

Building Java Enterprise Applications with Design Patterns: Hands-On - 4 Days

Course 318 Overview

- You Will Learn How To**
- Architect Java EE applications using industry-recognized best practices
 - Create flexible and powerful designs for core business logic
 - Design a data layer that manages transactions and optimizes queries
 - Centralize control logic in the Web presentation tier using Java EE patterns
 - Compare the designs of popular Java EE frameworks and choose the right one for your projects
 - Build software that can evolve in response to changing requirements

Course Benefits The wide variety of Java enterprise technologies presents many challenges to designing an effective Java system. Java EE design patterns help by providing best practices, design ideas and proven techniques. In this course, you gain experience building scalable and maintainable Java EE applications. You learn to apply Java EE patterns to solve commonly recurring design problems.

Who Should Attend Anyone currently designing or developing Java EE applications. As the emphasis is on software design, familiarity with Java code at the level of Course 471, "Java Programming Comprehensive Introduction," is required. Experience with Java EE is beneficial.

Hands-On Training Throughout this course, you gain experience designing flexible, robust Java EE applications. Exercises include:

- Writing a simple distributed chat application
- Implementing a complex Web-based Java EE application
- Designing and implementing a flexible domain model
- Refactoring an integration tier using design patterns
- Employing the Object/Relational mapping capabilities of Hibernate
- Designing detailed Web application workflows
- Utilizing the Struts Web Framework

Building Java Enterprise Applications with Design Patterns: Hands-On - 4 Days

Course 318 Outline

Java EE and Design Patterns

Enterprise system design

- Comparing OO and Java EE patterns
- The benefits of design patterns in Java EE

Distributed systems development

- Exploiting remote method invocation
- Design patterns in distributed systems

Business Tier Patterns

Eliminating inter-tier dependencies

- Illuminating problems associated with poorly designed tiered architectures
- Realizing an application's domain model
- Business Object
- Application Service

Implementing the business tier

- Patterns for locating objects
- Singleton
- Factory
- Inversion of Control

Simplifying object interaction

- Interfacing with adjacent application tiers
- Selecting scalable middle-tier technologies
- Reducing the impact of known performance bottlenecks
- Business Delegate
- Service Locator
- Session Facade

Building the Integration Tier

Abstracting the data layer

- Implementing effective Data Access Objects (DAO)
- Simplifying JDBC code with iBatis
- Highlighting difficulties associated with Object/Relational Mapping
- Representing highly relational data using Hibernate and JPA
- Refactoring the integration tier using an Abstract DAO Factory

Optimizing database queries

- Fast Track Access
- Value List Handler

Managing transactions effectively

- Handling long-running transactions

- Comparing optimistic and pessimistic transaction strategies
- Effecting complex concurrency management with a Transaction Context

Structuring the Web Presentation Tier

Separating control and presentation logic

- The role of JSPs and servlets
- Constructing Model View Control (MVC) architectures
- Front Controller
- Dispatcher View
- Service to Worker

Applying Web framework support

- Investigating the Struts MVC architecture
- Planning and implementing complex workflows
- Handling duplicate form submission with the Synchronizer Token pattern
- Investigate component-based frameworks such as JSF

Localizing disparate logic

- Improving maintainability of algorithms
- Intercepting Filter
- View Helper
- Composite View
- Reusing page layout with Tiles
- Writing modular JSPs

Lightweight Architectures

Reducing coupling in applications

- Inversion of Control (IoC) design pattern
- Configuring the Spring IoC container

Promoting code reuse

- Aspect-Oriented Programming
- Executing component reuse with Spring
- Sending e-mail using Spring
- Utilizing Spring data access templates

Performance and Scalability

Designing for performance

- Distributed components and performance
- Measuring runtime performance
- Optimizing Java EE applications
- Caching
- Connection Pooling

Planning for scalability

- Evaluating design trade-offs in distributed architectures
- Clustering applications across servers
- Managing session state effectively