Introduction to J2EE

Objectives

At the end of this session, you will be able to,

- Define J2EE
- Describe the J2EE tiers
- Describe Enterprise Application
- Describe J2EE Components
- Describe J2EE Roles

Agenda

Following topics are covered during the session,

- J2EE Overview
- J2EE Definition
- Need for J2EE
- Enterprise Application
- J2EE architecture
- J2EE Specification
- J2EE Components
- J2EE Roles

Introduction

J2EE Origin

- Sun Microsystems, together with partners such as IBM, designed J2EE to define a multi-tier architecture for developing enterprise information systems (EIS) to answer the needs from the industry.

- Goals:
  - reduce the cost and complexity of development
  - allow J2EE applications to be rapidly deployed and easily enhanced

What is J2EE?

- Java 2 Platform, Enterprise Edition (J2EE) is an open, standard-based, development and deployment platform for building n-tier, web-based, server-centric, and component-based enterprise applications.

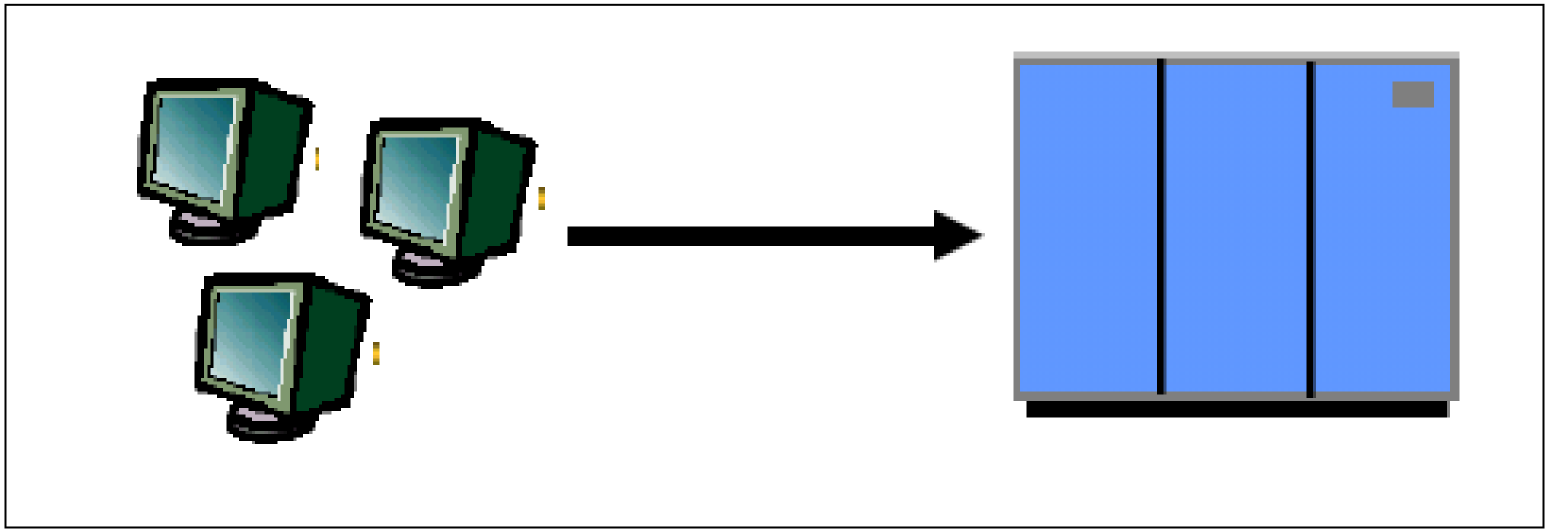
- The J2EE platform consists of a set of services, APIs, and protocols that provide the functionality for developing multitier, Web-based applications.

J2EE Tiers

Evolution of Enterprise Application Framework

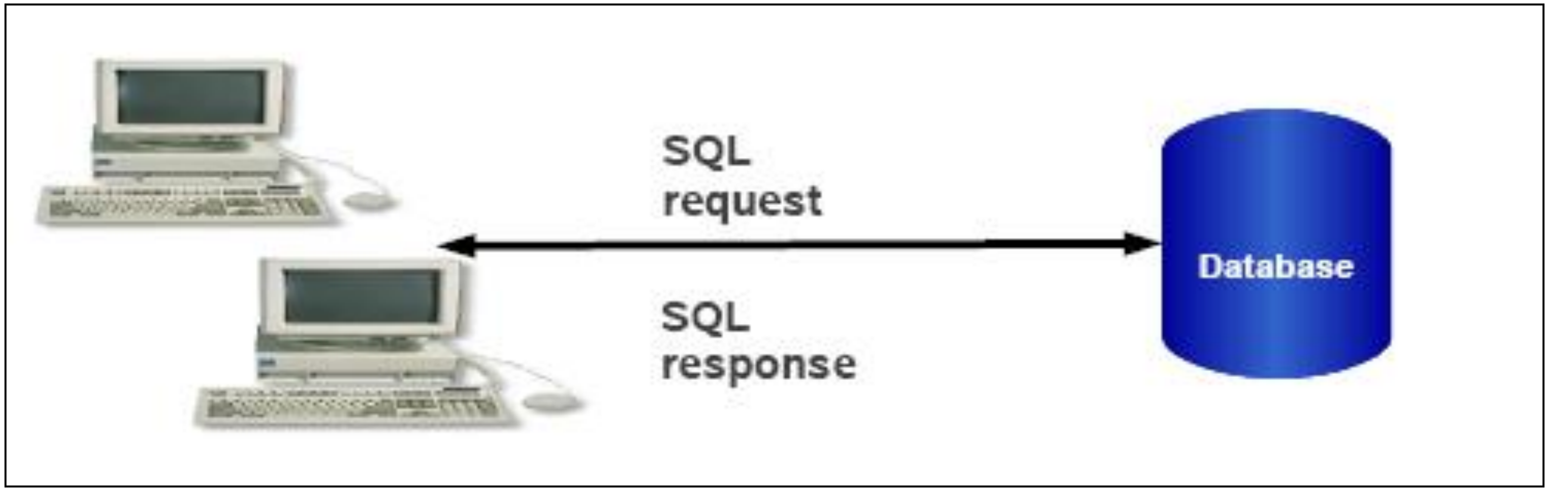
- Single tier
- Two tier
- Three tier (HTML browser and Web server)
- Proprietary application server
- Standard application server

Single-Tier (Mainframe Based)



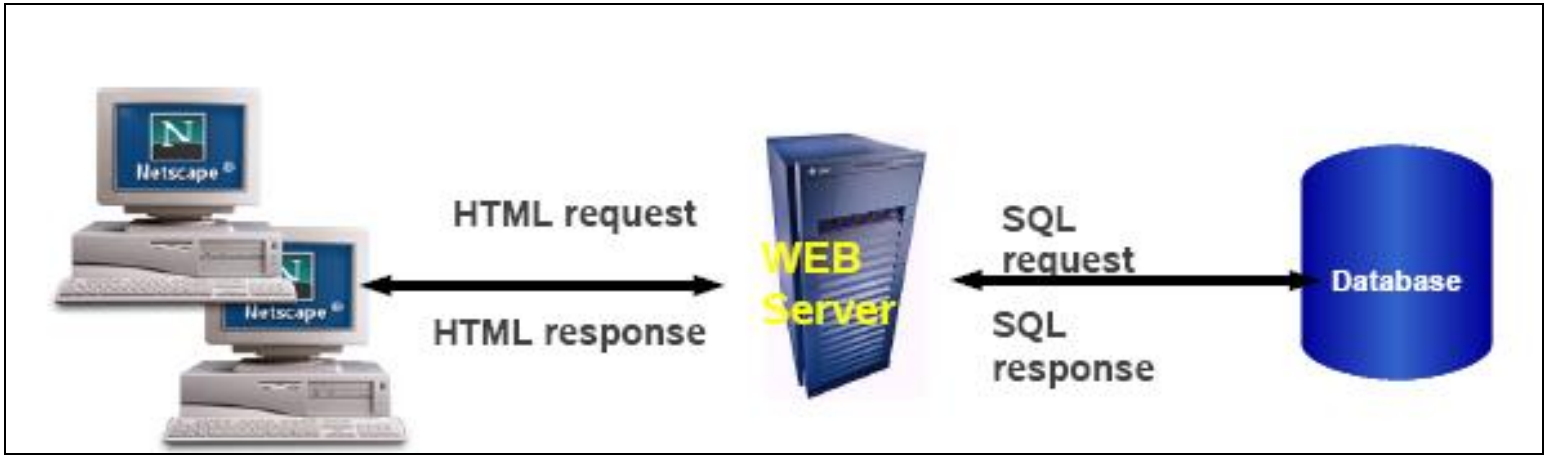
- Dumb terminals are directly connected to mainframe
- Centralized model (as opposed distributed model)
- Presentation, business logic, and data access are intertwined in one monolithic mainframe application

