



California State
University
Los Angeles

Server Site Web Development for E-Business in Java and .NET

Dr. Jongwook Woo
jwoo5@calstatela.edu

08/22/2003

For SookMyung Women's University, Seoul, Korea

Department of Computer and Information Systems
California State University, Los Angeles

Jongwook Woo



Computer and
Information Systems



CSLA

Outline

- E-Business
- N-Tier Web Architecture
- Distributed Systems for E-Business
- E-Business in J2EE
- E-Business in .NET
- Clusters and Load Balancing

Jongwook Woo



Computer and
Information Systems



E-Business

- **Business in Internet**

- Distributed Information Systems in Web

- **Business Model**

- Customer Relationship Management
- Supply Chain
- E-Commerce



E-Business (Cont'd)

- **Three Primary Functions in E-Business**

- Presentation Logic
 - GUI
 - Audio/Video input and output
- Business Logic
 - Defines how the application handles data
 - How to process data
- Data Management Logic
 - Deals with DBs
 - S/W to manage the DBs



Client Server Architectures

■ Traditional Client Server Architecture

- Thick Client
 - Presentation Logic and Business Logic in client
- Expensive in client maintenance
- Non Scalable in Business Logic for Web Application

■ N-Tier Web Architecture

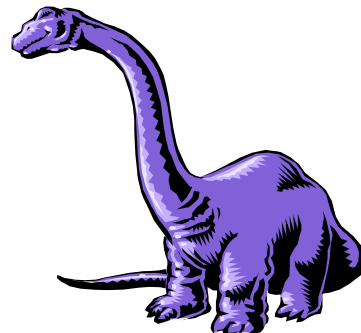
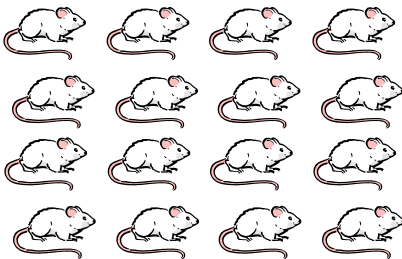
- Thin Client
- Separates an enterprise system into its individual roles
 - Presentation, Business, and Data Management Logics are separated
 - scalable
 - maintainable
 - better performance



Distributed System

■ A collection of autonomous computers linked

- by a network (Internet)
- with software designed to produce an integrated computing facility





Distributed System (Cont'd)

- **More**

- Available
 - Highly available 24 x 7 with multiple machines
- Scalable
 - Multiple machines can easily expand as a business grows up
 - Single machine is not easy to expand
- Maintainable
 - Business constantly changes
 - Separates data access logic from GUI
 - Makes the business logic reusable



Traditional Distributed Systems for E-Business

- **Developer needs to build sophisticated functions such as**
 - Transaction
 - Setting up message queues
 - Database Replication and Synchronization
 - Planning for fail-over
 - Web Security
- **Needs to spend time to implement such logics**
- **Needs a more sophisticated design to be scalable**



CSLA

Distributed Systems in J2EE for E-Business

Distributed System

■ Java Platform 2 Enterprise Edition

- JDBC, JNDI, RMI, Java IDL, Servlets, JSP, EJB, JTS, JTA, JavaMail, Java Message Services, (XML)

■ Applications published with and built in J2EE

- Sophisticated functions in old traditional systems are already built
- Standardized
- Written in Java
- Can be deployed into any server

■ Developer can mainly focus on the business logic

- Save time
- Scalable

Jongwook Woo



Computer and
Information Systems



CSLA

Distributed Systems in J2EE for E-Business (Cont'd)

J2EE

■ J2EE is the way to go for E-Business

- System View
 - Distributed System
- Client-Server Architecture View
 - N-Tier Architecture
- Function View
 - Three separated functional Logics

