Spring 3.0 and the Enterprise (5 days)

Spring 3.0 and the Enterprise **is a five day in-depth course geared for experienced Java and Spring developers who need to understand how to use Spring in conjunction with the enterprise resources and technologies available in today's systems and architectures. The course covers a wide spectrum of topics, so students should have a basic understanding of those technologies and resources prior to taking this class.**

The Spring framework is an application framework that provides a lightweight container that supports the creation of simple-to-complex components in a non-invasive fashion. Spring's flexibility and transparency is congruent and supportive of incremental development and testing. The framework's structure supports the layering of functionality such as persistence, transactions, view-oriented frameworks, and enterprise systems and capabilities. Spring's Aspect-Oriented Programming (AOP) framework enables developers to declaratively apply common features and capabilities across data types in a transparent fashion.

As an enabler for the integration of Java applications and enterprise resources, the Spring framework represents a significant step forward. If you want to deliver an enterprise application within the Spring framework, you'll find this course essential.

Note that our Spring Training Series covers the entire spectrum and is highly modularized. As such, we can customize courses to your specific needs. The following is a high-level listing of Spring topics to consider in building your customized Spring training:

- **Core Spring Framework (including Inversion of Control, Dependency Injection, and Aspect-Oriented Programming)**
- **Spring and Persistence (including Spring DAOs, Transactions, and working with JDBC or Hibernate)**
- **Spring Views and Rich Interfaces (working with Spring MVC, Spring WebFlow, Ajax, Struts, or JSF)**
- **Spring Security (including interceptors, authentication managers, access decision managers, and filters)**
- **Spring Integration (powerful framework for implementing message-based workflows)**
- **Spring and Java Messaging Services (working with JMS)**
- **Spring Enterprise Services (working with JNDI, Timers, JMX, and batch)**
- **Spring Remoting (working with RMI, Hessian, Burlap, and HttpInvoker)**
- **Spring-WS (including web service endpoints, marshalling/unmarshalling, and gateways)**
- **Spring and REST (including support for RESTful services)**
- **Spring and EJBs (working with various types of local and remote EJBs)**
- **Spring and Testing (working with JUnit and Mock Objects)**
- **SpringSource Tool Suite (covers tooling as well as use of the SpringSource's tc Server)**

Course Objectives: What You'll Learn

Students who attend **Spring 3.0 and the Enterprise** will leave the course armed with the required skills to design and implement Spring applications that effectively and transparently use various enterprise systems, tools, and technologies. This course provides coverage of the concepts and practices for interacting between Spring and relational databases, security components, distributed resources, web services, messaging, EJB3, and other components.

Working in a dynamic, lab-intensive hands-on coding environment students will learn to:

- **Examine how the Spring framework handles transactions**
- **Understand and work with various options for Enterprise Integration.**
- **Understand and work Spring Security to acquire and process authentication credentials as well as enforce authorization on enterprise resources**
- **Work with various Spring options for interacting with remotes resources, understanding which option is best for the context**
- **Use the contract-first approach to web services to deploy and consume SOAP-based web services**
- **Integrate JMS into the Spring framework to utilize messaging**
- **Using Spring to work with and/or implement EJB3**
- **Interoperate with JNDI, scheduling and JMX**
- **Understand and work with the Spring Batch framework**

The course provides a solid foundation in essential terminology and concepts, extended and built upon throughout the engagement. Processes and best practices are discussed and illustrated through both discussions and group activities.

Attending students will be led through a series of advanced topics comprised of integrated lectures, extensive hands-on lab exercises, group discussions and comprehensive demonstrations. Please see below for additional information about the hands-on lab work.

Workshop Topics Covered / Course Syllabus

Session: Review of Spring 3.0

Lesson: The Spring Framework

- Introduction to Spring
- Goals of the Spring Framework
- Key Features of Spring
- Spring Architecture
- Key Features of Spring
- Configuring Spring
- POJOS and Interfaces – A few Problems
- A Closer Look at POJOs and Interfaces
- Spring is an Object Factory(XML)
- Dependency Injection
- Spring Architecture
- Spring Jars
- Spring DI Container
- Initializing the Container
- Accessing Beans from the Container
- Defining and Naming Beans
- Spring 3 Annotations

Session: Transactions in Spring

Lesson: Data Access and Transactions

- Overview of Data Access Support
- DAO Implementations
- Transaction Support
- Isolation Level Concepts
- Isolation Level - Constants
- Propagation Behavior – Constants 1/2
- Propagation Behavior – Constants 2/2
- 3 Approaches to Transaction Management
- Transaction Config – Needed in All Cases
- Programmatic Transaction
- Spring – Declarative Transactions
- The Benefits of Declarative Transactions
- Benefits of Declarative Transactions in Spring
- Two Forms of Declarative Transactions
- Method 1 – Use the Spring Config File
- Spring Config
- Spring Config – The Needed Schemas
- Spring Config – tx:advice and aop:config
- Configuring the Transaction
- Method 2 – Use the @Transactional Annotation
- Declarative
- Declarative Inside the Code

