

APPLICATION SERVERS

Revised 11/10/2011

/training/etc

The Art of Knowledge.

This Page Intentionally Left Blank

Table of Contents

WebLogic

Developing Java EE Web Applications Using WebLogic™ 10.....	1
WebLogic™ 10 System Administration.....	2

WebSphere

Developing Enterprise Java Beans Using WebSphere.....	3
Introduction to Struts 1.2 using Rational Application Developer (RAD) v7.....	4
WebSphere Business Modeler: Process Mapping, Simulation and Analysis.....	5
New Features of RAD v7 for WSAD v5 Developers.....	6
Developing Java EE Web Applications using RAD v7.....	7
JMS Programming for WebSphere MQ.....	8
Developing Enterprise JavaBeans Using RAD V7.....	9

JBoss

JB295 JBoss Enterprise Application Development.....	10
JB297 JBoss Hibernate Technology.....	11
JB311 JBoss Seam Application Development.....	12
JB325 Advanced JBoss Enterprise Development.....	13
JB336 JBoss Application Administration.....	14
JB346 Advanced JBoss Administration.....	15
JBOSS Enterprise BRMS Implementation (JB433)	16

Ruby on Rails

Ruby on Rails 3.....	17
----------------------	----

This Page Intentionally Left Blank

Course Description: This course is an introduction to writing JEE-compliant Web applications using Oracle WebLogic Server 10.x and Oracle Workshop for WebLogic. An overview of JEE technology is provided, followed by hands-on experience with JNDI, JDBC, Java servlets, and JavaServer pages. Other topics covered include servlet filters, custom JSP tags, JavaMail, and an overview of JavaServer Faces (JSF).

Who Should Attend: Experienced Java programmers and software engineers preparing to write components for JEE Web applications hosted on Oracle WebLogic Server.

Prerequisites: Students should be comfortable with Java programming and object-oriented concepts. A minimum of six months coding experience is suggested. In addition, students should be familiar with writing simple Web pages using HTML. Prior experience using SQL and/or JDBC will be helpful.

Benefits of Attendance: Upon completion of this course, students will be able to:

- Start, stop, and configure WebLogic Server
- Write, deploy, and test Java EE components using the Oracle Workshop development tool
- Use JNDI to access JDBC data sources
- Write and deploy servlets and JavaServer pages on WebLogic server
- Use JDBC to read and update a database
- Create and process HTML forms
- Work with cookies and HTTP sessions
- Assemble and configure a J2EE-compliant Web application
- Use servlet filters for pre- and post-processing HTTP requests
- Create custom JSP tags
- Use the JavaMail API to send email from Web applications
- Create security principals and roles
- Apply security to Web pages
- Implement the Model View Controller architecture
- Write simple applications using the JavaServer Faces framework

Course Outline:

Overview of Java EE

Java Editions
Characteristics of "Enterprise" Computing
Java EE Technologies
Multi-Tier Architectures
Advantages of Multi-Tier Architectures
Container-Based Approach
Java EE Application Models
Java EE Web Application Model

Introduction to Oracle Workshop

What is Oracle Workshop?
Starting Oracle Workshop
Configuring the WebLogic Test Environment
Starting and Stopping WebLogic Administration Console
Setting Preferences in Oracle Workshop
Creating a Dynamic Web Project

Servlets

A Simple Servlet
Web Applications
Configuring Servlets
Running Servlets in Oracle Workshop
Servlet Initialization Parameters
Generating and Validating Forms
Servlets and Threads
Other Settings in web.xml
Creating a New Servlet

Session Management

Cookies
Sessions
Session ID's
Session Management
Session Management Example
Invalidating Sessions
Configuring the Session Timeout

JavaServer Pages

JavaServer Pages
A Simple JSP
Running JavaServer Pages in

Oracle Workshop
JSP Syntax
Configuring JavaServer Pages
JSP Directives
JSP Actions
JSP Example with Forwarding
JavaServer Pages and JavaBeans
JSP With JavaBean Example
JavaBean Class
Running the JSP Bean Example
Creating a New JSP

Custom JSP Tags

Using Custom Tags
Types of Tags
Defining Tags
The tag Element
Simple Tags
Simple Tags - Example
Tags with Attributes
Tags with Attributes - Example
Using JSP Expressions as Attributes
Including the Tag Body
Including the Tag Body - Example
Optionally Including the Body
Including the Body Multiple Times
Including the Body Multiple Times - Example
Running the Examples

Web Application Security

HTML Form for Survey Application
HTML Code for Survey Form
Servlet Code for the Survey Application
JavaBean Class for the Survey Application
Running the Survey Application
Java EE Security
Users and Groups
Adding Users in the Administration Console
Authentication
Configuring Authentication for Web Applications
Authorizing Access to Resources in a Web Application

Web Application Security – Example

Java Naming and Directory Interface

What is JNDI?
Benefits of JNDI
Naming Services
Directory Services
Using JNDI
Context Operations
JNDI Utility Class
JNDI Example
Running the JNDI Example
Naming Exceptions
Creating a New Standalone Program

Database Access using JDBC

A Simple JDBC Program
JDBC Driver Types
Connection Pools
JDBC Data Sources
Data Source Example
Configuring JDBC Data Sources
Running the JDBC Examples
Using JDBC in a Servlet
Using JDBC in a JSP

Design Concepts for Web Applications

Architecture and Design
Tiered Architectures
Model-View-Controller Architecture
Java EE Design Patterns
Composite View Pattern
Composite View Strategies
Running the Demo Application
Composite View Pattern – Implementation
View Helper Pattern
View Helper Pattern – Implementation
Front Controller Pattern
Front Controller Pattern – Implementation
Intercepting Filter Pattern

Servlet Filters

What is a Filter?
Sample Filter
The Filter API
Initializing Filters
Blocking the Response
Modifying the Response
Running the Filter Examples
Creating a New Filter

JavaMail

JavaMail
Example - Send Mail
Example - Read Mail
Running the Examples

JavaServer Faces

What is JavaServer Faces?
JSF Development Roles
Developing a JSF Application
Validators
Example – Creating a Form
Running the First Example
Backing Beans
Example – Processing a Form
Backing Bean Class
The faces-config.xml file
Running the Second Example

Appendix A: Web Resources

Java Technology
WebLogic

Appendix B: HTML Reference

Introduction
A Simple HTML Document
Basic Tags
Formatting Tags
Links
Forms

Appendix C: Web Accessibility

What is Accessibility and Why It is Important?
What is Section 508?
Accessibility Initiatives and Related Legislation

Types of Disabilities
Assistive Technologies
Benefits of Accessible Design
General Coding Practices
Non-Text Elements
Multimedia Presentations
Color and Style Sheets
Image Maps
Tables
Frames, Frequency, and Equivalents
Scripting
Programming
Forms
Repetitive Navigation Links
Timed Responses
Other Recommendations

Appendix D: A JSP Template Mechanism

A Sample Application
A JSP Template Mechanism
Implementing the Template Mechanism with Custom JSP Tags
Classes in the Sample Application
Tag Library Descriptor
Running the Sample Application

Appendix E: Web Services

Service-Oriented Architecture
Web Service Components
Simple Object Access Protocol
SOAP Message Format
The SOAP Envelope
WSDL and UDDI
Example
Creating a Web Services Project
Running the Example
Testing with the WebLogic Test Client

Course Description: This course covers System Administration for WebLogic Server, Version 10. Students will learn to configure, monitor, and tune components for J2EE web and enterprise applications, including JDBC connection pools, JMS destinations, servlets, JSPs, and EJBs. Students will also configure and deploy applications to WebLogic Clusters, including both local and remote server instances.

Who Should Attend: This course is for anyone needing to administer the WebLogic 10 Server in a production or development environment.

Prerequisites: Previous experience using WebLogic and/or developing Java EE applications is required. Java developers should take one of the WebLogic courses (Web Applications or Enterprise JavaBeans) or have equivalent knowledge. System administrators or other non-developers should take the J2EE Technology Overview prior to attending this course.

