The Java™ 2 Platform

Java 2 Platform Micro Edition (J2ME™)

Java 2 Enterprise Edition (J2EE)
Java 2 Standard Edition (J2SE)

Optional Packages

Java Card APIs

JVM

KVM
CardVM

Personal Basis Profile
Personal Profile
Foundation Profile
MIDP
CDC
CLDC
Java Technology Overview

- **Java 2 Standard Edition (J2SE)**
  - Compiler, tools, runtime and APIs for Java application development

- **Java 2 Enterprise Edition (J2EE)**
  - APIs for development, deployment and management of server-based, distributed, multi-tier and component-based applications

- **Java 2 Micro Edition (J2ME)**
  - Highly optimized version for resource-limited consumer device software development
The Java™ 2 Platform

- Java Technology Enabled Devices
- Java Technology Enabled Desktop
- Workgroup Server
- High-End Server
Evolution of Enterprise Applications

- **Single-Tier**
  - monolithic, direct connection

- **Two-Tier**
  - fat clients

- **Three-Tier**
  - Browser, Web Server, DB
  - Remote Procedure Calls (RPC-based)
  - Component-based (CORBA, RMI, DCOM)
Trends in Enterprise Applications

- Transition from single-tier/two-tier to multi-tier applications
- Transition from monolithic applications to object-based applications
- Transition from application "fat client" to a web browser client
Problems

• The "Middle Tier" is complex
• Each application must duplicate the same basic system services:
  – concurrent access to resources
  – transactional access to resources
  – load-balancing among resources
  – securing access to resources
  – managing resources
  – managing persistence of data
Solution - J2EE

J2EE provides

- a standard set of Application Programming Interfaces (APIs), and
- a component-based architecture, and
- the idea of a container to provide standardized system services to all applications,

to reduce the complexity of "Middle Tier" programming
J2EE Contents

- Platform Specification
  - Defines the standard APIs

- Reference Implementation and SDK
  - Implements the complete specification as an operational definition of J2EE

- Compatibility Test Suite
  - Tests compliance with the standard, guarantees portability

- Blueprints
  - Architecture and design guidelines
J2EE Platform Architecture
J2EE Core Concepts

- Open, published Specification
- Distributed Applications
- Component-Based
- Containers
- Packaging / Assembly
- Deployment
- Roles
J2EE Core Concepts (1)

- J2EE is an open, formal Specification
  - What must be supported, but not how
  - Agree on standard and compete on implementation
  - Sun Microsystems participates but does not control the standard

- Applications are distributed
  - Application components can run on different devices connected by a network
J2EE Core Concepts (2)

- Applications are based on components
  - A component is an application-level unit of code
  - Supported components:
    - JavaBeans (from J2SE)
    - Java Applets (client side)
    - Java Application Clients (client side)
    - Enterprise Java Beans (server side)
    - Web Components (server side)
    - Resource Adapter Components (server side)
  - A component is responsible for:
    - Presentation logic
    - Business logic
J2EE Core Concepts (3)

- Applications run in containers providing:
  - Services to components (transaction management, object distribution, concurrent access, security, persistence, resource management, life-cycle management)
  - Standardized access methods to Enterprise Information Systems (EIS), such as SAP
  - Control over application behaviour at assembly and deployment time, as well as at run time (in the code)
Components and Containers
Responsibilities

- Containers handle:
  - Concurrency
  - Security
  - Availability
  - Scaleability
  - Persistence
  - Transactions
  - Life-cycle Management

- Components handle:
  - Presentation logic
  - Business logic
A Typical J2EE Environment
J2EE Core Concepts (4)
Packaging / Assembly
J2EE Core Concepts (5)

• Deployment
  – Applications are configurable at packaging/assembly AND deployment time via deployment descriptors
  – Changes can be made without programming:
    • Transactional behaviour
    • Security characteristics (user/group, role assignments)
    • External resource references (databases, EIS)
    • Container-specific features (load-balancing, clustering)
J2EE Core Concepts (6)

