Consuming BLOB Type through Web Service Using Deployable Proxy

Summary

A web service return a BLOB type data in a byte array. A web module gets this byte array and displays it in client browser. Web module consumes the web service by using deployable proxy.

Author: Ashok Kumar Sarkar

Company: HCL Technologies Kolkata

Created on: 14 June 2007

Author Bio

Ashok Kumar Sarkar is working as a NetWeaver Associate Consultant for HCL Technologies Kolkata

Table of Contents

Introduction

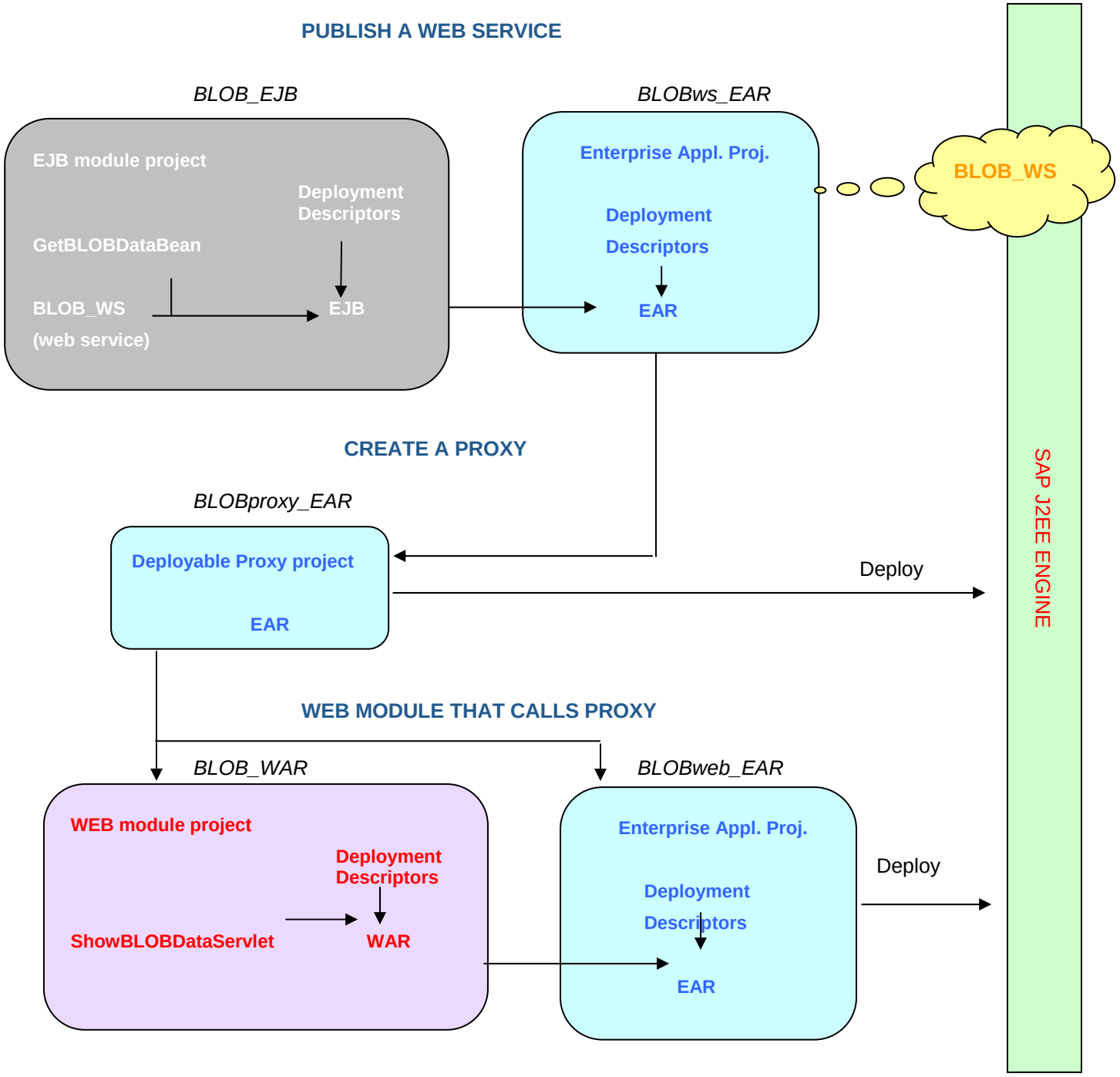
Due to the growing digitalization of images, spreadsheets and videos as well as any type of documents, applications require more and more to organize and maintain this information that, in many cases, is of critical importance for the organization.

