Architecture Refresher

Why use J2EE?

Distributed Applications

Definition of Distributed Application (from Wikipedia):

> An application made up of distinct components running in separate runtime environments, usually on different platforms connected via a network. Typical **distributed applications** are two-tier (client-server), three-tier (client-middleware-server), and multitier (client-multiple middleware-multiple servers)

Source: http://java.sun.com/j2ee/1.4/docs/glossary.html

Client/Server

- 2 Tier

  Client — Application
  - UI
  - Business Logic

  Server — Database

C/S Disadvantages

- Performance depends on client PC
- High network traffic
- Maintenance
- Client Consistency

3-Tier Architecture

View	Client	UI
Controller	Server	Bus. Logic
Model	Server	Data

3-Tier Advantages

- Can support many different user interfaces (but must have a good API to business logic)
- Can support many different data sources (but must have a good API to business logic)

Multi-Tier Architecture

- Divide business logic layer into functional subdivisions (not physical)
  - Presentation Logic – may differ for different UI's
  - Business/Application logic
  - Infrastructure Services
- Still follows MVC pattern

Enterprise Architecture

- Enterprise vs Collection of silos/stovepipes
- Allow multiple applications that work together in the middle tier
- Often have components that are used by several different applications (reusable)

Why Use Java?

- Platform independent
- Reusability
  - Objects (J2SE)
  - Components (J2EE)
- Modularity
- J2EE provides for distributed applications using a component-based application model
- Container Services

Distributed Computing Issues

- Historically client/server
  - Issues with threads, security, transactions, etc.
- CGI/Perl
  - Issues with efficiency and scalability
- CORBA
  - Issues with overhead, performance, portability

Other Design Issues

- Importance of modularity
- The best tools can't prevent a bad design
- Potential historical servlet issues
- Potential historical JSP issues
- Relevance of separation of tiers, tasks

J2EE Platform

- J2EE is a platform
- Provides common components as built-ins
- Specifies contracts for 3rd parties to follow to provide services
- Containers provide access to components in a standard manner (non-proprietary)

J2EE Components

- Self-contained functional units
- Application clients and applets run on the server
- JSP and Servlet (web) components run on the server
- EJB (business) components run on the server

J2EE Web Clients

- Run in browsers
- Server generates HTML or XML
- Transmits to client over HTTP
- Web container application components include JSP and Servlets
- Web container is responsible for handling requests and generating responses

J2EE EJB Clients

- Application clients access EJB components over RMI-IIOP
- Web container components (JSPs and Servlets) can also access EJB components over RMI-IIOP

J2EE Containers

Runtime environment to manage application components and provide access to APIs

- Web Container – hosts servlets and JSPs
- EJB Container – hosts EJBs
- Applet Container – for running applets
- Application Container – for running standard java applications (JVM)

Container Services

- Interface between components and low-level functionality
- Deploy components in a container
- Provide configurable support services
- Provide non-configurable services

J2EE APIs

- EJB – Enterprise JavaBeans
- Java Servlet Technology
- JSP – JavaServer Pages
- JSTL – JSP Standard Tag Library
- JavaServer Faces
- JMS – Java Messaging Service
- JTA – Java Transaction API
- JavaMail
- JAF – JavaBeans Activation Framework

J2EE APIs

- JAXP – Java API for XML Processing
- JAX-WS – Java API for XML Web Services
- JAXB – Java Architecture for XML Web Binding
- SAAJ – SOAP with Attachments API for Java
- JAXR – Java API for XML Registries
- J2EE Connector Architecture
- JDBC – Java Database Connectivity
- Persistence
- JNDI – Java Naming and Directory Interface
- JAAS – Java Authentication and Authorization Service

Web Services

- Web-based applications
- Use XML-based standards and transport protocols (SOAP over HTTP)
- J2EE provides support for web services via container services
  - Translate XML data streams
- Beyond the scope of this course

J2EE Applications

- One or more units, each having:
  - One or more functional components (EJB, Servlet, JSP, etc)
  - Optional Deployment Descriptor
- Must be deployed to a J2EE platform
- Packaged in an ear file

EAR file

- Jar file with an ear extension
- Means of combining reusable components into different applications
- Contain components and deployment descriptors