The New Enterprise Java Beans

EJB 3.x

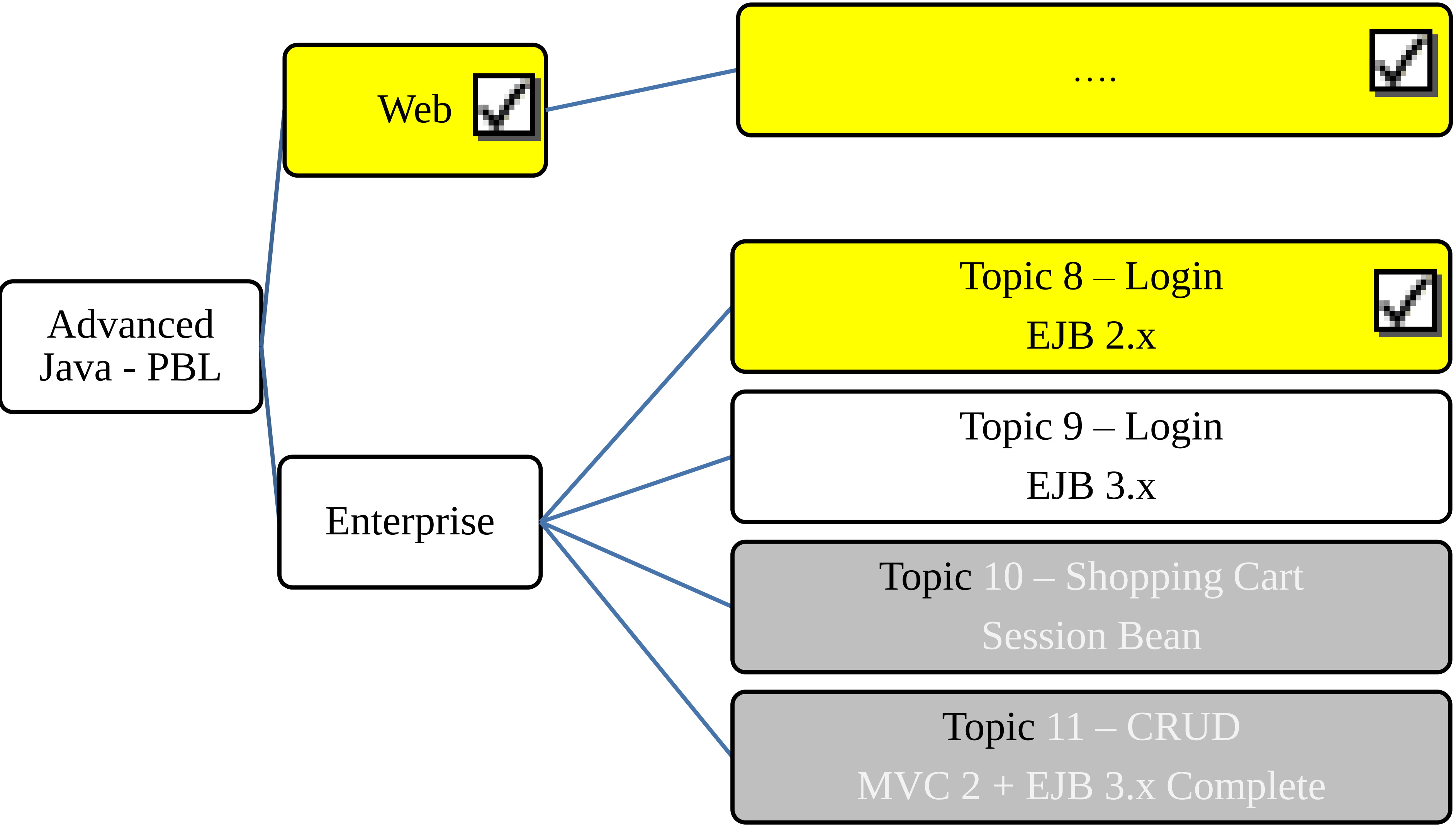
Review

- **Enterprise Java Beans**
  - J2EE/JavaEE Architecture
  - Logical Architecture
  - EJB Container (Transaction, Security, Persistent, Management, …)
  - Objects: Session Beans (Stateless, Stateful), Entity Beans (BMP, CMP), Message Driven Beans
  - Components: Component interface (Remote interface, Local interface), Home interface (Home interface, Local Home interface), Bean class, EJB deployment descriptors, server deployment descriptor, DB mapping descriptors, …
  - Implementation
- **JNDI**
  - API, SPI, Naming Manager
  - Context Factory, Initial Context Factory

Objectives

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

Objectives



Advanced Java - PBL

Web ☑

.... ☑

Enterprise

Topic 8 – Login
EJB 2.x ☑

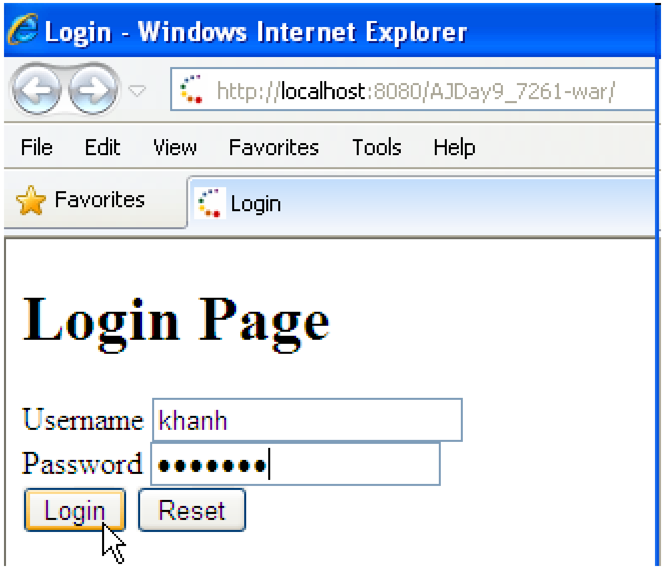
Topic 9 – Login
EJB 3.x

Topic 10 – Shopping Cart
Session Bean

Topic 11 – CRUD
MVC 2 + EJB 3.x Complete

Build Simple Application with EJB3

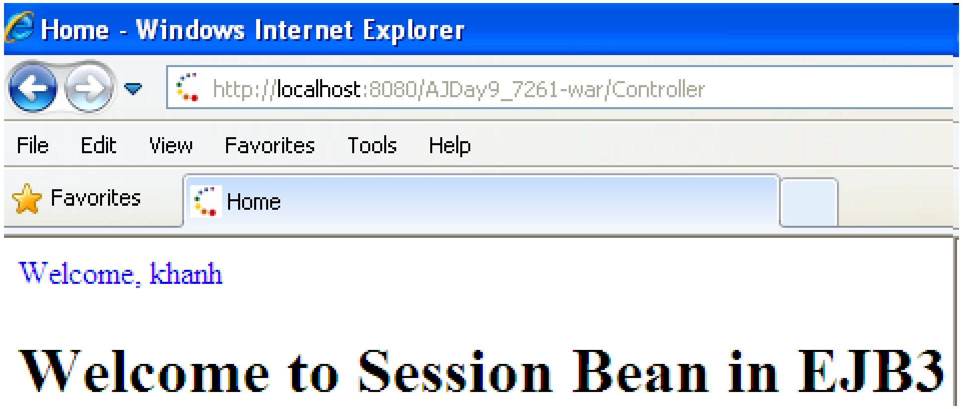
Requirement


Login - Windows Internet Explorer
http://localhost:8080/AJDay9_7261-war/
File Edit View Favorites Tools Help
Favorites
Login
Login
Reset

Login Page

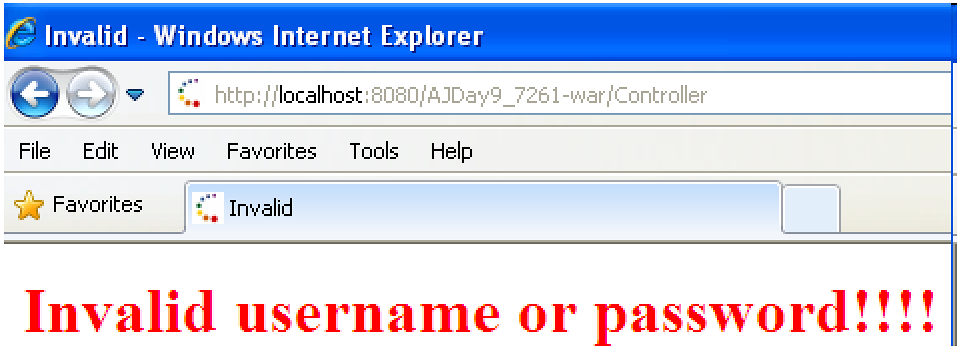
Username khanh

Password •••••••


Home - Windows Internet Explorer
http://localhost:8080/AJDay9_7261-war/Controller
File Edit View Favorites Tools Help
Favorites
Home

Welcome, khanh

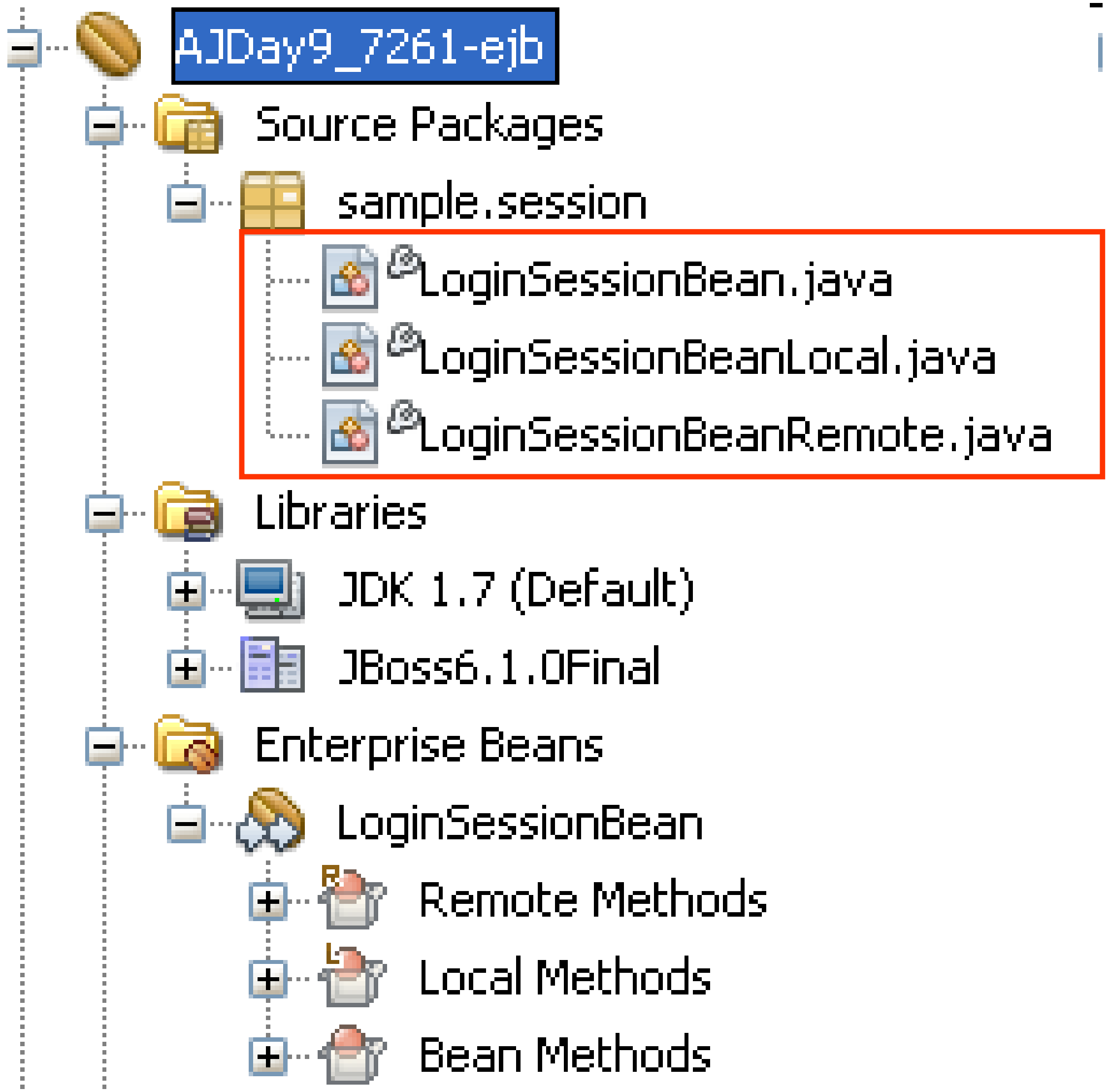
Welcome to Session Bean in EJB3


Invalid - Windows Internet Explorer
http://localhost:8080/AJDay9_7261-war/Controller
File Edit View Favorites Tools Help
Favorites
Invalid

Invalid username or password!!!!

Build Simple Application with EJB3

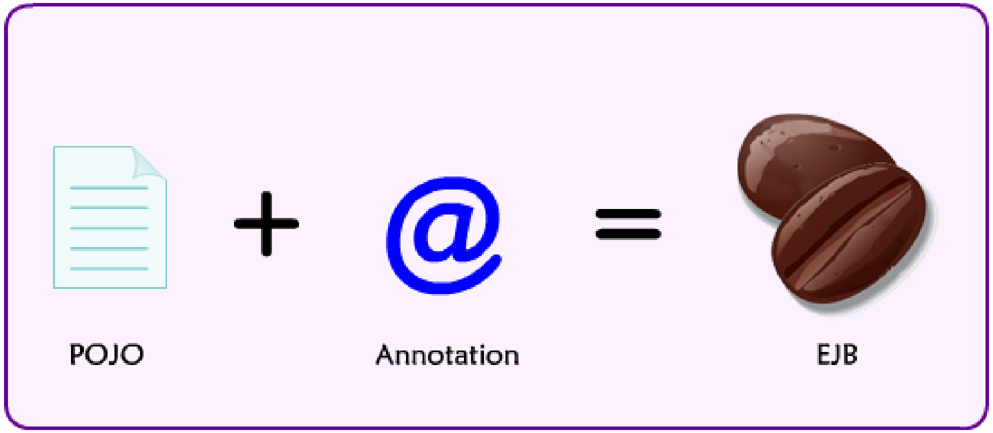
Expectation



AJDay9_7261-ejb
- Source Packages
  - sample.session
    - LoginSessionBean.java
    - LoginSessionBeanLocal.java
    - LoginSessionBeanRemote.java
- Libraries
  - JDK 1.7 (Default)
  - JBoss6.1.0Final
- Enterprise Beans
  - LoginSessionBean
    - Remote Methods
    - Local Methods
    - Bean Methods

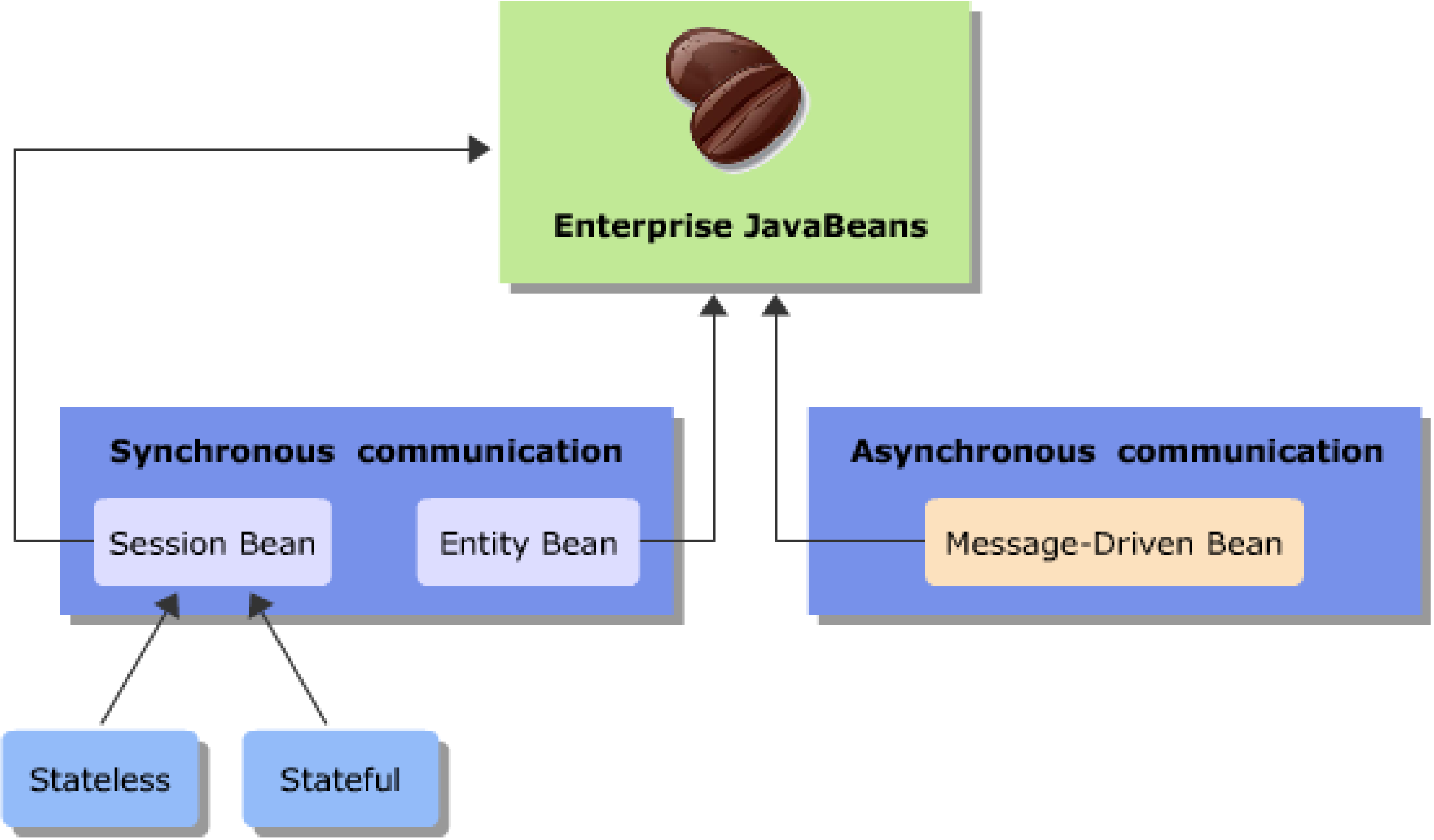
Introduction to EJB 3.0
Overview

- **EJB 3 uses Metadata annotations to specify** the **services** that the EJB components **will use when** it is **deployed in** the **containers**
  - **Annotations help** the **developers** to **provide** the **specification** and based on specification, the **system automatically adds code**
  - Annotations **help** the **developers** to **transform** a **simple POJO to** an **EJB**
  - Annotations can be only used to specify the required services but also **used** to **specify the component type**



Introduction to EJB 3.0

Types



Introduction to EJB 3.0

New Features

- **Callback Methods**
  - **Implementation** of call back methods **are optional** in EJB 3.0
  - **Container invokes** the method **when defined** by the developer
  - Any method can be designated as callback method to listen to life cycle events
  - The developer can **use** the **callback listener class instead** of **defining** the **callback methods in the bean class**
- **Elimination of Home Interface and Home Objects**
  - **Home interface** has been **replaced** by **Plain Old Java Interface (POJI)**
  - **Home object** has been **replaced by POJO**
  - **Business interface contains all the business methods**
    - Remote Business Interface
    - Local Business Interface

Introduction to EJB 3.0

New Features

- **Elimination of Component Interface**
  - Earlier, component interfaces were used as they provided a way for the container to notify the bean instance of the various life cycle events affecting it
  - Now, **bean** is **represented** as a **simple POJO** class **implementing** the **business interface** if it is a session bean
  - **Two ways** in which a bean class can get notification from the container are as follows:
    - Developer **writes a separate class containing** the **implementation of callback notification method**
    - Developer **implements** the **callback notification methods within** the **bean** class **and designate** each of these **methods to handle appropriate events**
  - Both these approaches **require the use of annotations**