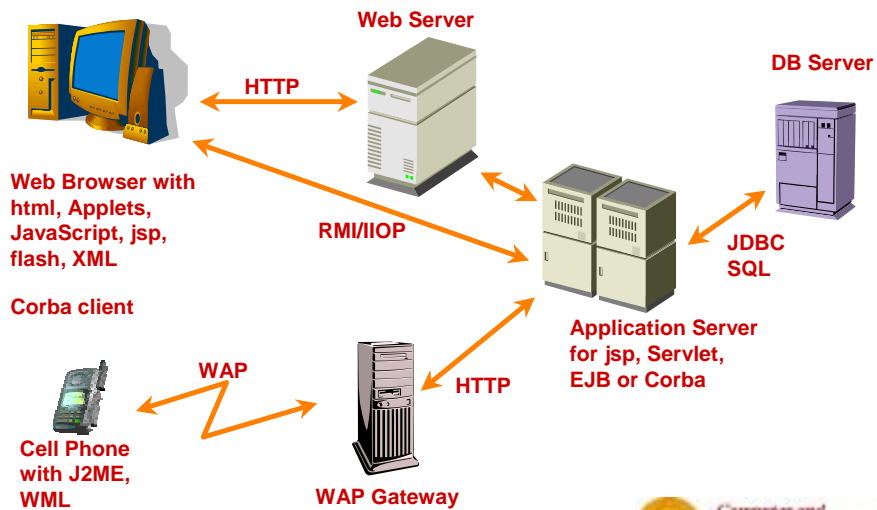
Jongwook Woo



Computer and
Information Systems



N-Tier Web Architecture



Jongwook Woo

Computer and
Information Systems

Clients

■ Web Clients

- Usually a browser
- Displays HTML or XML from a Web or Application server
- Interact with a Application server via JSP or Servlet
- May run JavaScript (or Applets)

■ Application Clients

- Client that do not execute within a browser
- Can communicate through RMI, IIOP, DCOM, TCP/IP

Jongwook Woo

Computer and
Information Systems



Servers

■ Web Servers

- Provide static web content
- Communicate through HTTP or FTP
 - Request/response
- Can be a proxy server of Application servers

■ Application Servers (App Servers)

- Provide dynamic web content
- Execute heavier functions such as JSP, Servlet, EJB etc.
 - Have Infrastructures such as transaction, security etc.

■ Proxy Servers

- Forward requests to other machines
- Can be a Load balancer



Functional APIs in Server Sites

■ Servlets

- Handle Request and Response Service in App Server
- Execute business or Data management logics in App Server
 - Retrieve/update shopping data in DB
 - Collect the information and create HTML documents

■ JSP (Java Server Pages)

- Text based and editable Servlets
 - Can be inserted in HTML documents
 - Good presentation logic

■ JDBC

- Standard Java Interface for accessing heterogeneous DBs

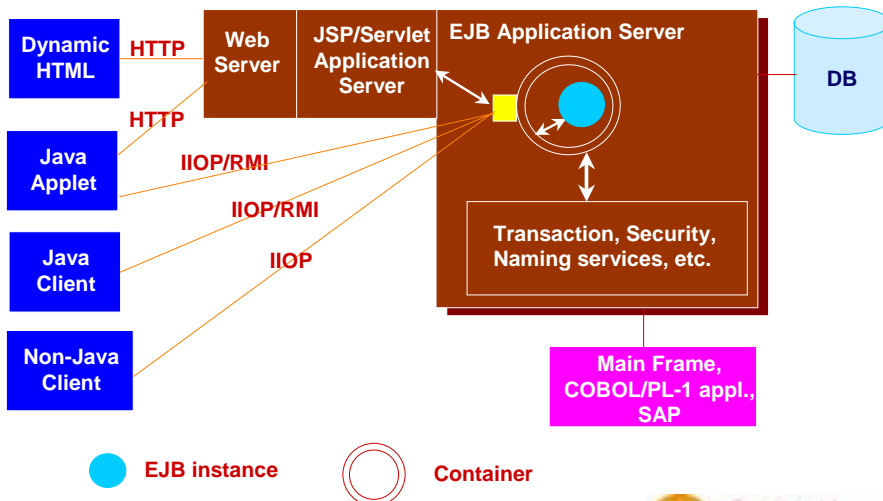


■ EJBs (Enterprise JavaBeans)