Two-Tier



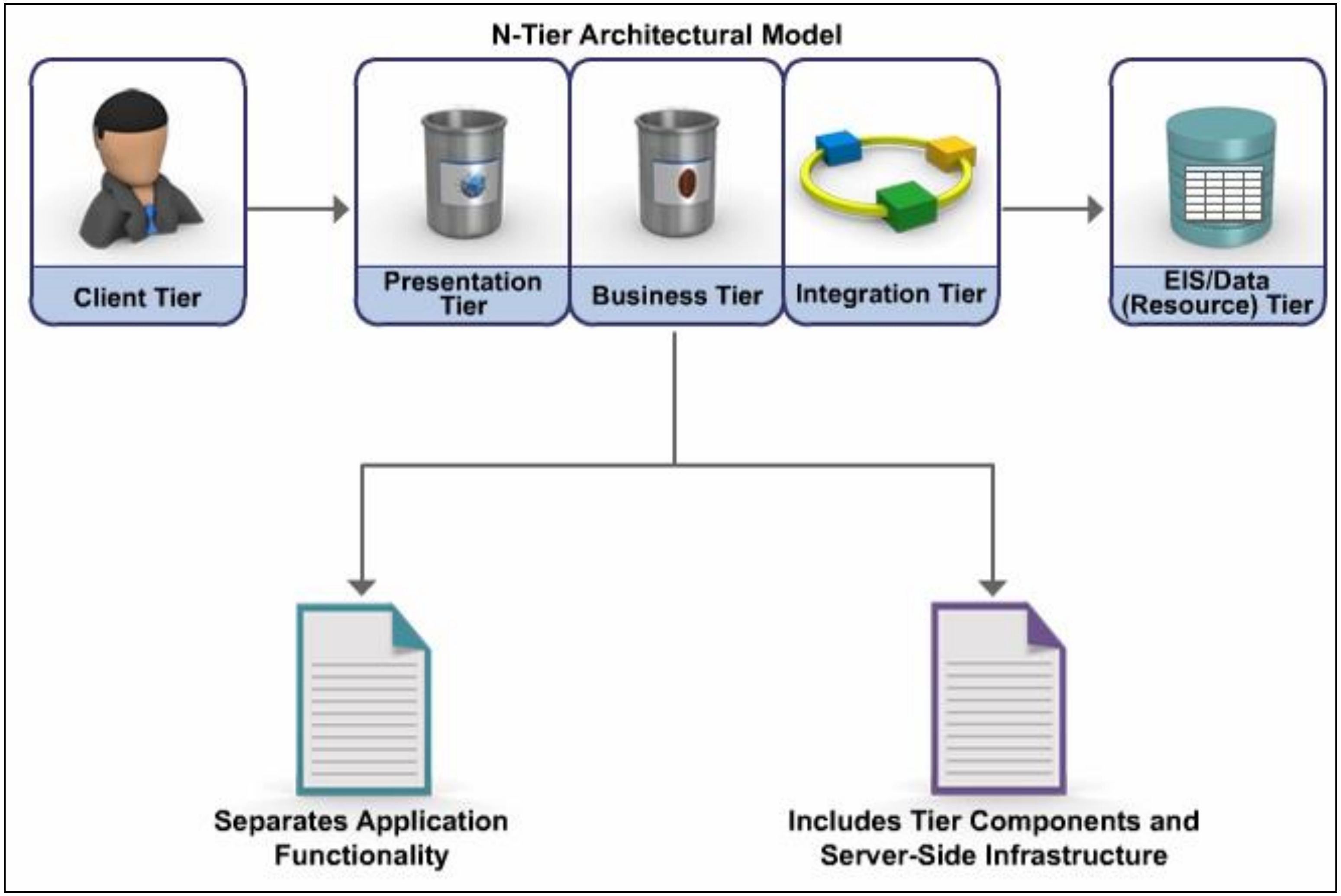
- Fat clients talking to back end database
- SQL queries sent, raw data returned
- Presentation, Business logic and Data Model processing logic in client application

Three-Tier

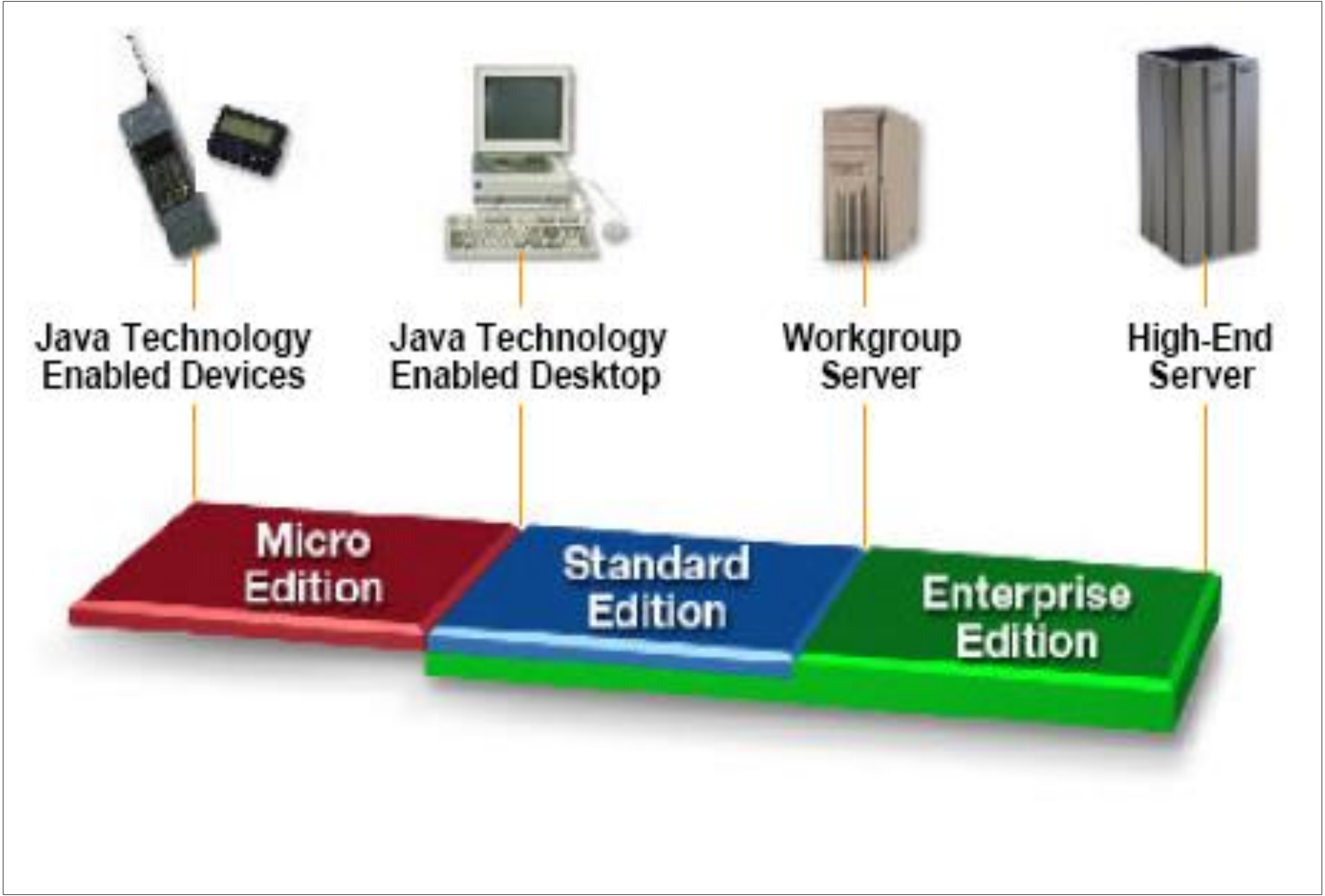


- Thinner client: business & data model separated from presentation
  - Business logic and data access logic reside in middle tier server while client handles presentation
- Middle tier server is now required to handle system services
  - Concurrency control, threading, transaction, security, persistence, multiplexing, performance, etc.

N-Tier Architecture


N-Tier Architectural Model
Client Tier
Presentation Tier
Business Tier
Integration Tier
EIS/Data (Resource) Tier
Separates Application Functionality
Includes Tier Components and Server-Side Infrastructure

J2EE and Java Technology Platform



Java Micro Edition

- Java Platform, Micro Edition (Java ME) is a collection of technologies and specifications to create a platform that fits the requirements for mobile devices such as
  - consumer products,
  - embedded devices,
  - and advanced mobile devices.
- It is a collection of technologies and specifications that can be combined to create a complete Java runtime environment specifically to fit the requirements of a particular device

Java Standard Edition

- **Java Platform, Standard Edition** or **Java SE** is a widely used platform for programming in the Java language.
- It is the Java Platform used to deploy portable applications for general use.
- In practical terms, Java SE consists of a virtual machine, which must be used to run Java programs, together with a set of libraries (or "packages") needed to allow the use of file systems networks, graphical interfaces, and so on, from within those programs.
- It is use for stand alone desktop application

Why J2EE?

- The downside of the three tier architecture is that the middle-tier components where the business logic is written had to provide other services also like concurrency control, threading, transaction, security, persistence, performance, and so on.
- How to solve this problem?

  The solution is to come up with an architecture or framework where the common container which can be shared among all applications provide system services mentioned above.
- What is the solution?

  Use "component and container" model in which container provides system services in a well-defined and as industry standard
- J2EE provides the solution

  The standard specification defines the contract of the component and container model in a well-defined and in an industry standard.

