Java™ 2 Platform, Enterprise Edition (J2EE)

Bruno Souza
Java Technologist, Sun Microsystems, Inc.

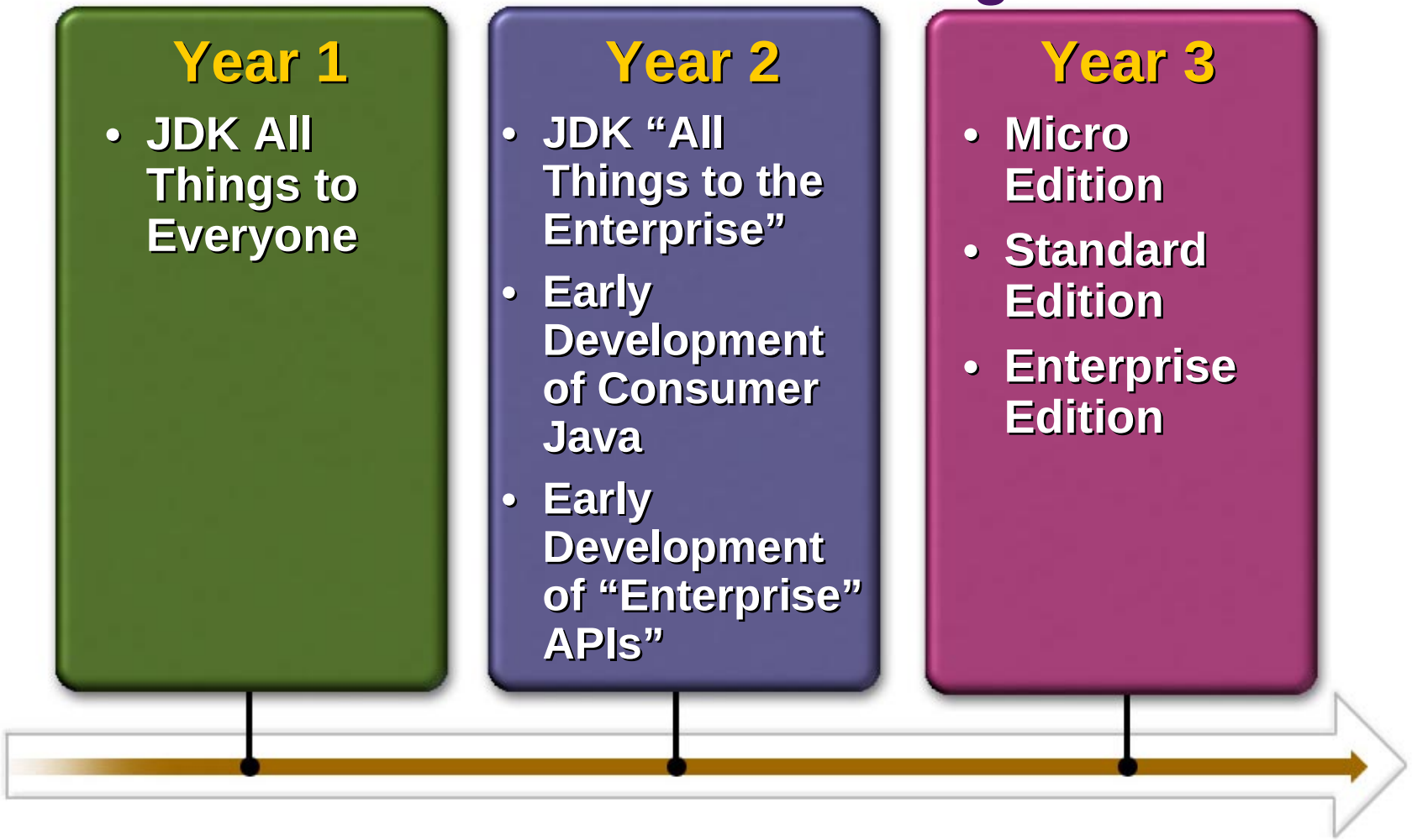
Contents

- The Java™ 2 Platform, Enterprise Edition (J2EE)
- J2EE Environment
- APM and key APM questions
- Application Scenarios and the Sample Application
- Deployment and Security
- Resources

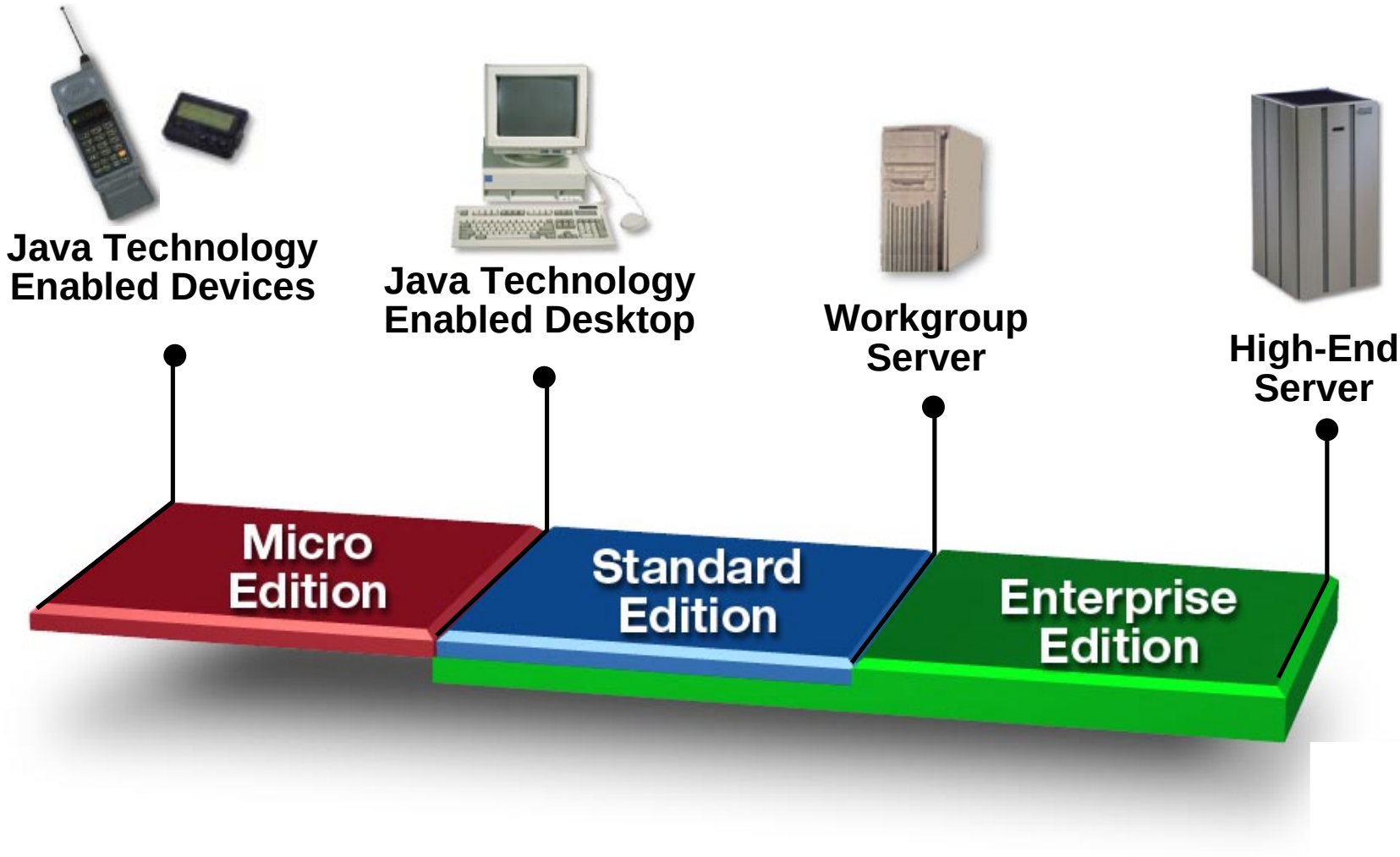
A Short History of Java Enterprise Technology