- Components based
 - Provide deployable and distributable business logics to clients
- Built-in Infrastructure Service of EJB server
 - Easy to implement specially in Database Management logic
- Business Application
 - Developer focuses on business logic not infrastructure service nor Data Management Logic
- Summary comparing to Servlet and JSP
 - (+) Much easier to implement Business and Data Management Logic
 - (+) More distributable (Can sell your EJBs)
 - (-) Performance Issue



EJBA





Containers

■ Infrastructure Services in Application Server

- Security
- Transaction
- Database Management
- Licensing

■ A Container in Application Server

- Intercepts communication between the client and EJB to allow automation of infrastructure code
- Communicate with EJB using direct method call



Enterprise JavaBeans™

■ Application Server

- JSP, Servlet, EJB server
 - Bea WebLogic, IBM WebSphere, ATG Dynamo, Oracle9iAS, Jrun, Jboss
- JSP, Servlet Server
 - TomCat



EJB 1.1 Interfaces

■ Home Interface

- Home Stub/Skeleton
- Create/remove an EJB instance by a client
- Has its stub placed into JNDI at server start-up

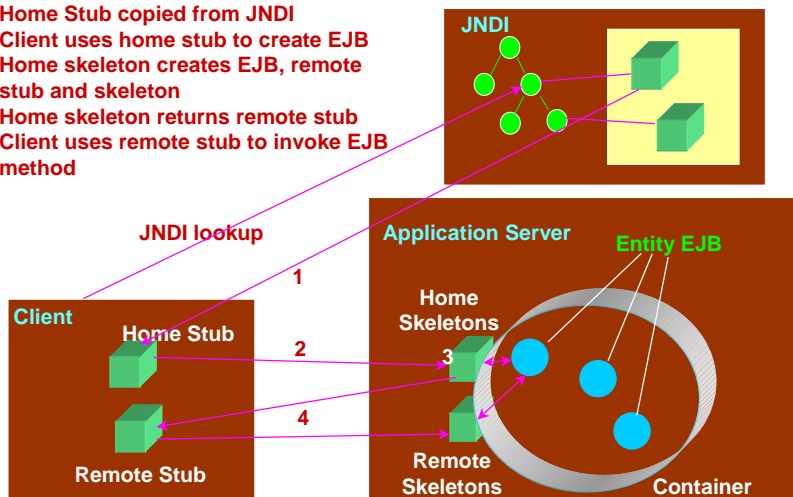
■ Remote Interface

- Remote Stub/Skeleton
- Contains the business operations of an EJB



EJB Invocation

1. Home Stub copied from JNDI
2. Client uses home stub to create EJB
3. Home skeleton creates EJB, remote stub and skeleton
4. Home skeleton returns remote stub
5. Client uses remote stub to invoke EJB method





Enterprise JavaBeans™

■ Basic beans in EJB1.1

- Session Bean
 - For Business Logic
 - Stateless Session Bean
 - Stateless sever approach
 - Stateful Session Bean
 - Session oriented approach
- Entity Bean
 - For Data Management Logic
 - Persistent object approach
 - CMP and BMP

■ Additional bean at EJB2.0, 2.1

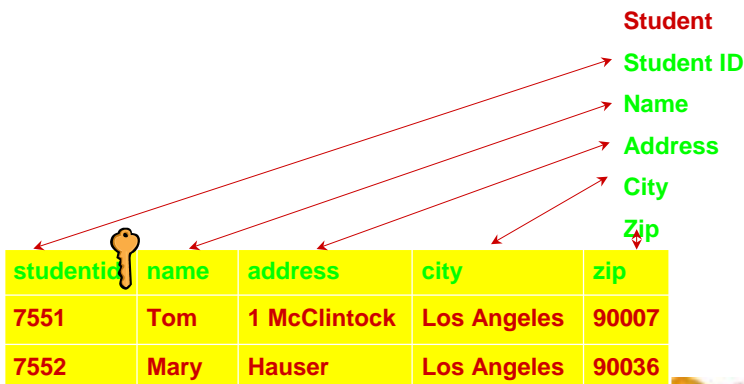
- Message Driven Bean (MDB)
 - Asynchronous message passing
 - No interfaces



EJB and Relational DB Mapping

■ Object-Relational Mapping

- Each object stores a single/set of row (s) of a table
- Each attribute of a class is associated with each column of a DB





