

Enterprise Java Beans



© Copyright IBM Corporation 2001

Agenda

- Enterprise JavaBean(EJB) Definition
- EJB Terms
- EJB Roles
- EJB Applications
- EJBs on the AS/400 System
- Building an AS/400 Application using EJBs



© Copyright IBM Corporation 2001

Objectives



- Offer an introduction to Enterprise JavaBeans
- Show how EJBs work on the AS/400 system



© Copyright IBM Corporation 2001

Enterprise JavaBeans Definition



- Requirement
 - Make it easier to write business applications
 - Remove the need to understand low-level APIs, security, transactions,
- What is Enterprise JavaBeans?
 - Architecture to define how to build components for business applications
 - Components that can be used to build distributed applications
 - RMI Object with some of its attributes configurable at runtime
 - Separate from, but complementary to, JavaBeans



© Copyright IBM Corporation 2001

What is EJB?



- EJB = Enterprise JavaBeans
 - Component model targeted at middle tier (ie, server) business logic development
 - Interface that insulates application programmer from complexities and platform dependencies of transactions systems, databases, security, etc.
 - Sun initiative with broad industry momentum
 - Specification, but no reference implementation
 - Defines a component interface, but not the underlying implementation or packaging
 - Container/server providers allowed to differentiate their offerings



© Copyright IBM Corporation 2001

What is EJS?



- EJS = Enterprise Java Services
 - IBM initiative
 - Specification and reference implementation for use across IBM products
 - Full implementation of EJB interfaces as specified by Sun's EJB specification
 - WebSphere Application Server-Advanced Edition
 - Specification and implementation of underlying services (eg, Directory)
 - Addition of "Connectors" for easy access to IBM programming systems and data (eg, CICS, MQSeries)



© Copyright IBM Corporation 2001

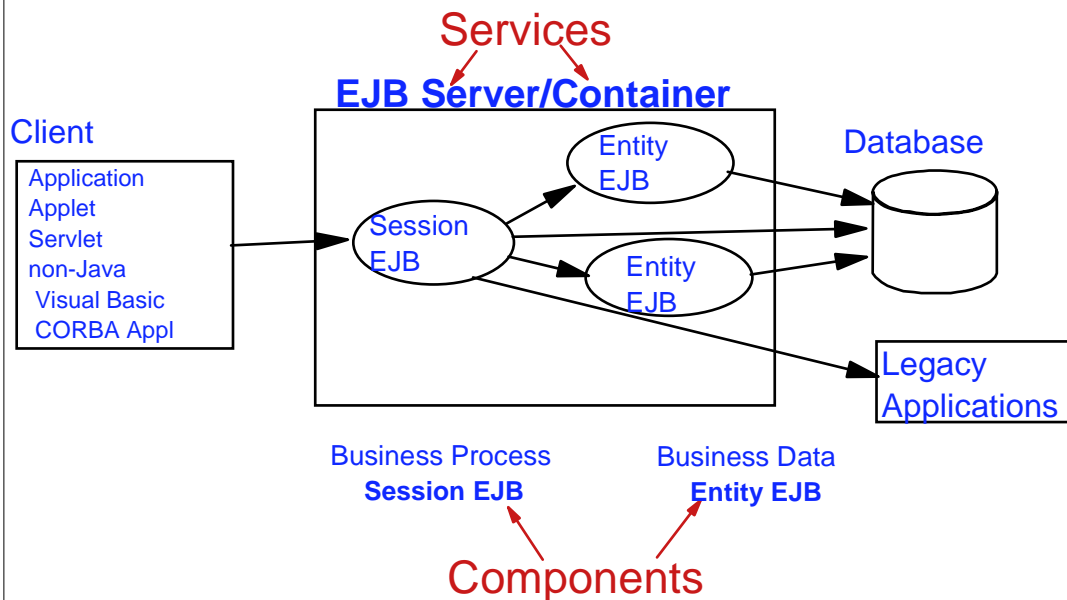
EJB Goals

- Standard server-side component model
- Binary portability -- "write once, run anywhere"
- Make application easy to write
 - Application assembly with visual tools
 - Complexity of security, transactions, and persistence hidden
- Standard interfaces for tool suppliers
- Interoperate with non-Java components (using IIOP and COM bridges)
- Coexist with and reuse of existing server components such as LDAP directories and XA Resource Managers.



