

This course is a new course based on the Spring 3 release. It includes complete coverage of the annotation based approach to configuration and the use of Java-5 capabilities that was first introduced in Spring 2.x, and which has been greatly enhanced in Spring 3. It also provides coverage of the traditional XML-based configuration that can still play an important role in existing and new projects. The course starts with the basics of Spring and in-depth coverage on using the powerful capabilities of the Core module to reduce coupling, and increase flexibility, ease the maintenance, and testing of your applications. It goes on to cover all the important capabilities of Spring 3, including using Spring to simplify the creation of a persistence layer with JDBC and/or persistence frameworks like Hibernate and JPA. It includes coverage of advanced capabilities such as using Spring's Aspect Oriented Programming (AOP) to program cross-cutting concerns such as transactions and security. It provides an introduction to Spring Security v3, its architecture, and how to use it to secure both Web application requests and bean invocations. The course includes integration of Spring with Java EE Web applications, an introduction to Spring's Web MVC, and thorough coverage of Spring Web Flow 2 (which is still the latest version available). Spring MVC is a Web framework based on the powerful Model-View-Controller pattern, and the introduction covers the basics of Spring MVC, and how it supports organizing your Web applications in a highly structured, loosely coupled manner. Spring Web Flow 2 is a Spring framework for defining user interface flow in a Web application. The course includes thorough coverage of Web Flow, including an overview of its capabilities and architecture, defining flows, flow variables and actions, the Unified EL, and flow programming.

**Course Objectives:**

- Understand the core principles of Spring, and of Dependency Injection (DI)/Inversion of Control
- Use the Spring Core module and DI to configure and wire application objects (beans) together
- Understand and use the complete capabilities of the Core module, such as lifecycle events, bean scopes, and the Spring API
- Work with the DAO and/or ORM modules to create a well structured persistence layer with JDBC
- Use Springs Data Integration with JDBC and technologies such as Hibernate or JPA.
- Understand and use Spring's powerful new AOP capabilities for programming cross-cutting concerns across multiple points in an application
- Understand and use Spring's transaction support, including its easy to use tx/aop XML configuration elements and Java 5 annotations
- Integrate Spring with Java EE Web applications
- Understand how Spring MVC works using the new @Controller model, and use it to build basic Web applications
- Understand the basics of Spring Security, and how to secure Web apps and Spring managed beans with it
- Understand and use Spring Web Flow 2 to define complex user interface flow in Web applications.

**Audience:** Java developers who need to work with Spring based applications.

**Prerequisites:** A good working knowledge of basic Java, JDBC, and Servlets/JSP.

**Number of Days:** 5 days

1. **Introduction to Spring**
  - The Challenge of Enterprise Applications
  - Shortcomings of Java/Java EE
  - What is Spring?
  - The Spring Modules
  - The Spring Distribution
  - Spring Introduction
  - Managing Beans
  - A Basic Spring Application
  - Some Bean Classes
  - Configuration Metadata
  - Declaring Beans
  - The Spring Container
  - Working with Spring
  - Why Bother?
  - Some BeanFactory Methods
  - Dependencies and Dependency Injection
  - Dependencies Between Objects
  - Dependency Inversion Principal
  - Dependency Injection (DI) in Spring
  - Dependency Injection Configuration
  - Advantages of Dependency Injection
  - Dependency Injection Reduces Coupling
2. **More about Bean Properties**
  - Working with Properties
  - Configuring Value Based Properties
  - Using Value Based Properties
  - Property Conversions
  - Constructor Injections
  - Constructor Argument Resolution
  - Setter Injection vs. Constructor Injection
  - Collection Valued Properties
  - Working with Collections
  - Configuring <list> and <set> Properties
  - Configuring Collections of Bean
    - References
  - Map Valued Properties
  - java.util.Properties Valued Properties
  - Additional Capabilities
  - Factory Methods
  - Instance Factory Methods
  - Bean Aliases
  - Bean Definition Inheritance
  - Autowiring
  - Autowiring byType
3. **The Spring Container and API**
  - Pros and Cons of Autowiring
  - To Autowire or Not to Autowire
  - ApplicationContext
  - ApplicationContext Interface
  - ApplicationContext Implementations
  - Constructors
  - Using an ApplicationContext
  - Spring Resource Access
  - Built-in Resource Implementations
  - Bean Scope and Lifecycle
  - Bean Scope
  - Specifying Bean Scope
  - Inner Beans
  - Compound Names
  - Depends On
  - Bean Creation Lifecycle
  - Bean Creation Lifecycle Details
  - Using the Lifecycle Interfaces for Beans
  - Bean Destruction Lifecycle
  - BeanPostProcessor
  - @PostConstruct and @PreDestroy
  - Event Handling
  - MessageSources
  - Issues with Messages
  - Resource Bundles
  - Defining Resource Bundles
  - Using Resource Bundles and
    - MessageSource
  - Localization/Internationalization
  - Paramaterizing Messages
  - Annotation Driven Configuration
  - Annotations in Spring
  - Enabling Spring Annotations
  - @Component and Auto-Detecting Beans
  - DI Using @Resource
  - Complete Declarations Using
    - Annotations
  - Other Stereotype Annotations
  - @Resource – Additional Uses
  - @AutoWired
  - Qualifiers
  - Lifecycle Annotations
  - XML Config – Annotations and
    - Scanning
  - Annotation Configuration – Pro / Con