Benefits of Attendance: Upon completion of this course, students will be able to:

- Describe the directory structures for WebLogic Server and a WebLogic Server domain
- Use the Configuration Wizard to create new domains
- Use the Administration Console extensively to modify and monitor domain and server configurations
- Administer WebLogic services, including JDBC connection pools and multi-pools, and JMS servers, stores, and destinations
- Manage users, passwords, and user lockouts
- Build, deploy, and update web applications, enterprise applications, and EJB components
- Configure and manage WebLogic clusters across a network
- Deploy applications in a clustered environment
- Configure WebLogic Plug-Ins for use with HTTP servers from other vendors
- Perform detailed monitoring of execute queues, JVM performance, connection pools, transactions, JMS destinations, and EJB's
- Use Apache JMeter to load-test a web application
- Tune settings to improve server and application performance

Course Outline:

WebLogic™ Server Basics

Overview of WebLogic
WebLogic Directory Structure
The config.xml File
Starting and Stopping WebLogic
WebLogic Development Environment Setup

The Administration Console

Overview of the Administration Console
Working with the Change Center
Domain Configuration
Servers Configuration
Viewing Log Files
Viewing the JNDI Tree

Configuring WebLogic Services

Connection Pools and Data Sources
Configuring Data Sources
Data Source Settings
Multi Data Sources
JMS Stores
Configuring a JMS JDBC Store
Configuring a JMS File Store
Configuring JMS
JMS Server Settings
Flow Control
JMS Destinations Settings

Security

Security Concepts
Managing Users
Managing Groups
Managing User Lockouts
Setting Minimum Password Length
Java EE Security

Deployments

Java EE Web Applications
Settings in the web.xml File
Example web.xml
Deploying a Web Application
Java EE Enterprise Applications
Settings in the application.xml File
Example application.xml
Deploying an Enterprise Application
The weblogic.Deployer Utility

Introduction to WebLogic Clusters

What is a Cluster?
Communications in a Cluster
Cluster-Wide JNDI Tree
Configuring Clusters
Node Manager
Deploying Applications to a Cluster
Creating a Cluster
Disabling Hostname Verification
Starting the Cluster
Deploying an Application to the Cluster
Testing the Deployment

Clusters and Remote Servers

Adding a Remote Server to the Cluster
Starting the Cluster
Deploying an Application to the Cluster
Testing the Deployment

Web Server Plug-Ins

Overview of Plug-Ins
Configuring IIS as the HTTP Server
Testing the IIS Configuration

Monitoring and Performance Tuning

General Information
Server Performance
User Logins
Transactions
Connection Pools
JMS Destinations
Guidelines for Enterprise JavaBeans
Stateless Session Beans
Stateful Session Beans
Entity Beans
Message Driven Beans
JVM Performance – Sun SDK
JVM Performance - JRockit

Command Line Utilities and Other Tools

The weblogic.Admin Utility
WebLogic Scripting Tool (WLST)
Recording WLST Scripts
Other Utilities

Appendix A: Ant

Ant Basics

Properties
Targets
Tasks
Example Build File
A Few Details
Ant Command Line Options

Appendix B: JMeter

What is JMeter
Creating a Test Plan
Running the Test Plan
Remote Testing

Appendix C: Troubleshooting Lab

Setting Up the Application
Running the Application

Course Description: This is an advanced intro to EJB technology, introducing concepts to those who have never used EJB. Also included are advanced concepts students can use to complete large-scale EJB projects.

Who Should Attend: This course is for intermediate to advanced Java programmers. System architects will find the advanced concepts especially beneficial in designing a framework.

Prerequisites: Students should understand the core Java libraries; be familiar with web application programming; and have experience using IBM RAD.

Benefits of Attendance: Upon completion of this course, students will be able to:

- Understand the EJB 2.1 specification at an expert level.
- Understand how to utilize EJBs with RAD.
- Gain knowledge of J2EE Design Patterns and Best Practices.

Course Outline:

RAD v7

WebSphere Studio Family
Eclipse Platform
WebSphere Studio Product Family
Key Features in RAD v7
WSAD Integration Edition
WebSphere Studio Enterprise Developer
Views, Perspective and Editor Areas
Basic Operations with RAD
Java & Debug Perspectives
Package Explorer
Navigator, Outline, and Task Views
Build and Validation
Templates and Code Completion
Searching
Setup Compiler Class Path
JRE Switching
Import and Export Project
Features of Eclipse 3.0
Eclipse SDK V3.0
JDK 1.4 – New Features
Control Flow of Logging
Logging API

J2EE and Rational Platform

Java Web Applications
J2EE Architecture
WAP Model – MVC
Infrastructure for Enterprise Web Applications
IBM WebSphere Platform
Rational Web Developer
RAD and WAS
WebSphere Editions
WAS Services and Architecture
WebSphere Administrative Topology
WebSphere Application Server Components
Server Profile
WAS Administration tools
RAD J2EE Development
RAD Project Structure
WebSphere Test Environment in RAD
EJB Test Client in RAD

Overview

Needs for EJB
Distributed Computing, Transaction, and Security
EJB Remote Method Call
EJB Architecture Components
EJB Client, JAR File, Server, and Container
Session and Entity Beans
EJB Classes and Interfaces
Basic Components of Entity and

Session Beans
EJB Home Interface and Object
EJB Remote Interface
EJB Local Interface
Remote and Local EJB Objects
EJB Implementation Class
EJB Container - Relationships
Remote v. Local EJBs
EJB Application Development
Deploying Enterprise Beans
Major Components of Deployed EJBs

Session Bean

Home, Remote, and Local Interfaces
Session Bean Class
ejbCreate() Method
Business Methods
EJB Context
Session Bean Lifecycle
Lifecycle - Stateless and Stateful
Concurrency Issues
Invoking Session Beans from client Application
Home Object
Create an EJB Object
RAD for EJB Development
Developing a Session Bean using RAD
EJB Project in RAD
Create an EJB project
Create a Session Bean
Code methods
Promote methods to interfaces
Deploying EJBs
Run on server
JNDI Explorer
Test the home and component interface
Exporting EJB JAR
EJB JAR File Structure

EJB 2.1 Changes

Web Service Clients
Component Interface
Web Service Component Interface
Web Service Client View
Web Service References
Web Service Reference
Programming Interfaces
Declaration Of Web Service References
Container-Managed Timer Service
EJB Timer Service
TimedObject Interface
Annotation Based EJB Development
@ejb.create-method
@ejb.finder, @ejb.home,
@ejb.interface, @ejb.transaction
Annotation Based EJB

Development
Annotation Based EJB
Development
Disabling Tag Sets
Enhanced EAR File Editor
Deployment Descriptor Of An Enterprise File
Exporting an Enterprise Application into an EAR File

Entity Bean

Container
Primary Key Class
(Remote) Home Interface
Local Home Interface
Remote and Local Interfaces
Entity Bean Class
Entity Instance
Entity Object Lifecycle
Bean Instance Lifecycle
Persistence
Writing BMP and CMP Beans
Persistent Fields
Writing finder methods
EJB Query Language
Using EJBQL in a CMP bean
Business Methods
EntityContext
Defining and Mapping CMP Fields
Developing Entity Beans in RAD
Adding a CMP Entity EJB
EJB Deployment Descriptor
Add an EJBQL query
Import Database Schema
Copy Database Schema
Creating a database mapping
Choosing a backend
Choosing a mapping type
Specifying the mapping
EJB and EJB 2.0 Relationships
WebSphere Extensions

EJB Exception

Exception Types
Bean Class
Containers and Clients
Standard Exceptions
Local EJBs

EJB Security

Goals of the EJB Security Specification
Declarative EJB Security – Method Permission
Configure Security Roles
Configure Method Permissions
Generated XML in ejb-jar.xml
Unchecked option
Excludes List
EJB Delegation Policy
Configuring EJB Delegation Policy with RAD

Programmatic EJB Security

Message-Driven Beans

Messaging to the Rescue
Message-Oriented Middleware
Messaging Domains
Publish/Subscribe
Point-to-Point
Java Message Service
JMS Programming: Overview
The JMS Interfaces
Integrating JMS and EJB
Durable Subscription
Message-Driven Bean Interfaces
javax.ejb.MessageDrivenBean
javax.jms.MessageListener
javax.jms.Message
Lifecycle
Class MessageCounter
Processing the Message
Deployment Descriptor Entry
Binding the Queue or Topic
JMS in RAD
Configure a Service Integration Bus in RAD
Add Application Server as Bus Member
Create and Configure a Queue
Configure the JMS Resources
Configure Connection Factory
Configure Activation Specification
Configure the Activation for MDB
Transactions and Security
Load Balancing
Clustering and Topics
Clustering and Queues
Poison Messages
Building a Response
Potential Problems
Type Checking and Messages
Testing Message-Driven Beans

Best Practices

Transaction Management
Object Transaction
EJB Transaction Basics
Transaction Propagation
Client, Bean, and Container Managed Transactions
Transaction Outcome
Vetoing a Transaction
Transaction Isolation
CMP EJB Access Intent
Setting Access Intent
Setting Data Source Isolation Level

EJB Design Patterns

Types of EJB patterns
Session and Message Facade Pattern
Session Facade

EJB Command pattern
Generic Attribute Access pattern
Data Transfer Object
Data Transfer Rowset
Version Number pattern
Fast Lane Pattern
Data Access Command Beans
EJBHomeFactory
Business Delegate
Sequence Blocks
Stored Procedures for Autogenerated Keys

Basic Admin.