EJB Object Pooling

■ When an application server starts

- Create a pool of EJB objects
- Save an overhead such as creation and destruction of EJB
- When releasing an EJB object, it is placed back to the pool

■ Passivation

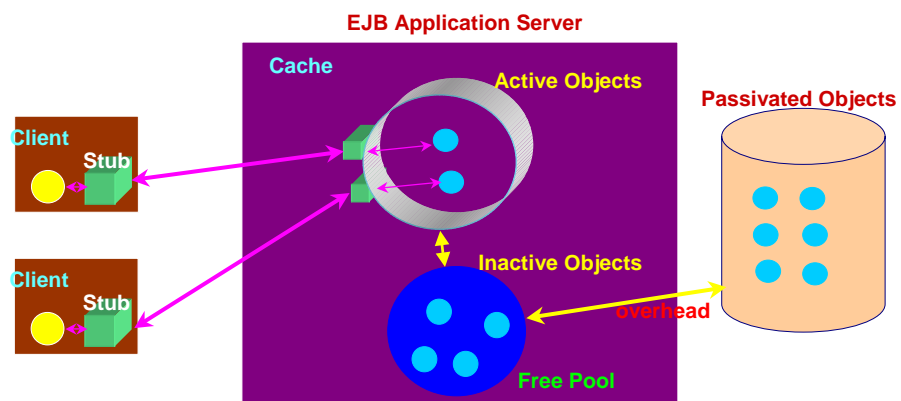
- The act of taking an EJB and placing it into secondary storage
- When an EJB meets its idle time
- Places an serialized instance of the EJB into storage

■ Activation

- When an EJB is requested by a client
- Its object is brought back into memory



Pooling/Passivation/Activation





.NET

- **Microsoft has presented .NET solution in June 2000**
 - Want to compete with J2EE
 - Want to occupy the world
- **Mainly web solution**
 - Platform Independent
 - Final goal but not now
 - Language Independent
 - Visual Basic .NET, C++ .NET, C# .NET, ASP .NET, ADO .NET
- **Component Model**
 - Easy to distribute
 - Same as EJB



.NET (Cont'd)

- **ASP .NET**
 - Active Server Pages in HTML tags of a ASPX file
 - Its function can be written in many languages
 - Similar to JSP
- **ADO .NET**
 - ActiveX Data Object
 - Manage data in DBs
 - Similar to JDBC



.NET Framework (Cont'd)

- **Need to install .NET Framework**
- **Frame Class Library (FCL)**
 - Infrastructure such as transaction, security
 - EJB container
- **Common Language Runtime (CLR)**
 - Compiled code from .NET language is Microsoft Intermediate Language (MSIL)
 - Codes in different languages can link each other via MSIL
 - Similar to Java IDL
 - CLR should be on each platform
 - Similar to JVM



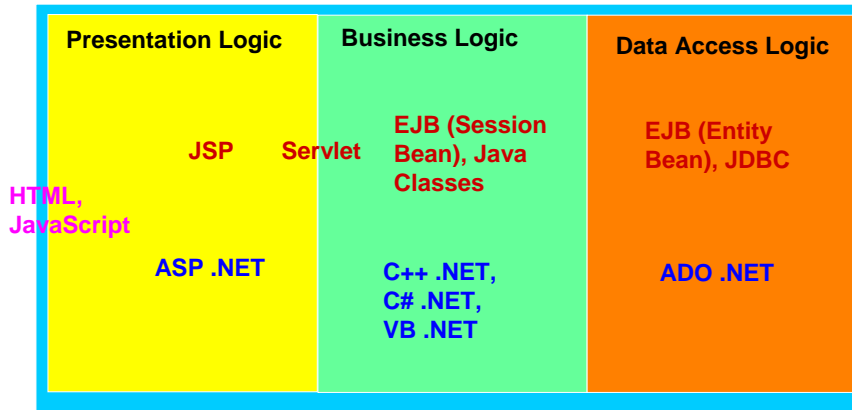
Comparing J2EE and .NET

	J2EE	.NET
Dynamic Web Content	JSP	ASP .NET
DB Access	JDBC	ADO .NET
Platform Independent	Yes	Not Yet (Microsoft OS only)
Languages	Java	C++, C#, Visual Basic
Component Model	Yes (EJB)	Yes
Market Proven	Yes	Not yet
Cost of product	Some freewares	No freewares
Performance	?	?