The Java™ Platform

Where Have We Been?
Where Are We Going?

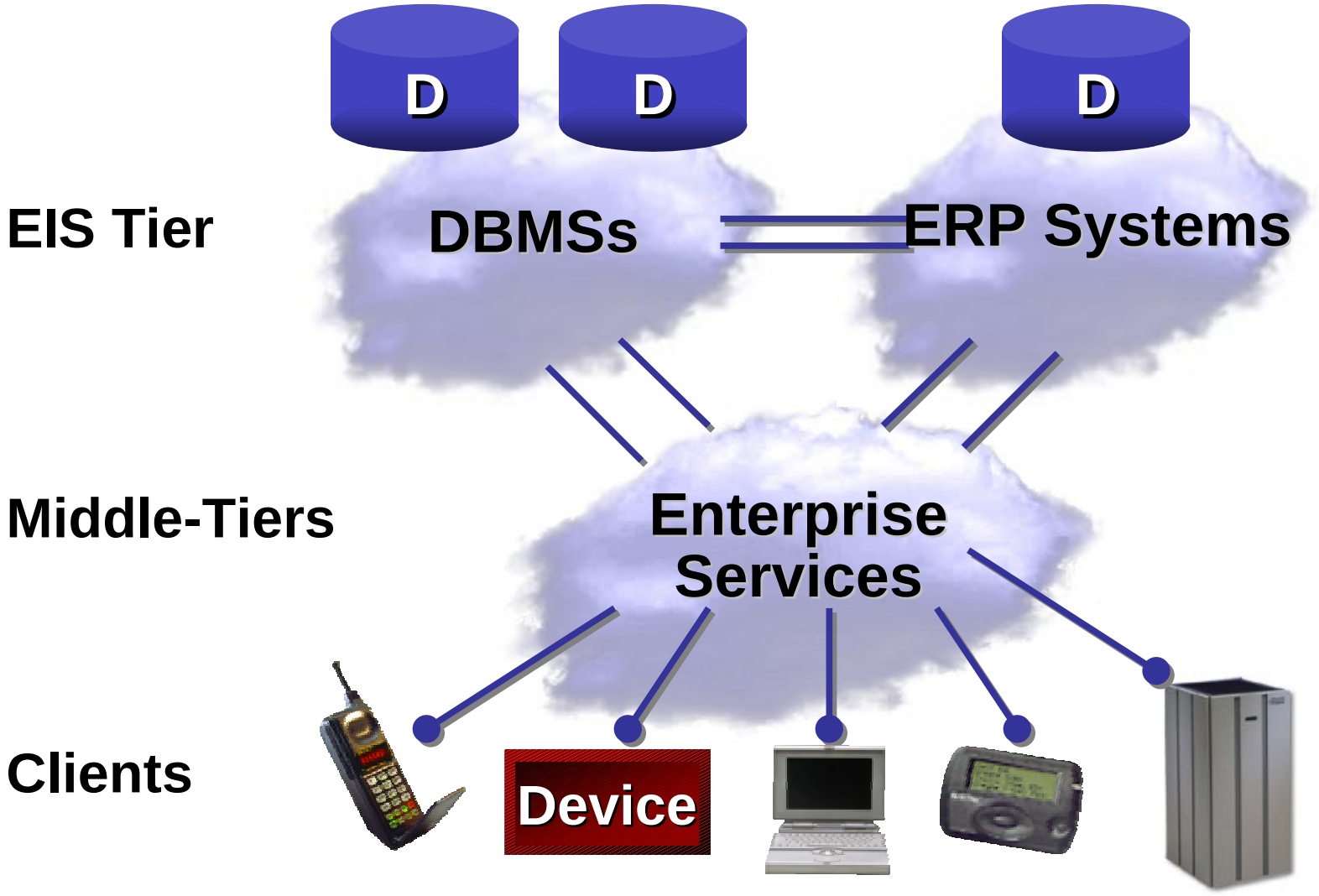


Java 2 Platforms



Why another Java™ 2 Platform?

The Global Enterprise



Enterprise Services Require

- Concurrency (multi user)
- Consistency (Transactions)
- Security
- Availability
- Scalability
- Administration
- EIS Integration
- Distribution