Roles

- Platform/Container provider
- Tools provider
- Component provider
- Application assembler
- Deployer
- System administrator
J2EE Application Architecture

- The J2EE Application Architecture defines - but does not require - these tiers:
  - Client tier (the user interface)
    - Browser-based clients
    - Standalone application clients, including J2ME clients
    - Non-Java clients; eg, VB making HTTP requests
    - Standards on the client tier - HTTP, HTML, XML
  - Middle tier (one or more, client services and business logic)
    - Web containers for Servlets and JSPs
    - EJB containers for Enterprise Java Beans
  - Back-end tier (data management)
    - Oracle Database or EIS such as SAP
J2EE Application Architecture
Full Multi-Tier Example
J2EE End-to-End Architecture

Client-Side Presentation
- MIDP Devices
- Rich Clients
- Browsers
- B2B Application

Server-Side Presentation
- Servlet
- JSP
- Web Container

Server-Side Business Logic
- EJB
- EJB Container
- J2EE Application Server

Communication Protocols
- XHTML/WML
- SOAP/HTTP
- RMI/IIOP
- XML/HTTP

External Information Sources (EIS)
- DBMS
- ERP
- CMS
- J2CA
- CORBA
- JDBC
- SOAP/HTTP
- SOAP/HTTP
- JMS

Web Containers
- Servlet
- JSP

EJB Containers
- EJB
- EJB
Best Practices/Design Patterns

• Java BluePrint for J2EE
  – Guidelines, patterns and code examples
  – http://java.sun.com/blueprints/enterprise

• J2EE Patterns
  – Best practices and design strategies (common solutions to common problems)
  – J2EE Pattern Catalog from the Sun Java Center
Benefits for Developers

- Containers provide common services
- Freedom on choice for servers, tools and components
- Comprehensive resources available
- Integration methods for existing information systems
- Configurable security model
Benefits for IT Managers

- Applications are portable
- No vendor lock-in
- Large marketplace, many vendors to choose from
J2EE APIs and Technologies

- JDBC (Java Database Connectivity)
- Java IDL (Interface Definition Language for CORBA)
- EJB (Enterprise Java Beans)
- Java Servlets
- JSP (Java Server Pages)
- JMS (Java Message Service)
- JNDI (Java Naming and Directory Interface)
- XML APIs (JAXP, JAXR, JAX-RPC)
- JavaMail
- J2EE Connector Architecture (JCA)
- Transactions (JTA / JTS)
J2EE/EIS Integration APIs

- J2EE Technologies used in EIS integration:
  - J2EE Connector Architecture (JCA)
    - For Enterprise Information Systems (EIS); for example, SAP
  - JDBC
    - For Databases; for example, ORACLE
  - Java Message Service (JMS)
    - For Message-Oriented-Middleware; for example, SunONE Message Queue or IBM MQ Series
  - Java Naming and Directory Interface (JNDI)
    - For Directory Services; for example, LDAP Directories
  - JavaMail
    - For E-Mail Systems
JCA Overview

- **Process**
  - EIS provides a JCA-compliant adapter
  - J2EE Application Server supports JCA
  - EIS Adapter plugs into Application Server
  - J2EE Applications running in Application Server access EIS via the Adapter

- **EIS Vendors provide only one Adapter for all Application Servers**

- **EIS accessed in a standard way in all Application Servers**
JCA Components
J2EE/SAP Integration Methods

- SAP Web Application Server (mySAP)
  - "Fully J2EE Compatible"
  - Java and/or ABAP Application runs in server
  - Application --> Integration Engine --> SAP
  - Adapters allow SAP Exchange Infrastructure to use Java to talk to external systems

- SAP Java Connector (JCo)
  - SAP-specific Java adapter (deprecated in favour of JCA Adapter)

