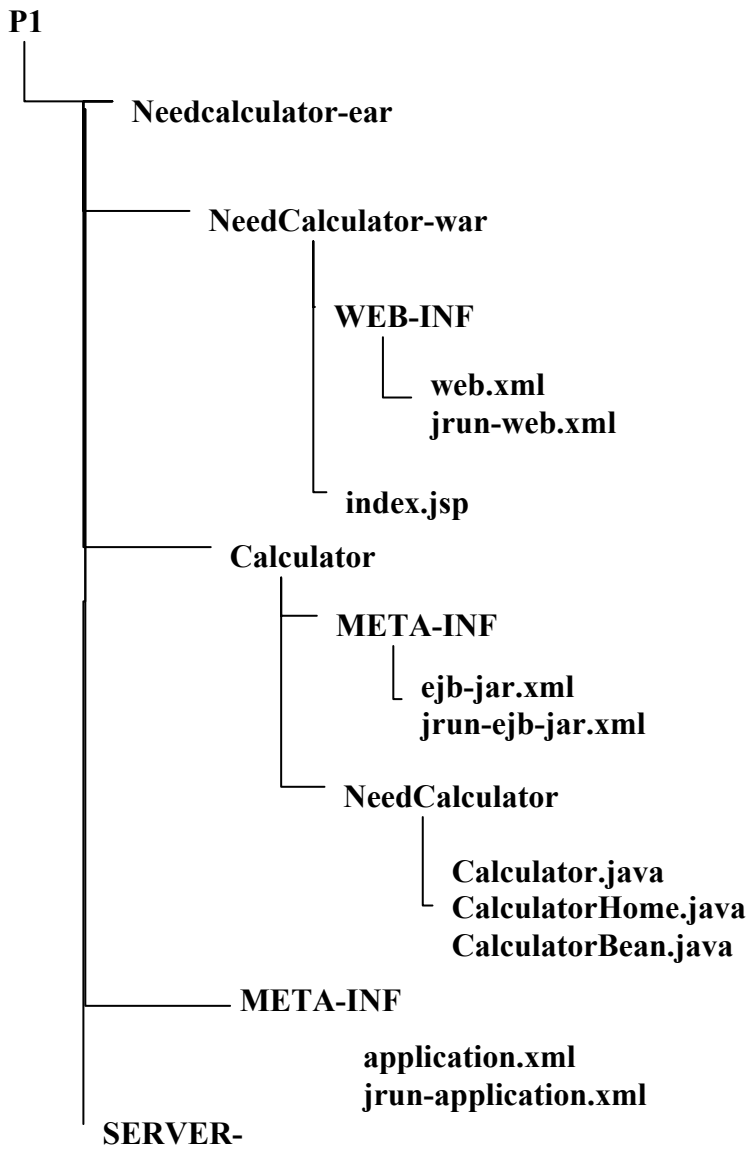
Application Design and Development Using
Java 2 Enterprise Edition (J2EE)
CSE4/587 Information Structures

Objective is to explore the complete process from coding to deployment and testing of a simple ejb-based web application.

1. We will create the server and application structure shown below.



2. Start the admin server and the JMC. Add a new server P1 (use lecture notes).

3. Start the server. We will use the directory structure provided by the default-ear and modify it to suit our new application NeedCalculator.

4. Rename the default-ear into NeedCalculator-ear

5. Rename the war directory NeedCalculator-war

6. Update the META-INF directory, application.xml to reflect the changes above. Also study the tags.

7. Rename default-ejb of the NeedCalculator-ear directory to Calculator.

8. Create the EJB files using the JRunWizard tool ; Since this tool automatically creates its META-INF directory of the Calculator, remove the ejb-jar.xml file in the META-INF corresponding to the Calculator.

9. Start the Jrunwizard tool. Start→run→command
Change directory to c:\jrun4\bin
Type jrunwizard at the command prompt.

a. File →new project→ provide a name FNC
b. Provide the source and destination directory for the bean files the path of the Calculator file by using the Browse button.
   C:\jrun4\servers\P1\NeedCalculator-ear\Calculator
c. Click ok, click on the EJB icon on the top line menu. (brown sphere)
d. Click on the new bean descriptor radio button at the top of the screen
e. When dialog box appears, click on stateless session bean
f. On the next dialog box provide the package name a NeedCalculator, bean name as Calculator, click on to the next dialog box
g. File→save
h. Click on edit bean descriptor radio button, and click on generate code.
i. Add methods to Remote interface and method definition to Bean:

```
public double calc (double cost, double avail) throws
  java.rmi.RemoteException; //  in remote interface
public double calc (double cost, double avail)
      { return (cost -avail); }  // in bean class
```

j. Click on the Compile button in the top line menu. Press compile in the window that appears.
k. Examine the Calculator/NeedCalculator directory and make sure java source and class files are present.
l. Minimize the wizard and move to the JMC to deploy the bean.

10. In the JMC refresh/restart the P1. Click on P1 server on the main windows, when components of the P1 server appears, deploy the NeedCalculator-ear by clicking the button on the left of it.

11. Prepare the war files. Copy the web.xml given below into the web.xml of the WEB-INF directory of the NeedCalculator-war.

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN" "http://java.sun.com/dtd/web-app_2_3.dtd">

<web-app>

<welcome-file-list>
  <welcome-file>index.jsp</welcome-file>
 </welcome-file-list>

 <!-- This application uses ejb-refs so that clients can always locate
      the ejb under the java:comp/env environment naming context (ENC).
      The jrun-web.xml file maps the ejb-ref-name to the actual JNDI location.
      Clients can then lookup the EJB using either the actual JNDI location
      or java:comp/env/*ejb-ref-name* -->
 <ejb-ref>
   <description>Calculator session bean</description>
   <ejb-ref-name>ejb/Calculator</ejb-ref-name>
   <ejb-ref-type>Session</ejb-ref-type>
   <home>NeedCalculator.CalculatorHome</home>
   <remote>NeedCalculator.Calculator</remote>
  </ejb-ref>
</web-app>
```

12. Prepare a simple index.jsp file to access the bean. You may modify the existing index.jsp or copy the following code into the index.jsp file.

```
<%@ page import="NeedCalculator.*" %>

<html>
<head>
<title>Confirmation</title>
</head>
<body>
<%
        try {
                javax.naming.InitialContext ctx = new javax.naming.InitialContext();
                Object obj = ctx.lookup("java:comp/env/ejb/Calculator");
                CalculatorHome home =
(CalculatorHome)javax.rmi.PortableRemoteObject.narrow(obj, CalculatorHome.class);
                Calculator needCalc = home.create();
                double need = needCalc.calc(10000,5000);
                out.println("Your Need is = $" + need);
%>
                Thank you. Your Need has been determined.<br><br>
```

```
<%
        } catch (Exception e) {
%>
                Sorry, unable to compute your need.
                <br>Ask your system administrator to review the log files.
<%
        }
%>

</body>
</html>
```

13. The application has been hot deployed and ready. Start a web browser and open page: http://localhost:8102/NeedCaculator

14. You should see the “financial need” displayed.

15. Now go through the “compass” application available in the samples server.