Introduction to EJB 3.0

New Features

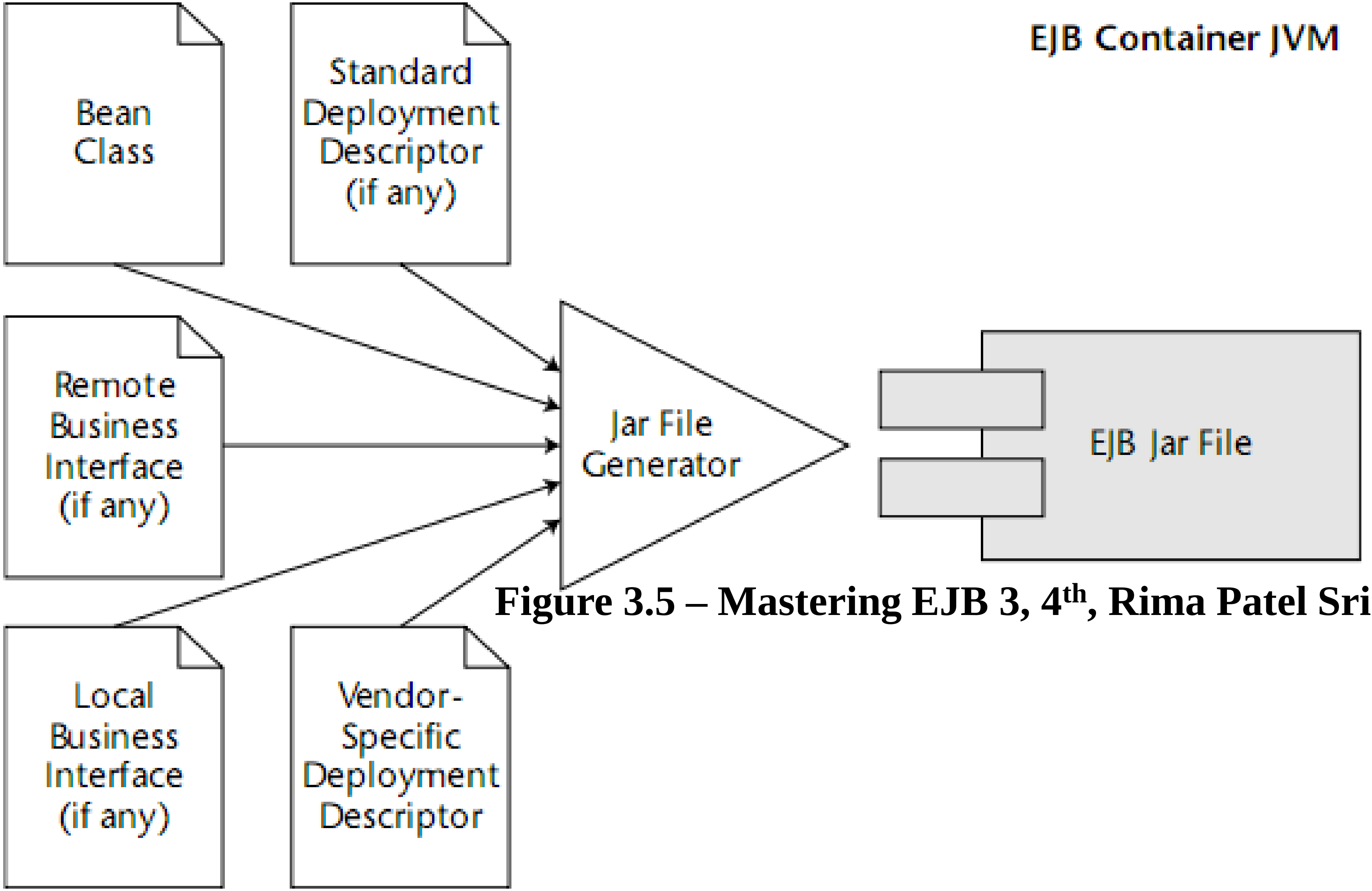
- **Dependency Injection/Simplified Access**
  - Earlier, JNDI APIs were used to gain access to environment entries
  - In EJB 3, **dependence injection and lookup**() method on **EJBContext interface have been added**
    - Dependency Injection is a means by which the **container makes available the requested environmental entry for the bean instance**
    - Environment variables are **injected into the bean's variables or methods**
    - These are made available to the bean instance before any business methods are invoked by the instance
  - Developer **uses deployment descriptor or annotations** to **specify these injections**
    - The bean methods **should follow the Java naming conventions** for properties (getter/ setter or accessor methods)
  - If the **dependency injection fails**, then the **container discards the bean instance and creates another bean instance**
  - **@Inject or @Resource annotations** can be **used for dependency injection**

Introduction to EJB 3.0
New Features

- **Interceptors**
  - Used to **intercept business method calls or life cycle callback method** calls
  - **Used** by **Stateless, Stateful, and MDB**
  - Help the developers to **enhance** the **business methods by adding additional functionality**
  - Can be **defined in a separate class**
- **Simple JNDI lookup for EJB**
  - The client can **easily invoke methods** on EJB than **creating an instance by invoking the ejbCreate method** as done previously
- **Java Persistence API**
  - EJB 3 **provides JPA** for simplifying the programming model for entity persistence
  - **Using** the **POJO model**, **instead** of the **abstract persistence schema model**

Introduction to EJBs

Packaging and Deploying



EJB Container JVM
Bean Class
Standard Deployment Descriptor (if any)
Remote Business Interface (if any)
Local Business Interface (if any)
Vendor-Specific Deployment Descriptor
Jar File Generator
EJB Jar File

Figure 3.5 – Mastering EJB 3, 4th, Rima Patel Sriganesh

Introduction to EJBs

Accessing

-

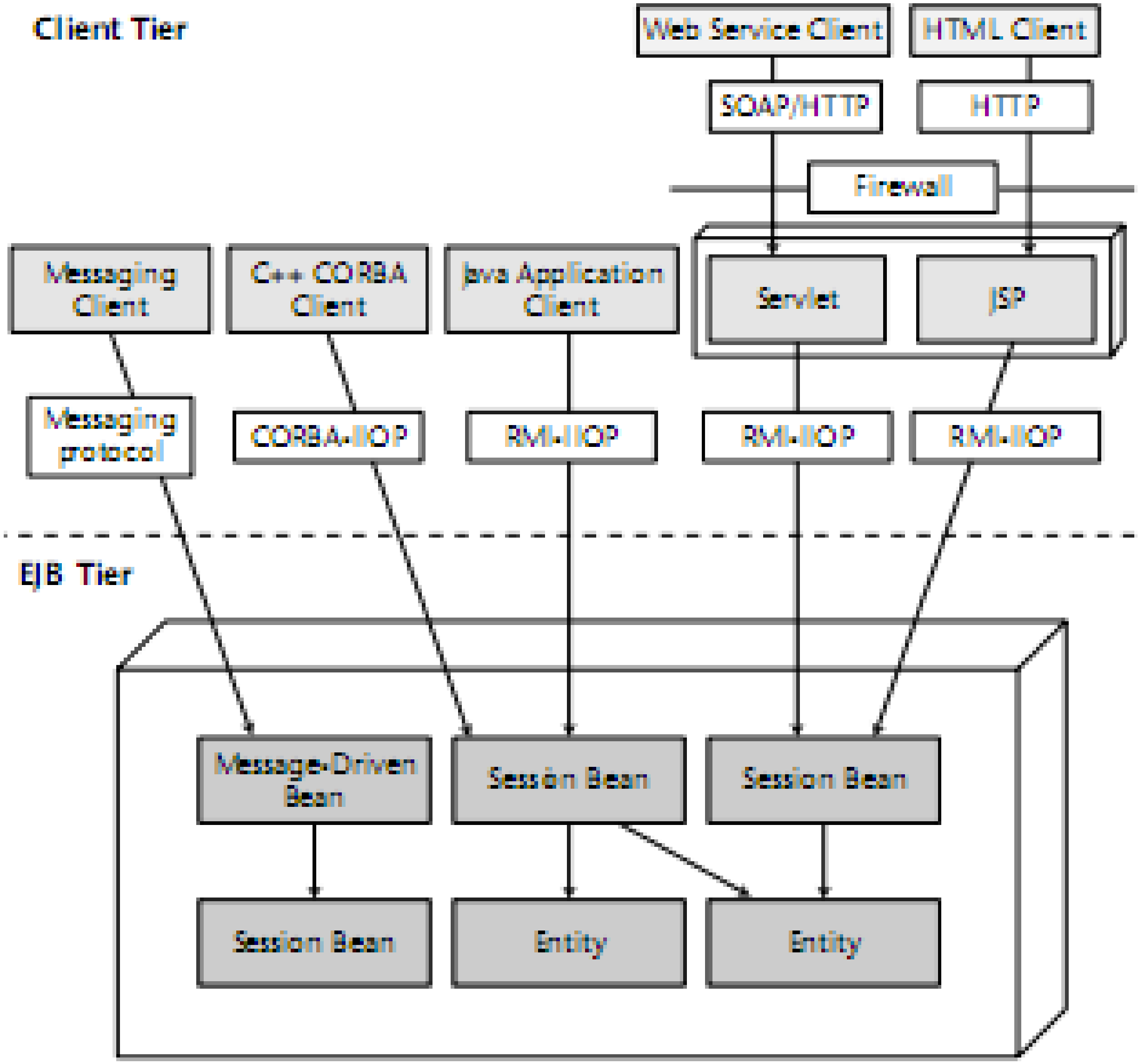


Figure 3.2 – Mastering EJB 3, 4th, Rima Patel Sriganesh

EJB Implementation

EJB Development Process

- **Requirement: JBoss 6.1.0 Final Application Server & Netbeans 7.4**
- **Step 1:** Creating a new EJB Module project/ Enterprise Application Project
- **Step 2:** Creating the new corresponding bean depending on your purpose.
- **Step 3:** Building the business on Beans
- **Step 4:** Mapping the JNDI to beans
- **Step 5:** Creating the web/client application to consume
- **Step 6:** Building the project to jar/ear file
- **Step 7:** Deploying the project on Application server
- **Step 8:** Running the client to test the EJB

Summary

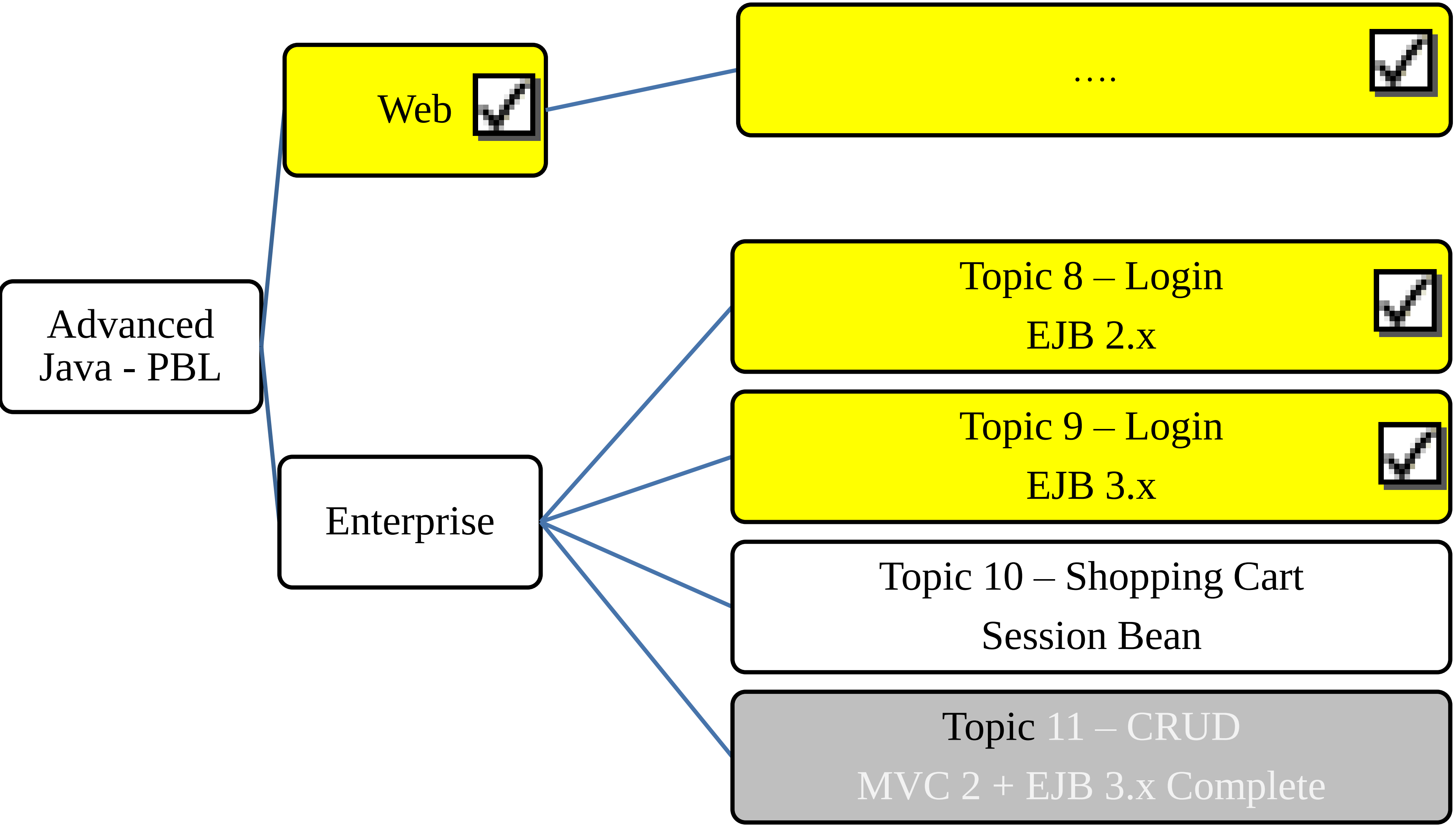
- **How to build the application using EJB 3**
  - Need of EJB 3
  - New Features

Q&A

Next Lecture

- **How to build enterprise application using Session Beans?**
  - Stateless
  - Stateful
  - Definition, Implementation, Life cycles

Next Lecture



Advanced Java - PBL

Web ☑

…. ☑

Enterprise

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

Appendix – Introduction to EJB 3.0
Overview

- EJB **implements** the **business logic and the persistence layer**
  - **Session and Message-driven** bean reside in and **use the services of business layer**
  - **Entities** reside in and **use** the **services of persistence layer**
    - Entities can be used to **model domain objects** including modeling state and behavior
- **Domain-Driven Design**
  - **Domain objects** should **contain business logic**
  - Represents **domain objects as entities** in EJB 3.0
  - Add business logic in domain object
  - Implements **business logic in the application layer** also known as **service layer**

Appendix – Introduction to EJB 3.0

Need of EJB 3.0

- **Benefits** are as follows
  - **Simple**
    - The developer can **focus** on developing the **business logic** instead of concentrating on other services such as transaction, security, resource pooling, …
    - EJB 3 **provides** a **practical outlook** and does **not demand much understanding** of **theoretical intricacies**
  - **Reusable**
    - EJB 3 is a **reusable component** because it can be **used** by **multiple application**s that can make **calls** to the **deployed enterprise bean**
  - **Scalable**
    - EJB is used when application needs to scale beyond **initial low level usage level and support multiple concurrent users**
  - **Transactional**
    - Transaction is **support** by EJB **container** that **helps** to **maintain** the **consistent state of the DB** when an error takes place
- EJB **cannot be used** where there is
  - **No need for** the application to have **scalability**, **transaction management**, or **security features**
  - **No need for** the application to be **platform independent**

Appendix – Introduction to EJB 3.0

Types

- **Session Beans & Message-driven Beans**
  - **Are same as EJB 2.x specification in concepts**

Appendix – Introduction to EJB 3.0

Types

- **Entity Beans/ Entity Classes**
  - **Represents business** data
  - **Are Java objects** that **store database information**
  - Are POJO classes that **use JPA** for **persisting data** into relational database using Object-Relational Mapping (ORM)
  - In EJB3, persistence activity is performed by JPA using ORM techniques.
  - The **standards defined by JPA** are as follows
    - ORM configuration metadata is created for **mapping of entities to relational table**
    - **EntityManager API** is used for **performing persistence operations for entities**
    - Java Persistence Query Language (**JPQL**) is used for **searching and retrieving data persistence in DB**

Appendix – Introduction to EJB 3.0

New Features

- **Eases development** of enterprise applications as it **removes** the **need** for **interfaces** and **deployment descriptors**
- **Uses** the **metadata annotations** to **generate** the **interfaces** and the deployment descriptors
- **Annotations**
  - Is a **metadata information** that is **attached** to an **element within** the **code** to characterize it
  - Are processed when the code containing it are compiled or interpreted by compilers, deployment tools, and so on
  - Can result in the **generation of code documents, code artifacts, and so on**
  - EJB 3 has defined **many built-in annotations**
    - This resulted in **changes in EJB programming** as it contains a mix of metadata tags and code constructs
    - This made the **configuration task easier for the developer** (The developer can use the defaults wherever possible)
    - It is the **responsibility of the compilers, code generators, deployment tools to generate the appropriate semantics**

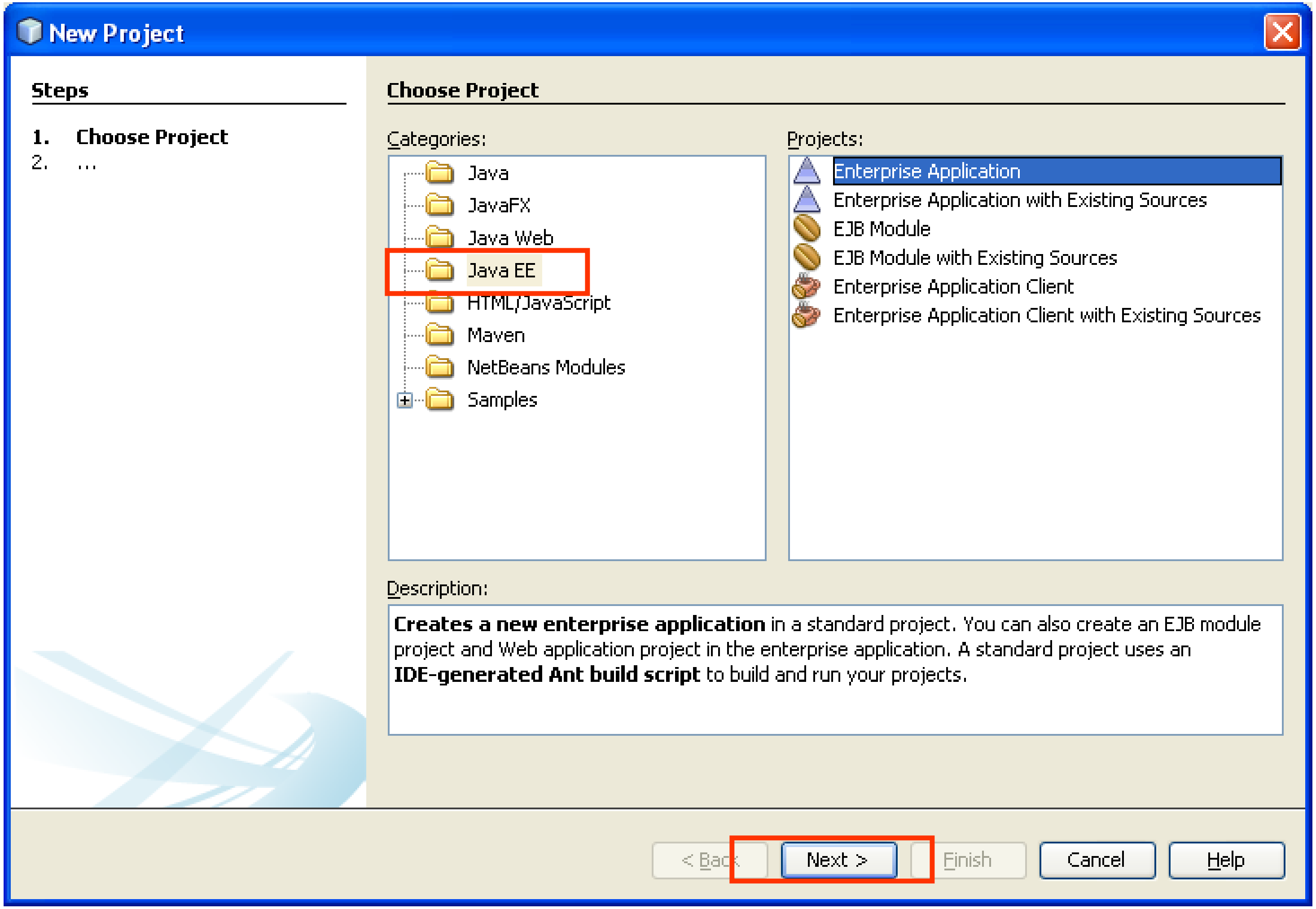
Appendix – Introduction to EJB 3.0

New Features

- **The advantages of using annotations**
  - **Ease** of use
    - Annotations are checked and compiled by the Java language compiler and are simple to use
  - **Portability**
  - **Type Checking**
    - Annotations are instances of annotation types and are compiled in their own class files
  - **Runtime Reflection**
    - Annotations are stored **in the class files** and **accessed for runtime access**
- **The disadvantages of using annotations**
  - It is **invisible** when the bean **developer** and **deployer** are two separate entities
  - **Before** the **deployer generates** the bean **deployment descriptor**, he/she **needs** to **read the code** so that the deployment descriptor **does not override** the **bean provider's deployment metadata-specified configuration**
  - Another important issue with metadata is that **each time a change** is **made to the bean code, the bean needs to be recompiled and repackaged**