© Copyright IBM Corporation 2001

Enterprise JavaBeans Architecture



© Copyright IBM Corporation 2001

EJB Development Roles

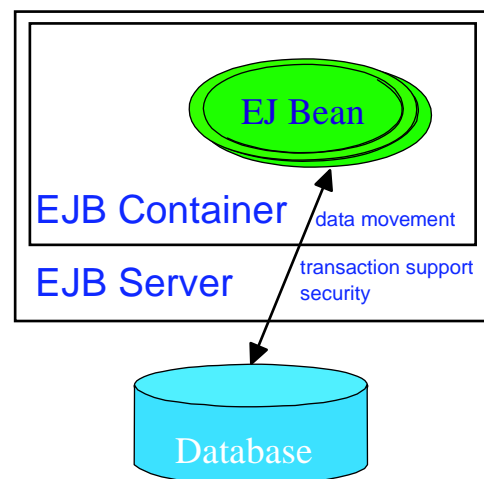
- System Services
 - EJB container provider
 - EJB server provider
- EJB Programming Model
 - Enterprise Bean provider
- EJB Component Model
 - Application assembler
 - Deployer



© Copyright IBM Corporation 2001

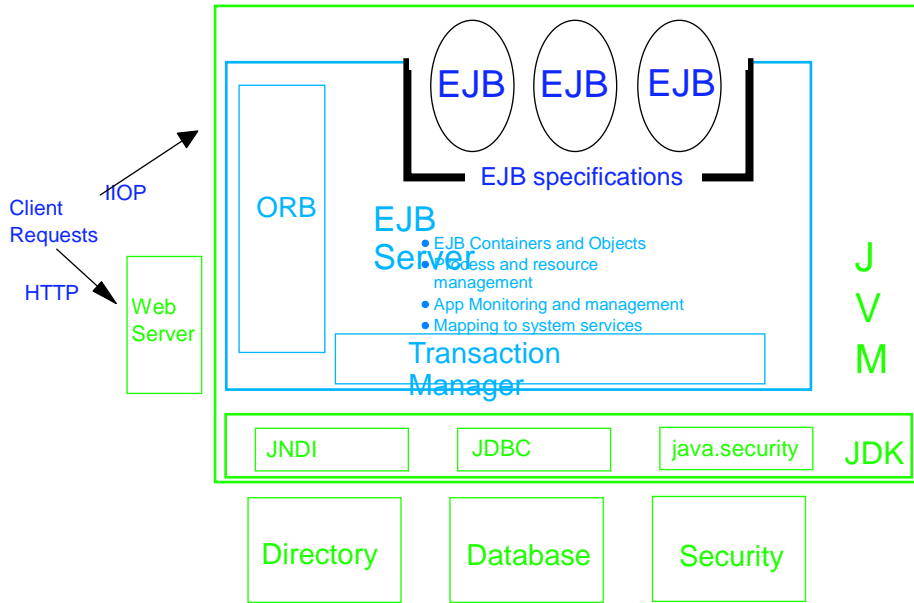
System Services Provider

- EJB container provider
 - moves data between EJ Bean and data base (or application)
 - protects EJ Bean provider from having to know EJB server interfaces
 - uses EJB server system services
- EJB server provider
 - system services such as transactions, security
 - may provide a container to persist data to a specific data source
 - may publish interfaces, so others can provide containers



© Copyright IBM Corporation 2001

EJB Server Structure

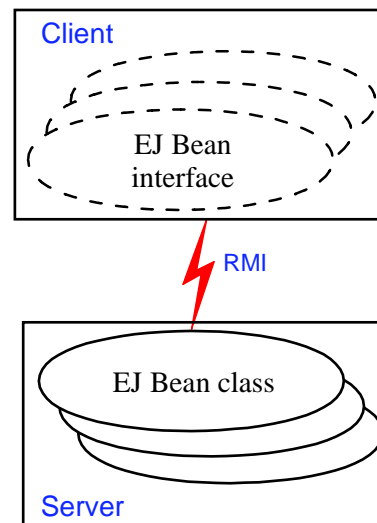


© Copyright IBM Corporation 2001

EJB Programming Model

Enterprise Bean provider

- Develops domain specific components that can be reused
- Provides
 - interfaces for application assembler
 - class to implement business logic
 - descriptor for deployment
- May rely on EJB container to provide system services



