Hands On Technology Transfer
The Best Way to Transfer Technology Skills

Contact Us
800-413-0939

# Introduction to the Spring 4 Framework

**Duration:** 3 Days *(Face-to-Face & Remote-Live)*, or 21 Hours *(On-Demand)*

**Price:** $1695 *(Face-to-Face & Remote-Live)*, or $1495 *(On-Demand)*

**Discounts:** We offer multiple discount options. Click here for more information.

**Delivery Options:** Attend face-to-face in the classroom or remote-live attendance.

## Students Will Learn

- Understanding the core principles of Spring and of Dependency Injection (DI) / Inversion of Control
- Using the Spring Core module and DI to configure and wire application objects (beans) together
- Knowing the different types of metadata (XML, annotations/`@Component`, and Java Configuration/`@Configuration`) and how and when to use them
- Understanding and using the complete capabilities of the Core module, such as lifecycle events, bean scopes and the Spring API
- Working with the ORM (Object-Relational Mapping) module to integrate Spring with technologies such as Hibernate or JPA
- Understanding and using Spring's powerful AOP capabilities for programming cross-cutting concerns across multiple points in an application
- Learning safe and maintainable techniques for programming with AOP
- Understanding and using Spring's transaction support, including the easy-to-use Java annotation support, as well as the tx/aop XML configuration elements
- Integrating Spring with Java EE Web applications

## Course Description

This course introduces the techniques for using the powerful capabilities of Spring 4 including the three main configuration styles: Java-based (`@Configuration`), annotation-based (`@Component`), and the traditional XML-based configuration that may still play an important role in existing and new projects. It also provides guidelines for when and how to use each one.

The course starts with in-depth coverage on using the powerful capabilities of Spring's Core module to reduce coupling and increase the flexibility, ease of maintenance, and testing of your applications. Coverage also includes integrating persistence layers (e.g. Hibernate/JPA) with Spring, using Spring's powerful Aspect Oriented Programming (AOP) to program cross-cutting concerns in a safe and maintainable way and using Spring's declarative transaction capabilities. It also covers integration of Spring with Java EE Web applications.

This course will enable you to build working Spring applications and give you an understanding of the important concepts and technology. Comprehensive hands-on labs provide reinforcement of the topics covered in the course and practical experience deploying solutions.

Students requiring an introduction to JEE Web Development, JDBC, JNDI, and JSP as well as Spring and Hibernate, may want to take the Web Application Development Using JEE, Frameworks, Web Services and AJAX class instead.

## Course Prerequisites

Java SE programming experience and an understanding of object-oriented design principles. Fundamental knowledge of XML is helpful but not required. HOTT's course Java Programming or equivalent knowledge provides a solid foundation.

## Course Overview

### Introduction to Spring

- Overview of Spring Technology
  - Challenges for Modern Applications
  - Motivation for Spring, Spring Architecture
  - The Spring Framework
- Spring Introduction
  - Managing Beans
  - Inversion of Control / IoC, Dependency Injection / DI
  - Configuration Metadata Overview, Configuring Beans (XML)
- The Spring Container
  - Overview of the Spring Container
  - `ApplicationContext` Overview
  - `ClassPathXmlApplicationContext`, `FileSystemXmlApplicationContext`, `AnnotationConfigApplicationContext`
  - API and Usage
- Dependencies and Dependency Injection (DI)
  - Examining Dependencies
  - Dependency Inversion
  - Configuration and Usage of Dependency Injection (DI) in Spring