Appendix – EJB Implementation

Step 1: Creating a new EJB project



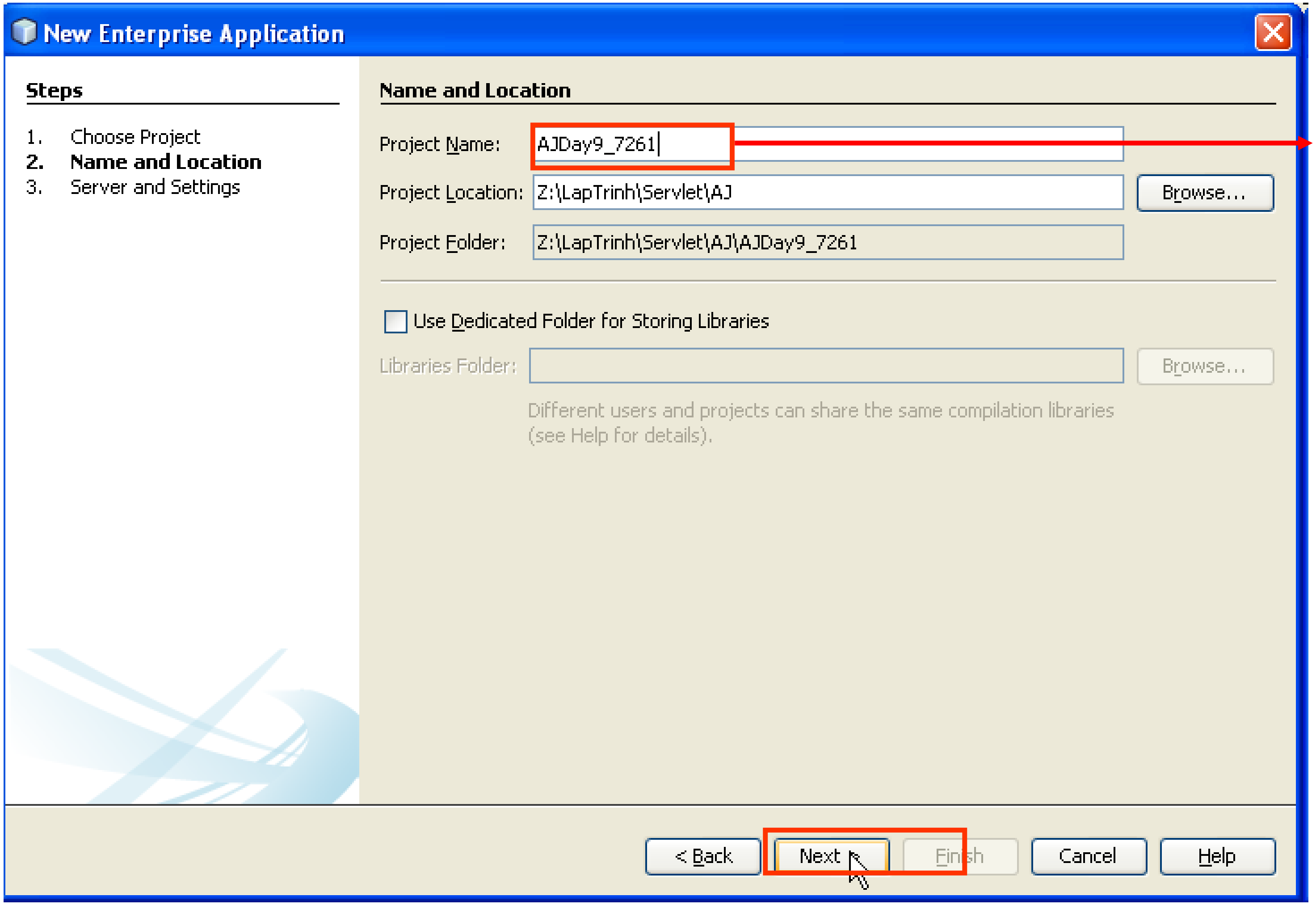
Description:

Creates a new enterprise application in a standard project. You can also create an EJB module project and Web application project in the enterprise application. A standard project uses an **IDE-generated Ant build script** to build and run your projects.

- Choose “**Enterprise Beans**” on “**Categories**”
- Then, choose “**Enterprise Application**” on “**Projects**”. Click **Next** button

EJB Implementation

Step 1: Creating a new EJB project



New Enterprise Application

Steps

1. Choose Project
2. **Name and Location**
3. Server and Settings

Name and Location

Project Name: AJDay9_7261

Project Location: Z:\LapTrinh\Servlet\AJ

Project Folder: Z:\LapTrinh\Servlet\AJ\AJDay9_7261

□ Use Dedicated Folder for Storing Libraries

Libraries Folder:

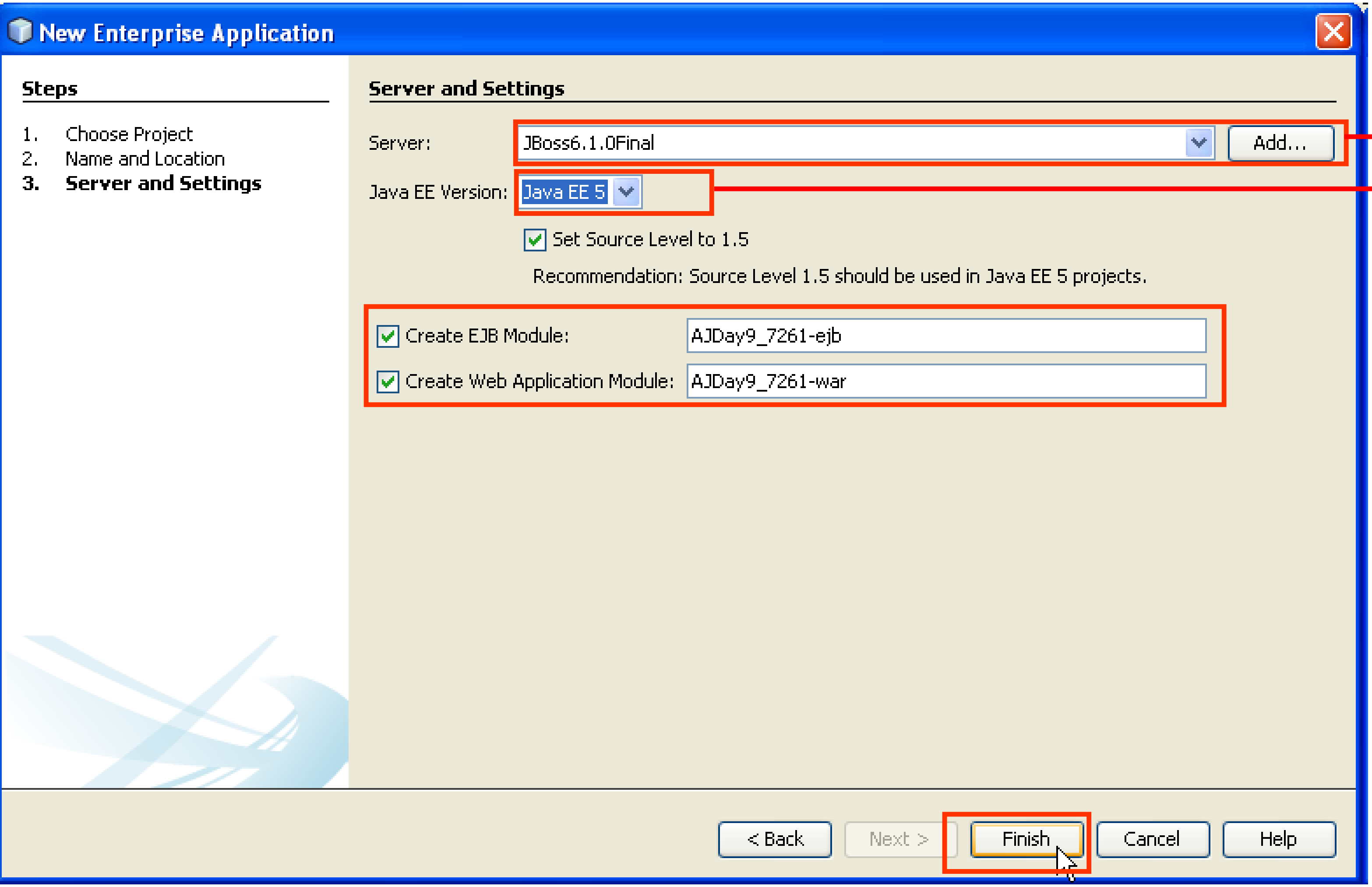
Different users and projects can share the same compilation libraries (see Help for details).

Fill your project name

- Click **Next** button

EJB Implementation

Step 1: Creating a new EJB project



New Enterprise Application

Steps

1. Choose Project
2. Name and Location
3. **Server and Settings**

Server and Settings

Server: JBoss6.1.0Final

Java EE Version: Java EE 5

☑ Set Source Level to 1.5

Recommendation: Source Level 1.5 should be used in Java EE 5 projects.

☑ Create EJB Module: AJDay9_7261-ejb

☑ Create Web Application Module: AJDay9_7261-war

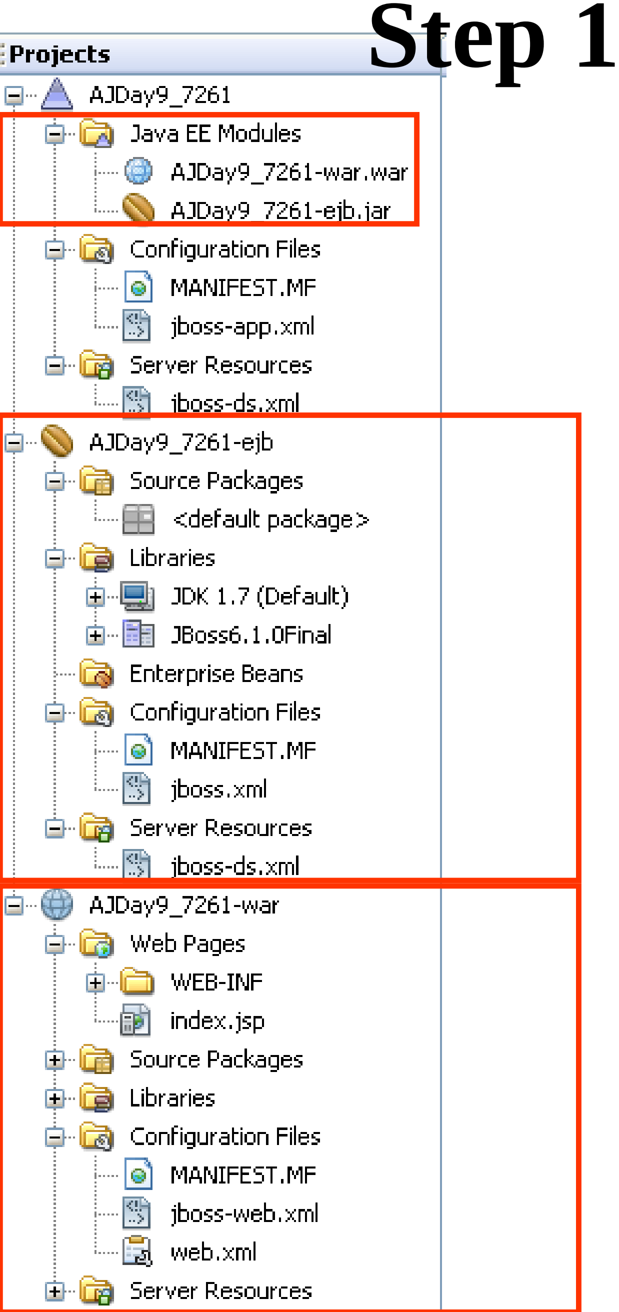
Choose the Jboss Server that is added

Choose the JavaEE5

- Click **Finish** button

EJB Implementation

Step 1: Creating a new EJB project

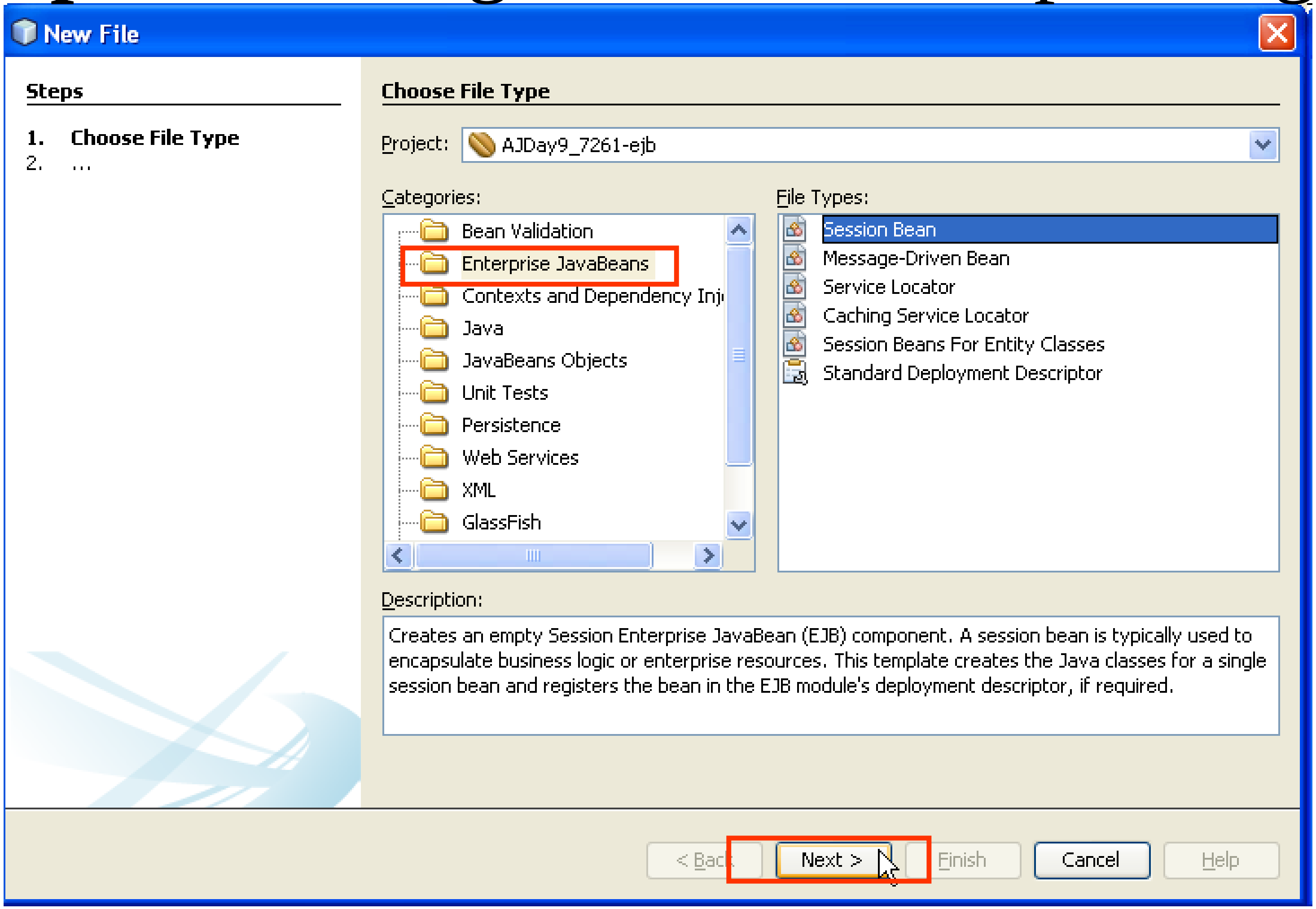


Projects

- AJDay9_7261
  - Java EE Modules
    - AJDay9_7261-war.war
    - AJDay9_7261-ejb.jar
  - Configuration Files
    - MANIFEST.MF
    - jboss-app.xml
  - Server Resources
    - jboss-ds.xml
- AJDay9_7261-ejb
  - Source Packages
    - <default package>
  - Libraries
    - JDK 1.7 (Default)
    - JBoss6.1.0Final
  - Enterprise Beans
  - Configuration Files
    - MANIFEST.MF
    - jboss.xml
  - Server Resources
    - jboss-ds.xml
- AJDay9_7261-war
  - Web Pages
    - WEB-INF
    - index.jsp
  - Source Packages
  - Libraries
  - Configuration Files
    - MANIFEST.MF
    - jboss-web.xml
    - web.xml
  - Server Resources

EJB Implementation

Step 2: Creating the new corresponding bean



New File

Steps

1. Choose File Type
2. ...

Choose File Type

Project: AJDay9_7261-ejb

Categories:

- Bean Validation
- Enterprise JavaBeans
- Contexts and Dependency Inj
- Java
- JavaBeans Objects
- Unit Tests
- Persistence
- Web Services
- XML
- GlassFish

File Types:

- Session Bean
- Message-Driven Bean
- Service Locator
- Caching Service Locator
- Session Beans For Entity Classes
- Standard Deployment Descriptor

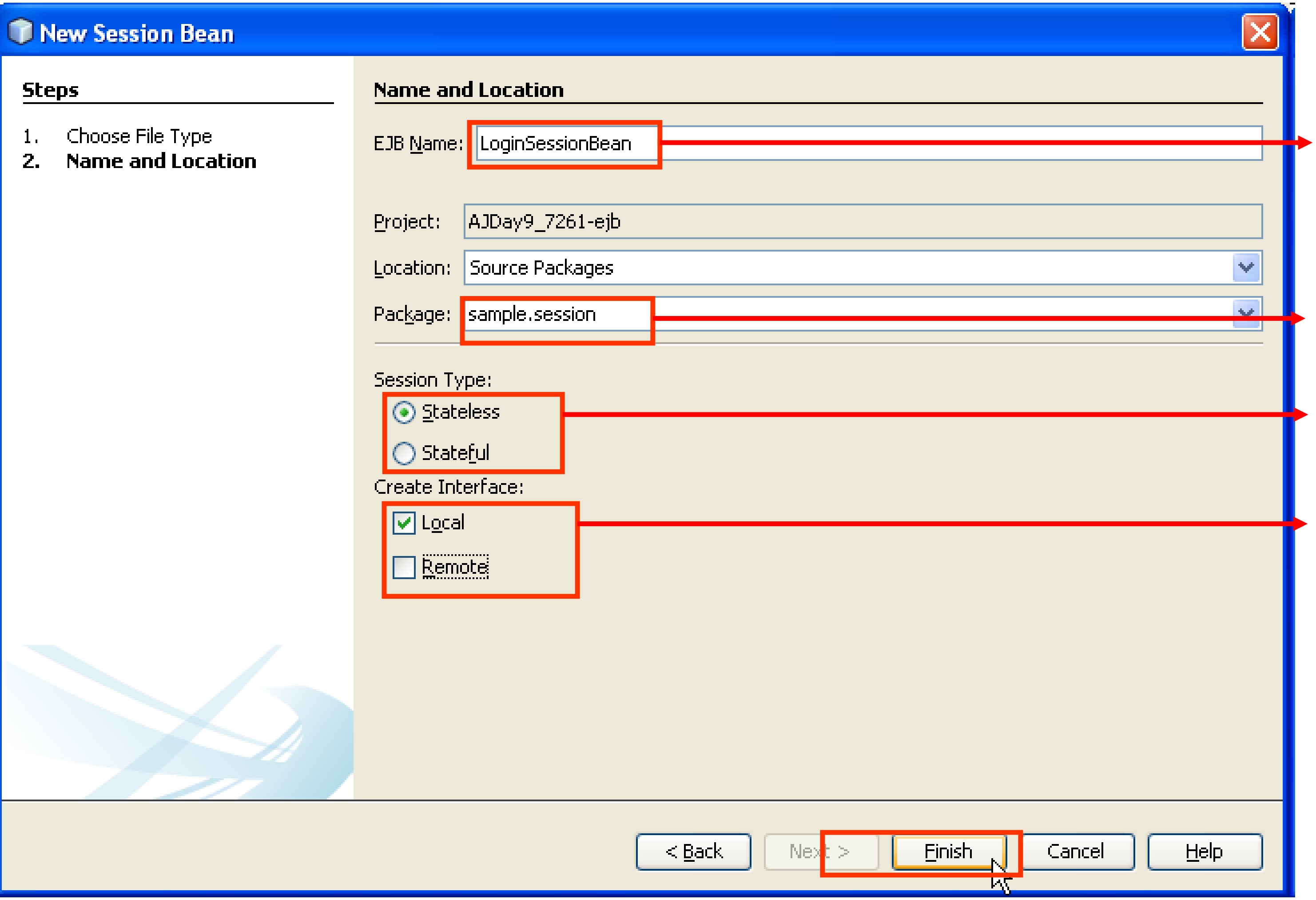
Description:

Creates an empty Session Enterprise JavaBean (EJB) component. A session bean is typically used to encapsulate business logic or enterprise resources. This template creates the Java classes for a single session bean and registers the bean in the EJB module's deployment descriptor, if required.