© Copyright IBM Corporation 2001

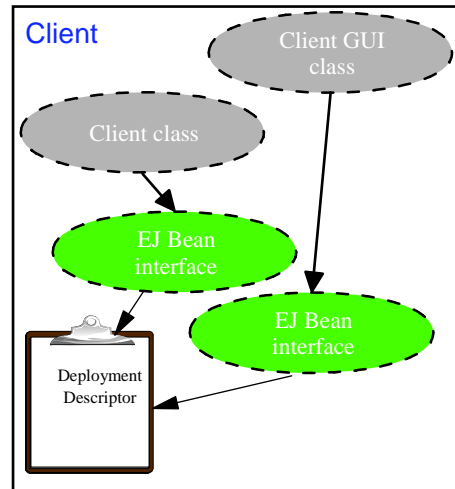
EJB Component Model

Application Assembler

- Creates applications using enterprise Beans with the contracts defined by the interfaces
- May customize Beans by changing the descriptor

Deployer

- Deploys enterprise Beans into containers for a specific server
- May customize Beans by changing the descriptor



© Copyright IBM Corporation 2001

EJB Deployment Descriptor

- Special object that describes the deployment attributes of an EJB
- Packaged and shipped with each EJB
- Attributes to describe
 - Security behavior
 - Transactional behavior
 - Persistence
- Originated by EJB provider
- May be customized during application assembly and deployment



© Copyright IBM Corporation 2001

Types of Enterprise JavaBeans



■ Session Beans

- Transient objects representing business tasks
- Typically reads/writes data to a database for a client
- Represent an action being taken on behalf of a single client
- May maintain state across method calls
- Examples: OrderPlacement, OrderEntryClerk

■ Entity Beans

- Persistent objects representing business data
- Each entity EJB is uniquely identifiable (primary key)
- Persistence managed by the container or the bean itself
- Examples: Stock, Order



© Copyright IBM Corporation 2001

Bean Persistence Choices



■ Applies only to entity beans

■ Container-managed persistence

- container provider manages reading and writing bean attributes to data base
- bean provider sets up a mapping from bean attributes to data base columns which the container uses
- advantage: less programming for bean provider and bean code is independent of persistence details

■ Bean-managed persistence

- bean provider manages reading and writing bean attributes to data base
- advantage: greater flexibility in mapping to data bases and bean can interface to existing applications



© Copyright IBM Corporation 2001

Enterprise JavaBean Interfaces



- Applies to entity and session beans
- Interfaces
 - Contract between bean provider and client
 - Defines the methods that the client may use
- Home interface
 - Methods used by clients to create, remove and locate EJB objects
 - Implemented by the container
 - Accessed via directory
 - Registered in directory by EJB Server
 - Located through directory by client
- Remote interface
 - Business methods that may be performed on an entity or session bean
 - Implemented by the bean provider in the EJB object



© Copyright IBM Corporation 2001

Java Packages



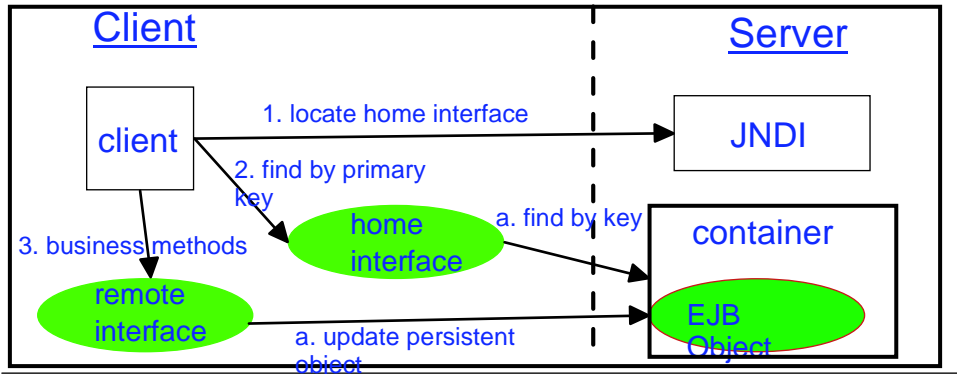
- javax.ejb - EnterpriseJavaBeans interfaces and classes
- javax.jts - Java transaction services
- javax.naming - Java naming and directory interfaces (JNDI)
- rmi - remote method invocation (RMI)



