



Architecting and Designing J2EE Applications (TT6600)

Five days

Course Description: Participants will gain an understanding of the strategies needed to create application blueprints that work well when implementing J2EE technologies. These strategies include effective decision making using systemic qualities (such as scalability and flexibility), J2EE technology blueprints and design patterns, and iterative and incremental development.

Audience: This is an intermediate to advanced-level J2EE training course, geared for designers and architects that need to relate real problems to J2EE-based solutions.

Prerequisites: Attendees should have practical experience developing basic J2EE applications and have basic development skills and a working knowledge in the following topics, or attend these courses as a pre-requisite: Understanding Internet Architectures Core Java Programming Fundamentals Developing JEE Component Compliant Applications or JEE Web Application Fundamental

Outline

THE BUSINESS ARCHITECTURE: SERVICE AND COMPONENT BASED DESIGN/DEVELOPMENT (SCBD)

Introduction to Service Orientation and CBD

- Understand the move in IT which leads us to Components and the direct business value it can bring.
- Understand the possible effect of Components on IT budgets.
- Define a Component
- See how a Component is rendered in UML2.0 and its relationship with a Port.
- Define Component Based Development and Design.
- Understand the relationship between Enterprise Architecture and (the need for) Components.
- Understand the relation between SOA and CBD
- Understand how the functional requirements find their way through the Business Architecture.
- Describe the requirements of the Software Development Process that leads to components.

- Understand the potential problem of a Waterfall approach.
- Define iterative and incremental development.

Describing a Component Based Software Development Process

- Examine the gap in Software Development between the model and the domain experts (the source).
- Examine the gap in Software Development between the model and the developers (the goal).
- Recognize the area of tension between these two gaps.
- Recap UML (based on version UML2 including relation between UML, MOF and XMI)
- Understand that (traditional) OO Development does not/might not leverage all the possibilities of SOA.
- Understand the concept of "Design by Contract"
- Describe the requirements of the Software Development Process that leads to services and components.

Architecting and Designing J2EE Applications (TT6600)

- Understand the potential problem of a Waterfall approach.
- Define iterative and incremental development.

APPLYING SCBD'S COMPONENT MODELING

The Requirements Capture Phase

- Use-Cases are covered to get a complete picture of a Software Development process that delivers Components for your technical environment.
- Understand how to gather system requirements.
- Introduce Use-Cases for the other lessons on component and solution delivery.
- Describe a way of documenting non-functional requirements

The Business Architecture / Component Design phase

- Understand the steps involved when specifying the component architecture.
- Illustrate a top-down approach for identifying components.
- Describe an informal process for identifying components.
- Describe a formal process for identifying components.
- Explain how you can guarantee that you have all services in place.
- Understand how to model document-oriented services

ENTERPRISE ARCHITECTURES AND A JUSTIFICATION FOR USING J2EE

The Technical Enterprise Architecture

- Define the Enterprise (Technical) Architecture and its relationship to Project Architectures.
- List the four technical characteristics/requirements of a Technical Enterprise Architecture.
- Understand how the non-functional requirements find their way through the Technical Architecture.
- For each of the four requirements list possible solutions.
- For each of these solutions, list possible technical approaches.
- Understand the challenges involved with these approaches.
- Understand the nature and reason for the technical challenges that were created.
- Explain what Middleware can address and what other challenges it brings to the table.
- Explain the need for a Framework.
- Introduce two Frameworks: .Net and J2EE

J2EE as a Candidate Framework for your Architecture

- Get a brief introduction of the J2EE Framework within this context.
- Recap the J2EE Framework architecture.
- Recap the Technical Architecture challenges and recap the technical solutions.
- For each of the technical solutions previously described, understand how J2EE can address these (or help you).

Architecting and Designing J2EE Applications (TT6600)

- Main goal of this part is to determine a foundation for choosing J2EE as the technical framework for your project or EA.
- Overview of the complete architecture briefly touching all tiers and their associated J2EE and J2SE technologies.
- J2EE Integration Patterns: Data Access Objects.
- J2EE Integration Patterns: Service Activator.
- Understand how motivations of certain J2EE Patterns have been addressed by a top-down approach of CBD.

