Enterprise JavaBeans

Mikhail Chalabine
(a number of) slides by
Jens Gustavsson

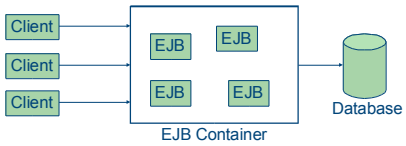
Introduction: EJB

- An EJB is standard distributed component
- The EJB is a part of the J2EE standard from Sun
- Server side component architecture
- Implementation by independent tool vendors
  - **Proprietory**: IBM (WebSphere), BEA (WebLogic), Sun and Netscape (iPlanet), Oracle, Borland
  - **Open source**: JBoss (www.jboss.org)
- Enterprise JavaBeans ≠ JavaBeans

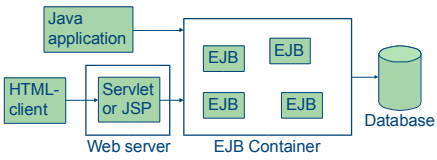
Introduction: EJB

- Separate business logic from middleware services:
  - networking
  - transactions
  - persistence
  - logging
  - resource pooling
- EJB Container / Application server
  - Manages beans
  - Provides middleware services
- Allows `JSP`s, `Servlet`s, Java applications, and other `EJB`s act as clients

EJB Architecture



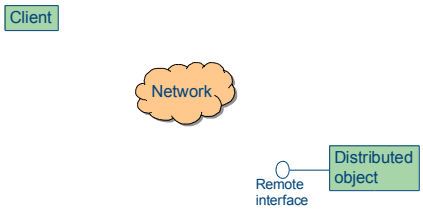
Clients (typical use cases)



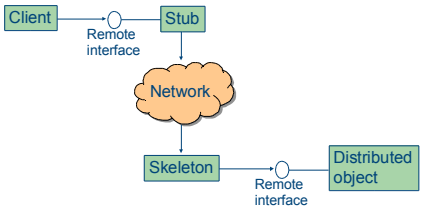
Implicit Middleware

- **Explicit middleware** (e.g. CORBA) :
  - Write to API
  - Difficult to write, maintain and support
- **Implicit middleware** (e.g. EJB)
  - Write isolated business logic
  - Declarative middleware service specifications
  - Middleware services automatically
  - Tool support

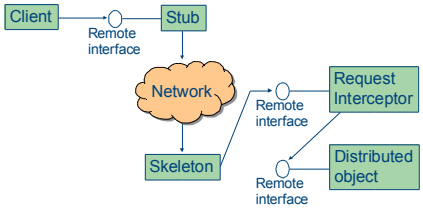
Distributed Objects



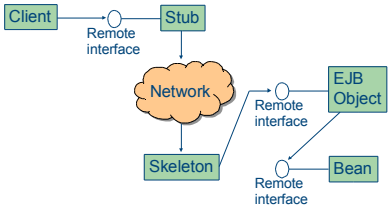
Distributed Objects



Distributed Objects



Distributed Objects the EJB way



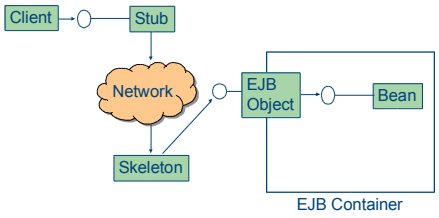
EJB Architecture

- Client calls a method on the EJB object
- EJB object delegates the call to a bean
- EJB receives the result
- EJB passes the result to the caller

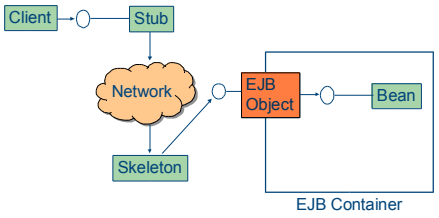
To create an EJB provide

- Home interface
  - Defines the life cycle methods of the bean
- Remote interface
  - Defines the business methods of the bean
- Bean class
  - Business logic

Enterprise JavaBeans



EJB Object (Remote Interface)

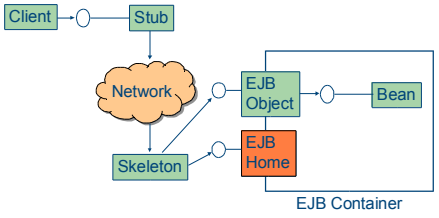


EJB Object (Remote Interface)

- Extends `javax.ejb.EJBObject`
- Defines business methods clients call (implementation in the bean class)
- Acts as a proxy

```
package ejbExample.interfaces
// This is a remote interface for HelloBean
public interface Hello extends javax.ejb.EJBObject
{
    public String Hello() throws java.rmi.RemoteException;
}
```

EJB Home Object (Home Interface)