Enterprise Application

- A typical enterprise Application is made up of
  - Presentation logic
  - Business logic
  - Data Access Logic
  - System Services

Enterprise Application

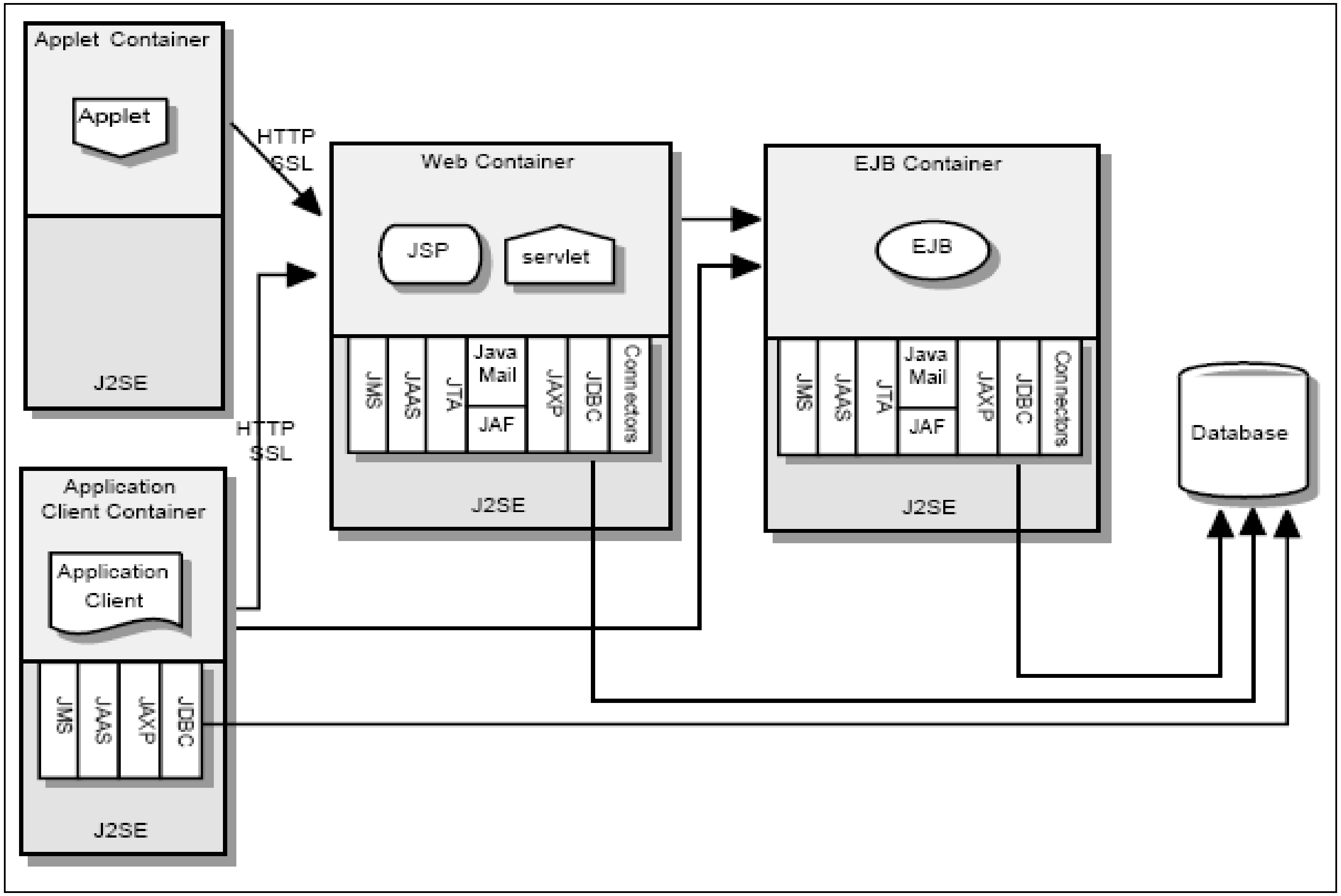
- Features of Enterprise applications
  - Scalability
  - Availability(24 * 7 without any downtime)
  - Maintainability
  - Reliable
  - Speed/Performance
  - Multi-User
  - Fault-Tolerant
  - Secure
  - Robust
  - Distributed
  - Transactional
  - Vendor independent
  - flexible

J2EE Platform and Architectures

J2EE Platform Architecture

- Component
  - A component is an application level software unit
  - The J2EE platform supports the following types of components:
    - Applets
    - Application clients
    - Web components
    - Enterprise Java Beans (EJBs)
- Container
  - All J2EE components depend on the runtime support of a system-level entity called a container
  - Containers provide components with services such as:
    - Connection and Object pooling
    - Security
    - Transaction
    - Deployment

J2EE Platform Architecture



J2EE Specifications

J2EE 5.0 Specifications

- Web Services Technologies
  - Java API for XML-Based Web Services(JAX-WS) 2.0
  - Java API for XML-Based RPC (JAX-RPC)1.1
- Web Application Technologies
  - Java Server Faces 1.2
  - Java Server Pages 2.1
  - JSTL
  - Java Servlet 2.5
- Enterprise Application Technologies
  - Enterprise Java Beans 3.0
  - Java Mail
  - Java Message Service API
  - Java Persistence API
  - Java Transaction API(JTA)
  - Java Beans Activation Framework(JAF) 1.1

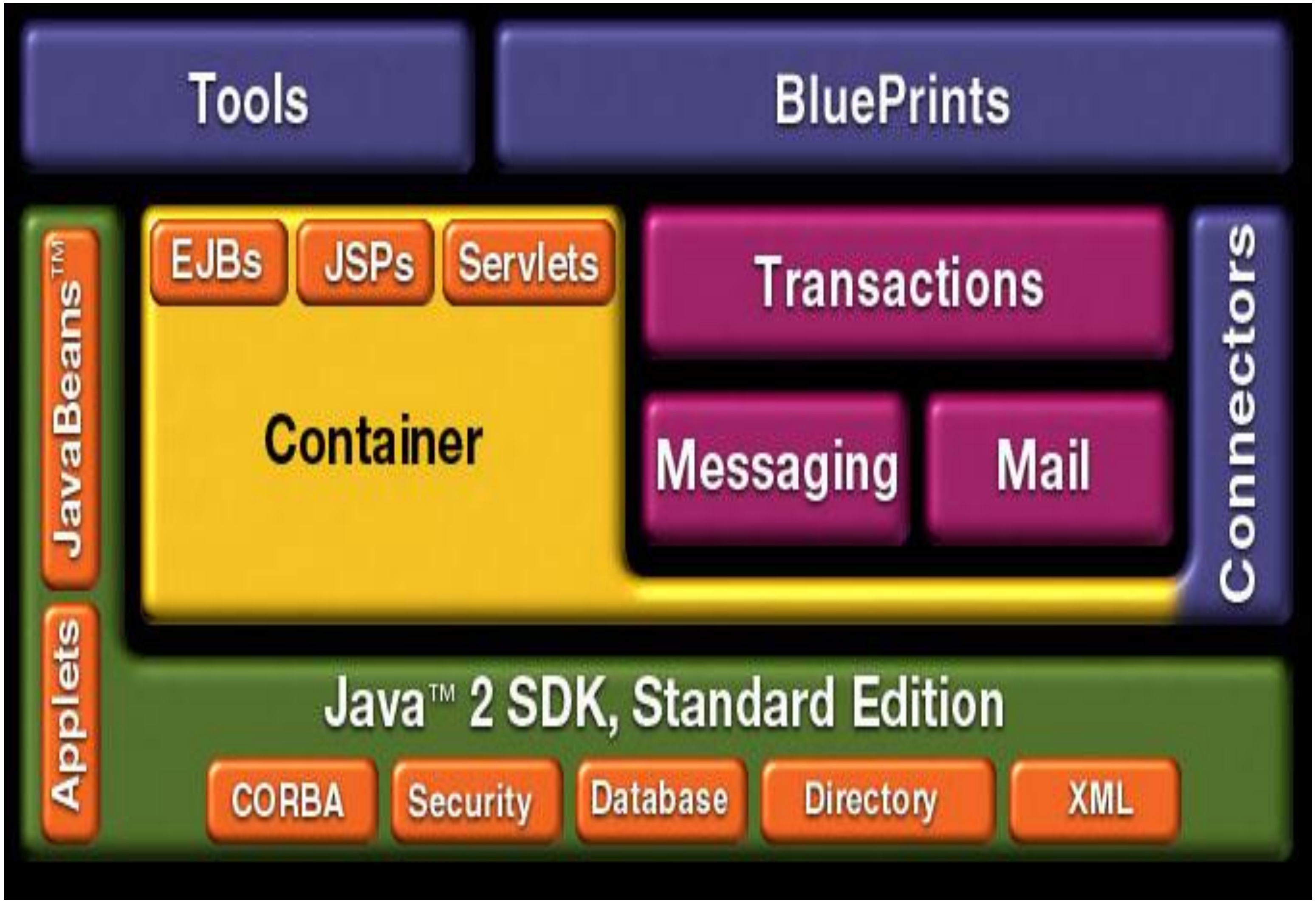
Future versions …. Java EE 6

- Web Services Technologies
  - Java API for XML-Based Web Services(JAX-WS) 2.2
  - Java Architecture for XML Binding(JAXB) 2.2
  - Java API for XML Messaging 1.3
  - Java API for XML Registries(JAXR) 1.0
- Web Application Technologies
  - Java Servlet 3.0
  - Java Server Faces 2.0
  - Java Server Pages 2.2/Expression Language 2.2
  - JSTL 1.2
- Enterprise Application Technologies
  - Dependency Injection for Java 1.0
  - Bean Validation 1.0
  - EJB 3.1
  - Java Persistence 2.0
  - JMS API 1.1

Future versions …. Java EE 6

- Java Mail 1.4
- Management and Security Technologies
  - Java Authorization Contract for Containers 1.3
  - Java EE Application Deployment 1.2
  - J2EE Management 1.1
- Java EE –related Specs in Java SE
  - JAXP 1.3
  - JDBC 4.0
  - Java Management Extension(JMX) 2.0
  - Java Beans Activation Framework (JAF) 1.1

J2EE Stack


Tools
BluePrints
JavaBeans™
Applets
EJBs
JSPs
Servlets
Container
Transactions
Messaging
Mail
Connectors
Java™ 2 SDK, Standard Edition
CORBA
Security
Database
Directory
XML

J2EE Suite

- Core Technology: Container infrastructure, language and environment support

- Web Technology
  - Java Servlets
  - JavaServer Pages
  - JavaServer Pages Standard Tag Library

- Enterprise Java Bean (EJB) technology
  - Session beans
  - Entity beans
    - Enterprise JavaBeans Query Language
  - Message-driven beans

J2EE Suite (contd.)