Administration Roles and Clients
The Console
Creating a Profile
wasprofile command
Managing an Application Server
Enable Server Process Restart
Basic Application Server Configuration
Configure the JVM
Web Container Transport Chain
Configure Transport Chain
Configure Session Management
Configure an EJB Container
Preparing to Host Applications
Resource Scope and Variables
Configure a Virtual Host
Configure a JDBC Provider
Select JDBC Provider Type
JDBC Provider Options
Data Source
WebSphere Data Source
Mail Providers
The Administrative Console
Installing an Enterprise Application into WAS
Saving the Master Configuration
Configuring the Web Server plugin
WebSphere Apps on the file system
The Plugin

Course Description: This course introduces the students to Jakarta Struts 1.2. During the course, students will learn to design and develop Struts based applications. Students learn how to incorporate JSPs, servlets, EJBs and JavaBeans into their design. Corresponding to every chapter, there is a lab reinforcing the concept.

Who Should Attend: This course is intended for programmers and designers who want to design and develop applications using the Jakarta Struts Model-View-Controller (MVC) framework.

Prerequisites: Students should have a good understanding of HTML and object-oriented programming using Java. Students should also have experience with JSPs and servlets with some understanding of JavaBeans or EJBs.

Benefits of Attendance: Upon completion of this course, students will be able to:

- Understand and explain the Jakarta Struts framework
- Design and build Struts based applications
- Use MyEclipse to create a simple Struts application
- Develop a simple custom tag lib
- Use The Struts Tag Lib
- Use the DynaActionForm and Validator feature of Struts 1.2.

Course Outline:

Struts Architecture and Overview

Review: MVC Model
Review: Request, Session and Application
What is Struts?
Struts Flow
Components in Struts
The Model
The View
The Controller
Struts Flow
struts-config.xml
struts-config.xml Content
The <data-sources> element in the struts-config.xml
The <form-beans> element in the struts-config.xml
The <global-forwards> element in the struts-config.xml
The <action-mappings> element in the struts-config.xml
Sample struts-config.xml
The Role of web.xml for the Application
Configuring Struts in the web.xml
web.xml Content
Steps to Configuring Struts in the web.xml

The First Struts Application

The application
The Structure
Setup Needed
Directory Structure
index.jsp
The index.jsp Page
struts-config.xml
MultiplyForm.java
MultiplyAction.java
result.jsp
The result page
ApplicationResources.properties

Development of Struts Applications Using RAD v7

Directory Structure of a Struts Application
RAD Struts Support
Create a Struts Web Application Project
View the Struts Project Structure
Create a Struts JSP
Insert Struts Tags
Creating a Struts Form Bean
Create a Struts Action
Edit a Struts Configuration File

Struts Development Cycle

Gathering Requirements
Defining Screen Requirements
Detailed screen design
Determining the Screen Flow
Defining the ActionMappings in the struts-

config.xml
Struts view components – Data vs. ActionForm Bean
Defining Screen Requirements – define FormBean
Developing the FormBean
Developing the FormBean – The reset() Method
Developing the FormBean – The validate() Method
Developing the FormBean – ActionErrors
Developing the FormBean – Defining the Message Key
Developing the Action Class – action and Action Class
Developing the Action Class
Developing Actions – The execute() Method
Developing Actions – ActionForward
Developing Actions – ActionMapping class
Developing Actions – execute() example
execute() Method Example:
Developing Business Logic – EJBs
Developing JSPs
Configuring struts-config.xml and web.xml
Build, Pack, and Deploy

Struts Tag Libraries

Commonality among the Struts Tags
Bean Tags
HTML Tags
Logic Tags
Logic Tags functionality

Struts Extensions, Internationalization and Error Handling

Struts Extensions
Extension Points
Plug-ins
Custom Configuration Class
Writing a Configuration Class
Custom ActionServlet
Custom RequestProcessor
Base Action Class
Base Form Bean
Custom JSP Tags
Internationalization (I18N)
Error Handling
The ActionError class
Error Handling : validate() method of ActionForm
Displaying the errors found in the validate method
Error Handling : execute() method of Action
Declarative Exception Handling
Syntax of declarative exception handling

Programmatic Exception Handling
Logging from Struts
Using Commons Logging and Log4J in Struts
Writing Commons Logging Code

Miscellaneous Advanced Features

Integration of Jakarta Common Libraries - BeanUtils
Integration of Jakarta Common Libraries - Digester
Multiple Sub-application Support
DynaActionForms
Validators
Adding the Validator framework to a Struts application
A tale of three files
A sample rule in the file validator-rules.xml
validation.xml file
Validator Dependency
Rule Variables
Error Message
Basic Validation Rules
Client Side Validation
Validating multiplication example using DynaActionForm
Writing Custom Validators
Validator Class Example
HTTP Redirection
Working With Check Boxes
Context Sensitive Form Validation

Database Programming

Basic Concepts
MVC Interaction
Database Connection
Transaction Management
Data Source
Defining a Struts Data Source
Opening a Connection from a Struts Data Source
Creating an Editor Form
Example Form Display Action

Templates and Tiles

Struts Templates
Defining the Template
A template
Using the template in a JSP
<template:insert>
<template:put>
A JSP that uses the template
Struts Templates
Templates as UI components
Struts 1.2 Tiles Support
Tiles
A JSP that uses the layout
Tiles go beyond templates

Inheritance in tiles definitions
Definitions as Struts "forwards"
Template or Tiles?

Unit Testing Struts Applications

What is JUnit?
Why JUnit?
A JUnit Test
Running the tests
JUnit Basics
Unit Testing Struts Applications
A simple Struts test case
MockStrutsTestCase Methods
Downloading STC
Testing Strategies

JSP Expression Language and Standard Tag Library

JSP Expression Language (EL)
Basic Usage
Built-in Objects
Working With Arrays and Maps
Operators
Full Example
JSP Standard Tag Library (JSTL)
Run Time Version
Basic Tags
Condition Tags
Iterator Tags
Internationalization (I18N)
Setting Preferred Locale
Specifying Resource Bundle
Display Translated Text
Display Number
Display Date
JDBC Tags
Specify Data Source
Performing a Query
Display Result
Pagination Example

Course Description: This course introduces the participants to Business Modeling. The course also introduces the fundamentals of Service Oriented Analysis and Design (SOAD). The course uses IBM WebSphere Business Modeler Advanced edition as the modeling tool. A well-constructed business process model can help modelers identify inefficiencies and problems with business models earlier in the cycle and eliminate those hidden inefficiencies leading to savings in costs and improving performance.

Who Should Attend: This course is designed for managers, architects and business analysts, who need to model, design, analyze and generate reports for business process at their organizations.

Prerequisites: No technical prerequisites are required for this course. However, it is expected that the audience has some experience with working on IT or other projects.

