Introduction to EJB

What is an EJB ?

An *enterprise java bean* is a server-side component that encapsulates the business logic of an application.

By invoking the exposed methods, remote clients can access the services provided by the application.

When to use EJB ?

- **The application must be scalable.** To accommodate a growing number of users, you may need to transparently distribute an application's components across multiple machines.

- **Transactions are required to ensure data integrity.** Enterprise beans support transactions, the mechanisms that manage the concurrent access of shared objects.

- **The application will have a variety of clients.** With just a few lines of code, remote clients can easily locate enterprise beans. These clients can be thin, various, and numerous.

What is J2EE?

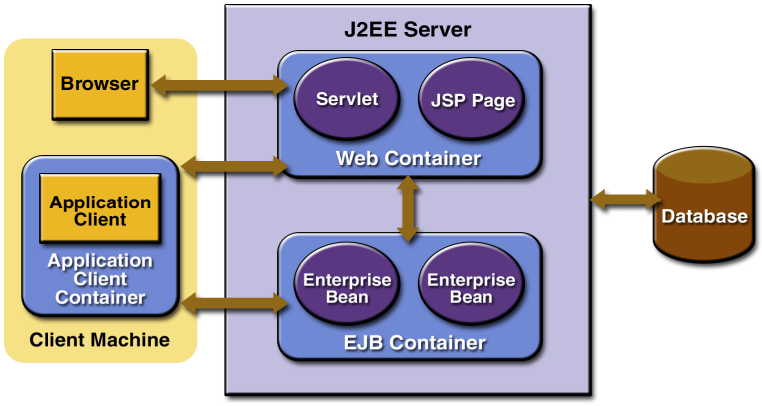
JAVA™ 2 PLATFORM ENTERPRISE EDITION

The Java™ 2 Platform, Enterprise Edition (J2EE™) is an integrated platform which allows to build solutions for multi-tier, enterprise applications.

It extends the cross-platform portability of Enterprise JavaBeans™ (EJB™) technology by providing a foundation of common services and APIs, including XML, JavaServer Pages™ technology, and Servlets.

The Java 2 SDK, Enterprise Edition (J2EE SDK) is the reference implementation for the J2EE™.

Architectural view



Types of EJB

Enterprise Bean Type	Purpose
Session	Performs a task for a client
Entity	Represents a business entity object that exists in persistent storage
Message-Driven	Acts as a listener for the Java Message Service API, processing messages asynchronously

A more detailed architectural view

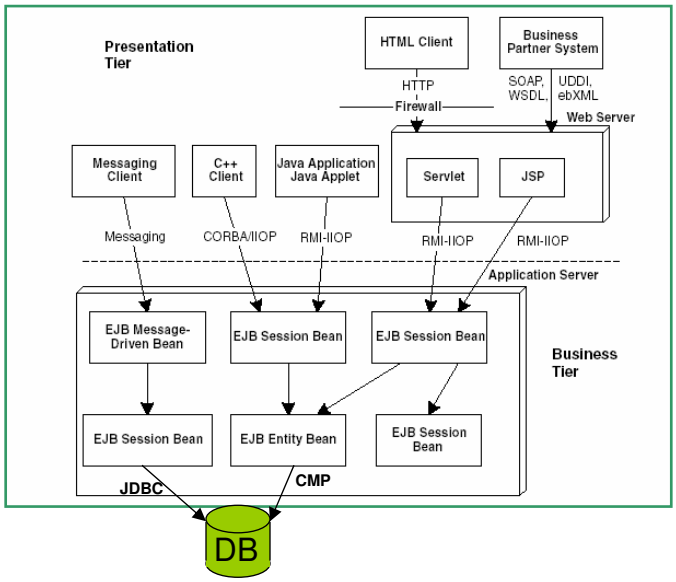


Image taken from "Mastering EJ2B"

Introduction to Entity Beans

EJB ingredients

Interfaces: The remote and home interfaces are required for remote access. For local access, the local and local home interfaces are required.

Enterprise bean class: Implements the methods defined in the interfaces.

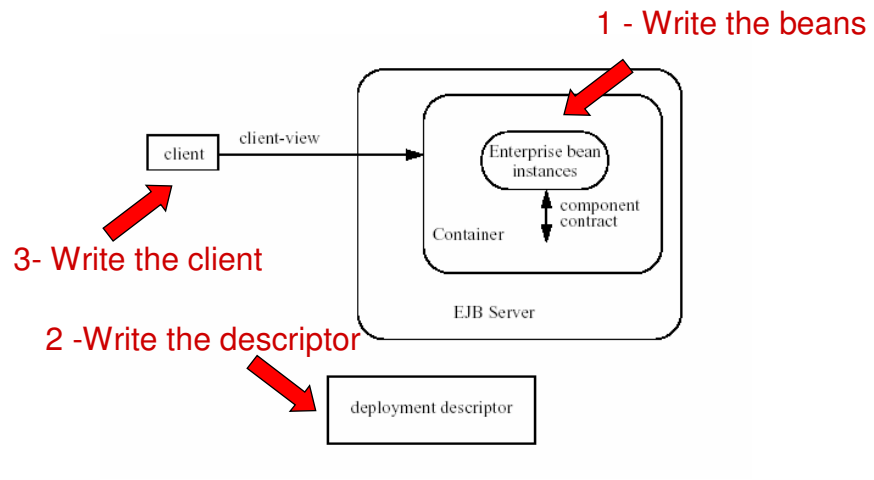
Helper classes: Other classes needed by the enterprise bean class, such as exception and utility classes.

EJB ingredients

Deployment descriptor: An XML file that specifies information about the bean such as its persistence type and transaction attributes.

- You package the files in the preceding list into an EJB JAR file, the module that stores the enterprise bean.
- To assemble a J2EE application, you package one or more modules--such as EJB JAR files--into an EAR file, the archive file that holds the application.

Architectural view



Entity Beans

- **An *entity bean* represents a business object in a persistent storage mechanism.** Some examples of business objects are customers, orders, and products.

  The bean represents a business entity, not a procedure. For example, **CreditCardEJB** would be an entity bean, but **CreditCardVerifierEJB** would be a session bean.

  The bean's state must be persistent. If the bean instance terminates or if the J2EE server is shut down or crashes, the bean's state still exists in persistent storage (a database).

Entity Beans: PERSISTENCE

- ***Persistence*** means that the entity bean's state exists beyond the lifetime of the application or the J2EE server process.

  There are two types of persistence for entity beans: bean-managed and container-managed.

  With *bean-managed persistence* (BMP), the entity bean code that you write contains the calls that access the database.

  If your bean has *container-managed persistence* (CMP), the EJB container automatically generates the necessary database access calls. The code that you write for the entity bean does not include these calls.

Entity Beans: SHARED ACCESS

Entity beans may be shared by multiple clients.

- Because the clients might want to change the same data, it's important that entity beans work within transactions.

  Typically, the EJB container provides transaction management. In this case, you specify the transaction attributes in the bean's deployment descriptor.

  You do not have to code the transaction boundaries in the bean--the container marks the boundaries for you.

Entity Beans: DB-like features

Like in a relational database:
- Each entity bean has a unique object identifier;
- An entity bean may be related to other entity beans.

The unique identifier, or *primary key*, enables the client to locate a particular entity bean.

You implement relationships differently for entity beans with BMP and those with CMP:
- **BMP:** the code that you write implements the relationships.
- **CMP:** the EJB container takes care of the relationships for you. (*container-managed relationships*).