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

EJB Implementation

Step 2: Creating the new corresponding bean



New Session Bean

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

EJB Name: LoginSessionBean

Project: AJDay9_7261-ejb

Location: Source Packages

Package: sample.session

Session Type:

● Stateless

○ Stateful

Create Interface:

☑ Local

□ Remote

Fill your bean name

Fill or choose the package name

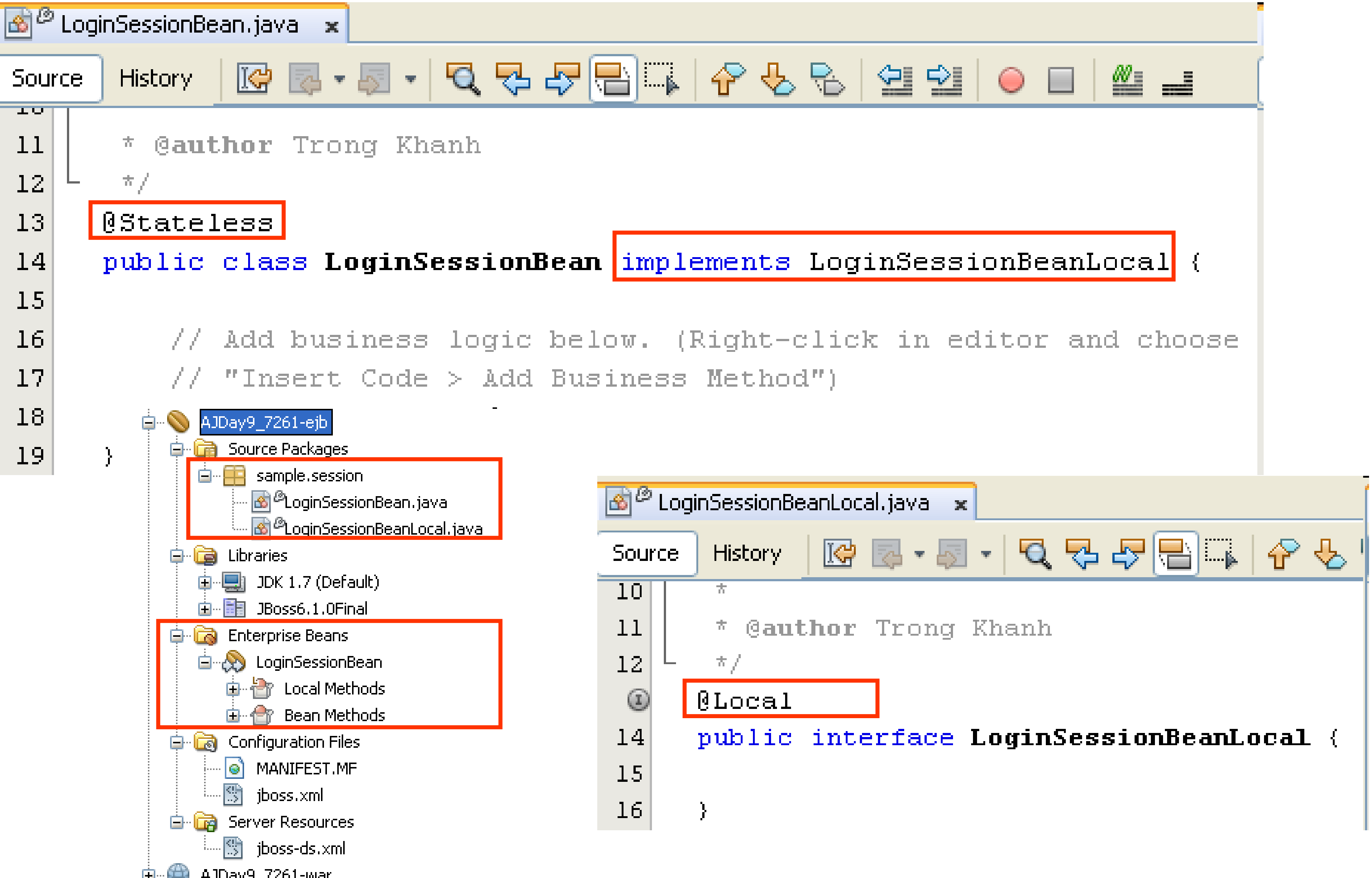
Choose stateless or stateful

Choose local

- Click **Finish** button

EJB Implementation

Step 2: Creating the new corresponding bean



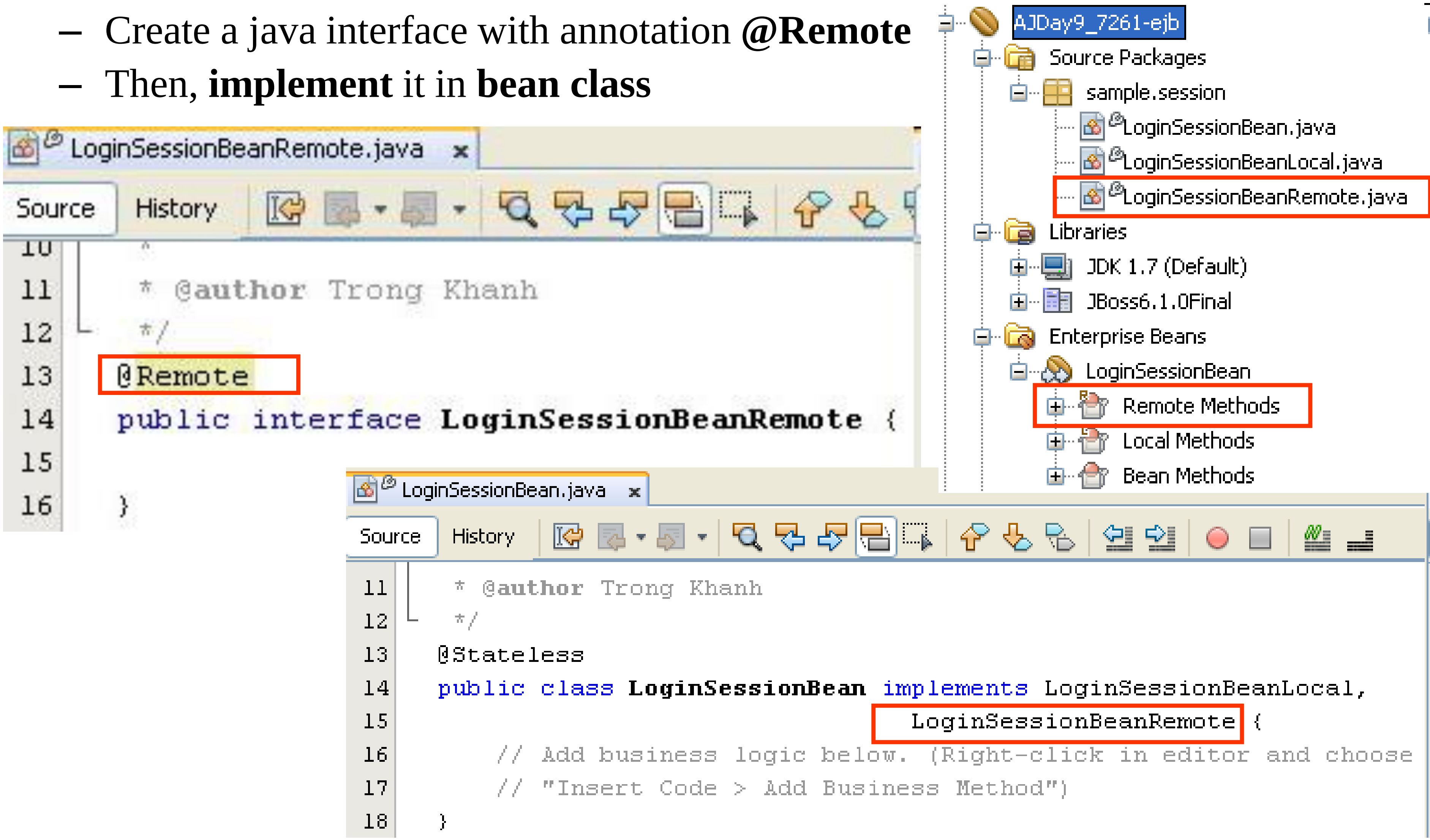
```java
 * @author Trong Khanh
 */
@Stateless
public class LoginSessionBean implements LoginSessionBeanLocal {

    // Add business logic below. (Right-click in editor and choose
    // "Insert Code > Add Business Method")

}
```

```java
 *
 * @author Trong Khanh
 */
@Local
public interface LoginSessionBeanLocal {

}
```

EJB Implementation

Step 2: Creating the new corresponding bean

- **Create** the **remote business interface**
  - Create a java interface with annotation **@Remote**
  - Then, **implement** it in **bean class**



LoginSessionBeanRemote.java

```
 *
 * @author Trong Khanh
 */
@Remote
public interface LoginSessionBeanRemote {

}
```

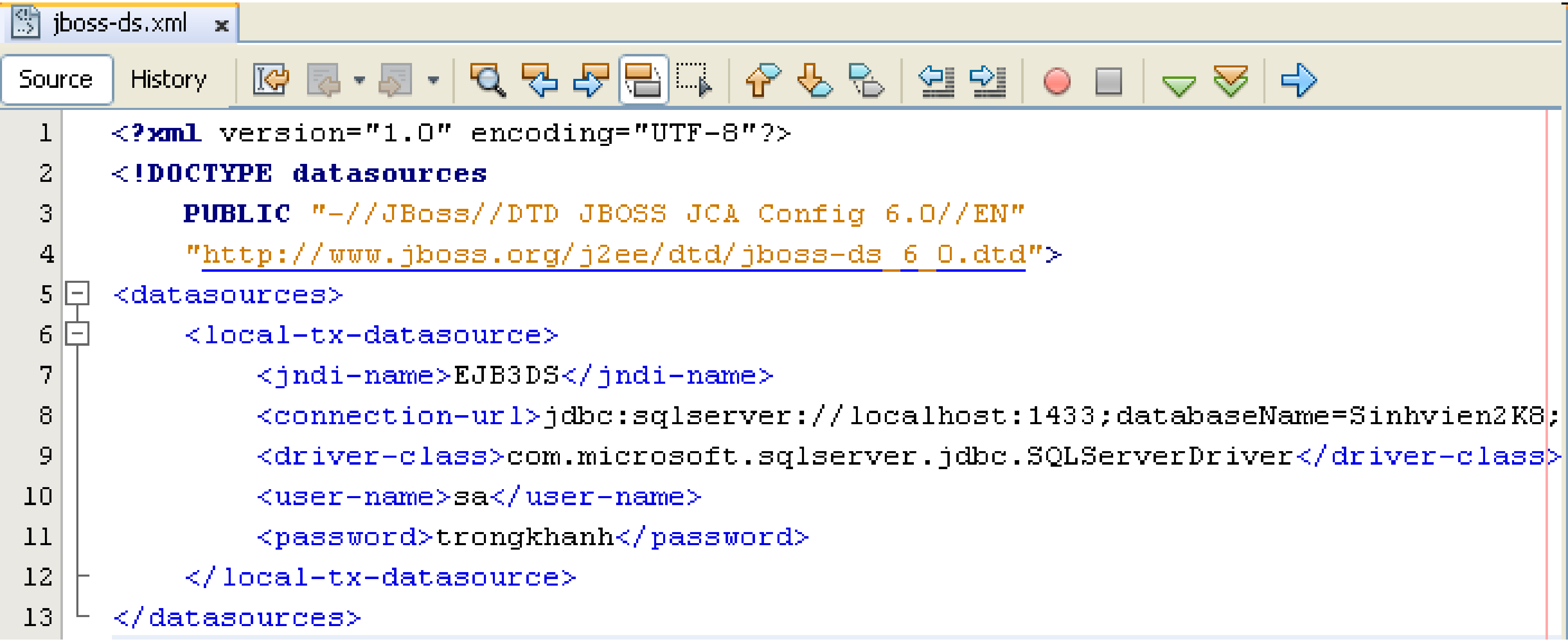
LoginSessionBean.java

```
 * @author Trong Khanh
 */
@Stateless
public class LoginSessionBean implements LoginSessionBeanLocal,
                                         LoginSessionBeanRemote {
    // Add business logic below. (Right-click in editor and choose
    // "Insert Code > Add Business Method")
}
```

EJB Implementation

Addition – Create the DS to connect DB

- Create the **jboss-ds. xml** in Server Resources in EJB module
  - **Modify** the **connection** to create the **data source** that is used to connect to DB

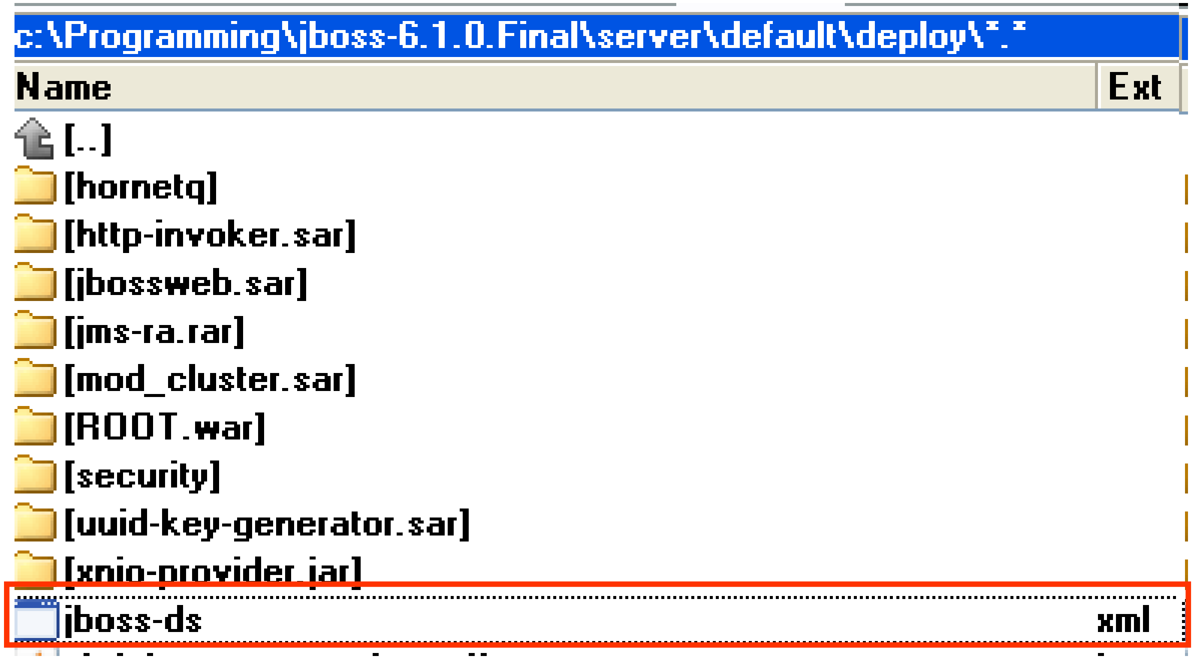


```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE datasources
    PUBLIC "-//JBoss//DTD JBOSS JCA Config 6.0//EN"
    "http://www.jboss.org/j2ee/dtd/jboss-ds_6_0.dtd">
<datasources>
    <local-tx-datasource>
        <jndi-name>EJB3DS</jndi-name>
        <connection-url>jdbc:sqlserver://localhost:1433;databaseName=Sinhvien2K8;
        <driver-class>com.microsoft.sqlserver.jdbc.SQLServerDriver</driver-class>
        <user-name>sa</user-name>
        <password>trongkhanh</password>
    </local-tx-datasource>
</datasources>
```

EJB Implementation

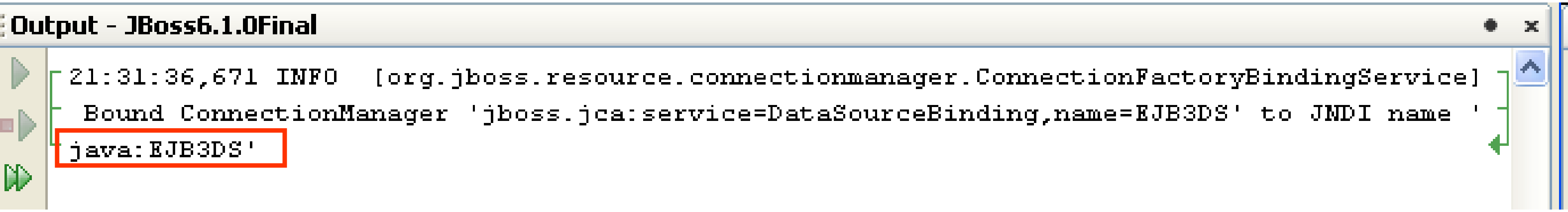
Addition – Create the DS to connect DB

- **Deploy** data source file: jboss-ds.xml to **server**



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

Name	Ext
[..]	
[hornetq]	
[http-invoker.sar]	
[jbossweb.sar]	
[jms-ra.rar]	
[mod_cluster.sar]	
[ROOT.war]	
[security]	
[uuid-key-generator.sar]	
[xnio-provider.jar]	
jboss-ds	xml



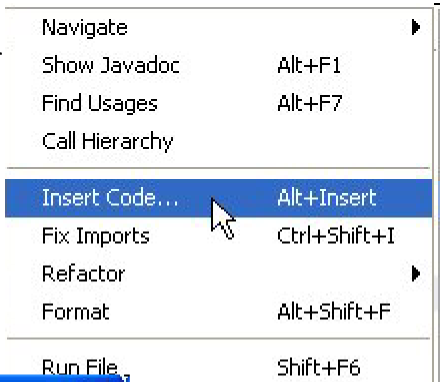
Output - JBoss6.1.0Final

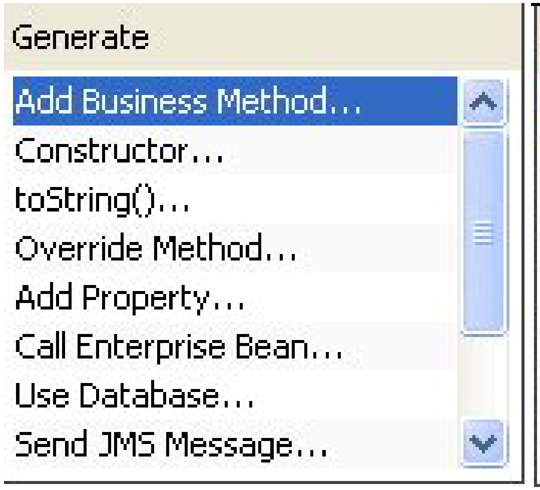
```
21:31:36,671 INFO  [org.jboss.resource.connectionmanager.ConnectionFactoryBindingService]
 Bound ConnectionManager 'jboss.jca:service=DataSourceBinding,name=EJB3DS' to JNDI name '
java:EJB3DS'
```

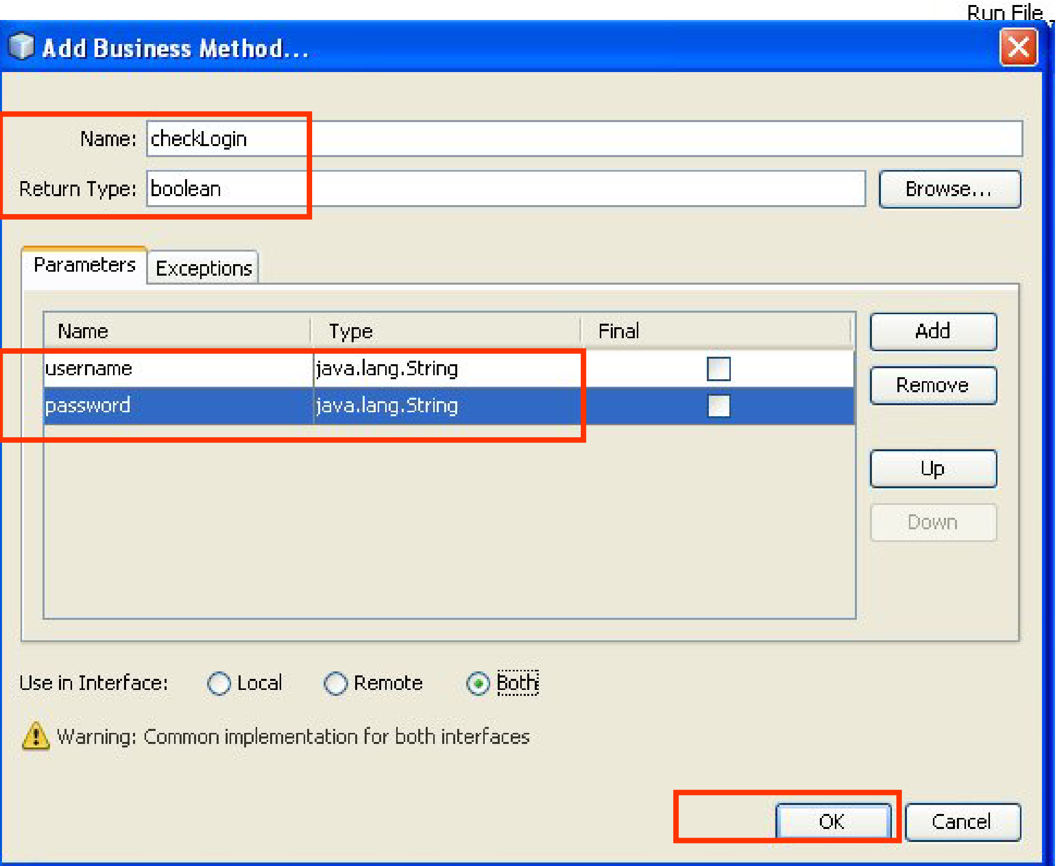
EJB Implementation

Step 3: Building the business methods

- **Adding** a new **business method**
  - Right click on source code of the Bean file
  - Then, choose Insert Code, click Add Business Method…