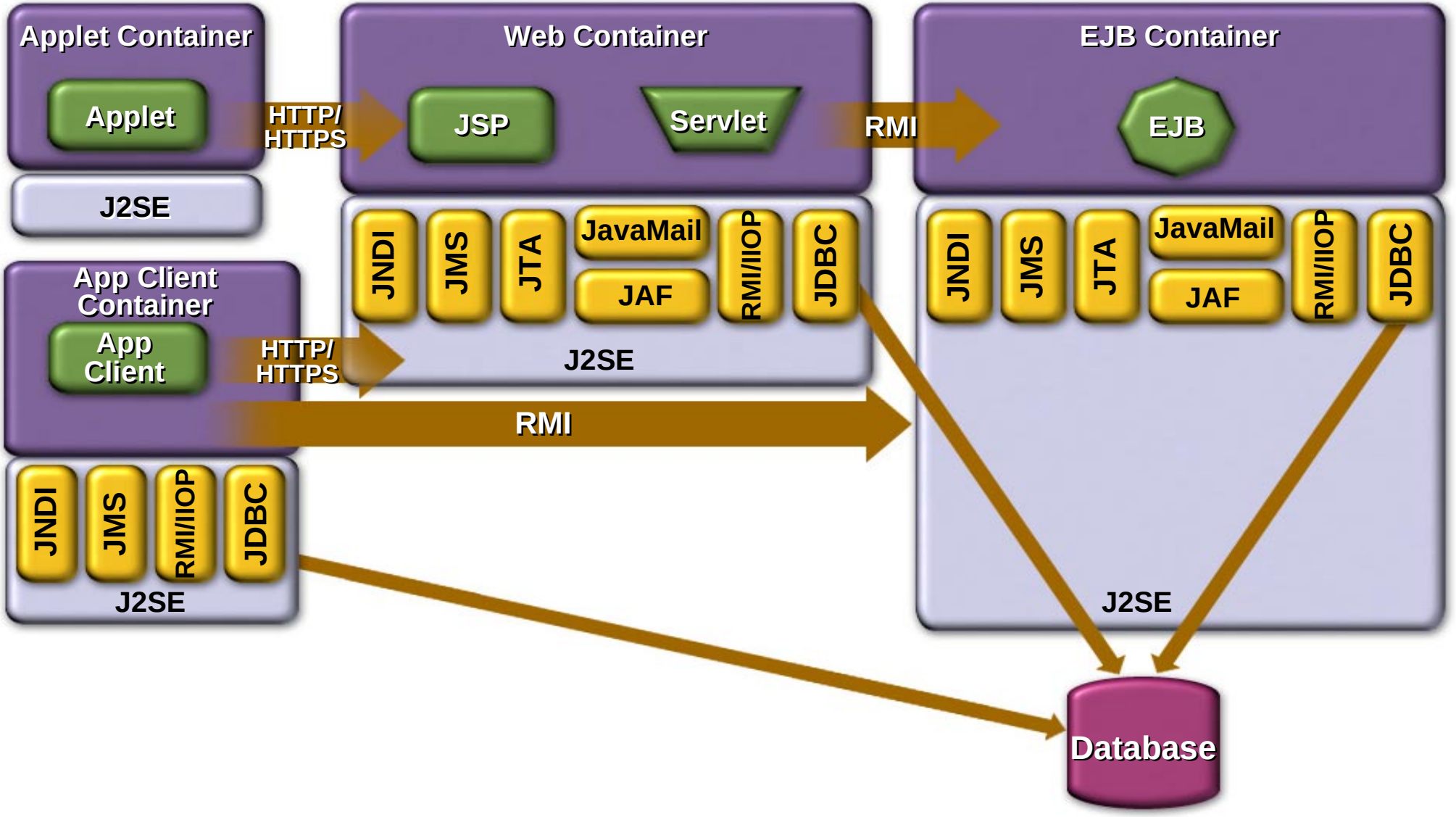
The Java™ 2 Platform, Enterprise Edition

- Develop
  - With the J2EE App Model
- Deploy
  - With the J2EE App unit
- Run
  - On the J2EE Platform

J2EE Platform

What is the purpose of J2EE Containers?

J2EE Platform



Containers and Components

- The container is the car
- The component is the driver

- The container is the platform
- The component is your application

J2EE Containers

- Containers do their work invisibly
  - No complicated APIs
  - They control by interposition
- Containers implement J2EE
  - Look the same to components
  - Have great freedom to innovate

J2EE Containers Handle

- Concurrency (multi user)
- Consistency (Transactions)
- Security
- Availability
- Scalability
- Administration
- Integration
- Distribution

J2EE Components Handle

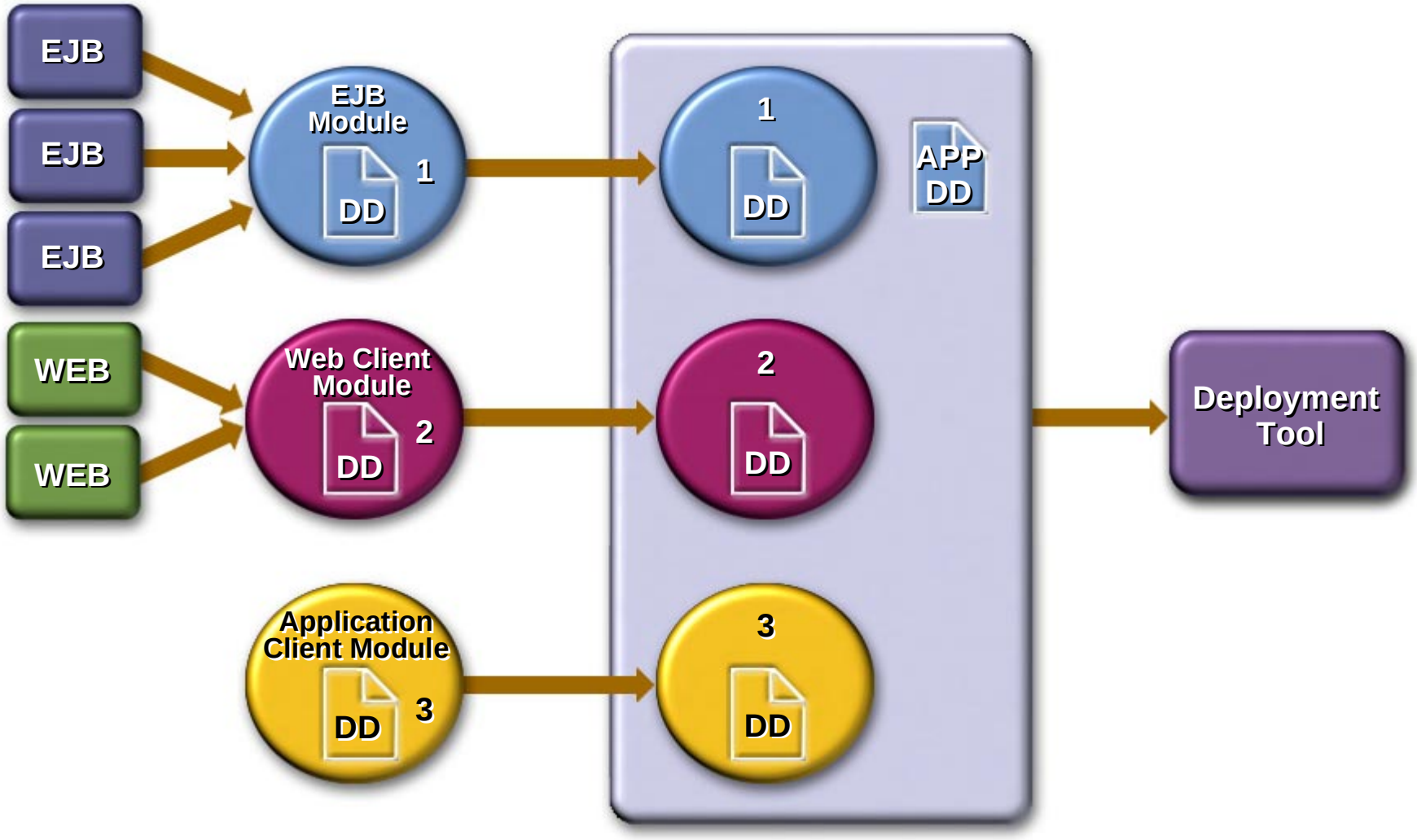
- Presentation
- Business logic
- Data access

