default, and so on.*

*Understanding Spring MVC “infrastructure” beans as well learning how to explore what they provide is important step to learning Spring MVC. One way to learn is to open each type listed in the sections below, review the class-level JavaDoc, and explore available subtypes with the help of the Ctrl+T shortcut.*

*Another exercise is to use Ctrl+Space in Spring configuration files on bean property names in order to see the list of available properties. These properties describe how a given infrastructure type can be customized.*

## HandlerMapping

The purpose of the HandlerMapping strategy and the details of configuring it.

How the DefaultAnnotationHandlerMapping and the ControllerClassNameHandlerMapping implementations work and what they would do in the case of specific incoming requests.

## HandlerAdapter

The purpose of the HandlerAdapter strategy and the details of configuring it.

## HandlerInterceptor

The purpose of the HandlerInterceptor strategy and the details of configuring it. What points in the lifecycle of the request can be intercepted, and what data is available at each point.

## MessageSource

The purpose of the MessageSource strategy and the details of configuring it.

## Spring MVC Programming Model Essentials

The basics of the annotation-based programming model in Spring MVC, configuring @Controller-annotated classes, writing request-handling methods, and testing them.

### @RequestMapping Annotation

The purpose of the annotation, what can be annotated, and what options (or attributes) it provides.

How a URL breaks down (web application context, servlet name, path info) as well as which part of the URL is used in Spring MVC for request mapping purposes.

#### **Tip:**

*In the exam you may be given basic examples of URL's and annotated controller methods. You will be expected to predict what methods will be invoked. The best way to experiment is to try it out. Use the course labs, and check the Spring MVC logging information on each request!*

*Also examine the logs for output from HandlerMapping beans. HandlerMapping beans construct URL mappings during startup and log that information. Log levels for org.springframework.web must be set to DEBUG.*

*Below are a couple of specific scenarios to experiment with.*

*Scenario 1: a DefaultAnnotationHandlerMapping with one Controller and a single method annotated with @RequestMapping("/foo").*

*Questions: What URL's reach the method successfully? Does adding an extension to the URL (.pdf, .xml) make a difference? How about more segments (/foo/bar)? Passing anything (/other)? Try switching to @RequestMapping(method=RequestMethod.GET), does it still work?*

*Scenario 2: Similar to scenario 1 but involving a `ControllerClassNameHandlerMapping`, a controller (`FooController`) and a method (`bar`) annotated with `@RequestMapping(method=RequestMethod.GET)`*

## Request Handling Methods

How to write methods to handle requests, what annotations can be used, what the signature of the method can be such as input argument types and return types, what happens if the method returns void.

### **Tip:**

*In addition to the annotations listed below, the JavaDoc of the `@RequestMapping` annotation is a good place to start for information on how input arguments and return values are interpreted on `@RequestMapping`-annotated methods.*

## `@RequestParam` Annotation

The purpose of the annotation, what can be annotated, what options (or attributes) it provides, and how to handle optional parameters.

## `@ModelAttribute` Annotation

The purpose of the annotation, what can be annotated, what options (or attributes) it provides, what is the default name given to a model attribute object if the attribute name is left unspecified, and what is the default attribute name given to a model attribute object that is an array or a collection.

## Spring MVC Views

The basics of how views work, what they do, how they're typically instantiated through the process of view resolution, what is a logical view name, and what is the default logical view name selected if a controller method leaves it unspecified (i.e. return value is void or null).

## View Resolvers

The purpose of the `ViewResolver` strategy, and the details of configuring it.

How `ViewResolver` chains work and how they can be used to render multiple content types – for example re-use the same controller method to render HTML, PDF, or XML depending on the content type requested by the client.

## Spring MVC Form Processing

The basics of working with forms such as how to configure data binding through the `@ModelAttribute` annotation and how data binding is used to populate the form object from request parameter values.

Whether data binding can be used on a POST or can it also be done on a GET request (consider search forms vs. forms updating data).

How a request handling method can get access to the results of data binding.

## @SessionAttributes Annotation

The purpose of the SessionAttributes annotation, what can be annotated, and what options it provides.

**Tip:** *The SessionAttributes annotation provides more than one ways to specify what objects should be added to the HTTP Session. Be sure to check them. A good way to learn what options any Spring annotation provides is the JavaDoc of the annotation.*

The lifecycle of attributes specified with the @SessionAttributes annotation – how long they remain around, when and how they can be removed from the HTTP Session.

How the SessionAttributes works when it's used with multiple controllers – for example is the data stored globally to the HTTP session (e.g. user preferences) or is it per-controller (e.g. account editing).

## @InitBinder Annotation and Data Binding Customizations

The purpose of the InitBinder annotation, what can be annotated, and what options it provides.

What customizations can be applied to the data binding mechanism.

What error codes are generated automatically during data binding.