Name: checkLogin

Return Type: boolean

Name	Type	Final
username	java.lang.String	□
password	java.lang.String	□

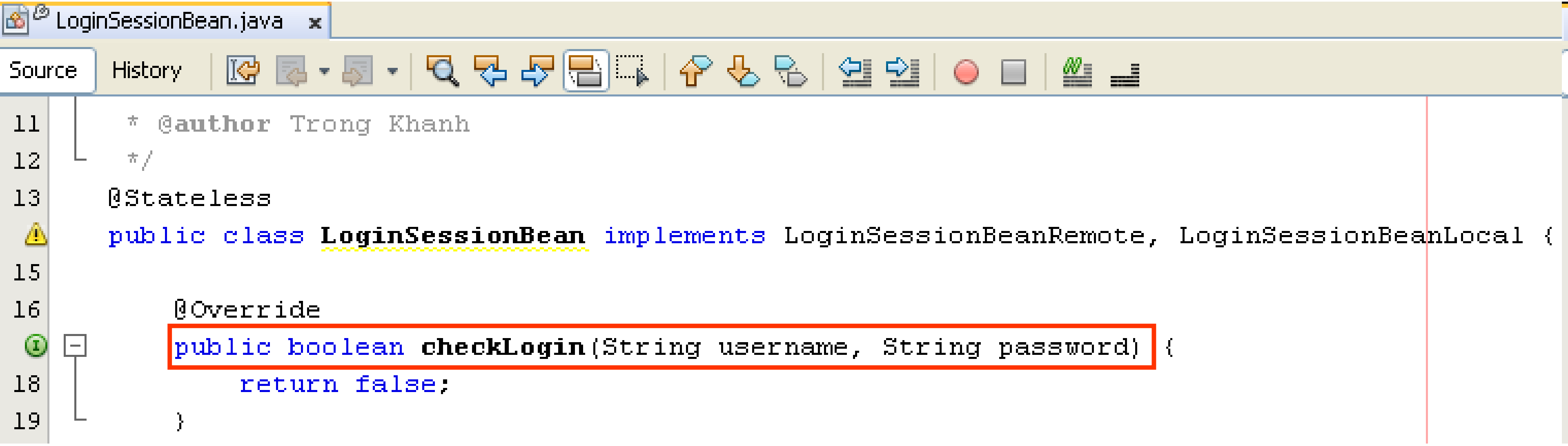
Use in Interface: ○ Local ○ Remote ● Both

Warning: Common implementation for both interfaces

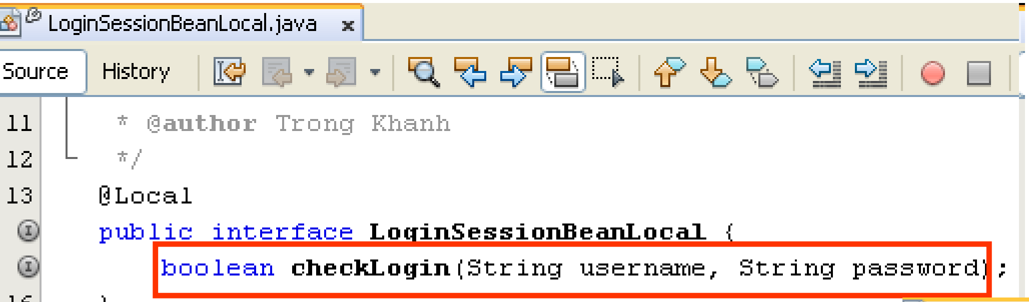
- Fill or type the method name with return type and add all parameters
- Then, click OK Button

EJB Implementation

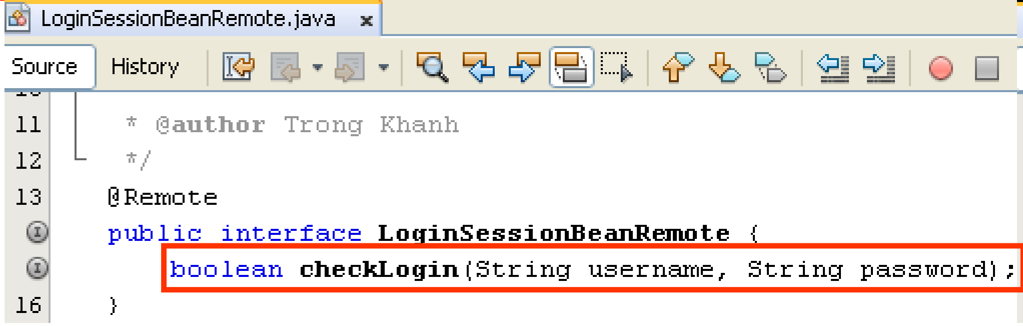
Step 3: Building the business methods



```
 * @author Trong Khanh
 */
@Stateless
public class LoginSessionBean implements LoginSessionBeanRemote, LoginSessionBeanLocal {

    @Override
    public boolean checkLogin(String username, String password) {
        return false;
    }
```



```
 * @author Trong Khanh
 */
@Local
public interface LoginSessionBeanLocal {
    boolean checkLogin(String username, String password);
}
```



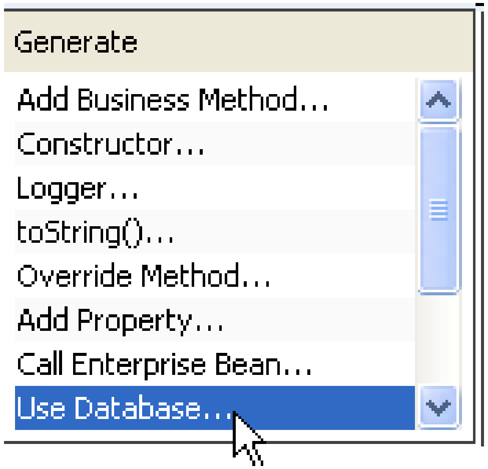
```
 * @author Trong Khanh
 */
@Remote
public interface LoginSessionBeanRemote {
    boolean checkLogin(String username, String password);
}
```

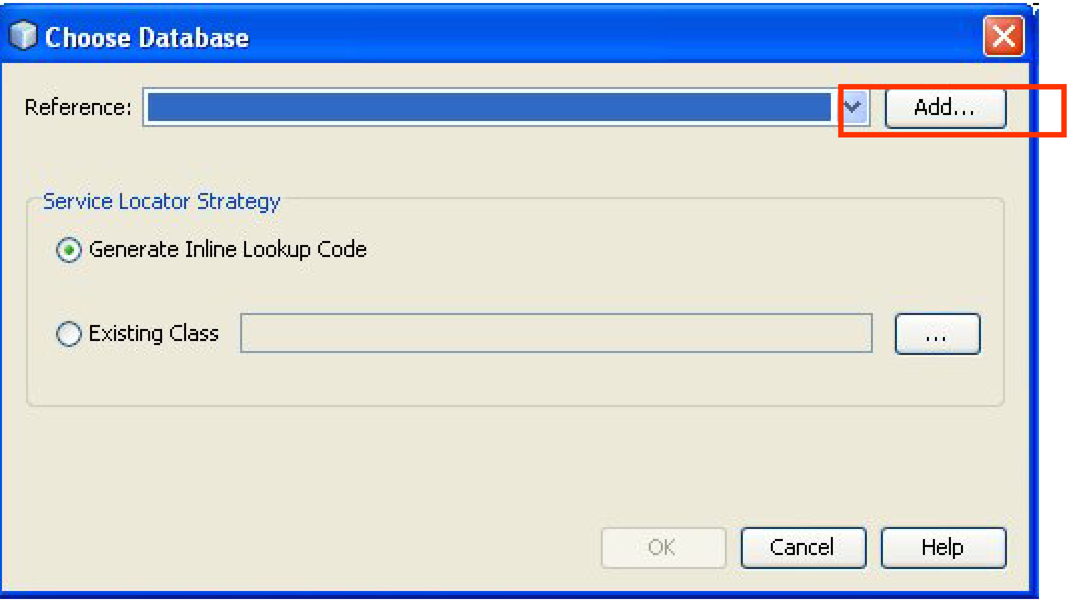
- Implement the body of method corresponding with your purpose

EJB Implementation

Addition – Using Resource in Bean

- **Adding** a **resource into bean**
  - Right click on source code of the Bean file
  - Then, choose Insert Code, click Use Database …

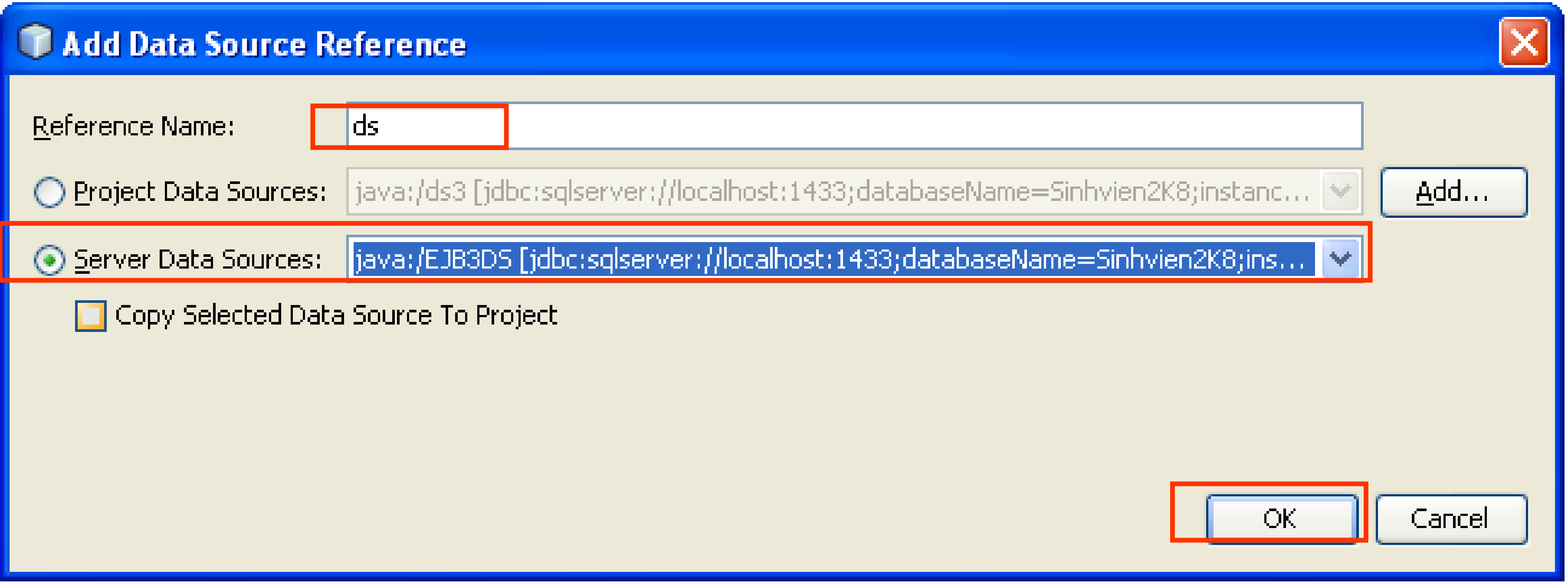




- Click Add button
- The Add Datasource Reference is shown

EJB Implementation

Addition – Using Resource in Bean



Add Data Source Reference

Reference Name: ds

○ Project Data Sources: java:/ds3 [jdbc:sqlserver://localhost:1433;databaseName=Sinhvien2K8;instanc...

● Server Data Sources: java:/EJB3DS [jdbc:sqlserver://localhost:1433;databaseName=Sinhvien2K8;ins...

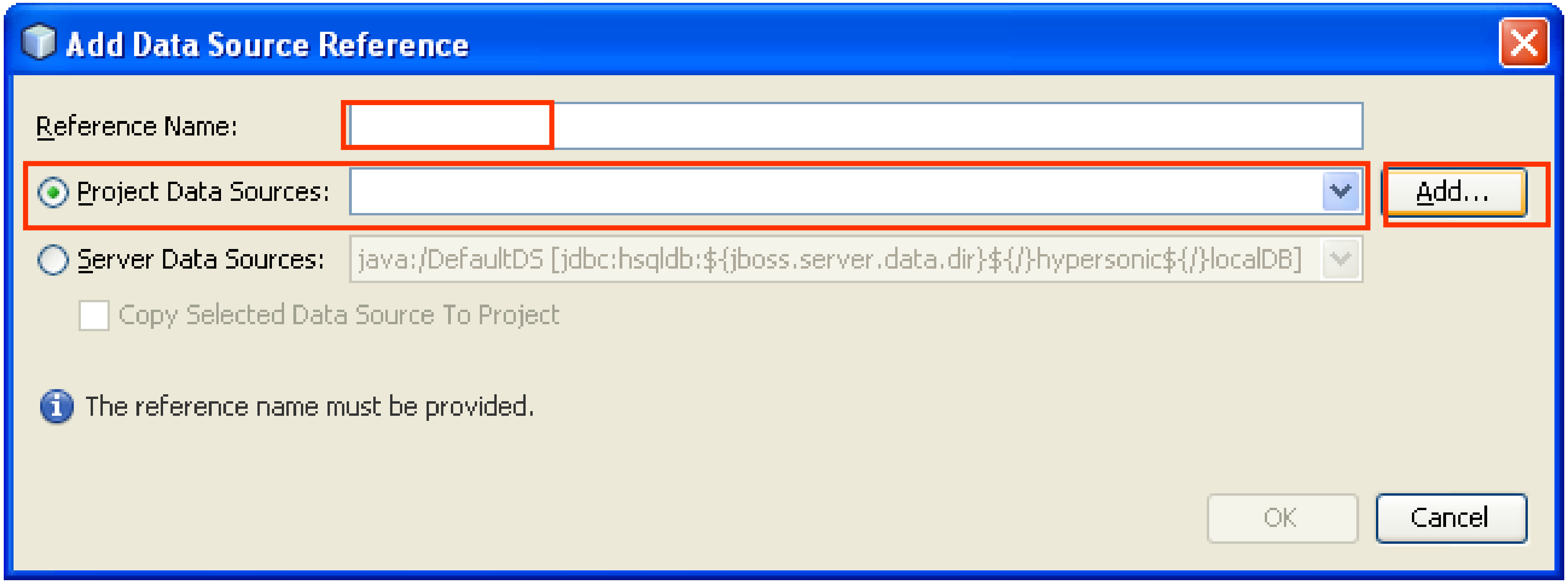
□ Copy Selected Data Source To Project

- Choose the Server Data Source then choosing the datasource that is deployed in server
- Type the reference Name
- If click the copy selected DataSource to Project, the reference code in jboss is not added
- Click OK button to return the choose Database dialog

EJB Implementation

Addition – Using Resource in Bean – 2nd Way

- **Do not create** the datasource file jboss-ds.xml
- **Use create** the data link in Database of Services tab
- **Use the Use Database** in Insert Code context menu
- **Choose** the Project Data Source then click Add



Add Data Source Reference

Reference Name:

● Project Data Sources:

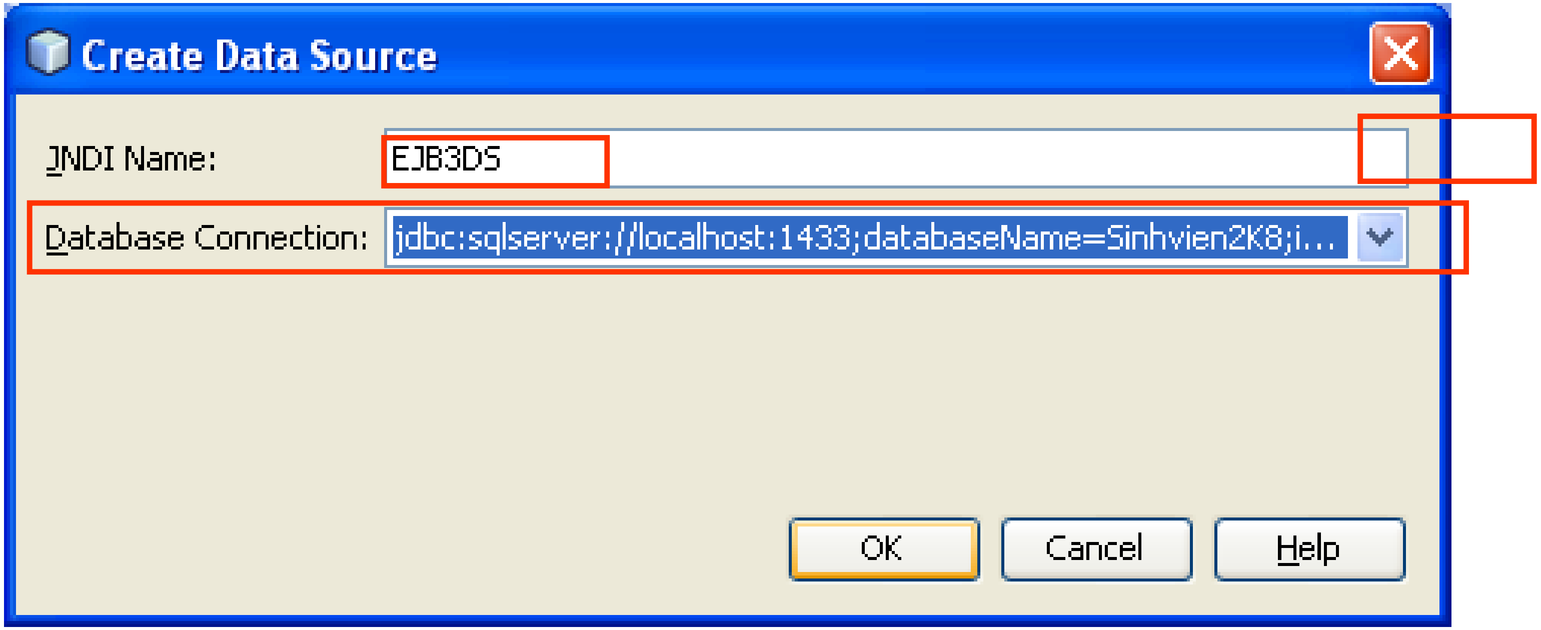
○ Server Data Sources: java:/DefaultDS [jdbc:hsqldb:${jboss.server.data.dir}${/}hypersonic${/}localDB]

□ Copy Selected Data Source To Project

The reference name must be provided.

EJB Implementation

Addition – Using Resource in Bean – 2nd Way



Create Data Source

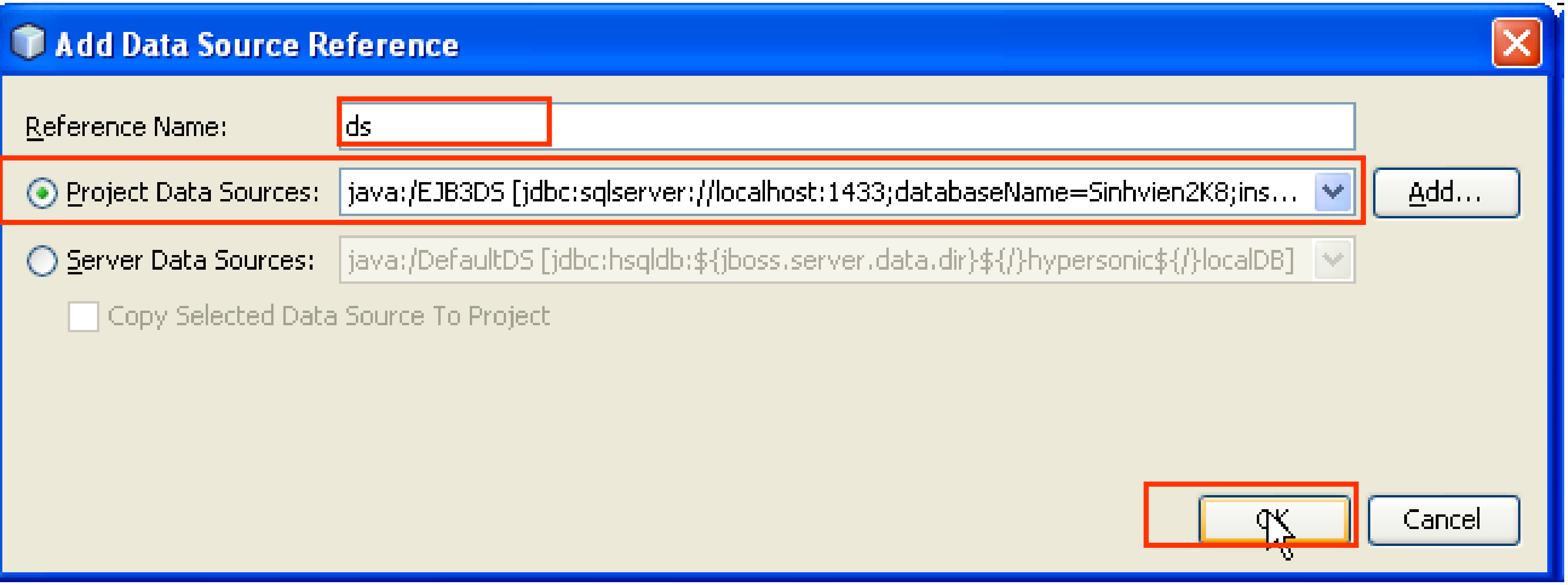
JNDI Name: EJB3DS

Database Connection: jdbc:sqlserver://localhost:1433;databaseName=Sinhvien2K8;i...

- Fill the JNDI name
- Choose the DB connection
- Click Ok button (The datasource file can be created in automatically in Server Resource folder)

EJB Implementation

Addition – Using Resource in Bean – 2nd way



Add Data Source Reference

Reference Name: ds

● Project Data Sources: java:/EJB3DS [jdbc:sqlserver://localhost:1433;databaseName=Sinhvien2K8;ins...