- XML Technology
  - The Java API for XML Processing (JAXP)
  - The Java API for XML-based RPC (JAX-RPC)
  - SOAP with Attachments API for Java (SAAJ)
  - The Java API for XML Registries (JAXR)

- Platform services
  - Security
  - Transactions
  - Resources
  - Connectors
  - Java Message Service

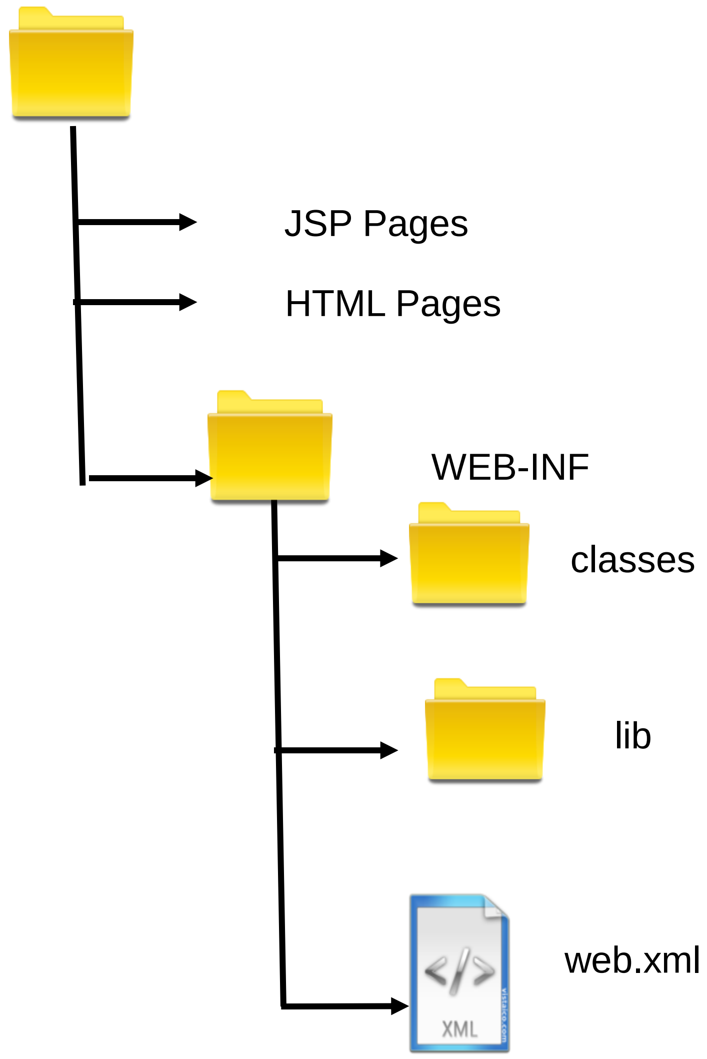
J2EE Communication Technologies

- Internet protocols - TCP/IP HTTP 1.0 SSL 3.0
- Remote Method Invocation protocols
- Object Management Group protocols - JavaIDL RMI-IIOP
- Messaging technologies - JMS, JavaMail
- Data formats - HTML 3.2, Image Files, JAR, class files,XML

J2EE Components

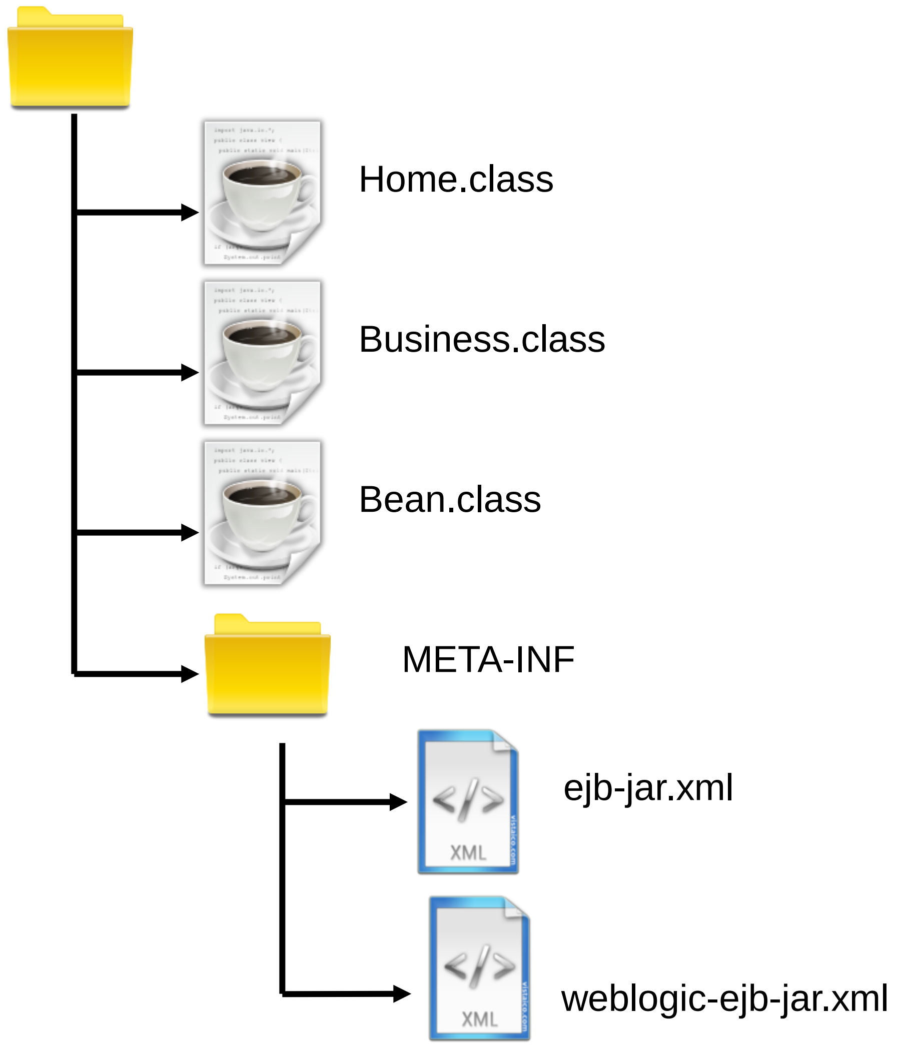
Web Components

- Java content
  - Servlet class
  - JSP Pages
- Deployment Descriptor
  - Web.xml



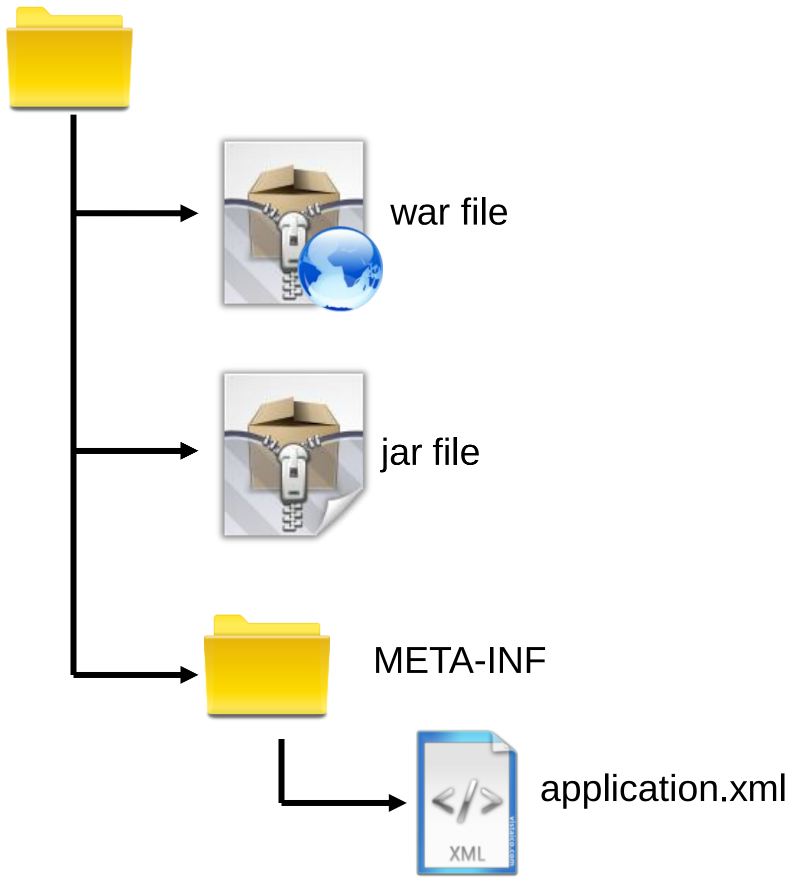
EJB Components

- Java content
  - Home Interface
  - Business Interface
  - Bean Class
- Deployment Descriptor
  - ejb-jar.xml
  - weblogic-ejb-jar.xml

