Understanding and Designing with EJB

B. Ramamurthy

Based on j2eetutorial documentation.

http://java.sun.com/j2ee/tutorial/1_3-fcs/index.html
Review

- Request/Response Model
- Distributed Objects: stubs and skeleton providing location transparency
- Naming and Lookup: registry and binding
- Server-side technology: servlets (project1)
- Web applications: can be written entirely using Java Server Pages (static and dynamic content and data access can be provided); JSP is wrapper on servlet technology.
- Concept of initial context: The starting point for resolution of names for naming and directory operations.
- Data base access: using Java Data Base Connectivity
When to use EJB

- For large scale applications: where resources and data are distributed.
- When the application is run on servers at many locations.
- Where scalability is critical.
- Where transactions are required to ensure data integrity.
- When a variety of clients need to be handled.
Types of Enterprise Bean: Session

- **Session bean**: represents a single client inside the J2EE server. Session represents an interactive session. When a client terminates the session bean terminates/is no longer associated with the client.

- **Stateful session bean**: maintains a conversational state for the duration of a session. Ex: items reviewed in a session at some sites.

- **Stateless session bean**: does not maintain a conversational state. Ex: computing a formula for a given value.
Types of Enterprise Bean: Entity

- An entity bean represents a business object in a persistent storage mechanism. Ex: customers, orders, and products.
- Each entity bean typically has an underlying table in a relational database (business data), and each instance of the bean corresponds to a row in that table.
- Transactional and recoverable on a server crash.
Types of Enterprise Bean: Message-Driven

- A message driven bean is an enterprise bean that allows J2EE applications to process messages asynchronously.
- It acts as a JMS listener, which is similar to an event listener except that it receives messages instead of events.
- The messages can be sent by any J2EE component: an application client, another enterprise bean, or a web component, or a non-J2EE system using JMS.
- Retain no data or conversational state.
Contents of an Enterprise Bean

- **Interfaces**: The remote and home interface for remote access. Local and local home accesses for local access.
- **Enterprise bean class**: Implements the methods defined in the above interfaces.
- **Deployment descriptor**: An XML file that specifies information about the bean such as its type, transaction attributes, etc.
- **Helper classes**: non-bean classes needed by the enterprise bean class such as utility and exception classes.
## Naming Conventions

<table>
<thead>
<tr>
<th>Item</th>
<th>Syntax</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enterprise Bean Name (DD)</td>
<td>&lt;name&gt;eJb</td>
<td>AccountEJB</td>
</tr>
<tr>
<td>EJB JAR display name (DD)</td>
<td>&lt;name&gt;eJb</td>
<td>AccountJAR</td>
</tr>
<tr>
<td>Enterprise bean class</td>
<td>&lt;name&gt;Bean</td>
<td>AccountBean</td>
</tr>
<tr>
<td>Home interface</td>
<td>&lt;name&gt;Home</td>
<td>AccountHome</td>
</tr>
<tr>
<td>Remote interface</td>
<td>&lt;name&gt;</td>
<td>Account</td>
</tr>
<tr>
<td>Local home interface</td>
<td>Local&lt;name&gt;Home</td>
<td>LocalAccountHome</td>
</tr>
<tr>
<td>Local interface</td>
<td>Local&lt;name&gt;</td>
<td>LocalAccount</td>
</tr>
<tr>
<td>Abstract Schema (DD)</td>
<td>&lt;name&gt;</td>
<td>Account</td>
</tr>
</tbody>
</table>
The life cycles of enterprise beans

An enterprise bean goes through various stages during its lifetime. Each type has different life cycle.
Session bean

Does not Exist -> Ready

create, remove

Does not Exist -> Ready

create, remove

passivate, activate
Entity and Message-driven Bean Lifecycle

Does not Exist

create   remove

Ready

Does not Exist

setContext   unsetContext

Pooled

ejbActivate   ejbPassivate

create   remove

Ready

onMessage

3/31/2003
Designing an application

- Start with Remote interface methods.
- For completion write the Home interface.
- Implement these methods in a (session) bean class.
- Update build.xml “ant” file and compile using ant
- Use the deploy tool to deploy the application on your j2EE server and set up the configuration.
- Write a client (preferably a web client) to test your enterprise application.
- Lets go through converter application. Your assignment is to make it a more meaningful and useful converter.
MidTerm Review

