Layers and Communications

Presentation Layer

- HTML Client
- XML Client
- WAP Client
- Web Server
- Java Client
- Windows Client

Business Logic Layer

- Application 1
- Application 2
- Application 3

Data Layer

- Relational Database
- External System (Host Application)
- Object-oriented Database
J2EE / Application Server

- provide necessary infrastructure for e-business processes
- easier and faster development of multitier enterprise applications
- highly available, scalable, reliable and secure enterprise applications (24x7)
- standard programming model
- standard platform for hosting J2EE applications
<table>
<thead>
<tr>
<th>J2EE Technologies</th>
</tr>
</thead>
<tbody>
<tr>
<td>• EJB – Enterprise Java Beans</td>
</tr>
<tr>
<td>• CORBA – Common Object Request Broker Architecture</td>
</tr>
<tr>
<td>• RMI – Remote Method Invocation</td>
</tr>
<tr>
<td>• JDBC – Java Database Connectivity</td>
</tr>
<tr>
<td>• J2EE Connector</td>
</tr>
<tr>
<td>• Transactions</td>
</tr>
<tr>
<td>• JMS – Java Messaging Service</td>
</tr>
<tr>
<td>• JavaMail</td>
</tr>
<tr>
<td>• Servlets</td>
</tr>
<tr>
<td>• JSP – Java Server Pages</td>
</tr>
<tr>
<td>• JNDI – Java Naming and Directory Interface</td>
</tr>
<tr>
<td>• XML – eXtensible Markup Language</td>
</tr>
<tr>
<td>• XSLT – XML Stylesheet / Transformation (J2EE 1.3)</td>
</tr>
</tbody>
</table>
EJB – Enterprise Java Beans
EJB – Enterprise Java Beans

- server side component model
- simplify development of transactional, scalable, portable and configurable distributed components
- provide complex middleware enterprise features (security, transactions, caching, pooling, ...)
- cross platform and cross vendor
Object Distribution

- Communication between objects which run in same process, on different processes or different machines
- Required network code is hidden from developer
- Based on RPC (remote procedure call): procedural invocation from a process on one machine to a process on another machine
- EJB uses RMI or RMI over IIOP (Internet Inter-Orb Protocol) protocol for communication
- IDL (Interface Definition Language) is used for a language independent description of distributed objects (CORBA)
Object Distribution:
Marshalling / Unmarshalling

- Used to transform parameters, return values and object references into a machine independent bytearray. (e.g. different byte representation between two machines)
Object Distribution:
Stubs and Skeletons
Object Distribution:
Naming Objects

- TCP/IP uses nameserver for resolving the correct IP out of a name (e.g.: www.javasoft.com \[192.18.97.137\])
- Distributed objects are also accessible with a name. Your objects will be registered at a nameserver.
- EJB uses the JNDI API to access a nameserver (e.g.: LDAP).
- How to get a reference to a distributed object at a client?
  - First you must know the TCP/IP address of the nameserver. (broadcast possible)
  - You must also know the registered name of the object
  - Ask the nameserver to resolve the name – returns a reference to the registered object.
EJB Applicationserver

• Divided into two parts
  – EJB Container
    • provides runtime environment for your beans
    • there may be many beans in one container
    • manages the beans running within them
    • container interacts with bean
  – EJB Server
    • Provides runtime environment for containers
    • Manages low level system resources which are needed by the containers
EJB Server & Container

EJB Server

EJB Container

Enterprise Bean

Enterprise Bean

Enterprise Bean

Enterprise Bean
Server & Container
Responsibilities

- Remote accessibility
- Persistency
- Transactions
- Multiclient support
- Security
- Resource management and bean life cycle
- Location transparency
The Parts of an EJB

- Remote Interface
  - Defines all methods a client may call on your implementation
- Enterprise Bean Class
  - Your implementation of the methods defined in the remote interface
- EJB Object
  - The „network-able“ implementation of your bean
- Home Interface
  - Defines methods to create, find and remove your beans
- Home Object
  - The „network-able“ implementation of the home interface
EJB Parts at a glance

1: create a new EJB object
2: Create EJB object
3: assign instance
4: return reference to EJB Object

Client

Home Object

EJB Object

Instance of Enterprise Bean Class

EJB Container/Server
Hollywood Principle
It’s a Framework not an API
Don’t call hollywood (the framework)!
Hollywood (the framework) calls you!