J2EE Vendors and Market

- List some of the important players in the J2EE Application Server region
- Through the list of players, describe the different types of Servers
- Discuss criteria for choosing an Application Server

TECHNICAL OVERVIEW OF J2EE

Overview of the Complete Architecture

- Get a complete technical overview of the J2EE Platform
- Recap how each tier is addressed in J2EE
- Recap the architectural role of each of the technologies
- Recap the concept of the EAR and the deployment descriptors
- Understand / recap the relationship between J2EE and Web Services
- List the supporting technologies (JMX, JACC, JTA, etc.)
- List the different versions of J2EE and the related versions and APIs

J2EE Design Patterns Summary

- J2EE Business Tier Pattern: Business Delegate (Proxy)
- J2EE Business Tier Pattern: Value Objects (Holder).
- J2EE Business Tier Pattern: Service Locator.

Presentation Tier

- List the architectural responsibilities of a Servlet and a JSP
- Understand the role of the Web Container
- Recap Servlet implementation
- Recap JSP implementation
- Understand the architectural role of Custom Tags
- Recap Custom Tag development and use
- Understand the architectural position of JSF
- Recap an Web Application and the web.xml

Application Tier

- Understand the three different ways of implementing the application tier.
- Understand Enterprise JavaBeans
- Understand the way an EJB container works
- List the different types of EJBs
- Recap EJB development
- Understand different ways of exposing services as a Web Service
- List the contents of the deployment descriptor
- Understand the life cycle of a Session and an Entity Bean
- Understand Transactions within the context of a distributed environment



Architecting and Designing J2EE Applications (TT6600)

- Understand the importance of the transaction isolation level
- List criteria for choosing the transactional configuration for your application(s)
- Understand the relationship between EJB, IIOP, SOAP and GIOP

EIS Tier

- Understand the different technologies to connect with EIS
- Understand the difference between access and integration
- List requirements for integration
- Recap JDBC, JMS and the Connector Architecture

Security

- Understand the security tasks and how they are addressed in J2EE
- Understand how to establish proof of identity in J2EE (web and desktop clients)
- Understand how to restrict access to parts of the web application
- Understand how to restrict access to services
- Understand how to establish data integrity and privacy
- Understand that SSL might not be sufficient in B2B web services
- Explain the configuration of security and the power of indirection
- Discuss different security configurations (including propagation and delegation)

MAPPING TO THE TECHNICAL ARCHITECTURE

Mapping Components to a (distributed) J2EE Environment

- Understand how the Business Components map to J2SE Components
- Understand how the Business Components map to J2EE Components.
- Understand how this process maps to MDA
- Make a well-founded decision when to use EJBs (or not).
- Emphasize once more the importance of non-functional requirements.
- Understand the impact of a Distributed Architecture on the Business Architecture (i.e., signatures of services)
- Appreciate the Value Object Pattern (including JavaBean, Structure and XML strategies).
- Discuss how Technical Components (e.g., Data Access) might be separate components or sub-components.
- Understand the difference of data transfer between local and a distributed Component Model
- Choose a deployment topology based on QoS requirements

The Internal Design Phase (Component Delivery)

- Understand the steps involved in the internal design of components
- Discuss patterns and best practices can be used inside the component design.
- Understand the role of the EJB local interface in component design.

Architecting and Designing J2EE Applications (TT6600)

- Understand how parts of XP can be used in this phase

Overall (Web-Based) J2EE Architectures

- Discuss different J2EE Web Application Architectures.
- Understand the benefits and challenges for each of these architectures.
- (Re)establish the architectural role of JSPs, Servlets, Custom Tags, EJBs, etc.
- Possible role in the Presentation tier of the XML/XSL combination in J2EE Applications

PATH TO USEFUL WEB SERVICES

Services via the Web

- Service Defined
- SOA Defined
- Organizational Framework
- Technical Framework
- Orchestration
- Services vs. SOA
- SOA in the past
- What is new in SOA
- Business impact/ROI
- Myths/Reality
- Adoption issues
- Characteristics of a Good Service
- Services and their Formal Contracts
- Services Should be Stateless
- Service Design Guidelines
- SOA Anti-Patterns

Web Services Overview

- Crossing Boundaries
- What are Web Services?
- Six Key Components
- Web Services Characteristics
- Web Services Architecturally
- Technology Comparison