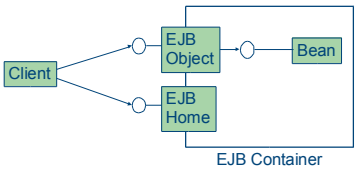


EJB Home Object Characteristics

- Extends `javax.ejb.EJBHome`
- Acts as a factory to create EJB instances
- Allows clients to create/remove/find EJBs

```
package ejbExample.interfaces
// This is a home interface for HelloBean
public interface HelloHome extends javax.ejb.EJBHome
{
    Hello create() throws java.rmi.RemoteException,
                          javax.ejb.CreateException;
}
```

Summary: EJB Architecture



Summary: an EJB consist of

- Enterprise Bean class
- Supporting classes
- EJB Object
- Remote interface
- Home object
- Deployment descriptor (XML)
- Vendor-specific files
- (Local interface)

EJB-jar file

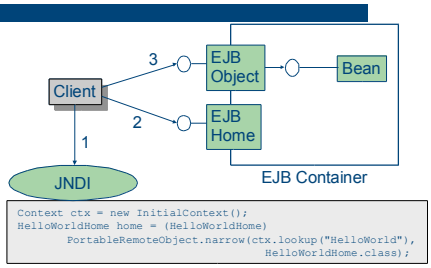
Deployment

- EJB deployment descriptor (XML)
- **ejb-jar.xml**
- Attributes of the beans specified declaratively
- Deployment descriptor language is a composition language
- EJB-jar file is verified by container
- Container generates stubs and skeletons

How clients find the Home object

- Java Naming and Directory Interface (**JNDI**)
  - Similar to CORBA naming service
  - Mapping between resource names and physical locations
- No machine address to home object hard coded
  - Address to JNDI server is needed
  - Kept in the initial context
  - Use initial context factory to acquire an **initial context** (is the JNDI driver)
  - Vendor specific, bound to J2EE server implementation

EJB Architecture




```
Context ctx = new InitialContext();
HelloWorldHome home = (HelloWorldHome)
        PortableRemoteObject.narrow(ctx.lookup("HelloWorld"),
                                    HelloWorldHome.class);
```

Types of Beans

- Session beans
  - Stateless
  - Stateful
- Entity beans
- Message-Driven beans

So, what does the container do?

- Generate stubs and skeletons
- Create EJB instances as needed. Pooling instances.
- Persisting entity beans.
- Handles security and transactions via EJB object

How can container vendors compete?

- Caching strategies
- Development tool integration
- Database access optimization
- Performance

XDoclet

- Deployment descriptor
- Generate from declarative specification
  - **Remote interface**
  - **home interface**
  - **local interface**
  - **local home interface**
  - **primary key class**
- Specification as comments in the `Bean class`

Demonstration

Our first bean

Local interfaces

- When beans call beans locally
- Optimization
- Call by value/reference problem

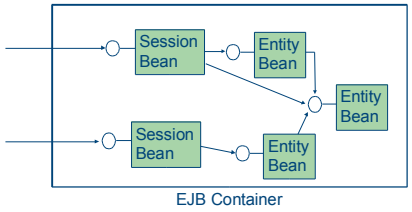
Entity Beans

- Represent business data stored in database
- Database types converted to Java types
- Change of values in the Entity Bean is propagated to the DB

How is Persistence Achieved?

- Bean managed persistence (**BMP**)
- Container managed persistence (**CMP**):
  - Object to relational database mapping (common)
  - Object databases (uncommon)
  - Container generates persistence as subclass
  - **EJB-QL**, query language
- **An entity bean is a view into the data source, e.g., a database**

Façade design pattern for EJB



Security

- Authentication - JAAS
- Authorization
- Deployment descriptor
  - Roles
  - Roles and methods
- No instance level based security

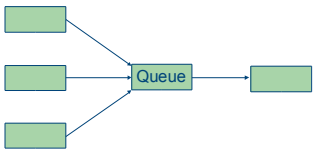
Demonstration

An entity bean

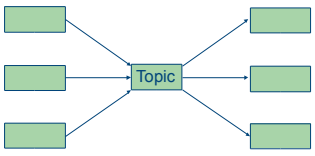
Message-Driven beans

- Don't have home, remote or local interfaces
- Have a single business method:
  - onMessage
- No static type check
- No return values
- No exceptions
- Stateless

Point-to-Point



Publish - Subscribe



Why Message-Driven Beans?

- Performance
- Reliability
- Support for multiple senders and receivers
- Easy integration to legacy systems

Final thoughts

- Is it object-oriented?
  - Separation of data and operations (entity beans and session beans)
  - No inheritance between beans
- Suitable for which tasks?
  - One architecture. Anomalies if trying to do anything else
- Component marketplace?
  - Not today!

Resources

- Szyperski, chapter 14
- Sun EJB tutorial
  http://java.sun.com/j2ee/learning/tutorial/index.html
- Ed Roman: Mastering EJB
  http://www.theserverside.com/books/wiley/masteringEJB/index.jsp
- JBoss, Open source EJB Container
  http://www.jboss.org