Module 6
EJB Component Model

Objectives

► Describe the role of EJB components in a Java EE Application
► Describe the EJB component model
► Identify the proper terminology to use when discussing EJB components and their elements
Role of EJB Components in a Java EE Application

- EJB components:
  - Are managed by the EJB container
  - Provide business and messaging functions
  - Are scalable, transactional, multi-user, and secure

EJB Component Types

- Enterprise JavaBeans Technology
  - Session
    - Stateless
  - Message-Driven
    - Stateful
EJB Timer Service

- Is new as of the Java EE platform 1.4
- Is a container-managed service
- Provides a callback service for modeling workflow-type business processes that depend on notifications that certain time-related events have occurred
- Can be configured to occur:
  - At a specific time
  - After a specific elapsed duration
  - At specific recurring intervals

Enterprise Application Tiers

Modern practice favors the division of enterprise components into tiers:

- The services tier, which consists mostly of session beans and message-driven beans
- The EIS-facing or object-relational mapping tier, which consists mostly of entity classes and their supporting classes
An EJB component has a life cycle that is controlled by the EJB container based on the functionality requested by the client and the operational requirements of the system.

Each type of EJB component has specific life-cycle operations and characteristics.

Understanding the life cycle of the various EJB component types might allow a component developer to optimize the operation of the bean.
EJB Method Types

- There are two basic categories of methods that are present in session EJB components:
  - Life-cycle or callback methods
  - Business methods

Analysis of the EJB Component Model

The EJB component model has the following characteristics:
- A container encapsulates each component.
- The container provides proxies that allow clients limited and controlled access to the EJB component.
- The proxies implement interfaces that are provided by the EJB component developer.
- Clients make method calls on these interfaces.
Role of the EJB Container

The EJB container provides the following functionality:

► Encapsulates access to external resources such as databases and legacy systems
► Manages the life cycles of instances of the EJB component’s implementation class
► Isolates the class that provides the implementation from its clients
► Provides timer services, and can invoke methods at certain times, which allows the EJB component to perform scheduled tasks
► Monitors, for message-driven beans, a message queue on behalf of the EJB component
You can provide an Session EJB component with the following three types of client access:

- Local
- Distributed (Remote)
- Web service-based
Elements of a EJB Component

- A Message driven EJB component has the following elements
  - Implementation class
- A distributable and locally accessible Session EJB component has the following elements:
  - Component interface
  - Implementation class

Calling EJB Components From Servlets

Servlets can freely call stateless and stateful session beans. However, the developer should observe the following guidelines in using servlets with EJB components:

<table>
<thead>
<tr>
<th>Guideline</th>
<th>Rationale</th>
</tr>
</thead>
<tbody>
<tr>
<td>Servlets should avoid calling entity classes (direct database access).</td>
<td>The narrowness of the transaction boundaries leads to inefficient synchronization.</td>
</tr>
<tr>
<td>Servlets should generally make a few complex method calls, rather than many simple method calls.</td>
<td>Reduce RMI overheads.</td>
</tr>
<tr>
<td>Stateless Session beans can be located at initialization time.</td>
<td>JNDI lookup as are expensive.</td>
</tr>
</tbody>
</table>
**Initializing a Reference to a Stateless Remote Session Bean from a Servlet**

```java
private BankMgr bankMgr = null;
public void init() {
    try {
        InitialContext ic = new InitialContext();
        bankMgr = (BankMgr)ic.lookup("java:comp/env/ejb/BankMgr");
    } catch (Exception e) {
        throw new ServletException(e);
    }
}
```

**Using Annotations to Obtain an EJB Reference in a Managed Component**

```java
@EJB private BankMgr bankMgr;
```

Access to a session bean from normal Java classes must be obtained with JNDI. Only managed components, such as a servlet and other EJBs can use this technique to obtain an EJB reference.

Servlets using dependency injection can cause a `ConcurrentAccessException` if concurrent access to a stateful session EJB occurs.
EJBs Before the Java EE 5 Platform

The following table lists the various names used for the EJB component elements for version 2.1 of the EJB specification.

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remote client view</td>
<td>Includes both the remote home interface and the remote component interface</td>
</tr>
<tr>
<td>Local client view</td>
<td>Includes both the local home interface and the local component interface</td>
</tr>
<tr>
<td>Home interface</td>
<td>Generic term for the factory interface for an EJB component regardless of it being local or remote</td>
</tr>
<tr>
<td>Local home interface</td>
<td>Extends the EJBImpLocalHome interface and provides a local home object</td>
</tr>
<tr>
<td>Remote home interface</td>
<td>Extends the EJBImpHome interface and provides a remote home object</td>
</tr>
</tbody>
</table>

EJB 2.1 Entity beans have been replaced with the Java Persistence API and non-EJB Entity classes.