Business-Driven Software Engineering (13.Vorlesung)
Pre EJB 3.0 Enterprise JavaBeans
Thomas Gschwind <thg at zurich.ibm.com>
Agenda

- Session Beans
- Message-Driven Beans
- Entity Beans
  - CMP Entity Beans
  - BMP Entity Beans
EJB 2.x vs. EJB 3.0

- Compatibility
  - Every EJB 3.0 server must also support the EJB 2.1 specification

- Mixing clients and servers
  - EJB 2.1 Client → EJB 2.1 Bean
  - EJB 2.1 Client → EJB 3.0 Bean
  - EJB 3.0 Client → EJB 2.1 Bean
  - EJB 3.0 Client → EJB 3.0 Bean

- Conceptual difference
  - No big difference, EJB 3.0 provides many simplifications
  - EJB 3.0 relies on JDK 1.5 to provide these simplifications
  - Many of the EJB 3.0 simplifications already provided for EJB 2.x by the xdoclet project (uses JavaDoc instead of code annotations)
Constituents of a 2.x EJB

- EJB Implementation Class
- Remote Interface
- Local Interface
- Home Object
- Local Home Interface
- Deployment Descriptor
- EJB Object (container generated)

- Similar to 3.0
- Almost same
- Almost same
- Missing in 3.0
- Missing in 3.0
- Typically more elaborate
- Similar to 3.0
The Bean Implementation

- EJB Specification defines standard interfaces to be implemented
  - Define methods for the bean’s management
  - In EJB 1.x and 2.x these interfaces must be implemented by the bean
  - In EJB 3.0, implementation of these interfaces is optional
EJB 2.x Session Bean

- Must implement the ...SessionBean interface
- Implementation of bean’s business logic
- Implements callbacks necessary for the EJB container
- Defines methods to allow the container to manage the bean
  - Initialization
  - Destruction
  - Passivation
  - Activation
Remote Interface

- Exports the business methods that may be invoked by clients

- Two flavors
  - Remote interface may be invoked by local and remote clients
  - Local interface may be invoked by local clients only
The Home Interface

- Each Home Object exposes a home interface
- Home interface must extend ….EJBHome
- Specifies methods for creating, locating, removing beans
- Must be remotely accessible
- Must adhere to all RMI-IIOP conventions

- In EJB 1.x and 2.x this is mandatory
- Optional for EJB 3.0 session and message-driven beans (use the same creation and removal)
- Java Persistence API Entities use a different API
The Home Object

- Clients deal only with EJBObjects
- EJBObjects cannot be instantiated by clients directly
- Home object is responsible for
  - Creating
  - Finding, and
  - Removing EJB objects
- Home objects are specific to a given bean
Home Interface & Home Object

1: Create bean

Client

2: Create EJB Object

Home Interface

Home Object

EJB Object

3: Initialize

Remote Interface

4: Return EJB Object ref.

Enterprise Bean

EJB Container/Server

Responsibility of Container
EJB 2.1 Bean Implementation Caveats

- Must not list home & remote interface in the class‘s implements clause
- Otherwise implementation object might be accidentally passed to clients
- Clients must only interact through Home & EJB objects
The SessionContext

- Container has already reference to bean
- SessionContext allows bean to interact with container
  - Retrieve home interfaces
  - Get and set transaction attributes
  - Obtain security attributes
- The setSessionContext method
  - Associates Bean with SessionContext
  - Typically, reference is stored in a local variable
The SessionContext

```java
public interface javax.ejb.SessionContext extends javax.ejb.EJBContext {
    public javax.ejb.EJBLocalObject getEJBLocalObject();
    public javax.ejb.EJBObject getEJBObject();
}
```

- EJBContext provides methods to obtain
  - The home reference
  - Transaction attributes
  - Security information
Cookie Server Object Model

- Java Distribution
  - EJB Distribution
    - <<interface>> .....Remote
    - <<interface>> .....EJBHome
    - <<interface>> .....EJBObject
    - <<interface>> .....EnterpriseBean
    - <<interface>> .....SessionBean
    - <<interface>> .....Serializable
    - <<interface>> .....Remote
    - <<interface>> .....CookieServerBean