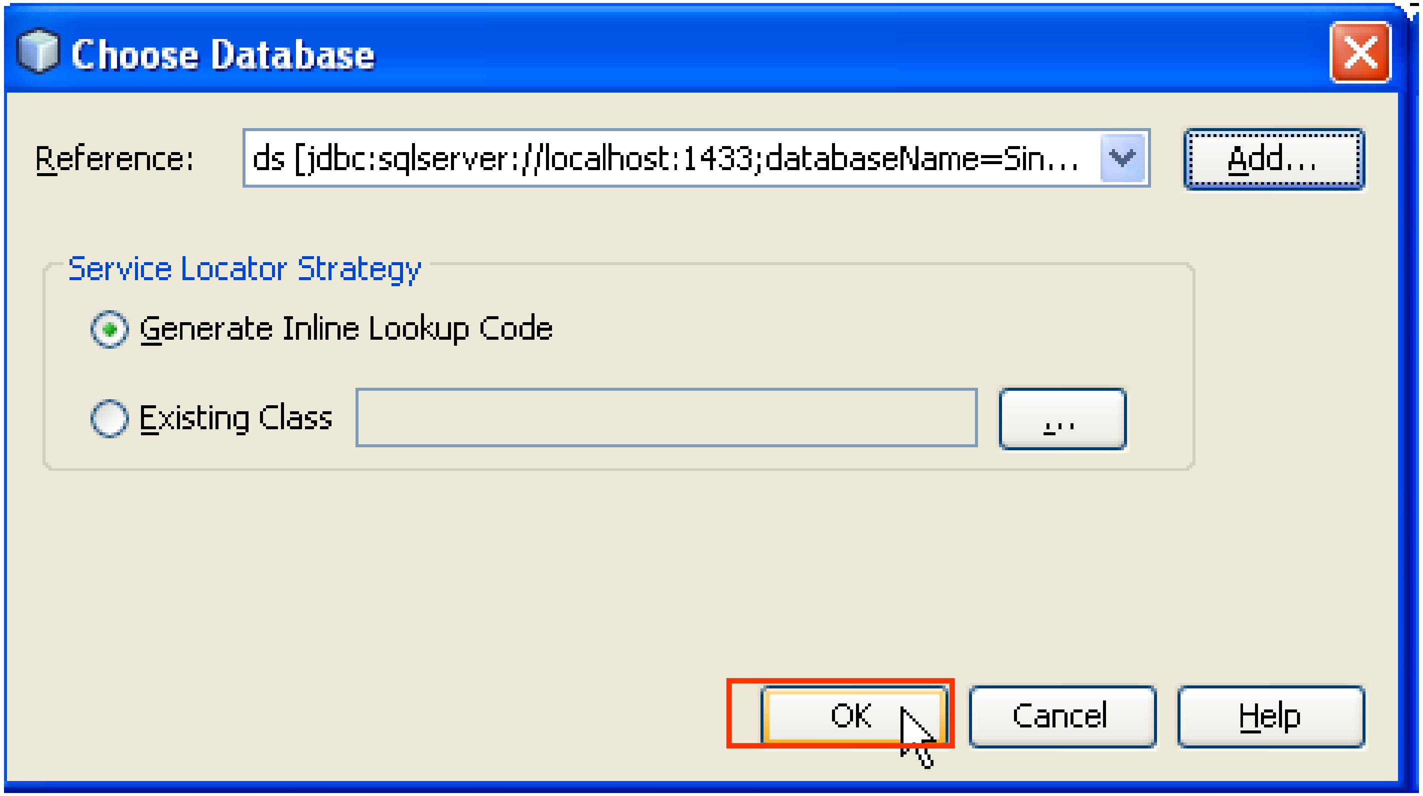
○ Server Data Sources: java:/DefaultDS [jdbc:hsqldb:${jboss.server.data.dir}${/}hypersonic${/}localDB]

□ Copy Selected Data Source To Project

- Type the reference Name
- Click OK button to return the choose Database dialog

EJB Implementation

Addition – Using Resource in Bean



Choose Database

Reference: ds [jdbc:sqlserver://localhost:1433;databaseName=Sin...

Service Locator Strategy

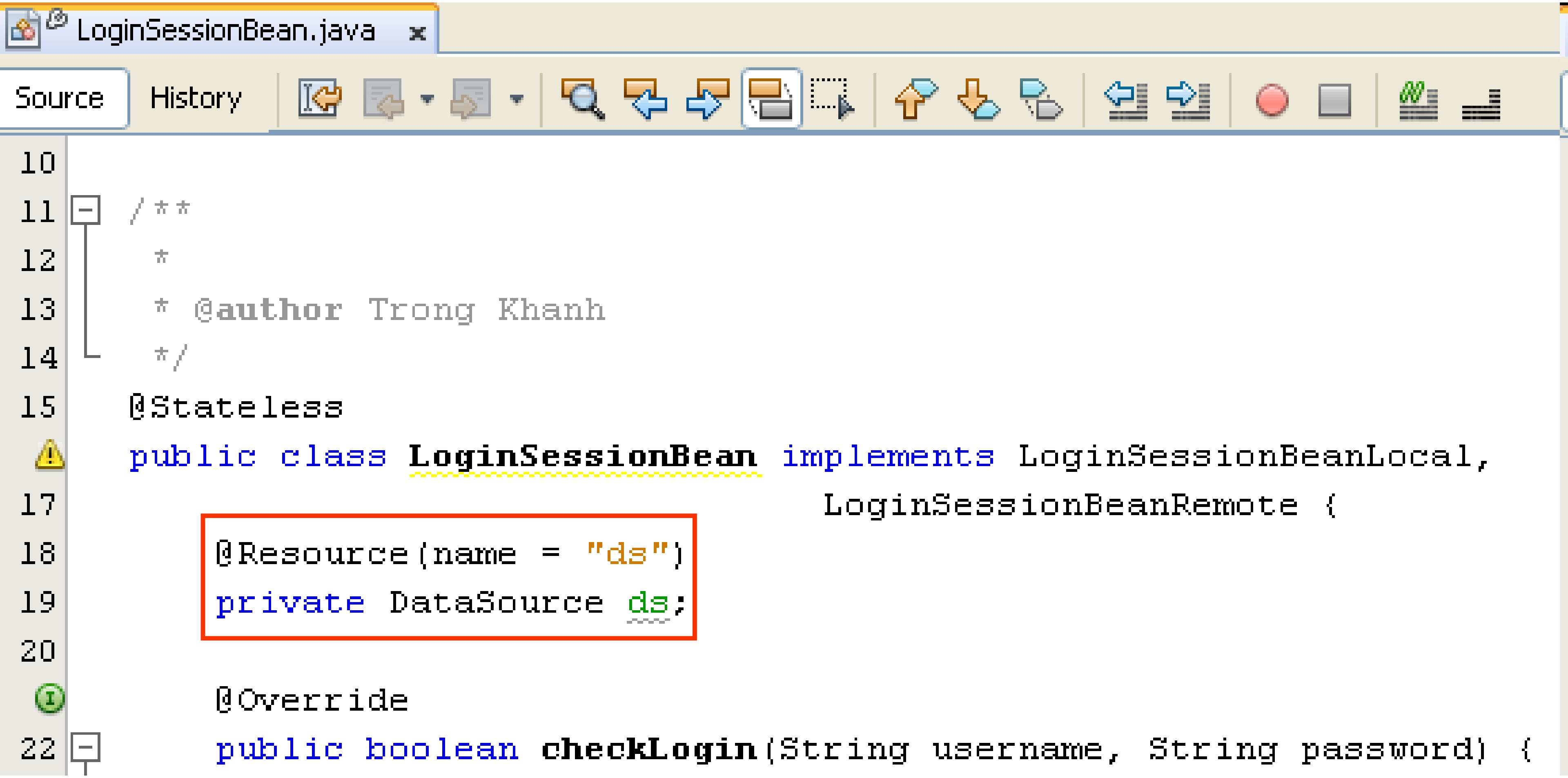
● Generate Inline Lookup Code

○ Existing Class

- Click OK button, the code is generated

EJB Implementation

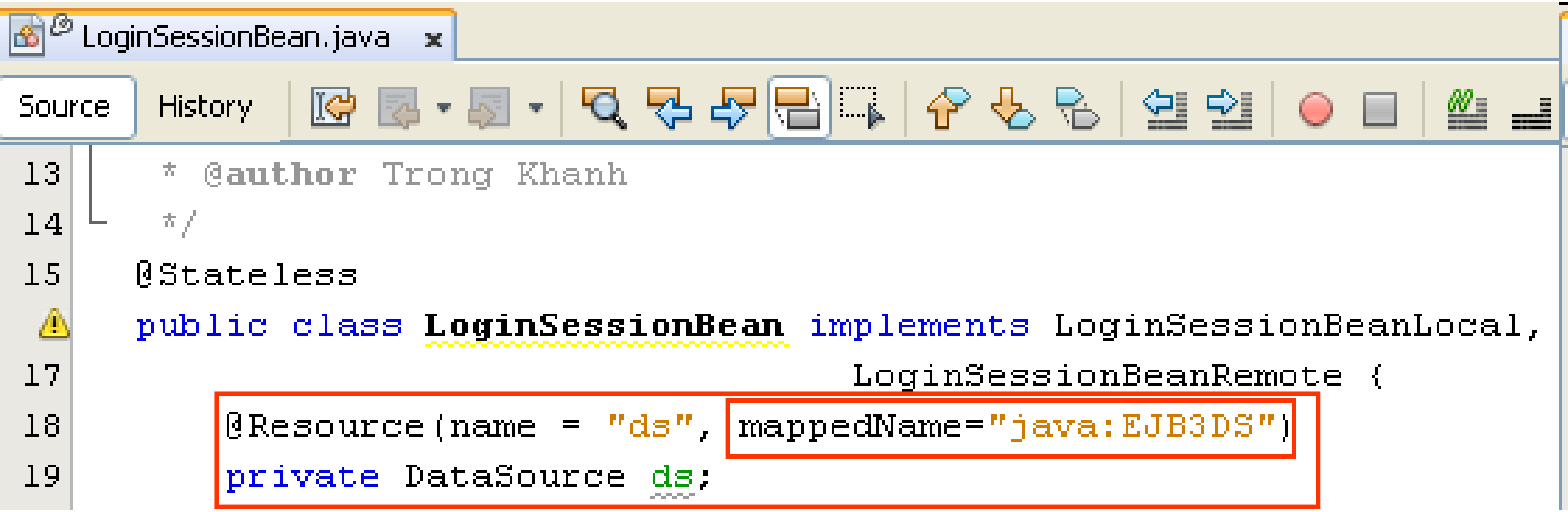
Addition – Using Resource in Bean



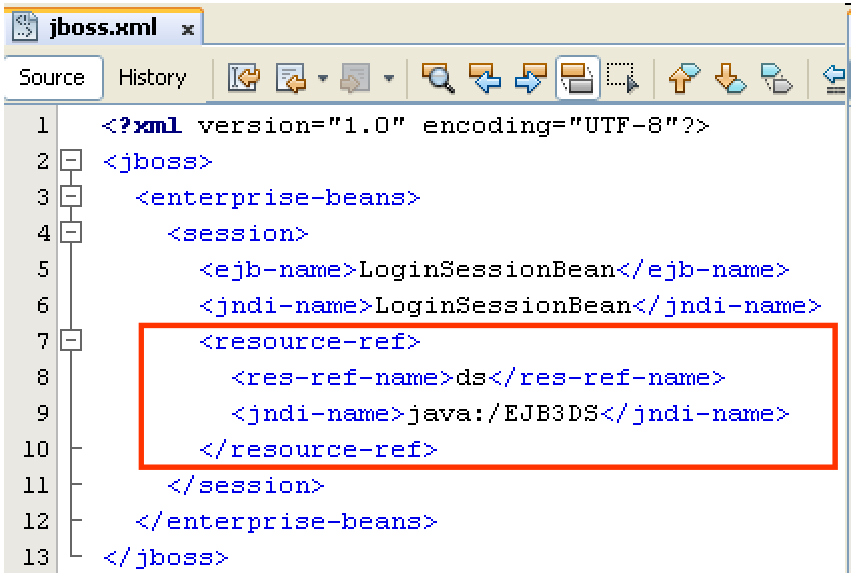
```
/**
 *
 * @author Trong Khanh
 */
@Stateless
public class LoginSessionBean implements LoginSessionBeanLocal,
                                         LoginSessionBeanRemote {
    @Resource(name = "ds")
    private DataSource ds;

    @Override
    public boolean checkLogin(String username, String password) {
```

EJB Implementation

Using Resource in Bean – determine DS name



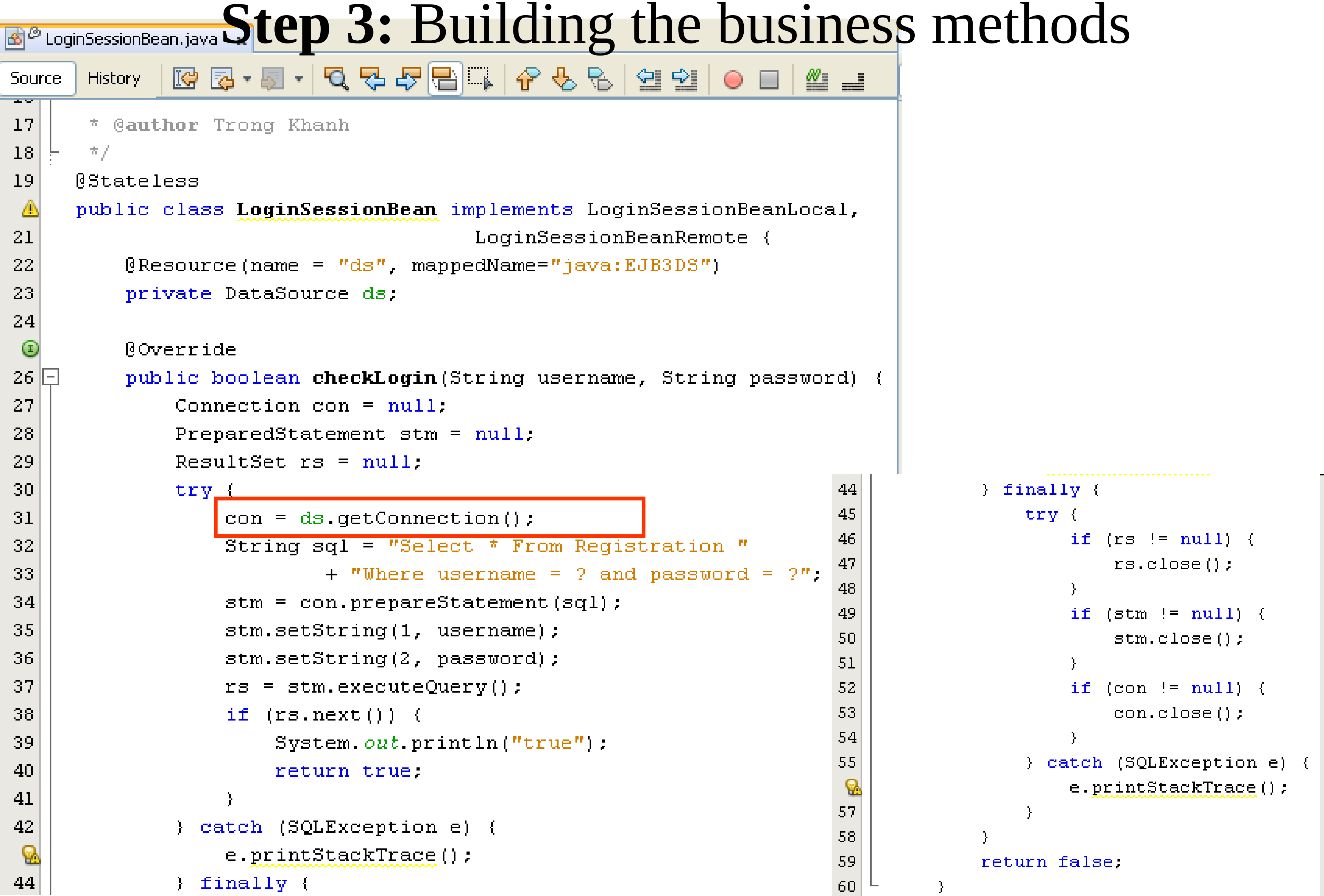
```java
   * @author Trong Khanh
   */
  @Stateless
  public class LoginSessionBean implements LoginSessionBeanLocal,
                                           LoginSessionBeanRemote {
      @Resource(name = "ds", mappedName="java:EJB3DS")
      private DataSource ds;
```



```xml
<?xml version="1.0" encoding="UTF-8"?>
<jboss>
  <enterprise-beans>
    <session>
      <ejb-name>LoginSessionBean</ejb-name>
      <jndi-name>LoginSessionBean</jndi-name>
      <resource-ref>
        <res-ref-name>ds</res-ref-name>
        <jndi-name>java:/EJB3DS</jndi-name>
      </resource-ref>
    </session>
  </enterprise-beans>
</jboss>
```

EJB Implementation

Step 3: Building the business methods

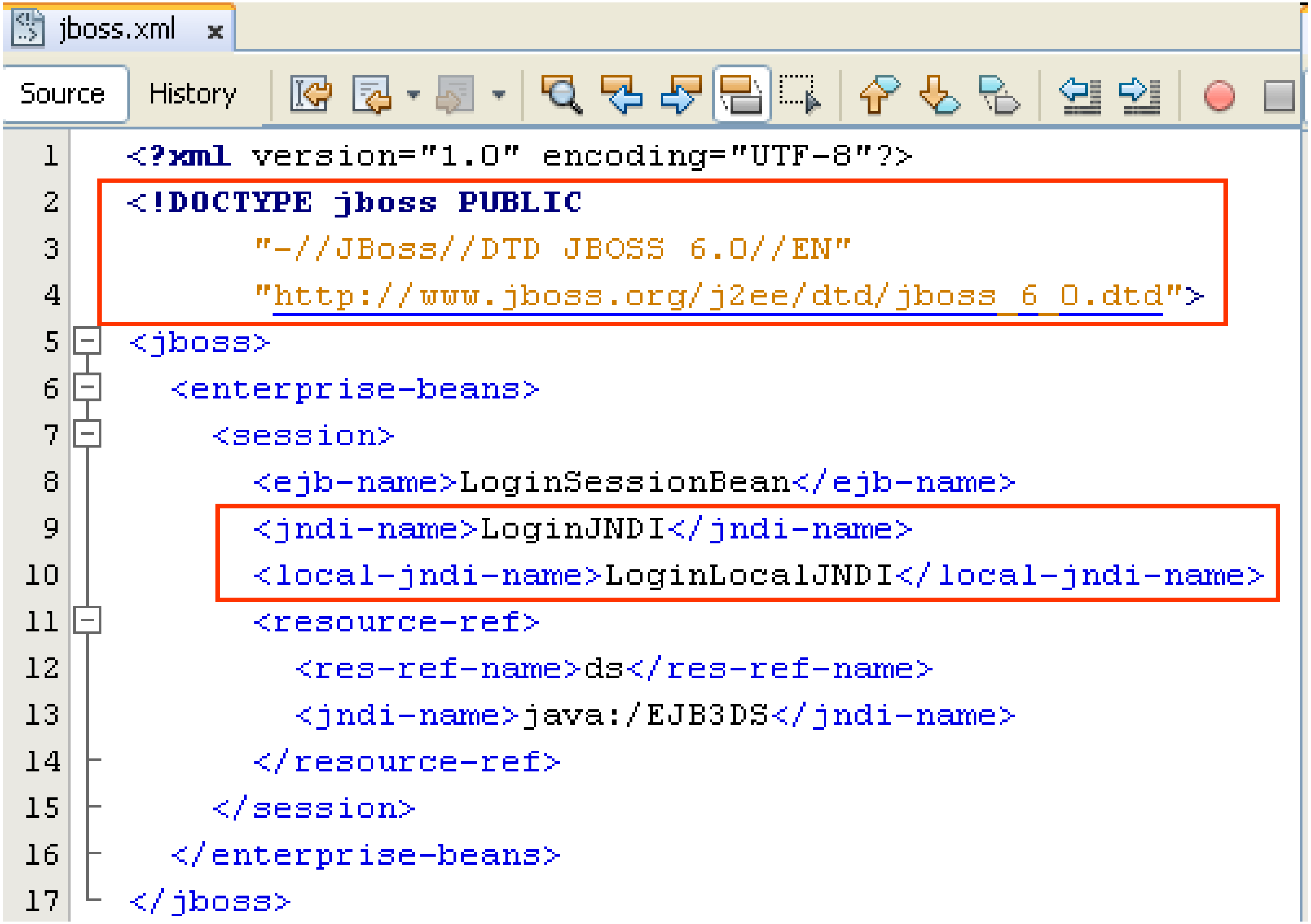


```
 * @author Trong Khanh
 */
@Stateless
public class LoginSessionBean implements LoginSessionBeanLocal,
                                         LoginSessionBeanRemote {
    @Resource(name = "ds", mappedName="java:EJB3DS")
    private DataSource ds;

    @Override
    public boolean checkLogin(String username, String password) {
        Connection con = null;
        PreparedStatement stm = null;
        ResultSet rs = null;
        try {
            con = ds.getConnection();
            String sql = "Select * From Registration "
                    + "Where username = ? and password = ?";
            stm = con.prepareStatement(sql);
            stm.setString(1, username);
            stm.setString(2, password);
            rs = stm.executeQuery();
            if (rs.next()) {
                System.out.println("true");
                return true;
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
```

```
        } finally {
            try {
                if (rs != null) {
                    rs.close();
                }
                if (stm != null) {
                    stm.close();
                }
                if (con != null) {
                    con.close();
                }
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
        return false;
    }
```