The Blob data type allows storing this information in the database, thus taking advantage of the different mechanisms of integrity and control provided by the DMBSs. i.e.: it allows seeing this information as one more data to be handled.

This article will guide the users through

- Returning BLOB field through web service
- Use deployable proxy.

Structure Diagram of Whole Process



Steps for Whole Process:

- Publish a Web Service.
- Create a Proxy.
- Web Module that calls Proxy.

Publish Web Service:

EJB module (BLOB_EJB)

1. Create a stateless bean GetBLOBDataBean and define a function getBLOBData which is returning a byte array.

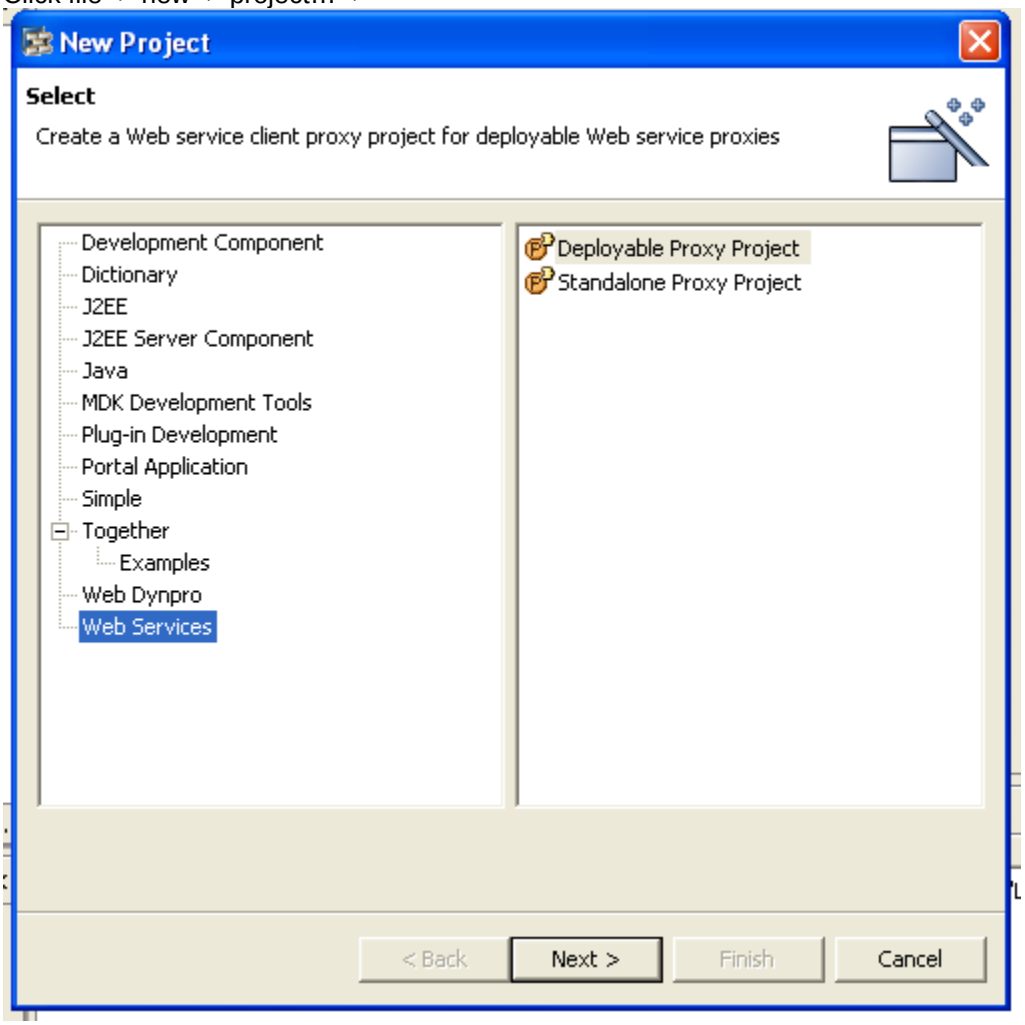
```
public byte[] getBLOBData()        {


//using Prepared Statement gets the BLOB data from the table
        String sql="select pic from test_blob where picno=11";
   java.sql.PreparedStatement  ps=con.prepareStatement(sql);
   java.sql.ResultSet rs=ps.executeQuery();
rs.next();
   java.io.InputStream sImage = rs.getBinaryStream("pic");
   ByteArrayOutputStream output = new ByteArrayOutputStream();
   byte[] rb = new byte[1024];
   int ch = 0;
   while ((ch=sImage.read(rb)) != -1){
            output.write(rb, 0, ch);
      }
   byte b[] = output.toByteArray();
   return b;
}
```

