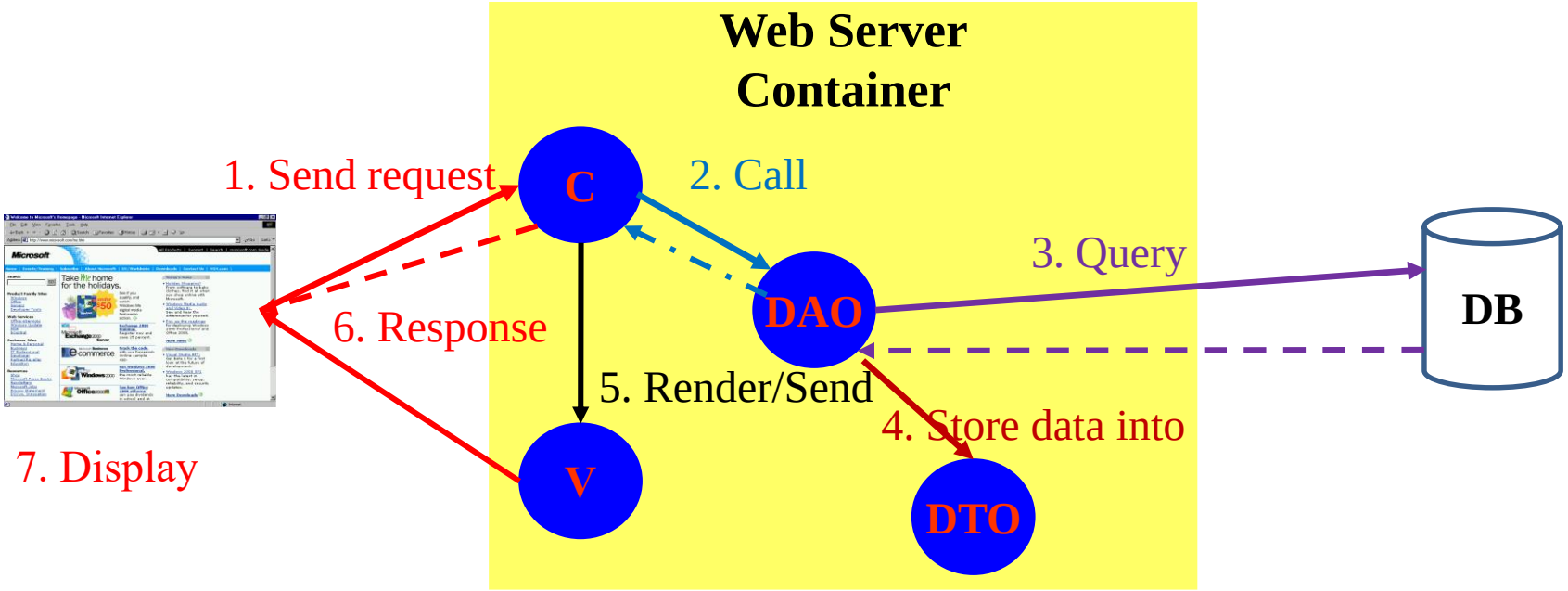
Introduction to Enterprise Java Beans

Introduction to EJBs
EJB Ecosystem
Enterprise Beans
What Constitutes an EJB?
JNDI

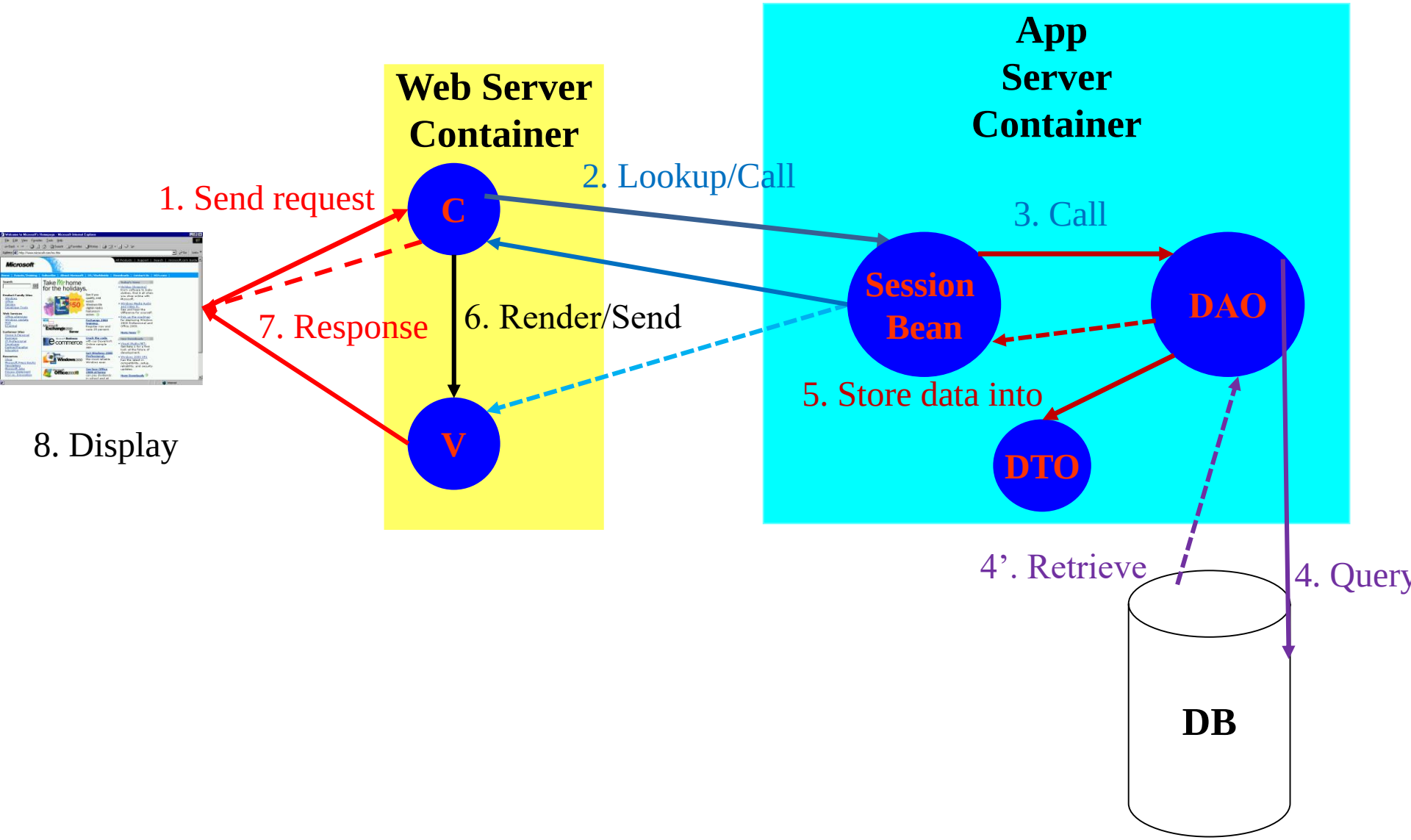
Review

- **JSTL**
  - Core, Functions, Sql
- **Custom Tag Libraries**
  - Classic Tags
  - Simple Tags
  - Tag Files
  - Tag components
    - Create tld, create tag handler class, import taglib in jsp file
  - Implementation
    - Tag without attributes, tag with attributes, empty tag, tag with body, iterative tag

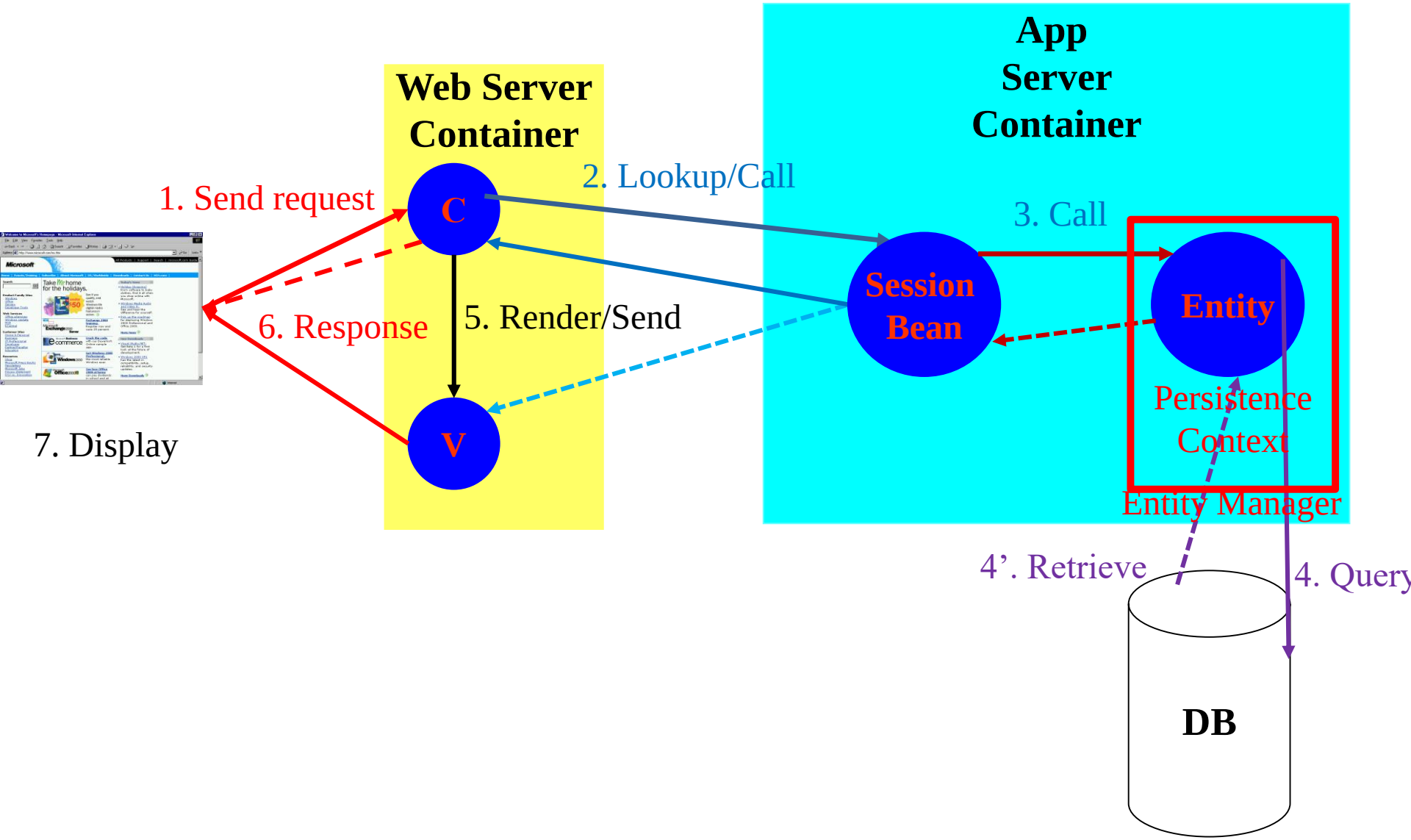
Review


Web Server Container
1. Send request
C
2. Call
3. Query
DAO
DB
6. Response
5. Render/Send
4. Store data into
7. Display
V
DTO

Overview


Web Server Container
App Server Container
1. Send request
2. Lookup/Call
3. Call
C
Session Bean
DAO
6. Render/Send
7. Response
5. Store data into
V
DTO
8. Display
4'. Retrieve
4. Query
DB

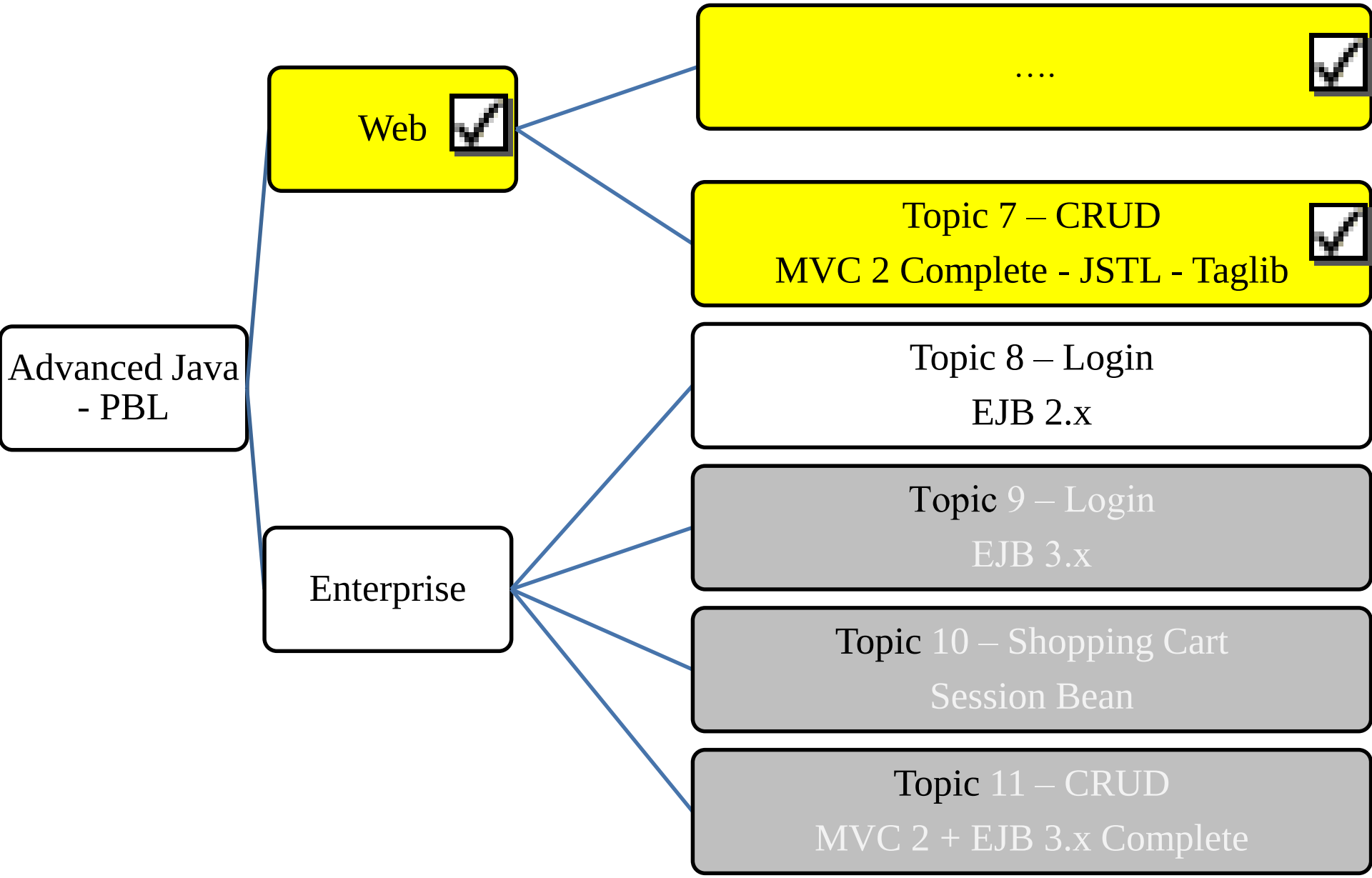
Overview


Web Server Container
App Server Container
1. Send request
2. Lookup/Call
3. Call
C
V
Session Bean
Entity
Persistence Context
Entity Manager
5. Render/Send
6. Response
7. Display
4'. Retrieve
4. Query
DB

Objectives

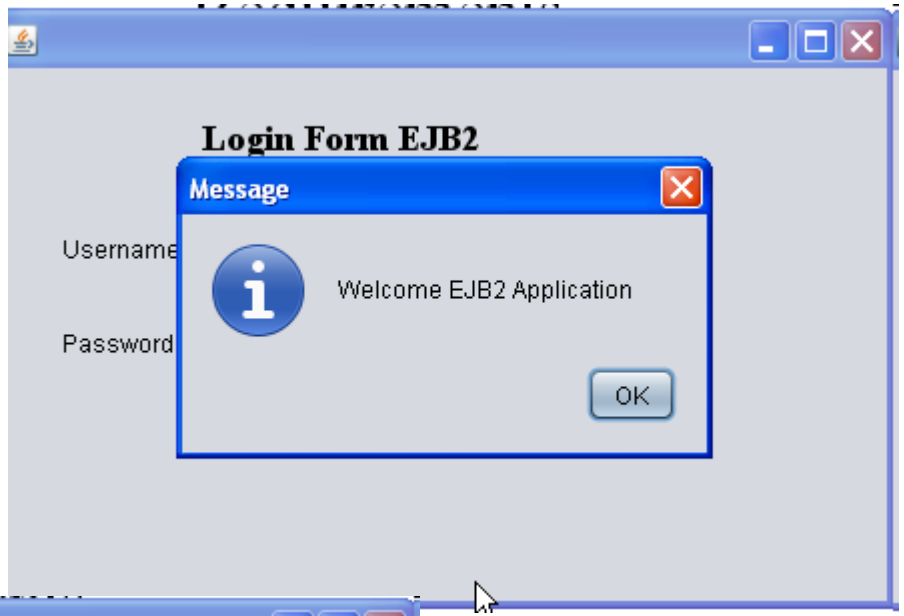
- **How to build the simple enterprise application – Login – using EJB 2.0 with GUI as Swing or web?**
  - Logical Architecture of EJB
  - Components of EJB
  - Accessing EJB from the Client/Web Side
  - File and Directory Structure of Enterprise Applications

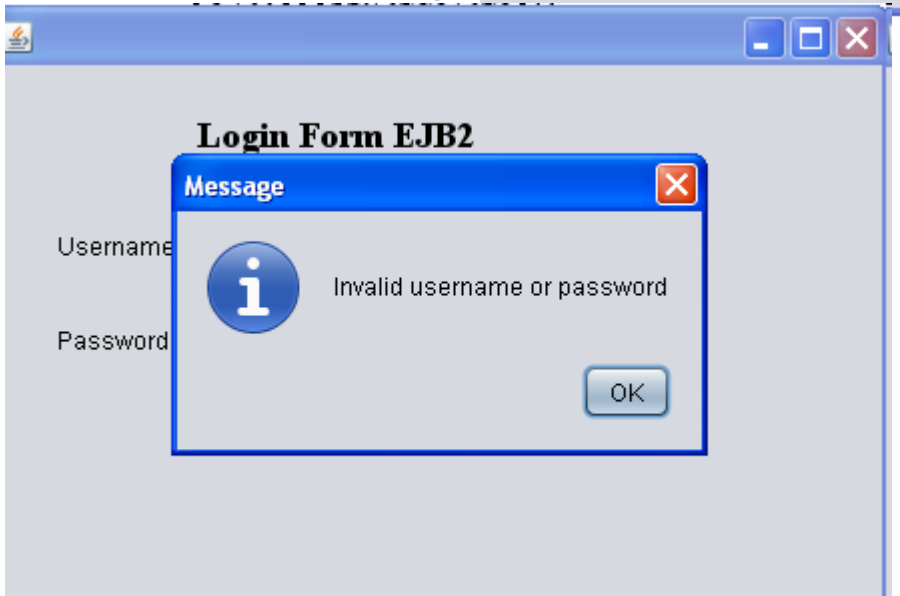
Objectives


Advanced Java - PBL
Web
Enterprise
….
Topic 7 – CRUD
MVC 2 Complete - JSTL - Taglib
Topic 8 – Login
EJB 2.x
Topic 9 – Login
EJB 3.x
Topic 10 – Shopping Cart
Session Bean
Topic 11 – CRUD
MVC 2 + EJB 3.x Complete

Build the simple enterprise application

Requirements





Build the simple enterprise application

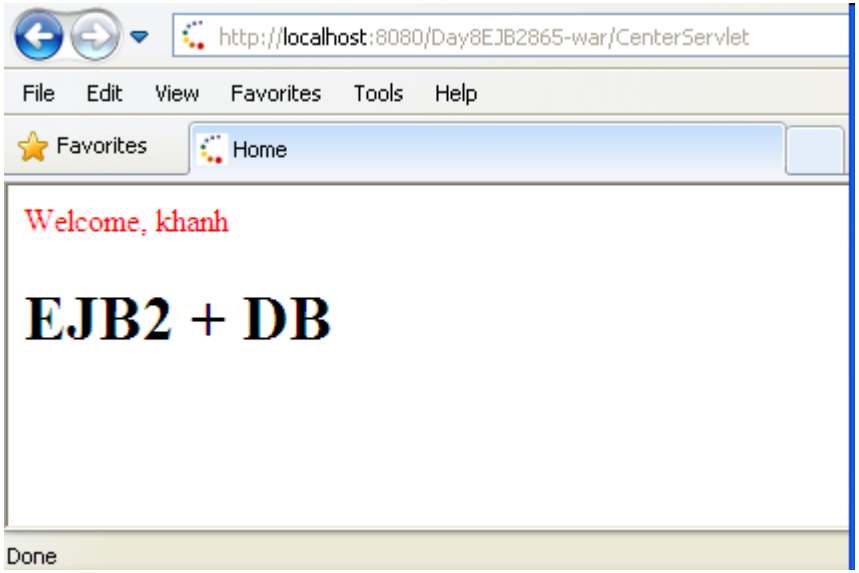
Expectation

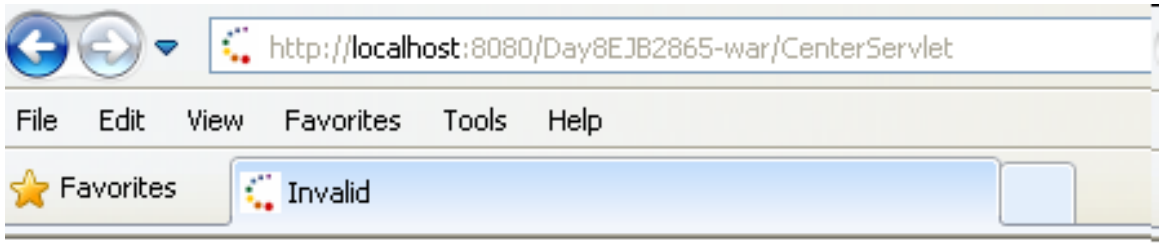
- Day8EJB2865-ejb
  - Source Packages
    - sample.registration
      - RegistrationDAO.java
    - sample.session
      - RegistrationSessionBean.java
      - RegistrationSessionBeanLocal.java
      - RegistrationSessionBeanLocalHome.java
      - RegistrationSessionBeanRemote.java
      - RegistrationSessionBeanRemoteHome.java
    - sample.utils
      - DBUtils.java
  - Libraries
  - Enterprise Beans
    - RegistrationSessionBeanSB
      - Remote Methods
      - Local Methods
      - Bean Methods
  - Configuration Files
    - MANIFEST.MF
    - ejb-jar.xml
    - jboss.xml
  - Server Resources
    - jboss-ds.xml

- Day8Java865
  - Source Packages
    - sample.app
      - LoginForm.java
  - Libraries
    - Day8EJB2865-ejb - dist/Day8EJB2865-ejb.jar
    - jboss-ejb-api_3.1_spec.jar
    - jbossall-client.jar
    - jnp-client.jar
    - JDK 1.7 (Default)

Build the simple enterprise application

Requirements





Invalid username or password

Build the simple enterprise application

Expectation

- Day8EJB2865
  - Java EE Modules
    - Day8EJB2865-war.war
    - Day8EJB2865-ejb.jar
  - Configuration Files
    - MANIFEST.MF
    - application.xml
    - jboss-app.xml
  - Server Resources
- Day8EJB2865-war
  - Web Pages
    - WEB-INF
    - calculator.html
    - index.jsp
    - invalid.html
    - welcome.jsp
  - Source Packages
    - sample.servlet
      - CenterServlet.java
      - LoginServlet.java
  - Libraries
    - Day8EJB2865-ejb - dist/Day8EJB2865-ejb.jar
    - JDK 1.7 (Default)
    - JBoss6.1.0Final
  - Configuration Files
    - MANIFEST.MF
    - jboss-web.xml
    - web.xml
  - Server Resources
- Day8EJB2865-ejb
  - Source Packages
    - sample.registration
      - RegistrationDAO.java
    - sample.session
      - RegistrationSessionBean.java
      - RegistrationSessionBeanLocal.java
      - RegistrationSessionBeanLocalHome.java
      - RegistrationSessionBeanRemote.java
      - RegistrationSessionBeanRemoteHome.java
    - sample.utils
      - DBUtils.java
  - Libraries
  - Enterprise Beans
    - RegistrationSessionBeanSB
      - Remote Methods
      - Local Methods
      - Bean Methods
  - Configuration Files
    - MANIFEST.MF
    - ejb-jar.xml
    - jboss.xml
  - Server Resources
    - jboss-ds.xml

Introduction to EJBs

Component

- Is a **piece of code** that **exhibits** the **behavior** of a **concept related** to the **real world**
- Can be **reused** in different applications
- **Main requirement** of a component is that it should **encapsulate** the **behavior** of an **application**
  - **Provides** a **set of services or functions**, such that it can easily interact with other applications or components
  - The **users are not aware** of the **internal processes** of the components in an application **but are aware** of **what they need** to **pass** in as **input and what** to **expect** as **output**
- **Component framework concept evolved** to **support development** and **deployment** of enterprise applications
- Components
  - Are **building blocks** of an application
  - Are **distributed over various tiers**

Introduction to EJBs

Component Architecture

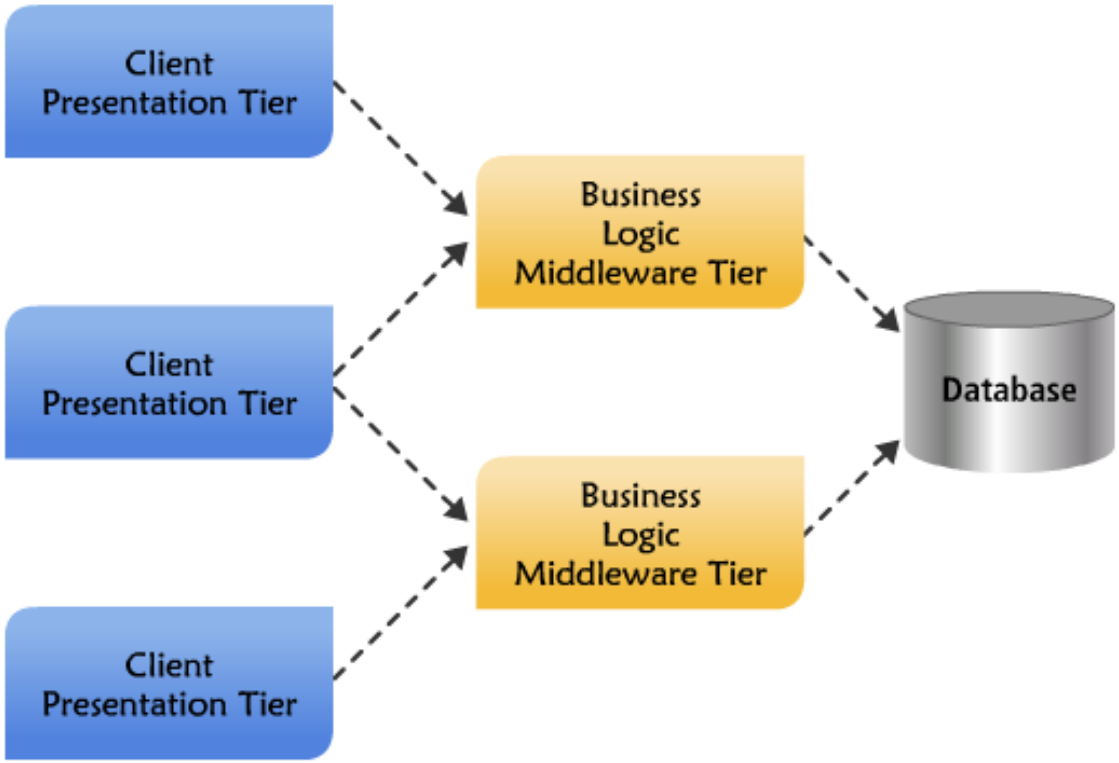
- Flexible, Portability and Reusable
- Consists mainly of **Web components** (JSP, Servlet, …), **Business components** (EJB), and **Service components** (Mail, JDBC, JMS …).
- An enterprise application is usually **composed of a three-layer architecture**
  - **Presentation** Layer (Web Component, GUI Component, Client console)
    - Is responsible for **rendering the graphical user interface** and **handling user input**
    - **Passes** down **each request for application functionality** to the business logic layer

Introduction to EJBs
Component Architecture

- An enterprise application is usually composed of a three-layer architecture
  - **Business** Layer (Business Component)
    - Is the core of the application
    - Comprises business logic and business objects
    - **Business** logic
      - **Comprises business rules or methods** using which **specific business functions** can be managed
      - **Refers** to the **workflow** or the **ordered task of passing** or **retrieving data** from one software sub-system to another
    - **Business** objects
      - Are the **set** of **objects** and the **relationships** between them
      - **Encapsulate** both the data & business behavior associated with the entity that it represents
      - Have the **required features**: reusability, access control, remote access, multi-user, highly available, state maintenance, transactional, and shared data
      - Are **stored** to **DB** or **storage** by **using an abstract layer** – **persistence layer** that lies over the DB layer and interacts with DB

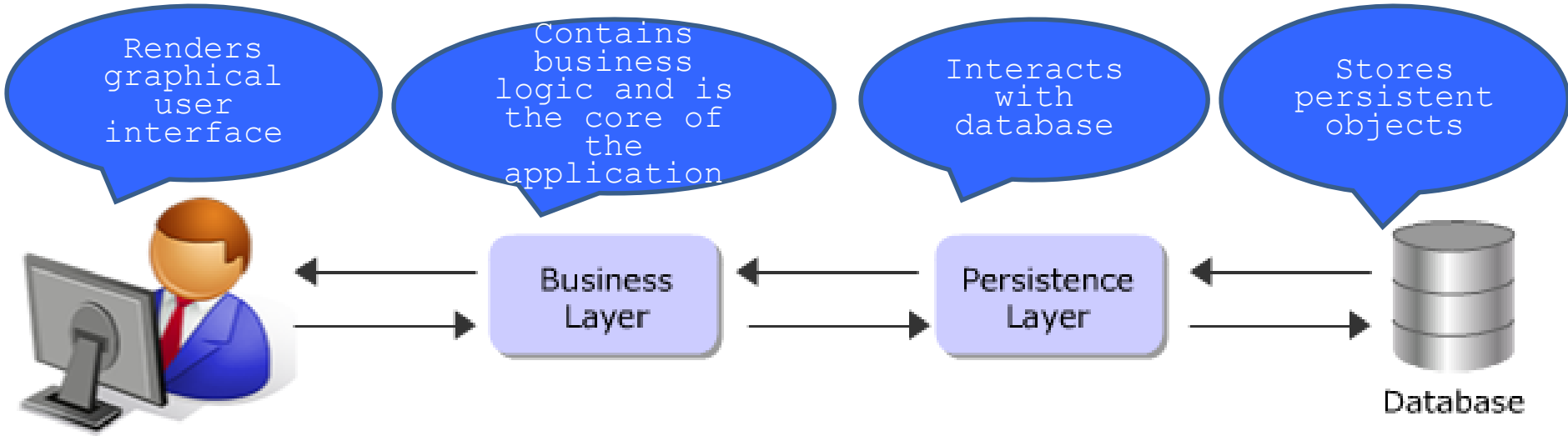
Introduction to EJBs
Component Architecture

- An enterprise application is usually composed of a three-layer architecture
  - **Data** Layer
    - Consists of relational database management systems (**RDBMS**) such as SQL Server, Oracle, DB2, … **for storing persistent objects**



Introduction to EJBs

Distributed Object Architecture



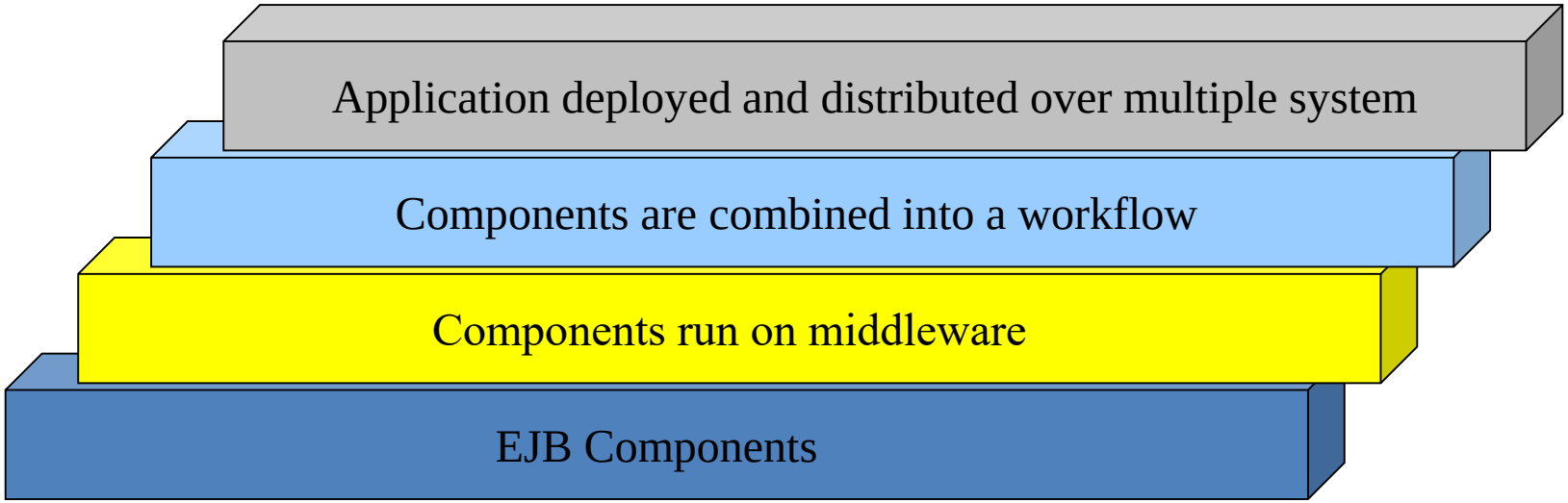
Introduction to EJBs

EJB Technology

- Building application software is complexity
  - The software **can process multi-data**
  - The software **is available online.**
  - The developer **worries** about the security, transaction, scalability, concurrency, resource management, persistent, error handling, and **many more system level problems.**
- Software assurance and performance are affected because the developer **can not concentrate fully on the developing the business logic (from implementation logic).**
- EJB was developed so that it would:
  - Specialize in **handling the business logic** of an application
  - Be **robust**
  - **Be secure** so that it cannot be tampered.
  - EJB **provides a component to create middleware** which is deployed on Application Server (3 tiers architecture).
  - EJB Component has been designed to **encapsulate business logic**.

