Enterprise Java Beans (EJB)

MIE456 Tutorial

Agenda

- What is EJB
- How does EJB work?
- What are the benefits of using EJB

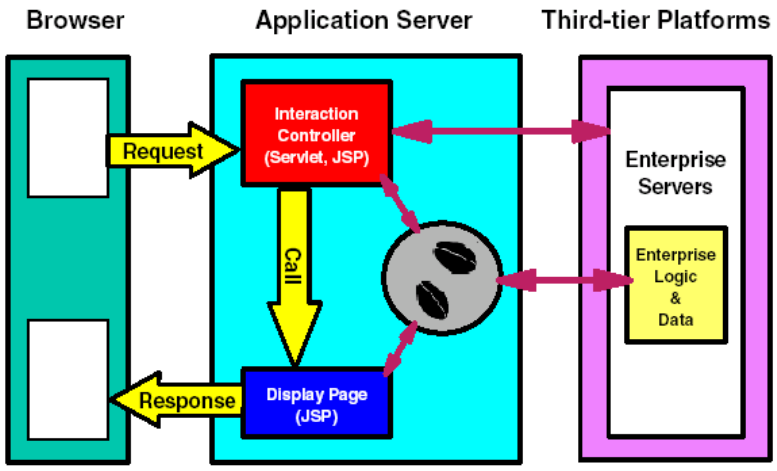
What is EJB?

- A component architecture
- A component model
- For developing object-oriented distributed enterprise-level applications.

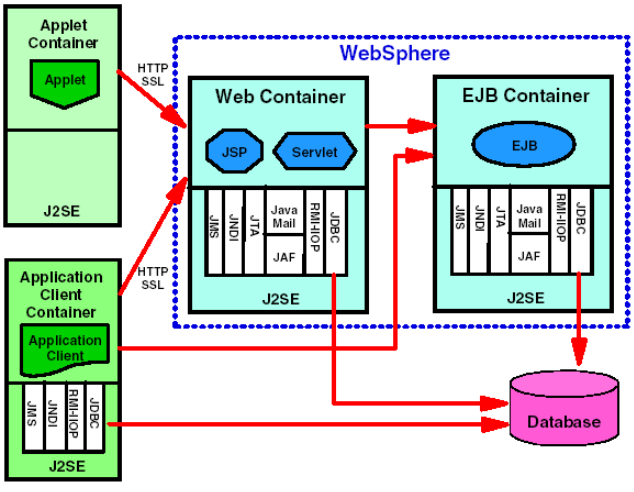
Why EJB?

- Common tasks of Enterprise Applications
  - Concurrency
  - Persistence
  - Transactions
  - Resource management
- The EJB/application server technologies can:
  - Take care of these common issues
  - Let developers focus on implementing the business logic

Architecture Overview



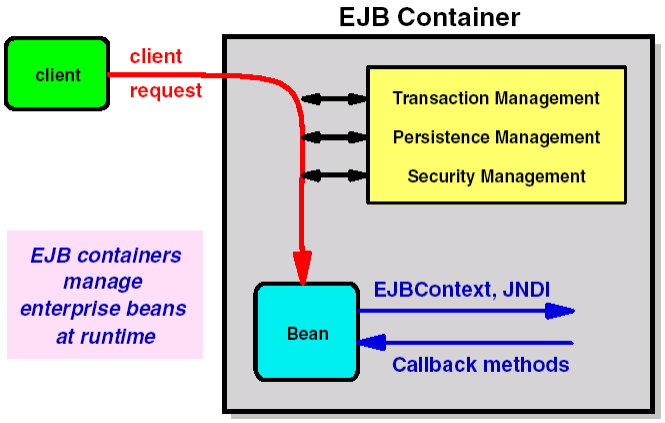
Architecture



Application Server and EJB Container

- Application Server
- Enterprise Java Server
- EJB Container
  - Insulates the EJBs from direct access from client applications.
  - Every time a bean is requested, created, or deleted, the container manages the whole process.

EJB Container



Interfaces of EJBs

- Home Interface
  - Defines the bean's life cycle methods
  - E.g. create, remove, and finding
- Remote interface
  - Defines the API of the methods of the EJB
  - The business logic method API

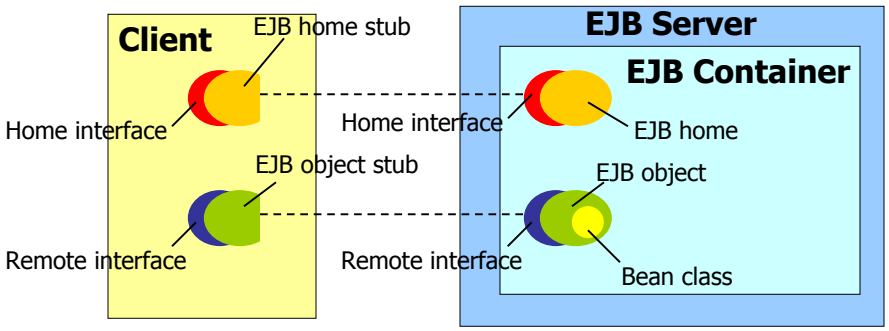
Classes of EJBs

- Bean class
  - Implements the bean's business methods
  - Does NOT implement home or remote interface
  - However, it must have methods that match the signatures of the remote interface
- Primary key class
  - For entity beans only,
  - more later…

