



# Open Source Live!

*I HEAR and I FORGET. I SEE and I REMEMBER. I DO and I UNDERSTAND.*

## Developing Enterprise Java Applications with Lightweight Frameworks

**Denver Course Details - \$300 OFF Super Special ends 7/11/07**

### Day 1: Track A Java Server Faces

This JSF course begins by explaining what JavaServer Faces is, and how it relates to Struts and other web frameworks currently on the market. You will learn about key JSF concepts, and some of the architectural principals behind the framework. You will then learn about IDEs that support JSF, as well as libraries that facilitate JSF development. You will also learn about the current market for off-the-shelf user interface components, such as grids, menus, toolbars, trees, and tabbed panes. We'll also examine the extension points within JSF, and how they can be leveraged to provide features such as security, alternate templating technology, and access to external resources.

- Introduction to JSF
- The Java web development landscape
- What is JSF?
- JSF fundamentals

#### Exploring the JSF landscape

- JSF implementations
- Third-party components
- Development tools
- Other frameworks and libraries

#### Getting Started

### Day 1: Track B Spring 2 Framework

The Spring is a very popular light-weight framework for building highly functional and easy to maintain Java SE and Java EE applications. This session will give you comprehensive understanding of Spring architecture and you'll acquire a deep understanding of Spring's dependency injection, AOP, Spring MVC and Spring persistence. Since every topic has a corresponding hands-on lab associated with it, at the end of the session, you will have acquired the know-how and confidence to start implementing application using Spring framework.

#### Introduction to Spring framework (1 hour lecture + 1 hour lab)

- What is & Why Spring framework?
- Spring framework architecture overview
- Usage scenarios
- Dependency Injection

#### Lab 1: Refactoring with Dependency Injection

- Dependency Injection with simple values
- Dependency Injection with Ref
- Dependency Injection with Naming
- Dependency Injection with Autowire

- Setting up your environment
- Structuring the application
- Developing views
- Navigation

Lab: Configuring a JSF application

- Get a JSF application to run
- Create a static login page
- Configure basic navigation

Creating Backing Beans and using Managed Beans

- The JSF Expression Language
- Managed Beans
- Backing Beans

Lab: Creating Backing Beans and using Managed Beans

- Configure application objects
- Write the AuthenticationBean backing bean
- Configure Authentication bean

Spring framework (0.5 hour lecture + 0.5 hour lab)

- Application Context
- Property Placeholder Configurer
- Property Editor
- Factory Bean

Lab 2: Dependency Injection with event notification

- Dependency Injection with Context
- Dependency Injection with Singleton and Non-singleton
- Dependency Injection with PropertyEditor

Spring MVC (1 hour lecture + 1 hour lab)

- What is and Why Spring MVC?
- Request life-cycle
- Dispatcher Servlet
- URL Handler mapping
- Controllers
- View & View Resolvers
- Validation
- Interceptors

Lab 3: Building Spring MVC application

- Using SimpleUrlHandlerMapping
- Using InternalViewResourceResolver
- Using Tiles
- Various Spring MVC applications

Spring AOP basics (0.5 hour lecture + 0.5 hour lab)

- Why AOP?
- AOP concepts
- Spring AOP

Lab 4: Building a Spring AOP application

- Building a simple Spring AOP application
- Building AOP application using before and after advices

Spring and Persistence

- DAO interface
- Spring DAO
- Spring and Hibernate integration

#### Lab 5: Implementing DAO

- Usage of DAO injection
- Usage of PropertyPlaceholderConfigurer

**Register Now!**

### Day 2: Track A Advanced Java Server Faces

Exploring the standard components

- Working with the standard components
- Using basic components and HtmlPanelGrid
- Using HtmlSelectItems
- Using HtmlDataTable

Internationalization, validators, and converters

- Internationalization
- The standard converters
- The standard validators

Lab: Working with the standard components

- Add JavaScript to the login page
- Develop the inbox page
- Write a backing bean

Developing with JSF

- Key APIs
- Building an application
- Best practices
- Security

### Day 2: Track B Hibernate 3

Hibernate is one of the most popular open source O/R frameworks. At the end of this session, you will have the know-how to implement persistence using Hibernate. This session will start with the basic concept and architecture of the Hibernate framework and a good understanding of how to implement DAO, transaction, queries and caching. You will also learn how to integrate Spring framework with Hibernate for implementing persistence. Every topic has a corresponding hands-on lab so you'll acquire the practical skills.