Benefits of Attendance: Upon completion of this course, students will be able to:

- Understand the need for Business Process Modeling
- Learn to use WebSphere Business Modeler to create Business Models
- Understand the basics of Service Oriented Analysis and Design and relate it to Business Process Models
- Model business processes
- Analyze Business Models
- Perform various activities such as querying and reporting associated with business process models
- Add Key Performance Indicators to a Business Model

Course Outline:

Introduction to Business Process Modeling

IBM Tools for BPM

Using WebSphere Business Modeler Advanced edition

Introduction to Service Oriented Analysis and Design

Business Process Modeling Methodology

Business Process Execution Language (BPEL) and Flow Definition Language (FDL)

Model Elements

Process Modeling

Process Simulation

Analyzing Process Models and Simulations

Creating queries and reports

Exporting Models

Publishing Projects

Versioning Projects

Business Metric Measurement

New Features of RAD v7 for WSAD v5 Developers

Course Description: The course is designed to give information on numerous new features available from RAD 7. After taking the class, students will be able to fully take advantage of WAS v6 and RAD v7.

Who Should Attend: This course is for developers who are currently using WSAD v5.x and are planning to move to RAD 7.

Prerequisites: Previous knowledge of Servlet, JSP and EJB development using WebSphere Studio Application Developer (WSAD) is required.

Benefits of Attendance: Upon completion of this course, students will be able to:

- fully take advantage of WAS v6 and RAD v7

Course Outline:

Introduction to RAD v7.0

The RAD 7 Product
Eclipse Platform
Rational Web Developer
RAD
Key Features in RAD v7.0
Views, Perspective, and Editor Areas
Basic Operations with RAD Views and Perspectives
The Java Perspective
The Debug Perspective
Navigator and Outline View
Package Explorer
Task and Problems View
Build and Validation
Import and Export Project
Code Completion, Templates and Snippets
Searching
Setup Compiler Class Path
JRE Switching
Refactoring
Changing Class, Method, and Variable Name
Moving a Class to a Different Package
Extracting Code to a Method
Pull Up and Push Down Methods
Migrating Workspace from RAD v6
Project Interchange
Migrating J2EE Applications
J2EE Migration Wizard

J2EE Tools

Project Explorer
Servers View
The J2EE and Web Perspective
Create an Enterprise App.
Setup Utility JAR
Create an EJB Project
Create a Web Project
Setup Dependent JAR Files
Create Server Instance and Server Configuration
Configure Server Instance
Add an Enterprise App. Proj. to the Test Server
Start and Stop the Server
Test and Debug a Servlet
Test and Debug a JSP

JDK 1.4 Features

JDK 1.4 New Features
Control Flow of Logging
Loggers
Logging API - Handlers
Logging API Formatters & Log Manager
JDK 1.4 - JAXP
JDK 1.4

JDK 1.4 - Assertions

New Features of Java SE 5

Generics - defined
Multiple Parameter Example
Using SampleGeneric
Using Multiple Parameters
Using Point
Java SE 5 Collection Interfaces
Using generics with Collection classes
Benefits of Generics
Advanced LinkedList Example
Generic Methods
Generic Methods Continued
Boxing and Unboxing
Autoboxing/unboxing
Enhanced for loop
Enhanced for loop in Generics
Enumeration data type
Enums in Java 5.0
Advanced Enums Reverse Lookup
Variable-length argument Lists

Web Application API Changes

Develop a Servlet in RAD
Create a Servlet
Run a Servlet in RAD
Servlet 2.4
New Methods of ServletRequest
SingleThreadModel
RequestDispatcher
Internationalization
Web.xml
Schema for Web.xml
Listeners
ServletRequestListener
ServletRequestEvent
ServletRequestAttributeListener
ServletRequestAttributeEvent
Simple Tag Handler API
Java Simple Tag Handler
The Tag Library Descriptor
JSP Page Using the Custom Action
Working With the Tag Body
JSP Tag Handler
Example: customer.tag File
JSP File Using Custom Tag
Working With the Body
Advantages of Using Tag Files

JSP Expression Language and STL

JSP Expression Language
Basic Usage
Built-in Objects
Working With Arrays and Maps
Operators
JSP STL (JSTL 1.1)

Run Time Version (JSTL 1.0)
Basic, Condition, and Iterator Tags
Internationalization (I18N)
Setting Preferred Locale
Specifying Resource Bundle
Display Translated Text
Display Number and Date, and Result
JDBC Tags
Specify Data Source
Performing a Query

EJB 2.1 Changes

Service Endpoint Interface
Other JAX RPC Artifacts
Developing a JAX RPC Web Services in RAD 6
JAX RPC Client programming Model
Container-Managed Timer Service
EJB Timer Service
Interaction between Timer Service and EJB
Timer Service API
Timer Service Interface
TimedObject Interface
Timer Interfaces
Timer and Transaction
Limitations of EJB Timer Service
EJB QL Enhancements
Annotation Based EJB Development
Annotation Scope
The @ejb.bean, @ejb.home, and @ejb.interface Tags
The @ejb.interface-method Tag
Annotation Based EJB Development
Create an Annotated Bean Class
Enhanced EAR File
Enhanced EAR File Editor
Deployment Descriptor Of An Enterprise File
Exporting an Enterprise Application into an EAR File
Exporting an Enterprise Application

Test-Driven Development with JUnit

Component Testing
Testing Frameworks
JUnit
JUnit TestCase
Assert Statements
TestSuite and TestRunners
JUnit in RAD
Add junit.jar Manually
JUnit in RAD
Component, EJB, and Web Service Testing

Application Profiling

Profiling Tool
Profiling Architecture
Profiling Sets (3)
Enable Profiling and Logging
Profiling a Java Class
Configure Profiling Criteria
Profiling Monitor View
Memory Usage Profiling
Memory Leak Analysis
Transaction-Oriented Memory Leak Analysis
Starting a server in profiling mode
Agent Controller and profiling mode
Collecting object references
Object Reference Table
Execution Flow View
Identify Performance Bottlenecks
Method Invocation Details view
Thread Analysis
Execution Flow - Threads
UML2 Trace Interaction view

Code Review

Rule
Rule Severity Level
Information on a Rule
Analysis Configurations
Configuring Code Reviews and Review Rules
Create User Defined Rule
Select a Template Window
Rule Properties
Perform Automated Code Review
Run Code Review
Code Review Details
Quick Fixes

Code Coverage

Code Coverage Displays Coverage, Package, and Class Statistics
Method Invocation View
Method Invocation Details
Method Invocation Details view
Code Coverage Statistics view
Coverage Statistics View

WebSphere Programming Model Exts. 1

Startup Bean
Creating a Startup Bean
Methods for Startup Beans
Enable Startup Bean Service
Asynchronous Beans
Transactions in Async. Beans
Work Managers
Work Manager Properties
Work Manager Screen
Using a Work Object

Scheduler Service
Configuring Schedulers
Admin Console Configuration
Scheduler Configuration
Scheduler Tasks
Developing Scheduler Clients
Create a Task Handler EJB
Create a Scheduler Bean
Create a Process() Method in an Session Bean
Create an EJB Task
Create a JMS Message Task

WebSphere Programming Model Exts. 2

Work Area
Work Area Service
Work Area Service Settings
UserWorkArea Interface
Work Area Property Modes
Work Area Considerations
Bidirectional Work Areas
Nested Work Areas
Using Work Areas
Completing Work Areas
Object Pool
Object Pool Manager
ObjectPoolManager and ObjectPool Interfaces
Object Pool Service Settings
Configure Object Pool Manager
Custom Object Pools
Sample Code
Application Profile
Access Intent
Application Profile
Unit of Work
Enable Application Profiling

Dynamic Caching for Web Apps.

Configuring Dynamic Cache
How Caching Works
Structure of cachespec.xml
URL Based Caching
Session Based Cache
Cache Priority
Configuring Cache Manager
ESI Cache

Course Description: This course is an introduction to writing Java EE-compliant Web applications using IBM WebSphere Application Server 6.1 and Rational Application Developer 7. An overview of Java EE technology is provided, followed by hands-on experience with JNDI, JDBC, Java servlets, and JavaServer pages. Other topics covered include servlet filters, custom JSP tags, JavaMail, and an overview of JavaServer Faces (JSF).

Who Should Attend: This course is for experienced Java programmers and software engineers preparing to write components for Java EE Web applications hosted on WebSphere Application Server.