EJB Implementation

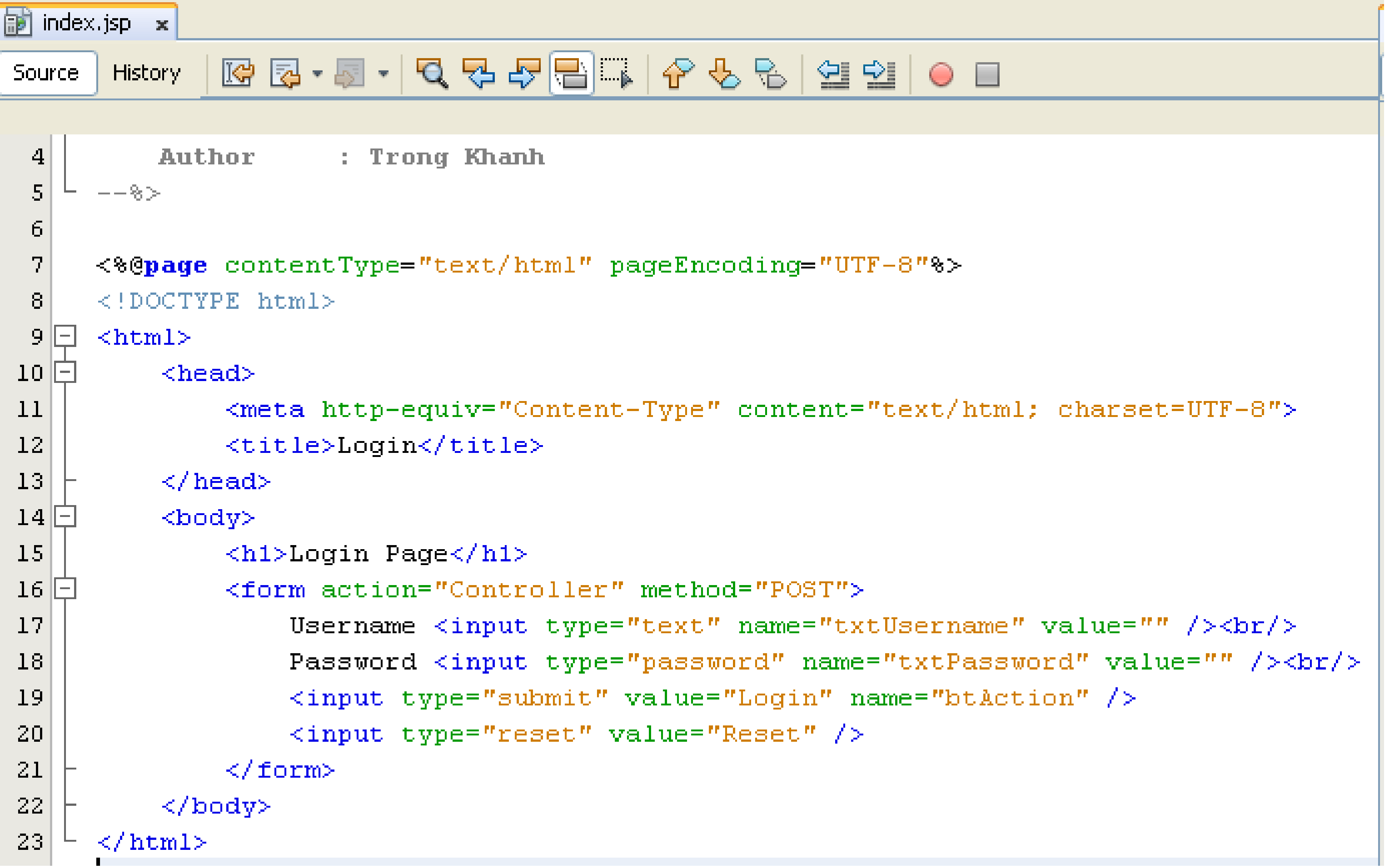
Step 4: Mapping the JNDI to beans



```
<?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>LoginSessionBean</ejb-name>
      <jndi-name>LoginJNDI</jndi-name>
      <local-jndi-name>LoginLocalJNDI</local-jndi-name>
      <resource-ref>
        <res-ref-name>ds</res-ref-name>
        <jndi-name>java:/EJB3DS</jndi-name>
      </resource-ref>
    </session>
  </enterprise-beans>
</jboss>
```

EJB Implementation

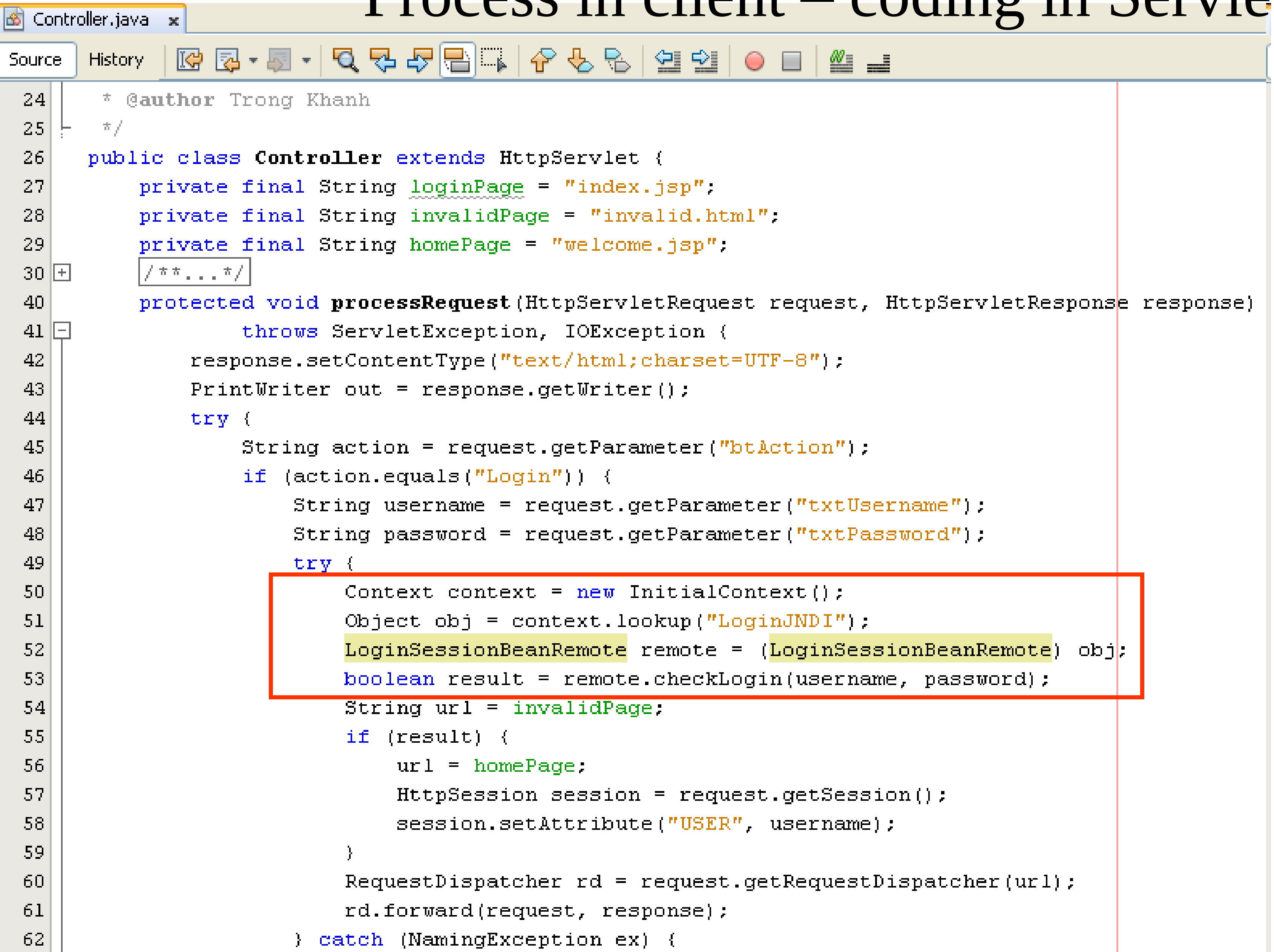
Step 5: Create the UI in the web project



```
    Author       : Trong Khanh
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Login</title>
    </head>
    <body>
        <h1>Login Page</h1>
        <form action="Controller" method="POST">
            Username <input type="text" name="txtUsername" value="" /><br/>
            Password <input type="password" name="txtPassword" value="" /><br/>
            <input type="submit" value="Login" name="btAction" />
            <input type="reset" value="Reset" />
        </form>
    </body>
</html>
```

EJB Implementation

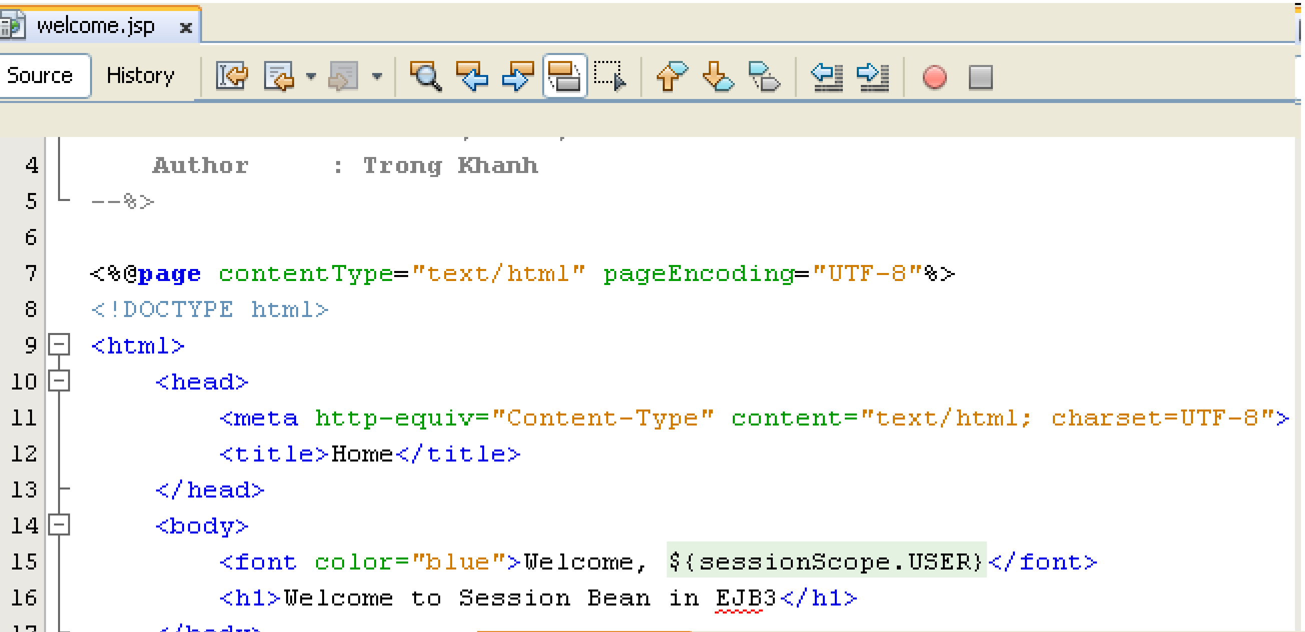
Process in client – coding in Servlet



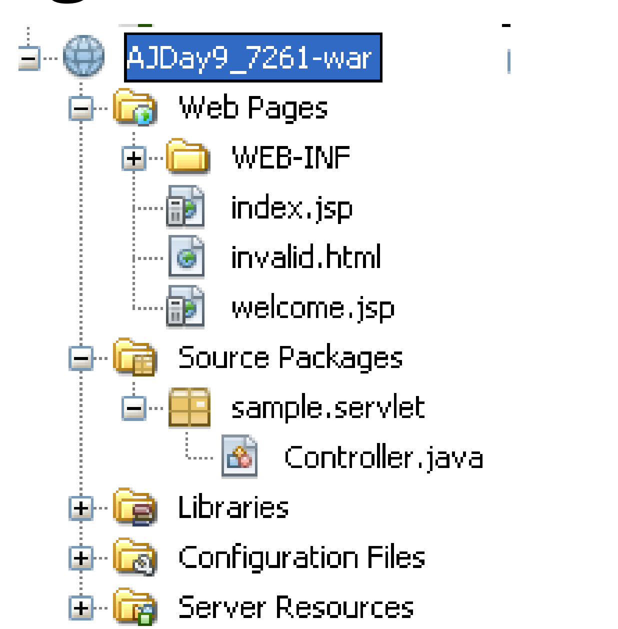
```
 * @author Trong Khanh
 */
public class Controller extends HttpServlet {
    private final String loginPage = "index.jsp";
    private final String invalidPage = "invalid.html";
    private final String homePage = "welcome.jsp";
    /**...*/
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        try {
            String action = request.getParameter("btAction");
            if (action.equals("Login")) {
                String username = request.getParameter("txtUsername");
                String password = request.getParameter("txtPassword");
                try {
                    Context context = new InitialContext();
                    Object obj = context.lookup("LoginJNDI");
                    LoginSessionBeanRemote remote = (LoginSessionBeanRemote) obj;
                    boolean result = remote.checkLogin(username, password);
                    String url = invalidPage;
                    if (result) {
                        url = homePage;
                        HttpSession session = request.getSession();
                        session.setAttribute("USER", username);
                    }
                    RequestDispatcher rd = request.getRequestDispatcher(url);
                    rd.forward(request, response);
                } catch (NamingException ex) {
```

EJB Implementation

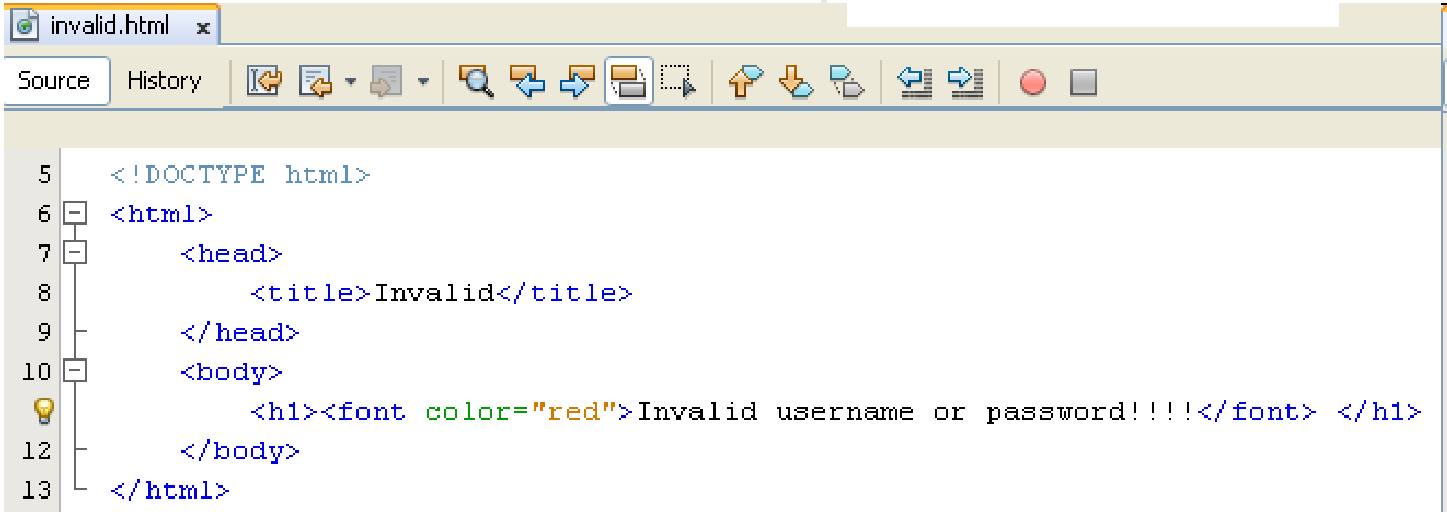
Process in client – coding in Servlet



```
    Author      : Trong Khanh
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Home</title>
    </head>
    <body>
        <font color="blue">Welcome, ${sessionScope.USER}</font>
        <h1>Welcome to Session Bean in EJB3</h1>
    </body>
</html>
```



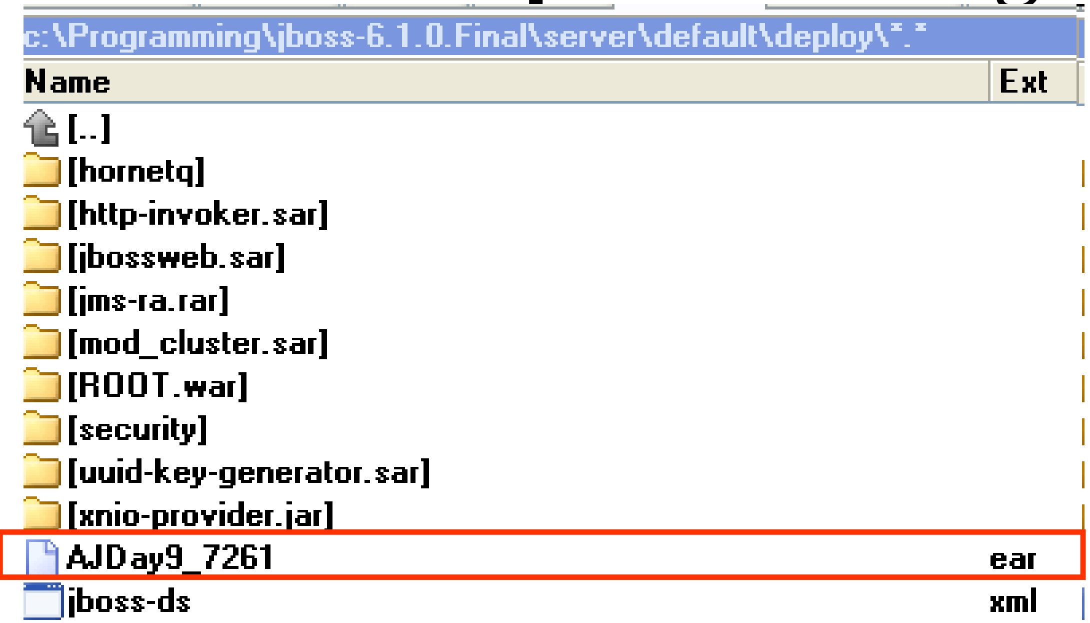
```
AJDay9_7261-war
    Web Pages
        WEB-INF
        index.jsp
        invalid.html
        welcome.jsp
    Source Packages
        sample.servlet
            Controller.java
    Libraries
    Configuration Files
    Server Resources
```



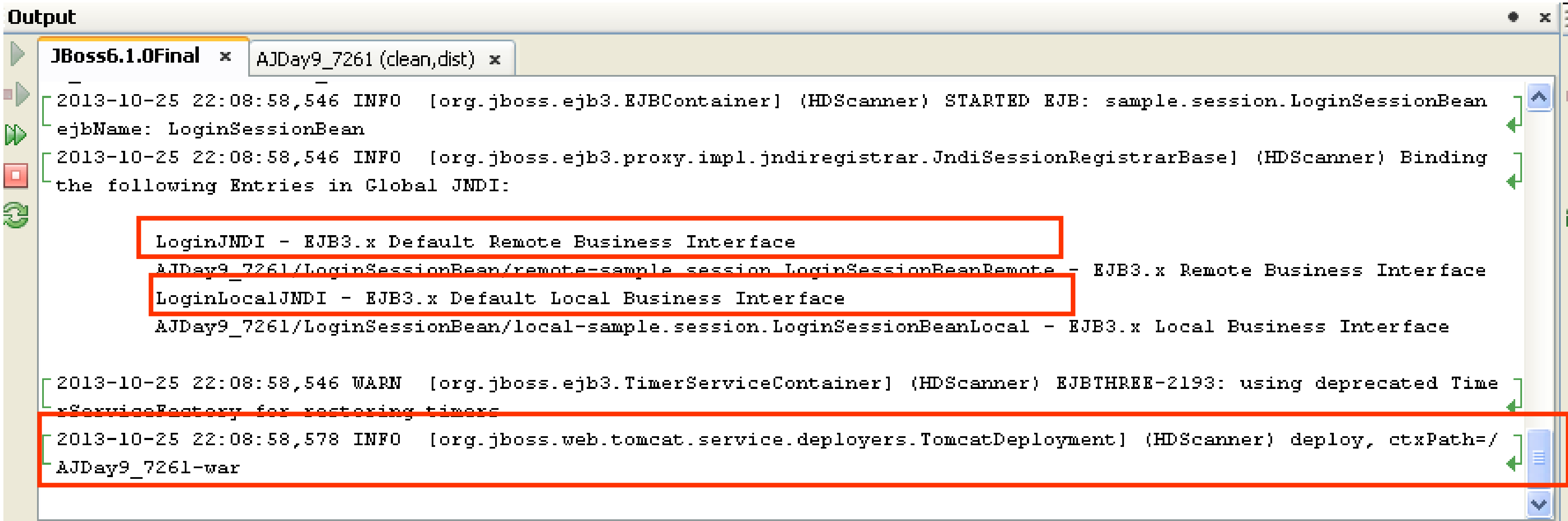
```
<!DOCTYPE html>
<html>
    <head>
        <title>Invalid</title>
    </head>
    <body>
        <h1><font color="red">Invalid username or password!!!!</font> </h1>
    </body>
</html>
```