© Copyright IBM Corporation 2001

Example Entity Bean Usage

- Use JNDI to locate home interface
- Use finder interface
 - locate a particular instance by primary key
- Execute one or more business methods
 - update persistent business object



© Copyright IBM Corporation 2001

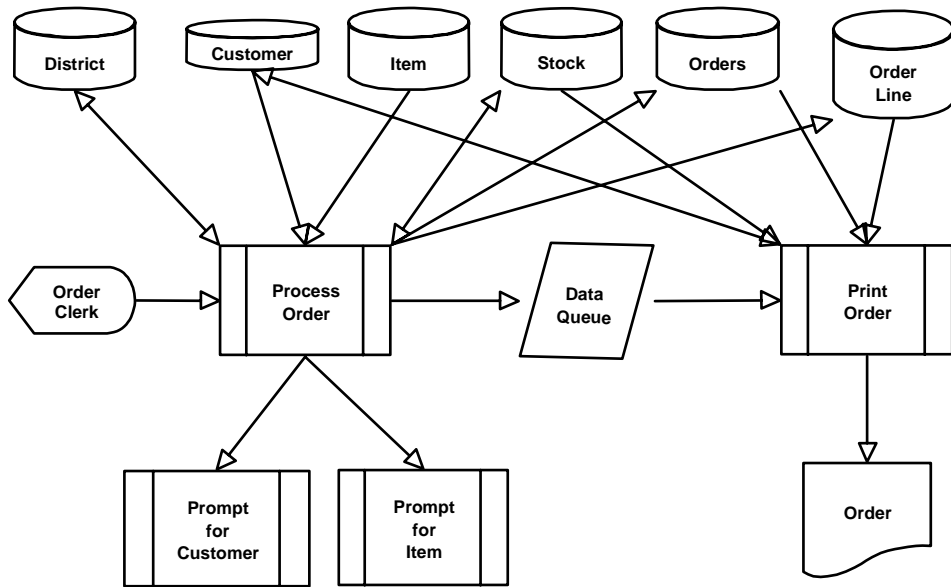
EJB Deployment Process

- EJB Provider
 - Delivers EJB jar file containing
 - Manifest file (identifies EJBs)
 - Class files for EJBs
 - Serialized Deployment Descriptors for EJBs
- Application Assembler
 - Assembles EJBs
 - Possibly from multiple EJB Providers
 - May customize Deployment Descriptor attributes
 - Develops client application to provide user interface
- Deployer (using Container provided tools)
 - May customize Deployment Descriptor attributes
 - Generates EJB code
 - Container-unique code based on Deployment Descriptor attributes
 - RMI stubs and skeletons for remote objects



© Copyright IBM Corporation 2001

Order Entry Application



© Copyright IBM Corporation 2001

Application Objects

District	
ivDistrictID	: String
ivWarehouseID	: String
ivNextOrderNumber	: int
ivDistrictName	: String
ivDistrictTax	: float
ivYTDBalance	: float

Customer	
ivCustomerID	
ivFirstName	
ivLastName	
ivInitials	
ivPhone	
ivCreditLimit	
ivBalance	
ivYTDBalance	

Address	
ivAddrLine1	
ivAddrLine2	
ivCity	
ivState	
ivZip	

Item	
ivItemID	: String
ivItemName	: String
ivItemPrice	: float
ivItemInfo	: String

Stock	
ivWarehouseID	: String
ivItemID	: String
ivStockQuantity	: int
ivOrderQuantity	: int
ivRemOrderQuantity	: int
ivYearToDate	: int