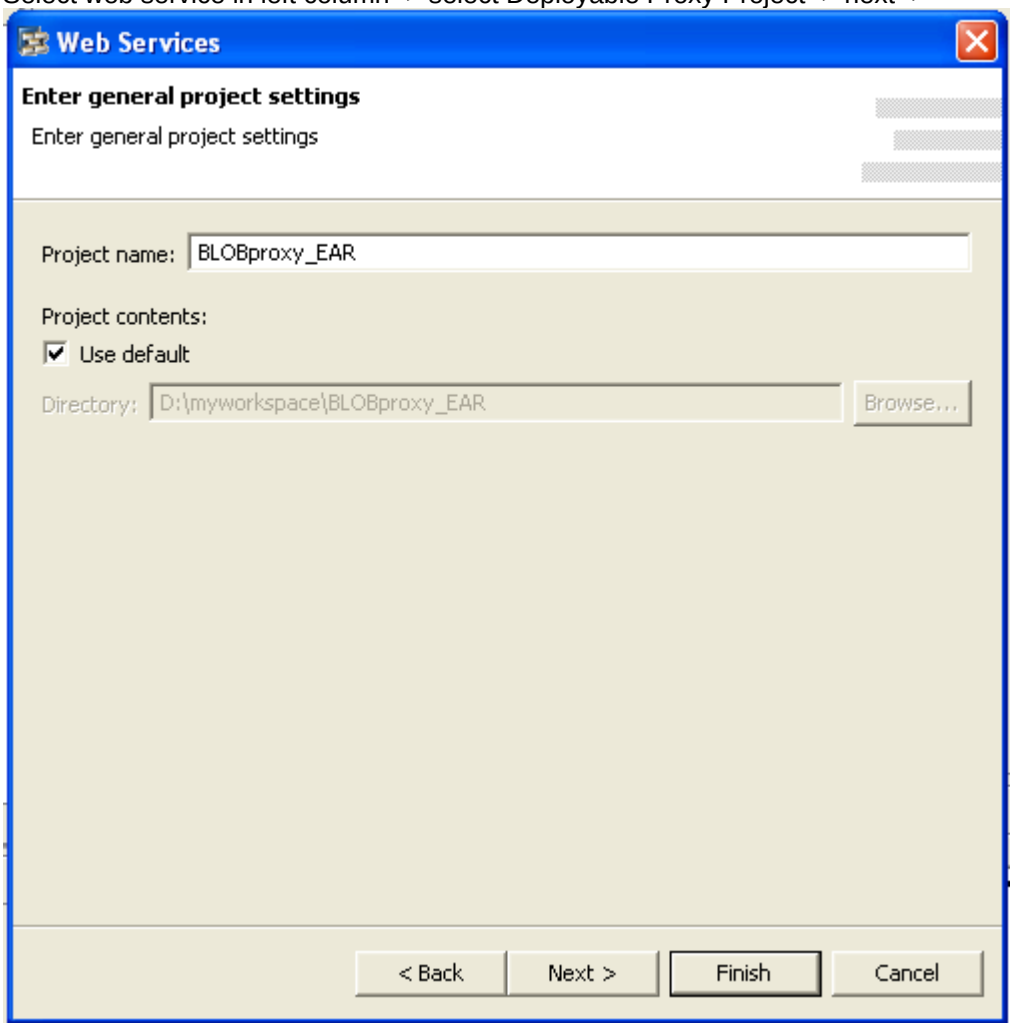
2. Create a web service BLOB_WS for the stateless bean
3. Add this EJB module to an EAR module BLOBws_EAR.
4. Deploy the EAR module