- A Note on the XML Configuration
- A Brief Note on Annotations
- Java-Based Configuration
- Java Configuration overview
- Using Java-Based Configuration
- Dependency Injection
- More on How it Works
- Dependencies Between Configuration Classes
- Other Usage Scenarios
- Classpath Scanning
- Other @Bean Capabilities
- Java-Based Configuration – Pro / Con
- Other Capabilities
- SpEL – Spring Expression Language
- Other SpEL Capabilities
- Validation
- Using Validation
- Configuring Validation
- Validation Constraints
- Additional Capabilities

#### 4. Database Access with Spring

- Issues with JDBC
- Problems Using JDBC Directly
- Let's Review Some Simple JDBC Usage
- Simple Query on the Database
- Problems with the Previous Approach
- Spring Support for the DAO Pattern
- Spring DAO Support
- The Spring Database API
- The JdbcTemplate Class
- The JdbcDaoSupport Class
- DataSources
- Spring Jdbc Exception Hierarchy
- DAO Based on Spring Classes
- Configuring a DataSource
- Looking up a DataSource in JNDI
- Building a DAO Without the Support Class
- Queries and Updates
- Querying with JdbcTemplate
- Mapping Result Rows to Objects
- Defining a RowMapper Class
- Inserting/Updating
- Other Kinds of Query Methods
- The SimpleJdbcTemplate Class

- Hibernate Overview
- Typical Hibernate Configuration File
- Using Hibernate Directly
- Spring Support for Hibernate
- Template Support for Hibernate
- LocalSessionFactoryBean
- Configuring a Hibernate Session Factory
- Contextual Sessions
- Spring Free DAO
- What Approach to Use
- Using Spring with JPA
- Template Support for JPA
- Support for Managing EntityManager
- LocalEntityManagerFactoryBean
- Obtaining an EntityManager from JNDI
- LocalContainerEntityManagerFactoryBean
- Container-Managed EntityManager
- Additional Spring Configuration
- JPA Data Access Object
- Extended Persistence Context

#### 5. Aspect Oriented Programming

- AOP Overview
- The Issue with Crosscutting Concerns
- Crosscutting Illustrated
- Aspect Oriented Programming (AOP) Defined
- Spring AOP Introduction
- Spring AOP with AspectJ Annotations
- Defining an Aspect with @AspectJ
- Defining a Pointcut
- Defining Advice
- Configuring Spring
- A Program that Triggers Advice
- More on How Spring AOP Works
- Pointcut Expressions and Advice
- Pointcut Expressions
- Other Designators Available in Spring AOP
- Combining Pointcut Expressions
- Kinds of Advice
- XML Based AOP Support
- Defining Aspects Using XML
- Specifying Advice with XML
- Other Considerations
- Spring Proxies and Direct Invocation

More on Spring Proxies  
 Issues with AOP  
 Is AOP Worth It  
 Other AOP Capabilities and  
 Functionality

**6. Transactions**

Transaction Managers  
 Configuring Transaction Managers  
 JTA Transaction Manager  
 Spring Declarative Transaction  
 Management  
 Transactional Scope  
 Transaction Attributes for Propagation  
 MANDATORY  
 NESTED  
 NEVER  
 NOT\_SUPPORTED  
 REQUIRED  
 REQUIRES\_NEW  
 SUPPORTS  
 Transaction Attributes – Some Choices  
 Specifying Transaction Attributes  
 Additional Transactional Attributes  
 Rolling Back and Exceptions  
 XML Configuration  
 Specifying Transactions Using XML  
 Linking Advice with Pointcuts  
 <tx:method> Attributes