Diagram:
- Container/EJB Object
- Your Bean Implementation

Arrows indicate the direction of interaction.
Types of Beans

• Session – Model business processes
  – Stateless
  – Stateful

• Entity – Model persistent data
  – BMP (Bean Managed Persistency)
  – CMP (Container Managed Persistency)

• Message Driven – Model asynchronous processes
  – New in EJB 2.0
EJB – Session Beans
Session Beans

- represent business processes
- short-lived component
  - as long as the client "session" – client holds a reference
  - not persistent
  - typically do not survive server crashes
- Bean implementation must provide management methods which the container calls (javax.ejb.SessionBean interface)
- Methods automatically are serialized – it’s not possible for two client to access the same session bean at the same time
Stateful – Stateless

• You define the type of the session bean in the deployment phase – not in the code!
• Stateless – Nonconversational
  – does not remember data between remote calls \( \bullet \) attributes makes no sense
  – same instance may be used by several clients (but not at the same time)
  – 2 method calls can be handled by two different instances!
  – Instances will be pooled for performance issues
• Stateful – hold conversational state
  – Bean belongs to **one** client - no possibility to share a stateful session bean between two clients
  – remembers data between method calls
Method call in Stateless Session Beans

1: invoke a method

Remote Interface

2: get a bean instance out of pool

3: call method

4: return result
Method call in
Stateful Session Beans

1: invoke a method

Client1

EJB Object

Remote Interface

Instance bound to client

1: invoke a method

Client2

EJB Object

Remote Interface

Instance bound to client

EJB Container/Server

1: invoke a method

2: call method

3: return result

2: call method

3: return result
Session Bean Sample - Interfaces

**Home Interface:**

```java
public interface CalculatorHome extends EJBHome {
    public Calculator create()
        throws CreateException, RemoteException;
}
```

**Remote Interface:**

```java
public interface Calculator extends EJBObject {
    public int add(int first, int second)
        throws RemoteException;
    public int sub(int first, int second)
        throws RemoteException;
}
```
Bean Implementation:

```java
public class CalculatorBean implements SessionBean {
    public void ejbCreate() {
    }
    public int add(int first, int second) {
        return first+second;
    }
    public int sub(int first, int second) {
        return first-second;
    }
    ....
}
```
**Session Bean Client - Implementation**

*Client Implementation:*

```java
InitialContext ctx = new InitialContext();

Object obj = ctx.lookup("ejbsamples.Calculator");

CalculatorHome calcHome;
calcHome = (CalculatorHome) PortableRemoteObject.narrow(obj, CalculatorHome.class);

Calculator calc = calcHome.create();
System.out.println("2+3 = " + calc.add(2, 3));
System.out.println("2-3 = " + calc.sub(2, 3));
calc.remove();
```
Session Bean Discussion

• What’s the tradeoff of stateful session beans?
• Disadvantages / Advantages?
• How to avoid stateful session beans?
• When to use stateful session beans?
EJB – Entity Beans
Entity Beans/1

- represent persistent data – view to a database
- know how to render themselves into persistent storage
- EJB specification does not dictate any specific storage mechanism
  - simple serialization storage
  - Object-relational mapping (SQL)
  - Object databases
- Long-living components (state stored)
- Each entity bean has an unique key – primary key
Entity Bean types

• **BMP – Bean Managed Persistency**
  – You are responsible to implement the required database code (load, store, remove)