Create a Deployable Proxy

1. Select the web service perspective
2. Click file -> new -> project… ->

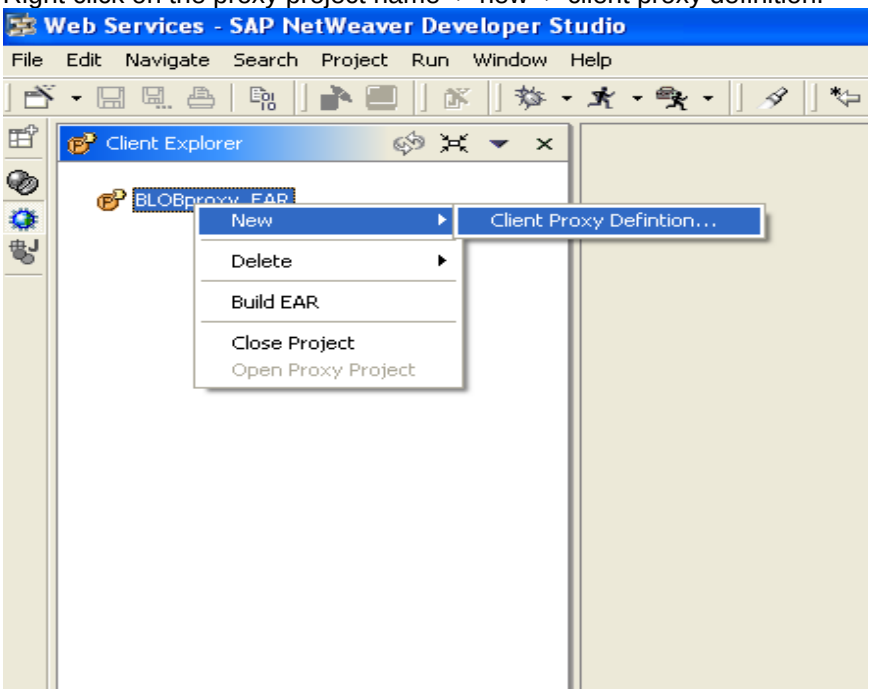


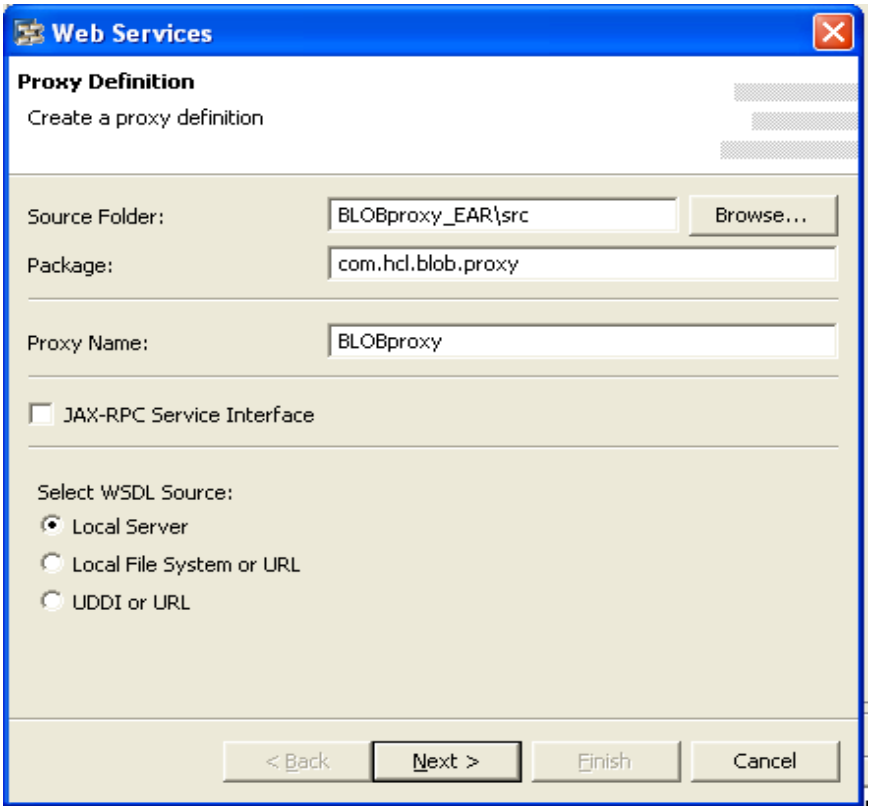
3. Select web service in left column -> select Deployable Proxy Project -> next ->