What error codes can be used to customize the errors generated during data binding.

## Spring Form Tag Library

The purpose and the value provided by the Spring form tags, and how to use them.

## Spring JavaScript

The goals of the Spring JavaScript framework, what it aims to provide in comparison to other JavaScript frameworks, and what benefits its benefits are.

*A couple of techniques promoted by Spring JavaScript are “progressive enhancement” and “unobtrusive JavaScript”. Review their meaning. The Obtrusive JavaScript Checker add-on in Firefox is useful for learning.*

What's bundled inside the Spring JavaScript distribution.

## JavaScript

*The amount of JavaScript knowledge expected in the exam is minimal. It's what you would need to know to understand and work with Spring JavaScript declarations. The Spring JavaScript API is a good source of examples for most topics listed below.*

*The JavaScript course presentation should also be used as a starting point for explorations. Also use the FireBug Command Line window to experiment with a few JavaScript code snippets (<http://getfirebug.com/cl.html>).*

Basic JavaScript syntax – what are the various ways to declare functions and to create objects.

JavaScript types and objects. What the available JavaScript types are, what are considered objects in JavaScript (for example are functions objects?)

Variable scoping such as global variables, the global browser object, techniques to reduce the footprint of global variables.

How argument passing works in JavaScript functions (what if I pass more parameters or fewer than the function expects? what if pass them in the wrong order?)

A common techniques for designing JavaScript functions that accept optional parameters. For example see the input to Spring JavaScript's ElementDecoration or AjaxEventDecoration.

## **ResourceServlet**

The purpose of the ResourceServlet, how it is typically configured in web.xml.

What resources the ResourceServlet can serve and the details of where it can find them.

If resources be located on the classpath and/or under the root of the web application.

*In the exam you may be given examples of a complete URL and are expected to know where the resource can be located.*

## **Spring JavaScript Decorations**

The details of configuring Spring JavaScript decorations, what input arguments each decoration expects, and the semantics of each input arguments.

How Spring JavaScript decorations can be included on a page and where they can appear relative to the elements being decorated.

## **Spring Web Flow Overview**

What the motivation for the existence of Spring Web Flow is and what common problems it solves.

A basic understanding of how Spring Web Flow fits into Spring MVC.

## **Spring Web Flow Configuration**

How Spring Web Flow is configured including both Spring MVC and Spring Web Flow-specific configuration.

*Much like Spring MVC, Spring Web Flow provides and expects a few infrastructure classes to be configured as Spring beans. Some of these infrastructure beans (e.g. FlowHandlerMapping, FlowHandlerAdapter) are Spring MVC-specific, while others (e.g. FlowExecutor, FlowRegistry, and others) are specific to Spring Web Flow. Listed below are the infrastructure types you should review and understand.*

## **Flow Handler Mapping**

The purpose of the FlowHandlerMapping strategy, the details of configuring it, and how it decides whether an incoming URL matches to any registered flows.

## Flow Handler Adapter

The purpose of the FlowHandlerAdapter strategy.

## Flow Registry

The purpose of the FlowExecutor and what configuration options it provides.

## Flow Registry

The purpose of the FlowRegistry, the details of how flows are registered and id's assigned to them.

*In the exam you may be given example FlowRegistry configuration and incoming URL's. You're then expected to know which URL would match to a given flow.*

*Specifically consider the possibility for registering flows one by one and many at once using a pattern. In each case how does a flow get assigned an id?*

## Flow Builder Services

The purpose of the FlowBuilderServices, what configuration options it provides.

What the development mode option is.

## Spring Web Flow Essentials

What a Web Flow view state is and what it represents. What happens when a view state is reached. How a view state is resolved to a specific view such as a JSP page. What the attributes of a view state are.

What a transitions in Web Flow is. What triggers a transition, what happens when a transition is triggered, and whether a transition can be prevented once it's been triggered. What global transitions are.

How a button or a link on an HTML page can be used to raise a specific Web Flow event.

What a Web Flow end state is. What happens when an end state is reached. What Web Flow does by a default for a flow that has ended. What is a good practice for what to do when a flow ends.

How to write a Web Flow unit test, what base class to extend, what methods you'd expect to override, and how to implement test methods that navigate through the flow.

## Spring Web Flow Actions

What scopes Web Flow provides and how long variables in each scope live.

The 3 ways to create a variable in a flow definition, what the scope of the created variable is and whether it will be dependency injected in each case.

How Web Flow resolves variables encountered in the EL expressions within a flow definition. For example will it search the various web flow scopes or in Spring configuration, and are there reserved words? Also important is the order in which Web Flow goes to try to resolve variables

encountered in EL.