Container Perspectives

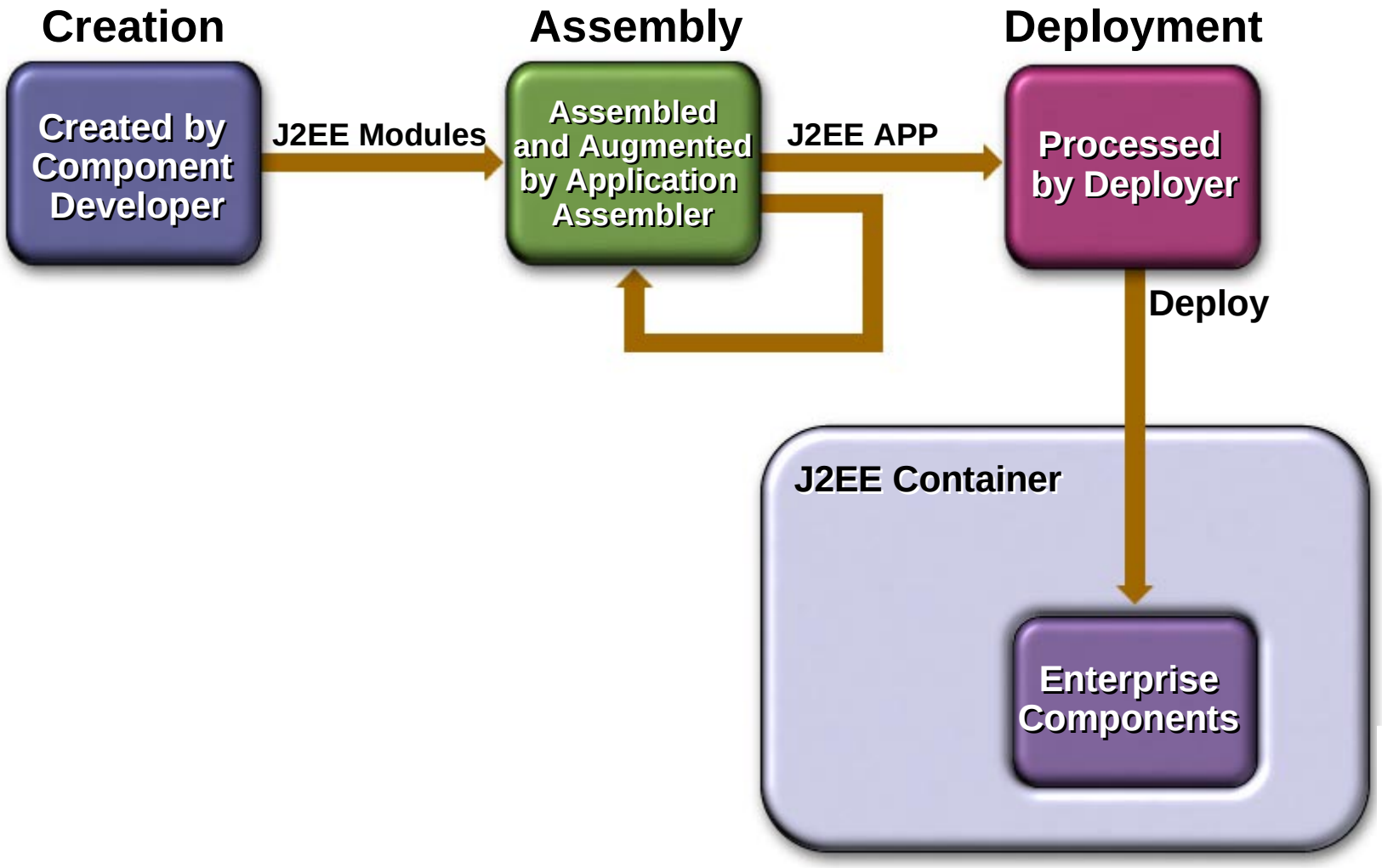
- To a J2EE vendor
  - It is their product
- To a component developer
  - It is a standard app model
- To an app assembler
  - It is a standard app package
- To an executing component
  - It is god

Why Does J2EE Focus on Deployment?

Application Packaging



Application Life Cycle



The Deployer

- Is an expert in the operational environment
  - Familiar with local security practices
  - Familiar with local EIS configuration
  - Familiar with local containers and their apps
- Uses J2EE platform product tools

Deployment Summary

- Apps may be written without knowledge of the operational environment
- Deployment Descriptor communicates app's needs
- A key interface between application developer and platform

What Exactly Is in J2EE?

J2EE API Summary

- J2SE 1.2
- JDBC™ 2.0
- RMI/IIOP 1.0
- EJB 1.1
- Servlet 2.2
- JSP 1.1
- JNDI 1.2
- JTA 1.0
- JMS 1.0
- JavaMail™ 1.1
- JAF 1.0

J2EE Standards

- TCP/IP
- HTTP 1.0
- HTML 3.2
- SSL 3.0
- IIOP 1.0

The Java™ 2 Platform, Enterprise Edition

- **Platform Specification**
  - Defines J2EE requirements
- **Compatibility Test Suite**
  - Validates J2EE platform compatibility
- **Reference Implementation**
  - Operational J2EE platform
- **Application Programming Model**
  - Describes how to build J2EE applications

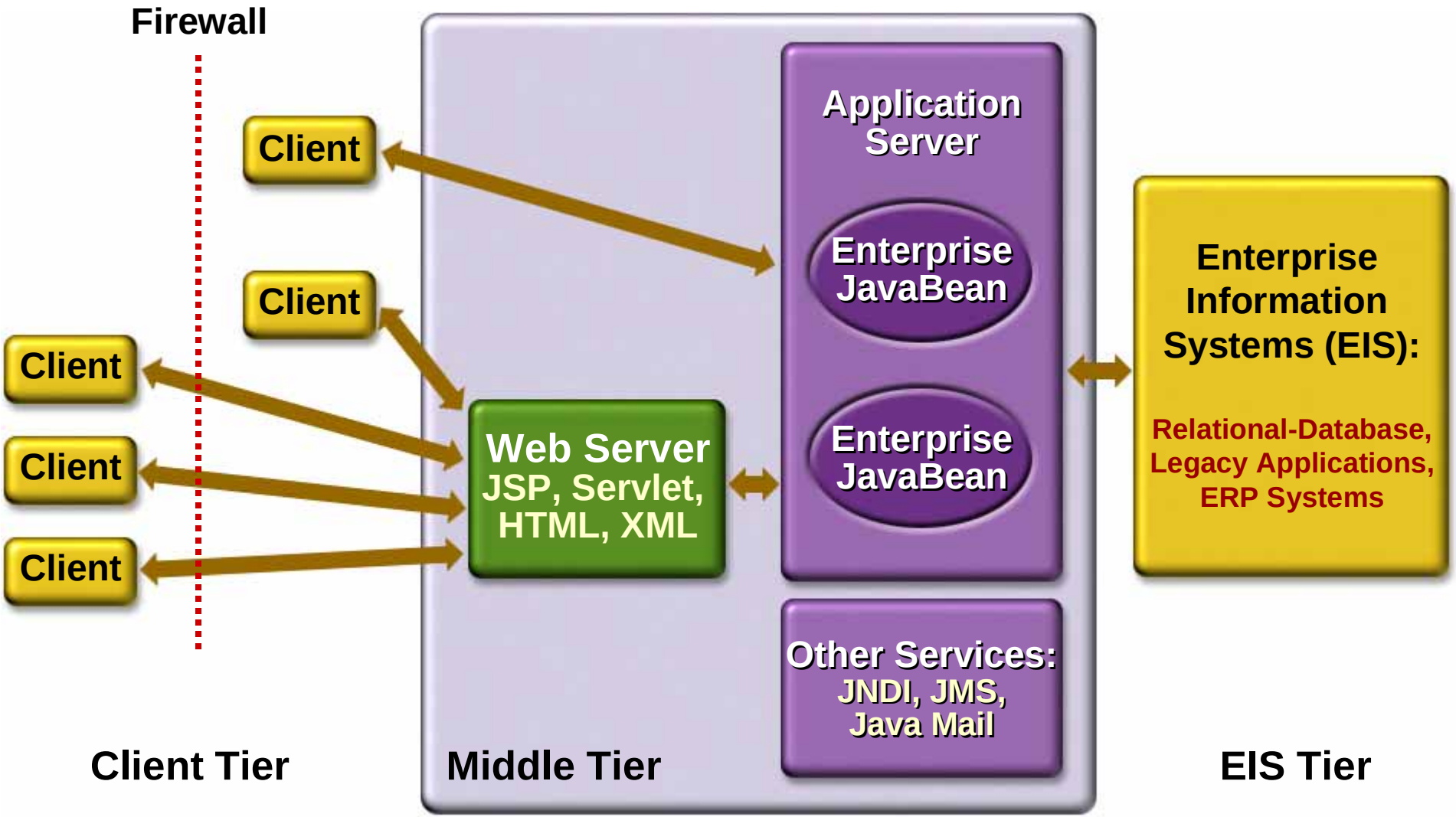
Java™ 2 Platform, Enterprise Edition
Application Programming Model (APM)

