Overview of the J2EE Specification

Dave Landers  
BEA Systems, Inc.  
Accelerated Development Center

dave.landers@bea.com
Agenda

• What is J2EE?
• J2EE Architecture
• Application Programming Model
• Roles
• Contracts (the APIs)
• Naming, Security, Deployment
What is J2EE?

• A Standard platform for Enterprise Applications
  – Distributed
  – Multi-Tier
  – Thin Client
• Use standard services
• Know what to expect from any J2EE platform
What is J2EE?

- Components
- Services
- Protocols
- Architecture
- Application Model
What is J2EE?

• Infrastructure for Enterprise Applications
  – Transaction management
  – Object life-cycle
  – Resource pooling

• Developer can concentrate on business logic
What is J2EE?

• Application Programming Model
  – Something to start with when you architect complex, multi-tier systems
  – Encapsulate layers of functionality in specific component types
    • Client, Servlet, EJB, Database, etc.
What is J2EE?

• Services
  – J2SE
  – EJB
  – JDBC
  – Servlets
  – JavaMail
  – etc.

• Protocols
  – HTTP
  – HTTPS
  – SSL
  – XML
  – HTML
  – RMI
  – IIOP
  – etc.
The J2EE Architecture

- Two views
  - Platform Architecture
    - Runtime
    - Services
    - Infrastructure
  - Application Model
    - Components
    - APIs
J2EE Platform Architecture

• Provides APIs to standard services and resources
• J2EE compliance guarantees availability of services and resources
• Based on Container concept
Containers

- Hides complexity, enhances portability
- Runtime support for Application Components
  - Transparantly inject services around components
  - Transactions
  - Life Cycle and State management
  - Security
  - etc.
Containers

- Access to J2EE Services
  - via the APIs
- Restricts access to denied services
  - File Access
  - Sockets
  - etc.

- Remember the Applet “Sandbox”?
  - J2EE calls it the Applet Container
Containers

• Applet Container
  – Provided by Browser or Plugin

• Application Client Container
  – J2SE plus JMS, RMI, JNDI, JDBC

• Web Container
  – For Servlets and JSP

• EJB Container
  – EJBs
J2EE Application Model

• Application Components
  – Application Clients
  – Applet Clients
  – Web Components
    • Servlets and JSP
  – Enterprise Java Beans Components
J2EE Application Model

- Categories of Components
  - Deployed, Managed, and Executed on J2EE server
    - Servlets, JSP, EJB
  - Deployed and Managed on J2EE Server, but executed on Client machine
    - Applets and HTML pages
  - Deployment and Management not defined by J2EE
    - Application Clients
J2EE Application Model

- In addition to the Components, J2EE defines standard Services (APIs)
  - Containers provide access to these services
  - JTA, JDBC, JMS, JNDI, JavaMail, JAF
- Communication Protocols are also specified
  - HTTP, HTTPS, SSL, RMI-JRMP, RMI-IIOP
J2EE Application Model

- Split Functionality into Components
  - If using MVC
    - “View” components in HTML, JSP, or Servlets
    - “View-Controller” in Servlets or JavaBeans
      - User Interface logic & processing
    - “Model-Controller” in EJB (Session Beans)
      - Business logic
    - “Model” in EJB (Entity Beans) and Database
      - Data Model
- Also have access to services (APIs)
J2EE Platform Roles

• Each Role has specific Responsibilities
• Roles help define “who does what”
• J2EE defines typical primary roles
• Subsets of some Roles are defined in Component specifications
  – EJB, JSP, Servlet
• An individual developer may perform many (or all) roles for a project
J2EE Roles

• J2EE Product Provider
  – Application Server Vendor

• Application Component Provider
  – EJB or Servlet developer
J2EE Roles

• Application Assembler
  – Takes multiple components and assembles them into an application

• Deployer
  – Installs application
  – Generate server-specific classes
  – Configures application for server
  – Starts application
J2EE Roles

• System Administrator
  – Networking and computing infrastructure
  – Oversees “runtime well-being” of application

• Tool Provider
  – Vendor for Tools used in deployment and packaging of application components
Roles - why do I care?