**7. Web Applications with Spring MVC**

Integration with Java EE  
 Spring and Java EE  
 Java EE Web Applications  
 Web Application Structure  
 Web Application Components  
 ApplicationContext and Web Apps  
 Configuring ContextLoaderListener  
 Using the Application Context  
 Spring MVC Basics  
 What is Spring MVC?  
 MVC Architecture  
 MVC Pattern Flow  
 Spring MVC Architecture  
 Simple Search App Model –  
 Servlets/JSP  
 Simple Search App Model – Spring  
 MVC

DispatcherServlet  
 DispatcherServlet Initialization  
 Controllers  
 Very Simple Controller  
 Control Flow  
 @RequestParam – Parameter Binding  
 Returning Model Data  
 The Associated JSP Pages  
 Other Handler Method Capabilities  
 View Resolvers  
 Controller with Logical Names  
 Other View Resolvers  
 Forms and Command Objects  
 Spring MVC Form Tags  
 A JavaBean Command Class  
 Working with Model Classes  
 Request Handling Flow  
 Exposing Reference Data in the Model  
 Pre-Annotation Based Configuration  
 MVC Without Annotations  
 Command Controllers  
 Configuring the Command Controller  
 Working with Forms  
 Defining a FormController  
 Configuring a Form Controller  
 Rendering the Form via Spring MVC  
 HandlerMappings  
 Summary Non-Annotation Method

**8. Spring Security Overview**

Spring Security  
 Spring Web Security – web.xml  
 Spring Web Security – Spring  
 Configuration  
 More <http> Capabilities  
 Other Authentication Providers  
 Method Security  
 Method Security – Annotations  
 Method Security – Pointcut Expressions  
 Method Security – XML Configuration

**9. Introducing Spring Web Flow 2**

Spring Web Flow Overview  
 The Need for Spring Web Flow  
 What is Spring Web Flow?  
 Benefits of Spring Web Flow  
 Overview of Flow in SWF  
 Elements of a Flow

- Defining a Flow
- Defining Flows
- Elements of a Flow Definition
- How a Flow Works – Start State
- How a Flow Works – Transition
- How a Flow Works – End State
- How a Flow Works – Web Pages
- Leaving a Flow
- Basic System Configuration
- Configuring Spring Web Flow
- Configuring a Flow Registry
- Integrating with Spring MVC
- Customizing the Flow Registry
- Complete Application Context Configuration
- Working with Data and Actions
- Data Available to a Flow
- Flow Instance Variables in Flow Definitions
- Binding Model Objects to Views
- View Pages and Model Binding
- Flow Inputs in Flow Definitions
- Actions and Programming in Flow Definitions
- Examining an Action: <set>
- 10. More on Spring Web Flow 2**
- The Spring Expression Language
- EL Syntax
- More of SpEL Expressions
- EL Literals and Implicit Objects
- EL Implicit Objects
- Flow Control Using EL Expressions
- Data Scopes
- Variable Scopes
- POST – REDIRECT – GET Idiom
- Request vs. Flash Scope
- Flow Scope
- Conversation Scope, and Subflows
- Session Scope
- Accessing the Scopes Within a Flow
- Flow Language Elements
- Overview of Language Elements
- Overview of Action Related Elements
- Overview of Other Elements
- <action-state> Routing
- decision-state
- on-\* Actions
- Configuring Security
- 11. Programming with Spring Web Flow 2**
- Creating Custom Actions
- Programming Your Own Actions
- The RequestContext
- The ExternalContext
- Using a Custom Action
- Validation and Error Reporting
- Validating a Model
- Defining a Validator Method in the Model
- Validation and ValidationContext
- The MessageContext
- Using Validation
- Validation at Work
- Defining a Validator Class
- Resource Bundles
- Parameterizing Error Messages
- Converters
- Data Conversion in Spring Web Flow
- Defining a Converter
- Registering a Converter
- Using Subflows
- Branching on Subflow End-State
- Using Input/Output Variables
- Using Conversation Scope
- When to Use Subflows