Enterprise Java Beans

Bina Ramamurthy

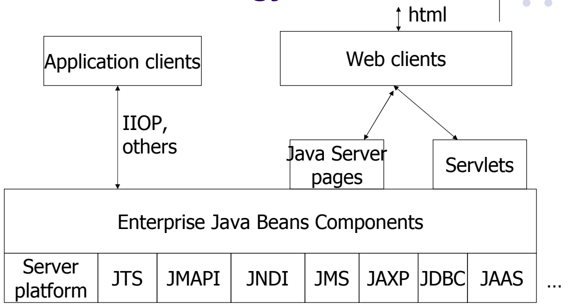
Introduction

- J2EE (Java2 Enterprise Edition) offers a suite of software specification to design, develop, assemble and deploy enterprise applications.
- It provides a distributed, component-based, loosely coupled, reliable and secure, platform independent and responsive application environment.

J2EE-based Application

- The J2EE APIs enable systems and applications through the following:
  - Unified application model across tiers with enterprise beans
  - Simplified response and request mechanism with JSP pages and servlets
  - Reliable security model with JAAS
  - XML-based data interchange integration with JAXP
  - Simplified interoperability with the J2EE Connector Architecture
  - Easy database connectivity with the JDBC API
  - Enterprise application integration with message-driven beans and JMS, JTA, and JNDI

J2EE Technology Architecture


html

Application clients

Web clients

IIOP, others

Java Server pages

Servlets

Enterprise Java Beans Components

Server platform	JTS	JMAPI	JNDI	JMS	JAXP	JDBC	JAAS	…

Enterprise Java Bean(EJB)

- An *enterprise bean* is a server-side component that contains the business logic of an application. At runtime, the application clients execute the business logic by invoking the enterprise bean's methods.
- Main goal of Enterprise Java Bean (EJB) architecture is to free the application developer from having to deal with the system level aspects of an application. This allows the bean developer to focus solely on the logic of the application.

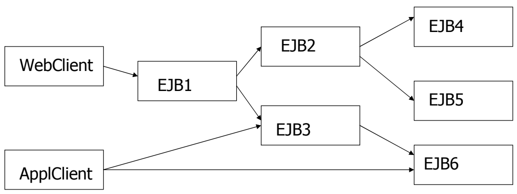
Roles in EJB Development

- Bean developer: Develops bean component.
- Application assembler: composes EJBs to form applications
- Deployer: deploys EJB applications within an operation environment.
- System administrator: Configures and administers the EJB computing and networking infrastructure.
- EJB Container Provider and EJB server provider: Vendors specializing in low level services such as transactions and application mgt.

Enterprise Java Bean (EJB)

- Deployable unit of code.
- At run-time, an enterprise bean resides in an EJB container.
- An EJB container provides the deployment environment and runtime environment for enterprise beans including services such as security, transaction, deployment, concurrency etc.
- Process of installing an EJB in a container is called EJB deployment.

Enterprise Application with many EJBs



WebClient

ApplClient

EJB1

EJB2

EJB3

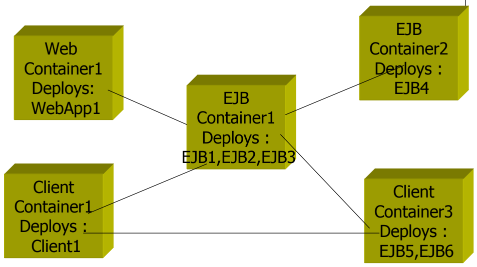
EJB4

EJB5

EJB6

Lets consider a shopping front application and figure out the possible components (EJBs)

Deployment with Multiple EJB Clients



Web Container1 Deploys: WebApp1

EJB Container1 Deploys : EJB1,EJB2,EJB3

EJB Container2 Deploys : EJB4

Client Container1 Deploys : Client1

Client Container3 Deploys : EJB5,EJB6

Business Entities, Processes and Rules

- EJB Applications organize business rules into components.
- Components typically represent a business entity or business process.
- Entity: is an object representing some information maintained in the enterprise. Has a “state” which may be persistent.
- Example: Customer, Order, Employee,
- Relationships are defined among the entities: dependencies.

Process

- Is an object that typically encapsulates an interaction of a user with business entities.
- A process typically updated and changes the state of the entities.
- A business process may have its own state which may exist only for the duration of the process; at the completion of the process the state ceases to exist.
- Process state may be transient or persistent.
- States ate transient for conversational processes and persistent for collaborative processes.

Rules

- Rules that apply to the state of an entity should be implemented in the component that represents the entity.
- Rules that apply to the processes should be implemented in the component that represents the processes.

EJB Types

- There are three types of EJBs:
  Entity, session and message-driven
- We will discuss message-driven bean in a separate discussion.
- The syntax of the session bean and entity bean client-view API is almost identical.
- But they have different life cycle, different persistence management, etc.
- EJBs can be stateless and stateful beans.

Life Cycle Differences

Session Bean	**Entity Bean**
• Object state:	
Maintained by container	Maintained by DB
• Object Sharing:	
No sharing: per client	Shared by multiple client
• State Externalization:	
State is inaccessible to other programs	Accessible to other programs
• Transactions:	
Not recoverable	State changed transactionally and is recoverable.
• Failure Recovery:	
Not guaranteed to survive failures	Survives failures and restored when the container restarts.