Prerequisites: Students should be comfortable with Java programming and object-oriented concepts. A minimum of six months coding experience is suggested. In addition, students should be familiar with writing simple Web pages using HTML. Prior experience using SQL and/or JDBC will be helpful.

Benefits of Attendance: Upon completion of this course, students will be able to:

- Start, stop, and configure WebSphere Application Server.
- Write, deploy, and test Java EE components using the RAD7 development tool.
- Use JNDI to access JDBC data sources.
- Write and deploy servlets and JavaServer pages on WebSphere Application Server.
- Use JDBC to read and update a database.
- Create and process HTML forms.
- Work with cookies and HTTP sessions.
- Assemble and configure a J2EE-compliant Web application.
- Use servlet filters for pre- and post-processing HTTP requests.
- Create custom JSP tags.
- Use the JavaMail API to send email from Web applications.
- Create security principals and roles.
- Apply security to Web pages.
- Implement the Model View Controller architecture.
- Write simple applications using the JavaServer Faces framework.

Course Outline:

Overview of Java EE

Java Platforms
Characteristics of "Enterprise" Computing
Java EE Technologies
Multi-Tier Architectures
Advantages of Multi-Tier Architectures
Container-Based Approach
Parties Involved in Java EE Deployment
Java EE-Compliant Application Servers
Java EE Application Models
HTTP Services Application Model
N-Tiered Application Model

Introduction to RAD

What is WebSphere@?
What is Rational@ Application Developer?
Starting RAD
Starting and Stopping WebSphere
Creating an Enterprise Application Project
Creating a Dynamic Web Project

Servlets

A Simple Servlet
Web Applications
WebSphere Deployment Descriptors
Running Servlets in RAD
Configuring Servlets
Configuring Servlets in RAD
Servlet Initialization Parameters
Generating and Validating Forms
Servlets and Threads
Other Settings in web.xml

Session Management

Cookies
Sessions
Session Id's
Session Management
Session Management Example
URL Rewriting
Invalidating Sessions
Configuring the Session Timeout

JavaServer Pages

JavaServer Pages
A Simple JSP
Running JavaServer Pages in RAD
JSP Syntax
Configuring JavaServer Pages

JSP Directives
JSP Actions
JSP Example with Forwarding
JavaServer Pages and JavaBeans
JSP with JavaBean Example
Running the JSP Bean Example
Creating a New JSP

Custom JSP Tags

Using Custom Tags
Types of Tags
Defining Tags
The tag Element
Simple Tags
Tags with Attributes
Using JSP Expressions as Attributes
Including the Tag Body
Optionally Including the Body
Including the Body Multiple Times
Running the Examples

Web Application Security

HTML Form for Survey Application
HTML Code for Survey Form
Servlet Code for the Survey Application
JavaBean Class for the Survey Application
Running the Survey Application
Java EE Security
Authentication
Configuring Authentication for Web Applications
Authorizing Access to Resources in a Web Application
Web Application Security – Example
Configuring Web Application Security in RAD
Enabling Security for the WebSphere Server

Java Naming and Directory Interface

What is JNDI?
Benefits of JNDI
Naming Services
Directory Services
Using JNDI
Context Operations
JNDI Utility Class
JNDI Example
Running the JNDI Example

Naming Exceptions
Creating a New Standalone Program

Database Access Using JDBC

A Simple JDBC Program
JDBC Driver Types
Using the Derby Database
JDBC Data Sources
Data Source Example
Configuring a JDBC Provider
Configuring a Data Source
Running the JDBC Examples
Using the Database Explorer
Using JDBC in a Servlet
Using JDBC in a JSP

Design Concepts for Web Applications

Architecture and Design
Tiered Architectures
Model-View-Controller Architecture
Java EE Design Patterns
Composite View Pattern
Composite View Strategies
Running the Demo Application
View Helper Pattern
Front Controller Pattern
Intercepting Filter Pattern

Servlet Filters

What is a Filter
Sample Filter
The Filter API
Initializing Filters
Blocking the Response
Modifying the Response
Running the Filter Examples
Creating a New Filter

JavaMail

JavaMail
Example - Send Mail
Example - Read Mail
Running the Examples

JavaServer Faces

What is JavaServer Faces?
JSF Development Roles
Developing a JSF Application
JSF Components

Component Tags
Validators
Example – Creating a Form
Running the First Example
Backing Beans
Example – Processing a Form
Backing Bean Class
The faces-config.xml file

Appendix A: Web Resources

Java Technology
WebSphere
Derby Database

Appendix B: HTML Reference

Introduction
A Simple HTML Document
Basic Tags
Formatting Tags
Links
Forms

Appendix C: Web Accessibility

What is Accessibility?
What is Section 508?
Accessibility Initiatives and Related Legislation
Types of Disabilities
Assistive Technologies
Benefits of Accessible Design
General Coding Practices
Other Recommendations

Appendix D: A JSP Template Mechanism

A Sample Application
A JSP Template Mechanism
Implementing the Template Mechanism with Custom JSP Tags
Classes in the Sample Application
Tag Library Descriptor
Running the Sample Application

Course Description: This course teaches Java programmers how to use the JMS (Java Messaging Service) classes for developing applications in the IBM WebSphere MQ environment. Through lectures and extensive hands-on exercises, the students learn to design, develop and deploy industrial strength messaging applications. The course topics include JMS architecture, point-to-point messaging model, publish/subscribe model, working with queue and message objects, JMS administered objects and many other topics.

Who Should Attend: This course is a must for Java developers who want to learn how to design, implement and deploy JMS application using the IBM WebSphere MQ transport.

Prerequisites: Students should have taken the course, Technical Introduction to WebSphere MQ, plus have Java programming experience.

Benefits of Attendance: Upon completion of this course, students will be able to:

- Understand JMS architecture as implemented in WebSphere MQ
- Work with JMS administered objects
- Develop JMS based messaging applications
- Understand JMS security implementation
- Implement WebSphere MQ specific functions in JMS applications

Course Outline:

What is WMQ

What is JMS?

What is JNDI?

JMS Administration

Examining a JMS Program

JMS and WebSphere MQ Resources

Messages

JMS Message Selection

JMS Message Types

JMS Program Initiation

Request-Reply Pattern

Publish Subscribe Pattern

Transaction Processing with JMS

Course Description: This course is a comprehensive introduction to writing Enterprise JavaBeans (EJB) using IBM WebSphere and the Rational Application Developer (RAD) tool. An overview of J2EE technology is provided, followed by hands-on experience with JNDI, JDBC, JMS, session beans, entity beans, and message-driven beans. The EJB 2.x specification is covered, with emphasis on container-managed persistence (CMP) and container-managed relationships (CMR).

Who Should Attend: This course is for experienced Java programmers and software engineers preparing to write Enterprise JavaBeans for J2EE applications hosted on IBM WebSphere Application Server.

Prerequisites: Students should be comfortable with Java programming and object-oriented concepts. A minimum of six months coding experience is suggested. In addition, students should have prior experience using JDBC and SQL.