- SAP J2EE Connector Architecture (JCA) Adapter
  - J2EE Application Server supports JCA
  - SAP JCA Adapter supports JCA
  - J2EE Application Server application calls SAP via the SAP Adapter
J2EE Releases

• **J2EE 1.4:**
  - New APIs for core Web Services protocols
  - New management and deployment APIs
  - New versions of JSPs, EJBs, Connector APIs

• **J2EE 1.5**
  - Simplification of J2EE Application Development
  - Enhance the influence of J2EE in the entire Java Development Community
  - XML Standards (XMLdsig, XML Encryption, WS-Security)
  - APIs (Portlets, Java Server Faces, JAXB)
Resources

• Starter Kit
  – J2SE, J2EE and J2ME JDK software
  – Java training and code samples
  – Development tools, including Studio 5 IDE
  – Web Services Tutorial and Building Services on the J2EE Platform
  – J2EE Middleware
Resources continued

  - Blueprint - "Designing Enterprise Applications with the J2EE Platform, 2nd Edition"

- [http://java.sun.com/j2ee](http://java.sun.com/j2ee)
  - Software, tutorials and documentation

- [http://java.net](http://java.net)
  - Java Developer Portal

  - Sun Application Server product web site

  - Sun Studio product web site
J2EE APIs and Technologies

- JDBC (Java Database Connectivity)
- Java IDL (Interface Definition Language for CORBA)
- EJB (Enterprise Java Beans)
- Java Servlets
- JSP (Java Server Pages)
- JMS (Java Message Service)
- JNDI (Java Naming and Directory Interface)
- XML APIs (JAXP, JAXR, JAX-RPC)
- JavaMail
- J2EE Connector Architecture (JCA)
- Transactions (JTA / JTS)
JDBC

• Standard API for accessing *tabular* data
  – Connect to a database or tabular data source
    (including spreadsheets and flat files)
  – Send SQL statements
  – Process the results

• Packages:
  – java.sql
  – javax.sql

Java IDL

- Adds CORBA capability
- Enables invocation of remote network services using OMG IDL and IIOP
- Includes an ORB for distributed computing using IIOP
- J2SE 1.3 includes the IDL-to-Java compiler
Enterprise Java Beans

- Standard server-side component
- Session Beans
  - Process and task management
- Entity Beans
  - Persistent objects in a database
- Message-Driven Beans
  - Send/receive asynchronous JMS messages
Java Servlets

• Component-based, platform-independent method for building web-based applications
• An applet that runs on the server side
• Generates dynamic content
• Packages:
  – javax.servlet
  – javax.servlet.http
• http://java.sun.com/products/products/servlet
Java Servlets vs. CGI

Request CGI1
CGI Based Webserver
   Child for CGI1

Request CGI2
CGI Based Webserver
   Child for CGI2

Request Servlet1
Servlet Based Webserver
   JVM
   Servlet1

Request Servlet2
Servlet Based Webserver
   JVM
   Servlet2
Java Server Pages

- Extension of Java Servlets
- Separates the user interface from content generation
  - Presentation in HTML or XML/XSLT
  - Business logic in JavaBeans or Custom Tags
  - Easier to maintain

- Packages:
  - javax.servlet.jsp
  - javax.servlet.jsp.tagext

Java Message Service

- Access to Enterprise Messaging, or Message Oriented Middleware, via standard API
- Requires a Message Provider (Sun Message Queue is built into Sun Application Server)
- Package javax.jms
- http://java.sun.com/products/jms
Java Naming and Directory Interface

- Standard extension to the Java Platform
- Unified interface to naming and directory services
- Class libraries and service providers for:
  - LDAP, DNS, NIS/NIS+, CORBA COS Naming, RMI Registry, file system
XML APIs

- **JAXP (Java API for XML Processing)**
  - Processing of XML documents using DOM, SAX and XSLT

- **JAXR (Java API for XML Registries)**
  - Bindings for Web Services Registries - ebXML Registry and UDDI Registry v2.0