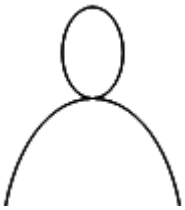
EJB EcoSystem

Stages in Developing Business Solution



EJB EcoSystem

Parties involved in EJB Development

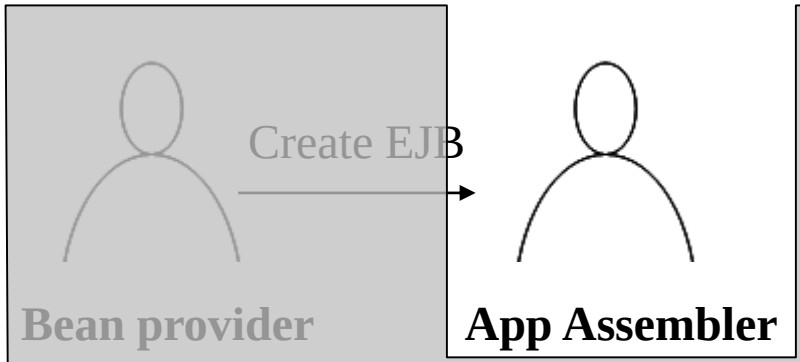


Bean provider

- Provide the components to **solve business problem** (that are packaged them to the ejb-jar file)
- **Reusable** components
- **Assemble** other components into application.
- **Distribution**

EJB EcoSystem

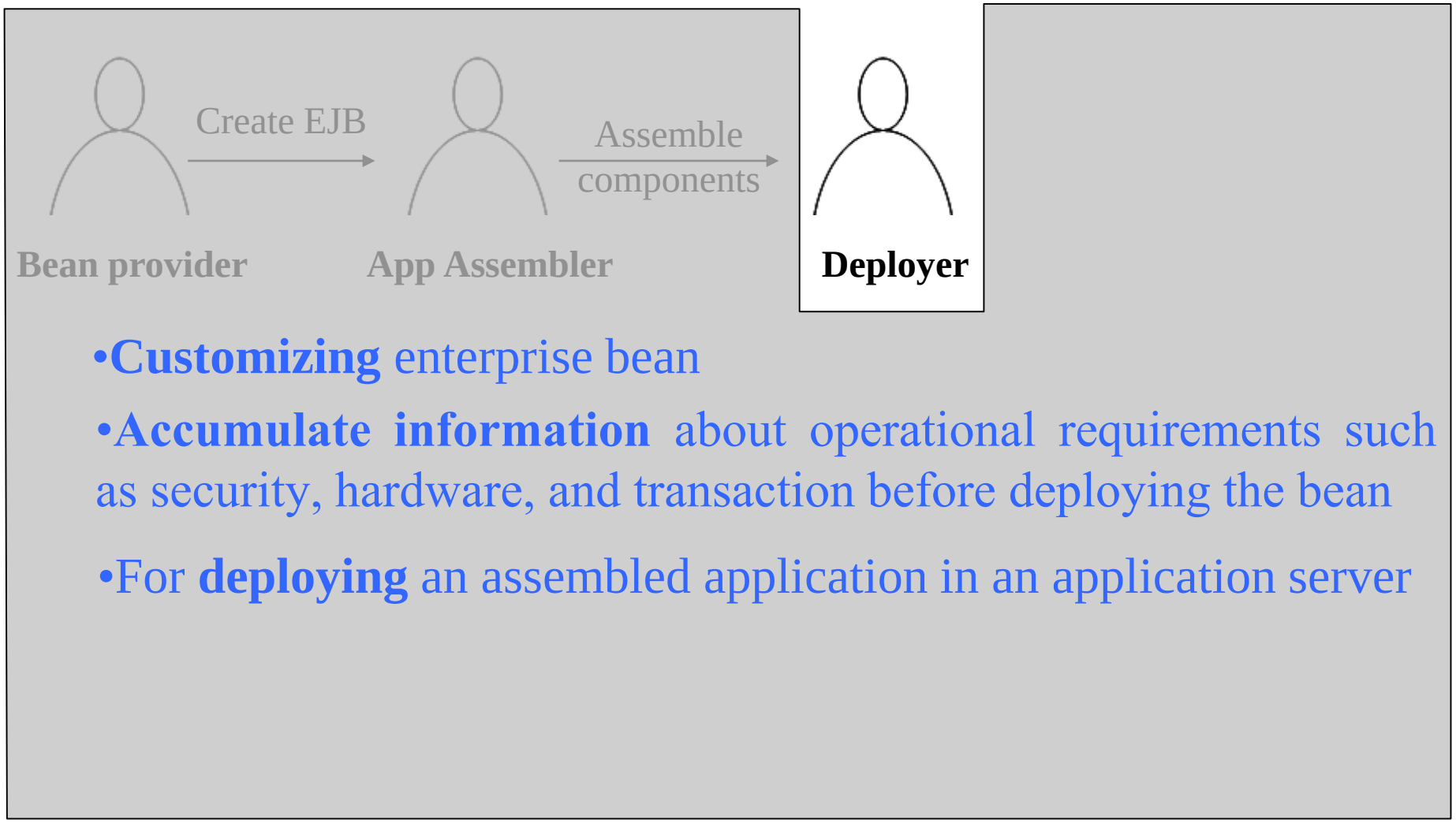
Parties involved in EJB Development



- For **assembling** different EJB Components in order to build a complete application
- **Analyzing** a business problem and **assembling** EJB components **accordingly to solve the problem**
- **Building** new EJB components
- **Writing the integration code** required to associate the EJB components build by different bean providers

EJB EcoSystem

Parties involved in EJB Development

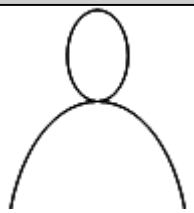


- **Customizing** enterprise bean
- **Accumulate information** about operational requirements such as security, hardware, and transaction before deploying the bean
- For **deploying** an assembled application in an application server

EJB EcoSystem

Parties involved in EJB Development

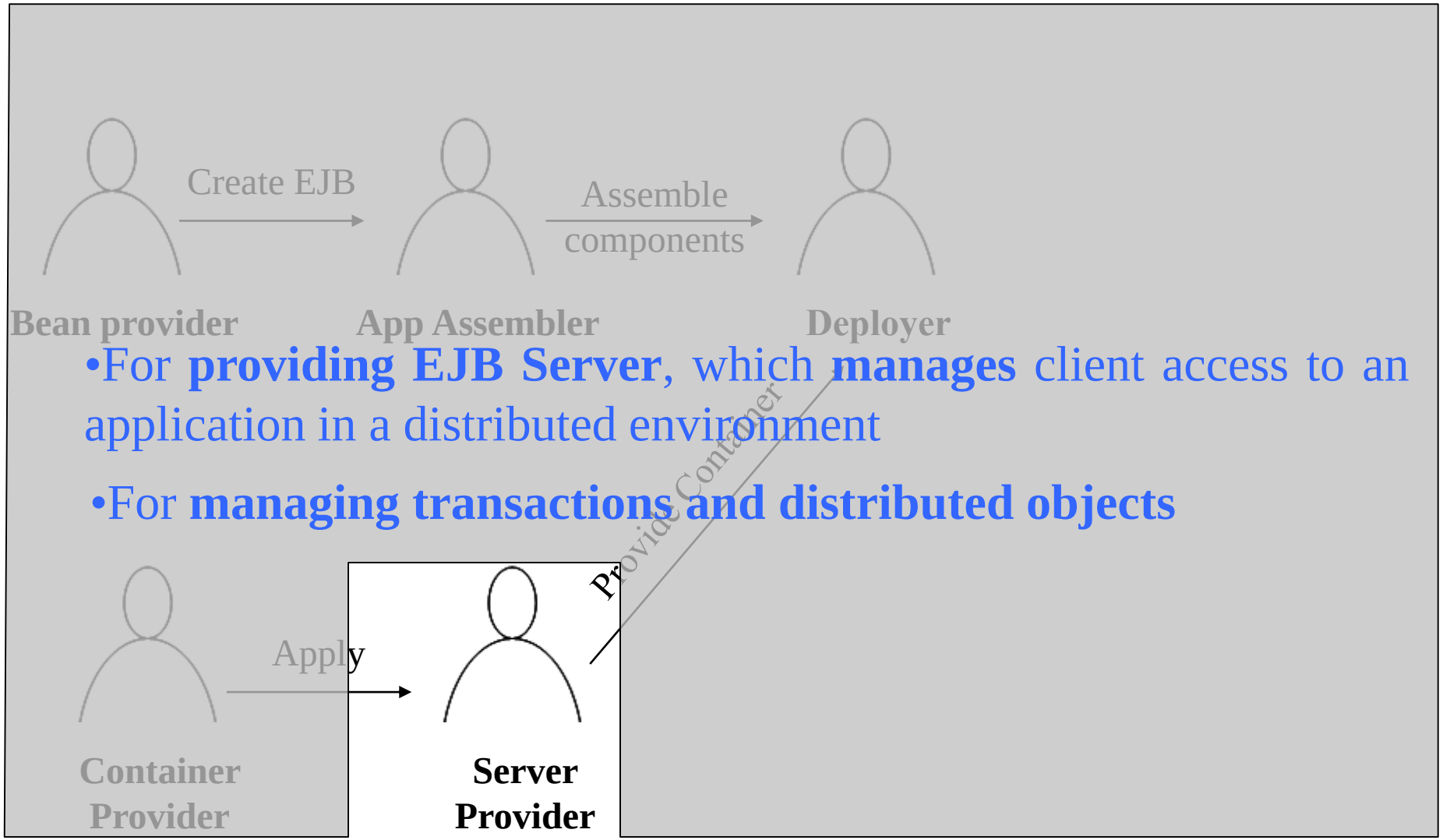
- **Providing the deployment tools** that are required by the deployer for deploying EJB components
- **Providing run-time support for beans** that are deployed on EJB Server



Container Provider

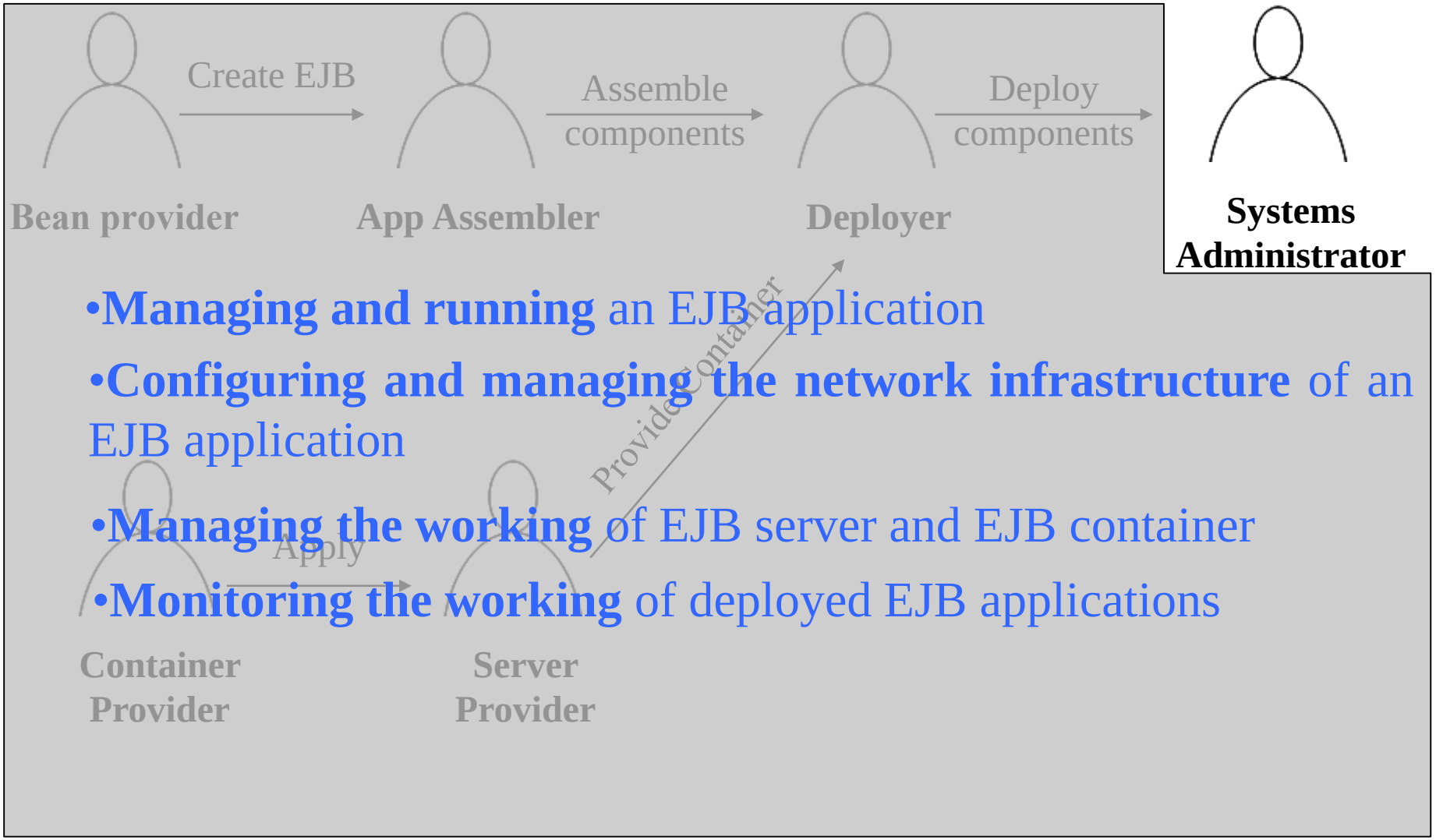
EJB EcoSystem

Parties involved in EJB Development



EJB EcoSystem

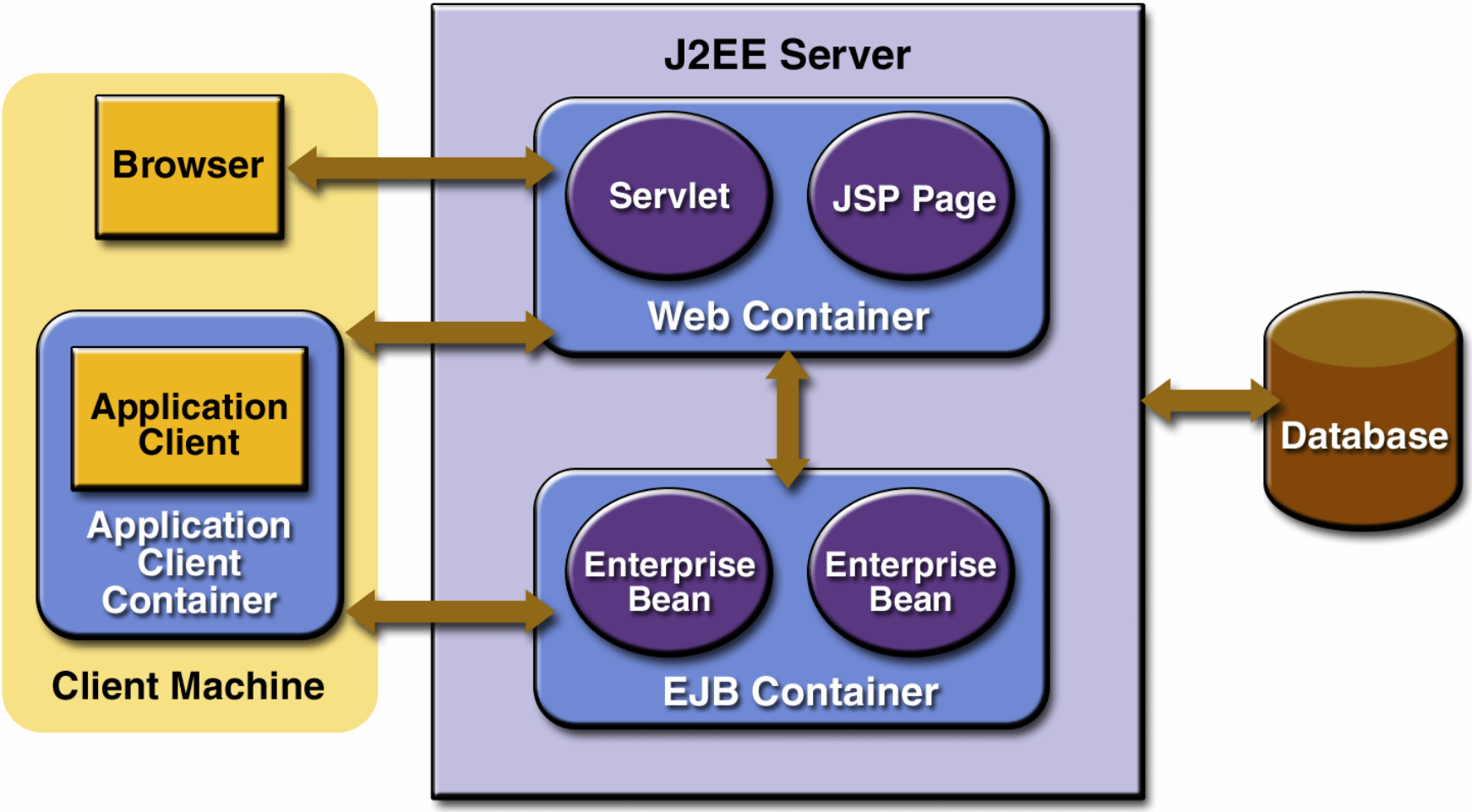
Parties involved in EJB Development



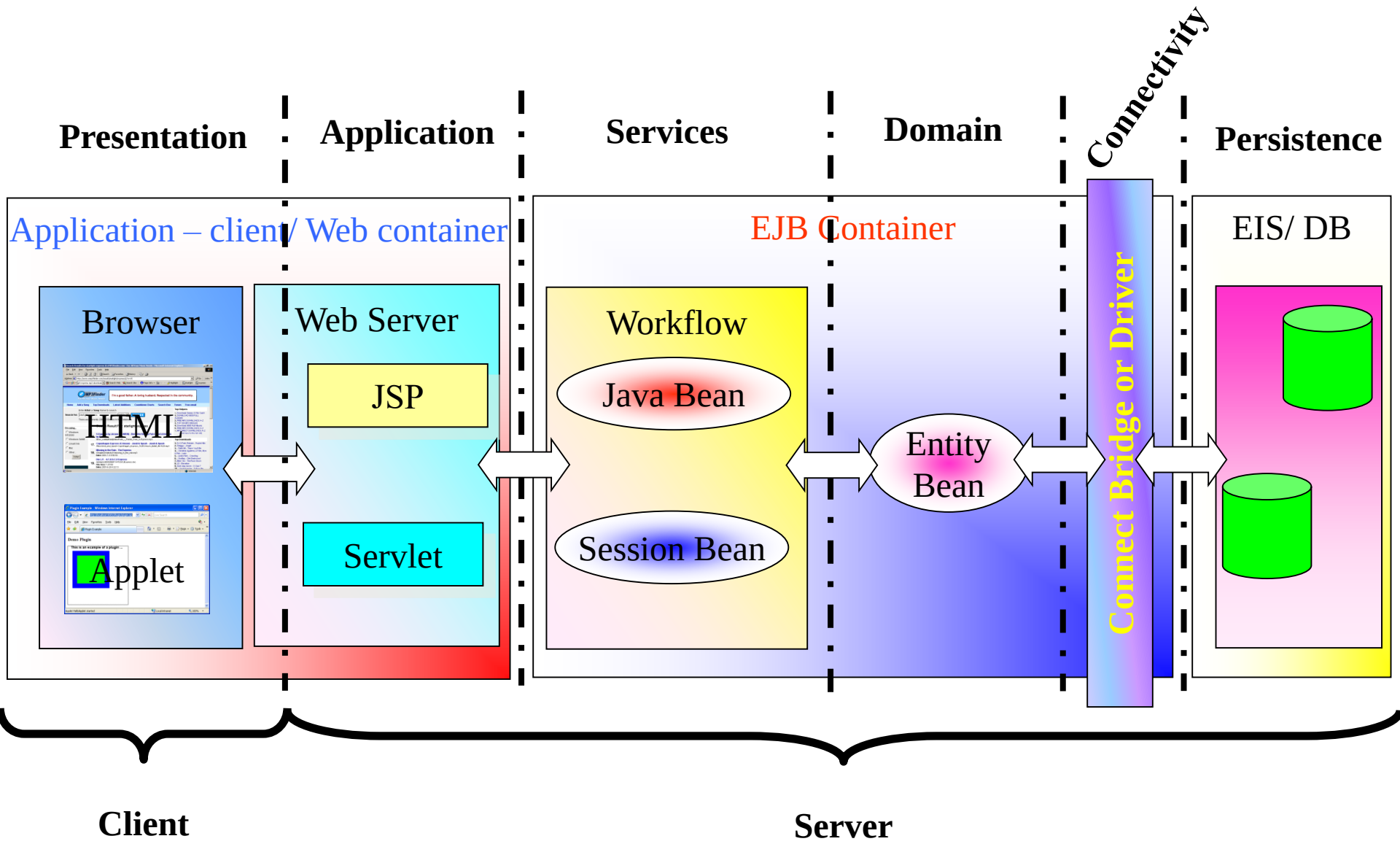
- **Managing and running** an EJB application
- **Configuring and managing the network infrastructure** of an EJB application
- **Managing the working** of EJB server and EJB container
- **Monitoring the working** of deployed EJB applications

Enterprise Java Beans

EJB in J2EE Architecture

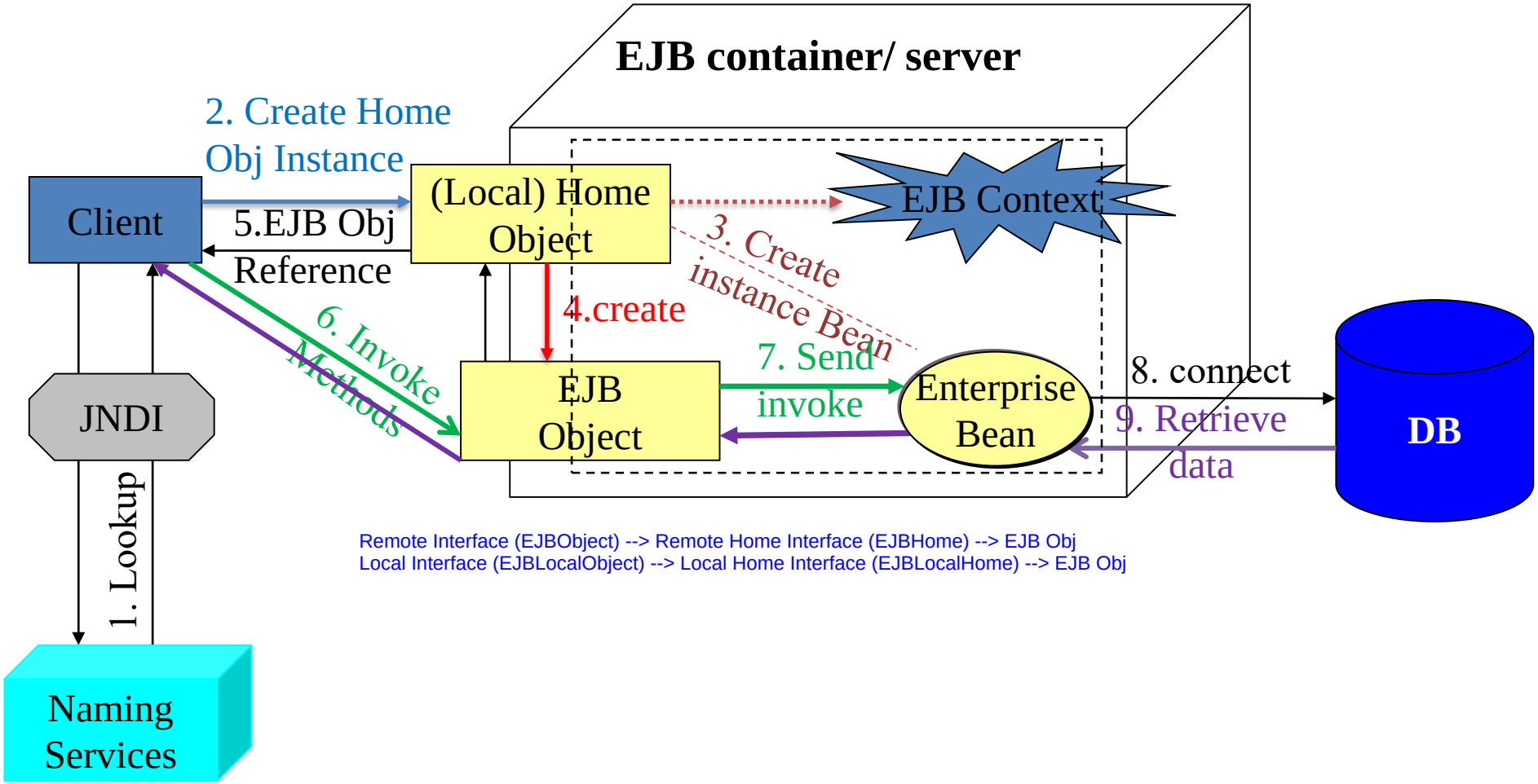


J2EE Architecture



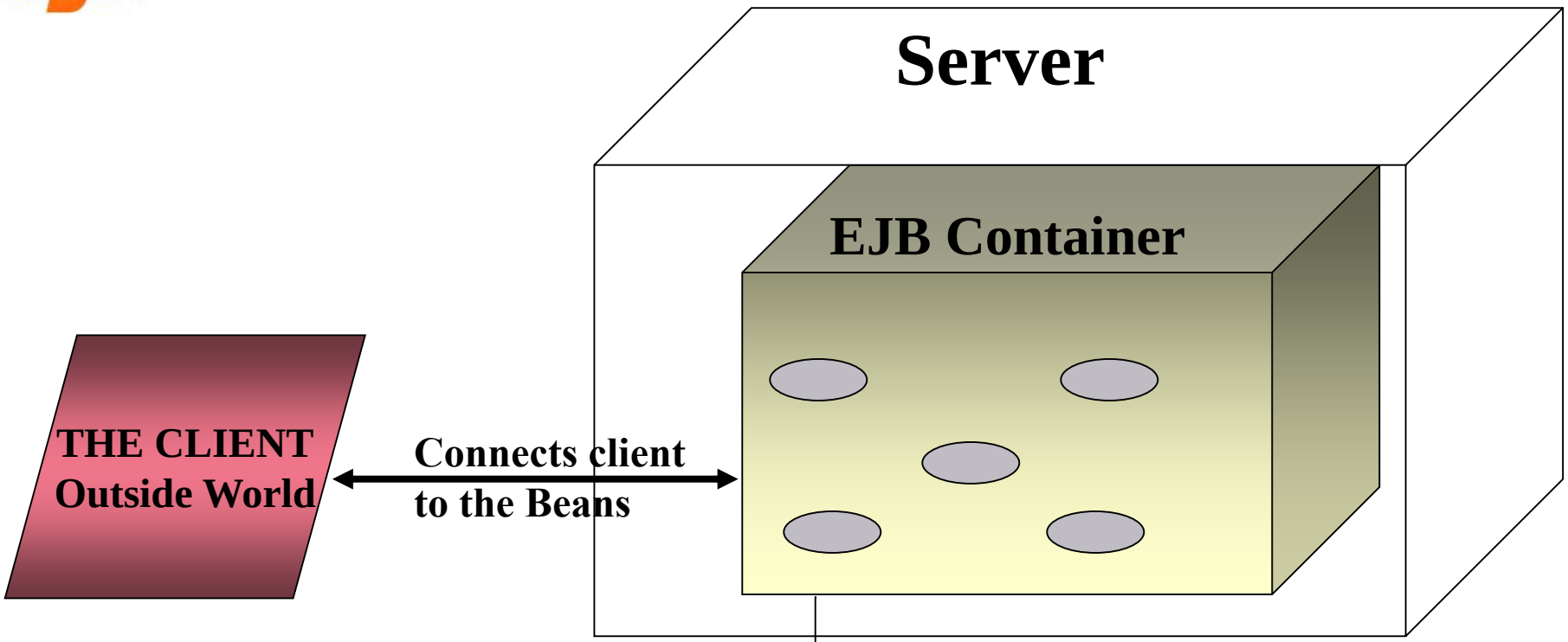
Enterprise Java Beans

Logical Architecture of EJB



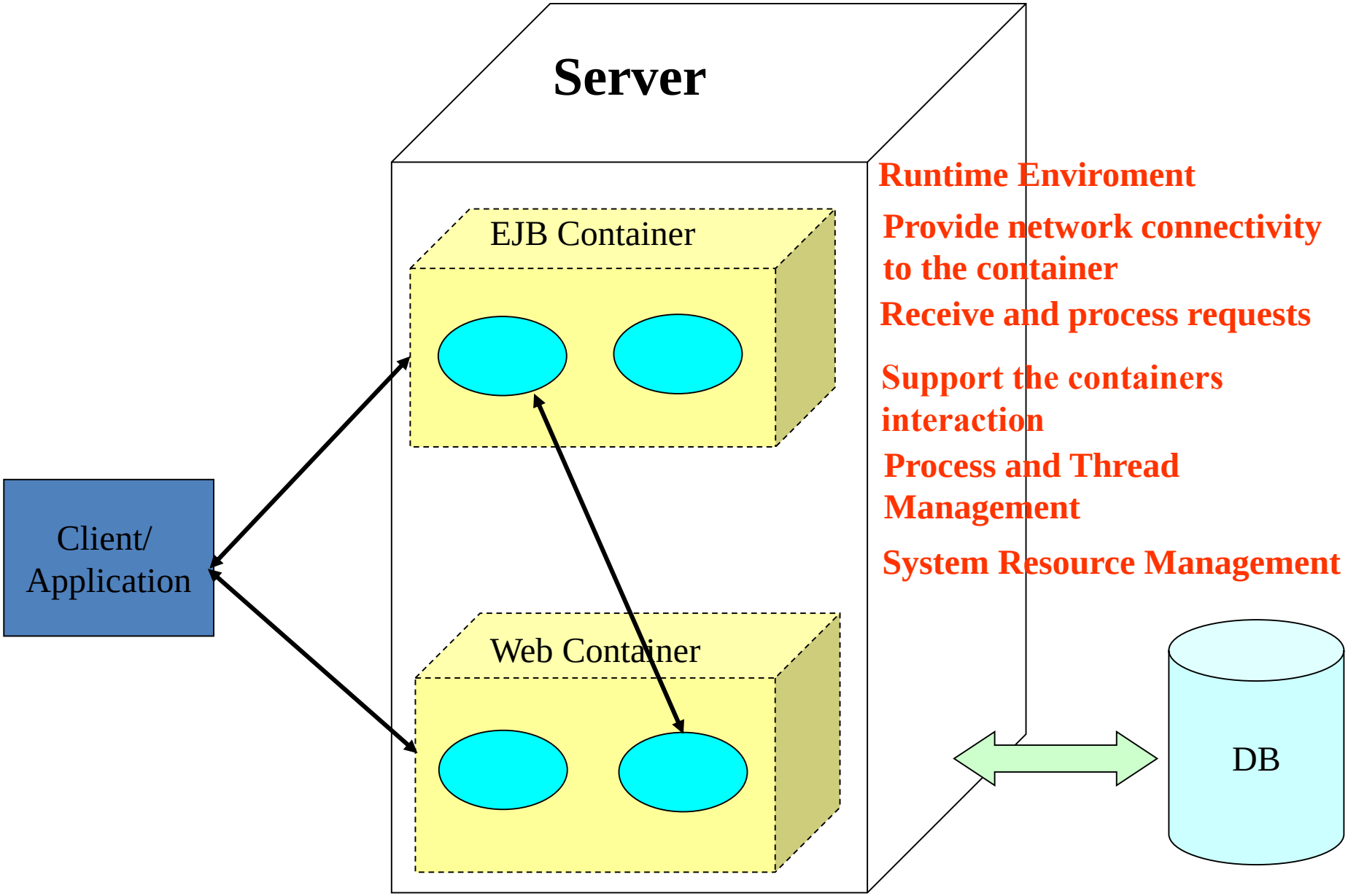
03 tiers Architecture

EJB Container



- Buffer: contain bean
- Forward Request
- Services to bean
- Manage life cycle of bean and keep track
- Run time enviroment
- Functions from behind the scenes
- Never directly codes
- Pool beans