Benefits of Attendance: Upon completion of this course, students will be able to:

- Start, stop, and configure IBM WebSphere Application Server
- Use JNDI to access database and EJB resources
- Create JDBC data sources
- Write stateless and stateful session beans
- Use bean-managed and container-managed persistence
- Understand and write XML-based deployment descriptors
- Configure and deploy EJBs using the Rational Application Developer tool
- Assemble J2EE-compliant enterprise applications
- Use the Session Facade pattern
- Implement container-managed relationships
- Create JMS destinations
- Write message-driven beans

Course Outline:

Overview of Java EE

Java Platforms
Characteristics of "Enterprise" Computing
Java EE Technologies
Multi-Tier Architectures
Advantages of Multi-Tier Architectures
Container-Based Approach
Parties Involved in Java EE Deployment
Java EE-Compliant Application Servers
Java EE Application Models
HTTP Services Application Model
N-Tiered Application Model

Introduction to RAD

What is WebSphere@?
What is Rational@ Application Developer?
Starting RAD
Creating a Java Project
Importing Existing Java Code
Creating a New Java Program

Java Naming and Directory Interface

What is JNDI?
Benefits of JNDI
Naming Services
Directory Services
Using JNDI
Context Operations
JNDI Utility Class
JNDI Example
Naming Exceptions
Creating a Server Instance
Starting and Stopping WebSphere
Running the JNDI Example

Using JDBC Data Sources

A Simple JDBC Program
JDBC Driver Types
Using the Derby Database
JDBC Data Sources
Data Source Example
Configuring a JDBC Provider
Configuring a Data Source
Running the JDBC Examples
Executing a Query
Using the Database Explorer

RMI and IOP

Object Serialization
Remote Method Invocation

RMI Architecture
The Remote Interface
CORBA

Enterprise JavaBeans

Enterprise JavaBeans Component Model
Types of Enterprise Beans
EJB Wrapper Interfaces
Deployment Descriptors
Context and Environment Objects
EJB Runtime Environment
The Remote Interface
The Home Interface
The Enterprise Bean Class
The Client Test Program
The `ejb-jar.xml` File
The `ibm-ejb-jar-bnd.xml` File
Creating an Enterprise Application Project
Deploying the Enterprise Application
Testing with the IBM Universal Client

Session Beans

Session Bean Lifetime
Session Bean Interface
Session Bean Lifecycles
Stateless Session Bean Example
Accessing Environment Entries
Stateful Session Bean Example
EJB Exceptions - Examples
Testing the Session Beans
Creating a New Session Bean

BMP Entity Beans

Entity Beans
Entity Bean Interface
Lifecycle of an Entity Bean
Bean-Managed Persistence Example
Deploying Entity Beans
Deployment Settings for BMP Entity Beans

CMR Entity Beans

Container-Managed Persistence
Primary Key Class
Implementing CMP Entity Bean Methods
Container-Managed Persistence Example
Deployment Settings for CMP Entity Beans
Deployment Settings for Custom Finders
EJB Query Language
Mapping Container-Managed Fields
Testing the Product Bean

Session Facade Pattern

J2EE Design Patterns
Session Facade Pattern
Local Interfaces
Example - `ItemOrderer` Bean
Deployment Settings for `ItemOrderer` Bean
Testing the Session Bean
Bottom-Up Mapping
Configuring the Application Client Project

Container-Managed Relationships

Container-Managed Relationships
Container-Managed Relationship (CMR) Example
CMR Example - Local Interfaces
CMR Example - Local Home Interfaces
CMR Example - Entity Bean Classes
Transfer Object Pattern
CMR Example - Session Bean
CMR Example - Deployment Descriptors
Creating New CMP Entity Beans
Creating a Relationship
Generating a Top-Down Mapping
Creating the Tables
Adding an Existing Session Bean
Running the Client Program

Java Message Service

Introduction
JMS and the J2EE Platform
Basic JMS Concepts
The JMS Programming Model
Point-to-Point Example - Sender
Point-to-Point Example - Receiver
Configuring JMS for WebSphere
Running the Point-to-Point Example
Publish/Subscribe Example - Publisher
Publish/Subscribe Example - Subscriber
Running the Publish/Subscribe Example
Reliable Message Delivery

Message-Driven Beans

Message-Driven Beans
Message-Driven Bean Lifecycle
Message-Driven Bean Example
Configuring an Activation Specification
Deploying Message-Driven Beans
Creating a New Message-Driven Bean

Appendix A: Web Resources

Java Technology
WebSphere
Derby Database

Appendix B: Using EJBs in a Web Application

Using Web Components as EJB Clients
Servlet Code for the Survey Application
Session Bean for the Survey Application
Deploying the Survey Application

Appendix C: EJB Transactions

Transactions
Container-Managed Transactions
Transaction Attributes
System vs. Application Exceptions
Rolling Back a Container-Managed Transaction
Configuring a Transactional Data Source

Appendix D: EJB Security

Java EE Security
Specifying Permissions for EJBs
Enabling Security for WebSphere

Appendix E: EJB Timer Service

Overview of the Timer Service
Timer Service API
Creating Timers
Canceling and Saving Timers
Example
Running the Example

Appendix F: Introduction to EJB 3

Limitations of EJB 2
EJB 3 Feature Overview
Comparing EJB 2 and 3
The EJB 3 Business Interface
The Annotated EJB Class
Dependency Injection
Container Callback Methods
Stateless Session Beans
Stateful Session Beans
Entity Beans

Course Description: Intended for entry to mid-level Java™ developers, JBoss Enterprise Application Development (JB295) course will expose students to JBoss® Java EE frameworks, specifications, and interfaces (APIs). Students will learn how to create and maintain Java EE-compliant applications from start to finish using the Eclipse-based JBoss Developer Studio. Comprehensive lecture and extensive use-case, hands-on labs will introduce the student to Seam, Hibernate, and other related technologies that help create a fully functional enterprise Java application. By the end of the course, students will build a complete multi-tier enterprise application, including a web-based front end, a web services layer, EJB-layer, and a persistence layer, as well as code for test cases at all levels.

Who Should Attend: This course is for students with existing Java programming experience wishing to enter Java Enterprise Edition (Java EE) development as well as for experienced JEE developers who are migrating to JBoss for the first time.

Prerequisites: Students must have fluency in HTML and Java programming language (Java SE, Java SE 5). They should also have basic experience with an integrated development environment (IDE) such as Eclipse or NetBeans; and build tools such as Ant or Maven. Basic knowledge of asynchronous JavaScript (AJAX) and Relational Database Management System (RDBMS) is also required.

Benefits of Attendance: Upon completion of this course, students will be able to:

- Create and maintain Java EE-compliant applications from start to finish using the Eclipse-based JBoss Developer Studio.
- Build a complete multi-tier enterprise application, including a web-based front end, a web services layer, EJB-layer, and a persistence layer, as well as code for test cases at all levels.

Course Outline:

Introduction to the JEE application stack, and JBoss EAP server technologies with a focus on web UI development

Technologies covered: JEE API, JBoss EAP, JSF, and Taglibs
Lab: Complete a JSF page, implement navigation, and deploy with Ant

Technologies: data model, annotations, bijection, and Seam conversations
Lab: Implement a robust flight search complete with suggestion functionality based on city names

Unit testing tools and methodologies, as well as integration of tests with JBDS and Ant for continuous testing

Technologies: JUnit/TestNG, DBUnit, Hibernate Tools, JSFUnit, and Ant
Lab: Complete a JSFUnit test, deploy and test code, integrate with Ant builds

A survey of integration testing technology and concepts

Technologies: SeamTest, TestNG, and Ant
Lab: Implement an end-to-end test of some functionality of the application

Beginning discussion about the business layer, including topics of code separation, business logic, and how the JEE stack separates the various layers

Technologies: EJB3, Session Beans, JNDI, and JTA
Labs: Loading a session bean with JNDI, completing code for a business task based on business rules

Examine how caching can enhance the functionality of a web application. Some of the pitfalls of caching are discussed, and how to use JBoss tooling for caching is covered

Technologies: JBoss cache
Lab: Use JBoss cache to pre-load airport information for better initial functionality for the first customers who connect

Details of JEE's asynchronous messaging technologies, compare and contrast various messaging constructs, and how to integrate messaging into an application

Technologies: JMS, JNDI, and message-driven beans (MDB)
Labs: Write an MDB to process confirmation messages asynchronously, look up a queue using JNDI

Advanced UI features, including details on usability, and designer-provided interfaces

Technologies: RichFaces, Ajax4JSF, and Facelets
Lab: Create a more robust interface with built-in UI widgets

Representation of business data in Java, as well as managing transactions to the database layer. Final portion of the three-tier architecture, the persistence layer

Technologies: Hibernate, JPA, entity beans, and JTA
Labs: Using JPA, students will commit changes in customer preferences to the database

Expose business services as web services, for simpler distribution of the application functionality to outside businesses. Available tooling for the consumption of web services via JBoss is also covered

Technologies: web services, JAX-WS, SAAJ, REST, and SOAP
Lab: Expose a business process, and consume the process via web service deployment

Using Seam as a migration tool for an application. This unit begins our discussion of JBoss enhancements to JEE development, to make the process more streamlined

Technologies: Seam, seam-gen
Lab: Recreate the framework for our application using seam-gen, and reuse previously developed JSF code as the front-end

Explore the various security features available in JBoss EAP 5.0

Technologies: JAAS, HTTP authentication, and Seam
Labs: Create a more robust login process, including a new profile page for the customer, and determine user roles based on history

Advanced data model concepts are introduced, supported by Seam tooling, and used to produce robust web application features

Course Description: JBoss Hibernate offers high-performing object/relational persistence and query services. The JBoss Hibernate Technology (JB297) three-day course gives Java developers the knowledge and skills required to leverage the powerful Java Hibernate Application Stack. Through clear interactive lectures and hands-on labs, students are introduced to Hibernate essentials and internals along with its practical applications and best practices strategies. Hibernate helps students produce and maintain well-designed, robust business applications while optimizing performance and reducing software maintenance costs. This course will also demonstrate to the students how Web Platform can use a slimmed down profile of the JBoss Application Server to provide an integrated platform for next-generation, standards-based Java applications.

