

## Developing Enterprise Java Applications with Spring and Hibernate®: Hands-On - 4 Days

*Course 517 Overview*

- You Will Learn How To**
- Develop scalable enterprise Java applications using Spring 3.1 and Hibernate
  - Build application infrastructure using Inversion of Control (IoC) and Dependency Injection (DI)
  - Modularize functionality using Aspect-Oriented Programming (AOP)
  - Add a flexible application user interface with Spring Model View Controller (MVC)
  - Implement object persistence with Hibernate
  - Optimize data access with Hibernate Query Language (HQL)

**Course Benefits** Developing robust Java enterprise applications is a complex process often requiring extensive infrastructure code. In this course, Java developers learn how to quickly build enterprise Java applications using the industry-standard Spring and Hibernate frameworks. Through intensive hands-on exercises, you learn how to implement high-performance applications while reducing development time.

**Who Should Attend** Architects, programmers, engineers, managers and those involved in the development of complex enterprise-level Java applications. Java programming knowledge at the level of Course 471, "Java Programming Comprehensive Introduction," is assumed.

**Hands-On Training** Exercises provide practical experience building enterprise applications, including:

- Injecting dependencies with Spring IoC
- Streamlining development with Spring 3.1 JDBC template support
- Applying modular code using AspectJ style AOP
- Implementing a Web Tier using Spring MVC
- Managing transactions with Spring 3.1 annotations
- Storing and retrieving data objects with Hibernate
- Integrating Spring and Hibernate

# Developing Enterprise Java Applications with Spring and Hibernate®: Hands-On - 4 Days

## Course 517 Outline

### Introducing the Spring Framework Spring architecture fundamentals

- Identifying Spring application components
- Defining the n-tier application architecture

### Applying Inversion of Control (IoC) and Dependency Injection (DI)

- Delegating object creation to the Spring bean factory
- Controlling bean creation with scopes and factory methods
- Initializing and destroying beans

### Minimizing Code with Aspect-Oriented Programming (AOP)

#### Transparently applying common functionality

- Evaluating the benefits of AOP
- Defining advice, pointcuts and advisors
- Minimizing configuration with Autoproxying

#### AspectJ style AOP

- AspectJ pointcut expression language
- Applying AspectJ style with annotations
- Building aspects with POJOs and XML schema-based configuration

### Constructing an Effective Data Access Tier with Spring

#### Simplifying data access with JDBC templates

- Streamlining runaway code with JDBC templates
- Structuring queries and callbacks for maintainability

#### Abstracting the Data Access Layer

- Supporting the Data Access Object (DAO) pattern
- Achieving implementation independence with platform agnostic exceptions

#### Managing transactions

- Analyzing Java EE transaction support
- Controlling transactions with the Spring transaction manager
- Declaring transaction policies with XML and annotations

### Building a Web Tier with Spring MVC Processing Web requests

- Analyzing Spring Model View Controller (MVC) architecture

- Mapping requests to controllers with annotations
- Processing commands, form submissions and simple wizards
- Server-side validation

#### Rendering the response

- Resolving views with ViewResolvers
- Spring JSP support
- View technology alternatives with Velocity

#### Building Ajax controllers

- Establishing the requirements for Ajax controllers
- Implementing REST-style URLs
- Returning JSON data

#### Persisting Objects with Hibernate

##### Integrating Hibernate

- Simplifying data access with O/R mapping
- Unraveling the Hibernate architecture
- Deploying and configuring Hibernate

#### Generating Hibernate applications

- Developing the persistent class
- Defining the Hibernate mapping rules
- Storing and retrieving Java objects

#### Handling Complex Object Relationships

##### The role of the Hibernate Session

- Establishing a thread-safe session object
- Defining object states: transient, persistent, detached

##### Mapping collections

- Persisting and retrieving collections
- Preserving collection order for data integrity

##### Strategies for building object associations

- Specifying one-to-many and many-to-many relationships
- Controlling the association life cycle

##### Effectively mapping inheritance relationships

- Applying class rules for inheritance
- Techniques for class-database mapping

#### Optimizing Data Access

##### Applying Hibernate Query Language (HQL)

- Selecting and filtering queries
- Improving structure with named queries

- Augmenting HQL with native SQL
- Maximizing Hibernate performance
- Accelerating data access via Hibernate cache

#### Integrating Spring and Hibernate

- Employing the Spring Hibernate template
- Configuring Hibernate resources in Spring