Session: Spring Integration

Lesson: Enterprise Integration

- Integration Introduction
- Loose Coupling in the Enterprise
- Type-Level Coupling
- System-Level Coupling
- Availability Assumptions Increase Coupling
- Level of Coupling Has an Impact
- Core Integration Style
- Messaging Patterns
- Integration Core Concepts
- Enterprise Integration Patterns – The Parts
- Point-to-Point Channel
- Publish-Subscribe Channel

Lesson: Spring Integration

- Spring Integration Provides Abstract Structure
- Spring Integration
- Endpoint
- Message
- Message Channel

Lesson: Basic Integration

- Messages in Enterprise Integration
- Messages in Spring Integration
- GenericMessage<T>
- StringMessage and ErrorMessage
- MessageBuilder
- Channels in Spring Integration
- Getting the Message to the Consumer
- Selecting the Right Type of Channel
- Working with Spring Channel Types
- Channel Flexibility in Spring
- Message Endpoints in Spring
- Polling or Not?
- Pollers
- Cron Pollers
- Inbound and Outbound Endpoints
- Bidirectional Endpoints

Lesson: Advanced Integration

- Enterprise Integration Patterns
- Channel Adapter: Summary
- Adapter: Supplementary Information
- Messaging Gateway: Summary
- Gateway: Solution
- Gateway: Consequences
- Service Activator: Summary
- Service Activator
- Demo
- Routers
- PayloadTypeRouter
- HeaderValueRouter
- RecipientListRouter
- Custom Logic Router
- Filters
- Defining Filters
- Splitters and Aggregators
- Message Transformer
- <transformer>
- Channel Adapters
- File Output Adapter
- Mail Output Adapter

Session: Spring and JMS

Lesson: JMS Overview

- Java Message Service (JMS)
- JMS Architecture
- The JMS API
- Two Messaging Models
- The JMS Factory model overview
- Administered Objects
- The JMS Factory Model
- JMS Queue Architecture
- More On Point-to-Point (P2P)
- Point-to-Point Interfaces
- Obtaining QueueFactory
- Obtaining Destination and Session
- Sending a message
- Topic Architecture
- More On Publish/Subscribe
- Publish-Subscribe Interfaces
- Message Consumption
- Messages
- Message Header Fields
- Message Servers

Lesson: Spring and JMS

- Spring JMS Access
- Overview of JmsTemplate Methods
- Callback Methods
- Receiving JMS Messages (Synchronous)
- Obtaining the ConnectionFactory
- Message Converters
- MessagePostProcessor
- Destinations

Session: Working with Enterprise Services

Lesson: Spring Enterprise Services

- Spring and JNDI
- Spring Container External objects
- Using JNDI in Spring -- Good
- Using JNDI in Spring -- Better
- Using JNDI in Spring -- Best
- Injecting JNDI Resources

- Timers
- Creating the Java Timer Task
- Using the Java Timer
- Sample Task
- Registering Sample Task
- ScheduledTimerTask
- TimerFactoryBean
- MethodInvokingTimerTaskFactoryBean
- Using Quartz
- Quartz Jobs
- Subclassing QuartzJobBean
- Quartz Triggers
- Using the Quartz Factory
- Spring and JMX
- Exporting beans as MBeans
- Spring Timers

Session: Spring Remoting

Lesson: Spring Remoting
- JEE Objects ... or Not
- Spring Remoting
- Spring Remoting Types
- Spring Remoting Core Concepts
- Service Exporter and Proxy
- Sample Service Bean Interface
- Sample Service Bean
- Java RMI Goals
- The RMI System Architecture
- A Remote Method Invocation
- RMI Object Model
- Interface Remote
- Object Serialization
- Storing and Retrieving Objects
- Remoting with Java Serialization: RMI
- Registering the Exporter
- Running the Service
- Spring Remoting in the Web Container
- RMI Clients
- RMI Spring Clients
- Defining the Client Side using RMI
- Create a Client-Server Application Using RMI
- Hessian Remoting
- ServiceExporter for Hessian Beans
- ProxyFactory for Hessian Beans
- Burlap Remoting
- ServiceExporter for Burlap Beans
- ProxyFactory for Burlap Beans
- HttpInvoker Remoting
- ServiceExporter for HttpInvoker Beans
- ProxyFactory for HttpInvoker Beans

Session: Path to Useful Web Services

Lesson: Web Services Overview
- What is a Service?
- Let's Focus on One Definition of SOA
- Architectural Style: Common Framework
- Loose Coupling: Spectrum of Options
- Software Agents: Services
- Interacting: Orchestrated
- SOA is Not Revolutionary
- What is the Difference Between Services and SOA?
- What are Web Services?
- Web Services Characteristics
- Web Services Architecturally
- Web Services Enable Decoupling
- Web Services Challenges
- Spec and Standard Evolution
- Web Services Interoperability Organization
- Basic Profile 1.0 Consists of:
- Additional WS-I Profiles
- .NET Platform
- .NET Web Services
- Java and Web Services