EJB Implementation

Step 6: Building & Deploying



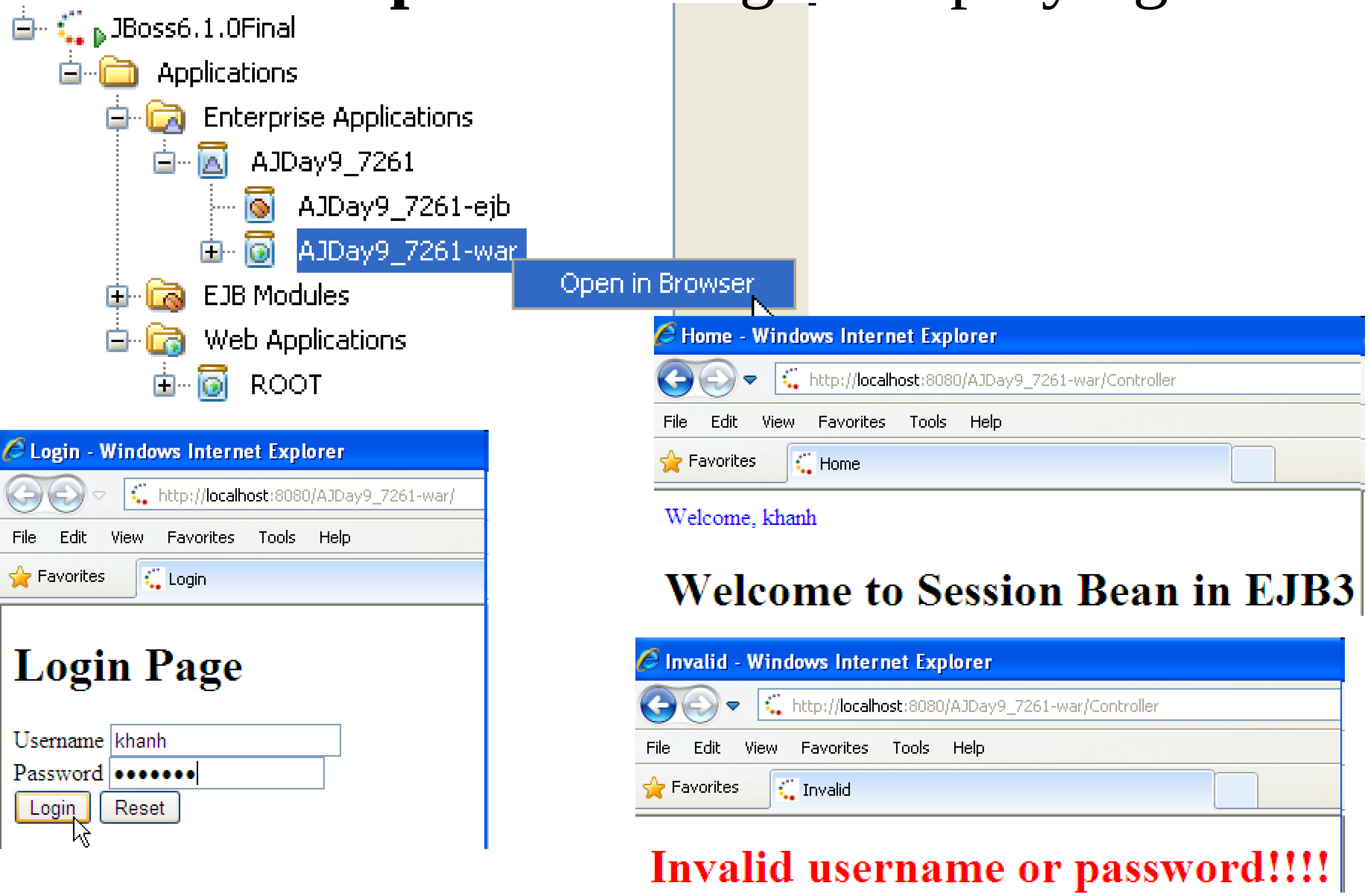
```
c:\Programming\jboss-6.1.0.Final\server\default\deploy\*.*
Name                                   Ext
[..]
[hornetq]
[http-invoker.sar]
[jbossweb.sar]
[jms-ra.rar]
[mod_cluster.sar]
[ROOT.war]
[security]
[uuid-key-generator.sar]
[xnio-provider.jar]
AJDay9_7261                            ear
jboss-ds                               xml
```



```
2013-10-25 22:08:58,546 INFO  [org.jboss.ejb3.EJBContainer] (HDScanner) STARTED EJB: sample.session.LoginSessionBean
ejbName: LoginSessionBean
2013-10-25 22:08:58,546 INFO  [org.jboss.ejb3.proxy.impl.jndiregistrar.JndiSessionRegistrarBase] (HDScanner) Binding
the following Entries in Global JNDI:

        LoginJNDI - EJB3.x Default Remote Business Interface
        AJDay9_7261/LoginSessionBean/remote-sample.session.LoginSessionBeanRemote - EJB3.x Remote Business Interface
        LoginLocalJNDI - EJB3.x Default Local Business Interface
        AJDay9_7261/LoginSessionBean/local-sample.session.LoginSessionBeanLocal - EJB3.x Local Business Interface

2013-10-25 22:08:58,546 WARN  [org.jboss.ejb3.TimerServiceContainer] (HDScanner) EJBTHREE-2193: using deprecated Time
rServiceFactory for restoring timers
2013-10-25 22:08:58,578 INFO  [org.jboss.web.tomcat.service.deployers.TomcatDeployment] (HDScanner) deploy, ctxPath=/
AJDay9_7261-war
```

EJB Implementation

Step 6: Building & Deploying



```
JBoss6.1.0Final
  Applications
    Enterprise Applications
      AJDay9_7261
        AJDay9_7261-ejb
        AJDay9_7261-war
    EJB Modules
    Web Applications
      ROOT
```

http://localhost:8080/AJDay9_7261-war/Controller

Welcome, khanh

Welcome to Session Bean in EJB3

http://localhost:8080/AJDay9_7261-war/

Login Page

Username khanh

Password •••••••

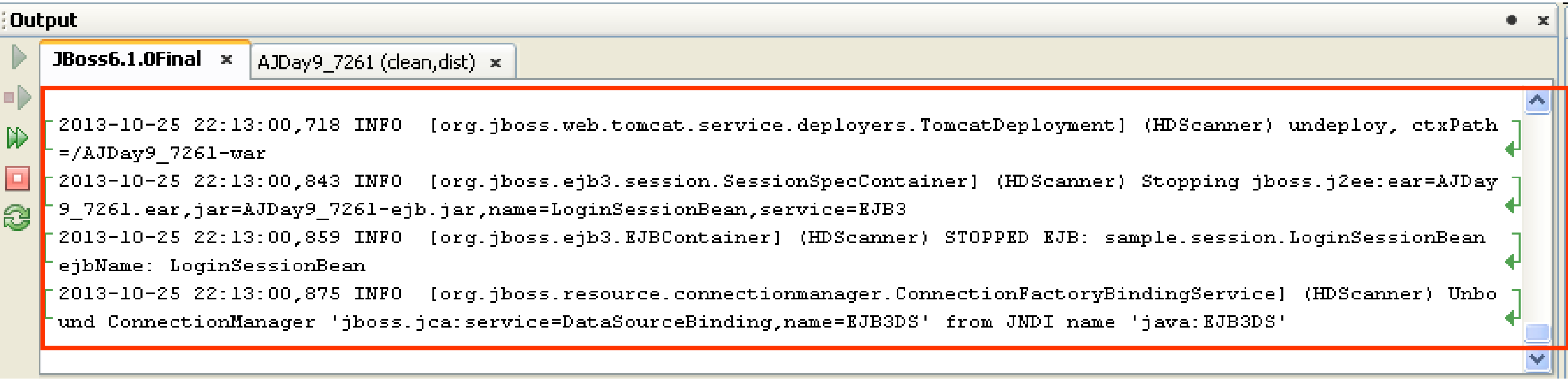
Login Reset

http://localhost:8080/AJDay9_7261-war/Controller

Invalid username or password!!!!

EJB Implementation

Un-Deploying

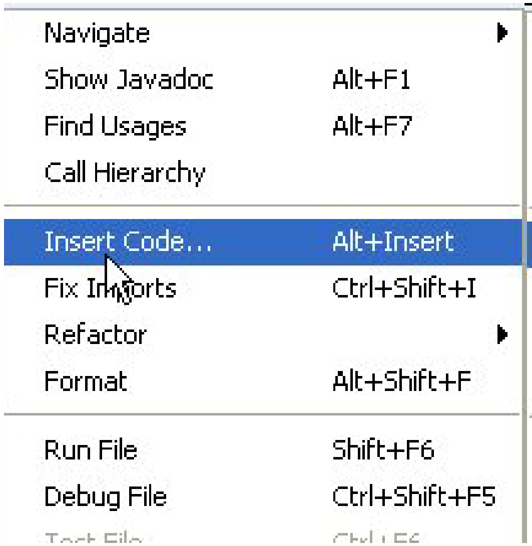


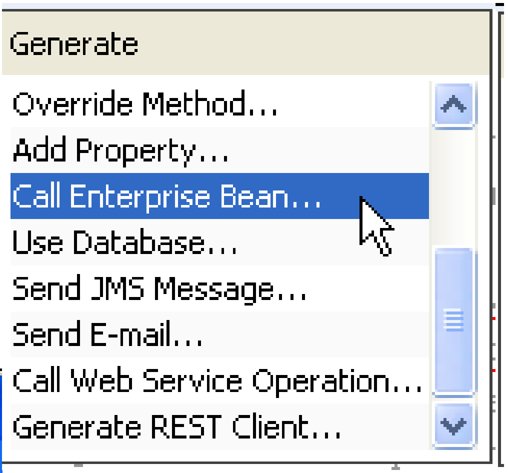
```
2013-10-25 22:13:00,718 INFO  [org.jboss.web.tomcat.service.deployers.TomcatDeployment] (HDScanner) undeploy, ctxPath
=/AJDay9_7261-war
2013-10-25 22:13:00,843 INFO  [org.jboss.ejb3.session.SessionSpecContainer] (HDScanner) Stopping jboss.j2ee:ear=AJDay
9_7261.ear,jar=AJDay9_7261-ejb.jar,name=LoginSessionBean,service=EJB3
2013-10-25 22:13:00,859 INFO  [org.jboss.ejb3.EJBContainer] (HDScanner) STOPPED EJB: sample.session.LoginSessionBean
ejbName: LoginSessionBean
2013-10-25 22:13:00,875 INFO  [org.jboss.resource.connectionmanager.ConnectionFactoryBindingService] (HDScanner) Unbo
und ConnectionManager 'jboss.jca:service=DataSourceBinding,name=EJB3DS' from JNDI name 'java:EJB3DS'
```

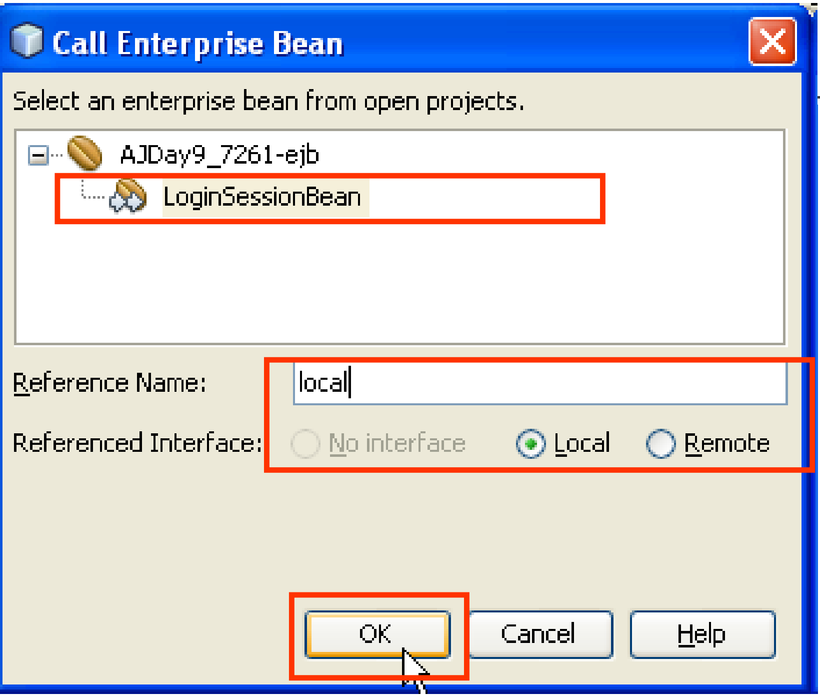
EJB Implementation

Process in client – coding in Servlet – other ways

- In the Servlet code
  - Right click on servlet code, click “Insert Code…”
  - Click “Call Enterprise Bean”







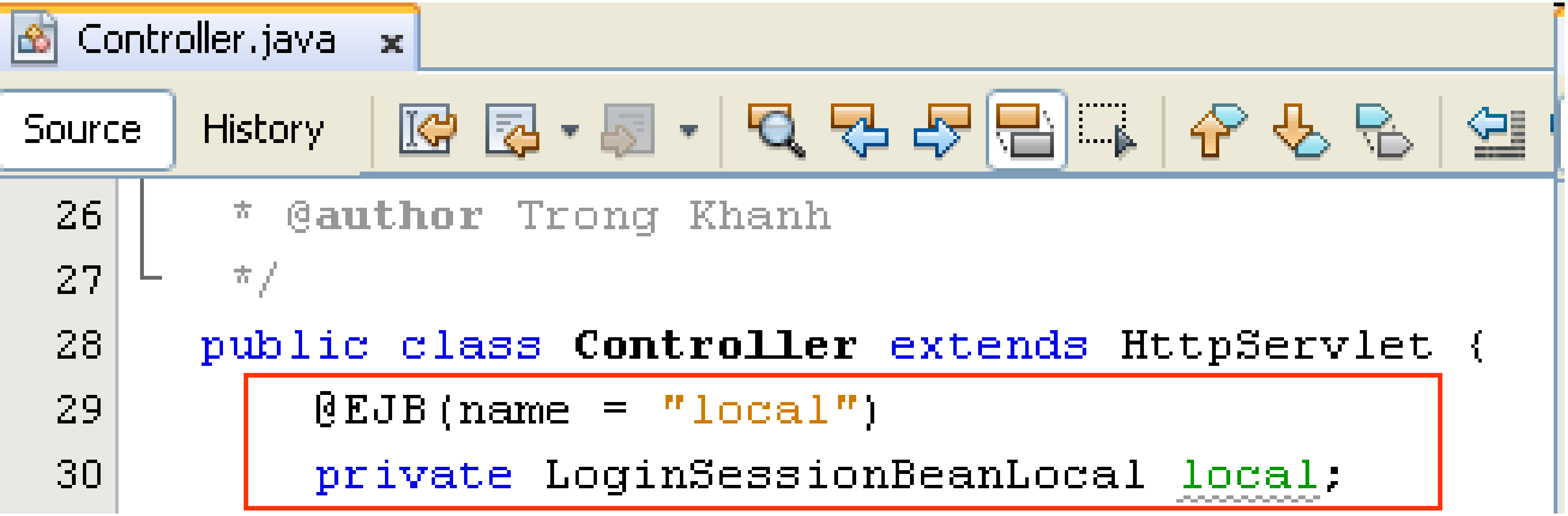
Reference Name: local

Referenced Interface: ○ No interface ● Local ○ Remote

- In Call Enterprise Bean dialog
  - Choose the bean
  - Input the Reference Name
  - Click OK

EJB Implementation

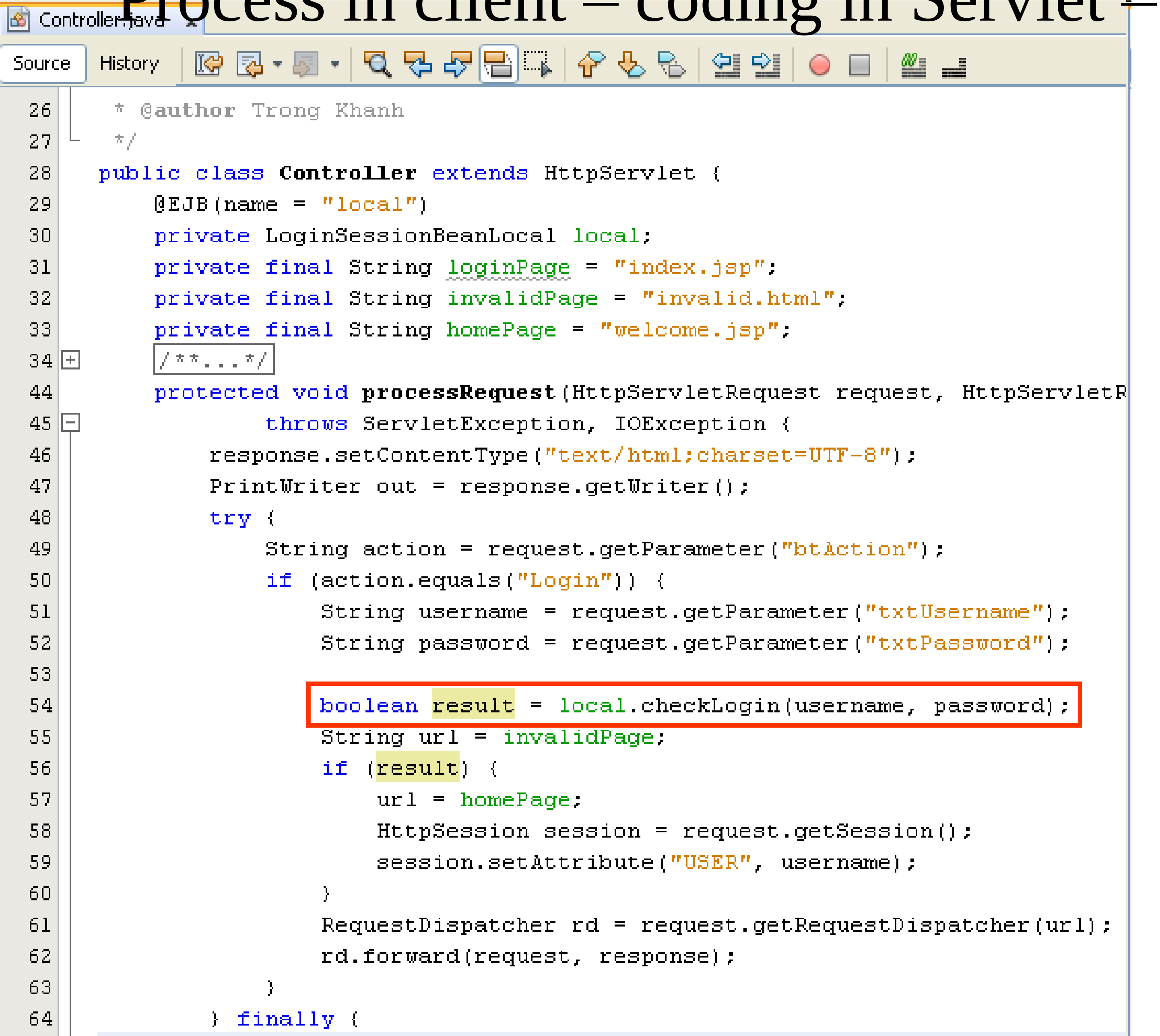
Process in client – coding in Servlet – other ways



```
     * @author Trong Khanh
     */
public class Controller extends HttpServlet {
    @EJB(name = "local")
    private LoginSessionBeanLocal local;
```

EJB Implementation

Process in client – coding in Servlet – other ways



```
     * @author Trong Khanh
     */
public class Controller extends HttpServlet {
    @EJB(name = "local")
    private LoginSessionBeanLocal local;
    private final String loginPage = "index.jsp";
    private final String invalidPage = "invalid.html";
    private final String homePage = "welcome.jsp";
    /**...*/
    protected void processRequest(HttpServletRequest request, HttpServletR
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        try {
            String action = request.getParameter("btAction");
            if (action.equals("Login")) {
                String username = request.getParameter("txtUsername");
                String password = request.getParameter("txtPassword");

                boolean result = local.checkLogin(username, password);
                String url = invalidPage;
                if (result) {
                    url = homePage;
                    HttpSession session = request.getSession();
                    session.setAttribute("USER", username);
                }
                RequestDispatcher rd = request.getRequestDispatcher(url);
                rd.forward(request, response);
            }
        } finally {
```