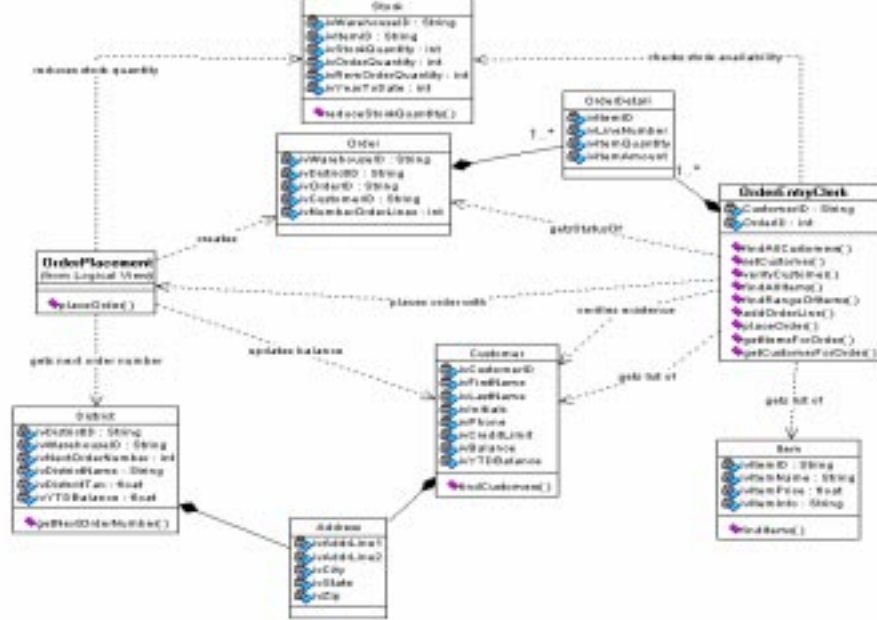
Order	
ivWarehouseID	: String
ivDistrictID	: String
ivOrderID	: String
ivCustomerID	: String
ivNumberOrderLines	: int

OrderDetail	
ivItemID	
ivLineNumber	
ivItemQuantity	
ivItemAmount	



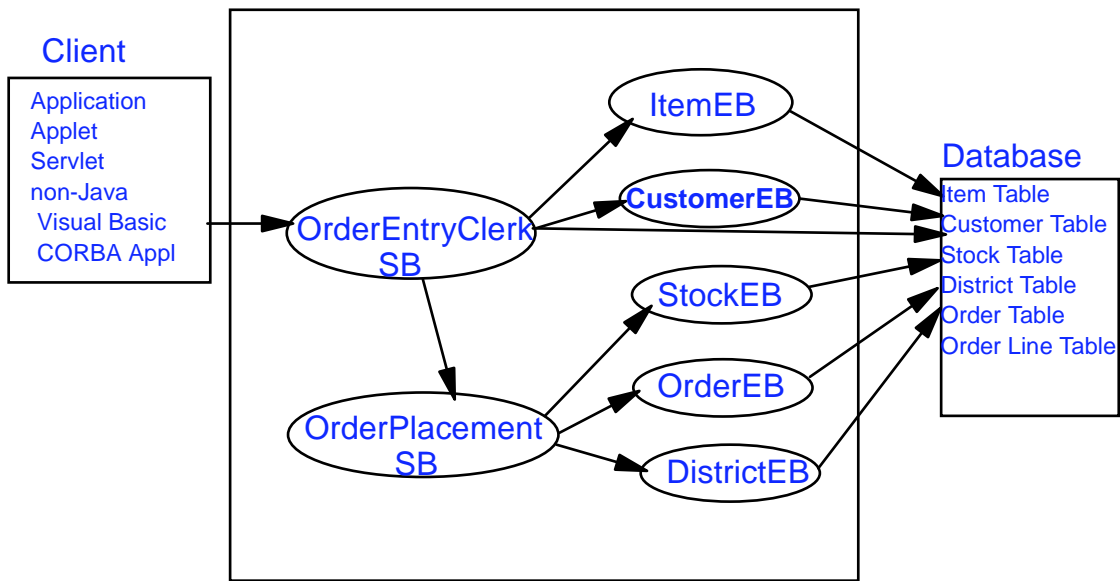
© Copyright IBM Corporation 2001

Object Relationships



© Copyright IBM Corporation 2001

Order Entry Application EJBs



© Copyright IBM Corporation 2001

OrderEntryClerk Session Bean



- The OrderEntryClerk encapsulates the ordering process
 - stateful session bean
 - keeps a record of who the customer is and what items they want to order
 - The source code for the OrderClerk bean consists of the following four files:
 - orderentryclerk.txt - The text file used to create the deployment descriptor
 - OrderEntryClerkHome.java - The bean's home interface
 - OrderEntryClerk.java - The bean's remote interface
 - OrderEntryClerkBean.java - The bean's implementation