Comparing J2EE and .NET (Cont'd)



Clusters

- **The number of servers connected via Internet**
 - Acts as one big server
 - To handle millions of clients
 - Gain High-Performance
 - One of parallel and distributed systems
- **Advantages – Same as Distributed Systems**
 - Scalable
 - Highly Available
 - Highly Reliable
 - Maintainable



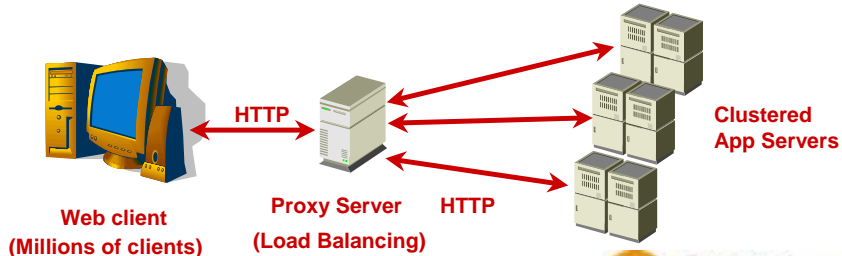
Load Balancing

■ Load Balancing Algorithm

- Round Robin
- Random
- Statically Weighted Round Robin – Pre-assigned weight

■ Load Balancing Issues

- E-Business applications are too dynamic
 - Not easy to measure response time for each request



Dynamic Load Balancing Algorithm

■ Dynamically Weighted Round Robin

- Assign a weight for each server dynamically
- Weight is determined by
 - The number of requests for each server
 - The total process times of requests for each server
- Need information for each server
 - The number of requests processed
- Need information for each request
 - Processing time



Future of J2EE and .NET in N-Tier Architecture

- **IT and E-Business world need many J2EE or .NET Engineers**
- **School who knows J2EE or .NET can be a leading school**
 - Distributed Computing in Objects
 - New and challenging for research and lecture
 - Students: Easy to find good jobs in IT
- **The Knowledge of J2EE or .NET can be useful for J2ME, Wireless world such as PDA and Cell Phone**



End of the Presentation

Thank you



Session EJB

■ Session EJB

- Do not survive when EJB server crashes

■ Stateless Session EJB

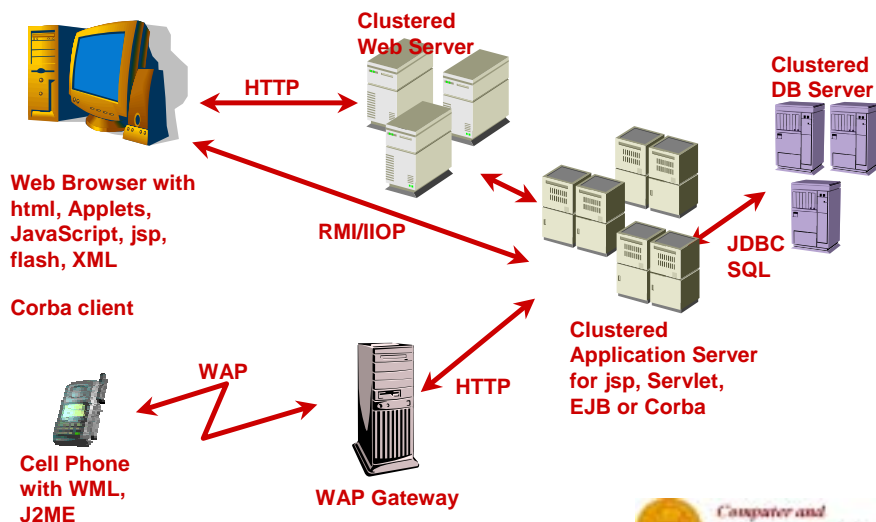
- Do not maintain a state on behalf of a client
- Any two instances of the same stateless session EJB types are always identical
- Ex: calculate $\cos(x)$