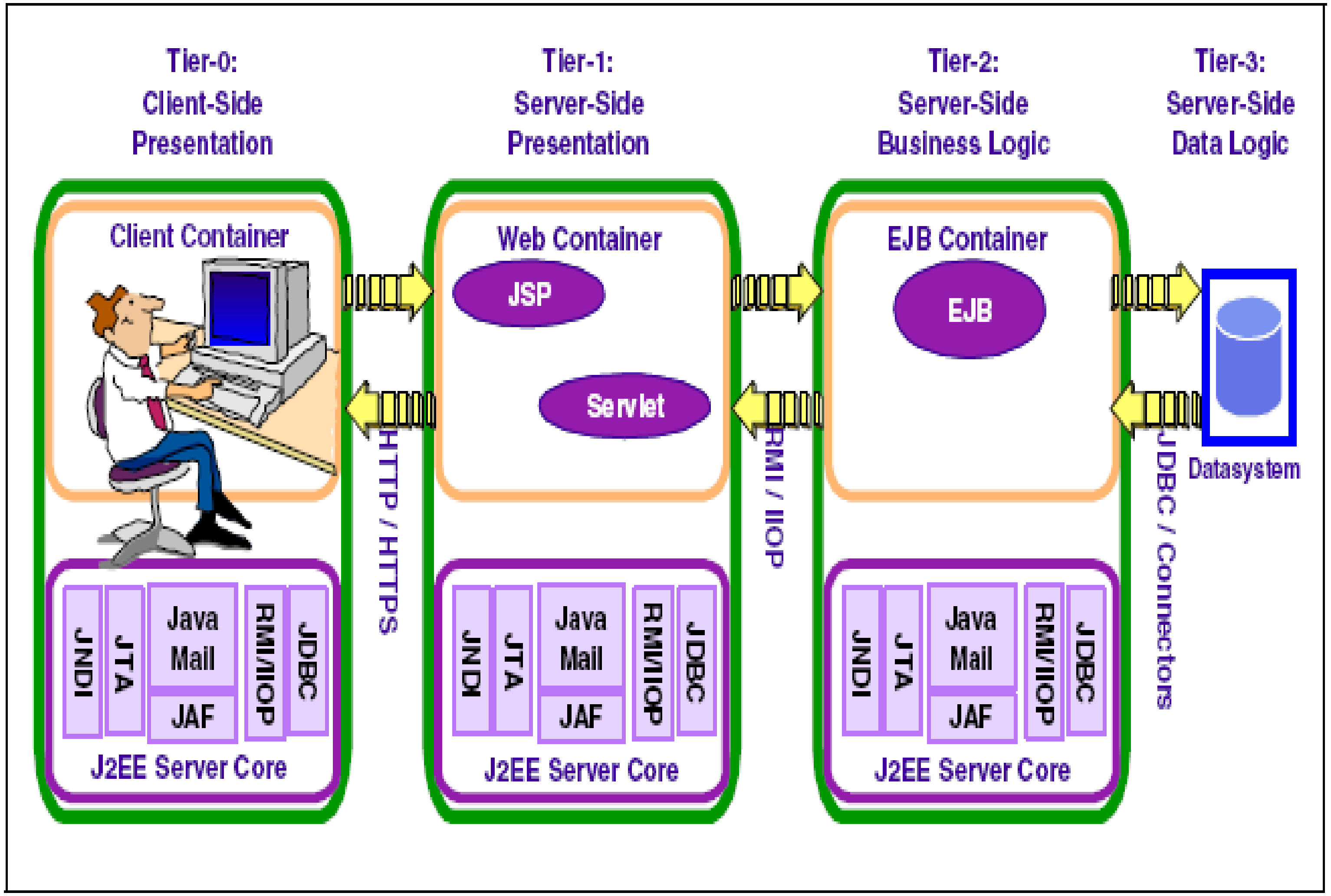


Enterprise Application

- J2EE Components
  - Web components
  - Ejb components
  - Utility jars
- Deployment Descriptor
  - application.xml