- Web application design: n-tier design from word problem. Represent using block diagram, use case and class diagram. Stepwise explanation; project 1
- J2EE Application model: application model
- Enterprise beans: Session, entity and message-driven beans: characteristics and life cycle
- Enterprise integration: Ch.1-5 of your text
Session Bean

- Home interface
- Remote interface
- Bean class
- Deployment descriptor
- Can be stateless (ex: converter) or stateful (ex: cart)
- Stateless: container can maintain a pool of instances of session bean and reuse them for many client requests.
- Stateful: instance pooling can be done but what to with the state? Activate and passivate the instances by storing and restoring them from secondary memory.
Cart Stateful Session Bean

- One or more ejbCreate methods (void methods) providing flexibility of instantiating the bean with various starting states.
- Business methods: addBook, removeBook, getContents
- Data (state): string customerId, string CustomerName, Vector contents
Client calls + bean and container response

- Client gets naming context.
- Client looks up the remote object interface.
- Narrows it to IIOP object and then casts it to CartHome to set "home" variable.
- Creates bean:
  
  ```java
  Cart shoppingCart = home.create("Bina", "1234");
  ```
- EJB container instantiates the enterprise bean.
- EJB container then calls the appropriate ejbCreate method in CartBean. Observe ejbCreate verifies the ID using an IDVerifier object.
- After successful creation and initialization of the bean the client can invoke business methods as follows:
Cart Business Method calls

shoppingCart.addBook("J2EE in 21 days");
shoppingCart.removeBook("Java 2: inside information");
booklist = shoppingCart.getContents();

Now the client can print out the vector booklist.
Other Features

- Stored in the enterprise bean’s deployment descriptor, an environment entry is a name-value pair that allows you to customize the bean’s business logic without changing its source code.
- A enterprise bean that calculates discounts, may have an environment variable named *discount percent*.
- Before deploying the *bean’s application* one can use the deploy tool to assign the proper value for discount percent.
- Look at CheckerBean example.
Entity Bean

- Data is at the heart of most business applications.
- In J2EE applications, entity beans represent the business objects that need persistence (need to be stored in a database.)
- You have choice of bean-managed persistence (BMP) and container-managed persistence (CMP).
- In BMP you write the code for database access calls. This may be additional responsibility but it gives control to the bean developer.
Entity Bean with BMP: Example: SavingsAccount

- The state of the SavingsAccountEJB is stored in the savingsAccount table of a relational database.

- savingsAccount table could be created by a SQL statement as shown below:

```
CREATE TABLE savingsAccount
(id VARCHAR(3) CONSTRAINT pk_savingsaccount PRIMARY KEY,
 firstname VARCHAR(24),
 lastname VARCHAR(24),
 balance NUMERIC(10,2));
```
SavingsAccountEJB

This contains:
- Remote interface (SavingsAccount)
- Home interface (SavingsAccountHome)
- Entity bean class (SavingsAccountBean)
- Utility class: InsufficientBalanceException
- And a client to test it: SavingsAccountClient
Entity Bean class

- Implements EntityBean interface
- Zero or more ejbCreate and ejbPostCreate methods
- Finder methods
- Business methods
- Home methods
Entity Bean Methods

- `ejbCreate` inserts the entity state into the database; initializes the instance variables and returns the primary key.
- `ejbRemove` will delete the record corresponding to the bean from the database.
- `ejbLoad` and `ejbStore` methods synchronize instance variables of an entity bean with the corresponding values stored in a database. `ejbLoad` refreshes the instance variables from the db and `ejbStore` writes variables to the database. Container does this not the client.
- `ejbFinder` allows client to locate entity beans. Find the collection of records with “Smith” as author.
- Business methods and home methods.
### SQL statements in SavingsAccountBean

<table>
<thead>
<tr>
<th>Method</th>
<th>SQL Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>ejbCreate</td>
<td>INSERT</td>
</tr>
<tr>
<td>ejbFindPrimaryKey</td>
<td>SELECT</td>
</tr>
<tr>
<td>ejbFindLastName</td>
<td>SELECT</td>
</tr>
<tr>
<td>ejbFindInRange</td>
<td>SELECT</td>
</tr>
<tr>
<td>ejbLoad</td>
<td>SELECT</td>
</tr>
<tr>
<td>ejbRemove</td>
<td>DELETE</td>
</tr>
<tr>
<td>ejbStore</td>
<td>UPDATE</td>
</tr>
</tbody>
</table>