■ Stateful Session EJB

- Provides **conversational** interaction
- Stores a state on behalf of a client
- Each instance can only be used by a single thread
- Ex: shopping cart, flight reservation

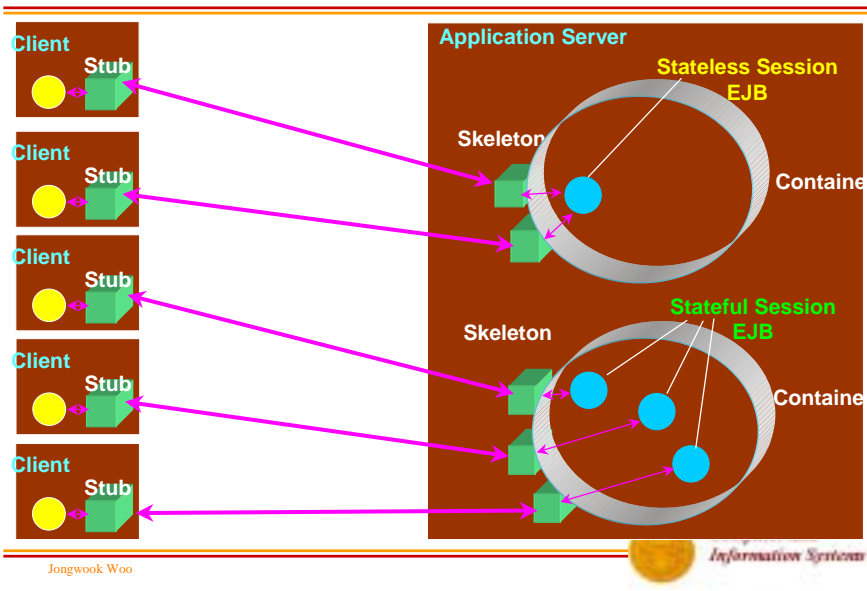


N-Tier Web Architecture





Session EJB Examples



Entity EJB

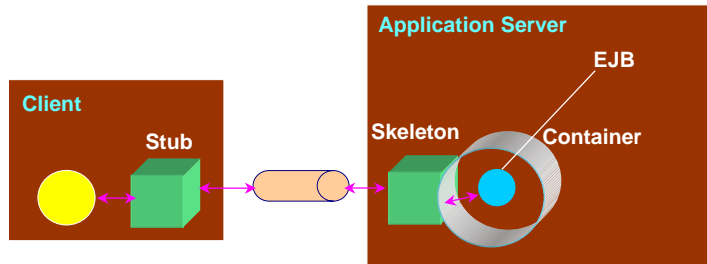
- Represents persistent data
- Can survive a crash
- Multiple clients can be using EJBs that represent the same data
- Has a copy of the data in the persistent store
 - Serialization to a file
 - Relational data mapping through JDBC
- Example
 - An EJB for a football player's career statistics
 - An EJB for an employee's timesheet



Containers

■ Container and Infrastructure Service

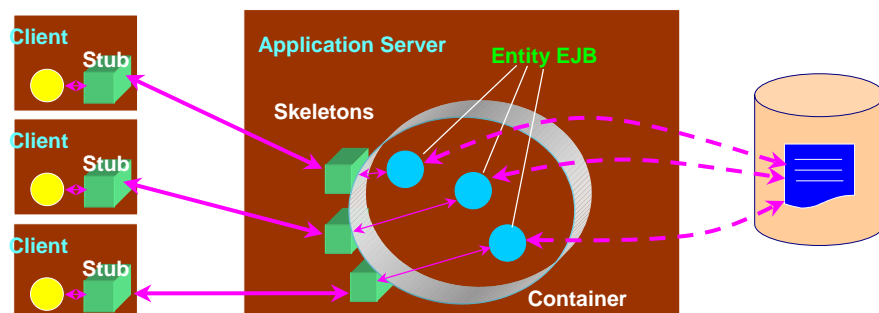
- EJB developers can focus on the business logic