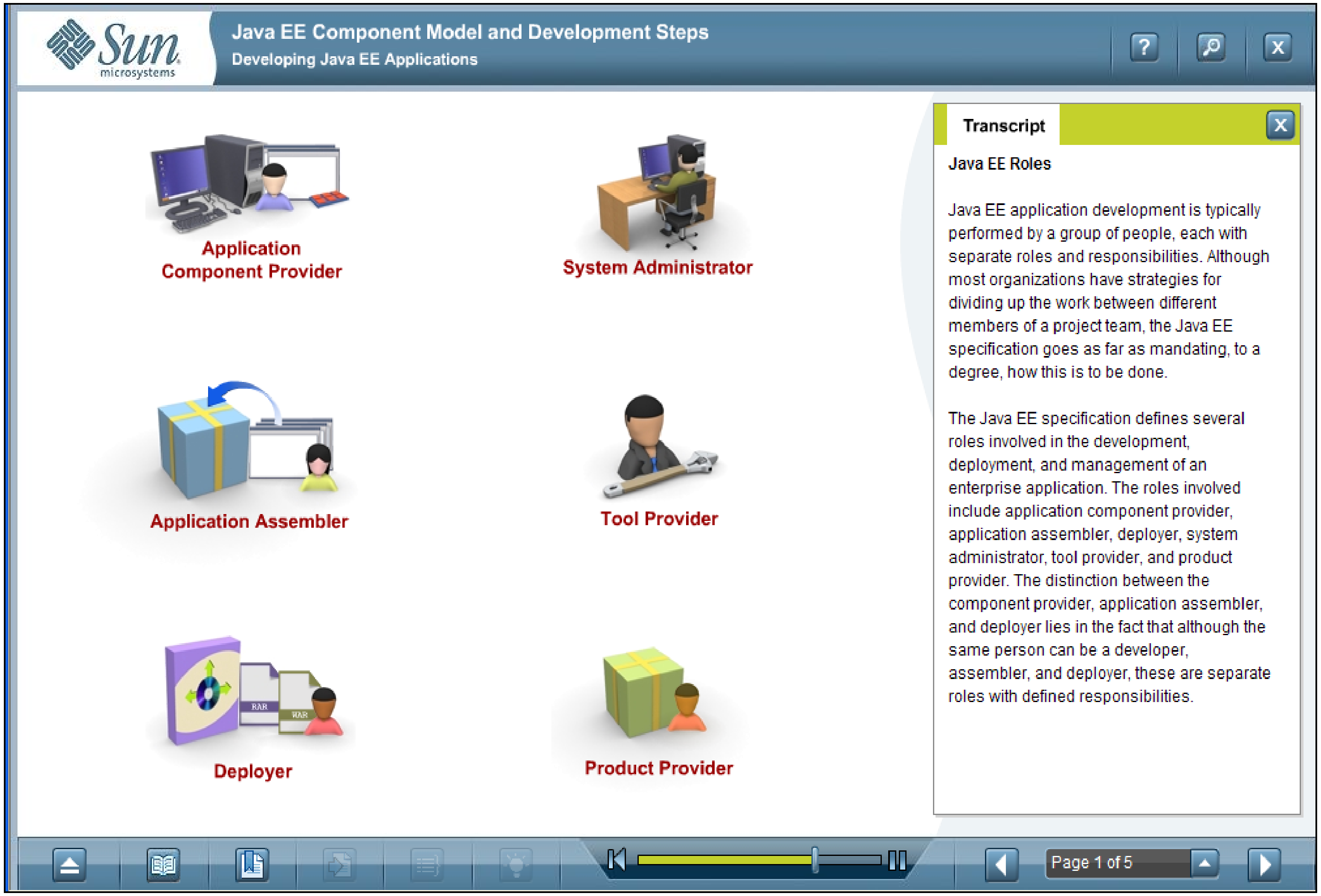


J2EE Runtime Architecture



J2EE Roles

J2EE Roles



Java EE Component Model and Development Steps

Developing Java EE Applications

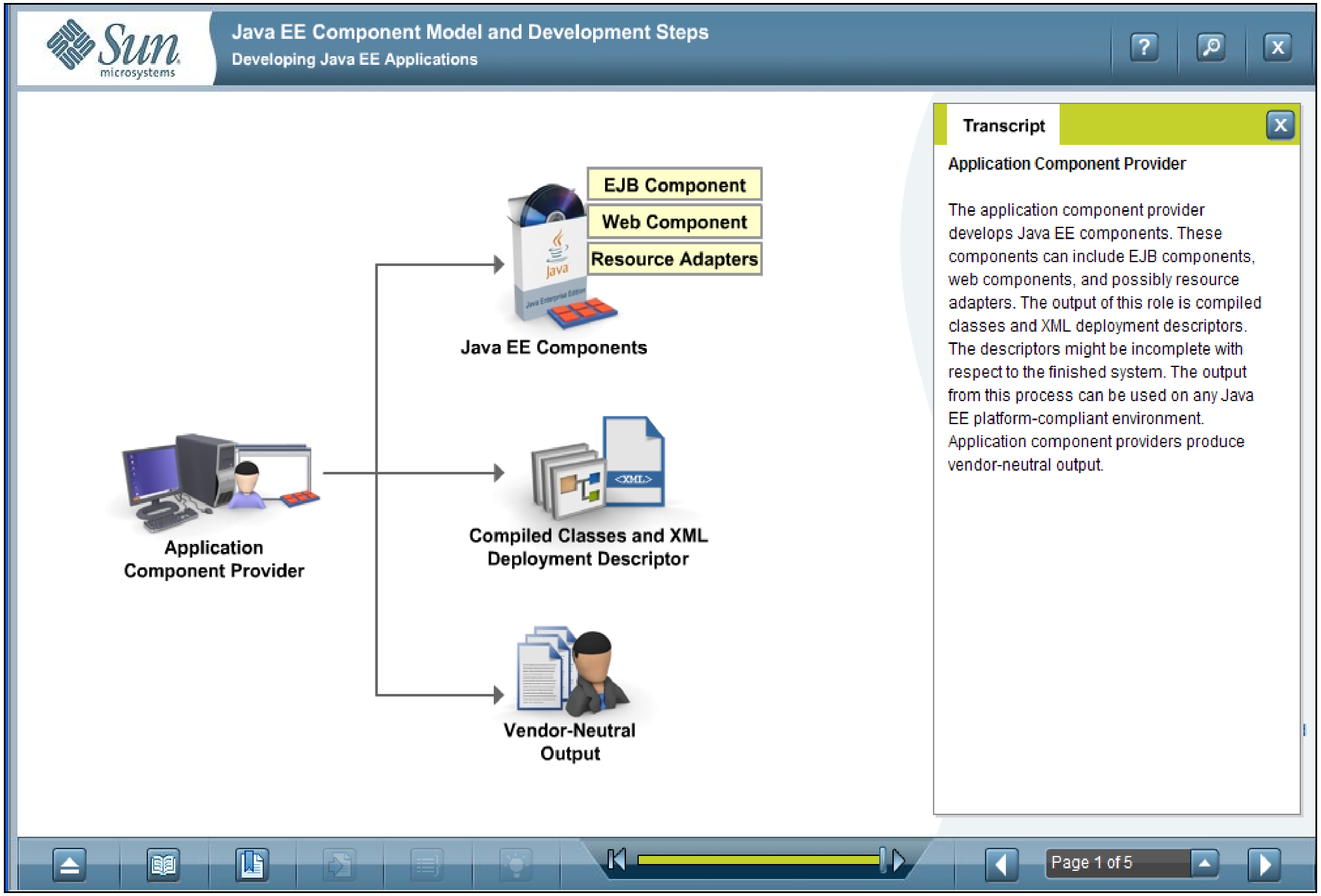
Transcript

Java EE Roles

Java EE application development is typically performed by a group of people, each with separate roles and responsibilities. Although most organizations have strategies for dividing up the work between different members of a project team, the Java EE specification goes as far as mandating, to a degree, how this is to be done.

The Java EE specification defines several roles involved in the development, deployment, and management of an enterprise application. The roles involved include application component provider, application assembler, deployer, system administrator, tool provider, and product provider. The distinction between the component provider, application assembler, and deployer lies in the fact that although the same person can be a developer, assembler, and deployer, these are separate roles with defined responsibilities.

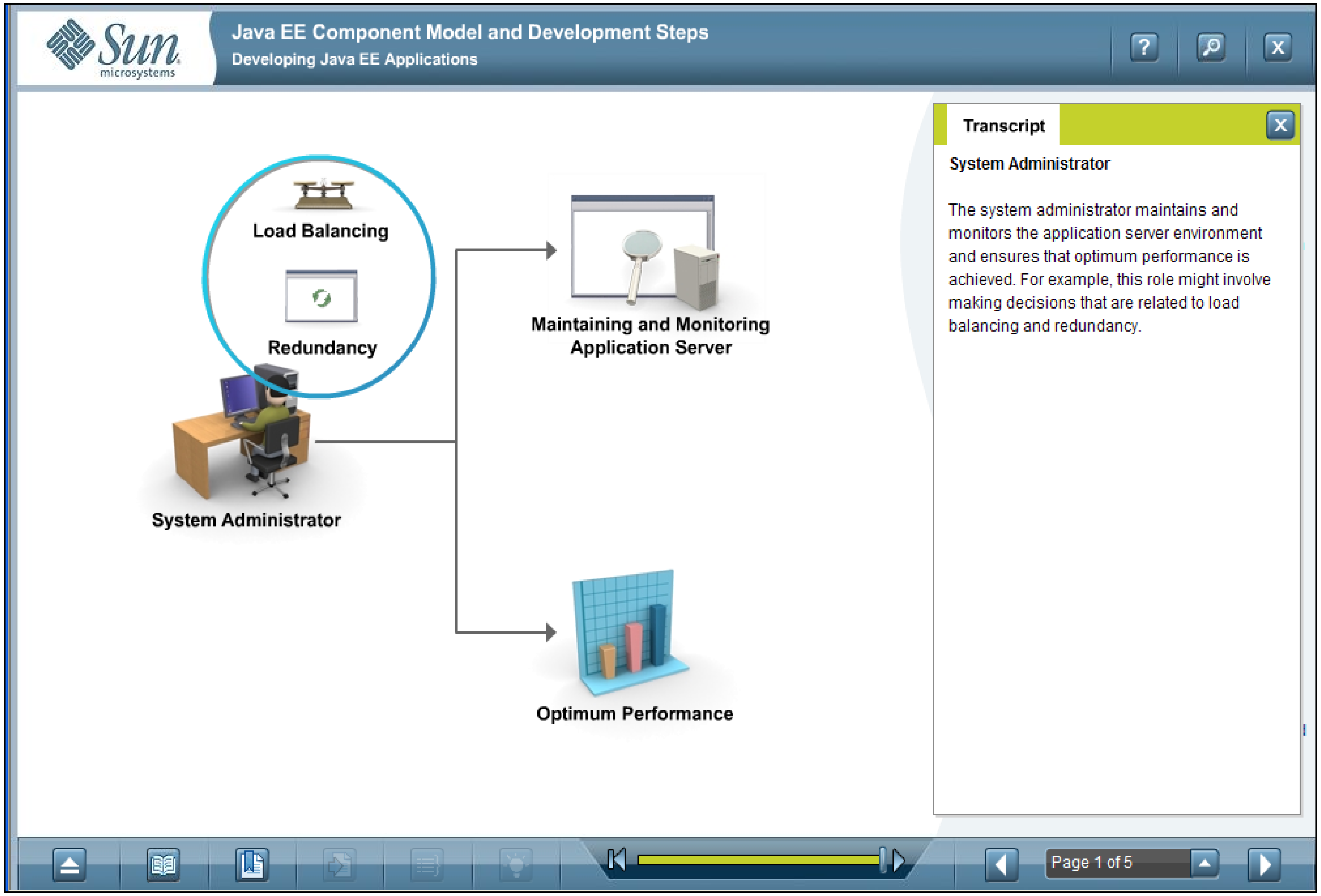
Application Component Provider


Sun microsystems
Java EE Component Model and Development Steps
Developing Java EE Applications
EJB Component
Web Component
Resource Adapters
Java EE Components
Application Component Provider
Compiled Classes and XML Deployment Descriptor
Vendor-Neutral Output
Transcript
Page 1 of 5

Application Component Provider

The application component provider develops Java EE components. These components can include EJB components, web components, and possibly resource adapters. The output of this role is compiled classes and XML deployment descriptors. The descriptors might be incomplete with respect to the finished system. The output from this process can be used on any Java EE platform-compliant environment. Application component providers produce vendor-neutral output.