- **Organized set of design patterns, templates and architectural principles**
  - **Focus is on design of manageable, deployable and maintainable J2EE applications**
  - **Results in faster product delivery time to market of enterprise solutions**
- **Recommends how the J2EE specifications should be applied to application domains**

Some Key APM Questions

- Choose Servlets or JavaServer Pages™ technology (JSP)?
- Access EIS Resources via Enterprise JavaBeans™ technology (EJB) server or Directly from JSP?
- Session Beans or Entity Beans?
- Distributed transactions or local transactions?
- How to exchange Data with External Systems?

The J2EE Environment
Enabling End-to-end Solutions



Web Access

- Exposes application logic to web client(s) as coarse grained service(s):
  - Responsible for handling “user” input/application presentation
  - Named via URI
  - Modeled as HTML (or XML)
- Comprised of dynamic and static content
  - Java Server Pages for dynamic content:
    - Higher level of abstraction than Servlets
    - Easier to generate HTML (and XML in future)

“Other” Clients

- **EJB from another “application”:**
  - **RMI/IIOP now, JMS later**
- **Standalone “clients”:**
  - **CORBA client:**
    - **IIOP access direct to application logic**
  - **Java technology client:**
    - **RMI/IIOP or JMS access**
  - **Desktop productivity application:**
    - **MS desktop integration solution via plug-in, JRE and RMI/IIOP**

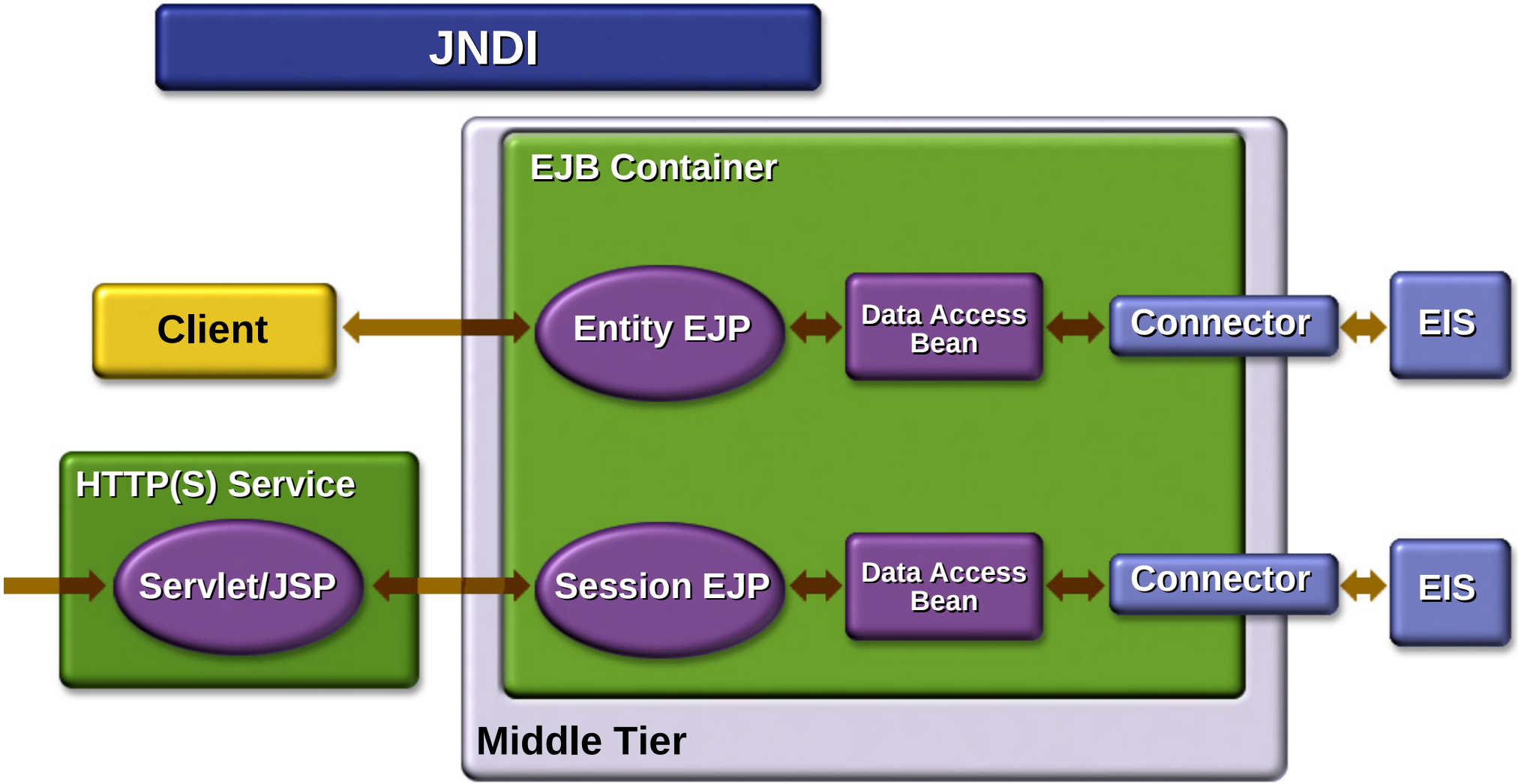
Application Logic

- **Modeled using EJBs:**
  - **`SessionBean` (stateless & stateful)**
  - **`EntityBean` (CMP or BMP)**
  - **Componentize:**
    - **Workflow, processes, business rules and entities…**
- **Fine grain imperative interface contract**
- **External resources are logically named internally**
  - **Customized via deployment descriptor**
  - **Resolved by deployer at deployment time**