Choosing Entity or Session Bean

- Entity (business entity) is typically implemented as entity bean or a dependent object of an entity bean.
- Conversational (business) process as a session bean.
- Collaborative bean as an entity bean.
- Any process that requires persistence is implemented as an entity bean.
- When exposure to other applications are not needed for an entity or process (local/private process) then they are implemented as bean dependent objects.

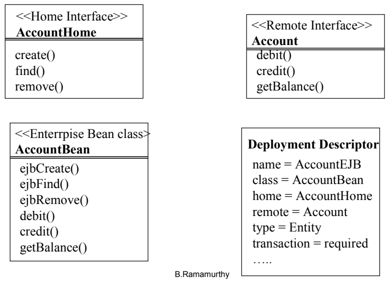
Parts of EJB

- EJB class that implements the business methods and life cycle methods; uses other helper classes and libraries to implement.
- Client-view API: consists of EJB home interface and remote interface.
  - Home interface: controls life cycle : create, remove, find methods
  - Remote interface: to invoke the EJB object methods

Parts of EJB (contd.)

- Deployment Descriptor: XML document for bean assembler and deployer;
  - A declaration about EJB environment needed for customizing the bean to the operating environment.
- Container Runtime services include: transactions, security,distribution,load balancing, multithreading, persistence, failure recovery, resource pooling, state management, clustering..

Enterprise Bean Parts



<<Home Interface>>
AccountHome
create()
find()
remove()

<<Remote Interface>>
Account
debit()
credit()
getBalance()

<<Enterrpise Bean class>
AccountBean
ejbCreate()
ejbFind()
ejbRemove()
debit()
credit()
getBalance()

Deployment Descriptor
name = AccountEJB
class = AccountBean
home = AccountHome
remote = Account
type = Entity
transaction = required
…..

AccountHome Interface

```
import java.rmi.RemoteException;
import javax.ejb.CreateException;
import javax.ejb.FinderException;
import java.util.Collection;

public interface AccountHome extends javax.ejb.EJBHome {
//create methods
Account create (String lastName, String firstName) throws RemoteException,
                              CreateException, BadNameException;
Account create (String lastName) throws RemoteException, CreateException;

// find methods
Account findByPrimaryKey (AccountKey primaryKey) throws RemoteException
                                    FinderException;
Collection findInActive(Date sinceWhen)
      throws RemoteException, FinderException, BadDateException;
```

Account Interface

```
import java.rmi.RemoteException;

public interface Account extends javax.ejb.EJBObject {

BigDecimal getBalance() throws RemoteException;

void credit(BiGDecimal amount) throws RemoteException;

Void debit(BigDecimal amount) throws RemoteException,
                                  InSufficientFundsException;
}
```

AccountBean class

```
public class AccountBean implements javax.ejb.EntityBean {
// life cycle methods from home interface
public AccountKey ejbCreate (String latName, String firstName) throws ... {
public AccountKey ejbCreate(String lastName) throws ...{...}
public AccountKey ejbFindByPrimaryKey(AccountKey primarykey)... {...}
Public Collection ejbFindInactive( Data sinecWhen).. {...}

// business methods from remote interface
public BigDecimal getBalance() {....}
public void credit(BigDecimal amt) {...}
Public void debit(BigDecimal amt) throws InsufficientFundException {....}

// container callbacks from EntityBean container
public ejbRemove( ) throws RemoveException{ ...}
public void setEntityContext(EntityContext ec) {...}
public unsetEntityContext(EntityContext ec) {...}
public void ejbActivate() {...}
public void ejbLoad() {....}
public void  ejbStore() {....}
}
```

Deployment Descriptor

```
...
<entity-bean>
   <ejb-name>AccountEJB</ejb-name>
   <home>com.wombat.AccopuntHome</home>
   <remote>com.wombat.Account</remote>
   <ejb-class>com.wombat.AccountBean</ejb-class>
   <persistence-type>Bean<\persistence-type>
   <primary-key-class>com.wombat.AccountKey</primary-key-class>
</entity-bean>

...
<container-transaction>
   <method>
     <ejb-name>AcoountEJB</ejb-name>
     <method-name>*</method-name>
   </method>
   <trans-attribute>required</trans-attribute>
</container-transaction>
```

Compilation and Deployment

- Compilation (building the executables) uses build tool such as Apache Ant.
- The components of the various tiers are packaged: .jar, .war, .ear
- Declarative resources are added.
- A deploy tool or management tool is used to deploy the packaged units into a J2EE server (container).

Summary

- J2EE environment provides a framework for bundling together the components into an application and provide the applications necessary common services such as persistence, security, mail, naming and directory service etc.
- Next class we will look a complete running example.
- Browse through:
  - http://java.sun.com/j2ee/faq.html
  - http://java.sun.com/blueprints/guidelines/designing_enterprise_applications_2e/index.html#chapters
  - http://java.sun.com/developer/onlineTraining/J2EE/Intro2/j2ee.html