Modeling the J2EE in UML and Rational Rose
Agenda

- J2EE
  - Terms and definitions
- EJBs
  - Terms and definitions
- Servlets and JSP
  - Terms and definitions
- Unified Modeling Language (UML)
- Rose 2001 demo
- Wrap and Q&A
Java 2 Enterprise Edition (J2EE)

- Defined standards for enterprise application development
  - J2EE Platform
  - Enterprise JavaBeans 1.1
  - Servlets 2.2
  - Java Server Pages 1.1
  - Others ..

- Released in December 1999
- Contains a standard set of Java technologies (APIs) for enterprise-class server-side applications written in Java
- The set of APIs enable client and server applications, applets, and servlets to access common server-side systems. This architecture of APIs simplifies the complex server side environment.
The Benefits of J2EE

- Simplifies the complex task of writing distributed, reliable, scalable, secure applications:
  - Provides a common application model available to all developers as a starting point
  - Provides a standard platform for hosting applications (different yet same!)
- Provides a generic infrastructure
  - Introduces a generic API to provide compatibility across vendors
- Provides a cleaner partition between development and deployment
  - Decoupling the two helps the Write Once, Run Anywhere (WORA) mantra
  - Enhances componentization
J2EE and the N-tier Approach

Presentation
- Web Server
  - Servlets
  - JSPs
- Thin client (Browser)
  - Applets
  - Java Beans
  - HTML...

Business Logic
- App Server
  - EJBs

Data & Other systems
- Common Services
  - Data base Connectivity
  - Naming and Directory lookup
  - Transactions
  - Mail,
  - Remote Method Invocation, etc.

Network

Web Server

App Server

Data
EJBs
What Are Enterprise JavaBeans (EJB)?

- A specification defining a server-side Java framework for
  - Distributed, transactional, secure components

- Vendors (like BEA, IBM, SilverStream, Oracle, etc..) are responsible for using this specification in order to make EJB-compliant Servers

- EJBs allow developers to write Java business logic, without having to worry about implementing the surrounding framework

- Vendors must supply tools that automatically generate distribution, transaction and security behavior
Deployment Descriptors

- How the EJB developer provides the EJB deployer with an implementation checklist
  - Transaction attributes
  - Security – access control and identification attributes
  - Environment properties – customizable properties for the deployer, i.e. the physical location of a database
  - Class names of the EJB classes – for the JNDI database
  - Home and remote interface class names
  - Name of the primary key type – an entity EJB may have a key that represents the table row it represents
  - Any server specific information
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ejb-jar PUBLIC "-//Sun Microsystems, Inc.//DTD Enterprise JavaBeans 1.1//EN" "http://java.sun.com/j2ee/dtds/ejb-jar_1_1.dtd">
<ejb-jar>
  <enterprise-beans>
    <session>
      <ejb-name>shoppingCartEJB</ejb-name>
      <home>shoppingCartHome</home>
      <remote>shoppingCart</remote>
      <ejb-class>shoppingCartEJB</ejb-class>
      <session-type>Stateful</session-type>
      <transaction-type>Container</transaction-type>
      <env-entry>
        <env-entry-name>email</env-entry-name>
        <env-entry-type>java.lang.Boolean</env-entry-type>
        <env-entry-value>true</env-entry-value>
      </env-entry>
    </session>
  </enterprise-beans>
</ejb-jar>
Session EJB

- An object that represents a transient conversation with a client
  - May be thought of as a logical extension of the client program on the server
  - Executes a business function or manipulates data safely
  - Considered private to the client and cannot be shared with other clients
  - Maintains client-specific session information, called “conversational state”
  - Generally not recoverable following a system crash or container restart (unless the EJB developer codes it)
Stateless and Stateful Session EJBs

- **Stateless Session EJB**
  - Does not save state information between method calls
  - Example: CreditCardAuthorization object

- **Stateful Session EJB**
  - Saves conversational state between method calls
  - Example: ShoppingCart object

- The state management options for a session EJB are defined in the `StateManagementType` attribute of the 1.0 deployment descriptor, or the `<session-type>` tag of the 1.1 deployment descriptor.
Entity EJBs

- An object representation of persistent data that is maintained in a permanent data store, such as a database
  - Inherently stateful
  - A primary key identifies each instance of an entity EJB
  - Created by inserting data directly into a database or by creating an object (using an object factory Create() method)
  - May be accessed by multiple clients simultaneously
    - By default, not reentrant
    - The container ensures no other method is called on the same instance until the first one has completed. It also ensures the same method is not called concurrently on the same entity EJB by multiple threads of execution.
    - From the developers’ perspective the Entity EJB is single threaded
    - Behaves like an object using the Active Object pattern (Lavender, Selic, others)
  - Are recoverable following a system crash
Persistence Management