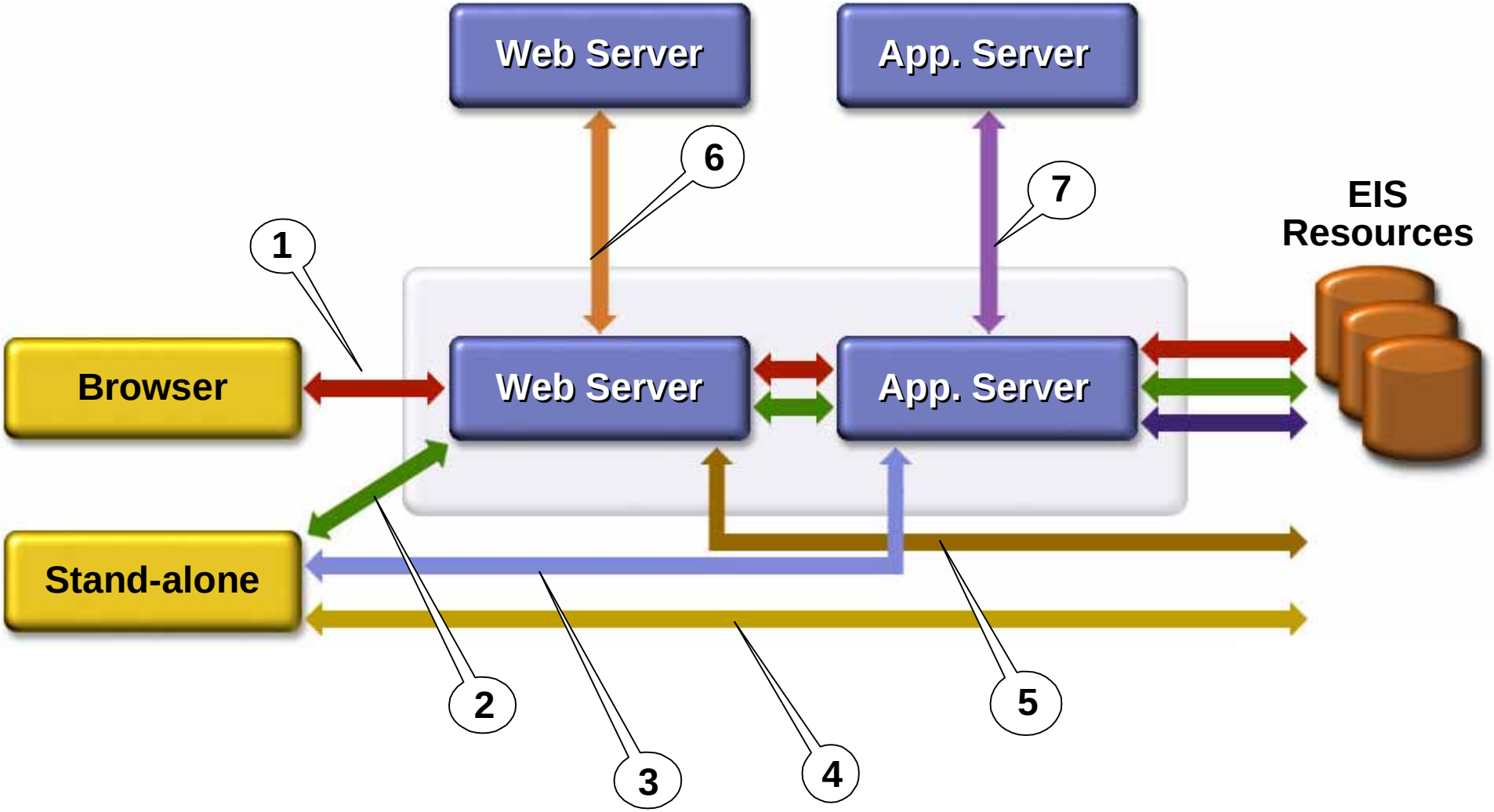
Enterprise Information Services (EIS)

- Integration is achieved by either:
  - Directly using platform API such as:
    - JDBC™
    - JMS
    - Java Naming and Directory Interface™ (JNDI)
  - Or via Connector(s):
    - Connector is:
      - Resource adapter, Access Bean, integration tool(s)
    - Ubiquitous Connector APIs for specific EIS products

EJB Components



Application Scenarios



Core Application Scenarios

1

- **n-tier Web Access**
  - HTML, XML, HTTP client
  - JSP/Servlets, RMI/IIOP
  - EJBs, JDBC (Connectors)

Sample App.

3

- **n-tier Intranet Access**
  - EJB Client, EJB Sever

5

- **2-tier Java Client**
  - JSP/Servlets and JDBC

7

- **B2B Enterprise Transactions**
  - EJBs, JMS and XML

Sample Application
High Level Requirements

- Must offer Web Presence
- Must be Robust and Scalable
- Must leverage existing EIS resources
- Must utilize core competencies of diverse development teams
- Must facilitate B2B transactions
- Get it done yesterday!