**Tip:** To make this easier, open an existing flow definition, and try to locate all the EL expressions in evaluate elements. For each expression determine what variables are used what their origin is – a reserved keyword, a scoped variable, a Spring bean? Also notice how when referencing scoped variables you don't have to specify what scope they come from – flash, view, flow, etc. Web Flow will automatically look in all scopes and try to locate the matching variable.

What the reserved EL keywords are in Web Flow. Much like the JSP EL provides reserved keywords (pageContext, request, response), so does Web Flow (flashScope, requestScope, currentEvent, and others).

What the 4 different types of Web Flow actions are. Where actions can be embedded in a flow definition. What actions are invoked when. For example what is the difference between on-render actions and on-entry actions?

How data binding and validation are enabled for a specific view state in Web Flow. How validation and data binding can be suppressed for a specific transition (e.g. cancel event).

The purpose of the data binder element.

How to implement validation logic – dedicated validator class or validate methods on the model attribute class. How Web Flow finds the validation logic. How to invoke validation logic for a specific view state.

## Spring Faces

What Spring Faces is, what it provides, and what the main underlying technologies it uses are.

The ways in which Spring Faces complements JSF.

What JSF components Spring Faces provides.

How Spring Faces command buttons and command links work, how they signal Web Flow events, what attributes they provide, and what Ajax capabilities they provide.

## Securing Web Applications With Spring Security

What web.xml configuration is required to enable Spring Security and what the mechanism Spring Security uses to protect web applications.

What Spring Security related configuration is needed to secure a web application.

How URL patterns should be configured.

**Tip:** Pay special attention to the order in which URL patterns are provided. Does it matter if you put the more general (e.g. /accounts/\*) or the more specific (e.g. /accounts/edit) pattern first?

How method level security can be added to an application.

How authentication and authorization relate to each other – for example does the choice of authentication affect authorization?

## Testing Web Applications

The difference between logical unit testing and integration testing and when to use which.

How controllers should be tested – for example unit test or integration test?

What functional tests are and what their role in testing web applications.

How Selenium and Apache JMeter can be used for testing applications. How tests can be created with each and how they work.

# Resources

This section contains a list of resources relating for learning.

**[Spring Community Forums](#)** – look for existing discussions or start your own, take advantage of one of the best parts of Spring: its community.

**[SpringSource Blog](#)** – point your favourite RSS reader or come back every so often for detailed, quality posts by Spring developers.

**[Reference Documentation](#)** – add bookmarks in your browser to the reference documentation pages for [Spring](#) (namely the chapter on [Spring Web MVC](#)), [Spring Web Flow](#) (including Spring JavaScript and Spring Faces), and [Spring Security](#).

**[Spring Samples](#)** – a subversion repository with projects that can be built with maven and imported into STS/Eclipse. Some of the samples have associated blog posts on the SpringSource Blog listed above. You could check by searching on the keywords: “sample name” springsource blog.

**[Spring By Example](#)** – another good repository with complete code samples and the ability to contribute your own samples.

**[Web Sites](#)** – it's hard to single out individual web sites. There are so many. If we had to name a few they would include [Infoq](#), [Dzone](#), [JavaWorld](#), [Spring Hub](#) among many others.

**[Books](#)** – there are many books. It's also hard to find ones that are up-to-date because it takes so much effort to write them or keep them up-to-date, so check the date of the last edition and the version of the framework covered. Publishers like (e.g. Apress, Manning) provide early access to book chapters in PDF as they are being written.

Books about Spring have chapters on Spring MVC at least. To name a few quality Spring books: ["Pro Spring 2.5"](#), ["Spring In Action"](#), ["Spring In Practice"](#).

Books about Spring Web Flow are harder to come by and your mileage with them will vary. ["The Definitive Guide to Web Flow"](#) is a book with good depth but is slightly out-of-date. There are other books you can find. However the one listed here is the most reliable one in terms of quality.

**[Spring Projects JIRA](#)** – most likely not the first place to come to in the beginning but overall a great learning resource when looking up information on very specific issues or new features. You can read comments, leave comments, as well as vote. Sometimes discussions on the community forums result in the creation of issues in JIRA.

# Tools

**SpringSource Tool Suite** – Spring project templates, tutorials, bean diagrams, code completion and more. See the PDF files under “Related Content” for details on available features. STS is updated often and it's free. It also supports Spring Roo and Grails.

**Spring Roo** – a rapid web application development tool based based on most of the technologies taught in the course and also a great tool for “rapid” learning.

**Grails** – a rapid web application Groovy & Java development platform that also uses Spring Web technologies. Grails has a diverse set of [plugins](#) that can introduce you to many web technologies, techniques, and approaches.

**Spring Insight** – when you download STS 2.2.0 or higher, it includes a version of tc Server bundled with the Spring Insight webapp. It gives you lightweight tracing of your Spring application including links to the source code. A great tool for learning that can be used with existing applications with no changes to the application.