- **Bean-managed persistence**
  - The entity EJB provides for its own persistence
  - The developer must put the database access code into the required EJB methods `ejbCreate()`, `ejbRemove()`, `ejbLoad()`, `ejbStore()`, and `ejbFind...()`
  - It can be deployed into any container without the container having to generate access call code

- **Container-managed persistence**
  - The entity EJB delegates persistence to the container
  - No persistence code required in the EJB methods by the developer
  - The container provider’s tools must generate the necessary functions at deployment time and implement them in the container
    - The developer must define all the fields requiring persistence in the deployment descriptor `containerManagedFields (1.0)` property or `transaction-type (1.1)`
The EJB specification defines two interfaces
- Remote Interface – the business methods
- Home Interface – provides a create method to create a new instance or find an existing one

A client never accesses the EJB instances directly
- Client uses the remote interface implemented by a remote object called EJBOBJ
- The EJBOBJ class *intercepts* method invocation and *delegates* to the bean instance
The EJB: Client View

- Interception allows container to provide services:
  - Lifecycle management
  - Transactions
  - Persistence

Auto Generated at deployment time
The home interface extends EJBHome and is used to:

- Locate,
- Create,
- Delete EJBs

A client creates an EJB by using the home interface’s create method(s) provided by the developer.

The EJB itself must have EJBCreate methods that correspond to the create method(s) in the home interface.
Remote Interface Definition

- The remote interface extends EJBOBJECT which acts as a mediator between the client, the container, and the EJB instance in the container
  - Maintenance methods:
    - getEJBHome() – obtains the EJB home interface
    - getPrimaryKey() – return unique identifier for the EJB
    - getHandle() – returns an abstract handle used to reestablish a reference to an EJBOBJECT
    - isIdentical() – tests whether a given EJBOBJECT is identical to the invoked EJBOBJECT
    - remove() – deletes the EJB instance and the EJBOBJECT

- All the logical “business” methods go in the remote interface
  - May be stereotyped with <<EJBRemoteMethod>>
Session EJB Example

EJBOBJECT (from ejb)

Remote

shopCart (Remote)

shopCart Home (Home)

sessionBean (from ejb)

shopCartEJB

+ shopCartEJB()
+ ejbActivate()
<<EJBCreateMethod>> + ejbCreate()
+ ejbPassivate()
+ ejbRemove()
+ setSessionContext()
<<EJBRemoteMethod>> + bixFunc()
Entity EJB Example

- **EJBHome** (from ejb)
- **EntityBean** (from ejb)
- **EJBOBJECT** (from ejb)

**CustomerHome**
- **EJBRemoteMethod** + findByPrimaryKey()
- **EJBFinderMethod** + create()

**Customer**
- **EJBRemoteMethod** + bizFunc()

**EntityContext** (from ejb)
+ sum : int
+ CustomerBean()  
  + CustomerBeanPK()  
  + context
+ ejbCreate()  
  + ejbPostCreate()  
  + ejbActivate()  
  + ejbPassivate()  
  + ejbLoad()  
  + ejbStore()  
  + ejbRemove()  
  + setEntityContext()  
  + unsetEntityContext()  
  + bizFunc()
What Are Servlets?

- A program that extends the functionality of a web server to deliver content to a client
  - Written in Java
  - Runs upon a request from a client
  - Dynamically generate the response
  - Send the response in an HTML or XML document to the client
  - Runs on the web server
  - Good at handling simple request/response operations
  - Extends the java.servlet.Servlet (generic or http) class
What Are Java Server Pages?

- An extension of the servlet API
- An HTML web page that contains additional bits of logic that execute to generate dynamic content
- Best used to separate application logic from presentation formatting
  - Can make a change to presentation code without recompiling application logic (like you would have to do if it were all in the servlet)
- Creates an interface to let Web Designers and Application Logic Designers work together efficiently
- Allows the creation of more adaptable systems
Servlets and JSP

- **Servlet**
  - Application logic
  - Accept and handle client requests
  - Orchestrate other server side resources

- **JSP**
  - Scripting complex formatting
    - Ex. Creating dynamically sized tables
  - Any presentation formatting
The Unified Modeling Language
Unified Modeling Language (UML)

- A graphical language for
  - Specifying
  - Visualizing
  - Constructing
  - And documenting
- The artifacts of software systems
- The de facto standard for visual modeling
  - www.omg.org/technology/uml/index.htm
UML Lineage

- Booch
- Rumbaugh
- Selic
- Odell
- Meyer
- Schlaer-Mellor
- Gamma et al.
- Jacobson
- Harel
- Fusion
- Embly

