

Spring Framework Training Course Outline

SUMMARY: This course enables the experienced Java developer to use the Spring Application Framework to create simple and complex Web applications. Spring is a far-reaching framework that aims to facilitate all sorts of Java development, including every level of multi-tier distributed systems. Here we focus on the Core and MVC modules, with a lighter (but not dismissive) touch on persistence through DAO and ORM modules.

The Core module gives the developer declarative control over object creation and assembly; this is useful for any tier of any Java application. So is Spring's validation framework, and so we study these things in a mix of standalone (J2SE) applications and Web applications deployed to the Tomcat server/container. Then students build Web applications that use the Spring MVC framework to rationalize their designs into coherent request/response cycles. They use Spring command objects to manage HTML forms and their data, and connect these to the validation framework. We connect our applications to persistent stores and study the DAO and ORM modules, to better understand JDBC and Hibernate persistence models and declarative transaction control.

PREREQUISITES

Java programming
Servlets programming.
JSP programming.
Basic knowledge of XML.

OBJECTIVES

- Understand the scope, purpose, and architecture of Spring
- Use Spring's Inversion of Control to declare application components, rather than hard-coding their states and lifecycles
- Use Dependency Injection to further control object relationships from outside the Java code base
- Create validators for business objects, and associate them for application-level and unit-testing uses
- Build a Web application as a Spring DispatcherServlet and associated application context, with declared beans acting as controllers, command objects, and view resolvers
- Build and manage HTML forms with Spring command objects and custom tags
- Use Spring interceptors to implement horizontal features in the Web application
- Connect business objects to persistent stores using Spring's DAO and ORM modules

Introduction

Basic environment
Spring modules

Spring Architecture

Spring Framework definition
Spring Framework design principals
Spring interfaces

Spring setup

Setting classpath and jar files
Setting configuration

Design Patterns

Inversion of Control
Dependency Injection
Spring & MVC

Spring Core

Dependency injection feature
Factory Pattern
BeanFactory
Spring Context definition

Inversion of Control (IoC)

Injecting dependencies
IoC in enterprise applicatio

Aspect Oriented Programming

Spring AOP
AOP in enterprise application
AOP to provide enterprise services
AOP to provide customer implementation

Bean Factories

Application Context and BeanFactory
Attaching and Populating beans
Injecting data through setters
Injecting data through constructors
Spring special beans
Post processing beans
Listening on events
Publishing events

JDBC Data Access

JDBC Abstraction layer
Data Access exception hierarchy
Error handling Strategy

Spring ORM

Database access layer for object, relational databases,
Mapping API for JDO, Hibernate

DAO Persistence ORM

Hibernate Mapping
JDO Mapping
iBATIS

Spring Abstract Transaction layer

Employing Spring transaction
Using EJB declarative transactions

Integration process

Spring Web
Spring Web application
Integrating Spring MVC in web application

MVC Framework

Build on core spring functionality
Configuring using Strategy Pattern
Accommodating different views like Tiles, JSP, Velocity
Using other frameworks like struts

Spring Remote Objects

Spring Web Services
Spring RMI

Unit testing of components

Integrated Testing for Beans