- EJB Server Generated
  - Cookie Home Object
  - Cookie EJB Object
  - That's us
Implementation (cont’d)

```java
import javax.ejb.SessionBean;
import javax.ejb.SessionContext;
public class CookieServerBean implements SessionBean {
    private SessionContext ctx;
    public void setSessionContext(SessionContext ctx) {
        this.ctx = ctx;
    }

    public void ejbCreate() {
        System.err.println("ejbCreate()");
    }

    public void ejbRemove() {
        System.err.println("ejbRemove()");
    }

    public void ejbActivate() {
        System.err.println("ejbActivate()");
    }

    public void ejbPassivate() {
        System.err.println("ejbPassivate()");
    }

    public String getCookie() {
        return "Beam me up, Scotty, there is no intelligent life...";
    }
}
```
Remote Interface

- Provides the business methods (functionality) of the bean
- Similar to RMI Remote interface

```java
import java.rmi.RemoteException;
import javax.ejb.EJBObject;

public interface CookieServer extends EJBObject {
    public String getCookie() throws RemoteException;
    // inherits methods to remove and compare EJB objects
}
```
Home Interface

- Creation and removal of beans
- New compared to Java RMI

```java
import java.rmi.RemoteException;
import javax.ejb.EJBHome;
import javax.ejb.CreateException;

public interface CookieServerHome extends EJBHome {
    CookieServer create() throws RemoteException, CreateException;
    // remove method inherited from EJBHome
}
```
<!DOCTYPE ejb-jar PUBLIC
  "-//Sun Microsystems, Inc.//DTD Enterprise JavaBeans 2.0//EN"
  "http://java.sun.com/dtd.ejb-jar_2_0.dtd">
ejb-jar
  <enterprise-beans>
    <session>
      <ejb-name>CookieServer</ejb-name>
      <home>ch.unizh.bdse.CookieServerHome</home>
      <remote>ch.unizh.bdse.CookieServer</remote>
      <ejb-class>ch.unizh.bdse.CookieServerBean</ejb-class>
      <session-type>Stateless</session-type>
      <transaction-type>Container</transaction-type>
    </session>
  </enterprise-beans>
ejb-client-jar>CookieClient.jar</ejb-client-jar>
ejb-jar>
The Cookie Client

- Get a reference to the JNDI interface
  - Possibly specify system properties where to locate the repository
- Look up the bean’s home object
- Use the home object to create an EJB object
- Call business methods on EJB object
- Remove the EJB object
The Cookie Client (cont’d)

```java
import java.rmi.*; import java.util.*; import javax.naming.*;
public class CookieClient {
    public static void main(String[] args) throws Exception {
        Properties props=System.getProperties();
        Context ctx=new InitialContext(props);
        Object obj=ctx.lookup("CookieHome");
        CookieServerHome h=(CookieServerHome)
        PortableRemoteObject.narrow(obj,CookieServerHome.class);
        CookieServer s=h.create();
        System.out.println(s.getCookie());
        s.remove();
    }
}
```
Deployment & Use

EJB Container/Server

- **Home Interface**
- **Enterprise Bean**
- **Remote Interface**

- **Home Object**
- **EJB Object**
- **Client**

1. create bean
2. use bean

Supplied by Bean Developer

Generated by Container

IBM Research – Zurich
**Pre EJB 3.0 Bean Interaction**

1. Retrieve Home Object reference
2. Request EJB object creation
3. Create new EJB object
4. Return EJB object
5. Invoke business method
6. Invoke container
7. Delegate request to bean
8. Invoke container
9. Return result
Agenda

- Session Beans
- Message-Driven Beans
- Entity Beans
  - CMP Entity Beans
  - BMP Entity Beans
EJB 2.x MDB Implementation

- **Must implement**
  - `javax.ejb.MessageDrivenBean`
  - `Javax.jms.MessageListener`