• **CMP – Container Managed Persistency**
  – Container handles all the database code
  – Typically special „clue“ code is generated
  – CMP is required since EJB 1.1
  – mapping has many restrictions
    • No support for dependent objects (complex java objects as attributes of an entity bean)
    • No support for attributes which reference other EJB
    • No „dirty detection“
    • Improvements in EJB 2.0
Entity Bean Primary Key

- Identifies one entity bean instance
- A primary key may contain any number of attributes
- Primitive values may be used as primary keys
- You can define your own primary key classes
  - PK Class must be serializable
  - You should overwrite Object.equals() and Object.hashCode()
- for SQL mapping - Entity Bean primary keys are typically equal to the primary key of a SQL-Table
Pooling and Caching

- **Pooling**: Container holds a collection of bean instances
- **Caching**: Container holds a map of ready(loaded) entity beans.
Entity Bean Lifecycle

Quelle: Ed Roman
Mastering Enterprise Java Beans
Wiley Computer Publishing
Discussion: CMP versus BMP

- Disadvantages / Advantages ?
- How to avoid unnecessary database queries
- When to use Entity Beans ?
- Business Logic and Entity Beans ?
The ejb-jar file is the standard format for packaging of enterprise Beans.
The ejb-jar file format is used to package unassembled enterprise beans (partly configured), and to package assembled applications (full configured).

Parts of the EJB-JAR
- Home- and Remote Interfaces (compiled)
- Bean Implementations (compiled)
- dependend classes
- Deployment descriptor (Configuration)
Deployment Descriptor (DD)

- Typically the deployment descriptor will be configured with a tool provided by your application server provider
- DD is encoded as XML document
- There are two basic kinds of information in the deployment descriptor:
  - Enterprise beans’ structural information (mandatory)
  - Application assembly information describes how the enterprise bean (or beans) in the ejb-jar file is composed into a larger application deployment unit. (optional)
Deployment Descriptor Sample

```xml
<ejb-jar>
  <description>Description of the contents of this ejb-jar</description>
  <enterprise-beans>
    <session>
      <description>bean description</description>
      <ejb-name>EmployeeService</ejb-name>
      <home>com.wombat.empl.EmployeeServiceHome</home>
      <remote>com.wombat.empl.EmployeeService</remote>
      <ejb-class>com.wombat.empl.EmployeeServiceBean</ejb-class>
      <session-type>Stateful</session-type>
      <transaction-type>Bean</transaction-type>
      <env-entry>
        <env-entry-name>envvar1</env-entry-name>
        <env-entry-type>String</env-entry-type>
        <env-entry-value>some value</env-entry-value>
      </env-entry>
    </session>
  </enterprise-beans>
  <assembly-descriptor>
  </assembly-descriptor>
</ejb-jar>
```
EJB Transactions

• Transaction may be
  – Bean managed (you code it) – programmatic
    • You get access to the UserTransaction interface from the Context object
  – or container managed (you configure it) – declarative

• Note: Entity beans always use container managed transactions!
Transaction attributes

- May be either defined for all methods or for none!
- Transaction attributes are configured – not coded
- Possible types:
  - **NotSupported** – the method does not support transactions – an already running transaction will be suspended
  - **Required** – the method requires to run in a transaction (a new transaction will be created if no one exists)
  - **Supports** – the method supports a transaction but does not require one for execution
  - **RequiresNew** – the method requires a new transaction (an already running transaction will be suspended)
  - **Mandatory** – the method requires an already running transaction
  - **Never** – the method does not support transactions – if there is an already running transaction an exception is thrown
Further Information

- J2EE at javasoft: www.javasoft.com/j2ee
- The Serverside.com: www.theserverside.com
- Applicationserver:
  - JBoss (Open Source): www.jboss.org
  - BEA Weblogic: www.bea.com
  - IBM WebSphere: www.ibm.com/websphere
  - Borland Appserver: www.borland.com
  - Oracle, ...