- Architectural Perspective
- Web Services Enable Decoupling
- Many Web Services Challenges
- Secure Services
- Spec and Standard Evolution
- Web Services Interoperability Organization
- WS-I Has Many Deliverables
- Basic Profile 1.0 Consists of:
- Has > 100 Requirements and Suggestions
- .NET Platform & .NET Web Services
- Java and Web Services

Web Services, Java, and J2EE

- XML and Java APIs at a Glance
- XML Signature
- XML Digital Signatures
- XML Encryption
- JAXP
- JAXB
- JAX-WS
- SAAJ
- JAX-WSA and XWSS
- Web Services APIs
- Web Services for J2EE (JSR109)
- J2EE and Web Services
- Web Services Metadata
- Web Services Stacks at a Glance
- WSIT
- Apache Axis2
- JBossWS
- JWSDP
- WebSphere WS
- Spring-WS
- Key Features

Web Services Quickstart

- What is WSIT?
- WSIT Tool Support
- How is WSIT Used?

Architecting and Designing J2EE Applications (TT6600)

- Web Service Development with WSIT
- Debugging Web Services
- TCP/IP Monitors Provide View of Wire

FOUNDATION - XML AND JAVA

XML, Namespaces, & Schema

- What is XML?
- XML Can Provide Application-Specific Information
- Content: XML Document Syntax Rules
- Structure: A Document Type Definition
- XML Transformation to HTML
- XML Separates Structure, Content and Format
- TriveraTunes Purchase Order
- Content as Markup
- Tell Parser That Text is Data
- Use Predefined Entities
- Well-Formed and Valid XML Documents
- Why Are These Definitions Important?
- XML Namespaces
- Name Collision – Example
- Inter-Organization Name Collisions
- W3C's Solution: Namespaces
- Uniform Resource Indicator
- Declaring a Namespace
- Namespace Scope
- Parsers Use URI, Not the Alias
- Default Namespace
- Attributes and Namespaces
- Example of Namespaces
- Namespaces Best Practices
- Benefits From Valid XML
- W3C XML Schemas
- Impacts of Schemas

- General Form of an XML Schema
- Elements, Attributes, and Types
- Simple Schema and XML Document
- Element Definitions
- Corresponding XML Schema
- Simple Types - Primitive Datatypes
- Restricting Simple Types: Facets
- Complex Types Bring More to Validation
- Repetition Control
- Restricting Simple Types
- Complex Types Can be Derived
- Derivation by Extension
- Extension of Phone Number
- Associating Schemas with XML Instances
- Using XML Schema with Namespaces
- Namespaces Provide Thread of Connection
- Schema Defines a Target Namespace
- XML Doc Uses Schema-Defined Namespace
- schemaLocation Links Namespace to Location
- Relating Schemas to XML

XML in Java - JAXP and JAXB

- XML Parsers Are Complex and Powerful
- Parsers Are Integral to XML Processing
- Parsers and API's
- Parser Generates DOM, Then Hands to App
- Parsing With a DTD or Schema
- Many Options to Consider
- XML and Java
- Security Concerns Relative to Parsing
- Bridging Application Data and XML
- JAXP: Java API for XML Processing

Architecting and Designing J2EE Applications (TT6600)

- JAXP and Transformations
- Challenges to Mapping XML
- Generating XML is Nondeterministic
- JAXB: Binding XML to Java
- JAXB 2.0 Incorporated Changes
- JAXB 2.0 and Java Versions
- Defining the Rules in JAXB
- Turning Rules into Java Classes
- Using the Generated Classes
- Creating Content
- Some JAXB Type Bindings
- XML Schema for List of Items
- Corresponding Class Interface
- A Word about Validation...
- Tutorial: Configuring the Database in MyEclipse

HANDS-ON LABS & TUTORIALS

- Define The System Requirements (Creating the PIM)
- Identifying the Business Components (Elaborate the PIM)
- Identify Component Services
- Writing a Servlet and a JSP applying MVC (Optional)
- Writing a Session Bean (Optional)
- Define High-Level Technical Architecture for the Business Tier
- Mapping the Component Model to J2EE (Creating the PSM)
- Design Components (“Design by Contract”)
- Web Services in Action
- Implementing a Web Service
- Debugging a Web Service
- Namespaces and Schemas
- Working with JAXB
- Tutorial: Configuring JBoss Using MyEclipse
- Tutorial: Creating J2EE Projects in MyEclipse
- Tutorial: Deploying J2EE Projects into JBoss