JBoss Hibernate adapts to your development process, no matter if you start with a design from scratch or work with an existing database. It supports any application architecture. Combined with Hibernate EntityManager and Hibernate Annotations, you can use Hibernate as a certified Java Persistence provider.

Who Should Attend: The primary audience is intended to be Enterprise systems architects, experienced Java developers who work with SQL-based database systems, business component and database developers, and database administrators who need to understand how ORM may affect performance and how to tune the performance of the SQL database management system and persistence layer.

Prerequisites: Students should have experience with Java Platform, Enterprise Edition (Java EE) or Java 2 Platform, Enterprise Edition (J2EE), a high-level understanding of enterprise software systems development, and an understanding of legacy systems integration.

Benefits of Attendance: Upon completion of this course, students will be able to:

- Leverage the powerful Java Hibernate Application Stack.
- Produce and maintain well-designed, robust business applications, while optimizing performance and reducing software maintenance costs.

Course Outline:**State of the union**

Introductions
Explanation of .org vs. .com
Brief overview of products

Understanding the object-relational mismatch**Introduction to Hibernate ORM****Hibernate tooling and development****Persistent classes in Hibernate****O/R mapping details and inheritance****Hibernate Transactions****Understanding Hibernate Query technology****Best practices and improving performance****Additional Hibernate modules**

Course Description: The JBoss Seam Application Development (JB311) course teaches experienced Java developers how to efficiently use Seam to intelligently tie components together and manage increasingly complex IT systems. The course focuses on the core of the JBoss Seam Technology: rapid application development, an industry standard UR, Java Persistence APO (JPA) integration, and end-to-end security, and integrated tooling in JBoss Developer Studio. Using clear interactive lectures and hand-on labs, this course also introduces SEAM integration points for rules engines, business process management, and web services, in addition to covering the new SR-299 CDI standard, a feature of Java EE 6.

Who Should Attend: This course is for Enterprise systems architects, experienced enterprise Java developers, technical managers, and developers who already have JSF and basic Seam knowledge.

Prerequisites: Students must have experience with Java Platform, Enterprise Edition (JavaEE) or Java 2 Platform, Enterprise Edition (J2EE). They must also have a high-level understanding of enterprise and modern web application development. In addition, students should have completed the Advanced JBoss Enterprise Development (JB295) course or have equivalent experience.

Benefits of Attendance: Upon completion of this course, students will be able to:

- Use Conversations to manage User Tasks and Workspaces
- Create Seam Components to support business logic
- Apply Seam technologies, such as Bijection, to Persistence, Security, Navigation, and Validation
- Leverage AJAX to build a better UI using RichFaces
- Integrate web application development with JBoss Developer Studio (Eclipse)

Course Outline:

Conversations to manage User Tasks and Workspaces

Seam Components to support business logic

Seam technologies

Bijection
to Persistence
Security
Navigation
Validation

Leveraging AJAX to build a better UI using RichFaces

Integrating web application development with JBoss Developer Studio (Eclipse)

Course Description: Advanced JBoss Enterprise Development (JB325) dives into the JBoss Enterprise Application Platform (EAP) with an emphasis on advanced Java EE application programming interfaces (APIs). This course challenges experienced Java EE developers by providing a deep dive into JBoss EAP details, features, internals, and Java EE best practices. Leveraging JBoss EAP allows students to build, deploy, and maintain highly performing, scalable applications.

Diving into areas of JBoss technologies that differ from the non-JBoss enterprise middleware stacks, developers will be exposed to aspect-oriented programming, interceptors, JMX, and JBoss Services. In addition, students will be introduced to the new JBoss Messaging (for EAP 4.3).

Using JBoss Developer Studio extensively as a lab integrated development environment (IDE), hands-on labs allow developers to experience and explore JBoss Cache, JGroups, clustering, dynamic proxies, transactions, and performance tuning.

Who Should Attend: This course is for the following groups of students: * Experienced Java developers seeking to enhance their utilization of JBoss, * Java developers who need a deeper understanding of JBoss to implement customized services based on remoting, JMX, or other protocols outside the normal JEE .ear or .war deployments, * ISV development teams who need to know JBoss more intimately to customize the server environment to better fit their applications' deployment needs, and * Application architects seeking to produce leaner, meaner deployment artifacts, resulting in better performance and integrity results.

Prerequisites: Students should have two years of experience with Java Platform, Enterprise Edition (Java EE) or Java 2 Platform, Enterprise Edition (J2EE). They should also be proficient in HTML and have experience with an integrated development environment (IDE), such as Eclipse or NetBeans, and build tools, such as Ant or Maven. Students should also have basic knowledge of open source relational database management system (RDBMS).

Benefits of Attendance: Upon completion of this course, students will be able to:

- Build, deploy, and maintain highly performing, scalable applications.

Course Outline:

Introduction to JBoss

The JBoss technology stack
Installing and starting JBoss
Lab: Install JBoss

JBoss architecture

Technologies: JBoss EAP 4.3
Lab: Classloading in JBoss

JMX

Technologies: JMX
Lab: Create and deploy a custom MBean
Bonus lab: Manage the MBean from a client

Using aspects in JBoss

Technologies: JBoss, AOP
Lab: Create and deploy a custom interceptor

Connecting to JBoss

Technologies: Java Connector architecture
Lab: Set up data sources and tune them

Transactions in JBoss

Technologies: JEE Transactions, JBoss Transactions

JBoss Cache

Technologies: JBoss TreeCache, PojoCache
Lab: Create and deploy a cached application

Clustering applications in JBoss

Technologies: JBoss Clustering, PojoCache
Lab: Deploy and cluster a stateless EJB
Bonus lab: Deploy and cluster a stateful EJB

JGroups

Technologies: JBoss Clustering, JGroups
Lab: Configure JGroups to deploy and cluster a web application

Fine-tuning applications in JBoss

Technologies: JBoss, JConsole
Lab: Use JConsole to monitor garbage collection in JBoss

Container-managed security

Technologies: JAAS, JBossSX

Lab: Secure a web application in JBoss

Bonus Lab: Secure and deploy a stateless EJB in JBoss

JBoss Messaging

Technologies: JMS, JBoss Messaging

Lab: Deploy four JMS queues and monitor performance under load
Bonus lab: Test JBoss Cache state replication in JMS queues

Course Description: JBoss Application Administration (JB336) focuses on installation and deployment of the JBoss Application Server, as well as configuring and monitoring the server for production usage. This class balances the essential concept-based lectures with the real world task and project oriented labs. It reaches beyond the application programming interface (API) and enforces the applied knowledge of the technology. With the heavy emphasis on real-world scenarios, the JBoss Application administrator will be able to better understand, maintain and troubleshoot their environment.

Who Should Attend: This course is targeted at systems administrators, build/deployment managers, and quality assurance engineers who want to optimally administer JBoss Application Server deployments.

Prerequisites: Students should have basic experience with system administration on Windows, UNIX, or Linux operating systems as well as understanding of hardware and networking. No prior knowledge of Java™, scripting, or JBoss Developer Studio is required.