- **JAX-RPC (Java API for XML-Based RPC)**
  - Core API for web services development/deployment
  - Builds SOAP-based web services
JavaMail

- Platform-independent and protocol-independent API to build mail and messaging applications
- Requires JavaBeans Activation Framework extension (javax.activation package)
J2EE Connector Architecture

- Standard architecture for connecting the J2EE platform to heterogenous Enterprise Information Systems (EIS) - such as SAP
- EIS vendor provides a standard resource adapter for its EIS
- Adapter plugs into the Application Server, providing connectivity between the EIS and the enterprise application
- http://java.sun.com/j2ee/connector
Transactions

• Supports distributed transaction management

• JTA (Java Transaction API)
  – high-level, implementation-independent, protocol-independent API allowing access to transactions

• JTS (Java Transaction Service)
  – Implementation of a Transaction Manager supporting JTA
  – Implementation of Java mapping to OMG Object Transaction Service (OTS) 1.1

• http://java.sun.com/j2ee/transactions.html
J2EE Flexibility

- J2EE specifies, but does not require, multi-tier architectures
- Both the Web and EJB Containers are optional
- There is no bias or preference for one architecture over another; however, there can be preferred ways of doing things
Possible Architectures

Following are just three possibilities:
- Web-Oriented application
- Standalone-Client application
- Multi-Tier application
Web-Oriented

- Cases where EJB Container components would be overkill and deliver poor performance
- Web Container hosts presentation logic (JSP) and business logic (Java Servlets)
- Web Container components use JDBC, JMS and JCA to access EIS Resources
Standalone-Client

- Client accesses EJB Container components directly
  - EJB Container components access EIS Resources via JDBC, JMS and JCA
- Client accesses content provided by Web Container components
  - Client handles display logic
  - Web container components handle business logic, may use EJB Container components for EIS access, or may access directly
- Client access EIS directly using JDBC, JMS and JCA
  - Client handles all logic
Multi-Tier

- Client is a Web Browser
- Presentation and business logic provided by Web Container components
  - Java Servlets recommended for request processing and application control
  - JSPs recommended for user interface/display logic
- EJB Container components manage access to EIS resources using JDBC, JMS and JCA
Logical View

• Tries to use existing security services

• Role-based Security Implementation
  – Role - Application Developer
    • Declarative Security (deployment descriptors)
    • Programmatic Security (Java code)
  – Role - Application Deployer
    • Configures Security Policy
  – Role - Application Container (J2EE Container)
    • Enforces Security Policy
Mechanisms

- Authentication
  - Web-Tier Authentication
  - EJB-Tier Authentication
  - EIS-Tier Authentication

- Authorisation
  - Declarative and Programmatic
  - Provider and User of component

- Signing

- Encryption
  - Integrity and Confidentiality

- Auditing
Why Use EJBs

- Takes advantages of the server-side component model; that is, container services
- Separates business logic from system code
- Enables component portability across J2EE Servers and Operating Systems
- Enables configuration at deployment time as well as at development time
EJB Architecture

Client → EJB Home → EJB Object → Enterprise JavaBeans™ Component → Database or Component
EJB Component Types

- **Session Bean**
  - Implementation of Workflows or Services
  - Stateless (no session state from request to request)
  - Statefull (session state persists across requests)

- **Entity Bean**
  - Represents state and behaviour of an actual Object
  - State is persistent across requests and clients
  - Container-Managed Persistence
    - EJB Container manages database interaction
  - Bean-Managed Persistence
    - Programmer codes SQL statements to manage database interaction

- **Message Driven Bean**
  - Allows the sending and receiving of JMS Messages within an EJB Container
J2EE and .NET Development

- **J2EE**
  - One language (Java)
  - Many platforms (Solaris, Windows, Linux...)

- **.NET**
  - One platform (Windows)
    - Mono effort in progress (Linux and UNIX)
  - Many languages (C#, Managed C++, VB.NET...)