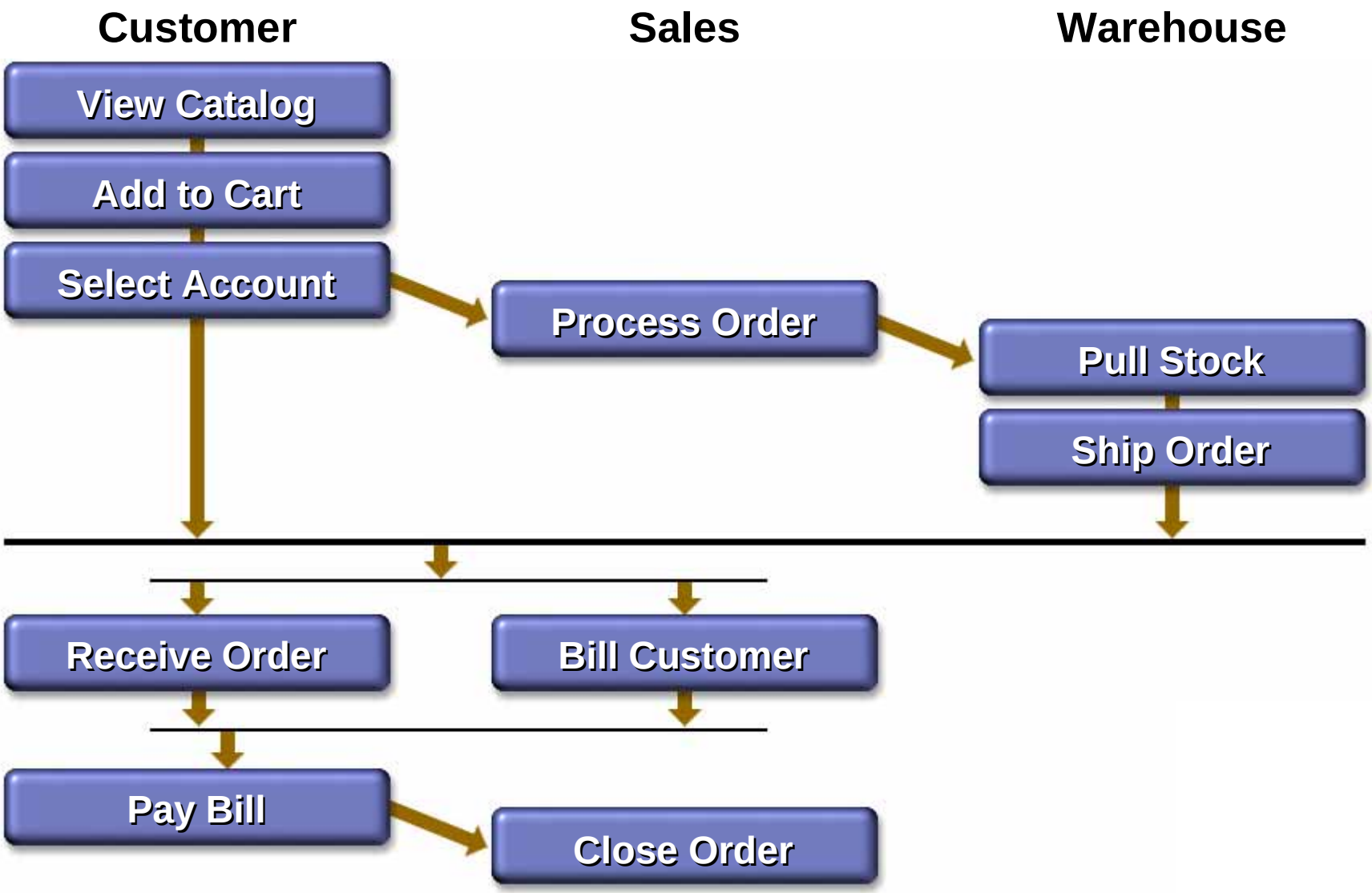
Sample Application
High Level Technical Decisions

- **Client tier**
  - Chose HTML Clients for End-User Web Access
  - Chose XML for selective Data externalization
- **Web Server tier**
  - Chose JSP/Servlets to be WS neutral
- **App Server tier**
  - Chose EJB's to be AS neutral
  - JDBC Data Access Beans to encapsulate EIS access
  - Chose to optionally support distributed transactions

SA—Scenario

- Web-based browsing of a product catalog
- Creation and maintenance of a shopping cart
- User account creation
- Placing orders
- Secure order processing
  - B2B transactions
  - Externalization of order data (expressed in XML)
  - Order confirmation using e-mail

SA—Activity Diagram



SA—DB Schema

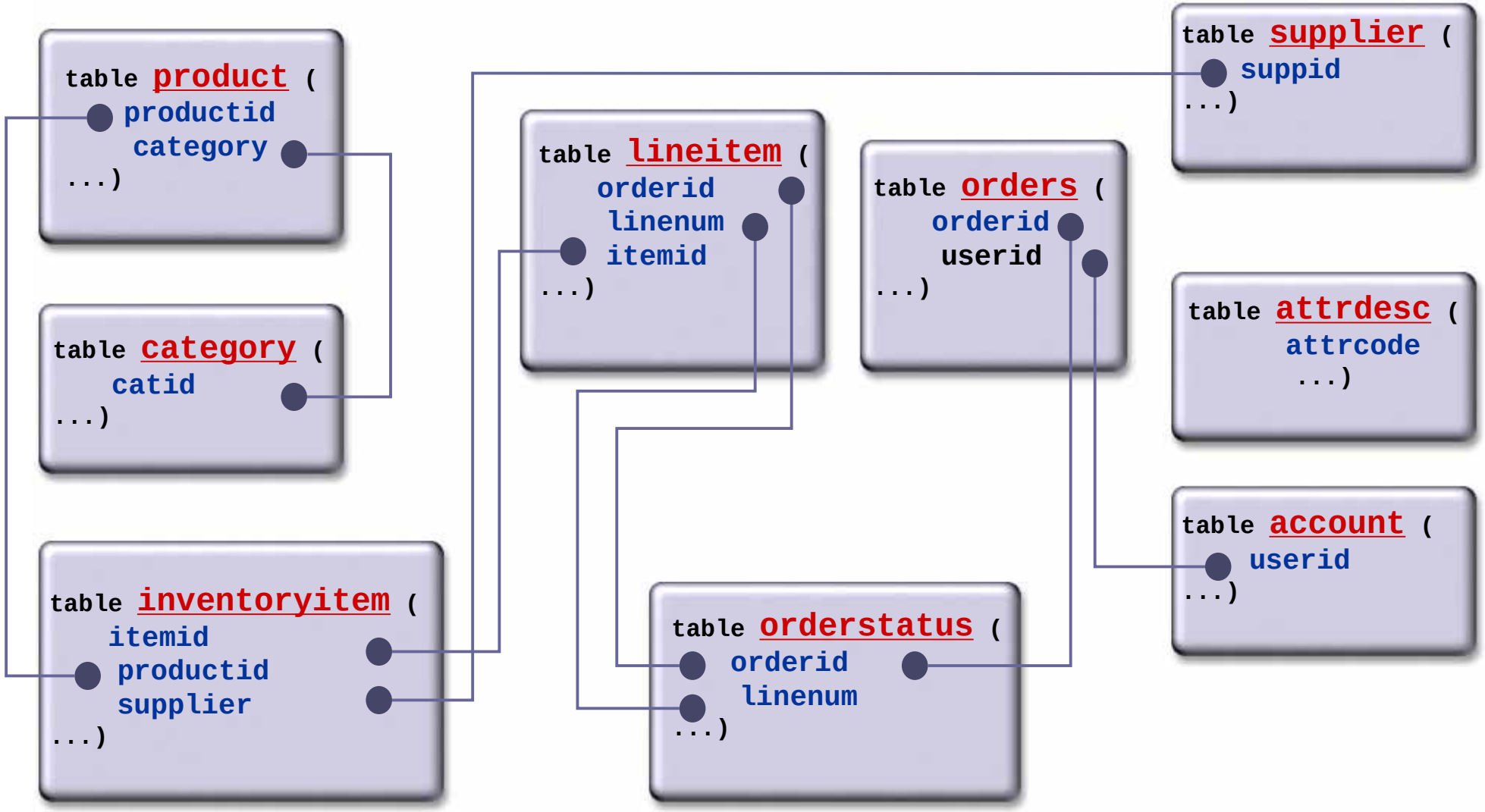
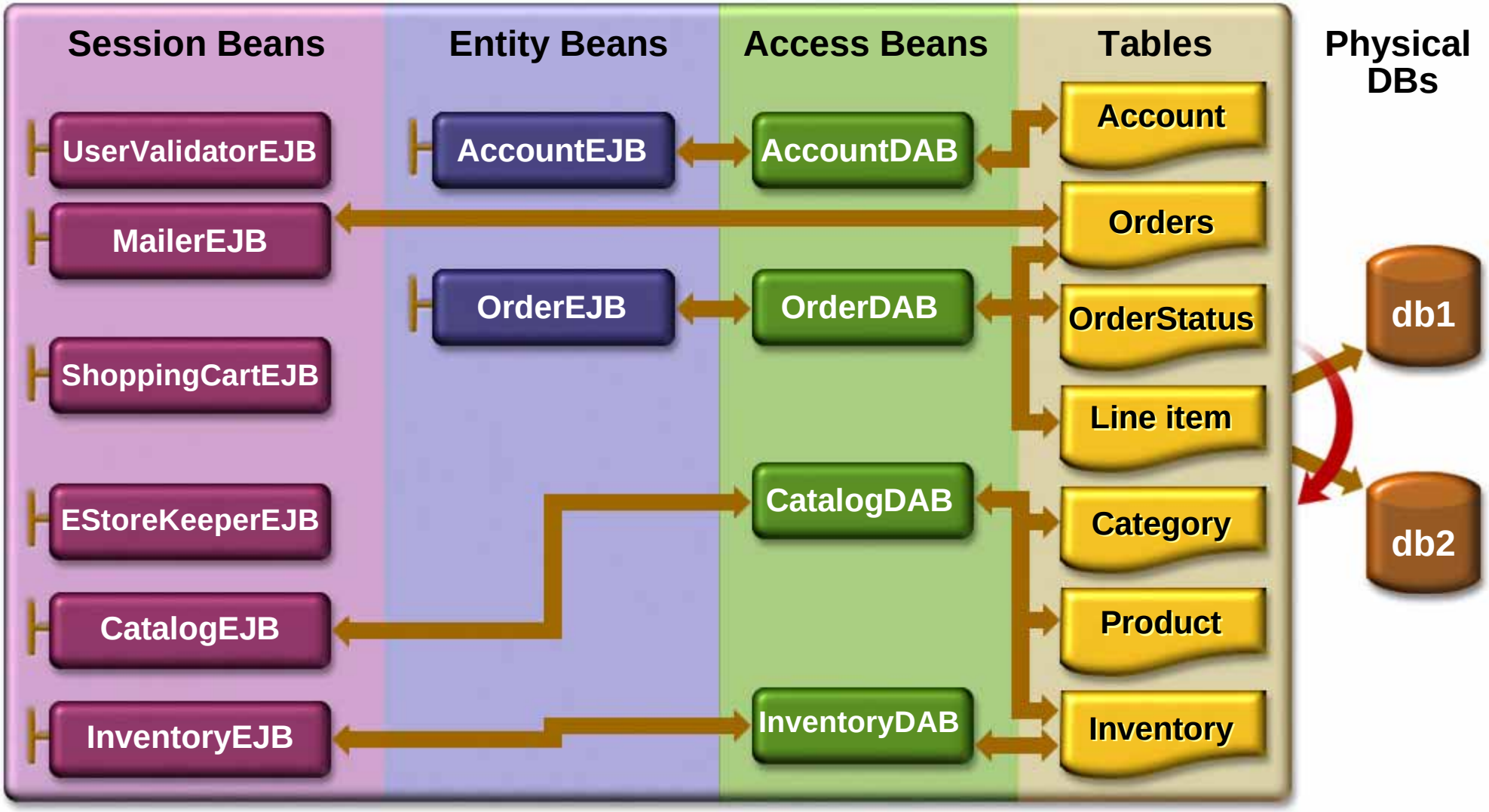


