Chapter 6 – Enterprise Java Beans

- Overview of the EJB Architecture and J2EE platform
- The new specification of Java EJB 2.1 was released by Sun Microsystems Inc. in 2002.
- The EJB technology is widely used for large scale distributed applications where the resources, data, and users are distributed. Such distributed applications usually require system scalability and transaction managements for data integrity.
- An EJB component is a reusable, WORA (Write Once Run Anywhere), portable, scalable, and compiled software component which can be deployed on any EJB servers such as Java 2 Platform Enterprise Edition (J2EE), JBoss, and WebLogic Enterprise environment.
- The Java EJB technology is part of J2EE which provides a set of APIs, and other system services. The EJB implementations concentrate on business logic.
- J2EE is a enterprise reference architecture for Java enterprise applications. J2EE not only supports EJB components but also supports other Web components such as JSP and Servlets.
Overview of the EJB Architecture and J2EE platform (cont.)

• The EJB architecture makes Web enterprise application development much easier because most of system level services such as transaction management, security management, and multithreading management are supported by the EJB container instead of applications themselves.

• The EJB architecture also manages the EJB component lifecycle from the creation to the termination including activation and deactivation of an EJB component.

• An EJB component is a server side component which provides services to remote Web clients or local and remote application clients.
Overview of the EJB Architecture and J2EE platform (cont.)

- Web clients access this application via a Web browser in the client tier; The services may be provided by Java Servlets or JSPs on Web servers in the Web tier; The Servlets or JSPs need to access services provided by EJB beans located on remote distributed application servers in the business tier; The business tier is supported by databases in the Enterprise Information System (EIS) data tier.

- The Web servers, application servers, and data servers may be all located in different locations connected by Internet. The EJB technology is suitable for developments of very large and complex distributed applications such as business to business (B2B).
Overview of the EJB Architecture and J2EE platform (cont.)

J2EE EJB architecture
EJB Container

- All EJB instances are running within the EJB container. The container is a runtime environment (set of classes generated by deployment) that controls an EJB component instance and provides all necessary management services for its whole lifetime. Below is a list of such services:
  - Transaction management: ensuring transaction properties of multiple distributed transaction executions.
  - Persistence management: ensuring a persistent state of an entity bean that is backed up by database.
  - Life cycle management: ensuring the EJB component state transitions in its life cycle.
  - Security management: authentication and authorization services, integrity, and encryption management.
EJB Container (cont.)

• All access requests to the EJB component and responses from the EJB component must get through the EJB container.

• The EJB container is a run time environment which isolates EJB component from direct access by its clients.

• The container will intercept the invocation from clients to ensure the persistence, properties of transaction, security of client operations on EJB.
EJB Container (cont.)

• The EJB container supports all services EJB components need and an EJB component needs the container to reach outside and to obtain necessary information from its context interface.

• The EJB container is in charge of generating an EJB home object, which helps to locate, create, and remove the EJB component object.

• The EJB context interface provided by the EJB container encapsulates relevant information of the container environment and initialization parameters.
EJB Components

- An enterprise bean is a distributed server component that lives in an EJB container and is accessed by remote clients over network via its remote interface or is accessed by other local enterprise beans on the same server via its local interface.
- The EJB component is a remotely executable component deployed on its server and it is a self-descriptive component specified by its Deployment Descriptor (DD) in a XML format.
- Each EJB component has a business logic interface that clients can run the business logic operations via this interface without knowing the detail implementation behind the interface.
EJB Components (cont.)

• We call such interface as a *remote or local interface*. An instance of an EJB component is created and managed by its factory named *home interface* on the EJB container.

• Every enterprise bean must have a home interface and a remote (local) interface. The EJB component can be configured at the deployment time by specifying its deployment descriptor.
EJB Components (cont.)

• The EJB classes behind home and remote (or local) interfaces are the implementations of these two interfaces.

• An EJB component is a black-box component. A client of an EJB component only knows what the component does but not how it does.

• A client makes a request to an EJB component with its deployed name by looking up at JNDI to get an Object Reference (OR) of this EJB component.
EJB Components (cont.)

• The client can then create an instance of this EJB component on the server according to the reference. Finally, the client invokes the business methods of this EJB instance.

• The EJB class may also locate and access other EJB beans at remote sites by using EJB context information.

• The message driven bean is an exception which does not have any interface like the home and remote interfaces that a classic EJB has.
The Client access to EJB on server

1. Lookup("EJB")
2. Create()
3. Invoke method
EJB Components (cont.)

- The EJB technology supports the following enterprise bean categories:
- **Session Bean**
  - Stateless session beans that implement various business logics, such as language translation, logon process, tax calculation, and currency conversion
  - Stateless session beans that is wrapped in a Web service
    Any existing enterprise bean can be encapsulated in an external web service by a WSDL document which describes the web service endpoint of the bean implementations. Such special bean does not provide interfaces that a regular EJB component provides.
  - Stateful session beans, which play the same roles as stateless session beans except they keep tracking the states of the conversation during a session. For instance, a shopping cart bean can be a typical stateful session bean.
- A session bean does not have its permanent state.
Entity Bean

• Bean Managed Persistence (BMP) entity beans, where persistent storage management (JDBC SQL) is coded by bean developers.

• Container Managed Persistence (CMP) entity beans, where the persistent storage management is specified by the deployment tool and managed by the container.

• An entity bean is backed up by a relational database.
Message-Driven Beans (MDB)

- MDB represents a new EJB component type that works in an asynchronous communication mode just like an event-driven delegation model in Java.

![EJB interfaces diagram]
Message-Driven Beans (MDB) (cont.)


- A local interface implements `javax.ejb.EJBLocalObject` interface and a local home interface implements `javax.ejb.EJBLocalHome` interface. The local interface is used by another EJB component running on the same server so that it can reduce the overhead caused by remote access.
Message-Driven Beans (MDB) (cont.)

• The remote interface provides the location independence but it is more expensive since it must provide stub and skeleton to support the remote communications.

• The local interface makes invocation much more efficient than remote. Another important difference between local and remote interfaces is that method invocation in local interface uses *passing by reference* and the method invocation in remote interface uses *passing by value* which needs serialization, i.e., marshaling and unmarshaling.
The EJB implementation class implements either sessionBean or entityBean interface, both of that implement EnterpriseBean interface.