### Configuration in Depth

- Annotation Driven Configuration
  - JSR 330 (`@Named`) and Spring (`@Component`) Annotation Styles
  - `@Named`/`@Component`, `@Inject`/`@Autowired`, `@Repository`, `@Service`
  - Configuring Beans and Autowiring with Annotations
  - Enabling Annotations - `context:component-scan`
- Java Based Configuration (`@Configuration`)
  - Overview - code-centric Configuration
  - `@Configuration` and `@Bean`
  - Dependency Injection
  - Resolving Dependencies on Other Beans
  - Injecting Configuration Classes
- Integrating Configuration Types
  - Choosing a Configuration Style
  - Integrating Configuration Styles
  - Importing: `@Import` and
  - Scanning with `@Configuration` style
- Bean Scope and Lifecycle
  - Bean Scope Defined - Singleton, Prototype, and Other Scopes
  - Configuring Scope
  - Bean Creation Lifecycle
  - Lifecycle Callbacks
  - `BeanPostProcessor`
  - Event Handling

### Wiring in Depth

- Value Injection
  - Configuring Value Properties
  - Property Conversions
  - Externalizing Values in Properties Files
- Constructor Injection
  - Constructor Injection Overview
  - Configuration - `@Configuration` and XML

### Database Access with Spring

- Overview of Spring Database Support
- Configuring a DataSource
- Using Spring with Hibernate
  - High Level Hibernate Overview
  - `SessionFactory` configuration
  - `LocalSessionFactoryBean`
  - Contextual Sessions and Spring Integration
- Using Spring with JPA

    - `p:` and `c:` namespaces for XML configuration
- Qualifiers / Domain Specific Language (DSL)
    - Limitations of Autowiring
    - Qualifiers and DSL
    - Creating and Using an Annotation-Based DSL for Bean Configuration
    - Benefits of Qualifiers for Bean Configuration
- Profiles
    - Profiles Overview
    - Configuring Profiles (XML and `@Configuration`)
    - Activating Profiles
- Overview of SpEL

### Aspect Oriented Programming (AOP)

- Overview of AOP
    - Crosscutting Concerns
    - AOP Basics, Aspect, Joinpoint, Advice, Pointcut
- Spring AOP Introduction
    - Configuration - XML and `@AspectJ`
    - Defining an Aspect, Pointcut, and Advice
    - How Advice is Triggered
- Pointcut Expressions and Advice
    - Pointcut Expression Overview
    - The `execution()` Designator
    - Other Designators (within, target, args, `@target`, ...)
    - Kinds of Advice - before, after, around, after-returning, after-throwing
- Marker Annotations (Rubber Stamp AOP)
    - Issue with AOP Configuration
    - Defining an AOP Marker / Rubber Stamp
    - Configuring AOP Using a Marker
    - Advantages of Marker Annotations
- `@AspectJ` Based AOP Support
    - `@AspectJ` Annotations Overview
    - Defining an Aspect, Pointcut, and Advice
- Other Considerations
    - Spring AOP Proxies and Self-Invocation Issues
    - Load-Time Weaving
    - Caveats of AOP

### Web Applications with Spring

- Integrating Spring with Java EE Web Apps
- `ContextLoaderListener`
- `WebApplicationContext`
- Using Spring Beans in Wep App Controller Logic

- Managing the EntityManager (EM)
- `LocalContainerEntityManagerFactoryBean` and Container-managed EMs
- JEE and JNDI Lookup of the EM
- Configuration and Vendor Adaptors
- Creating a JPA Repository/DAO Bean - `@PersistenceUnit`, `@PersistenceContext`

### Spring Transaction (TX) Management

- Introduction to Spring Transaction Management
    - Spring Transaction Managers
    - Spring Declarative TX Management
    - Spring TX Scope and Propagation
    - Spring TX Attributes (REQUIRED, SUPPORTS, etc)
- XML Configuration of Transactions
    - Specifying Advice, TX Attributes, and Methods
    - Linking Advice with Pointcuts
    - Benefits of XML Configuration of TX Behavior

### XML Specific Configuration

- Collection Valued Properties
    - Configuring and Using Lists and Sets
- Factory Classes and Factory Methods
- Definition Inheritance (Parent Beans)
- AutoWiring with XML

Inner Beans
- Compound Names

Hands On Technology Transfer
The Best Way to Transfer Technology Skills

1 Village Square, Suite 8
14 Fletcher Street
Chelmsford, MA 01824