System Administrator

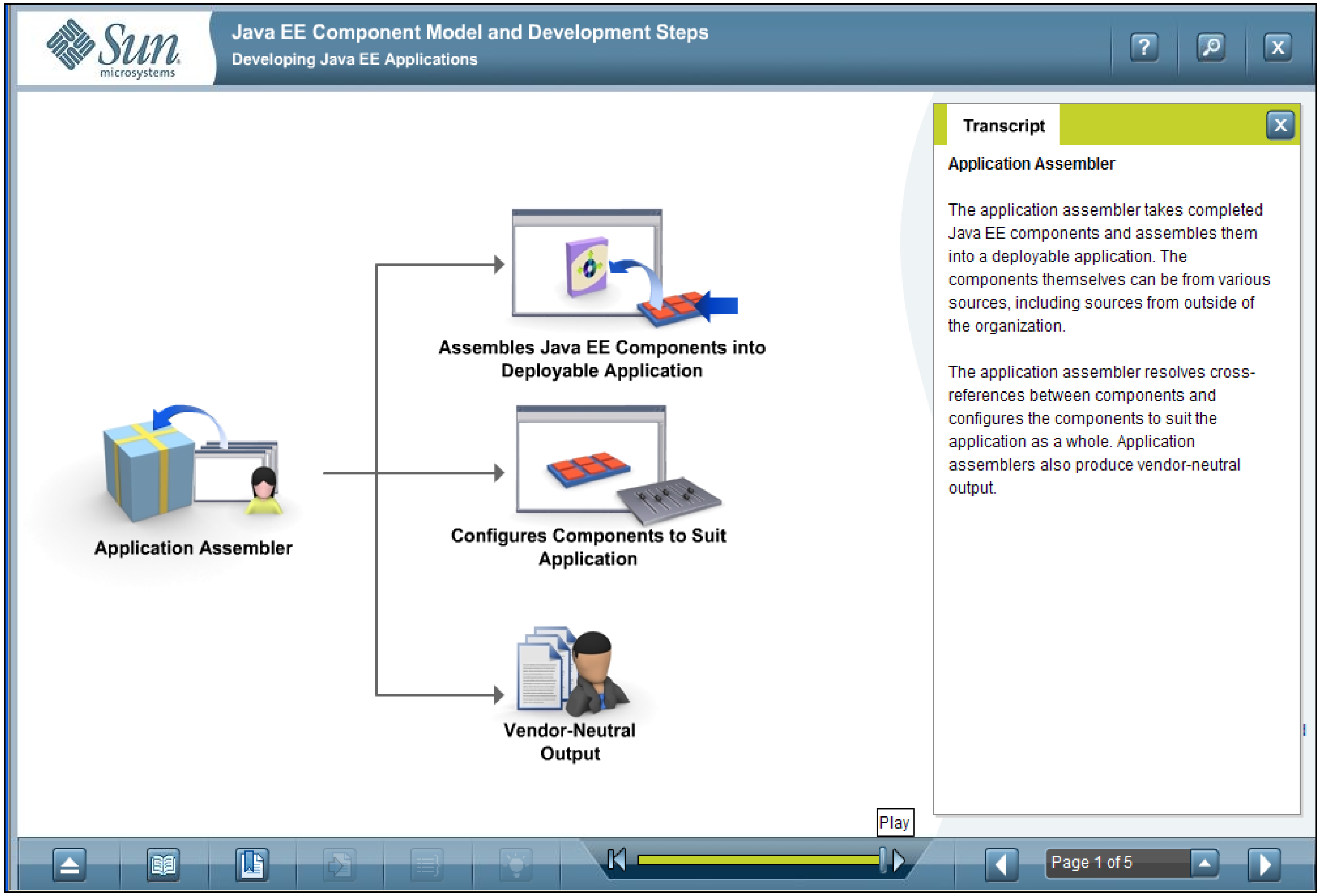

Sun microsystems
Java EE Component Model and Development Steps
Developing Java EE Applications
Load Balancing
Redundancy
System Administrator
Maintaining and Monitoring Application Server
Optimum Performance
Page 1 of 5

Transcript

System Administrator

The system administrator maintains and monitors the application server environment and ensures that optimum performance is achieved. For example, this role might involve making decisions that are related to load balancing and redundancy.

Application Assembler



Sun microsystems
Java EE Component Model and Development Steps
Developing Java EE Applications
?
X
Assembles Java EE Components into Deployable Application
Configures Components to Suit Application
Vendor-Neutral Output
Application Assembler
Play
Page 1 of 5

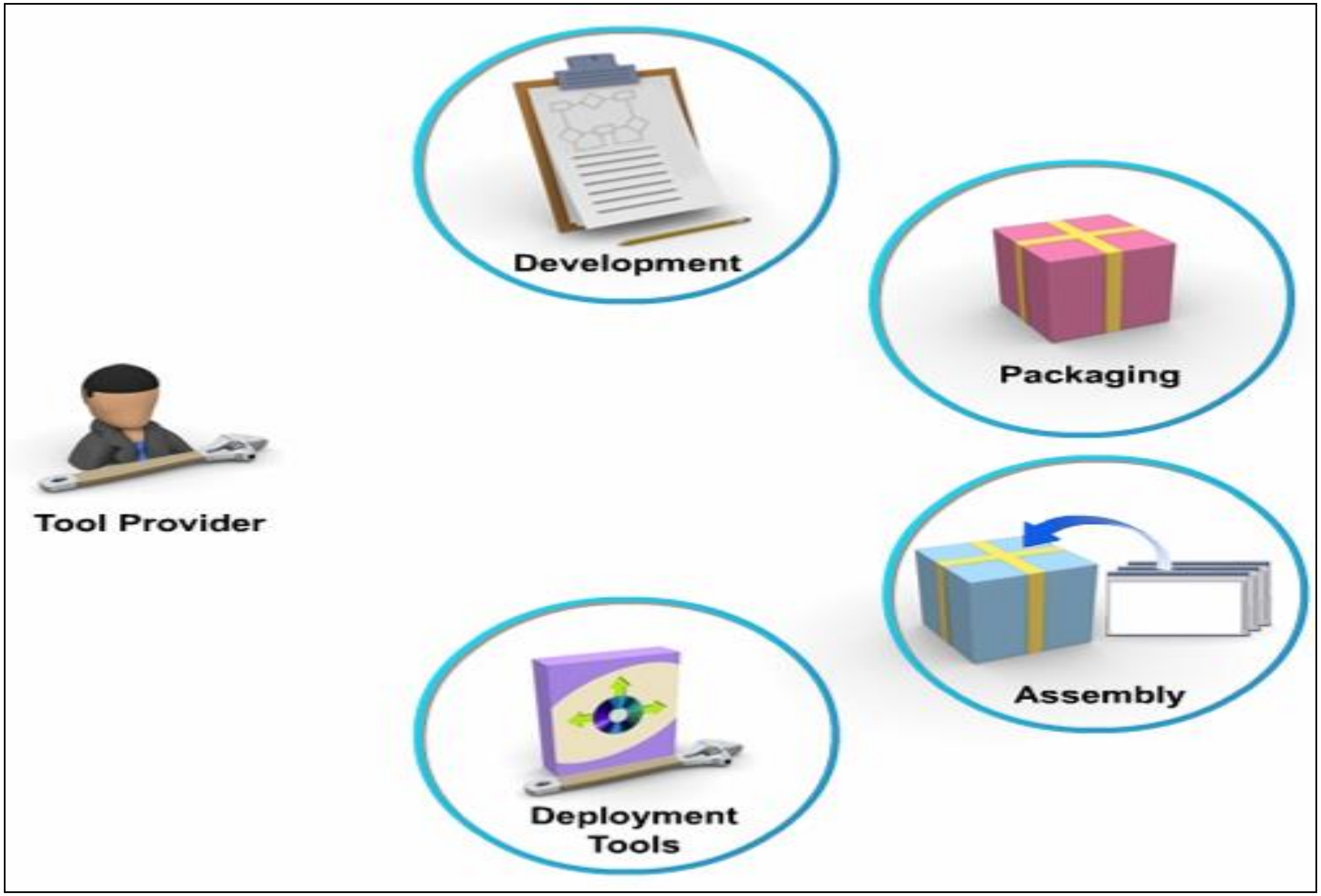
Transcript

Application Assembler

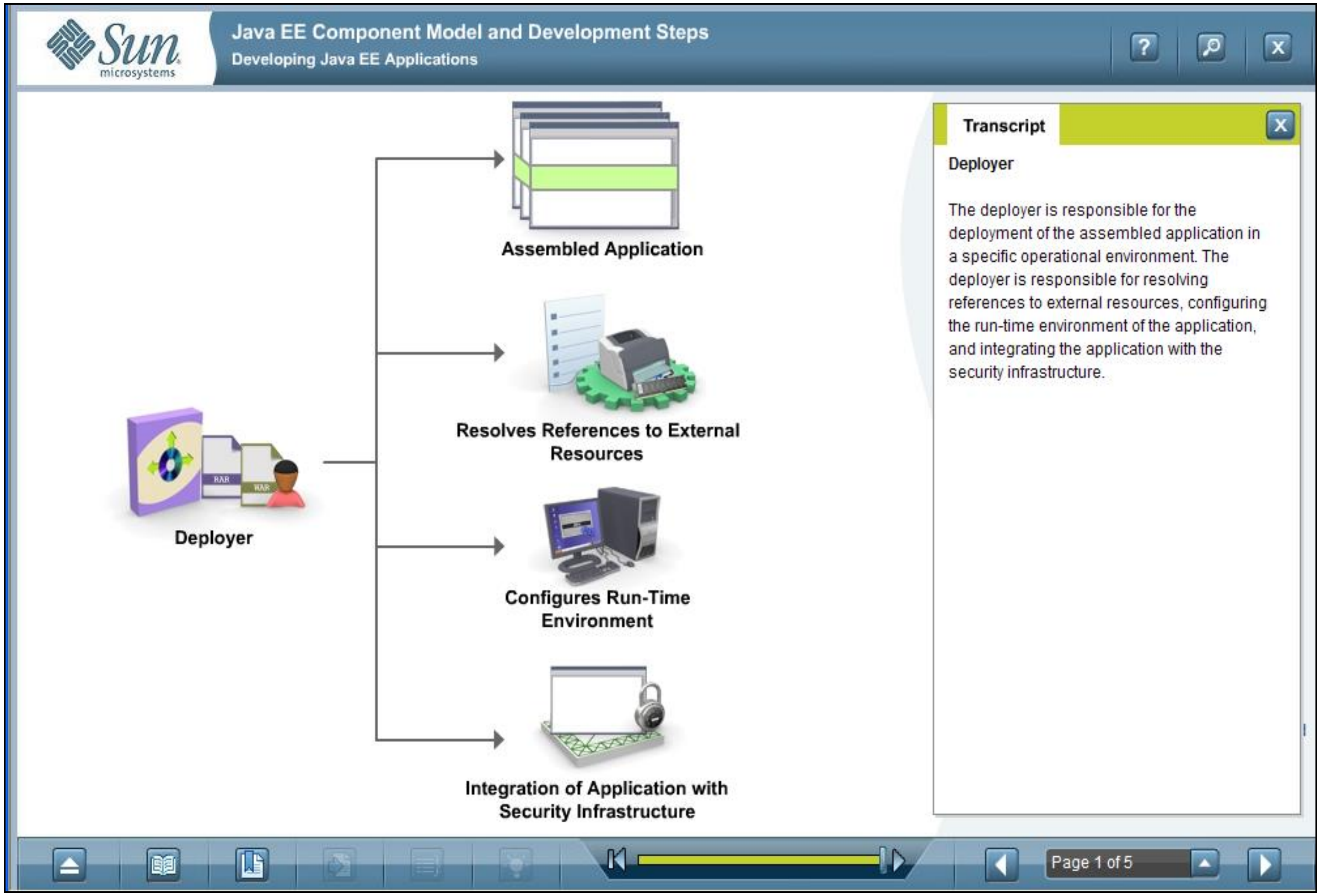
The application assembler takes completed Java EE components and assembles them into a deployable application. The components themselves can be from various sources, including sources from outside of the organization.