Lesson: Web Services, Java, and JEE
- XML and Java APIs at a Glance
- XML Signature
- XML Encryption
- JAXP
- JAXB
- JAXP, JAXB and Web Services
- Web Services APIs at a Glance
- JAX-WS
- SAAJ
- JAX-WSA and XWSS
- Web Services APIs
- Web Services for JEE (JSR109)
- JEE and Web Services
- Web Services Metadata
- Web Services Stacks at a Glance
- Apache Axis 2.x
- JBossWS
- WSIT/Metro
- WebSphere WS
- WebSphere/RAD Web Service Stacks
- Spring-WS

Session: Spring Web Services
Lesson: Spring Web Services
- Spring-WS
- Typical Process for Using Spring-WS
- MessageDispatcherServlet
- Configuring MessageDispatcherServlet
- The Spring-WS Beans Config File
- Endpoints and Endpoints

Lesson: Implementing Spring-WS
- Server-Side Web Service Components
- Spring Endpoints
- Building Endpoints
- Echo Endpoint (JDOM)
- Registering the Endpoint in the web.xml
- Registering the Endpoint in the Spring Config File
- Endpoint Mapping
- PayloadRootQNameEndpointMapping
- SoapActionEndpointMapping
- End Point Interceptors
- PayloadLoggingInterceptor
- DynamicWsdl11Definition
- Spring-WS Client Development
- WebServiceTemplate
- WebServiceTemplate Methods
- Client Code
- Spring-WS Web Service
- Marshalling the Service Document
- Marshaller Interface
- UnMarshaller Interface
- Web Services Server-side
- Simple Marshalling
- Client-Side Web Services
- Web Services Client-side
- WebServiceGatewaySupport
- Configuring WebServiceGatewaySupport

Session: Enterprise Spring Security

Lesson: Enterprise Spring Security
- The Main Goals of Security
- Core Security Concepts
- Spring Security Framework
- Spring works well with the following
- Other Spring Features
- Spring Security Transparent to Client
- Spring Security Works by Interception
- Security Interceptors – Function and Types
- Authentication Managers
- Performing Authentication
- Authentication Providers Supplied by Spring
- Single Sign-on (SSO)
- Protecting Authentication Data
- Username/Password Creation Process
- Authentication Process Compares Credentials
- Wiring in Encoders and Salts
- Configuring an DAO Provider
- Access Decision Managers
- Votes and Voters
- Access Decision Manager
- Run-As and After-Invocation Managers

Session: Introduction to Spring Batch

Lesson: Spring Batch
- Batch Introduction
- Usage Scenarios
- Batch Vocabulary
- Batch NameSpace

- Job Repository
- Job Repository In-Memory
- Job Repository RDBMS
- Job Repository RDBMS Creation
- Getting Started - Batch
- Chunk Processing Flow
- Job
- Job and JobInstance
- ItemReaders
- Reading Flat Files
- Reading Flat Files -LineMapper
- Default Line Mapper
- FlatFileItemReaders
- ItemWriters
- Custom ItemWriters
- Building Jobs
- Job Launcher
- Bring It all together
- Basic Batch
- Job Listeners
- Registering Job Listeners
- Job Inheritance
- Job Restartability
- Tasklet Oriented
- Building a Tasklet
- Sample Tasklet
- Registering the Tasklet
- Scalable Batch Processing
- Multi-Threaded Step
- Parallel Steps
- Scheduling Batch Jobs
- Java Timer
- Quartz Timer
- Spring Batch Admin
- Installing Batch Admin
- Configuring Batch Admin
- Batch Admin home
- Monitoring Jobs with Batch Admin
- Controlling Executions with Batch Admin
- Uploading Jobs with Batch Admin

Session: Spring and EJBs

Lesson: EJB Overview
- Defining Enterprise JavaBeans
- Compliant J2EE Framework Ready for an Application
- JavaBeans™ vs. Enterprise JavaBeans™
- EJB Architecture Overview
- EJB Container
- Types of EJBs
- Enterprise Beans
- Session Bean
- Entity Bean
- Message-Driven Bean
- EJBObject/EJBLocalObject
- Home Object (EJB Factory)
- Deployment Descriptor
- EJB-Jar File

Lesson: Spring and EJBs
- Home Object Lookup
- Local Stateless Session Beans
- Remote Stateless Session Beans
- Accessing Both Local and Remote EJBs
- Implement EJBs Using Spring
- AbstractStatelessSessionBean
- Loading Spring's Application Context (XML)
- AbstractStatefulSessionBean
- AbstractMessageDrivenBean/ AbstractJmsMessageDrivenBean
- Singleton Context
- BeanFactoryLocator Implementations
- Sharing Contexts in EJBs
- beanRefContext.xml

3373 /h13