• Roles identify distinct responsibilities

• Gives you a language to use when...
  – Defining your development process
  – Setting up your build & test environment
  – Delivering code to customers
  – Purchasing components from vendors
Roles - Advantages

• Encourages and Enables J2EE products and services
  – “Breakpoints” in the process delimited by boundaries between the Roles
  – Gives vendors a clear statement of Responsibility for what they should deliver

• Component and Application vendors
• Service Providers (deployment and Hosting)
  – Example: ejip.net, LoudCloud doing hosting
Roles - Advantages

• Helps spec authors ensure proper compartmentalization of functionality
  – Example: EJB Deployment Process
    • Defined by Role
    • Each Role delivers a jar file to the next Role
J2EE Contracts

• The APIs
  – Services
  – Protocols

• What you usually think of when you think J2EE
J2EE 1.2 Required APIs

• EJB 1.1
• Servlet 2.2
• JSP 1.1
• JDBC + 2.0 Extensions
• etc.
EJB 1.1

- Enterprise Java Beans
- Business Components
- Multi-tier architecture
- Session Beans
  - For behaviors
  - Stateless or Stateful
- Entity Beans
  - Data Model
  - Persistent to database
Servlet 2.2

- “Active” web components
- Provide Dynamic Content
- HTTP “front” to application components
JSP 1.1

- Java Server Pages
- Extension of Servlets
- Like HTML with embedded Java code
- View (of MVC pattern)
- Tag Extensions allow HTML Authors to drop in dynamic content without touching code
- Discipline required to stick to OO and Design principals
JDBC 2.0 Extensions

- Java Data Base Connectivity
- JDBC Core API included in J2SE
- J2EE adds some of JDBC 2.0
  - DatabaseMetaData and ResultSetMetaData
  - Stored Procedures
  - Batch Updates
    - Non-batching implementations allowed
RMI-JRMP

- Remote Method Invocation
- JRMP is the “Default” RMI protocol
- Distributed objects and methods
- Access to services and components
  - Access for clients
  - The J2EE platform can be distributed across several JVMs and/or machines
    - Example: Web Server & Servlet Engine separate from EJB server
RMI-IIOP / JavaIDL / CORBA

• CORBA interoperability
• JavaIDL is included in J2SE
  – org.omg packages
• IIOP is CORBA’s protocol
• All application components (except applets) can be clients of RMI-IIOP
• Only Application Clients can export RMI-IIOP objects
• EJB access via IIOP encouraged but not required
JMS 1.0

• Java Messaging Service
  – Asynchronous communication among distributed components
  – Publish / Subscribe queues
  – Point-to-point communication
  – Messages can be durable (backed by database)
  – Can specify guaranteed message delivery

• Transactional
JMS

- The JMS API is required by J2EE
- However, the implementation of (or access to) the principal interfaces is not required
  – ConnectionFactory and Destination
- Will be required in the future
JNDI 1.2

- Java Naming and Directory Interface
- Lookup of objects mapped to a name
- J2EE only requires lookup of things in the java: namespace
  - EJBHome objects
  - JTS UserTransaction objects
    - java:comp/UserTransaction
  - JDBC DataSource objects
  - JMS ConnectionFactory and Destination (if supported)
JTA 1.0

- Java Transaction API
- Declarative and Programmatic demarcation
  - Deployment Descriptors
  - UserTransaction object
  - Available to Application Components
  - Not required for Application Clients or Applets
JTA

• Not Required:
  – Nested Transactions
  – Multiple Databases, Multiple J2EE Servers
  – XAResource support
    • X/Open standard XA interface for resources in a distributed transaction environment
Other APIs and Protocols

- JavaMail 1.1 / JAF 1.0
- XML
- HTTP
- HTTPS / SSL
Naming

- JNDI used to access resources
  - UserTransaction
  - EJBHome
  - EJB Components access to Environment Properties
  - Resource Factories
    - JMS factories
    - JDBC DataSource
    - JavaMail
Security

- Declarative and Programmatic
  - Deployment Descriptors
  - EJB: isCallerInRole and isCallerPrincipal methods
  - Servlet: isUserInRole and isUserPrincipal methods

- Role Mapping

- HTTPS / SSL
Applications are composed of
- One or more Components
- Deployment Descriptors

Application Life Cycle
- Create Components
- Collect Components into Modules
- Assemble into Application (ear-file)
- Deploy Application and/or Modules to a Server
Future of J2EE

- J2EE 1.3 preliminary
  - Community Process
- Proposed:
  - JMS Required
  - Connector Architecture
  - EJB 2.0
  - JSP 1.2, Servlets 2.3
  - XML & XML Data Binding
  - SQLJ
References

• J2EE Specification
  http://java.sun.com/j2ee/download.html

• Designing Enterprise Applications with J2EE ("Blueprints")
  http://java.sun.com/j2ee/blueprints/

• Simplified Guide to J2EE

• Specifications for Services, etc.
  http://java.sun.com/j2ee/docs.html

• Community Process for J2EE 1.3