© Copyright IBM Corporation 2001

Home Interface for a Session EJB



```
package com.ibm.itso.roch.cpwejb.interfaces;

import javax.ejb.*;
import java.rmi.RemoteException;

public interface OrderEntryClerkHome extends EJBHome {

    public OrderEntryClerk create()
        throws CreateException, RemoteException;
}
```



© Copyright IBM Corporation 2001

Home Interface for an Entity EJB

```
package com.ibm.itso.roch.cpwejb.interfaces;
import java.rmi.*;
import javax.ejb.*;
public interface ItemHome extends EJBHome {
    public Item create(String id, String name, float price, String data)
        throws RemoteException, CreateException;
    public Item findByPrimaryKey(ItemID primaryKey)
        throws FinderException, RemoteException;
    public Enumeration findAllItems( )
        throws FinderException, RemoteException;
    public Enumeration findByName(String name)
        throws FinderException, RemoteException;
    public Enumeration findByPrice(float price)
        throws FinderException, RemoteException;
    public Enumeration findCheaperItems(float price)
        throws FinderException, RemoteException;
    public Enumeration findMoreExpensiveItems(float price)
        throws FinderException, RemoteException;
}
```



© Copyright IBM Corporation 2001

Remote Interface for OrderEntryClerk

```
package com.ibm.itso.roch.cpwejb.interfaces;
import java.rmi.RemoteException;
import java.rmi.Remote;
import javax.ejb.*;
import java.util.*;
/*=====*/
public interface OrderEntryClerk extends EJBObject, Remote {
    public Vector findAllCustomers() throws RemoteException;
    public Vector findAllCustomersVB() throws RemoteException;
    public void setCustomer(String cid) throws RemoteException, CpwejbException;
    public Vector findAllItems() throws RemoteException;
    public Vector findAllItemsVB() throws RemoteException;
    public Vector findRangeOfItems(String lowID, String highID) throws RemoteException;
    public void addOrderLine(String iid, int quantity) throws RemoteException;
    public boolean verifyCustomer(String cid) throws RemoteException;
    public String placeOrder() throws RemoteException, CpwejbException;
    public void placeOrderRPG() throws RemoteException, CpwejbException;
}
```



© Copyright IBM Corporation 2001

Bean Implementation



```
package com.ibm.itso.roch.cpwejb.server;
import com.ibm.itso.roch.cpwejb.interfaces.*;
import javax.ejb.*;
import javax.jts.*;
import javax.naming.*;
import java.sql.*;
import java.util.*;
import java.io.Serializable;
import java.rmi.RemoteException;
import com.ibm.as400.access.*;

// ***** //
public class OrderEntryClerkBean implements SessionBean {
// Following is the state in this stateful session bean
    public String custID;
    public Vector items;
```



© Copyright IBM Corporation 2001

setCustomer



```
public void setCustomer(String cid) throws RemoteException {
    if ( verifyCustomer(cid) ) {
        custID = cid;
    } else {
        throw new CpwejbException("Customer id " + cid + " not valid");
    }
}
```



© Copyright IBM Corporation 2001

Finding other EJBs



```
//Looking up the Remote Interface an Entity EJB
InitialContext initCtx = (InitialContext)getInitialContext();
ItemHome homeForItem = (ItemHome) initCtx.lookup
    ("com.ibm.itso.roch.cpwejb.ItemHome");
```

```
//Looking up the Remote Interface for a Session EJB
OrderPlacementHome home = (OrderPlacementHome) initCtx.lookup
    ("com.ibm.itso.roch.cpwejb.OrderPlacementHome");
OrderPlacement placement = home.create();
```



© Copyright IBM Corporation 2001

addOrderLine



```
// Add an item and quantity to the order
public void addOrderLine(String iid, int quantity) throws RemoteException {
    if ( custID == null ) {
        throw new CpwejbException("OrderEntryClerkBean: Customer ID must be set first");
    }

    Item item = null;
    try {
        item = homeForItem.findByPrimaryKey(new ItemID(iid));
    } catch (Exception e) {
        throw new RemoteException(e.getMessage());
    }
    OrderDetail ol = new OrderDetail(iid, item.getItemPrice()*quantity, quantity);
    items.addElement(ol);
}
```