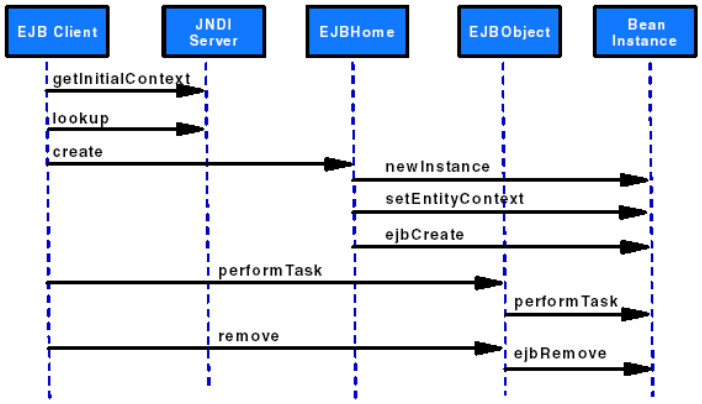
Deploying EJBs

- After the home interface, remote interface, bean implementation, and primary key class (if needed) are prepared (generated by tools or coded manually), the EJBs must be added to the EJB container.
- This process is called deployment.
- During deployment, many files are generated
  - Home stub, object stub, EJB home, EJB object, …

Deploying EJBs



Life Cycle of EJB Instance



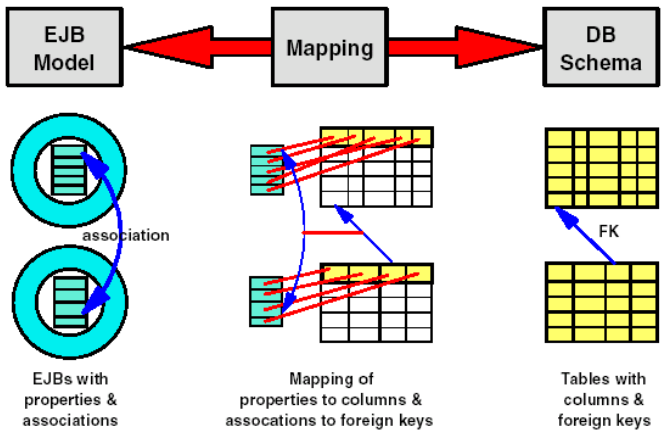
Types of EJBs

- Entity Beans
  - Container-managed persistence (CMP)
  - Bean-managed persistence (BMP)
- Session Beans
  - Stateful session Beans
  - Stateless session Beans

Entity Beans

- Represent permanent data
- Provide methods to manipulate data
- Usually, permanent data is stored in a data source, such as a relational or object database
- Each bean is identified by a primary key

Mapping Schema to Entity Beans



Container Managed Persistence (CMP)

- Delegate their persistence to their EJB container
- Do not have to know which source is used to provide the persistent state of the bean.
- You just have to specify which fields are persistent.
- All the required JDBC code for accessing the database is generated for you.
- Therefore, there is absolute portability and the EJB developer can focus on the business logic.

Bean-managed persistence (BMP)

- Entity beans manage their own persistence
- The EJB developer manages the persistent state of the bean by coding database calls
- Usually, the developer uses JDBC for coding the persistence logic

Comparing CMP and BMP

- A BMP entity bean is inappropriate for large applications.
- CMP is more scalable
- BMPs may provide better portability than CMPs, because less container-generated code is used.

Session Beans

- Encapsulates typical business processes
- May contain a conversational state associated with a particular client
- Unlike entity beans, states are not stored in a permanent data source and will not survive a server failure
- Session beans implement business logic, business rules, and workflow.

Stateful Session Beans

- Maintains client-specific session information (called conversational state) across multiple method calls and transactions
- Aware of client history
- Each stateful session bean has a *timeout* value
- The bean instance is destroyed and the remote reference is invalidated after the timeout period is elapsed.

Stateless Session Beans

- does not maintain any conversational state.
- Stateless session beans are pooled by their container to handle multiple requests from multiple clients.

Client Code

```
try {
     BankAccountHome acctHome;
     BankAccount acct, acct2;
     BankAccountKey acctKey;
     Checking acctChk;
     Savings acctSav;
     String acctID;
     Enumeration eAcct;
     javax.naming.InitialContext initialContext =
     new javax.naming.InitialContext();
     Object objHome = initialContext.lookup("itso/ejb35/bank/BankAccount");
     acctHome =
        (BankAccountHome)PortableRemoteObject.narrow(objHome,BankAccountHome.class);
     eAcct = acctHome.findAll();
     while (eAcct != null && eAcct.hasMoreElements()) {
          acct = (BankAccount)PortableRemoteObject.narrow
             (eAcct.nextElement(),BankAccount.class);
          acctKey = (BankAccountKey)acct.getPrimaryKey();
          acctID = acctKey.accID;
          acct2 = (BankAccount)acctHome.findByPrimaryKey(acctKey);
          if (acct2 instanceof Checking) {
               acctChk = (Checking)acct2;
               System.out.println("Checking "+acctID+": "+acct.getBalance() +"
                  "+acctChk.getOverdraft());
          } else if (acct2 instanceof Savings) {
               //...
          } else {
               //...
          }
     }
} catch(Exception ex) { ex.printStackTrace(); }
```

Benefits of using EJBs

- Independence from database schema
- Transaction management
- Platform independence
- Scalable environment
- Secure access
- Multi-tier architecture
- Code generation → easier development process