Benefits of Attendance: Upon completion of this course, students will be able to:

- Identify the hardware and software required to run JBoss products and determine which software versions to use
- Define an enterprise application, how to deploy it to JBoss EAP, and what types of supporting components are needed
- Identify what tools are available to monitor JBoss EAP installations, how to configure those tools, and what information they offer
- Understand web protocols such as HTTP, and secure socket layer (SSL) via HTTP/S
- Understand the various forms security takes within an enterprise system and how they are best used
- Slim down their EAP installation to trim away any unnecessary services or applications and understand application-level security and how encryption can be implemented
- Identify the tools available to help understand and identify potential application problems as they arise
- Deploy a clustered application into a tuned and clustered production environment
- Load-balance the embedded web server in JBoss
- Configuring an application for deployment to the load-balanced server and tune request-handling parameters for optimum scalability

Course Outline:

Unit 1 - Installation and basic configuration of JBoss Enterprise Application Platform (EAP)

Technologies covered: Java, JBoss EAP, JMX

Lab: Install JBoss Enterprise Application Platform. Also install JBoss Admin Console as a front-end GUI for management and control.

Unit 2 - Deploying enterprise applications to JBoss EAP

Technologies: JBoss EAP, J2/JEE, deployable packages

Lab: Verify that your application is deployed. Use JBoss Admin Console to deploy packages.

Unit 3 - Monitoring and controlling JBoss EAP

Technologies: JBoss EAP, JMX, JConsole, JBoss Admin Console

Labs: Monitoring the application server with Jconsole. Using JBoss Admin Console to monitor datasources, Enterprise Java Beans, web servers, and other application components.

Unit 4 - Connecting to JBoss EAP. In this unit, students learn what connections are available and can provide access to JBoss components

Technologies: HTTP, SSL, AJP, JNDI, JMS

Labs: Protect incoming ports from denial-of-service (DOS) attacks

Unit 5 - Securing applications with JBoss solutions

Technologies: JAAS, LDAP, HTTP/S, SSL certificates

Labs: Secure the system by insuring that no unnecessary connections are possible. Use an LDAP server (Red Hat Data Server) to authentication of user logins.

Unit 6 - Troubleshooting applications on JBoss EAP

Technologies: Java, JBoss EAP, stack traces, profilers, software patches

Lab: Identify a performance bottleneck in the installed application and decide how to resolve the performance issue.

Unit 7 - Clustering applications with JBoss EAP

Technologies: JBoss EAP, JBoss Operations Network, JGroups, Apache modules

Lab: Verify that the provided applications are fully configured for the production environment (datasources, deployment descriptors, etc.).

Unit 8 - Optimizing applications for JBoss EAP

Technologies: JBoss Cache, Apache load balancing modules

Labs: Cluster a web-based application with JBoss. Start two clustered instances of JBoss, then deploy the application to the cluster, and watch what happens when the serving node crashes.

Course Description: The Advanced JBoss Administration (JB346) course is designed for experienced system administrators responsible for deploying and administering JBoss EAP in large-scale production environments. It deep-dives into clustering, performance tuning, and provisioning EAP instances. In addition to learning how to provision JBoss and applications using JBoss Operations Network bundles, students will also learn how to performance-tune JBoss and its JVM by locating and correcting bottlenecks in their deployments. Finally, students learn how to provision and configure JBoss clusters, including best practices for load balancing, session replication, and rolling out application upgrades.

Who Should Attend: This class is for experienced system administrators responsible for deploying and administering JBoss EAP and JavaEE applications in large-scale production environments.

Prerequisites: Students should have the skills covered in JBoss Application Administration (JB336) training course or demonstrated on the JBCAA Exam (EX336).

Benefits of Attendance: Upon completion of this course, students will be able to:

- Perform clustering, performance tuning, and provisioning EAP instances.
- Provision JBoss and applications using JBoss Operations Network bundles.
- Performance-tune JBoss and its JVM.
- Provision and configure JBoss clusters.

Course Outline:

Introduction to JBoss Operations Network (JBoss ON)

Install and configure JBoss Operations Network

Provisioning

Use JBoss ON Bundles to provision JBoss servers and applications

Configuring JBoss clusters

Configure and deploy JBoss EAP for a high-availability, production-level environment

Managing clustered applications

Configure and deploy JavaEE applications onto a JBoss EAP cluster

Cluster caching

Implement and configure caching of JavaEE applications in a cluster

Performance tuning JBoss

Improve the performance and throughput of a JBoss EAP server

Performance tuning in enterprise environments

Understand and fine-tune the JVM memory settings and garbage collector to maximize the performance of a platform running JBoss EAP

Course Description: JBoss® Enterprise BRMS is an open source business rules management system that enables easy business policy and rules development, access, and change management.

The JBoss Enterprise BRMS Implementation (JB433) course is designed for developers who are implementing a BRMS solution in an enterprise environment. Concentrating on business rules development, developers learn how to author, test, debug, and control business rules in a production environment. In addition, they will generate rules packages and learn how the BRMS runtime environment executes rules. The principles in this course apply equally to customers implementing a standalone BRMS solution.

Who Should Attend: This course is for Business analysts and enterprise service-oriented architecture (SOA) architects who are responsible for creating and adapting business policies. This course is also for those responsible for authoring and testing rules. JEE application developers responsible for integrating business rules into SOA and JEE enterprise applications will also benefit.

Prerequisites: Basic Java EE programming experience is required. Business rule development experience is not required.

Benefits of Attendance: Upon completion of this course, students will be able to:

- Author business rules in JBoss Developer Studio and Guvnor
- Author and test business rules
- Author rule templates and generate rules from them
- Understand BRMS architecture and rule execution at runtime
- Control rule execution and prevent conflicts
- Package, debug, and deploy business rules

Course Outline:

JBoss overview

Authoring basic rules

Using the Guvnor web interface and JBDS to author basic business rules

Authoring basic rules as text files, technical rules, and decision tables

Authoring rules using text files, the Guvnor technical rule editor, and decision tables. In addition, students learn how to work with Domain Specific Language files and rule templates

Testing business rules

Testing business rules using the Guvnor web interface

Complex rule authoring

Authoring complex business rules using advanced conditions and field constraints

BRMS architecture and rule execution

Learn the architecture of the JBoss Enterprise BRMS and how the runtime environment processes rules

Controlling rule execution

Controlling rule execution and how to avoid rule conflicts

Debugging business rules

Debugging business rules using JBDS

Course Description: This course describes how to write database-backed Web Applications using the Ruby on Rails (also pronounced RoR, or Rails) Framework. Students are taken through the various steps of creating a full-fledged Web Application. Topics include static and dynamic pages, sign up forms, signing in and out, sessions, managing users, and various other Rails topics.

Who Should Attend: A typical student in this course has either been tasked with writing a Rails application, or has been tasked with evaluating the Rails framework. Thus, this course should be attended by developers and managers alike, and possibly system administrators.

Prerequisites: Students should have been through a course in the Ruby Programming language. They should also have an understanding of HTML. Students should also have experience with a SQL compliant database such as MySQL.

Benefits of Attendance: Upon completion of this course, students will be able to:

- Install and set up the development environment.
- Understand and create a basic Rails application.
- Create static pages.
- Understand the Ruby language underlying Rails.
- Create a site layout, user data model, and full registration and authentication system.
- Add microblogging and social features.

Course Outline:

From Zero to Deploy

Introduction
Up and Running
Version Control with Git
Deploying

A Demo App

Planning the Application
The Users Resource
A Microposts Resource

Mostly Static Pages

Static Pages
Our First Tests
Slightly Dynamic Pages
Exercises

Rails Flavored Ruby

Motivation
Strings and Methods
Other Data Structures
Ruby Classes
Exercises

Filling in the Layout

Adding Some Structure
Layout Links
User Signup: A first Step
Exercises

Modeling and Viewing Users

User Model
User Validations
Viewing Users
Exercises
Insecure Passwords
Secure Passwords
Better User Views
Exercises

Sign Up

Signup Form
Signup Failure
Signup Success
RSpec Integration Tests
Exercises

Sign In, Sign Out

Sessions
Signin Failure
Signin Success
Signing Out
Exercises

Updating, Showing, and Deleting Users

Updating Users

Protecting Pages
Showing Users
Destroying Users
Exercises

User Microposts

A Micropost Model
Showing Microposts
Manipulating Microposts
Exercises

Following Users

A Relationship Model
A Web Interface for Following and Followers
The Status Feed
Exercises