© Copyright IBM Corporation 2001

placeOrder

```
public String placeOrder() throws RemoteException {
    String orderNumber = null;
    if ( custID == null ) {
        throw new CpwebjException("OrderEntryClerkBean: Customer ID must be set before placing an order");
    }
    if ( items.size() == 0 ) {
        throw new CpwebjException("OrderEntryClerkBean: No order lines. Cannot place order.");
    }
    try {

        float number = placement.placeOrder(whid, did, custID, items);
        orderNumber = Float.toString(number);
        orderNumber = orderNumber.substring(0,orderNumber.length()-2);
        // Clear out the state of the session bean at this point
        items = new Vector();
        custID = null;
    } catch (Exception e) {
        throw new RemoteException(e.getMessage());
    }
    return orderNumber;
}
```



© Copyright IBM Corporation 2001

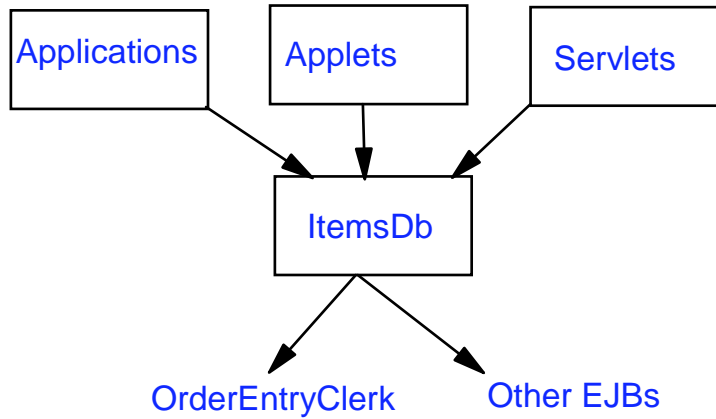
findRangeOfItems

```
public Vector findRangeOfItems(String lowID, String highID) throws RemoteException {
    Vector itemVector = new Vector();
    Item item = null;
    try {
        Connection con = getConnection();
        Statement stmt = con.createStatement();
        ResultSet rs = stmt.executeQuery("select iid, iname, iprice, idata, stqty from " + getCollection() + "item, " +
            getCollection() + "stock where iid = stiid and iid >= " + lowID +
            " and iid <= " + highID + "");
        while ( rs != null && rs.next() ) {
            String itemString[] = new String[5];
            itemString[0] = rs.getString("iid");
            itemString[1] = rs.getString("iname");
            itemString[2] = "$" + Float.toString(rs.getFloat("iprice"));
            itemString[3] = Integer.toString(rs.getInt("stqty"));
            itemString[4] = rs.getString("idata");
            itemVector.addElement(itemString);
        }
        stmt.close();
        releaseConnection(con);
    } catch (Exception e) {
        throw new RemoteException(e.getMessage());
    }
    return itemVector;
}
```



© Copyright IBM Corporation 2001

Client Applications



© Copyright IBM Corporation 2001

Client code

```
import com.ibm.itso.roch.cpwejb.interfaces.*;
import javax.ejb.*;
import javax.naming.*;
import java.rmi.RemoteException;
import java.rmi.Remote;
import java.util.*;

public class ItemsDb extends java.lang.Object {
```



© Copyright IBM Corporation 2001

Client code



```
public static Context getInitialContext() throws Exception {
    Properties p = new Properties();
    try {
        p.put(Context.INITIAL_CONTEXT_FACTORY, "weblogic.jndi.T3InitialContextFactory");
        p.put(Context.PROVIDER_URL, "T3://" + getSystemName() + ":" + getPort());
        p.put(Context.SECURITY_PRINCIPAL, getUserId());
        p.put(Context.SECURITY_CREDENTIALS, getPassword());
        InitialContext cx = new InitialContext(p);
        return cx;
    } catch (Exception e) {
        System.out.println("error creating Context");
        throw (e);
    }
}
```



© Copyright IBM Corporation 2001

Client code



```
public String connect() throws Exception {

    try {
        Context ctx = getInitialContext();
        OrderEntryClerkHome home = (OrderEntryClerkHome)
            ctx.lookup("com.ibm.itso.roch.cpwejb.interfaces.OrderEntryClerkHome");
        clerk = (OrderEntryClerk) home.create();
    } catch (FinderException fe) {

        return ("Could not find order clerk " + fe.getMessage());
    } catch (Exception e) {

        return ("Could not connect " + e.getMessage());
    }

    return "Connected to " + getSystemName() + ":" + getPort();
}
```