Shared Entity EJB

■ Multiple clients share an Entity EJB

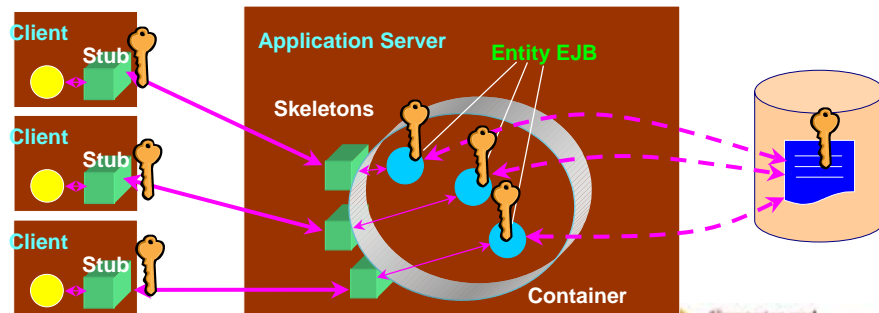
- They have their own instances
- The instances share the underlying data
- Container and server handles the transactions



Identical Entity EJB

Primary Keys

- Every Entity EJB has a unique set of attributes
- The aggregated attributes are called **primary key** of the Entity EJB
- Two Entity EJBs are identical if the contents of the **primary keys** are identical



CMP and BMP Entity EJB

Container-Managed Persistence (CMP)

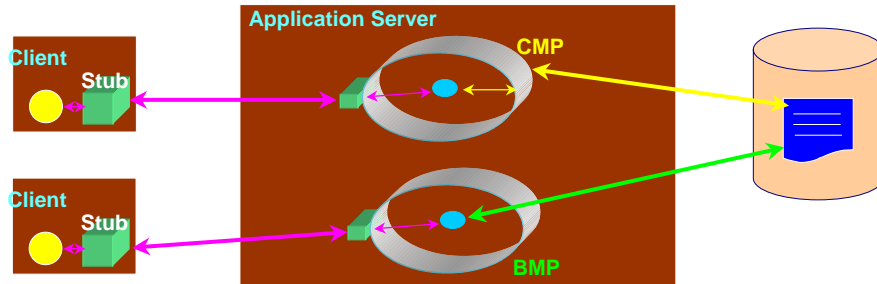
- EJB developer does not need to build the persistence access logic
- The logic is inside the container
- (+) Easy to build
- (-) Performance issue

Bean-Managed Persistence (BMP)

- EJB developer need to build the persistence access logic
- (+) Can handle performance even limited
- (-) Expensive to build



CMP and BMP Entity EJB



Message Driven Bean

- **JMS in J2EE**
 - Synchronous message passing in Session and Entity EJB
- **Message Driven Bean (MDB)**
 - Asynchronous message passing
 - No home and remote interface
 - Has only a Bean class



Review

■ Research Interests

- Clustering in web and App servers
- Load-Balancing algorithm in BGP on IPv6
- Voice XML in Web
- WML (XML)



Example codes

■ Session EJB

- EJB name: SessionDemoBean
- Home Interface: SessionDemoHome
- Remote Interface: SessionDemo

// JNDI lookup

```
Context ctx = getInitialContext();
```

```
SessionDemoHome dhome
```

```
    = (SessionDemoHome)
```

```
        PortableRemoteObject.narrow(ctx.lookup("SessionDemoBean"),  
                                    SessionDemoHome.class);
```

// Create an instance of the SessionDemo bean from a home interface

```
SessionDemo demo
```

```
    = (SessionDemo)
```

```
        PortableRemoteObject.narrow (dhome.create(), SessionDemo.class
```



Example codes

■ Entity EJB

- EJB class: EntityDemoBean
- Home Interface: EntityDemoHome
- Remote Interface: EntityDemo
- Primary Key class: EntityDemoPK

// JNDI lookup

```
EntityDemoHome Context ctx = getInitialContext();  
EntityDemoHome dhome  
    =(EntityDemoHome)PortableRemoteObject.narrow(  
        ctx.lookup("EntityDemoBean"), EntityDemoHome.class
```

// create Primary Key class

```
EntityDemoPK dPK= new EntityDemoPK("USC");
```

// find a remote interface by using the primary key

```
EntityDemo demo = (EntityDemo) dhome.findByPrimaryKey (dPK);
```