EJB Server



Runtime Enviroment

Provide network connectivity to the container

Receive and process requests

Support the containers interaction

Process and Thread Management

System Resource Management

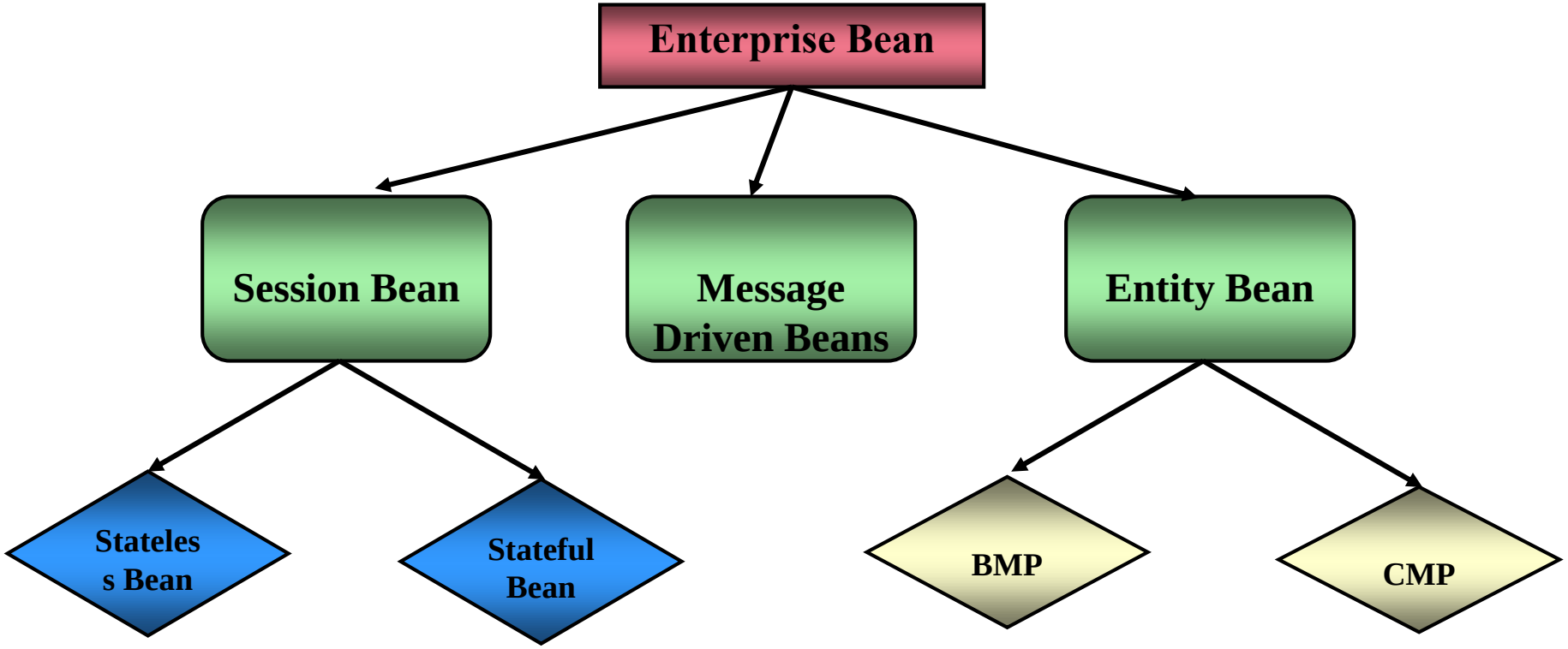
Enterprise Java Beans

Services provided by the Container and Server

- Transaction
- Security
- Persistence
- Support for Management of multiple instances
- Remote Accessibility
- Location transparency

Enterprise Java Beans

Components of EJB



Enterprise Java Beans

Components Summary

Feature	Session	Message-Driven	Entity
Process	Business	Communication	DB models
Life Cycle	Short	Short	Longer
Reuseable	Lower	N/A	Higher

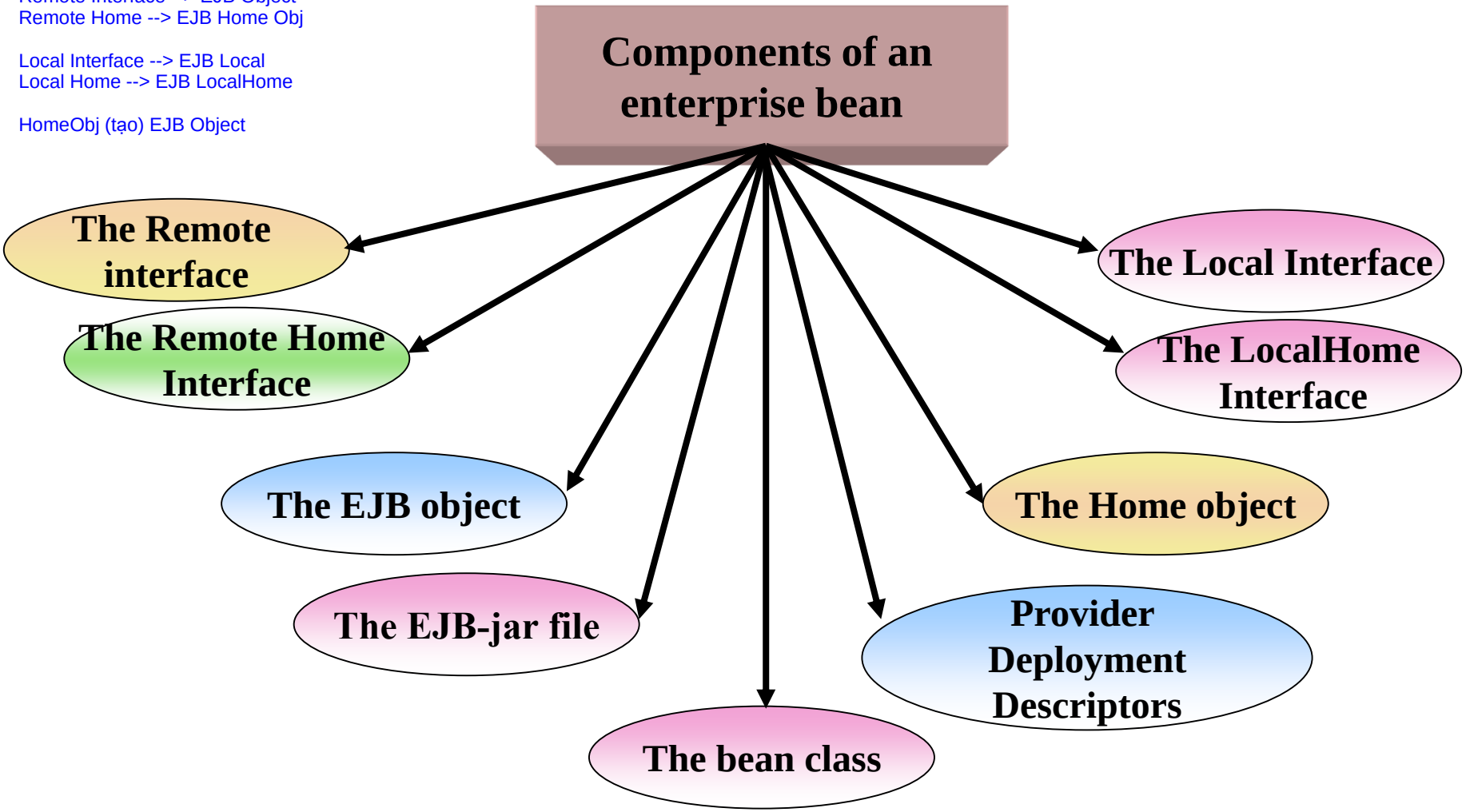
What Constitutes an EJB?

Components of EJB

Remote Interface --> EJB Object
Remote Home --> EJB Home Obj

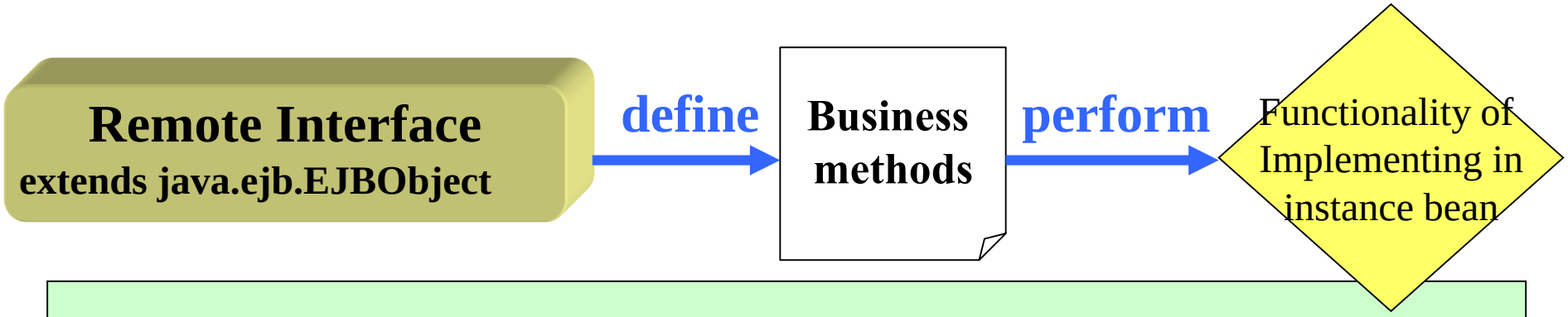
Local Interface --> EJB Local
Local Home --> EJB LocalHome

HomeObj (tạo) EJB Object



What Constitutes an EJB?

Remote Interface



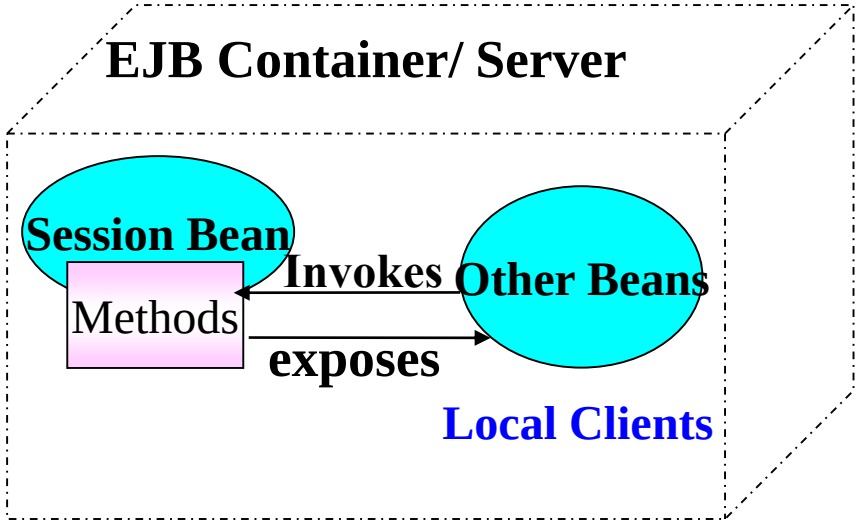

```
public interface WelcomeRemote extends javax.ejb.EJBObject {

    public String welcome() throws java.rmi.RemoteException;

    ...
}
```

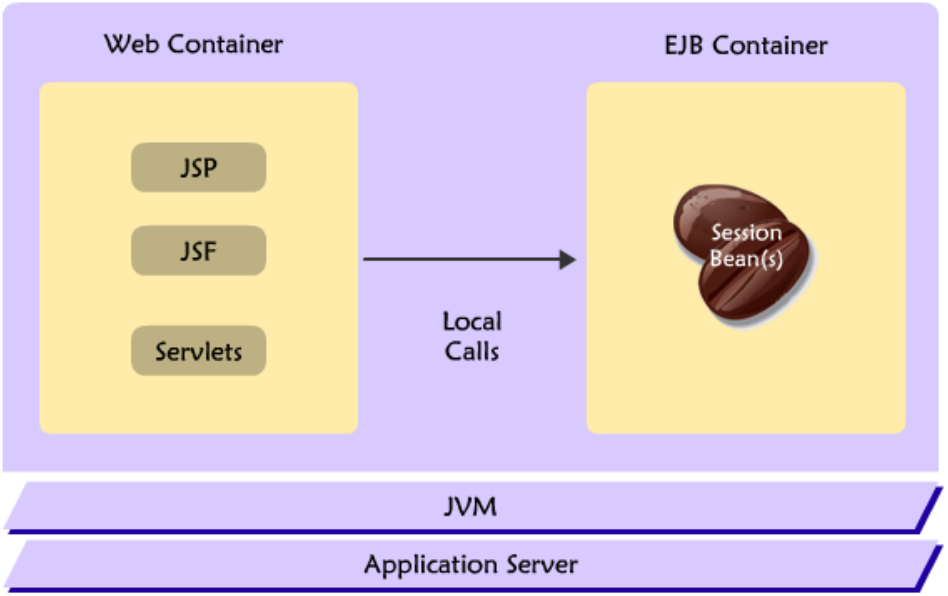
•**Note**: System level operations such as persistence, security and concurrency are not included in remote interface.

What Constitutes an EJB?

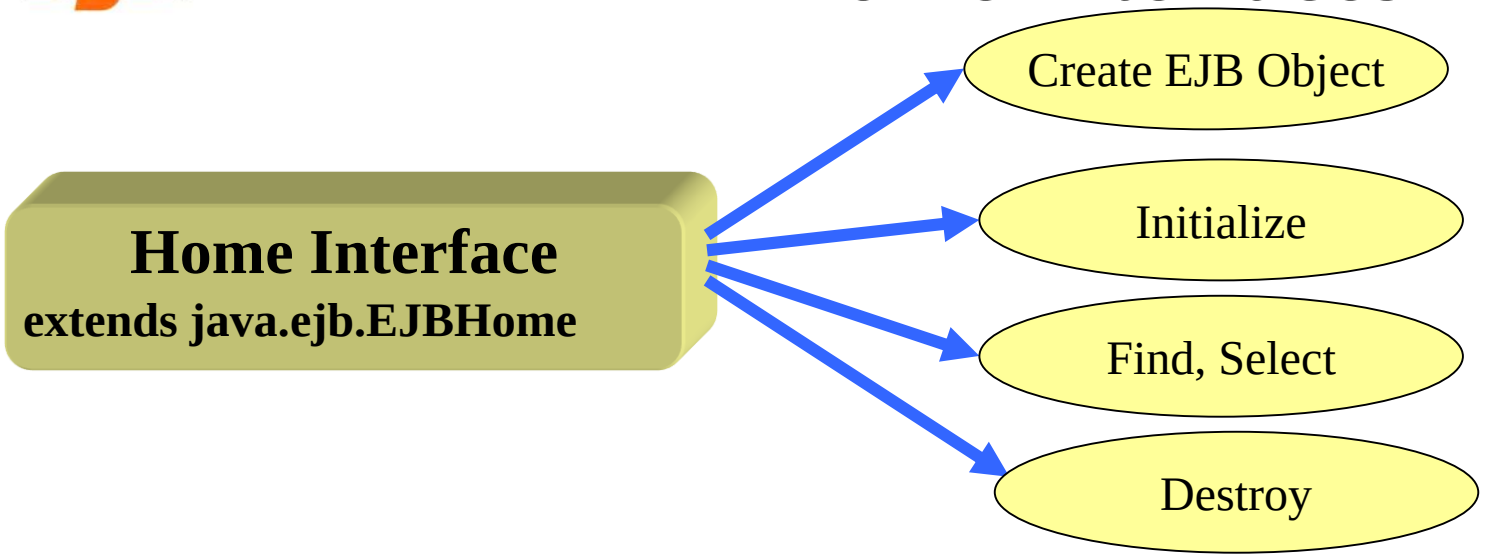
Local Interface

```
public interface WelcomeLocal extends javax.ejb.EJBLocalObject {
   public String welcome();
   ...
}
```





Home Interfaces




```
public interface WelcomeRemoteHome  extends java.ejb.EJBHome {

 public WelcomeRemote   create() throws java.rmi.RemoteException;

 public WelcomeRemote findByPrimaryKey() ...;

 ...
}
```

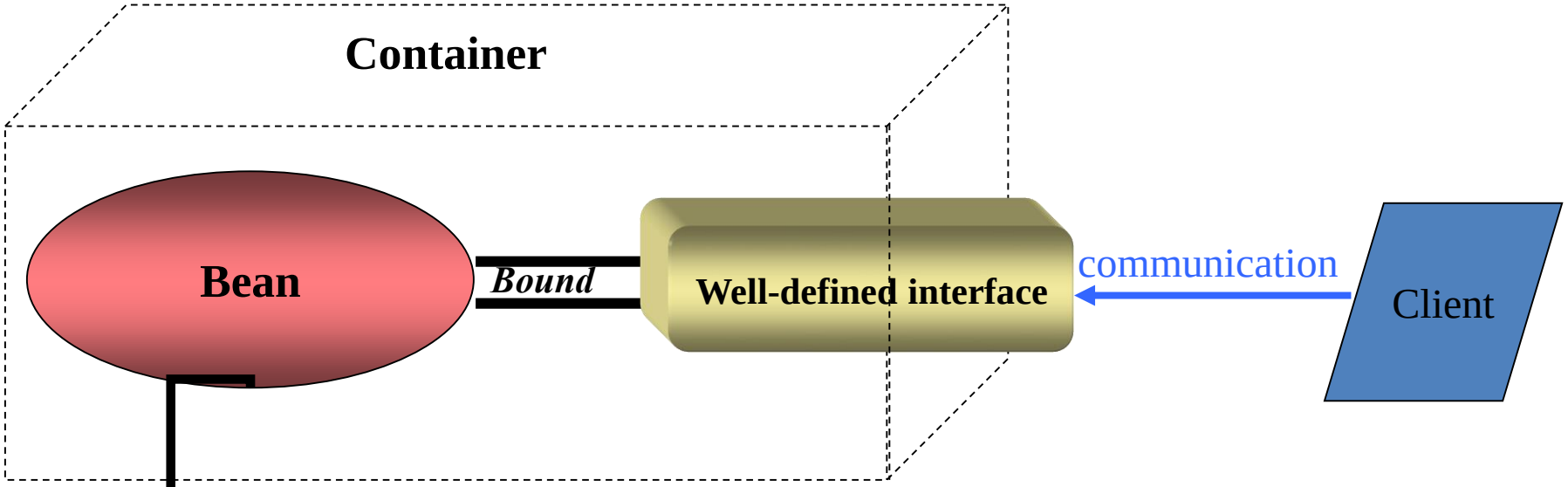
What Constitutes an EJB?

Local Home Objects

- Standard Java interface which allows the beans to expose its methods to other bean **reside within the same container** (local clients)
- **Eleminate the overhead** of the remote method call (java.rmi.RemoteException)
- Use pass by reference semantics (speed up in processing and efficiency)
- extends javax.ejb.EJBLocalHome
- **Notes**: LocalObject is used as Return Values

```
public interface WelcomeLocalHome extends javax.ejb.EJBLocalHome {
  public WelcomeLocal create();
  public WelcomeLocal findByPrimaryKey();
  ...
}
```

Bean Class



- **Implements defined method from Component Interface (Remote and Local)**
- Override default Bean class
- **These methods are then called by container manage bean and keep the bean informed of important events**
- EJB can share the propertiess of the serialiable objects because the javax.ejb.EnterpriseBean extends Serializable
- **Once the interface javax.ejb.EnterpriseBean is implemented, the bean class is confirmed**

What Constitutes an EJB?

Deployment Descriptors

Deployment Descriptor

….
<home>WelcomeRemotehome</home>
<remote>WelcomeRemote</remote>
<ejb-class>Welcomebean</ejb-class>
……

EJB Server/Container

Remote Interface, Home Interface, Bean ... → The DD points out how the beans must interact with one another

Declare middle ware services requirements of components → The DD supply the bean component and performs the requirements

-Life-cycle requirements and bean management: specify how the container should manage the beans

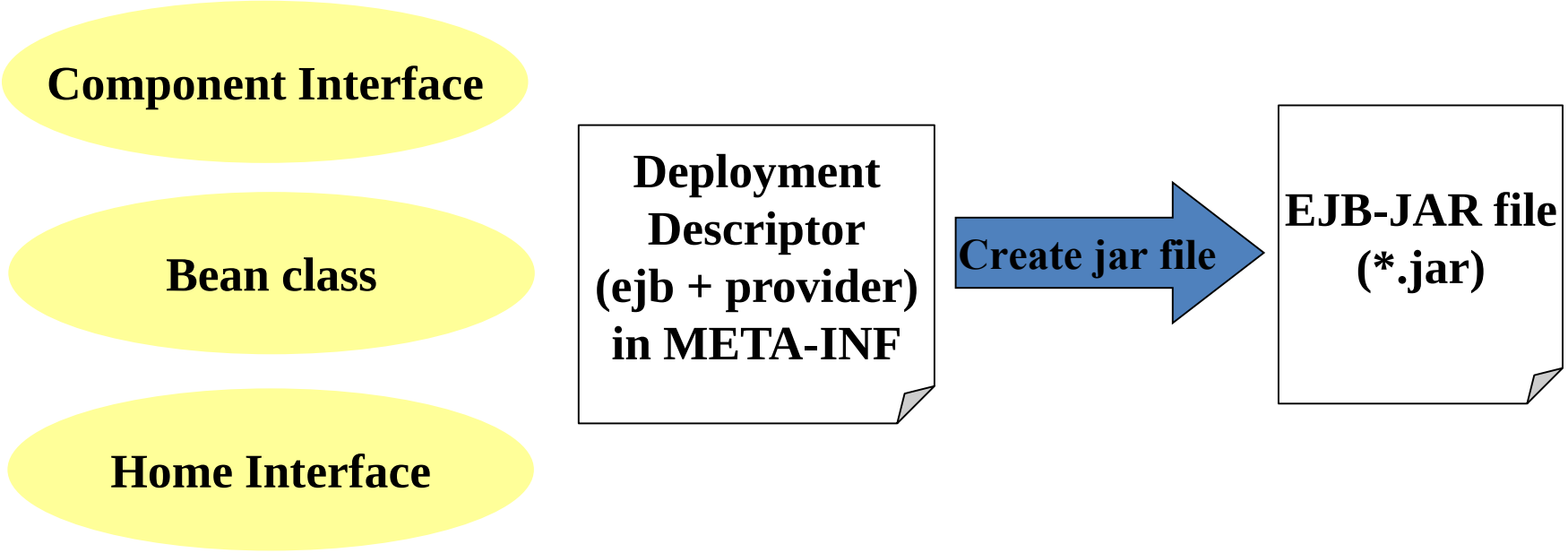
-Persistence requirements: inform EJB container whether the bean take care of/ or delegate persistence

-Transaction requirements: support transaction

-Security management

What Constitutes an EJB?

EJB-JAR file



- EJB container **decompress**, **read and extract** information contained in the EJB-JAR file.
- **Generation** of the EJB object and the home objects, and the bean. (**deployer**)

JNDI

API and Libraries

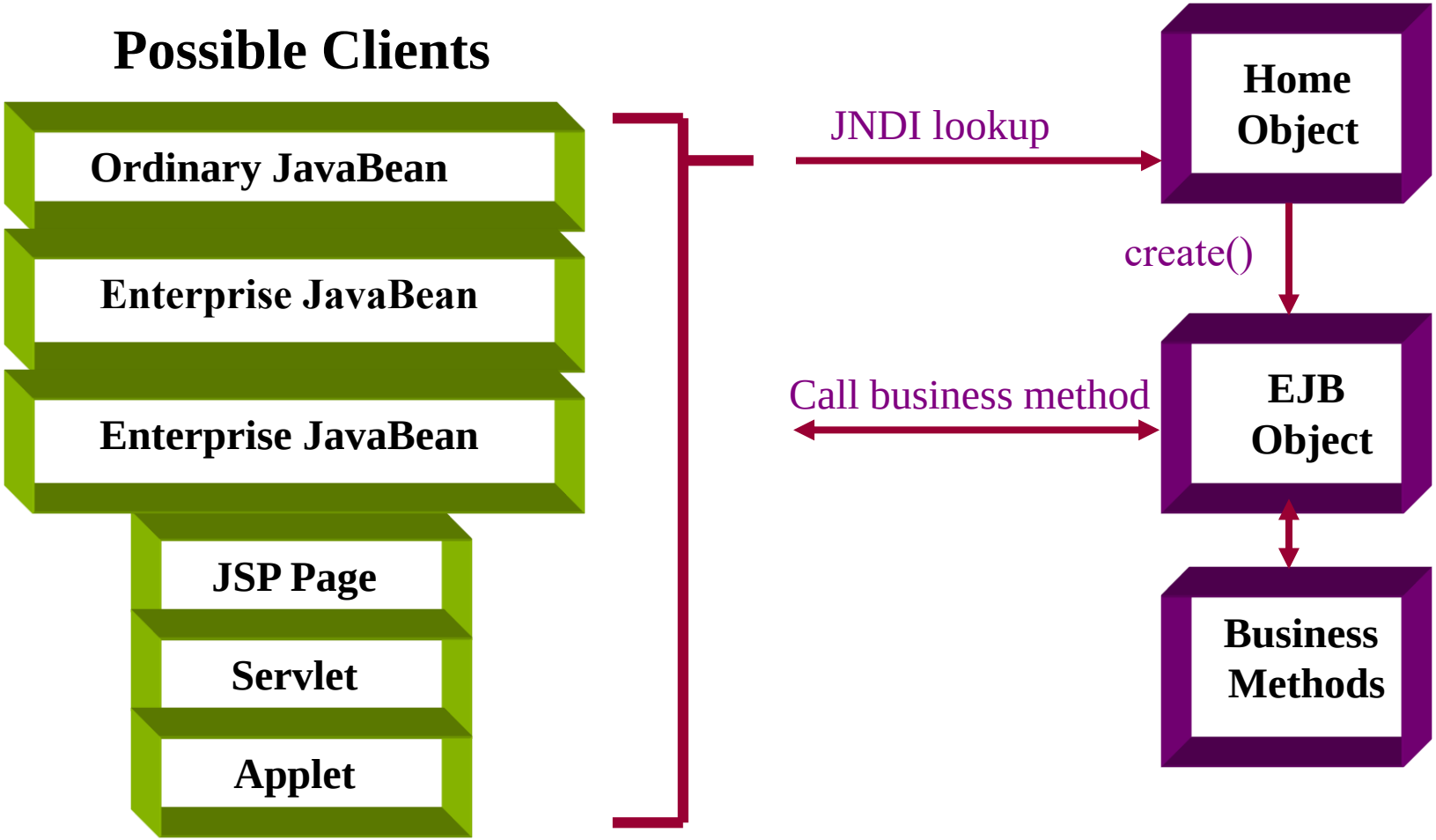
- The **Context** is represented by the **javax.naming.Context interface** that has the necessary methods to put objects into the naming service, and also to locate them.
- The starting point is called an **InitialContext**, represented by **javax.naming.InitialContext interface**
- The references in JNDI are represented by **javax.naming.Reference interface**
  - **The lookup() method** retrieves the object bound to the name and throws a **javax.naming.NamingException**, if a naming exception is encountered

<context_variable>.lookup("object_name")

- The remote calls in EJB make use of **RMI-IIOP** (Remote Method Invocation-Internet Inter-Orb Protocol) which **does not support explicit casting of the EJB object** obtained **from the remote object to a local object**. Instead, Java a RMI-IIOP provides **a mechanism to narrow** the Object you have received from your lookup to the appropriate type by using the **javax.rmi.PortableRemoteObject** class & its **narrow**() method
  - The method narrow() of which parameters narrowFrom is the object that has to be narrowed and narrowTo is the desired type. It returns the object which is cast to the desired type and throws **ClassCastException**, if narrowFrom cannot be cast to narrowTo
- The supported files are **jndi.jar, fscontext.jar, providerutil.jar**

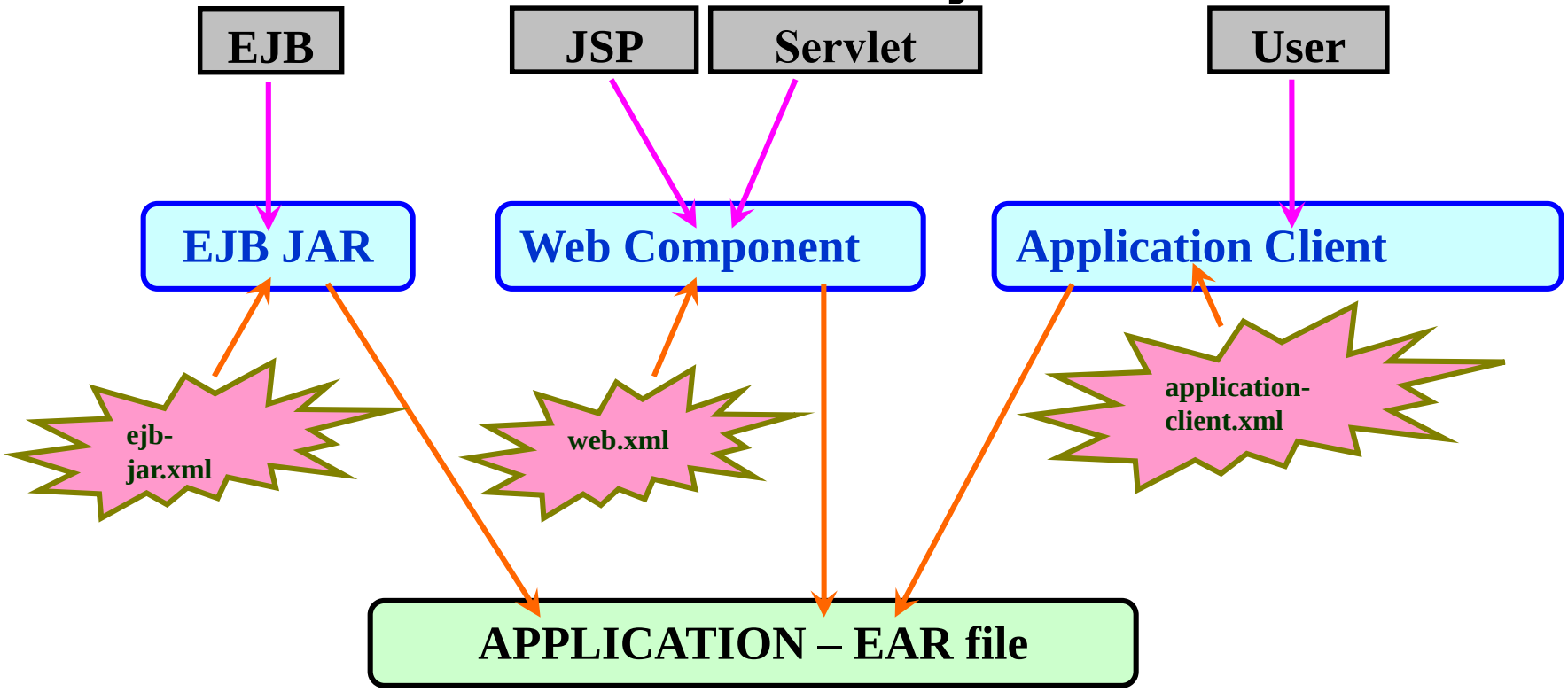
EJB Implementation

Accessing EJB from Client Side



EJB Implementation

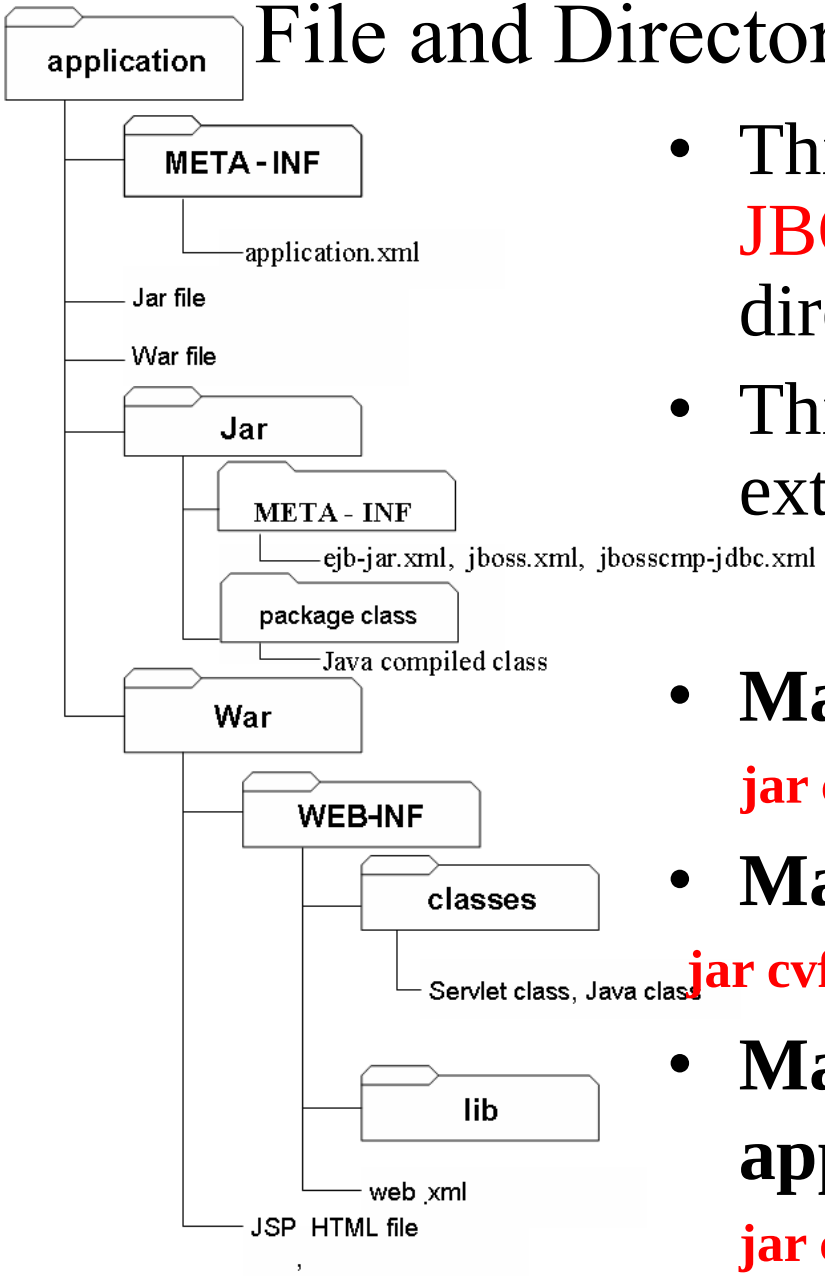
File and Directory Structure



- **Modules** of J2EE Platform (**must** contain **at least one** J2EE module)
  - **EJB modules** cover the data layer and part of the logic layer
  - **Web application** modules cover part of the **logic layer** and the **presentation layer (web application)**
  - **Application client** modules cover part of **logic layer**, and the **presentation layer** (**desktop application**)

EJB Implementation

File and Directory Structure of Enterprise Apps



- This structure is **deployed at** JBOSS_HOME\server\default\deploy directory
- This structure is name with the extension **.ear (include jar and war)**