© Copyright IBM Corporation 2001

Client code



```
public java.util.Vector findRangeOfItems(String itemnoMin, String itemnoMax) {
    try {
        ejbItems = clerk.findRangeOfItems(itemnoMin,itemnoMax);

    } catch (Exception e) {
        System.out.println("::::::::::: Unexpected Error ::::::::::");
        e.printStackTrace();
    }
    System.out.println("all items returned");
    return ejbItems;
}
```



© Copyright IBM Corporation 2001

Client code

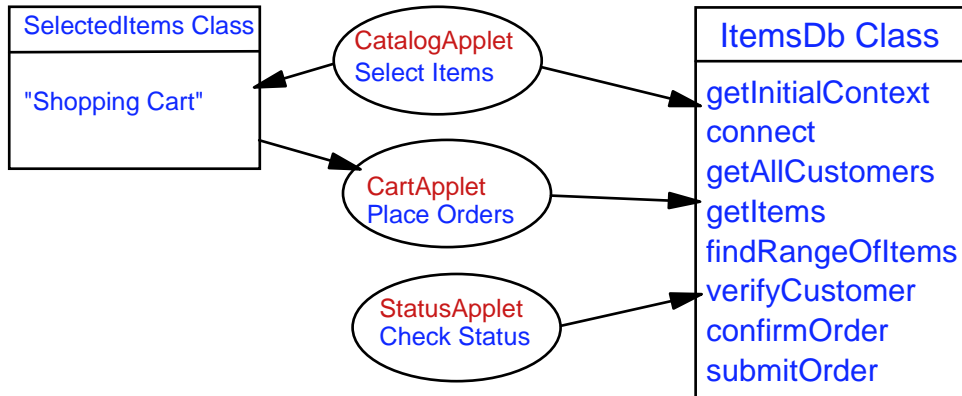


```
public String submitOrder(Order anOrder) throws Exception {
    try {
        OrderDetail orderLine = null;
        //set the customer number
        clerk.setCustomer(anOrder.getCustomer().getId().toString());
        BigDecimal orderLineCount = new BigDecimal(anOrder.getOrderDetails().size());
        for (java.util.Enumeration itemE = anOrder.getOrderDetails().elements(); itemE.hasMoreElements();) {
            orderLine = (OrderDetail) itemE.nextElement();
            clerk.addOrderLine(orderLine.getItem().getId(),
                orderLine.getQuantity());
        }
        // place the order
        String orderNumber = clerk.placeOrder();
        return orderNumber;
    } catch (Exception e) {
        throw (e);
    }
}
```



© Copyright IBM Corporation 2001

Applet Order Entry



© Copyright IBM Corporation 2001

IBM WebSphere Advanced Edition

The complete picture, where
do we use what?



© Copyright IBM Corporation 2001

Servlets



■ Advantages

- Simple
- Easy to develop
- Performance

■ Disadvantages

- Difficult to maintain
- Mix of View and Business logic
- Java programmer writes HTML



© Copyright IBM Corporation 2001

JSP



■ Advantages

- Build with HTML designer
- Dynamic recompile

■ Disadvantages

- Slightly less performant than pure servlets
- Maintenance problems
- Code and View are mixed



© Copyright IBM Corporation 2001

JSP/Servlet



■ Advantages

- Combines advantages of both technologies
- Separation of concerns

■ Disadvantages

- Performance impact
- Maintenance becomes painful for big applications



© Copyright IBM Corporation 2001

EJB



■ Advantages

- Component model
- Maintainable even with large models
- Separation of Concerns
- Many resource related aspects are configurable

■ Disadvantages

- More complex to develop
 - requires "real" object skills
- Performance overhead



© Copyright IBM Corporation 2001

Where to go now....



© Copyright IBM Corporation 2001

Classes

■ WebSphere AS/400 class (4 days)

- Dutch 26 th of March and 8th of May
- French 17th of April and 5th of June
- CONTACT Pierre Danze (Pierre_Danze@be.ibm.com)

■ Watch prerequisites

- Java skills
 - Basic knowledge about the technologies explained here
- Basic OO
- General AS/400 and SQL



© Copyright IBM Corporation 2001