Structure Diagrams
Classification
Pre- and post-conditions
Object life cycles
Patterns, frameworks, notes
State charts
Operations descriptions, Message numbering
Singleton classes
UML Diagrams

- Activity Diagram
- Use Case Diagram
- Sequence Diagram
- Collaboration Diagram
- Class Diagram
- State Transition Diagram
- Component Diagram
- Deployment Diagram
UML Profiles

- Specialize or refine the UML for a specific purpose
- Does not add any basic concepts, but instead
  - Specializes existing concepts
  - Define conventions for applying them
- UML Profiles examples:
  - Business Modeling, Data Modeling, Real-time Modeling, XML DTD Modeling, Web Modeling, …
- Uses UML extension mechanisms
  - Stereotypes
  - Tagged Values
  - Constraints
Stereotypes

- Creates a virtual UML metamodel construct based on an existing UML metamodel construct
- A way to introduce new things that “speak” the vocabulary of a domain
- May be rendered three ways:

  - With its name enclosed in guillemets, <<stereotypename>>
    
    ```
    <<Web Client Page>>
    LoginPage
    ```

  - With the name and icon
    
    ![Login Page Icon](image)

  - As an icon with no stereotype name
    
    ![Login Page Icon](image)
Tagged Values

- An attribute or property for a metamodeling element
- Allows arbitrary information to be attached to an instance
- A set of tagged values can be associated with a stereotype
- Rendered as a tag name, an equal sign, and value enclosed by braces near the element it modifies
  - May also be attached in a note, or in a separate compartment
Constraint

- A semantic restriction represented as a text expression or more formally (ex. Object Constraint Language)
- Express restrictions and relationships that cannot be expressed using UML notation
- One or more constraints can be applied to any model element to govern the use of its instances
- Rendered as text enclosed in braces, attached to a dependency, or enclosed in a note

<table>
<thead>
<tr>
<th>Account</th>
<th>balance must not be negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>- balance : float</td>
<td></td>
</tr>
</tbody>
</table>
UML Profile for EJB

- Response to Java Specification Request (JSR) 26
  - JSR-000026 UML/EJB Mapping Specification
  - Endorsed by Sun, Rational, and IBM
- Currently in Sun’s Community Process Program
- Defines standard UML extensions that combine and/or refine existing UML constructs to create a dialect to describe EJB-based artifacts
- Current version supports UML 1.3 and EJB 1.1

java.sun.com/aboutJava/communityprocess/jsr/jsr_026.uml.html
EJB Stereotypes

- **Logical Model**
  - **Class Diagram**
    - **External View**
      - <<EJBHomeInterface>> class
      - <<EJBHomeMethod>> method
      - <<EJBCreateMethod>> method
      - <<EJBFinderMethod>> method
    - <<EJBRemoteInterface>> class
      - <EJBRemoteMethod>> method
    - **Internal View**
      - <<EJBSession>> class
      - <<EJBEntity>> class
      - <<EJBPrimaryKey>> class
  - **Internal View**

- **Physical Model**
  - **Component Diagram**
    - <<EJBDeploymentDescriptor>> component
    - <<EJB-JAR>> component
The Home and Remote interfaces will have an instantiate relationship with the EJB class

- <<EJBRealize>>

This will not generate an import statement
Rational Rose

- **Supports**
  - UML 1.3
  - EJB 1.1
  - UML Profile for EJB

- **Stereotypes**
  - Predefined for EJBs
  - Available in Class and Component diagrams

- **Tagged Values**
  - Captured in dialogs and used for code generation of classes, interfaces, and deployment descriptors

- **Constraints**
  - Implemented in code generator and reverse engineering
Rose EJB Support Highlights

- EJB Configuration dialogs for building EJBs
- Forward engineering of EJB 1.1 compliant EJB classes, interfaces, and XML deployment descriptor
- Java aware editor to compile EJBs
- Reverse engineering of jar files
  - Drag and drop jar files into Rose
  - EJB Profile stereotypes are added to modeling elements
  - Tagged value dialogs are filled in
- Jar file creation from Rose
Modeling Servlets in UML

- Modeled as ordinary Java classes
- Use stereotypes to distinguish between servlet types
  - <<Http_Servlet>>
  - <<Generic_Servlet>>
Modeling JavaServer Pages in UML

- How do you model something that’s a hybrid between
  - An HTML page
  - A servlet?

- Model as two logical entities!
  - A <<Server Page>> is a class stereotype that abstracts the web page’s behavior on the server.
  - A <<Client Page>> abstracts the behavior of a web page on the client
<<Server Page>> and <<Client Page>> classes in the logical view have a special stereotyped relationship: <<build>>
Rose 2001 Demo