- **Make deploy ejb file (*.jar)**

  jar cvf user.jar [package/]*.class META-INF/*

- **Make deploy web file (*.war)**

  jar cvf user.war [dir/]*.jsp [dir/]*.html WEB-INF/*

- **Make deploy enterprise application file (*.ear)**

  jar cvf user.ear user.jar user.war META-INF/*

EJB Implementation

Additional – Configure Jboss 6.1.0 Final

- **Go to JBOSS_HOME/bin**
- Open run.bat
- **Edit following content**
  - **Search the line "set JAVA_OPTS=-Dprogram.name=%PROGNAME% -Dlogging.configuration=file:%DIRNAME%logging.properties %JAVA_OPTS%" (line 43)**
  - **Change** the **%DIRNAME% to** the **absolute path** to the "bin" directory of the installed JBoss
  - **Ex:** set JAVA_OPTS=-Dprogram.name=%PROGNAME% -Dlogging.configuration=file: "C:\Programming\jboss6.1.0\bin\logging.properties" %JAVA_OPTS%

EJB Implementation

EJB Development Process

- **Requirement: JBoss 6.1.0 Final Application Server & Netbeans 7.2.1**
- **Step 1:** Creating a new EJB Module project
- **Step 2:** Creating the new corresponding bean depending on your purpose.
- **Step 3:** Building/ Modifying the business/callback methods on Beans
- **Step 4:** Mapping the JNDI to beans
- **Step 5:** Building the project to jar file
- **Step 6:** Deploying the project on Application server
- **Step 7:** Creating the client application to consume
- **Step 8:** Running the client to test the EJB

Summary

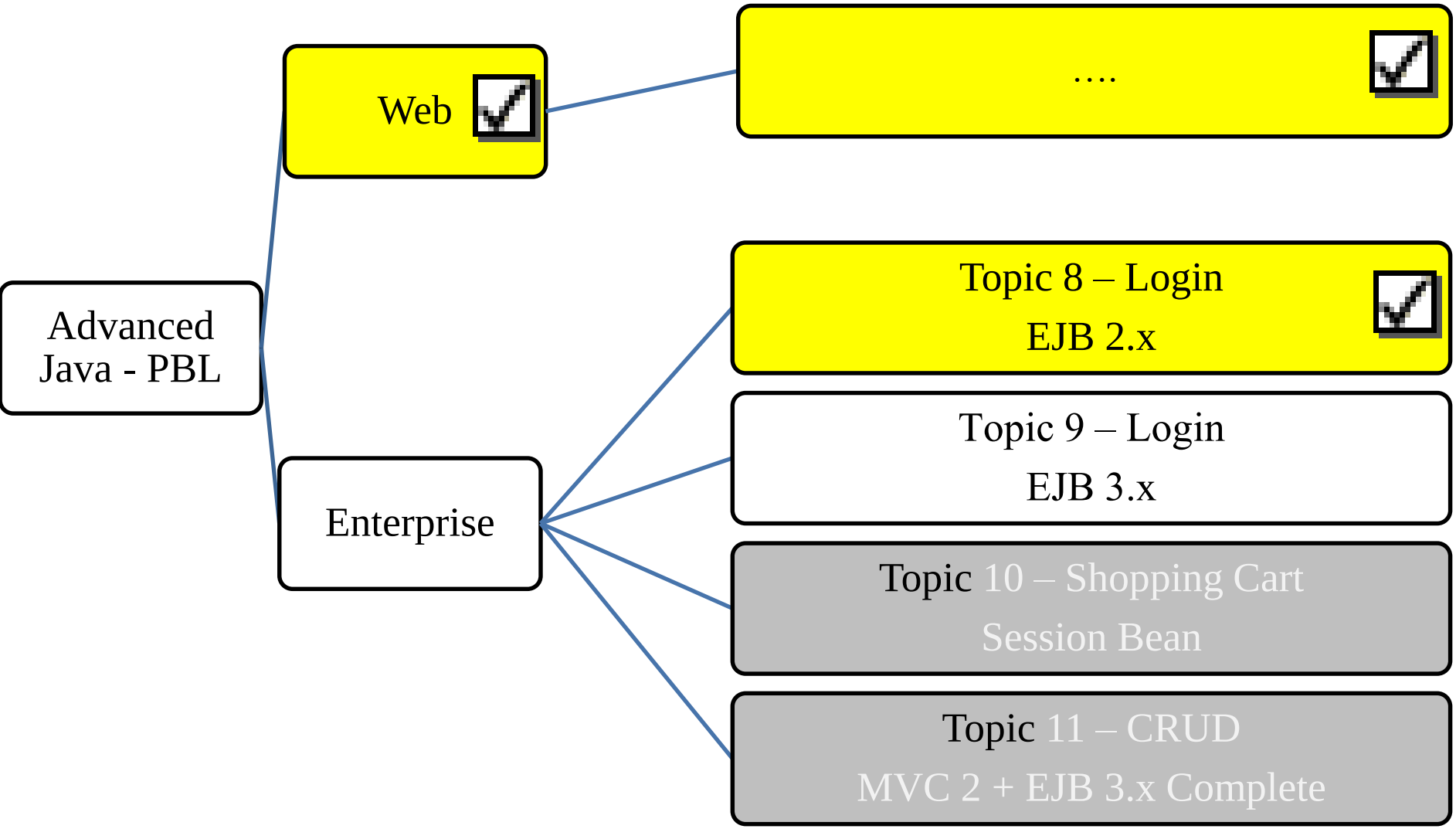
- **How to build the simple enterprise application using EJB 2.0 with GUI as Swing or web?**
  - Logical Architecture of EJB
  - Components of EJB
  - Accessing EJB from the Client/Web Side
  - File and Directory Structure of Enterprise Applications

Q&A

Next Lecture

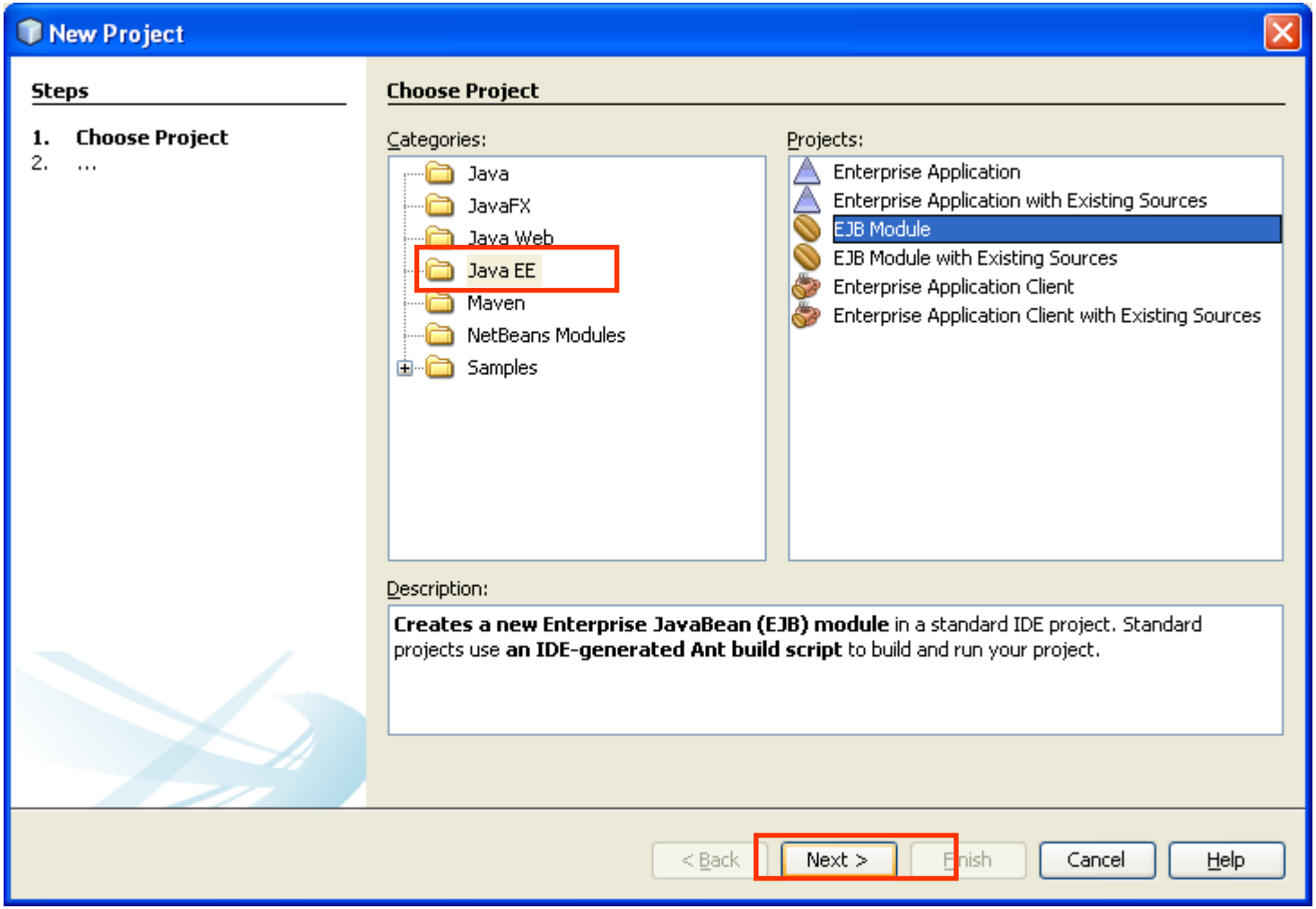
- **How to build the application using EJB 3**
  - Need of EJB 3
  - New Features

Next Lecture



Appendix – EJB Implementation

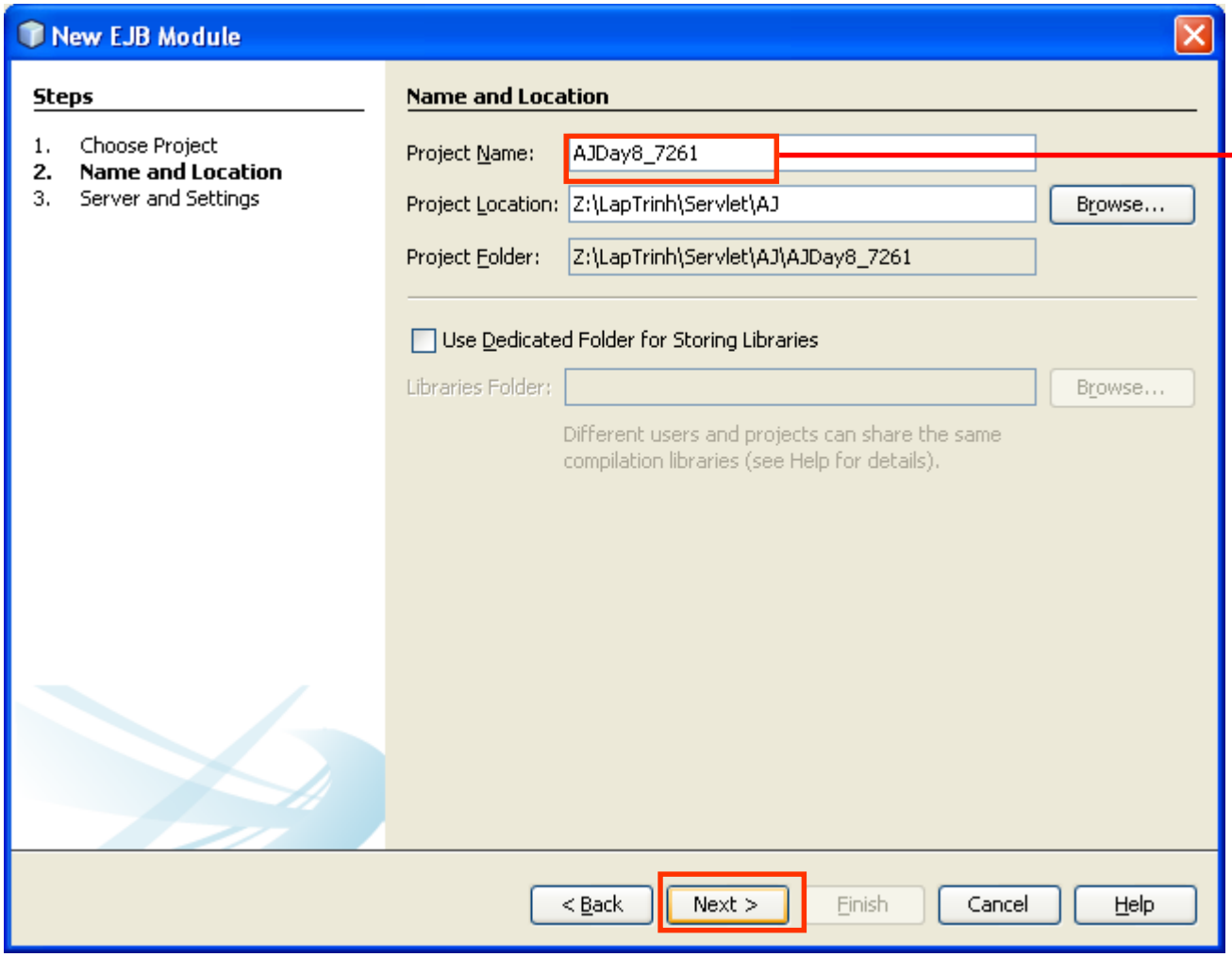
Step 1: Creating a new EJB Module project



- Choose "**Enterprise Java Beans**" on "**Categories**"
- Then, choose "**EJB Module**" on "**Projects**". Click **Next** button

EJB Implementation

Step 1: Creating a new EJB Module project

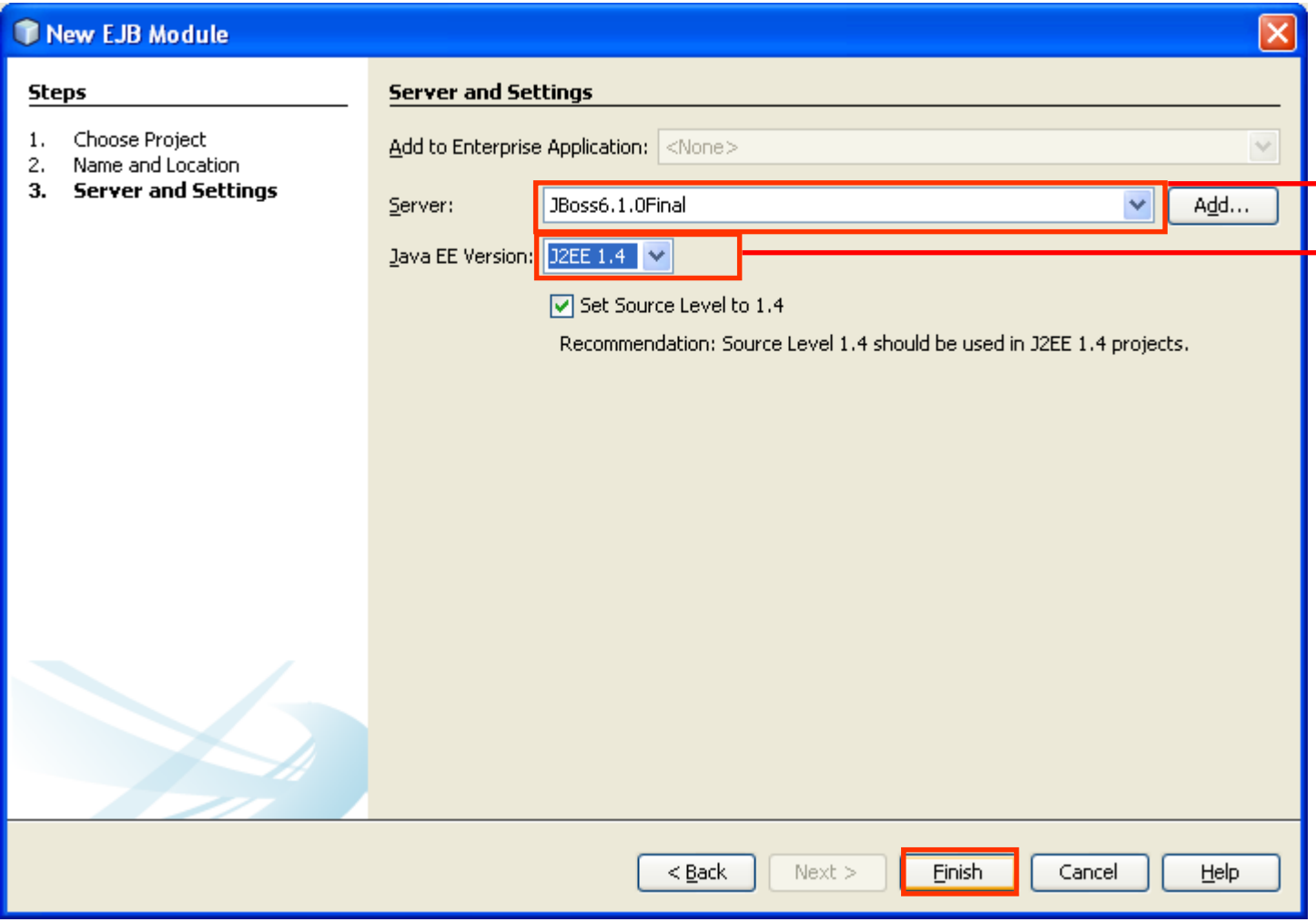


Fill your project name

- Click **Next** button

EJB Implementation

Step 1: Creating a new EJB Module project



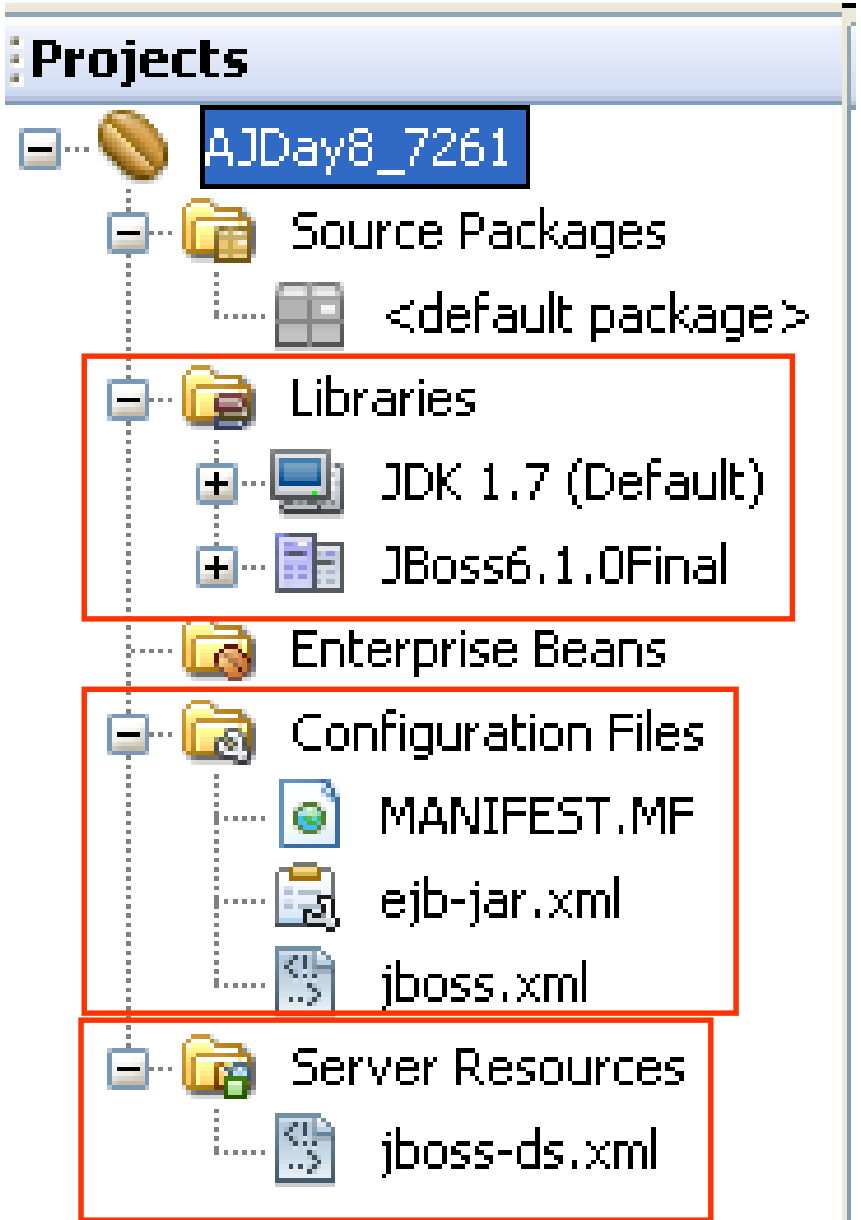
Choose the Jboss Server that is added

Choose the J2EE1.4

- Click **Finish** button

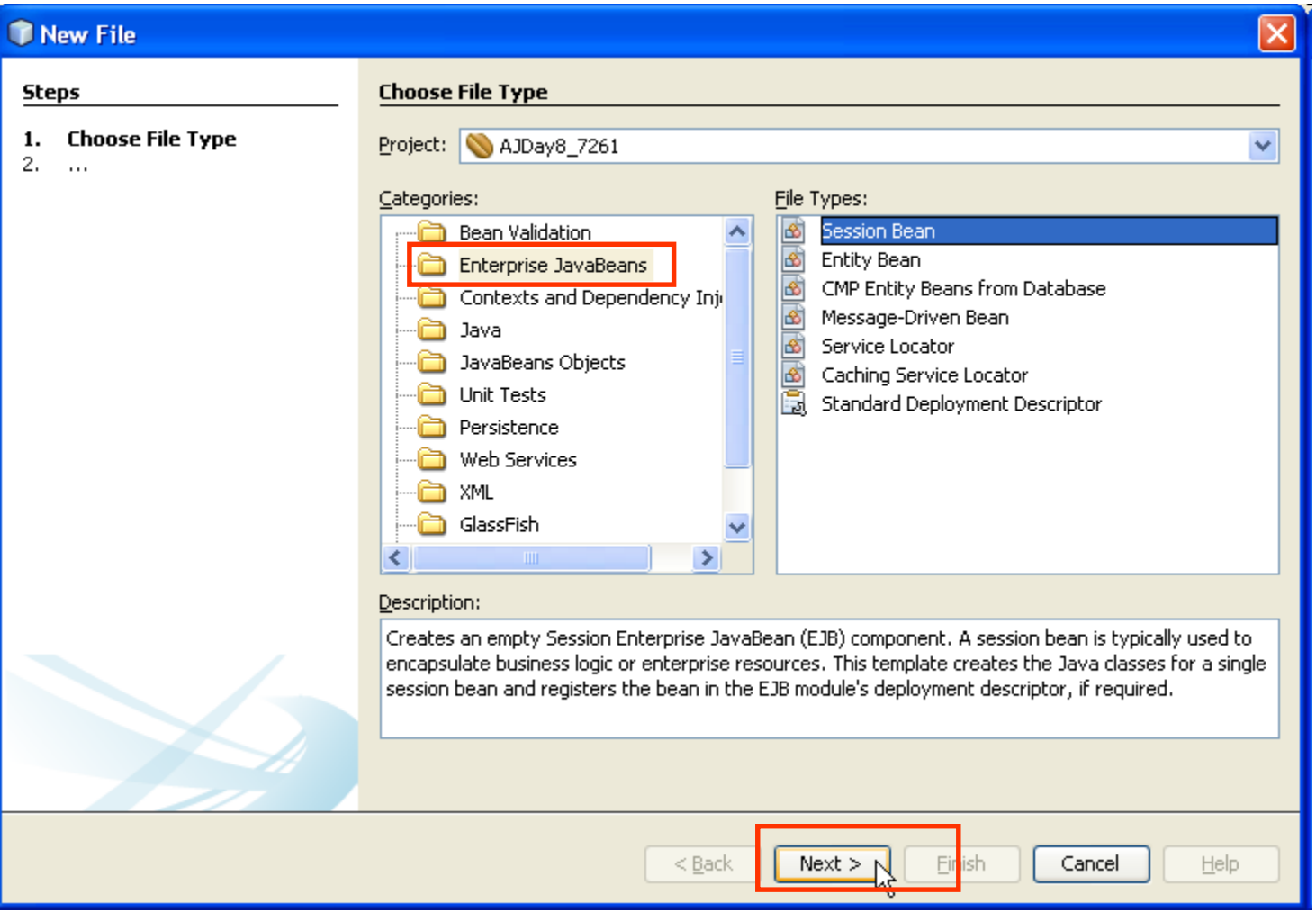
EJB Implementation

Step 1: Creating a new EJB Module project



EJB Implementation

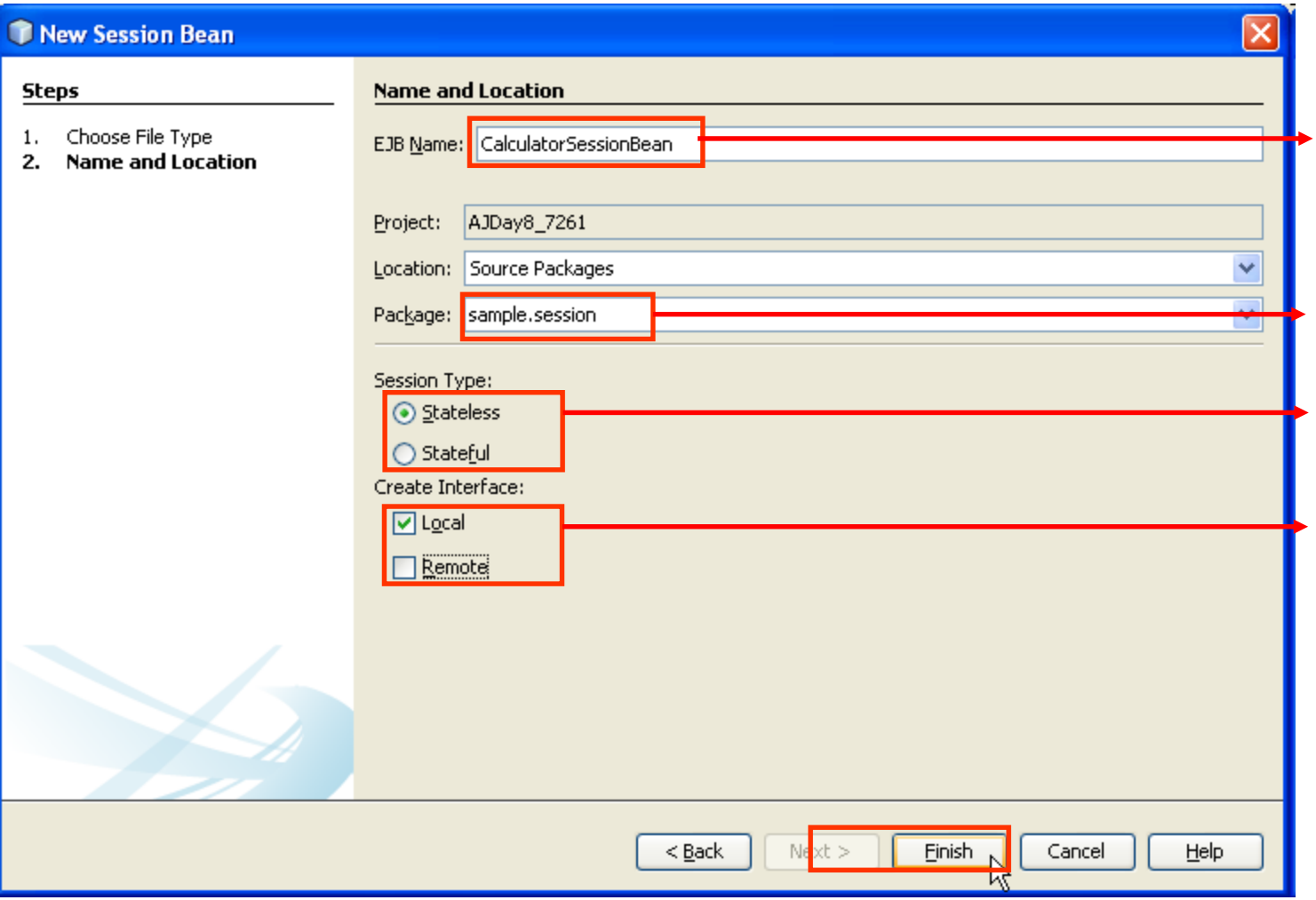
Step 2: Creating the new corresponding bean



- Choose "**Enterprise JavaBeans**" on "**Categories**"
- Then, choose "**Session** Bean" on "**File Types**". Click **Next** button

EJB Implementation

Step 2: Creating the new corresponding bean



Fill your bean name

Fill or choose the package name

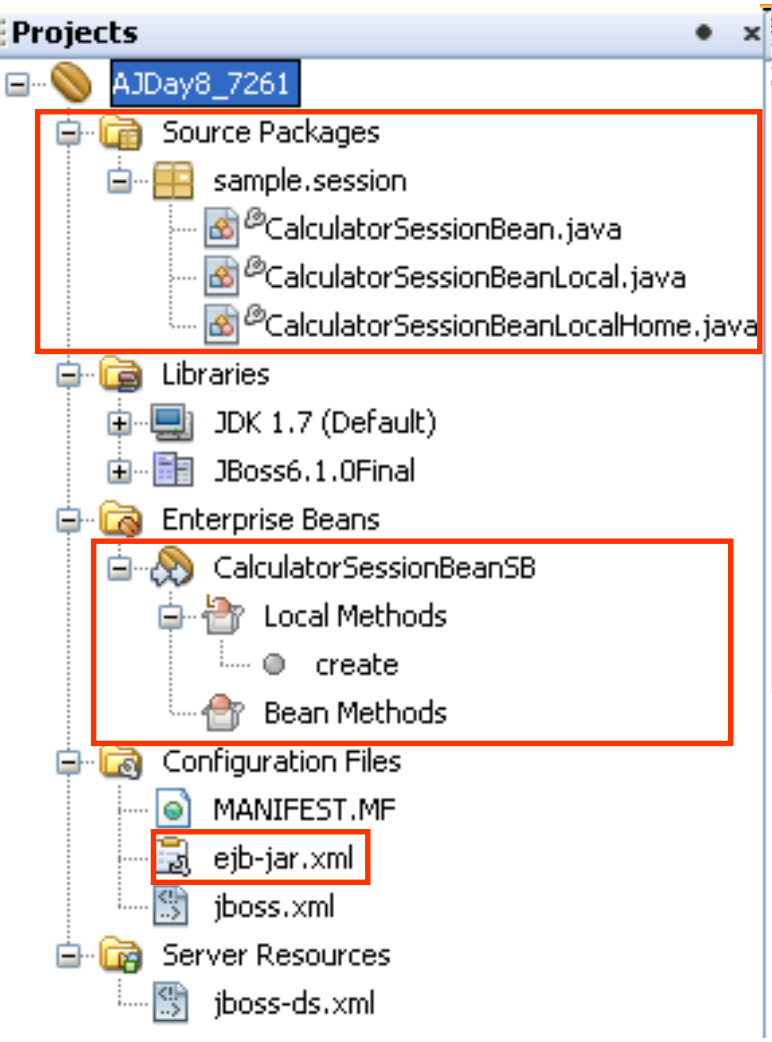
Choose stateless or stateful

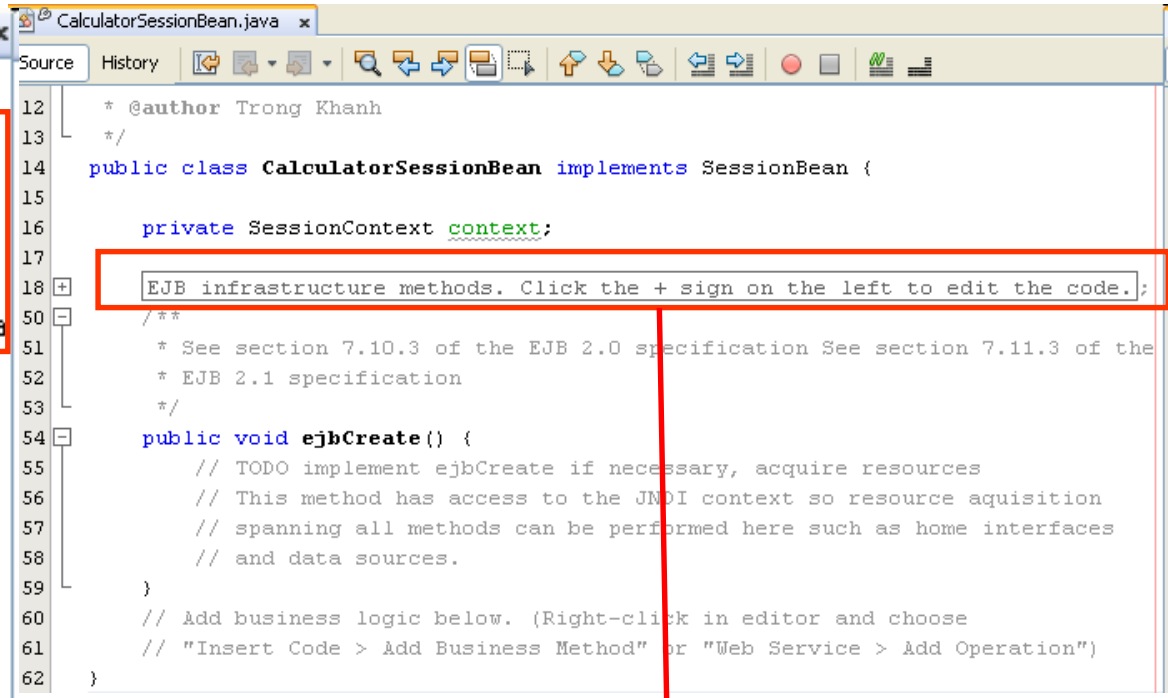
Choose remote or local or both

- Click **Finish** button

EJB Implementation

Step 2: Creating the new corresponding bean





Generate automatically four callback methods:

- setSessionContext
- ejbActivate
- ejbPassivate
- ejbRemove

EJB Implementation

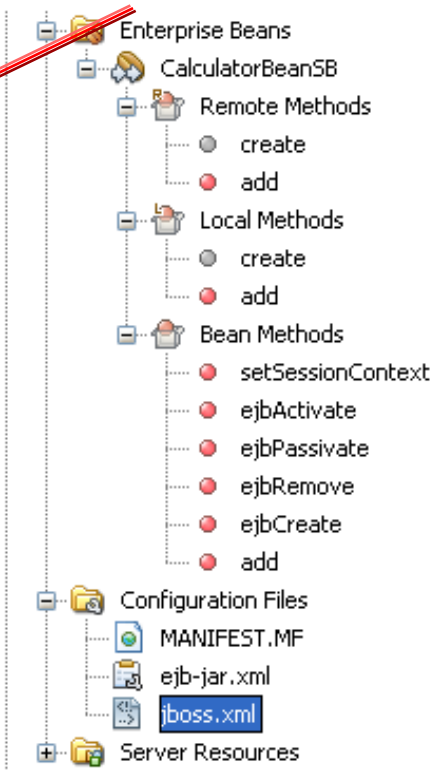
Step 2: Creating the new corresponding bean

ejb-jar.xml

Source | General | CMP Relationships | History

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ejb-jar version="2.1" xmlns="http://java.sun.com/xml/ns/j2ee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/ejb-jar_2_1.xsd">
    <display-name>AJDay8_7261</display-name>
    <enterprise-beans>
        <session>
            <display-name>CalculatorSessionBeanSB</display-name>
            <ejb-name>CalculatorSessionBean</ejb-name>
            <local-home>sample.session.CalculatorSessionBeanLocalHome</local-home>
            <local>sample.session.CalculatorSessionBeanLocal</local>
            <ejb-class>sample.session.CalculatorSessionBean</ejb-class>
            <session-type>Stateless</session-type>
            <transaction-type>Container</transaction-type>
        </session>
    </enterprise-beans>
    <assembly-descriptor>
        <container-transaction>
            <method>
                <ejb-name>CalculatorSessionBean</ejb-name>
                <method-name>*</method-name>
            </method>
            <trans-attribute>Required</trans-attribute>
        </container-transaction>
    </assembly-descriptor>
</ejb-jar>
```

The container creates a new transaction



EJB Implementation

Step 2: Creating the new corresponding bean

CalculatorSessionBeanLocalHome.java

Source History

```
12    * @author Trong Khanh
13    */
14   public interface CalculatorSessionBeanLocalHome extends EJBLocalHome {
15       sample.session.CalculatorSessionBeanLocal create() throws CreateException;
16   }
```

CalculatorSessionBeanLocal.java

Source History

```
11    * @author Trong Khanh
12    */
13   public interface CalculatorSessionBeanLocal extends EJBLocalObject {
14
15   }
```

EJB Implementation

Step 2: Creating the new corresponding bean

- **Create** the **remote interface** and the **remote home interface**
  - **Create** two **java interface**, then **extends** the **EJBHome** and **EJBObject**
  - **Then**, update the **ejb-jar.xml** file

CalculatorSessionBeanRemoteHome.java

Source History

```
13    * @author Trong Khanh
14    */
15   public interface CalculatorSessionBeanRemoteHome extends EJBHome {
16       sample.session.CalculatorSessionBeanRemote create()
17               throws CreateException, RemoteException;
18   }
```

CalculatorSessionBeanRemote.java

Source History

```
11    * @author Trong Khanh
12    */
13   public interface CalculatorSessionBeanRemote extends EJBObject {
14
15   }
```

EJB Implementation

Step 2: Creating the new corresponding bean

ejb-jar.xml

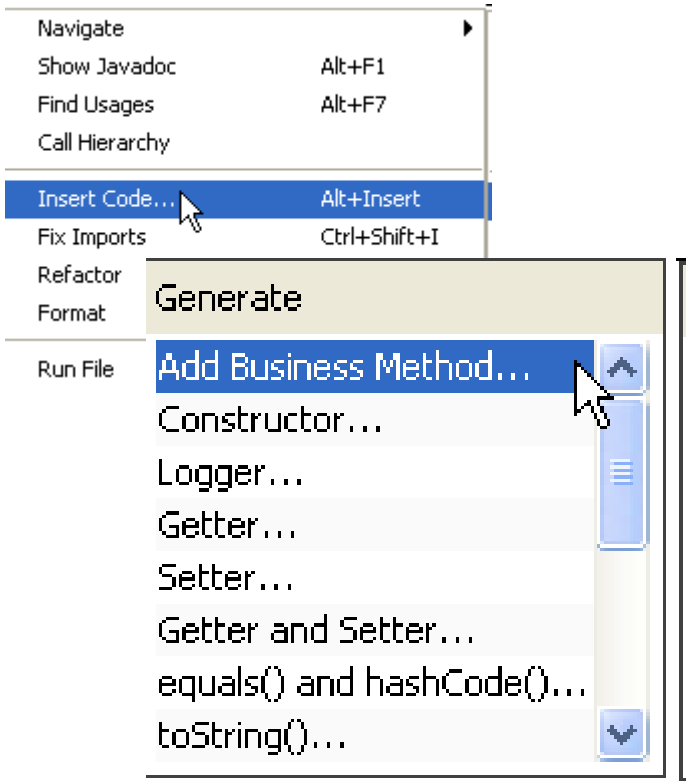
Source | General | CMP Relationships | History

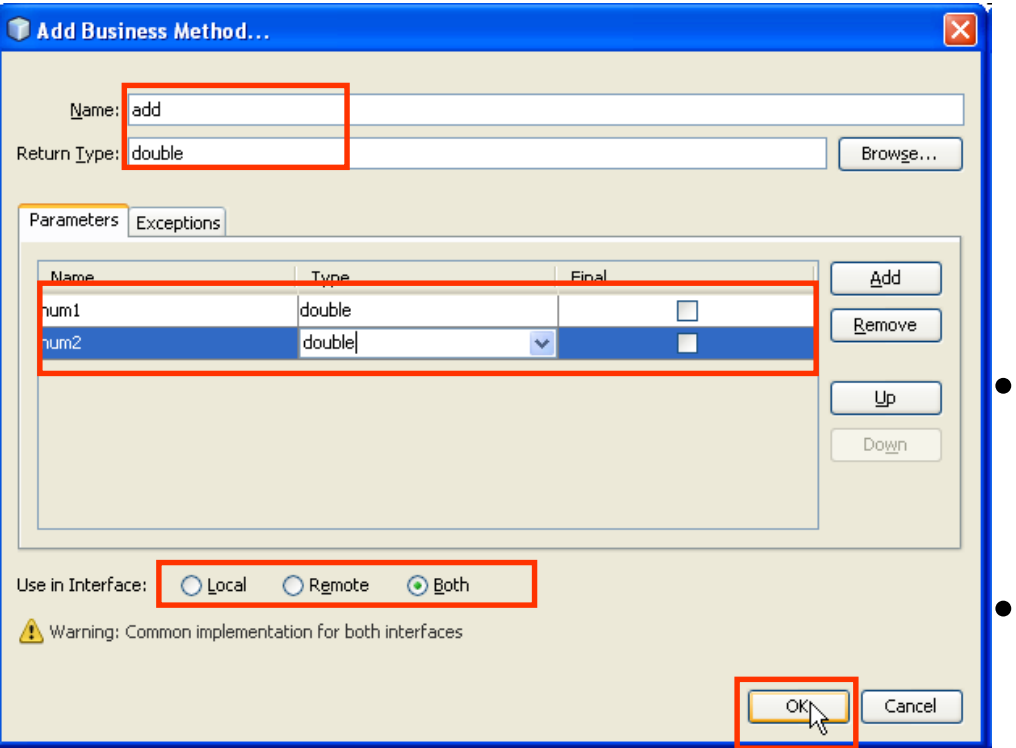
```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <ejb-jar version="2.1" xmlns="http://java.sun.com/xml/ns/j2ee"
3           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4           xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
5  http://java.sun.com/xml/ns/j2ee/ejb-jar_2_1.xsd">
6      <display-name>AJDay8_7261</display-name>
7      <enterprise-beans>
8          <session>
9              <display-name>CalculatorSessionBeanSB</display-name>
10             <ejb-name>CalculatorSessionBean</ejb-name>
11             <home>sample.session.CalculatorSessionBeanRemoteHome</home>
12             <remote>sample.session.CalculatorSessionBeanRemote</remote>
13             <local-home>sample.session.CalculatorSessionBeanLocalHome</local-home>
14             <local>sample.session.CalculatorSessionBeanLocal</local>
15             <ejb-class>sample.session.CalculatorSessionBean</ejb-class>
16             <session-type>Stateless</session-type>
17             <transaction-type>Container</transaction-type>
18         </session>
19     </enterprise-beans>
20     <assembly-descriptor>
```

EJB Implementation

Building/ Modifying the business/callback methods

- **Modifying** the **callback method** if necessary
- **Adding** a new **business method**
  - Right click on source code of the Bean file (Ex: CalculateBean)
  - Then, choose Insert Code, click Add Business Method…





- Fill or type the method name with return type and add all parameters
- Then, click OK Button

EJB Implementation

Building/ Modifying the business/callback methods

CalculatorSessionBean.java

Source History

```
12      * @author Trong Khanh
13      */
14    public class CalculatorSessionBean implements SessionBean {
15        private SessionContext context;
16        EJB infrastructure methods. Click the + sign on the left to edit the code.;
48        /**...*/
52        public void ejbCreate() {...}
58        // Add business logic below. (Right-click in editor and choose
59        // "Insert Code > Add Business Method" or "Web Service > Add Operation")
60        public double add(double num1, double num2) {
61            return 0.0;
62        }
63    }
```

CalculatorSessionBeanRemote.java

Source History

```
12      * @author Trong Khanh
13      */
14    public interface CalculatorSessionBeanRemote extends EJBObject {
15        double add(double num1, double num2) throws RemoteException;
16    }
```

EJB Implementation

Building/ Modifying the business/callback methods

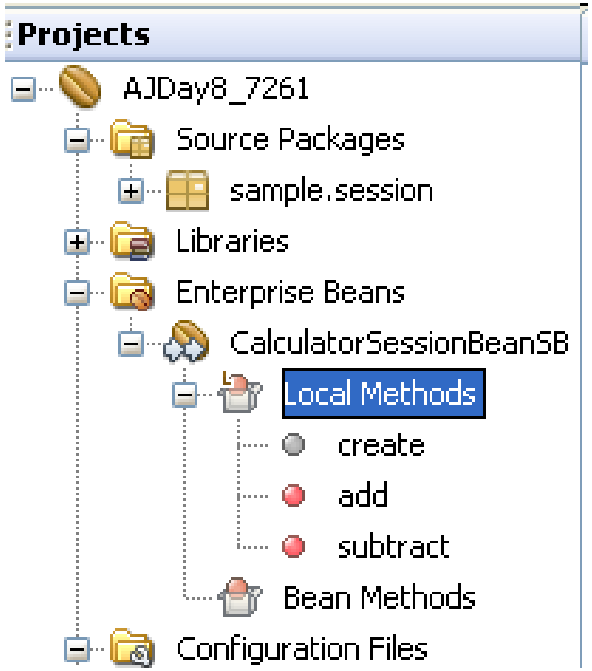
CalculatorSessionBeanLocal.java

```
11     * @author Trong Khanh
12     */
13    public interface CalculatorSessionBeanLocal extends EJBLocalObject {
14        double add(double num1, double num2);
15    }
```

- Implement the body of method corresponding with your purpose

CalculatorSessionBean.java

```
12    * @author Trong Khanh
13    */
14   public class CalculatorSessionBean implements SessionBe
15       private SessionContext context;
16       EJB infrastructure methods. Click the + sign on the
48       /**...*/
52       public void ejbCreate() {...}
58       // Add business logic below. (Right-click in editor
59       // "Insert Code > Add Business Method" or "Web Serv
60       public double add(double num1, double num2) {
61           return num1 + num2;
62       }
63
64       public double subtract(double num1, double num2) {
65           return num1 - num2;
66       }
67   }
```



EJB Implementation

Building/ Modifying the business/callback methods

CalculatorSessionBeanRemote.java

Source History

```
12     * @author Trong Khanh
13     */
14    public interface CalculatorSessionBeanRemote extends EJBObject {
15        double add(double num1, double num2) throws RemoteException;
16
17        double subtract(double num1, double num2) throws RemoteException;
18    }
```

CalculatorSessionBeanLocal.java

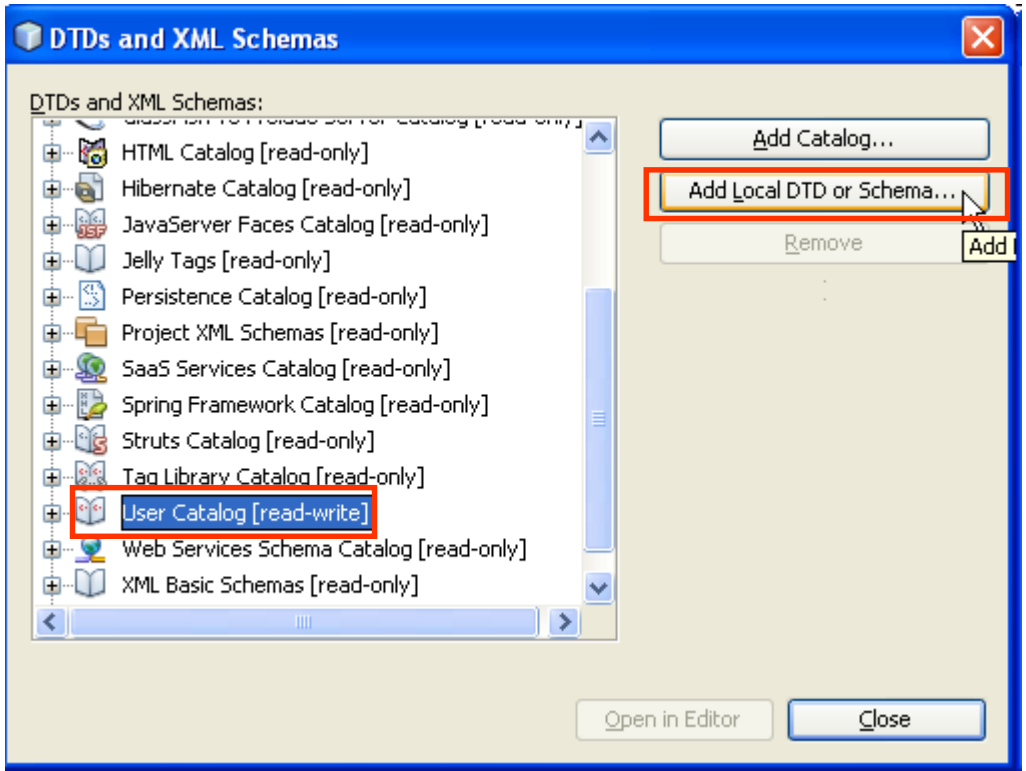
Source History

```
10     *
11     * @author Trong Khanh
12     */
13    public interface CalculatorSessionBeanLocal extends EJBLocalObject {
14        double add(double num1, double num2);
15
16        double subtract(double num1, double num2);
17    }
```

EJB Implementation

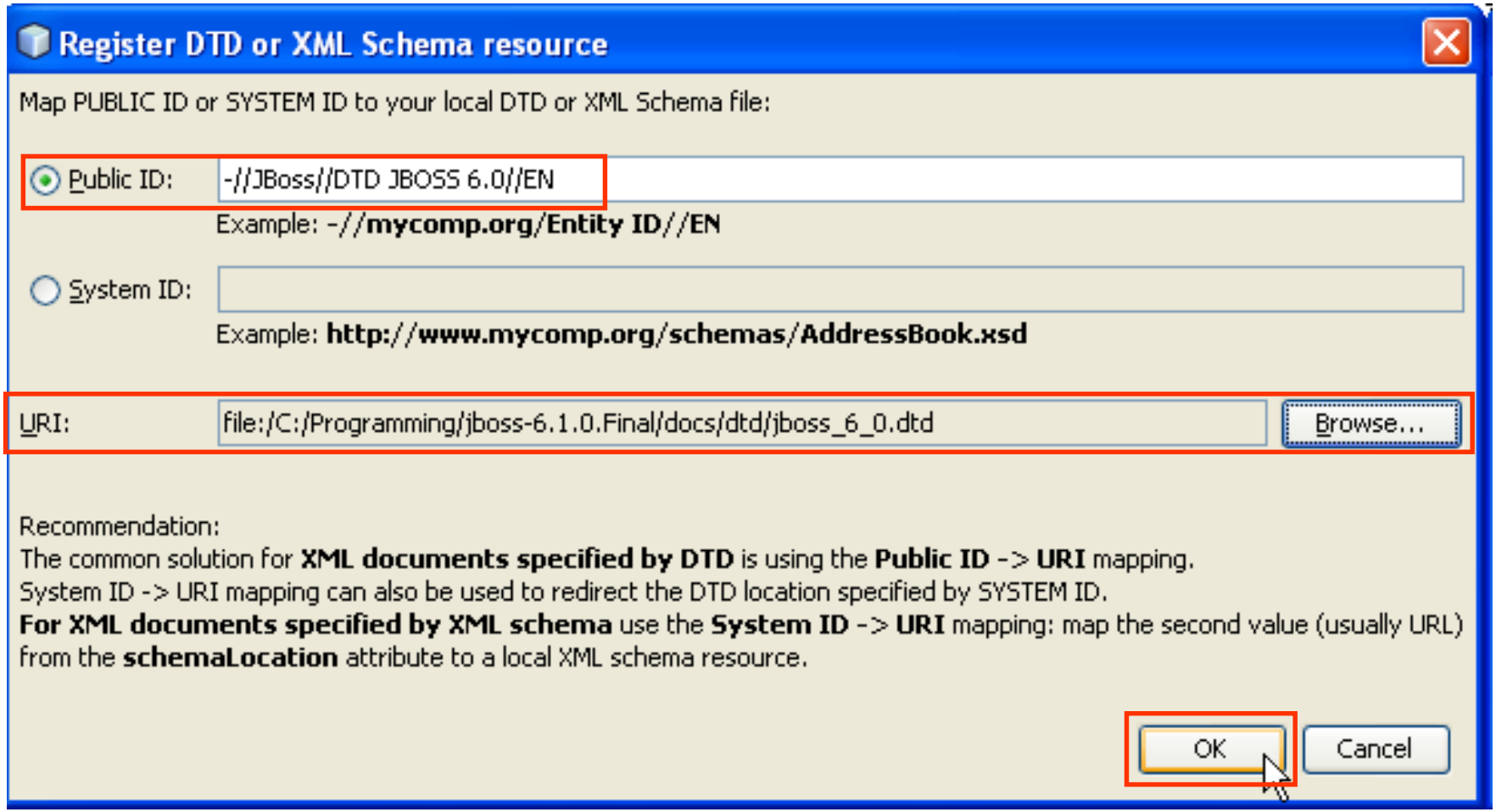
Additional - Mapping JNDI

- **Set up** visual typing to **jboss.xml file**
  - Copy **the jboss_6_0.dtd** file to your local disk
  - Mapping this file to Netbeans as following steps
    - Click menu Tools, click "**DTDs and XML Schemas**" items



EJB Implementation

Additional - Mapping JNDI



EJB Implementation

Additional - Mapping JNDI

jboss.xml

Source History

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE jboss PUBLIC
3          "-//JBoss//DTD JBOSS 6.0//EN"
4          "http://www.jboss.org/j2ee/dtd/jboss_6_0.dtd">
```

EJB Implementation

Step 4: Mapping the JNDI to beans

- Modify the jboss.xml file as following

jboss.xml

Source | History

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE jboss PUBLIC
        "-//JBoss//DTD JBOSS 6.0//EN"
        "http://www.jboss.org/j2ee/dtd/jboss_6_0.dtd">
<jboss>
    <enterprise-beans>
        <session>
            <ejb-name>CalculatorSessionBean</ejb-name>
            <jndi-name>CalJNDI</jndi-name>
            <local-jndi-name>CalLocalJNDI</local-jndi-name>
        </session>
    </enterprise-beans>
</jboss>
```

Fill your wanted JNDI that you want to reference

ejb-jar.xml

Source | General | CMP Relationships | History

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ejb-jar version="2.1" xmlns="http://java.sun.com/xml/ns/j2ee" x
    <display-name>AJDay8_7261</display-name>
    <enterprise-beans>
        <session>
            <display-name>CalculatorSessionBeanSB</display-name>
            <ejb-name>CalculatorSessionBean</ejb-name>
```

EJB Implementation

Building & Deploying

z:\LapTrinh\Servlet\AJ\AJDay8_7261\dist*.*

Name	Ext
[..]	
AJDay8_7261	jar

z:\LapTrinh\Servlet\AJ\AJDay8_7261\dist\AJDay8_7261.zip*.*

Name	Ext	Size
[..]		<DIR>
[META-INF]		<DIR>
[sample]		<DIR>

z:\LapTrinh\Servlet\AJ\AJDay8_7261\dist\AJDay8_7261.zip\META-INF*.*

Name	Ext	Size
[..]		<DIR>
ejb-jar	xml	1.317
jboss	xml	410
MANIFEST	MF	103

z:\LapTrinh\Servlet\AJ\AJDay8_7261\dist\AJDay8_7261.zip\sample\session*.*

Name	Ext	Size
[..]		<DIR>
CalculatorSessionBean	class	1.027
CalculatorSessionBeanLocal	class	221
CalculatorSessionBeanLocalHome	class	305
CalculatorSessionBeanRemote	class	281
CalculatorSessionBeanRemoteHome	class	334

c:\Programming\jboss-6.1.0.Final\server\default\deploy*.*

Name	Ext	Size
[..]		<DIR>
[hornetq]		<DIR>
[http-invoker.sar]		<DIR>
[jbossweb.sar]		<DIR>
[jms-ra.rar]		<DIR>
[mod_cluster.sar]		<DIR>
[ROOT.war]		<DIR>
[security]		<DIR>
[uuid-key-generator.sar]		<DIR>
[xnio-provider.jar]		<DIR>
AJDay8_7261	jar	5.522

Output

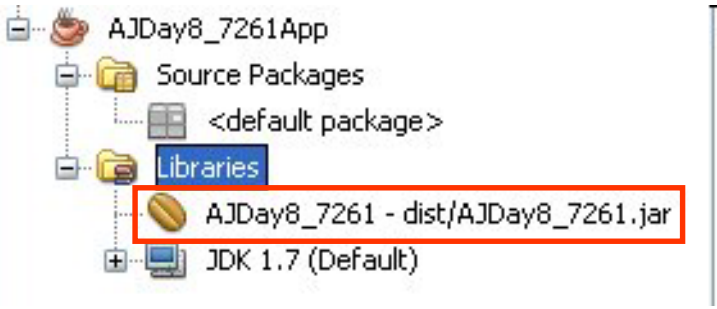
JBoss6.1.0Final × AJDay8_7261 (clean,dist) ×

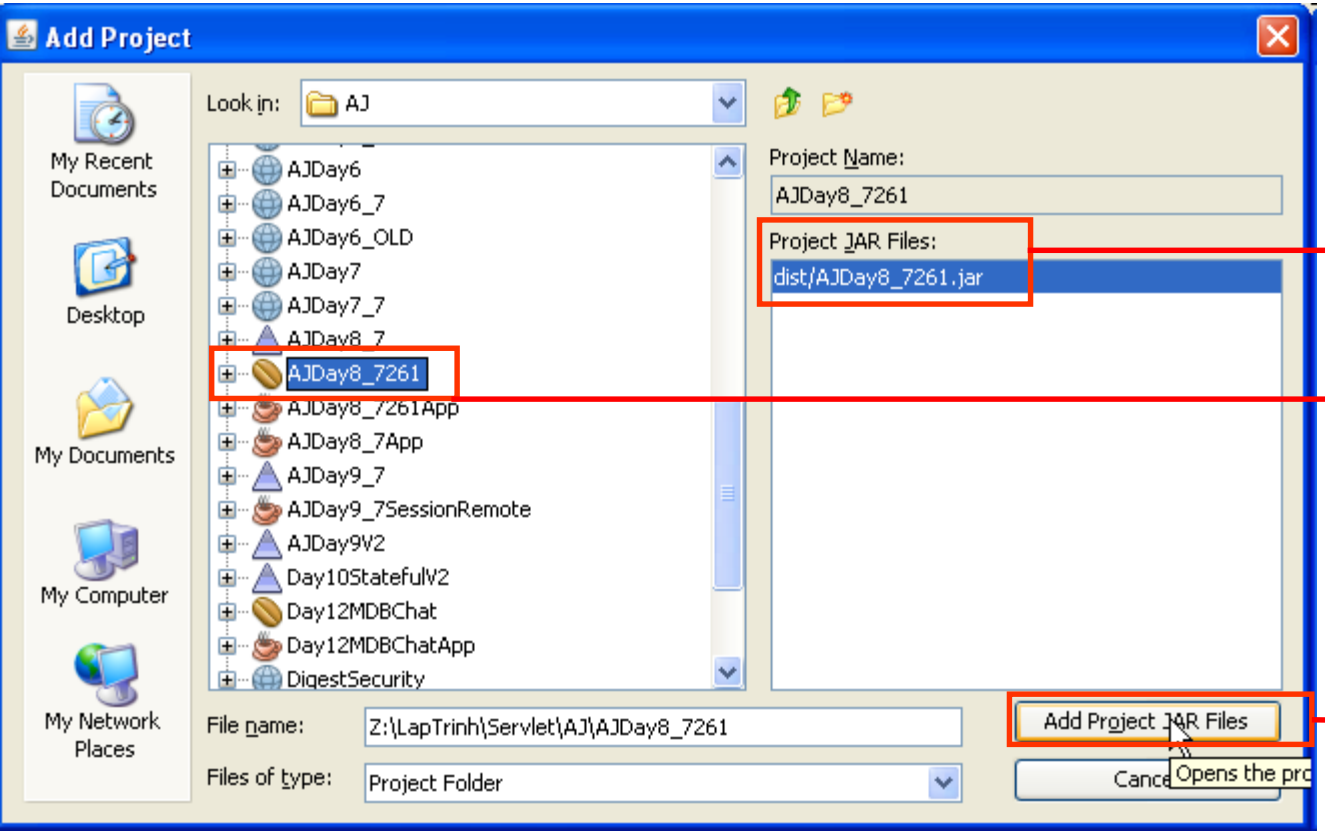
```
22:15:26,046 INFO  [org.jboss.ejb.deployers.EjbDeployer] installing bean: ejb/#CalculatorSessionBean,uid28945344
22:15:26,046 INFO  [org.jboss.ejb.deployers.EjbDeployer]   with dependencies:
22:15:26,046 INFO  [org.jboss.ejb.deployers.EjbDeployer]   and supplies:
22:15:26,046 INFO  [org.jboss.ejb.deployers.EjbDeployer]        jndi:CalculatorSessionBean/sample.session.CalculatorSessionBeanRemote
22:15:26,046 INFO  [org.jboss.ejb.deployers.EjbDeployer]        jndi:CalculatorSessionBean/sample.session.CalculatorSessionBeanLocal
22:15:26,046 INFO  [org.jboss.ejb.deployers.EjbDeployer]        jndi:CalJNDI
22:15:26,046 INFO  [org.jboss.ejb.deployers.EjbDeployer]        jndi:CalLocalJNDI
22:15:26,078 INFO  [org.jboss.ejb.EjbModule] Deploying CalculatorSessionBean
22:15:26,109 INFO  [org.jboss.ejb.plugins.local.BaseLocalProxyFactory] Bound EJB LocalHome 'CalculatorSessionBean' to jndi 'CalLocalJNDI
22:15:26,109 INFO  [org.jboss.proxy.ejb.ProxyFactory] Bound EJB Home 'CalculatorSessionBean' to jndi 'CalJNDI'
```

EJB Implementation

Creating the client application

- Create Java console application
- Add reference to EJB project mapping to invoke the remote method on application Server
  - Right click on library of client project/ click "Add Project …"





Choose the jar file

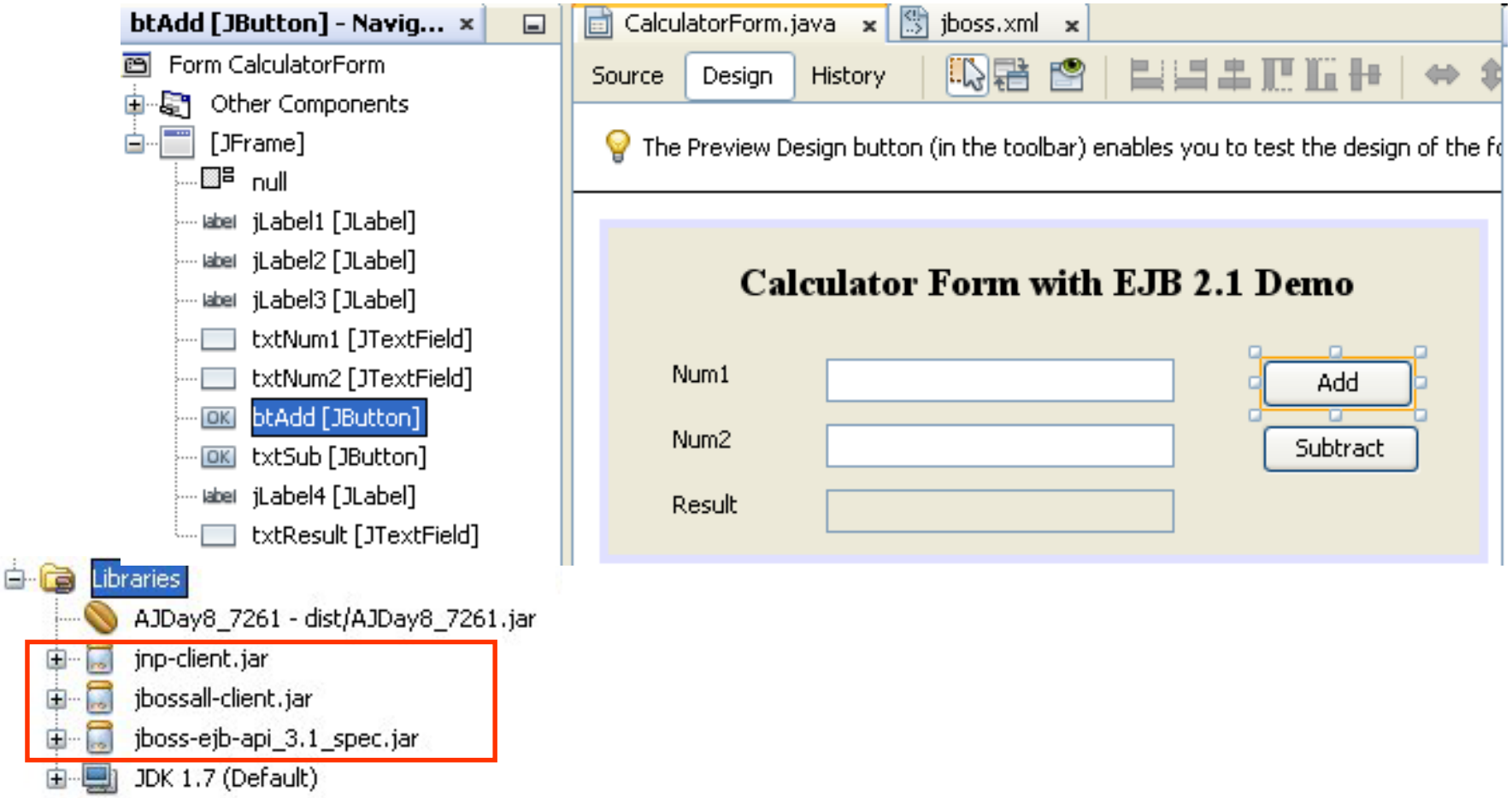
Choose EJB module

Click Add Project Jar File

EJB Implementation

Creating the client application

- Adding the code as following (**notes: addition the jbossall-client.jar, jnp-client.jar, and jboss-ejb-api_3.1_spec.jar from JBOSS_HOME\client to application project**)



EJB Implementation

Creating the client application

CalculatorForm.java

Source | Design | History

```
20   * @author Trong Khanh
21   */
22  public class CalculatorForm extends javax.swing.JFrame {
23      /**...*/
26      public CalculatorForm() {...}
29      /**...*/
34      @SuppressWarnings("unchecked")
35      Generated Code
92
93      private void btAddActionPerformed(java.awt.event.ActionEvent evt) {
94          try {
95              System.setProperty("java.naming.factory.initial",
96                      "org.jnp.interfaces.NamingContextFactory");
97              System.setProperty("java.naming.provider.url", "localhost:1099");
98              Context context = new InitialContext();
99              Object obj = context.lookup("CalJNDI");
100             CalculatorSessionBeanRemoteHome ejbHome = (CalculatorSessionBeanRemoteHome)
101                     PortableRemoteObject.narrow(obj, CalculatorSessionBeanRemoteHome.class);
102             CalculatorSessionBeanRemote ejbObj = ejbHome.create();
103             String n1 = txtNum1.getText();
104             String n2 = txtNum2.getText();
105             double num1 = Double.parseDouble(n1);
106             double num2 = Double.parseDouble(n2);
107             double result = ejbObj.add(num1, num2);
108             txtResult.setText(result + "");
            } catch (CreateException ex) {
110             Logger.getLogger(CalculatorForm.class.getName()).log(Level.SEVERE, null, ex);
111         } catch (RemoteException ex) {
112             Logger.getLogger(CalculatorForm.class.getName()).log(Level.SEVERE, null, ex);
113         } catch (NamingException ex) {
114             Logger.getLogger(CalculatorForm.class.getName()).log(Level.SEVERE, null, ex);
```