- **Methods to be implemented**
  - `void ejbCreate();`
  - `void ejbRemove();`
  - `void onMessage(Message message);`
  - `void setMessageDrivenContext(MessageDrivenContext ctx);`
public class LogBean
implements MessageDrivenBean, MessageListener {
    private MessageDrivenContext ctx;
    public void setMessageDrivenContext(... ctx) {
        this.ctx=ctx;
    }
    public void ejbCreate() { }
    public void ejbRemove() { }
    public void onMessage(Message msg) {
        if(msg instanceof TextMessage) {
            TextMessage tm=(TextMessage)msg;
            try {
                System.err.println("received "+tm.getText());
            } catch(JMSException e) {}
        }
    }
}
Agenda

- Session Beans
- Message-Driven Beans
- Entity Beans
  - CMP Entity Beans
  - BMP Entity Beans
Entity Beans

- Mainly used in EJB 1.x and 2.x systems
  - EJB 1.x Entity Beans cumbersome, slow (lack of local interfaces)
  - EJB 2.x Entity Beans “only” cumbersome to use

- Tips:
  - Use coarse grained objects
  - Optimize data access
  - Use Java Persistence API instead
Constituents of an EJB (pre 3.0)

- EJB Implementation Class
- Business methods
  - Remote Interface
  - Local Interface
- Deployment Descriptor
- EJB Object (container generated)

- Lifecycle methods (optional in EJB 3.0)
  - Home Interface
  - Local Home Interface
- Home Object (container generated)
Object Relational Mapping

- More sophisticated than serialization
- Object-Relational Mapping
  - Store objects in a database
  - Retrieve data from db when object is created
- Differentiate
  - Bean data
  - Bean instance

<table>
<thead>
<tr>
<th>accountID</th>
<th>owner</th>
<th>balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>John Doe</td>
<td>-104.67</td>
</tr>
<tr>
<td>2</td>
<td>Mark Smith</td>
<td>985.00</td>
</tr>
<tr>
<td>3</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Account Class
int accountID
string owner
double balance

Account Data
accountID=1
owner="John Doe"
Balance=-104.67
Entity Beans

- Provide a view into a database (beans know how to store themself)
- Bean update ⇔ database update
- Synchronize bean data
  - ejbLoad() reads data from persistent storage
  - ejbStore() saves data to underlying storage
- These methods are called by the container when necessary
- Bean data may be modified externally
Requirements

- Have a home and a remote interface
- Entity bean class maps to a database representation
- Must implement some standard callbacks
- Provide a primary key class (must be serializable)
Pooling of Entity Beans

- Creating & destroying beans for each request is inefficient
- An entity bean defines only the kind of data to be stored
  - E.g., account#, account holder, balance
  - Entity bean can represent any account

=> Containers reuse beans
Reusing Beans

- Simply load the new data
- Problem
  - Bean may hold references to local resources
- Solution
  - Use callbacks to activate/deactivate bean
  - ejbActivate() ... bean is taken out of pool
  - ejbPassivate() ... bean is returned to pool
Activation & Passivation

1. ejbStore()
2. ejbPassivate()

Passivation entails a state save.

1. ejbActivate()
2. ejbLoad()

Activation entails a state load.
Lifecycle of an Entity Bean
Entity Bean Creation & Removal

- Creation of entity bean should construct a row in a database
Entity Bean Lookup

- Entity beans may be looked up
- Accomplished using finder methods
  - Listed in Home interface
  - Return primary keys of matching beans
Entity Contexts

- Provides additional information to EJB Context
- Set using setEntityContext() and unsetEntityContext()
- `getEJBLocalObject()`, `getEJBObject()`
  - EJB way of saying this
  - Remember, this may not be passed to clients
- `getPrimaryKey`
  - Primary key currently associated with this instance
  - Necessary for activation (after bean was passivated)
  - Useful for `ejbLoad()` and `ejbRemove()`
CMP Entity Beans

- Defined independently of the containers data representation
  - Serialization
  - Database
  - Object Database

=> No persistence logic must be present in bean
=> Developer does not declare fields
=> Persistence logic generated by the container
Accessing the Data

- How do we access the data if not declared by the class?
- Declare abstract getter & setter methods supplied by the container

```java
public abstract class Account implements EntityBean {
    // no fields
    //abstract getter & setter methods
    public abstract long getAccountId();
    public abstract void setAccountId(long id);
    public abstract String getOwner();
    // ...
}
```
Creating a CMP Entity Bean

- `ejbCreate(...)`
  For CMP Entity Beans database row is created by container. Just initialize the data using the set* methods.