The application assembler resolves cross-references between components and configures the components to suit the application as a whole. Application assemblers also produce vendor-neutral output.

Tool Provider



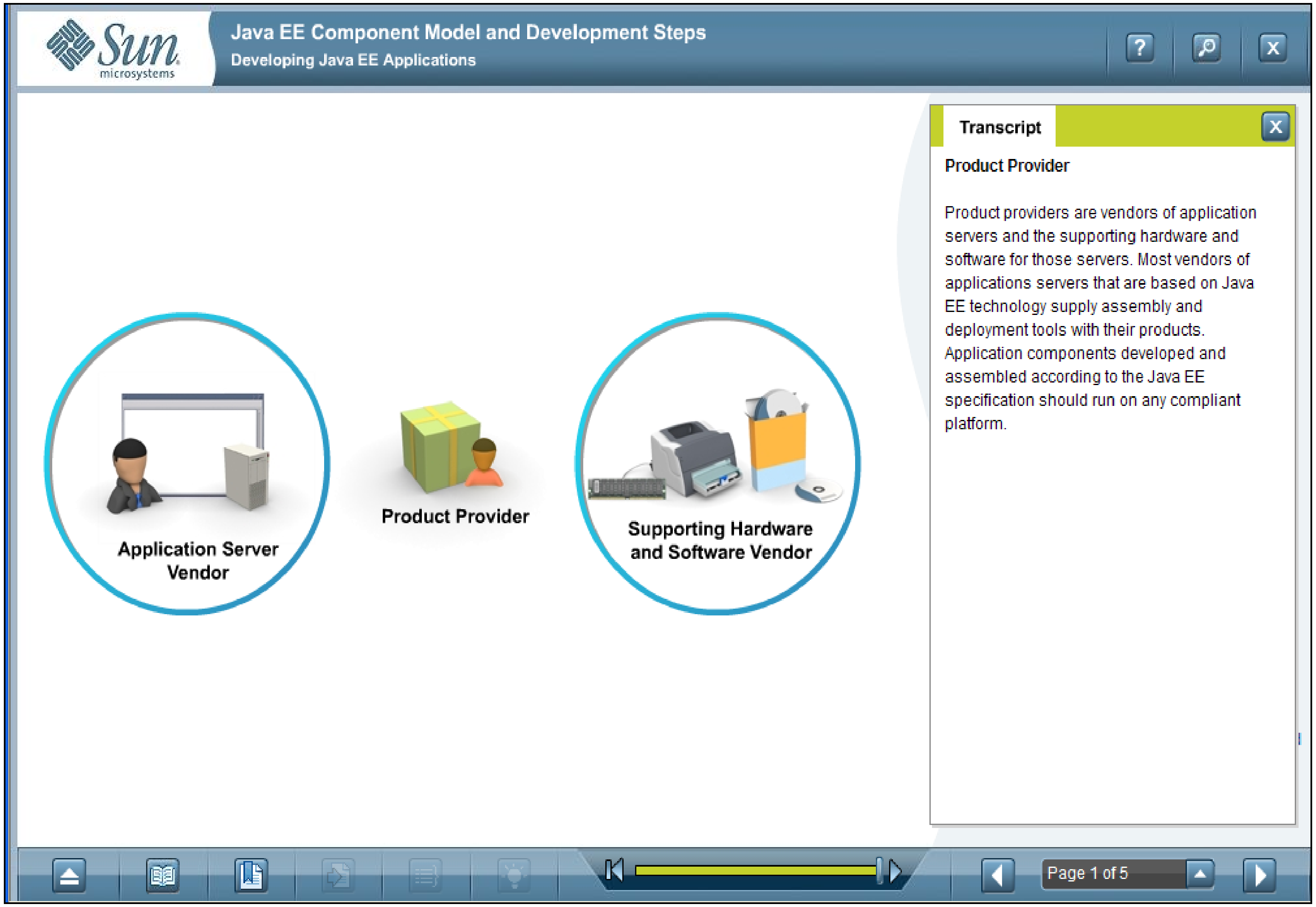
Deployer


Sun microsystems
Java EE Component Model and Development Steps
Developing Java EE Applications
Deployer
Assembled Application
Resolves References to External Resources
Configures Run-Time Environment
Integration of Application with Security Infrastructure
Transcript
Page 1 of 5

Deployer

The deployer is responsible for the deployment of the assembled application in a specific operational environment. The deployer is responsible for resolving references to external resources, configuring the run-time environment of the application, and integrating the application with the security infrastructure.

Product Provider


Sun microsystems
Java EE Component Model and Development Steps
Developing Java EE Applications
Application Server Vendor
Product Provider
Supporting Hardware and Software Vendor
Page 1 of 5

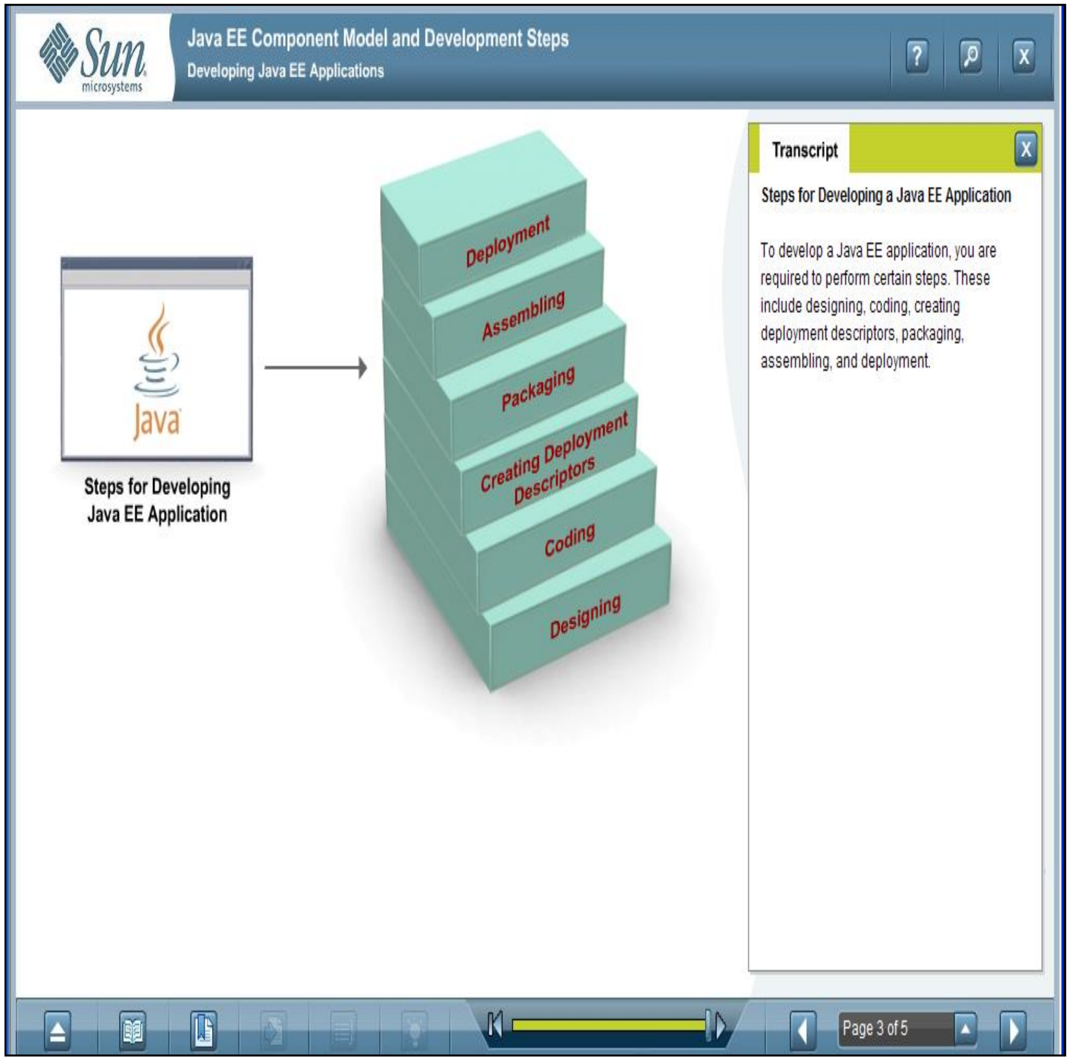
Transcript

Product Provider

Product providers are vendors of application servers and the supporting hardware and software for those servers. Most vendors of applications servers that are based on Java EE technology supply assembly and deployment tools with their products. Application components developed and assembled according to the Java EE specification should run on any compliant platform.

Steps for Developing J2EE Applications

- Developing J2EE application involves following steps:
  - Designing
  - Coding
  - Creating deployment descriptor
  - Packaging
  - Assembling
  - Deployment


Sun microsystems
Java EE Component Model and Development Steps
Developing Java EE Applications
Java
Steps for Developing Java EE Application
Deployment
Assembling
Packaging
Creating Deployment Descriptors
Coding
Designing
Transcript
Page 3 of 5

Steps for Developing a Java EE Application

To develop a Java EE application, you are required to perform certain steps. These include designing, coding, creating deployment descriptors, packaging, assembling, and deployment.

Summary

In this session, we have covered,

- What is J2EE ?
- The J2EE tiers
- Enterprise Application
- J2EE Components
- J2EE Roles

Thank You