table product (
productid
category
...)
table category (
catid
...)
table inventoryitem (
itemid
productid
supplier
...)
table lineitem (
orderid
linenum
itemid
...)
table orderstatus (
orderid
linenum
...)
table orders (
orderid
userid
...)
table supplier (
suppid
...)
table attrdesc (
attrcode
...)
table account (
userid
...)

SA—Architecture



SA—JSP Usage

```
<html>
<jsp:request include="/banner.html"/>
<form action="validatenewuseraccount.jsp" name="login">
<table border ="0">
<tr><td>User ID:</td>
    <td align="left"><input type="text" size="15" name="user_name"></td>
</tr>
<tr><td>Password:</td>
    <td align="left"><input type="password" size="15" name =
    "password"></td>
</tr>
.....
</table>
</html>
```

SA—JSP Usage

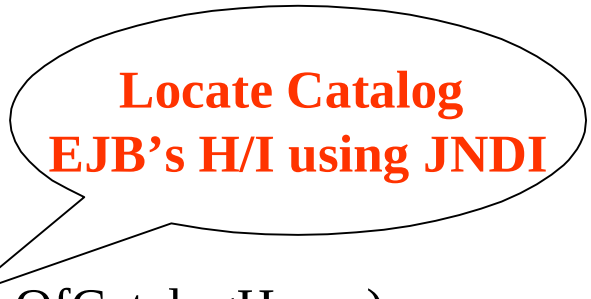
```
<html>

...

<% if (formValid){ %>

<jsp:useBean id="useraccount"
    class="com.sunw.estore.account.jspbeans.UserAccountJSPBean"
    scope="session" />

<jsp:setProperty name="useraccount" property="init" value="<%= session %>"
    />

<jsp:setProperty name="useraccount" property="billingLastName"
    param="last_name" />

...

<% } else { %>

...

<% } %>

</html>
```

SA—Controller Interface

```
public interface EStorekeeper extends EJBObject {
    public Catalog          getCatalog()          throws RemoteException;
    public ShoppingCart     getShoppingCart()     throws RemoteException;
    public Credential       getCredential()       throws RemoteException;
    public Account          getAccount()          throws RemoteException;
    public Enumeration      getOrders()           throws RemoteException,
                                                         FinderException;
    public void handleEvent(EStoreEvent se) throws RemoteException;
}
```

SA—Controller Implementation

```
public class EStorekeeperEJB implements SessionBean {
    public void ejbCreate() throws RemoteException {
        sm = new StateMachine(this);
    }
    public Catalog getCatalog() throws RemoteException {
        if (catalog == null) {
            try {
                Context initial = new InitialContext();
                Object objref = initial.lookup (EJBUtil.jndiNameOfCatalogHome);
                CatalogHome catalogHome = (CatalogHome)
                    PortableRemoteObject.narrow(objref, CatalogHome.class);
                catalog = catalogHome.create();
            } catch (NamingException ne) {
                ...
        }
        return catalog;
    }
```



Application Model

- Makes Recommendations across all tiers
  - Client
  - Middle-tier (includes Web server and EJB)
  - Data access
  - Messaging
  - External application integration
  - Communication protocols
  - Deployment file formats
- Results in Faster product delivery

J2EE Resources

- **White paper:**
  - **http://java.sun.com/j2ee**