Hibernate Step by Step (0.5 hour lecture + 0.5 hour lab)

- Creating POJO class
- Creating mapping files
- Creating Hibernate configuration file

Lab 1: Your First Hibernate Application

- Building a simple Hibernate application

Hibernate Basics (0.5 hour lecture + 0.5 hour lab)

- Why use O/R Mapper?
- Hibernate architecture
- Instance states
- Persistence life-cycle operations

## Inside the JSF Architecture

- Using Phase Listeners and Pluggable Extensions

### Lab: Using Phase Listeners and Pluggable Extensions

- Write an authorization phase listener
- Write a custom navigation handler

### Writing custom components, validators, and converters

- Developing custom components
- Developing converters and validators

### Wrap up & Future Directions

- DAO
- Transaction

### Lab 2: Building Hibernate application using DAO interface

- Exercising various life-cycle operations
- Generating different primary key types
- Exercising sessions

### Hibernate Mapping (0.5 hour lecture + 0.5 hour lab)

- Mapping cardinality relationships
- Mapping inheritance relationships

### Lab 3: Building an application using O/R Mapping

- Building an application using One to Many Mapping
- Building an application using One to Many with List
- Building an application using One to Many with Array
- Building an application using One to Many with Bag
- Building an application using Single table inheritance strategy
- Building an application using Joined table inheritance strategy

### Hibernate Query Language (0.5 hour lecture + 0.5 hour lab)

- Criteria API
- Hibernate Query Language (HQL)
- Native SQL Query

### Lab 4: Implement Query features

- Building applications using various Criteria API such as Condition, Associations, Sorting, Projection, and paging
- Building applications using features of HQL
- Building applications using native SQL query

### Hibernate Caching (0.5 hour lecture \_ 0.5 hour lab)

- What is Caching?
- Caching implementations
- Caching strategies

### Lab 5: Implement Caching

- Building and measuring the performance of an application using 2nd-

# Register Now!

## Day 3: Track A

### AJAX

Web 2.0 is all about building lightweight rich, interactive experience and AJAX (Asynchronous JavaScript + XML) is one of the key technologies that makes the web more interactive and exciting. This session will first give you a basic understanding of JavaScript, CSS, DOM, and XMLHttpRequest to build a solid foundation. Next, the session will focus on how to use various toolkits such as Dojo and GWT to build Ajax application with detail discussion using an example code. You will also learn how to implement and how to effectively use jMaki to create dynamic web application. There'll be plenty of hands-on labs to you'll acquire the skills to start implementing Ajax application.

Ajax Basics and development tools (0.5 hour lecture + 0.5 hour lab)

- Rich Internet Application (RIA) technologies
- Ajax real-life examples and usage cases
- What is and Why Ajax?
- Technologies used in Ajax
- Anatomy of Ajax operation
- XMLHttpRequest methods and properties
- DOM APIs and InnerHTML
- Ajax security
- Debugging tools

Lab 1: First AJAX Sample app

- Build, run, and debug "Data Validation with AJAX" sample application
- Use "Firebug" Debugger
- Build, run two other AJAX sample applications
- Use innerHTML (instead of tedious DOM APIs) for setting inline contents
- Use JsUnit for Unit testing

Dojo Toolkit (1.5 hour lecture + 1.5 hour lab)

- What is and Why Dojo toolkit?