- `ejbPostCreate(...)`
  Is called after the container has associated the bean with an EJB object. Complete initialization of the bean.
Looking up CMP Entity Beans

- Specify finder methods in Home interface
- Corresponding query is to be defined in the deployment descriptor
- Query is defined using EJB-QL
- Example:
  - public Collection findBigAccounts(int min);
  - SELECT object(a) FROM Account AS a WHERE a.balance>?1
Agenda

- Session Beans
- Message-Driven Beans
- Entity Beans
  - CMP Entity Beans
  - BMP Entity Beans
BMP Entity Beans

- Bean is responsible for persisting its state
  - Necessary if CMP mechanisms too simple
- Implement callbacks to save and load data
- Callbacks invoked by the container when necessary
  - ejbLoad()
  - ejbStore()
BMP Entity Bean Object Model

- BMP Entity Bean
  - ▶️ Interface
    - Serializable
  - ▶️ Interface
    - EnterpriseBean
  - ▶️ Interface
    - EntityBean

Java Distribution

EJB Distribution

That's us
Creating a BMP Entity Bean

- **ejbCreate(...)**
  For BMP Entity Beans database needs to be created by bean.

- **ejbPostCreate(...)**
  Is called after the container has associated the bean with an EJB object. May pass EJB reference to other beans.
Looking up BMP Entity Beans

- Entity beans may be looked up
- Accomplished using finder methods
  - Listed in Home interface
  - Return primary keys of matching beans
- Query is to be implemented as part of the bean
The BMP Cookie Entity Bean

- Manage a database of fortune cookies

- Requirements
  - Create cookies
  - Count how often various cookies have been returned
  - Look up cookies
Bean Managed State Fields

public class CookieBean implements EntityBean {
    private EntityContext ctx;
    private int cookieID;
    private String cookie;
    private int stat;

    public CookieBean() {}

    public int getCookieID() { return cookieID; }
    public void setCookieID(int id) { cookieID=id; }

    public String getCookie() { return cookie; ++stat; }
    public void setCookie(String c) { cookie=c; }

    public int getStat() { return stat; }

    ... // other business methods
}
public class CookieBean implements EntityBean {

    public Connection getConnection() throws Exception {
        Context ctx=new InitialContext();
        javax.sql.DataSource ds=(javax.sql.DataSource)
            ctx.lookup("java:comp/env/jdbc/ejbPool");
        return ds.getConnection();
    }

    public CookiePK ejbCreate(int cookieID, String cookie)
        throws CreateException {
        this.cookieID=cookieID; this.cookie=cookie; stat=0;
        try {
            Connection c=getConnection();
            ...
        }
    }
Creation (cont’d)

... PreparedStatement stmt=c.prepareStatement("insert into cookies(cookieID,cookie,stat) " + "values(?,?,?)");
stmt.setInt(1,cookieID);
stmt.setString(2,cookie);
stmt.setInt(3,stat);
stmt.executeUpdate();
return new CookiePK(cookieID);
} catch(Exception e) {
throw new CreateException(e.getMessage());
} finally {
try { if(stmt!=null) stmt.close(); } catch(…) {}
try { if(c!=null) c.close(); } catch(…) {}
}
...
Other Methods

- These methods must be implemented similar to `ejbCreate()`
  - `ejbRemove()`
  - `ejbLoad()`
  - `ejbStore()`
  - `ejbFindByPrimaryKey()`

- Others (e.g., `ejbPostCreate()`) may be left blank
Summary

- Session Beans
- Message-Driven Beans
- Entity Beans
  - CMP Entity Beans
  - BMP Entity Beans
Review Questions

- Explain the difference between EJB2.0 and EJB3.0
  - Session Beans
  - Message Driven Beans
  - Entity Beans and JPA Entities

- What is the difference between EJB2.0 container managed and bean managed entity beans
Outlook

- Repetitorium (practical part)
- Lab submission