

Course 445: Mastering the Spring Framework and Hibernate (5 days)

This in-depth, advanced workshop using Spring 2.x will present developers with best practices for software development, as well as tips and techniques for working with the tools and technologies within their specific environment.

Who can benefit?

This is an intermediate level Java programming course, designed for developers who wish to understand Spring. The student should be an experienced J2EE / Java programmer, with practical development experience using Servlets/JSPs.

Spring provides a light-weight, powerful framework that can dramatically simplify the development of Java enterprise applications and services. Spring applications are easier to maintain, better performing and more robust than applications built using only the core JEE technologies.

Attendees in this course will perform hands-on exercises, building an enterprise application as they learn the concepts. Therefore, developers, system architects and managers with some real-world experience of working with enterprise applications in Java will benefit the most from this course.

Who should attend...

- ◆ Java programmers involved in developing enterprise applications
- ◆ Software system architects
- ◆ Managers of technical projects

Prerequisites...

Knowledge and understanding of Java to the level of Course 430 (Essential Java) is required. Familiarity with Java web applications (Servlets/JSPs) and XML syntax is helpful, but not required.

Learning Objectives...

- Best practices in architecting a JEE system
- Injecting dependencies using the Spring bean factory
- Simplifying database access
- Persisting business objects using Hibernate
- Implementing a presentation tier with Spring MVC or Struts
- Validating user inputs using Spring
- Aspect-oriented programming
- Remoting and clustering Spring applications
- Applying security using the Acegi security framework



Course Outline

Chapter 1: Architecting JEE Systems

- Evolution of JEE Systems, leading to Spring
- EJB issues: performance, maintainability and testability
- How Spring addresses EJB issues
- The JEE 3-Tier Architecture
- Core JEE Design Patterns
- Quick introduction to Eclipse, Ant and Tomcat
- **Exercise 1: Exploring the case study**
- Unit testing with JUnit
- **Exercise 2: Unit testing**

Chapter 2: Inversion of Control using Spring

- Designing to Interfaces
- Factory Design Pattern
- **Exercise 3: Implementing a Factory**
- Inversion of Control
- Spring's core IoC container
- BeanFactory
- XML configuration
- Differences between Spring 1.2 and Spring 2.0.
- **Exercise 4: Spring as a Factory**
- ApplicationContext
- Registering a shutdown hook
- **Exercise 5: Bean destroy**

Chapter 3: Dependency Injection using Spring

- Dependency Injection
- Injecting values
- Try It Now: Dependency Injection
- PropertyEditor
- Try it Now: Property Editor
- Invoking constructor
- Constructor or setter?
- Instantiating beans using factory methods
- **Exercise 6: TicketXMLService**

Chapter 4: Database Access

- Limitations of JDBC
- How Spring improves on JDBC
- JdbcTemplate
- Queries and callbacks
- Mapping rows to objects
- **Exercise 7: Implementing and testing a Spring DAO**
- Modeling DAO operations as classes
- SqlUpdate and MappingSqlQuery
- Try It Now: Improved DAO
- Support for Hibernate and JDO
- Try It Now: Using Hibernate from Spring

Chapter 5: Spring Web MVC and/or Struts + Spring

- Model 2 architecture
- Spring's implementation of MVC
- Maintaining user state



- Session Façade
- **Exercise 8: Implementing a TicketingSessionFacade**
- DispatcherServlet
- Convention over Configuration
- Controllers and Mappings
- Reusing bean definitions
- Resolving views
- Using Spring with Struts, JSF or other web frameworks
- Configuring Spring Actions as Spring Beans
- Injecting Spring-controlled objects into Struts Actions
- **Exercise 9: Implementing the museum web application with Spring MVC OR Struts**

Chapter 6: Validation using Spring

- AntPathMatcher, StringUtil, Stopwatch
- Resource and resource implementations
- Message Bundles
- Validation
- **Exercise 10: Validating exhibit name in controller**
- Try it now: Validation in middleware

Chapter 7: Advanced Spring concepts

- Autowiring
- Try it now: Autowiring
- Checking dependencies
- Factory aware, bean postprocessing, lookup method injection
- Try it now: Bean Factory Aware
- Try it now: Lookup method injection
- Aspect-oriented programming
- ProxyFactoryBean
- Intercepting methods
- Applying advice
- Try it now: Aspect-oriented programming
- Pointcuts
- Transactions are ACID
- Programming for transactions
- Declarative transactions in Spring
- Try it now: Declarative transactions to prevent accidental updates

Chapter 8: Remoting and clustering Spring applications

- Need for remoting
- Clustering a Spring application across tiers
- Remoting with RMI, HTTP, web services or Hessian/Burlap
- Exposing beans via HTTP
- Accessing services via HTTP
- Try it now: Using HTTP invoker to expose and call remote services
- Scheduling tasks
- **Exercise 11: Remoting a session façade and accessing it from a GUI**

Appendix A: The Acegi Security Framework

- How to secure a web application declaratively
- Limitations of JAAS
- Intercepting web requests
- Authenticating against a database



- Role-based authentication
- Pattern-based authorization
- **Exercise 12: Securing a web application**

Appendix B: The Hibernate Persistence Framework

- Domain Store Design Pattern
- Persisting Java Beans with Hibernate
- Hibernate Query Language
- Try it now: Hibernate without Spring
- Configuring and invoking Hibernate from Spring
- **Exercise 13: Using Hibernate+Spring to persist a simple POJO**
- Persistence requirements
- Implementing equals()
- Entities and value types
- Mapping collections and associations
- One-to-many, Many-to-one, one-to-one, many-to-many
- Generating IDs
- Mapping inheritance hierarchies
- Choosing the inheritance mapping strategy
- **Exercise 14: Using Hibernate+Spring to persist an inheritance hierarchy**
- Short-lived and long-lived transactions
- Optimistic transactions with version
- Pessimistic transactions with detached objects
- Try it now: Declarative transactions with AOP + Spring + Hibernate
- Setting Hibernate properties
- Native and custom SQL
- Optimizing performance

Appendix C: Spring Best Practices

- Good OO and distributed design practices
- Testing with wired up beans
- Testing with mock beans, pros and cons
- Try it now: Testing a session façade with mock dependencies
- Maintainable bean definitions
- When to inject and when not to inject
- Thread-safety in Spring beans

Appendix D: For Future Exploration

- Extensions
- References

Please contact your ROI representative to discuss course tailoring!!!