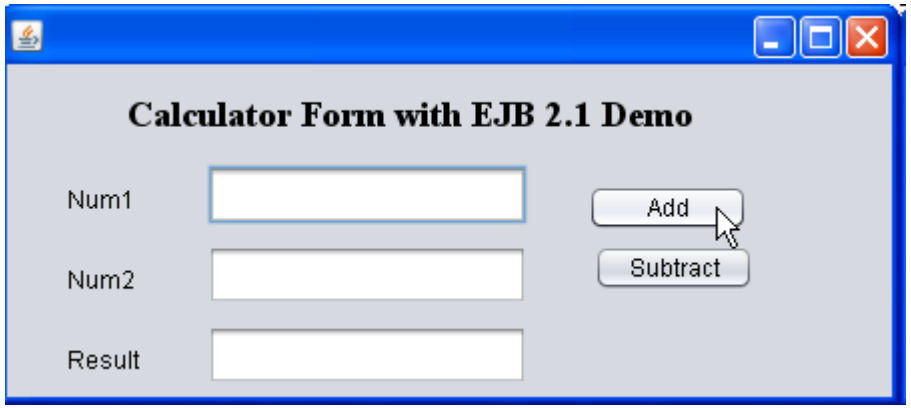
EJB Implementation

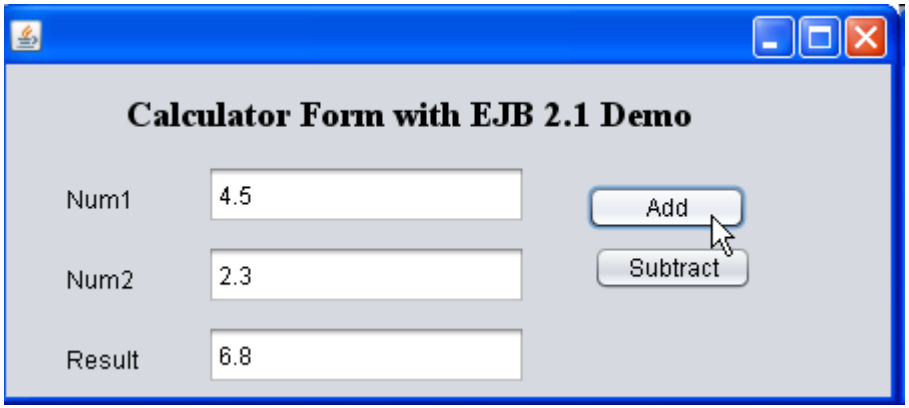
Creating the client application

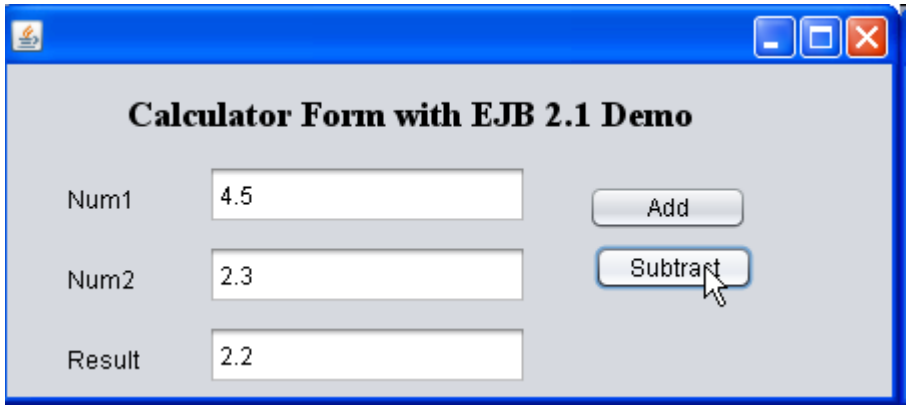
```
123     private void txtSubActionPerformed(java.awt.event.ActionEvent evt) {
124         try {
125             System.setProperty("java.naming.factory.initial",
126                     "org.jnp.interfaces.NamingContextFactory");
127             System.setProperty("java.naming.provider.url", "localhost:1099");
128             Context context = new InitialContext();
129             Object obj = context.lookup("CalJNDI");
130             CalculatorSessionBeanRemoteHome ejbHome = (CalculatorSessionBeanRemoteHome)
131                     PortableRemoteObject.narrow(obj, CalculatorSessionBeanRemoteHome.class);
132             CalculatorSessionBeanRemote ejbObj = ejbHome.create();
133             String n1 = txtNum1.getText();
134             String n2 = txtNum2.getText();
135             double num1 = Double.parseDouble(n1);
136             double num2 = Double.parseDouble(n2);
137             double result = ejbObj.subtract(num1, num2);
138             txtResult.setText(result + "");
        } catch (CreateException ex) {
140             Logger.getLogger(CalculatorForm.class.getName()).log(Level.SEVERE, null, ex);
141         } catch (RemoteException ex) {
142             Logger.getLogger(CalculatorForm.class.getName()).log(Level.SEVERE, null, ex);
143         } catch (NamingException ex) {
144             Logger.getLogger(CalculatorForm.class.getName()).log(Level.SEVERE, null, ex);
145         }
146     }
```

EJB Implementation

Creating the client application







Build the simple enterprise application

DB Utilities Library

DBUtils.java

Source History

```
16      *
17      * @author Trong Khanh
18      */
19     public class DBUtils implements Serializable {
20
21         public static Connection makeConnection(SessionContext current, String dsName) {
            DataSource ds = null;
23
24             ds = (DataSource) current.lookup(dsName);
25
26             Connection con = null;
27             if (ds != null) {
28                 try {
29                     con = ds.getConnection();
30                 } catch (SQLException ex) {
                    ex.printStackTrace();
32                 }
33             }
34
35             return con;
36         }
37     }
```

Build the simple enterprise application

Data Source Description

jboss-ds.xml

Source History

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE datasources
    PUBLIC "-//JBoss//DTD JBOSS JCA Config 6.0//EN"
    "http://www.jboss.org/j2ee/dtd/jboss-ds_6_0.dtd">
<datasources>
    <local-tx-datasource>
        <jndi-name>EJB2DS</jndi-name>
        <connection-url>jdbc:sqlserver://localhost:1433;databaseName=Sinhvien2K8;instanceName=SQL2008</connection-url>
        <driver-class>com.microsoft.sqlserver.jdbc.SQLServerDriver</driver-class>
        <user-name>sa</user-name>
        <password>trongkhanh</password>
    </local-tx-datasource>
</datasources>
```

Build the simple enterprise application

EJB-JAR

ejb-jar.xml

Source | General | CMP Relationships | History

```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <ejb-jar version="2.1" xmlns="http://java.sun.com/xml/ns/j2ee" xmlns:xsi="http://www.w
3       <display-name>Day8EJB2865-ejb</display-name>
4       <enterprise-beans>
5
16          <session>
17              <display-name>RegistrationSessionBeanSB</display-name>
18              <ejb-name>RegistrationSessionBean</ejb-name>
19              <home>sample.session.RegistrationSessionBeanRemoteHome</home>
20              <remote>sample.session.RegistrationSessionBeanRemote</remote>
21              <local-home>sample.session.RegistrationSessionBeanLocalHome</local-home>
22              <local>sample.session.RegistrationSessionBeanLocal</local>
23              <ejb-class>sample.session.RegistrationSessionBean</ejb-class>
24              <session-type>Stateless</session-type>
25              <transaction-type>Container</transaction-type>
26          </session>
27          </enterprise-beans>
```

Build the simple enterprise application

JBoss

jboss.xml

Source History

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE jboss PUBLIC
        "-//JBoss//DTD JBOSS 6.0//EN"
        "http://www.jboss.org/j2ee/dtd/jboss_6_0.dtd">
<jboss>
    <enterprise-beans>
        <session>
            <ejb-name>CalculatorSessionBean</ejb-name>
            <jndi-name>CalJNDI</jndi-name>
            <local-jndi-name>CalLocalJNDI</local-jndi-name>
        </session>
        <session>
            <ejb-name>RegistrationSessionBean</ejb-name>
            <jndi-name>RegJNDI</jndi-name>
            <local-jndi-name>RegLocalJNDI</local-jndi-name>
        </session>
    </enterprise-beans>
</jboss>
```

Build the simple enterprise application

Remote

RegistrationSessionBeanRemoteHome.java

Source History

```
13  * @author Trong Khanh
14  */
15 public interface RegistrationSessionBeanRemoteHome extends EJBHome {
16     RegistrationSessionBeanRemote create() throws CreateException, RemoteException;
17 }
```

RegistrationSessionBeanRemote.java

Source History

```
12  * @author Trong Khanh
13  */
14 public interface RegistrationSessionBeanRemote extends EJBObject{
15
16     boolean checkLogin(String username, String password) throws RemoteException;
17
18 }
```

Build the simple enterprise application

Local

RegistrationSessionBeanLocal.java

Source History

```
11  * @author Trong Khanh
12  */
13 public interface RegistrationSessionBeanLocal extends EJBLocalObject {
14
15     boolean checkLogin(String username, String password);
16
17 }
```

RegistrationSessionBeanLocalHome.java

Source History

```
12  * @author Trong Khanh
13  */
14 public interface RegistrationSessionBeanLocalHome extends EJBLocalHome {
15
16     sample.session.RegistrationSessionBeanLocal create() throws CreateException;
17 }
```

Build the simple enterprise application

Bean

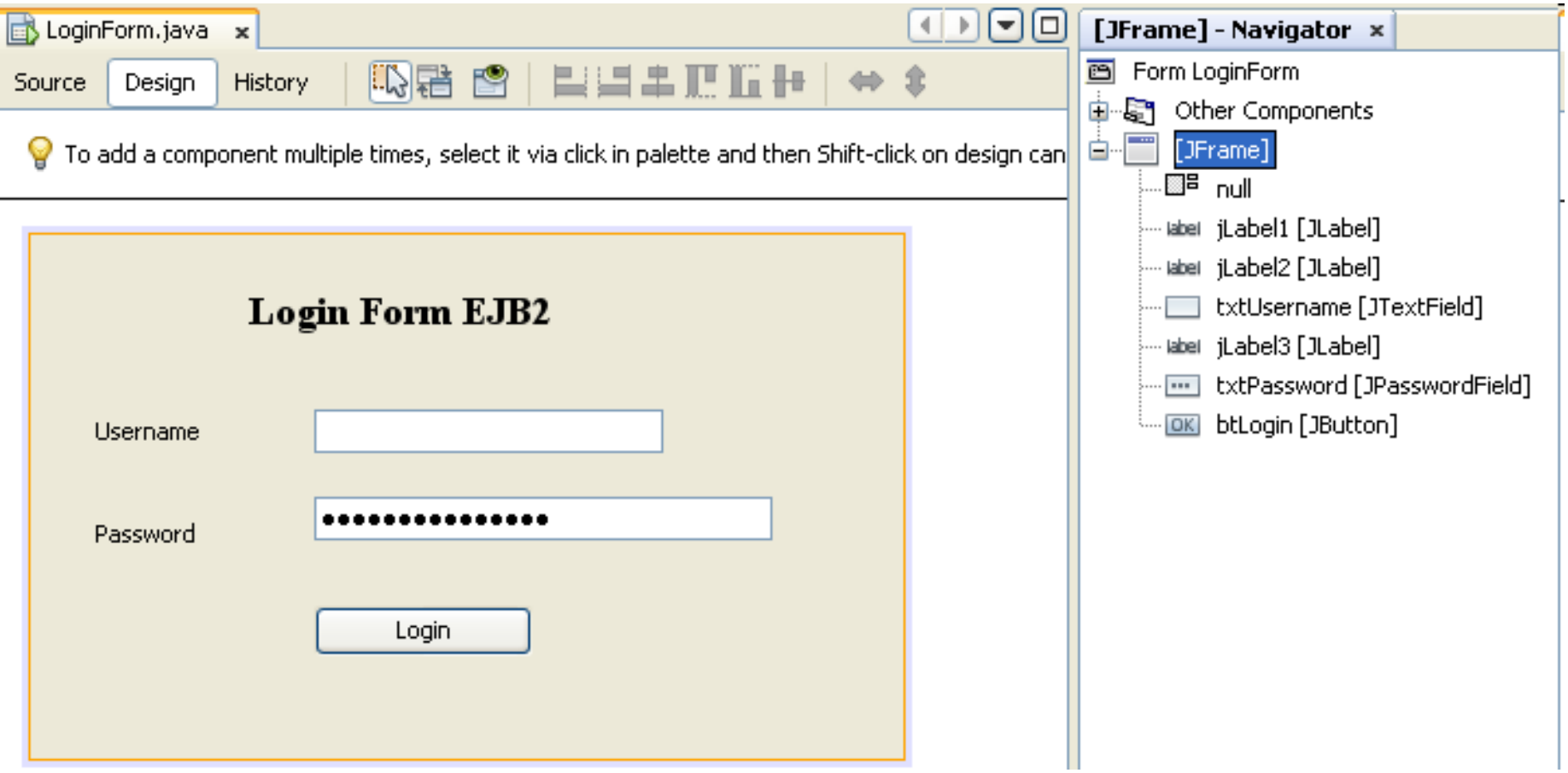
RegistrationSessionBean.java

Source History

```
14      * @author Trong Khanh
15      */
16     public class RegistrationSessionBean implements SessionBean {
17
18         private SessionContext context;
19
20         EJB infrastructure methods. Click the + sign on the left to edit the code.;
52         /**
53          * See section 7.10.3 of the EJB 2.0 specification See section 7.11.3 of the
54          * EJB 2.1 specification
55          */
56         public void ejbCreate() {...6 lines }
62         // Add business logic below. (Right-click in editor and choose
63         // "Insert Code > Add Business Method" or "Web Service > Add Operation")
64
65         public boolean checkLogin(String username, String password) {
66             RegistrationDAO dao = new RegistrationDAO();
67             boolean result = dao.checkLogin(username, password, context, "java:EJB2DS");
68             return result;
69         }
```

Build the simple enterprise application

Client Consume



Build the simple enterprise application

Client Consume

LoginForm.java

Source | Design | History

```
19      * @author Trong Khanh
20      */
21     public class LoginForm extends javax.swing.JFrame {
22
23         /** Creates new form LoginForm ...3 lines */
26         public LoginForm() {...3 lines }
29
30         /** This method is called from within the constructor to initialize the form ...5 line
35         @SuppressWarnings("unchecked")
36         Generated Code
80
81         private void btLoginActionPerformed(java.awt.event.ActionEvent evt) {
82             //0. TRuy cap server
83             System.setProperty("java.naming.factory.initial",
84                     "org.jnp.interfaces.NamingContextFactory");
85             System.setProperty("java.naming.provider.url", "localhost:1099");
86
87             RegistrationSessionBeanRemoteHome homeObj = null;
88             try {
89                 Context context = new InitialContext();
90
91                 Object obj = context.lookup("RegJNDI");
92                 homeObj = (RegistrationSessionBeanRemoteHome) PortableRemoteObject.narrow(obj,
93                         RegistrationSessionBeanRemoteHome.class);
94             } catch (NamingException ex) {
                   ex.printStackTrace();
96             }
```

Build the simple enterprise application

Client Consume

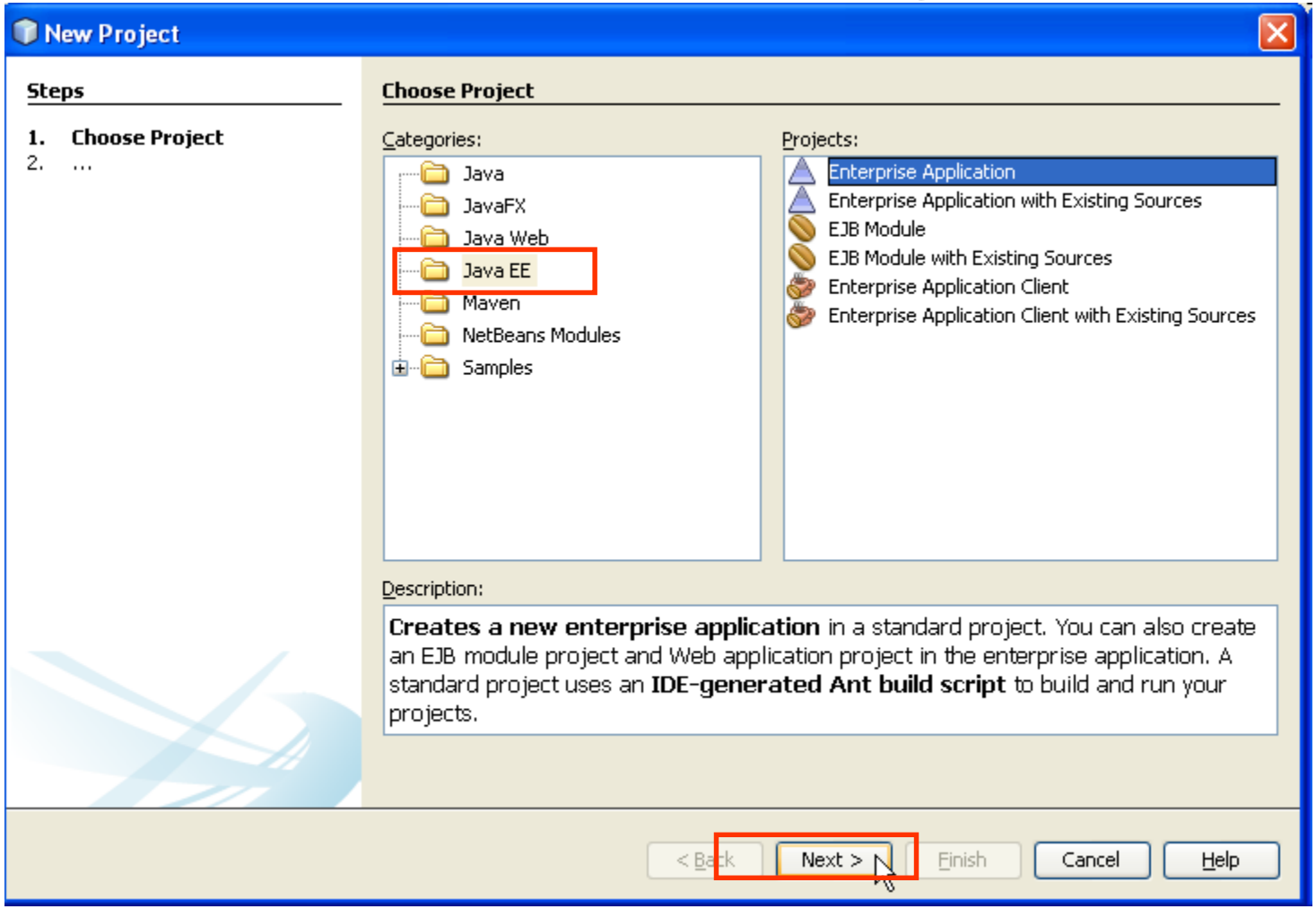
```
98          RegistrationSessionBeanRemote ejbObj = null;
99          try {
100             if (homeObj != null) {
101                 ejbObj = homeObj.create();
102             }
            } catch (CreateException ex) {
                ex.printStackTrace();
105         } catch (RemoteException ex) {
                ex.printStackTrace();
107         }
108
109         if (ejbObj != null) {
110             try {
111                 String username = txtUsername.getText();
112                 String password = new String (txtPassword.getPassword());
113                 boolean result = ejbObj.checkLogin(username, password);
114
115                 if (result) {
116                     JOptionPane.showMessageDialog(this, "Welcome EJB2 Application");
117                 } else {
118                     JOptionPane.showMessageDialog(this, "Invalid username or password");
119                 }
120             } catch (RemoteException ex) {
                    ex.printStackTrace();
122             }
123         }
124     }
```

EJB Implementation

Enterprise Application Development Process

- **Step 1:** Creating a new Enterprise Application project (EJB and Web Client – ear file)
- **Step 2:** Creating the new corresponding bean depending on your purpose.
- **Step 3:** Building/ Modifying the business/callback methods on Beans
- **Step 4:** Mapping the JNDI to beans
- **Step 5:** Creating the GUI to consumes EJB on web modules
- **Step 6:** Building and Deploying Enterprise application on Application Server
- **Step 7:** Executing the Enterprise Application

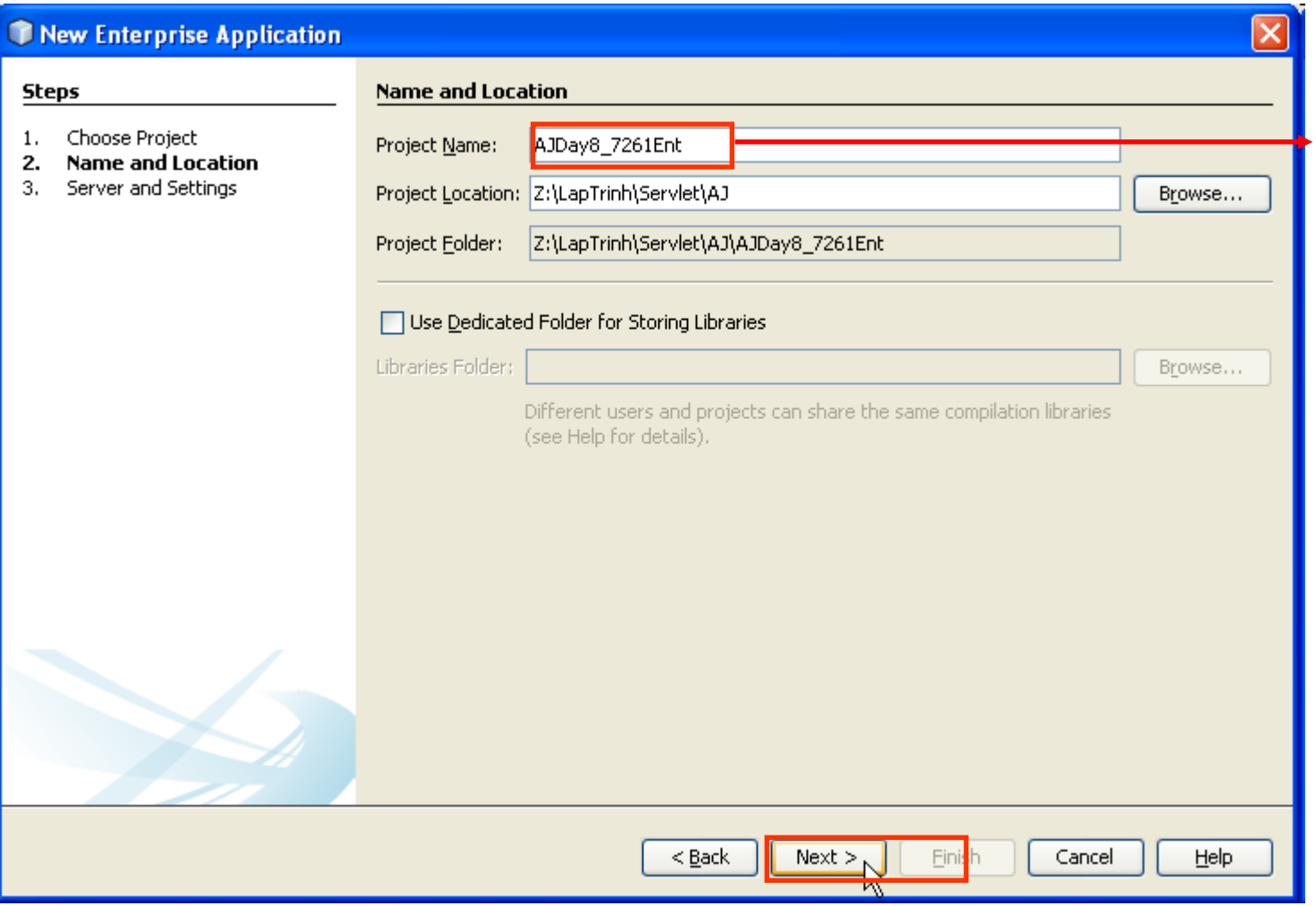
EJB Implementation
Creating



- Click Next Button.
- Then type the Project Name, then click Next button

EJB Implementation

Creating

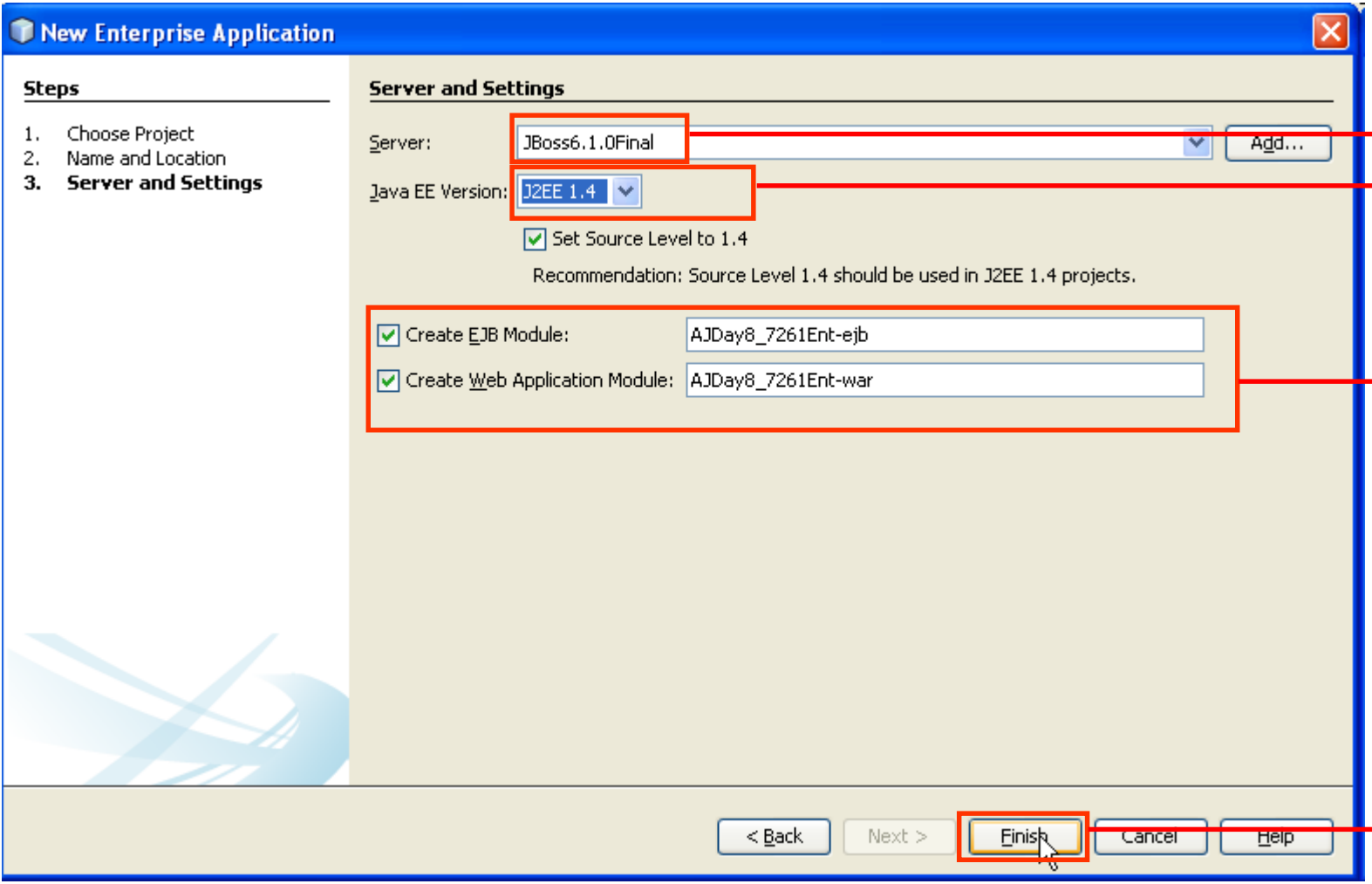


Fill your project name

- Click Next Button.

EJB Implementation

Creating



Choose the Jboss 4.2.3

Choose J2EE 1.4

Don't change anything that is default by tools

Click Finish Button

EJB Implementation
Creating

- AJDay8_7261Ent
  - Java EE Modules
    - AJDay8_7261Ent-war.war
    - AJDay8_7261Ent-ejb.jar
  - Configuration Files
    - MANIFEST.MF
    - application.xml
    - jboss-app.xml
  - Server Resources
- AJDay8_7261Ent-ejb
  - Source Packages
    - <default package>
  - Libraries
    - JDK 1.7 (Default)
    - JBoss6.1.0Final
  - Enterprise Beans
  - Configuration Files
    - MANIFEST.MF
    - ejb-jar.xml
    - jboss.xml
  - Server Resources
- AJDay8_7261Ent-war
  - Web Pages
  - Source Packages
  - Libraries
    - AJDay8_7261Ent-ejb - dist/AJDay8_7261Ent-ejb.jar
    - JDK 1.7 (Default)
    - JBoss6.1.0Final
  - Configuration Files
  - Server Resources

z:\LapTrinh\Servlet\AJ\AJDay8_7261Ent*.*

Name
[..]
[AJDay8_7261Ent-ejb]
[AJDay8_7261Ent-war]
[nbproject]
[setup]
[src]
build

application.xml

Source | History

```xml
<?xml version="1.0" encoding="UTF-8"?>
<application version="1.4" xmlns="http://java.sun.com/xml/ns/j2ee"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/application_1_4.xsd">
  <display-name>AJDay8_7261Ent</display-name>
  <module>
    <web>
      <web-uri>AJDay8_7261Ent-war.war</web-uri>
      <context-root>/AJDay8_7261Ent-war</context-root>
    </web>
  </module>
  <module>
    <ejb>AJDay8_7261Ent-ejb.jar</ejb>
  </module>
</application>
```

EJB Implementation

Next Steps

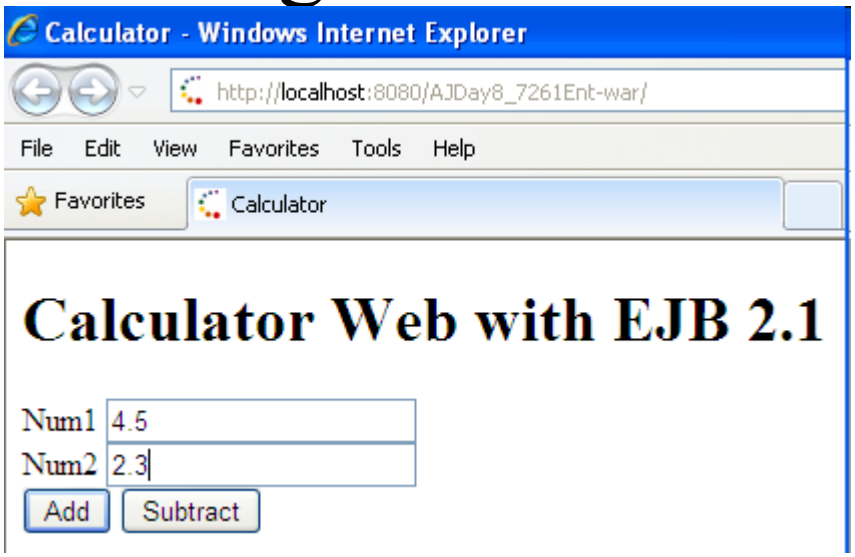
- **Step 2:** Creating the new corresponding bean
- **Step 3:** Building/ Modifying the business/callback methods on Beans
- **Step 4:** Mapping the JNDI to beans

Creating stateless bean as whole steps in previous tutorials in EJB Development process on the Xxx-ejb module

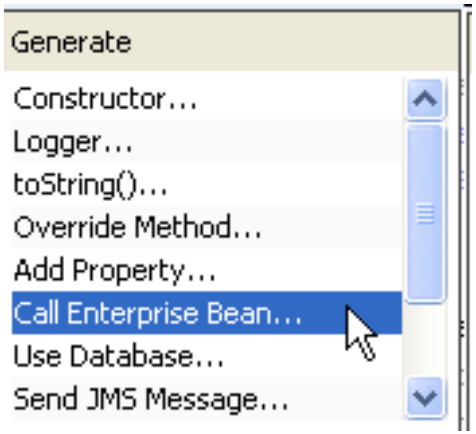
EJB Implementation

Creating GUI with Web Page and consumes

- Creating the GUI application

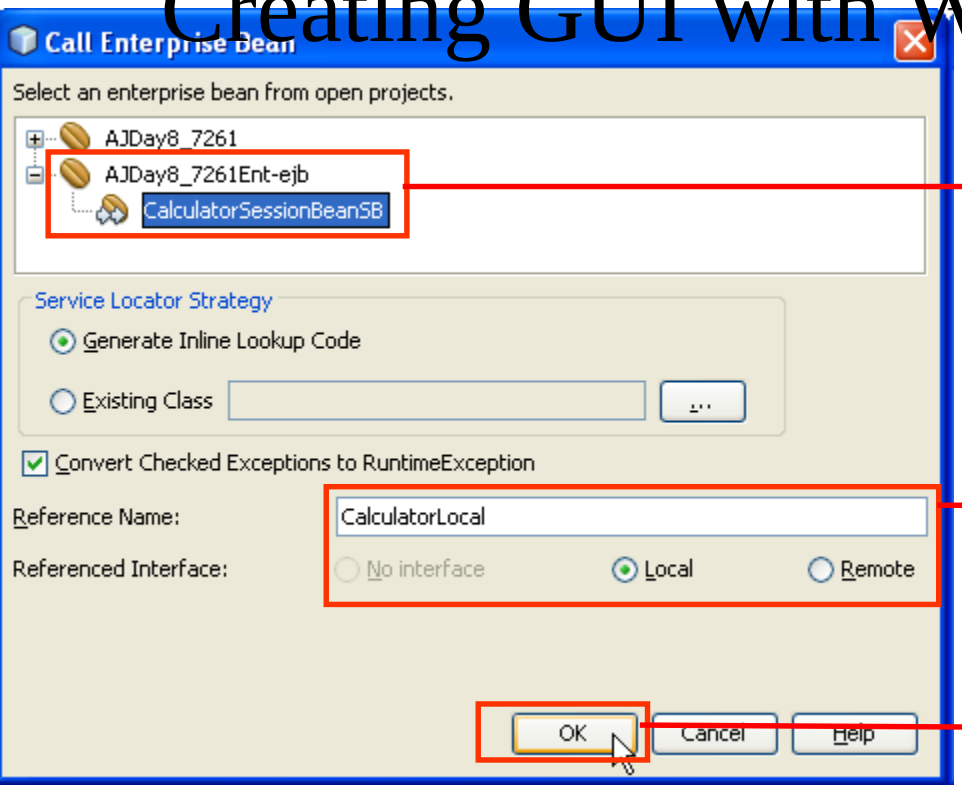


- Creating the Servlet to process and consume the EJB
  - Creating the reference to the EJB on the coding by right click on code
  - Then choose Insert Code, click Call Enterprise Bean …