## Day 3: Track B

### Maven

Maven 2 is a popular open source build tool for enterprise Java projects that take much of the hard work out of build process. In this class you will learn how to use Maven to build better software. You'll learn all about versioning, configuration management, dependencies, testing resources, team members etc contained in project object model (POM). When you finish this intense one day session you will have a broad understanding, knowledge and a working example of how to put Maven 2 to work on your next project.

### Introduction to Maven 2

- Untangling crosscutting concerns
- What is Maven and what does it provide
- Major Principals
- Convention over Configuration<
- Reuse of Build Logic
- Coherent Organization of Dependencies
- Getting Started
- Your First Maven Project
- Using mvn
- Using Maven Plugins
- **Lecture 2: Using Maven to Build your Apps**

### Lecture 2: Using Maven to Build your Apps

- The Build Phases
- Project Directory Structure (Convention over Configuration)
- Project Inheritance
- Managing Dependencies
- Build Phases and the Lifecycle
- Profiles
- Deploying
- Introduction to Building a Project Web Site with Maven
- **Lab 2: The whole cycle, builds, test, deploy, repeat**

### Lecture 3: Building Java EE Applications with Maven

- Dojo toolkit package system
- Remoting via dojo.io.bind
- Backward/forward buttons
- Dojo event system
- Widgets

#### Lab 2: Build apps using Dojo toolkit

- Modify "Data Validation with AJAX" sample application to use dojo.io.bind() call
- Dojo event model
- Usage of Dojo Widgets
- Dojo and JSON
- Modify "Dojo and JSON" sample application from Exercise 4 to use Dojo Fisheye widget

#### Direct Web Remoting (1 hour lecture + 1 hour lab)

- What is and Why DWR?
- Steps for building DWR-based Ajax application
- Callback functions
- Utility functions
- Engine functions
- Error handling
- Security
- DWR and Web application frameworks

#### Lab 3: Build four DWR Sample Apps

- Build and run "Chat Demo" sample DWR application
- Build and run "Dynamic Form Editing" sample DWR application
- Build and run "Dynamic Table Editing" sample DWR application
- Build and run "Dynamically Populating a Selection List" sample DWR application

#### Google Web Toolkit (1 hour lecture + 1 hour lab)

- What is and Why GWT?
- GWT Widgets
- Event handling
- Styling
- Remote Procedure Call (RPC)

#### Lab 4: Build ten apps using GWT

- Installation and Configuration

- Web Applications
- Enterprise Bean Applications
- Building an EAR
- Deploying Java EE Applications
- Improving Productivity with the Jetty 6 Plugin
- **Lab 3: Build, Deploy, Repeat - Java EE Style**

#### Lecture 4: Understanding Project Health with Maven

- Adding and Configuring Reports
- Static Code Analysis
- PMD and Checkstyle
- Testing Reports
- Cobertura
- JUnit Reports
- TestNG Reports
- Dependency Reports
- JXR Reports
- Tag List
- Release Notes
- **Lab 4: Reports and Project Health**

- Build and run HelloWorld sample applications
- Build and run "Hello" and "Kitchen Sink" sample applications, Add CSS style
- Create custom widgets
- Invoke remote service via RPC
- Build and run "Dynamic Table" sample application
- Build and run "JSON" sample application
- Build and run "Mail" sample application
- Build and run "SimpleXML" sample application
- Build and run "HistoryExample" sample application
- JavaScript Native Interface (JSNI)

jMaki (0.5 hour lecture + 0.5 hour lab)

- What is and Why jMaki?
- jMaki widgets
- Event model

Lab 5: Build apps using jMaki

- Explore jMaki widgets using jMaki demo application
- Build your own application using various jMaki widgets
- Handling jMaki widget events using Publish and Subscribe mechanism
- Understanding List widget
- Creating your own HelloWorld jMaki widget

**Register Now!**

**Developing Enterprise Java Applications with Lightweight Frameworks**