EJB Implementation

Creating GUI with Web Page and consumes



Choose the appropriate bean

Modify the Reference Name, then choose the scope reference of the Bean

Click Ok Button

```
151     private CalculatorSessionBeanLocal lookupCalculatorSessionBeanLocal() {
152         try {
153             Context c = new InitialContext();
154             CalculatorSessionBeanLocalHome rv = (CalculatorSessionBeanLocalHome)
155                     c.lookup("CalLocalJNDI");
156             return rv.create();
157         } catch (NamingException ne) {
158             Logger.getLogger(getClass().getName()).log(Level.SEVERE, "exception caught", ne);
159             throw new RuntimeException(ne);
160         } catch (CreateException ce) {
161             Logger.getLogger(getClass().getName()).log(Level.SEVERE, "exception caught", ce);
162             throw new RuntimeException(ce);
163         }
164     }
```

Modify the Reference Name that is named in jboss.xml at <[local-]jndi-name> tag

EJB Implementation

Creating GUI with Web Page and consumes

web.xml

Source | General | Servlets | Filters | Pages | References | Security | History

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <web-app version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee" xmlns:xsi="http:
3      <servlet>
7      <servlet-mapping>
11     <session-config>
16     <welcome-file-list>
17     <ejb-local-ref>
18         <ejb-ref-name>CalculatorLocal</ejb-ref-name>
19         <ejb-ref-type>Session</ejb-ref-type>
20         <local-home>sample.session.CalculatorSessionBeanLocalHome</local-home>
21         <local>sample.session.CalculatorSessionBeanLocal</local>
22         <ejb-link>AJDay8_7261Ent-ejb.jar#CalculatorSessionBean</ejb-link>
23     </ejb-local-ref>
24 </web-app>
```

EJB Implementation

Creating GUI with Web Page and consumes

- Modifying the code in servlet as following

ProcessServlet.java

```
26    * @author Trong Khanh
27    */
28   public class ProcessServlet extends HttpServlet {
29       /**...*/
36       protected void processRequest(HttpServletRequest request, HttpServletResponse response)
37               throws ServletException, IOException {
38           response.setContentType("text/html;charset=UTF-8");
39           PrintWriter out = response.getWriter();
40           try {
41               String button = request.getParameter("btAction");
42               if (button.equals("Add")) {
43                   String n1 = request.getParameter("txtNum1");
44                   String n2 = request.getParameter("txtNum2");
45                   double num1 = Double.parseDouble(n1);
46                   double num2 = Double.parseDouble(n2);
47                   //Su dung JNDI de tim Initial Context Factory
48                   Context context = new InitialContext();
49                   //Tim Home Object
50                   Object obj = context.lookup("CalJNDI");
51                   //Xac dinh kieu cua Home Obj
52                   CalculatorSessionBeanRemoteHome home =
53                           (CalculatorSessionBeanRemoteHome) PortableRemoteObject.narrow(obj,
54                           CalculatorSessionBeanRemoteHome.class);
55                   //tao EJB obj tu home obj
56                   CalculatorSessionBeanRemote ejbObj = home.create();
57                   //goi business method tren ejb obj
58                   double result = ejbObj.add(num1, num2);
59                   out.println("Add " + n1 + " + " + n2 + " = " + result);
60               }
```

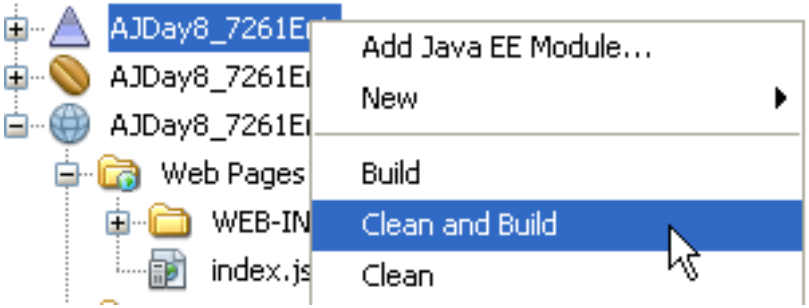
EJB Implementation

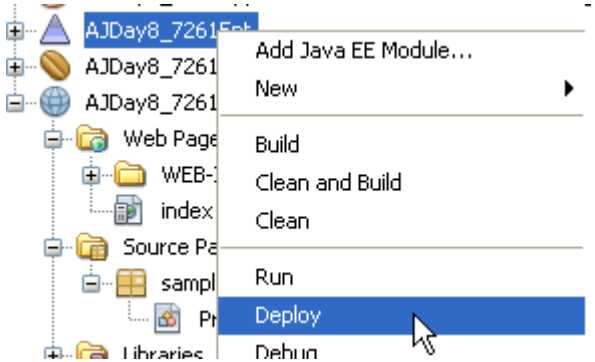
Creating GUI with Web Page and consumes

```
60          } else if (button.equals("Subtract")) {
61              String n1 = request.getParameter("txtNum1");
62              String n2 = request.getParameter("txtNum2");
63
64              double num1 = Double.parseDouble(n1);
65              double num2 = Double.parseDouble(n2);
66
67              Context context = new InitialContext();
68              Object obj = context.lookup("CalLocalJNDI");
69              CalculatorSessionBeanLocalHome home =
70                      (CalculatorSessionBeanLocalHome) obj;
71              CalculatorSessionBeanLocal local = home.create();
72              double result = local.subtract(num1, num2);
73              out.println("Sub " + n1 + " - " + n2 + " = " + result);
74          }
```

EJB Implementation

Building, Deploying, and Executing





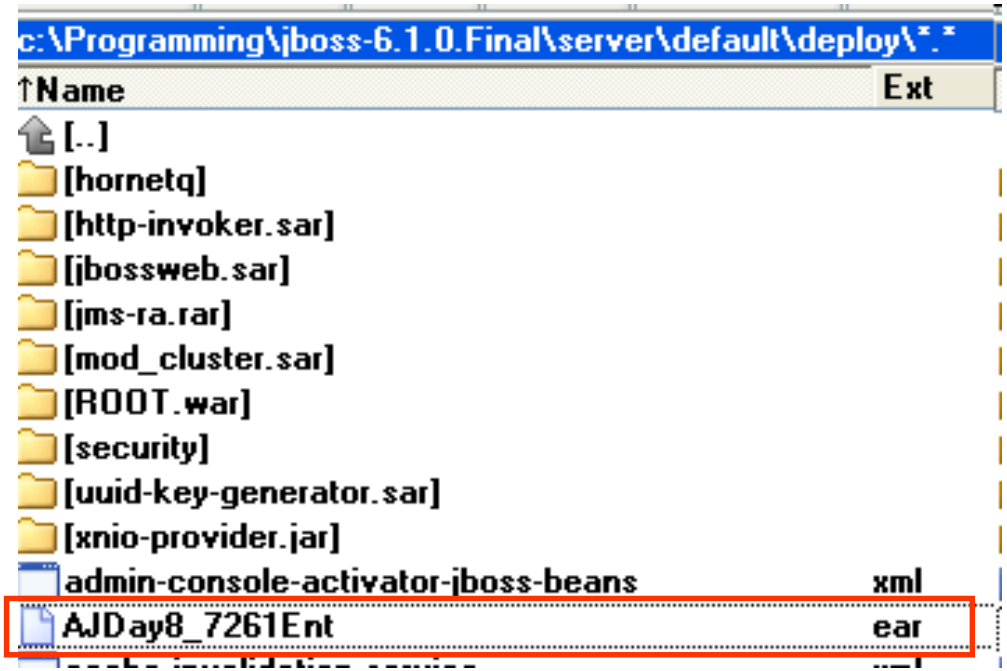
Output

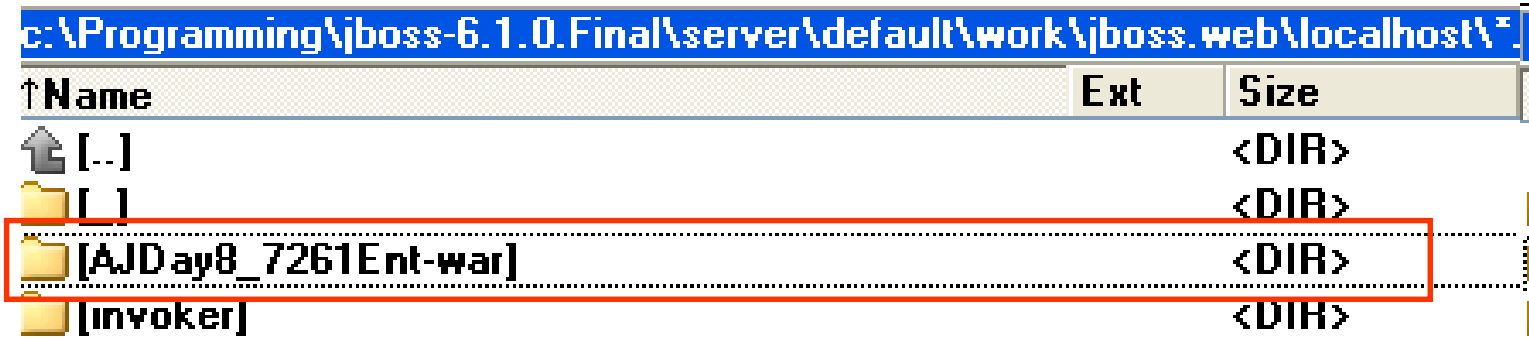
JBoss6.1.0Final × AJDay8_7261Ent (clean,dist) ×

```
21:54:42,593 INFO  [org.jboss.ejb.deployers.EjbDeployer] installing bean: ejb/AJDay8_7261Ent-ejb.jar#CalculatorSessionBean,uid20721493
21:54:42,593 INFO  [org.jboss.ejb.deployers.EjbDeployer]   with dependencies:
21:54:42,593 INFO  [org.jboss.ejb.deployers.EjbDeployer]   and supplies:
21:54:42,593 INFO  [org.jboss.ejb.deployers.EjbDeployer]       jndi:AJDay8_7261Ent/CalculatorSessionBean/sample.session.CalculatorSessionBeanLocal
21:54:42,593 INFO  [org.jboss.ejb.deployers.EjbDeployer]       jndi:CalJNDI
21:54:42,593 INFO  [org.jboss.ejb.deployers.EjbDeployer]       jndi:AJDay8_7261Ent/CalculatorSessionBean/sample.session.CalculatorSessionBeanRemote
21:54:42,593 INFO  [org.jboss.ejb.deployers.EjbDeployer]       jndi:CalLocalJNDI
21:54:43,468 INFO  [org.jboss.ejb.EjbModule] Deploying CalculatorSessionBean
21:54:44,156 INFO  [org.jboss.ejb.plugins.local.BaseLocalProxyFactory] Bound EJB LocalHome 'CalculatorSessionBean' to jndi 'CalLocalJNDI'
21:54:44,203 INFO  [org.jboss.proxy.ejb.ProxyFactory] Bound EJB Home 'CalculatorSessionBean' to jndi 'CalJNDI'
21:54:44,531 INFO  [org.jboss.web.tomcat.service.deployers.TomcatDeployment] deploy, ctxPath=/AJDay8_7261Ent-war
```

EJB Implementation

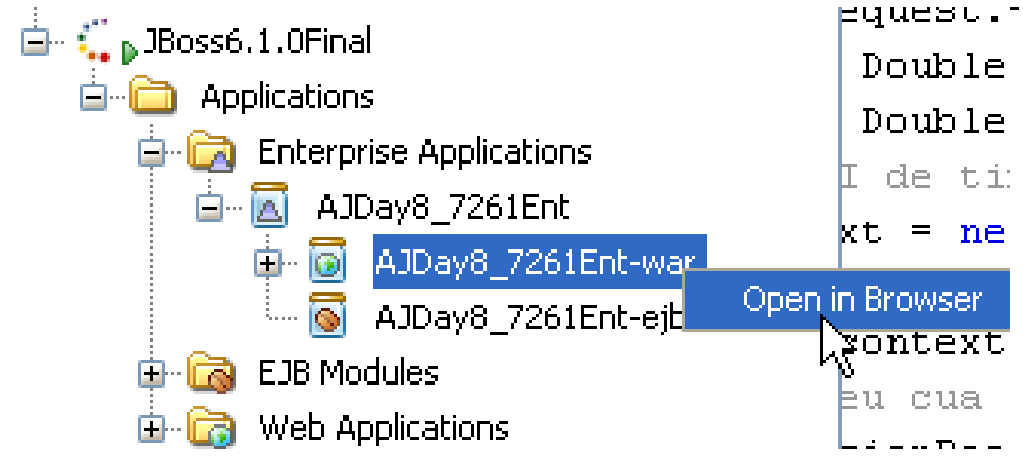
Building, Deploying, and Executing

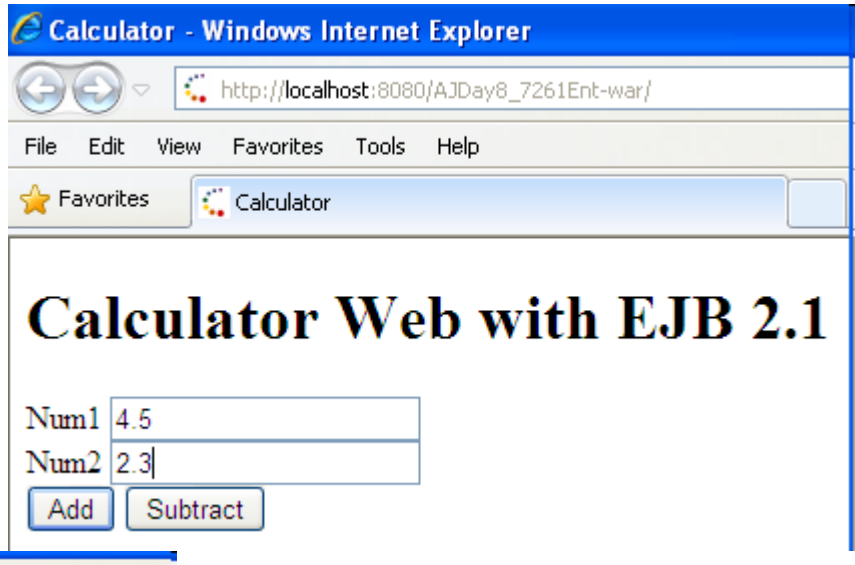


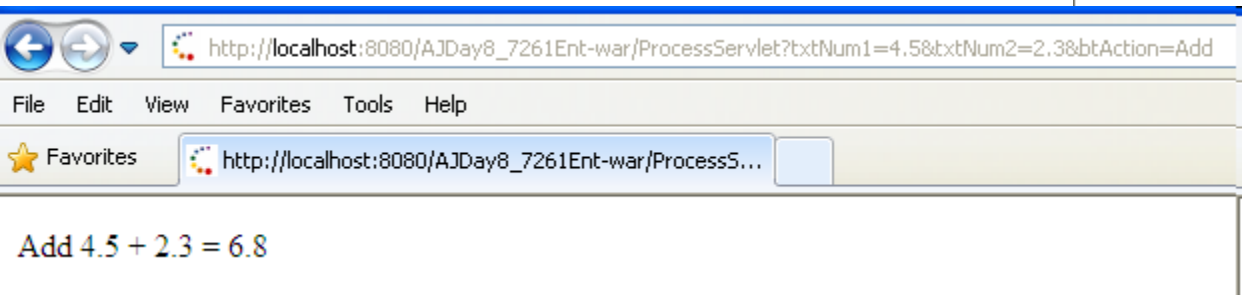


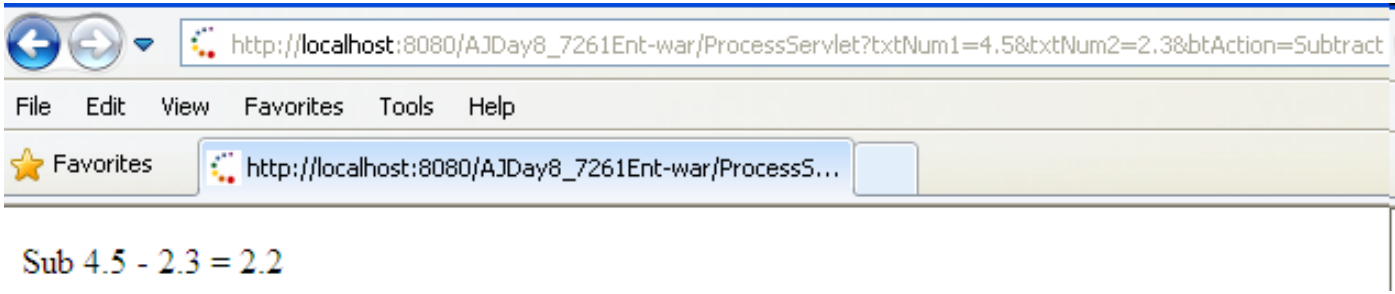
EJB Implementation

Building, Deploying, and Executing









Build the simple enterprise application

Web Requirements

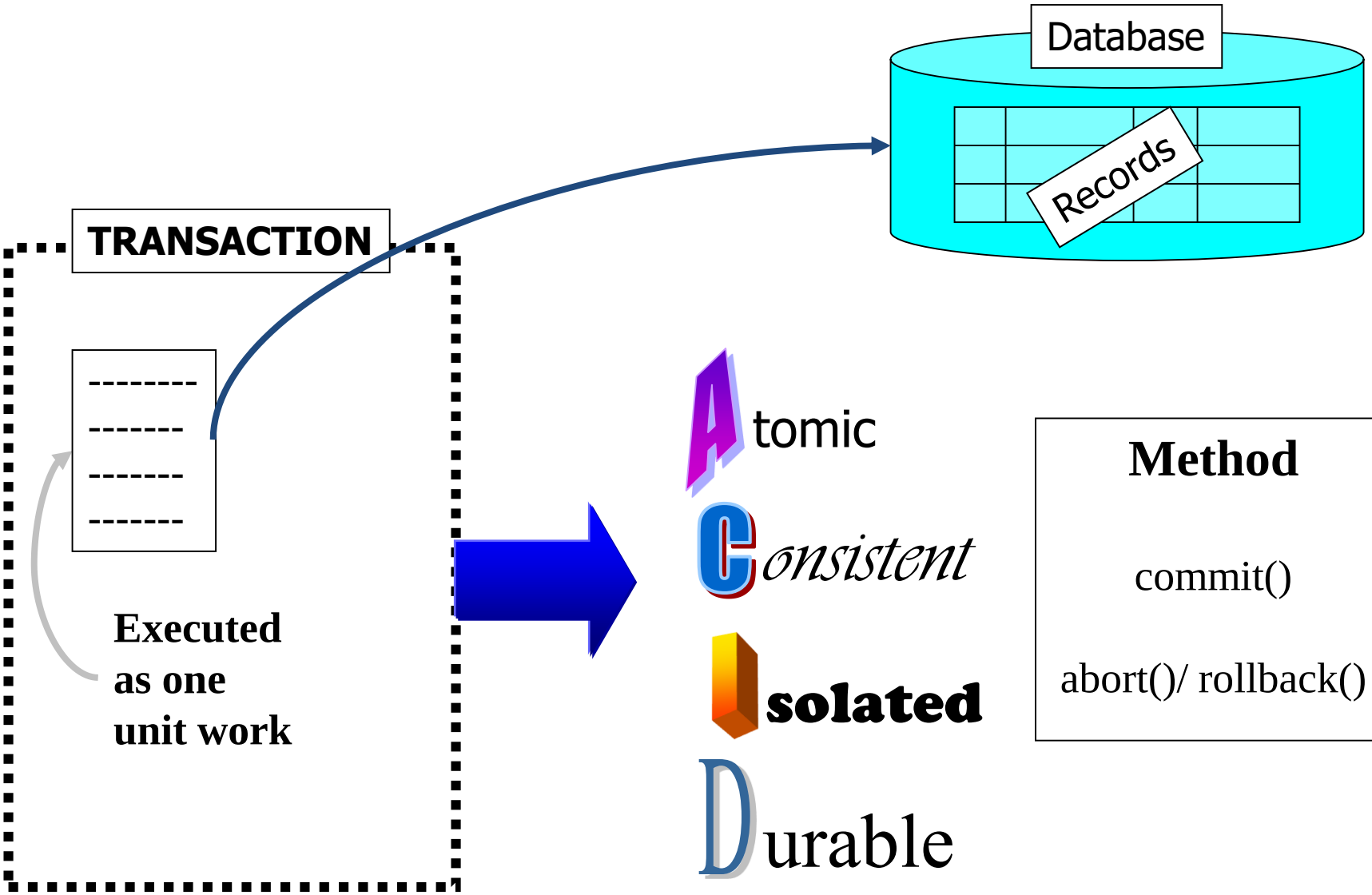
- **Take your self. It is easy**

Appendix – EJB Container

- **Acts** as an **interface between** an **enterprise bean and client**
- **Provides** following **services**
  - Security
  - Transaction Management
  - Persistence
  - Life Cycle management
  - Remote Client Connectivity
- **Responsible** for **providing several APIs**
  - J2SE API
  - EJB Standard Extension
  - JDBC Standard Extension
  - JNDI Standard Extension
  - JMS Standard Extension
  - JavaMail Standard Extension (for Sending mail only)
  - JAXP (Java API for XML Processing)
  - JTA Standard Extension (Only UserTransaction interface)

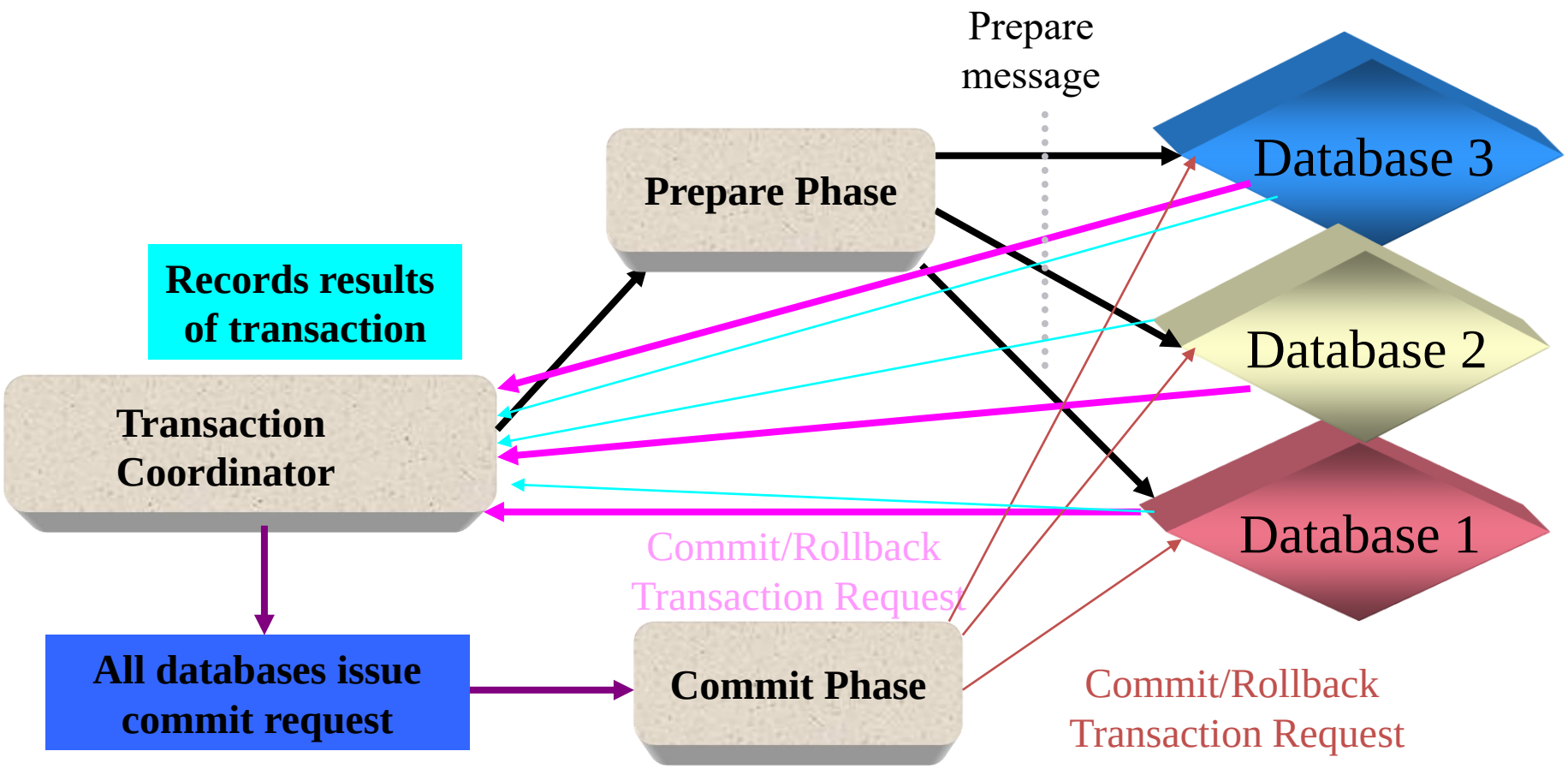
Enterprise Java Beans

Transactions



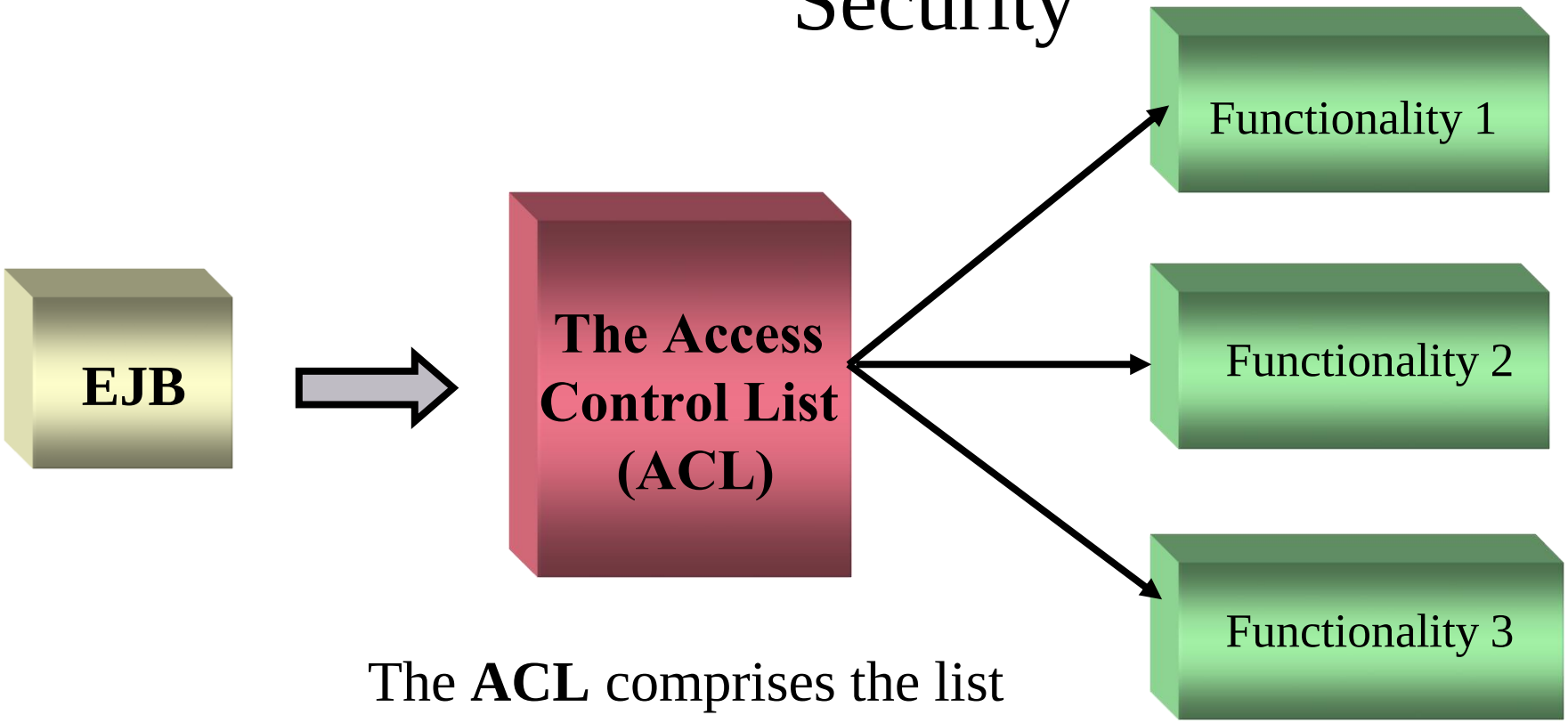
Enterprise Java Beans

Two phase Commit Protocol



Enterprise Java Beans

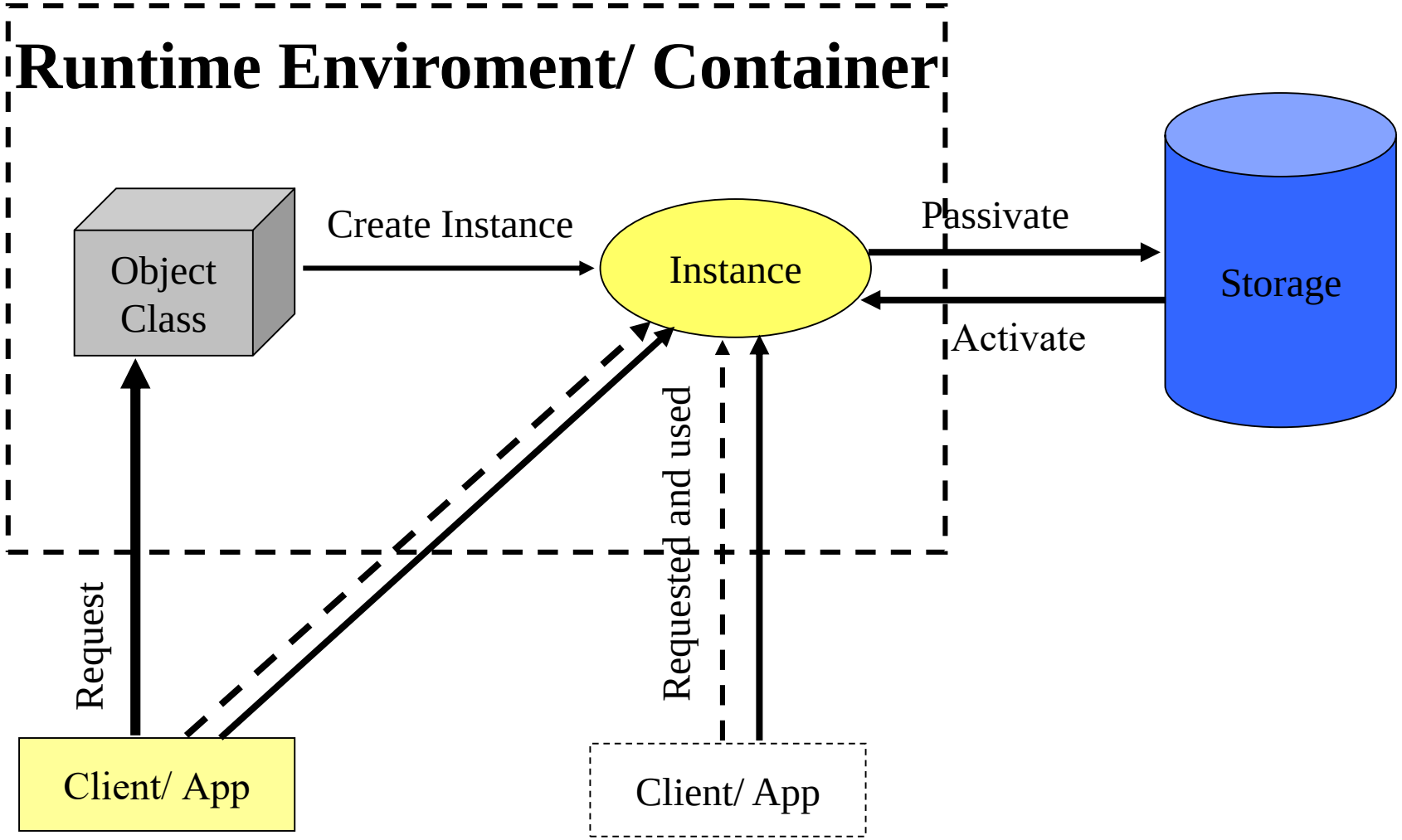
Security



The **ACL** comprises the list of persons who are allowed to access particular sections of functionality.

Enterprise Java Beans

Persistent

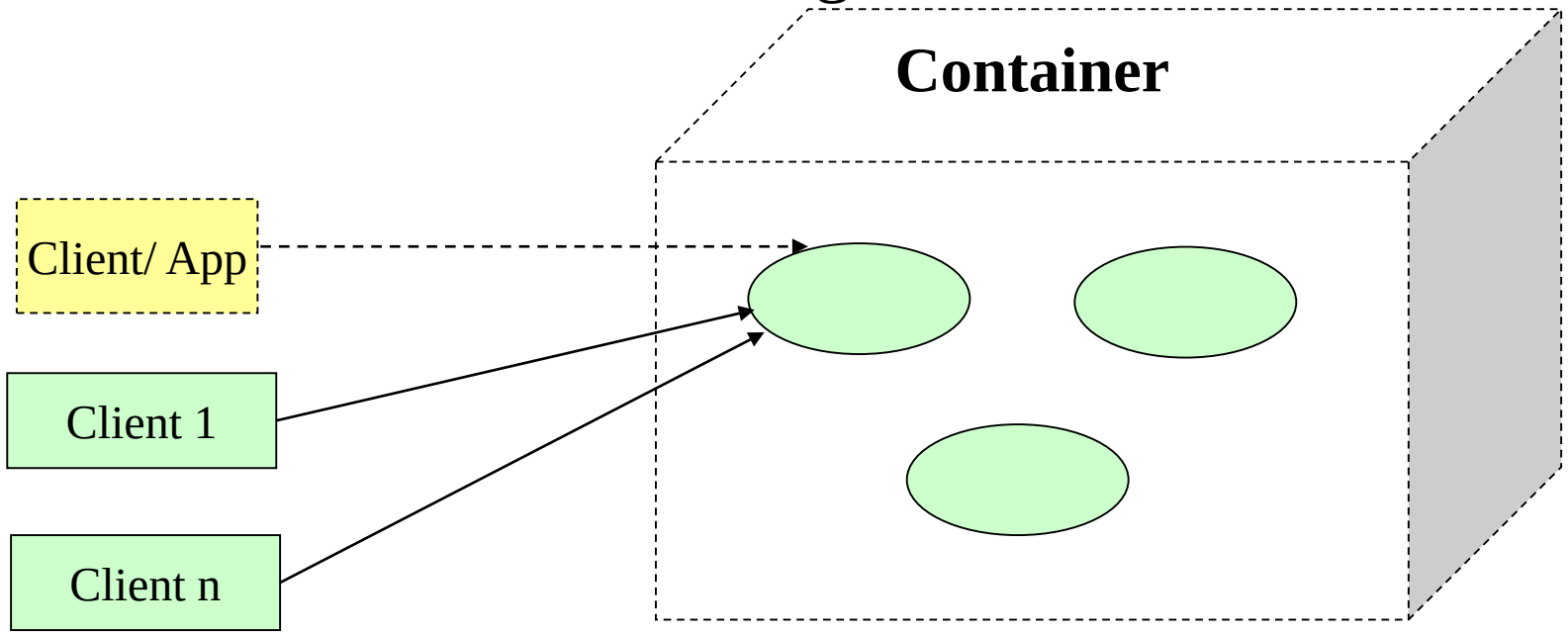


Persistence can be defined as saving the state of an object to a constant storage.

Enterprise Java Beans

Management of Multiple Instance

- Instance Passivation
- Instance Pooling
  - **Advantages**: reduces the memory allocation and garbage-collection cycles
- Database Connection Pooling



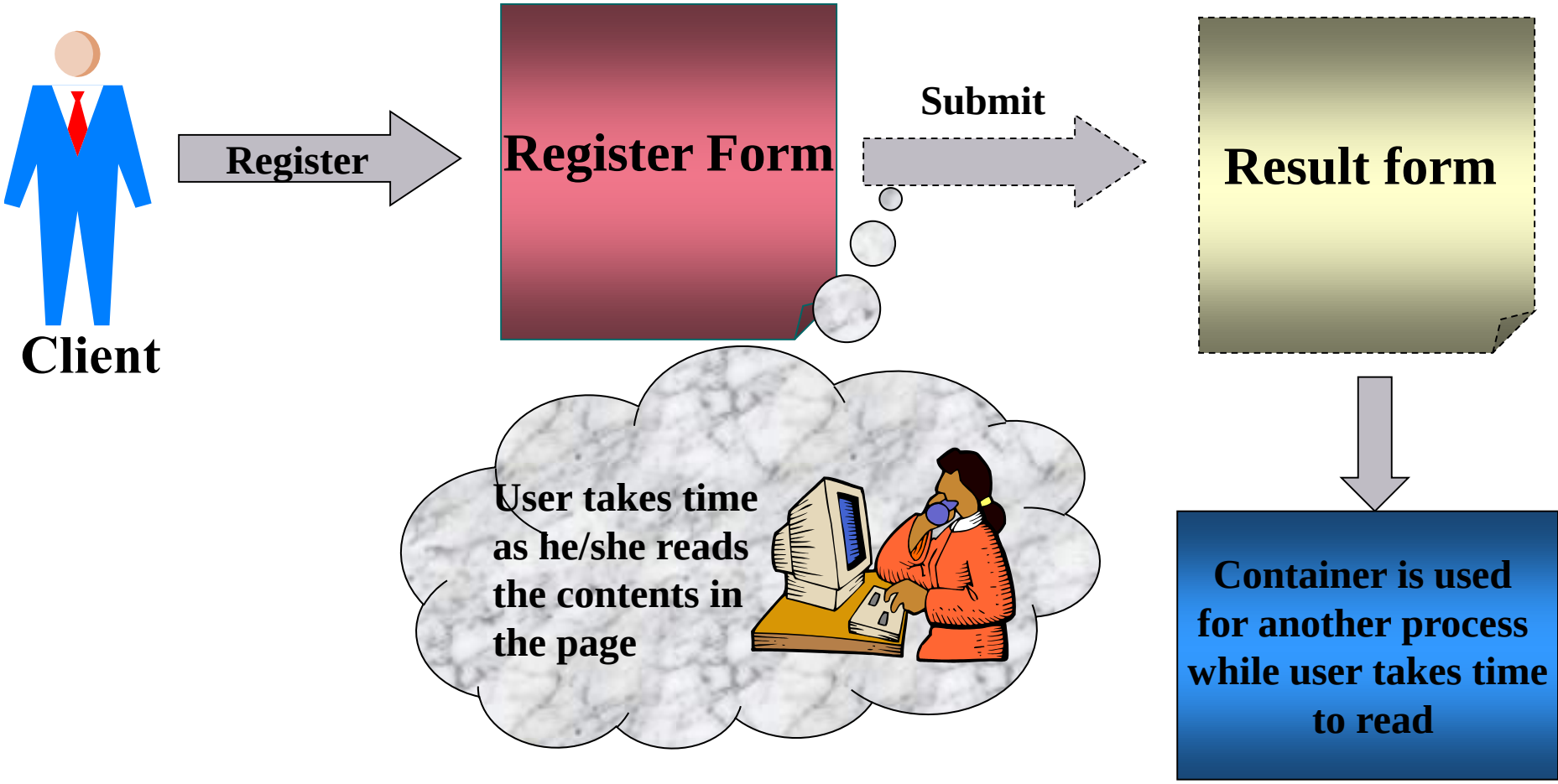
Enterprise Java Beans

Resource and lifecycle Management

- Management of resources **enhances the scalability** of a multi-tier architecture.
- The container **provides resource-management services** for resources such as:
  - Threads
  - Socket Connections
  - Database Connections
- EJB Container **responsible the life cycle** of the bean (control the life of the bean).
  - **Notes: The life time of the bean is managed by EJB server**
- EJB Container **instantiates, destroys and reuses** the beans required.
- EJB Container **supports instance pooling.**

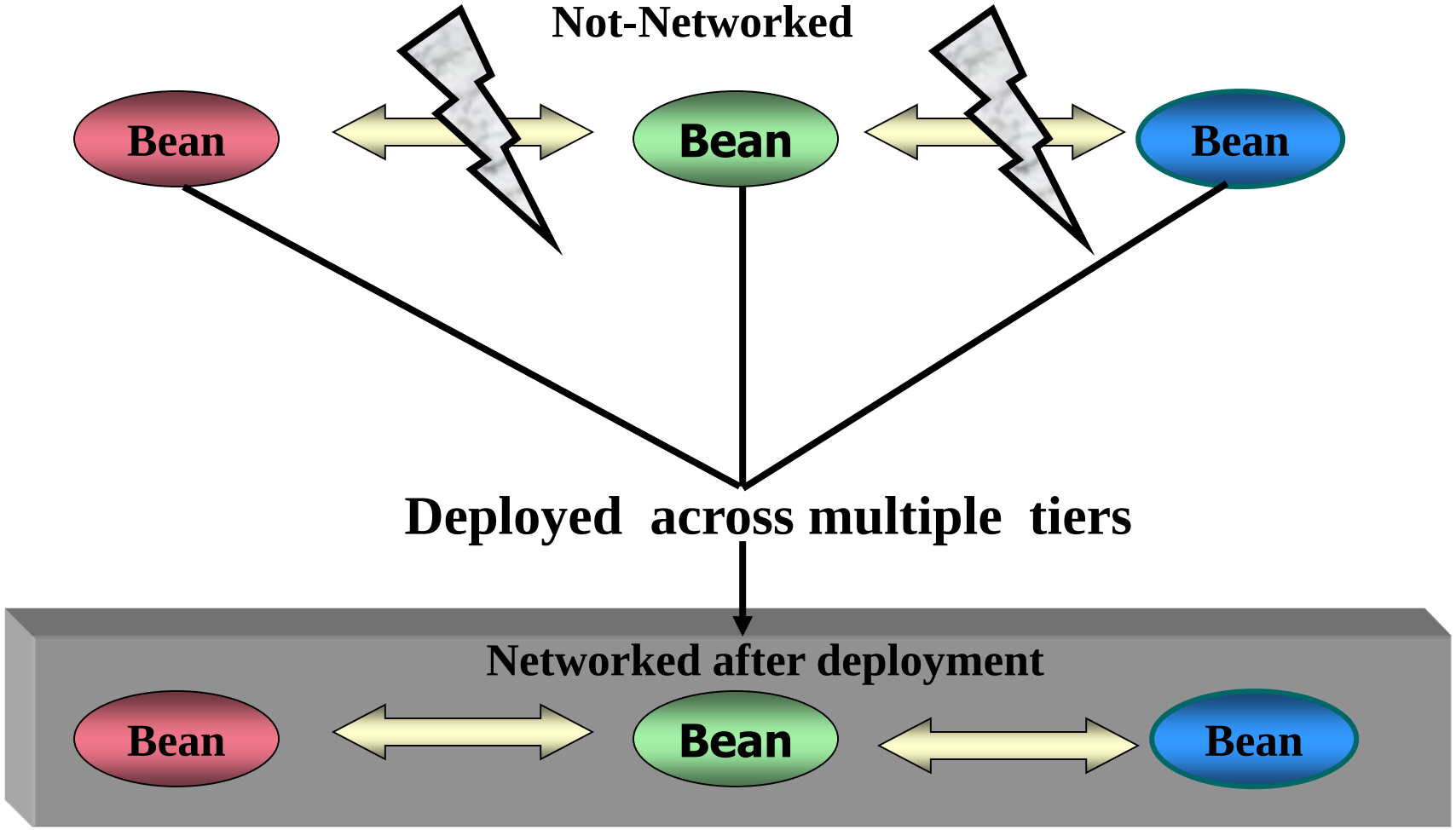
Enterprise Java Beans

State Management



Enterprise Java Beans

Remote Accessibility



Enterprise Java Beans

Location Transparency

- Clients **do not know where** the components are, and whether these components are local or remote
- **Advantages**
  - **Reusable**
  - Vendors can provide value additions in terms of
    - Ability to perform maintenance on a system connected to a network because location transparency **allows a different system provide components for a particular client**
    - **Install new software**
    - **Upgrade the components on a system**
  - When a **system crashes**, the **requests** are **redirected** to **another** system **without the client getting to know about the crash.**

Enterprise Java Beans

Components of EJB

- The Enterprise Java Bean is a **server-side component** that is **employed on a distributed multi-tier environment**.
- EJB does **not allow multithreading** (single thread)
- Important Object of EJB is Bean
- **Types**
  - **Session Bean** – **Represents business process without** having **persistent storage** mechanism
    - Stateless Session Bean
    - Stateful Session Bean
  - **Entity Bean** – **Persists across multiple sessions** and **multiple clients** & Having **persistence storage** mechanism
    - Bean-managed Persistence [BMP]
    - Container-managed Persistence [CMP]
  - **Message-driven Bean** – **Asynchronous messaging between components** of EJB.

Appendix

J2EE Terminologies in J2EE design patterns

- **Service Locator**
  - **Implement** and **encapsulate** service and component lookup
  - **Hides** the implementation details of the lookup mechanism and encapsulates related dependencies

  →**Transparently locate business components and services in a uniform manner (ex: EJB Home Interface)**
- **Business Delegate**
  - **Encapsulate** access to a business service
  - **Hides** the implementation details of the business service, such as lookup and access mechanisms

  →**Hide clients from the complexity of remote communication with business service components**
- **Abstract Factory**
  - Provides a way to **encapsulate a group of individual factories** that have a common theme

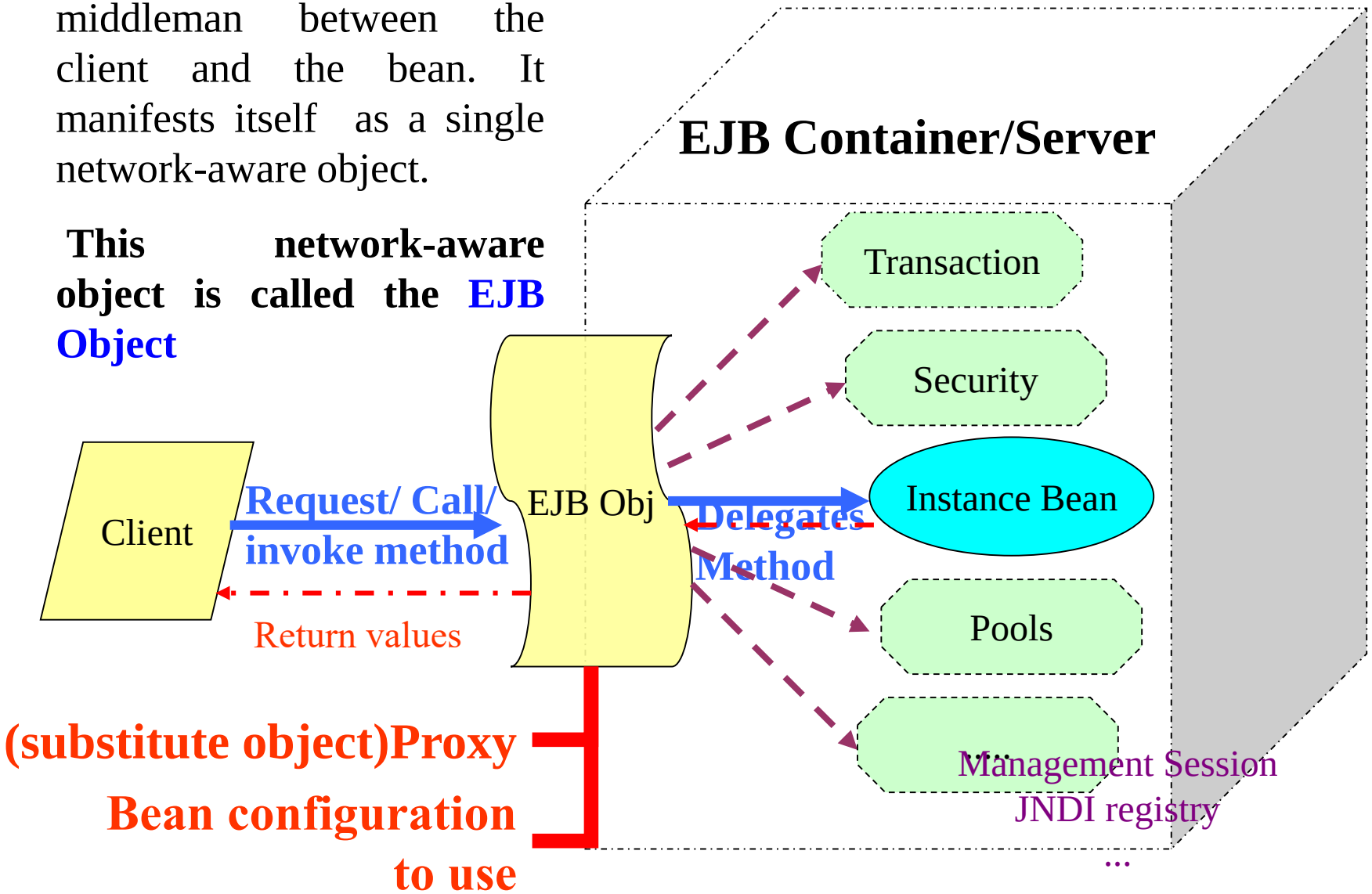
  →**Separates the details of implementation of a set of objects from their general usage (ex: EJBHome interface)**

What Constitutes an EJB?

EJB Objects

The container is the middleman between the client and the bean. It manifests itself as a single network-aware object.

This network-aware object is called the EJB Object



What Constitutes an EJB?

EJB Objects

- Interface **javax.ejb.EJBObject**

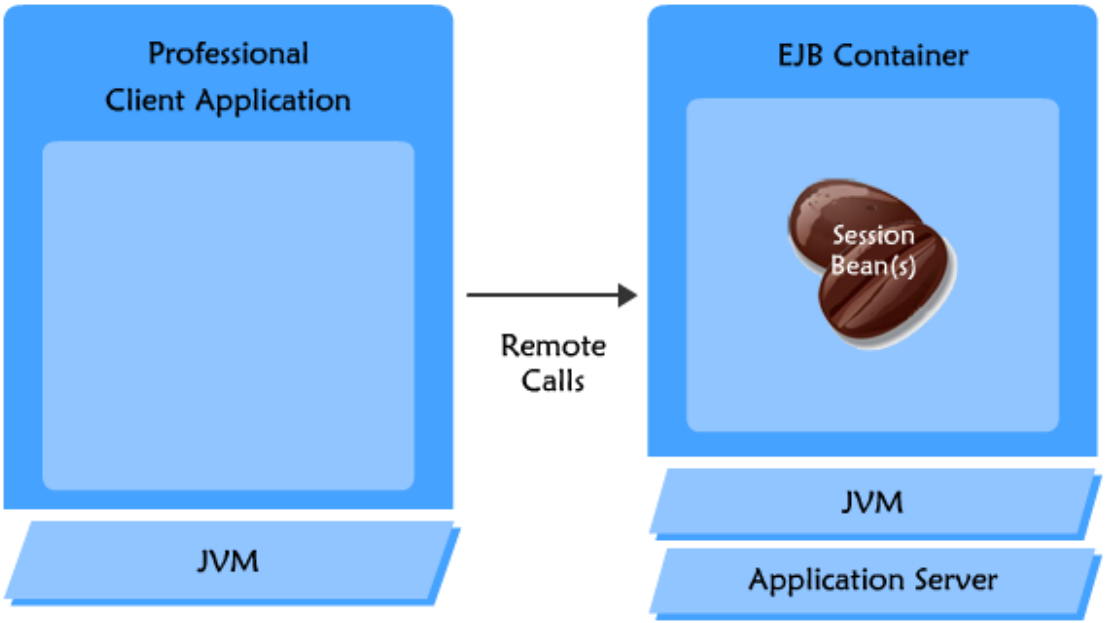
Methods	Descriptions
getEJBHome()	**Retrieves** the **reference** to the corresponding Home Object
getPrimaryKey()	**Return** the **Primary Key** for EJB Object (Entity Bean)
remove()	**Destroy** EJB Object (**delete** the bean from the underlying persistent store, means delete a record on DB – Entity Bean)
getHandle()	Obtain the **handle** (is a persistent reference to the EJB Object) for the EJB Object
isIdentical()	Checks whether two EJB Objects are similar

- Relationship between Java RMI and EJB Objects
  - public interface javax.ejb.EJBObject **extends java.rmi.Remote** (*The physical location of remote object is hidden from the Client RMI*)
  - Can be called from a different JVM
  - **Offers Location Transparency** (Portability of Client Code)

What Constitutes an EJB?

Remote Interface

- **Is used when** the **client application runs** on a **separate JVM than** the **one that** is **used to run** the **Session beans** in an EJB Container
  - The **method invocation** in remote business interfaces are **received from networked clients**
  - The method **parameters and** the **return values** are **copied** and is **known** as **call-by-value**

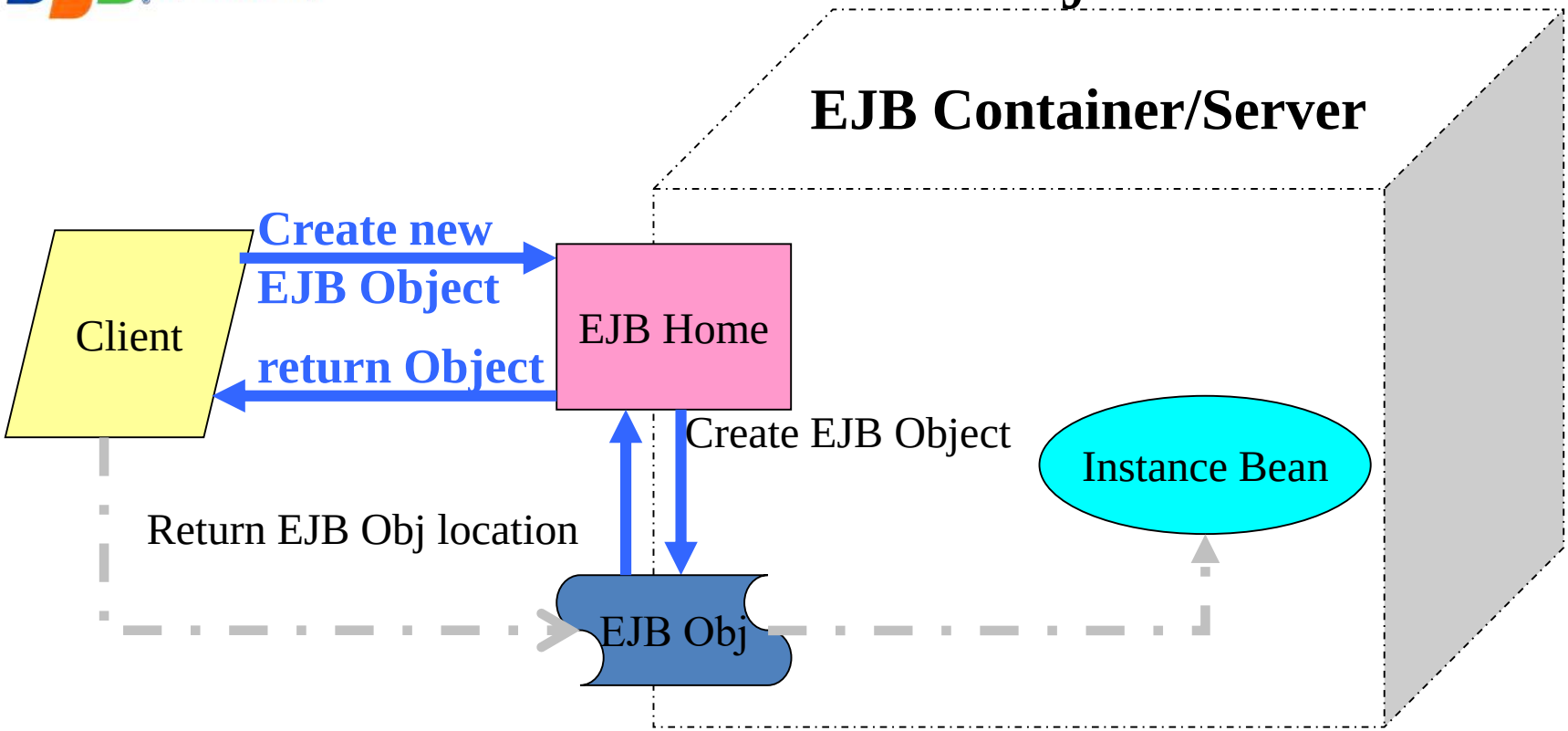


What Constitutes an EJB?

Local Interface

- EJB 2.0 can **expose** their **methods to clients through new Local** Interface
- Standard Java interface which **allows the beans to expose its methods** to other bean **reside within the same container** (local clients)
- **Eleminate** the **overhead** of the **remote method call** (java.rmi.RemoteException)
- Is used when the application **uses the same JVM to run both the client application** and the **Session beans**
  - The method **parameters** and the **return values** are **not copied** and **hence**, it is known as **call-by-reference**
  - Speed up in processing and efficiency
- **Not inherit from RMI** (extends javax.ejb.EJBLocalObject)

Home Objects



- Client code will request for an object from **the EJB Object Factory**, which know as **the home object** (instantiates EJB Object)
- **Responsibilities**
  - Create (Instantiate) EJB Objects
  - Initial information for EJB Object s
  - Find or search for existing EJB Objects(Entity Bean)
  - Remove EJB Objects (deletes the bean from the underlying persistent store)
  - Select EJB Objects (Entity Bean)

What Constitutes an EJB?

Home Objects

- Interface javax.ejb.EJBHome (extends java.rmi.Remote)

Methods	Descriptions
getEJBMetaData()	**Retrieve information** about **EJB** (Beans' information) that are **being worked on**. The information received is encapsulated in the EJBMetaData object, which returns the method.
remove()	**Destroy EJB Object** following - **Passing** the **javax.ejb.Handle object**, which **remove EJB Object** that is based on the already retrieved EJB Handle - **Passing a primary key** to **remove beans** (**one record**) from the **underlying persistent store.**

Appendix

J2EE Terminologies in J2EE design patterns

- **Transfer Object**
  - A serializable class that **groups related attributes, forming a composite value**
  - A class is used as **the return type of a remote business method**
  - Clients receive instances of this class by calling coarse-grained business methods, and then locally access the fine-grained values within the transfer object. Fetching multiple values in one server roundtrip decreases network traffic and minimizes latency and server resource usage.
- **Session Façade**
  - A higher-level business component contains and centralizes complex interactions between lower-level business components
  - Is implemented as a session enterprise bean.
    - It **provides** clients with **a single interface** for the functionality of an application or application subset.
    - It also **decouples lower-level business components** from one another, making designs more flexible and comprehensible

APPLICATION/EJB SERVER

- Provides **many services**
  - **Network connectivity** to the container
  - **Instance Passivation** – Temporarily swap out a bean from memory storage
  - **Instance Pooling** – Multiple clients share same instance
  - **Database Connection Pooling** – Contains a set of database connection
  - **Precached Instances** – Maintains cache, which contains information about the state of the EJB
- **Other** services
  - Runtime Environment
  - Support the containers interaction
  - Process and Thread Management
  - Receive and process requests
  - System Resource Management

Enterprise Java Beans

Session Beans

- **Survive only as long as** the client exists.
- Are created solely in response to a call made by the client.
- Are **used to implement business logic, business rules and the workflow**
  - **Ex**: Check login, computation, Document process, book tickets
- **Are not shareable between clients** (only one client can deal with that particular session bean)
- **Stateless Session Bean**
  - **Single** Request
  - **Stateless**
  - **Redirect** the **others bean** when the errors occur.
  - **Ex**: check Login
- **Stateful Session Bean**
  - **Multiple** requests (The life cycle is very complex)
  - **Keep track**
  - **Persistence**
  - **Ex**: Shopping Cart

Enterprise Java Beans

Entity Beans

- DB Model: Entity beans are the **object representations of the underlying data and provide access to data**.
- The components **are persistent**.
- **Have a long life** because they can be reconstructed by reading the data back from the permanent DB.
- Data in Relational DB can be treated as **real objects** and an entire chunk of data from DB can be read at once into an entity bean component. (Allow the transformation of the data in the DB into Java Objects)
- EJB Container **synchronous** between EJB and DB
- **BMP –** Bean managed Persistent Entity Bean
  - **The developer has to write** the code (**CRUD**) to interpret the fields stored in the memory to an underlying DB
- **CMP –** Container managed Persistent Entity Bean
  - The **container perform** all the operations
  - The **developer** has to **describe** that needs to persist and inform the container
  - The developer **concentrates** the **business processes**.

Enterprise Java Beans

Message Driven Beans

- New type in EJB Version 2.0
- Process **messages asynchronously** (The bean acts a a message listener)
- **Communication** between software components/ application (onMessage(Message msg) method)
- **Similar t**o Stateless session bean
- Created and Controller by Container.
- Do **not** have **a home and remote interface**.
- Support both container managed (which may delivered message within a transaction context) and bean managed transactions.

JNDI

Overview

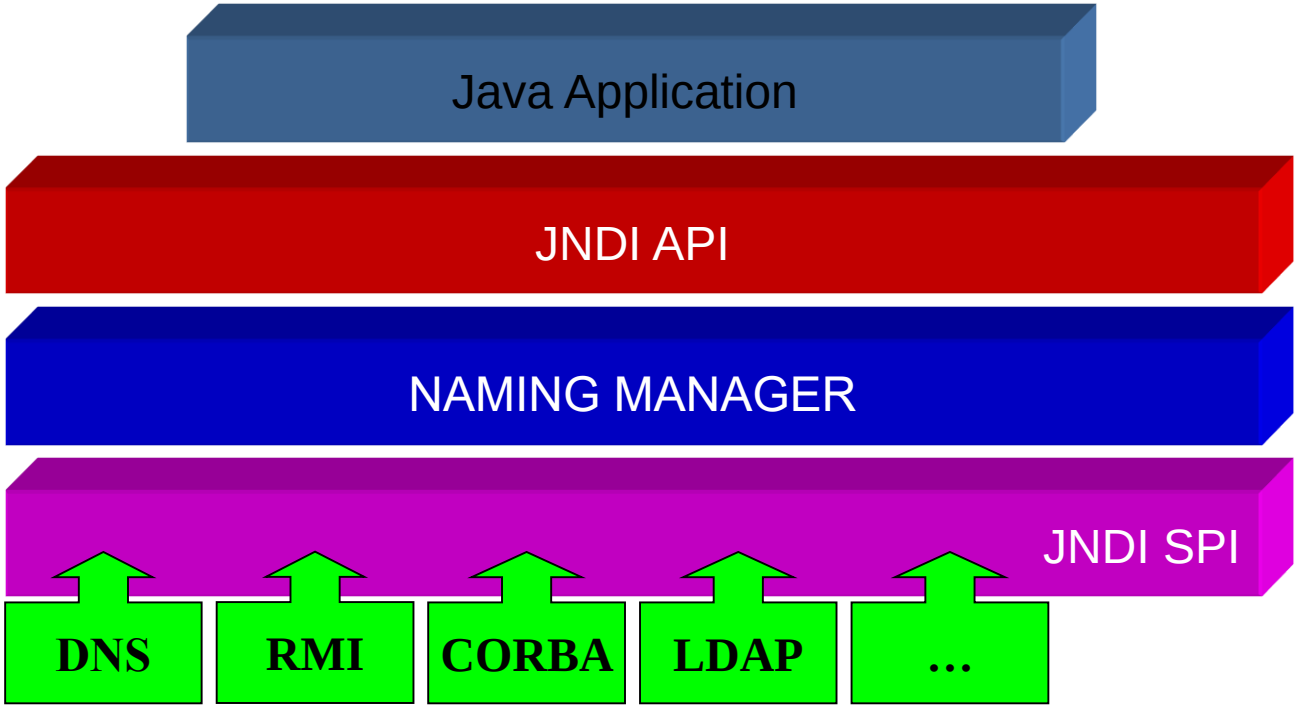
- **A naming service** (which has its own set of rules for creating valid names) allows you **finding an object** in a system based on the **name associated with the object** which is called "**binding**".
- **A directory service** is an **extension of a naming service** (an object is also associated with a name, which can be look up, and allowed to have attributes)
- Java Naming and Directory Interface (**JNDI**) is a **specification** for accessing naming and directory services
- Java Naming and Directory Interface **provides the naming and directory functionality** to Java applications.
- Provides **a standard interface to locate** the components, users, networks, and services placed **across the network**.
- **Bridges the gap between directory services** and makes it possible for the developer to write portable naming and directory services
- JNDI **abstracts the code from a directory service and allows the user to plug in a different directory services**. (without changing the service code)

JNDI

Overview

- JNDI provides **javax.naming.*** interface
- JNDI separate two parts
  - **JNDI API**
  - **JNDI SPI**
- Naming Concepts of JNDI
  - **Atomic**: It's a simple and basic name. Ex: Windows
  - **Compound**: the collection of one or more atomic names.
    - Ex: C:\Windows\System32
  - **Composite**: A name has multiple naming system.
    - Ex: http://localhost:8080/JSP/index.html

JNDI
Architecture

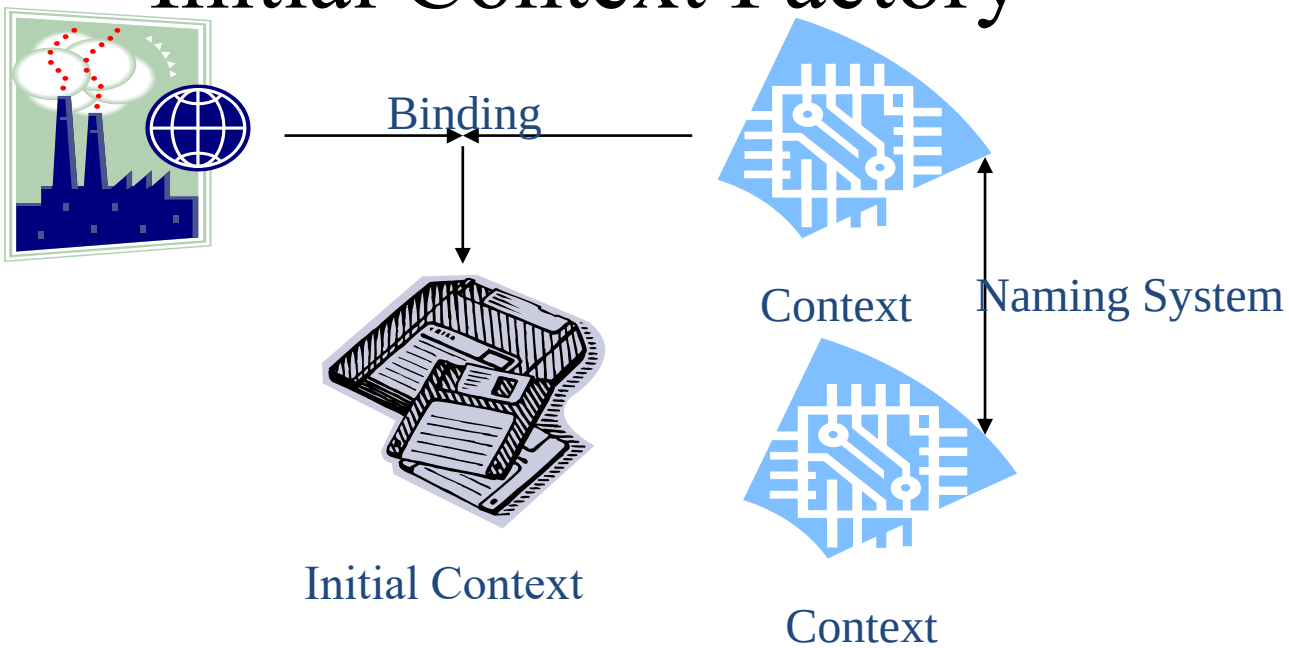


Accessor Naming Directory services through APIs

allows Naming and Directory vendors can plugin transparency

JNDI

Initial Context Factory



- Initial Context Factory is the **point** where **all naming and directory operations** are **first performed**.
- When the initial context **is acquired**, all information **pertaining** to this must be provided to JNDI.
- The **internal storage** of JNDI **emulates tree data structure**. Each InitialContext **acts like an internal node** and **each reference to the resources acts like the leaves**
- The directory context or directory object is another type of context. It is used to **define methods for inspecting** and **modifying attributes** associated with a *directory object*.