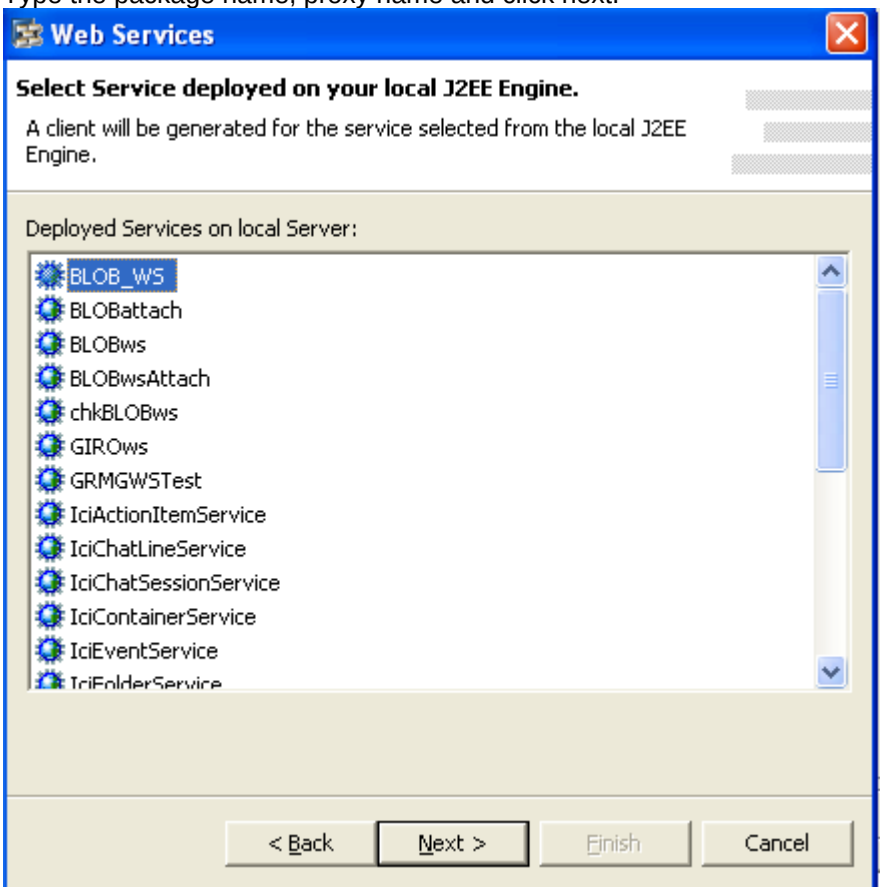
4. Give a project name -> finish.

5. Right click on the proxy project name -> new -> client proxy definition.

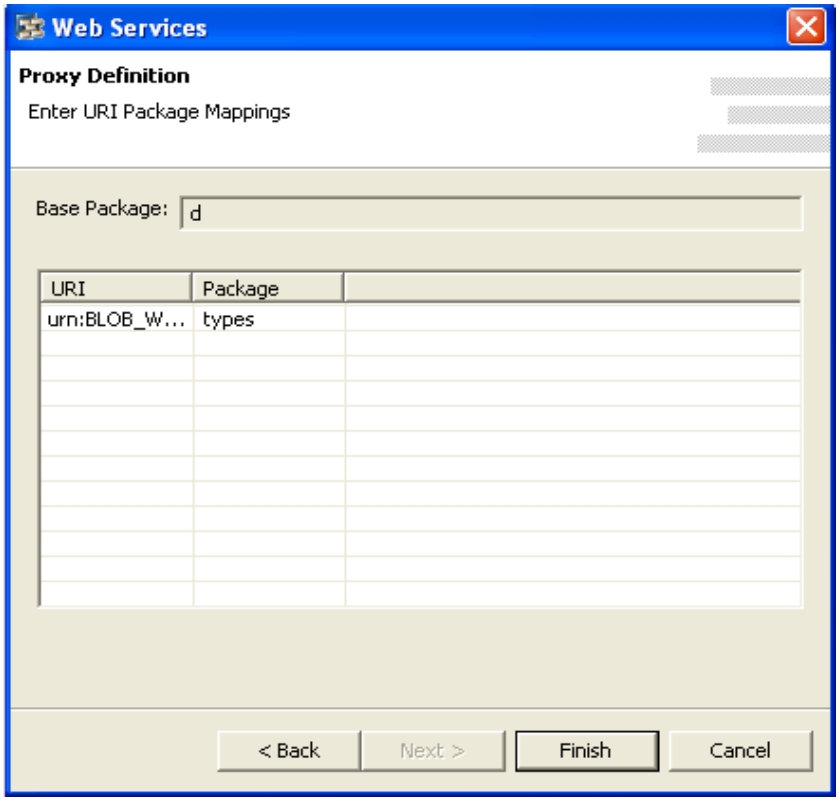




6. Type the package name, proxy name and click next.

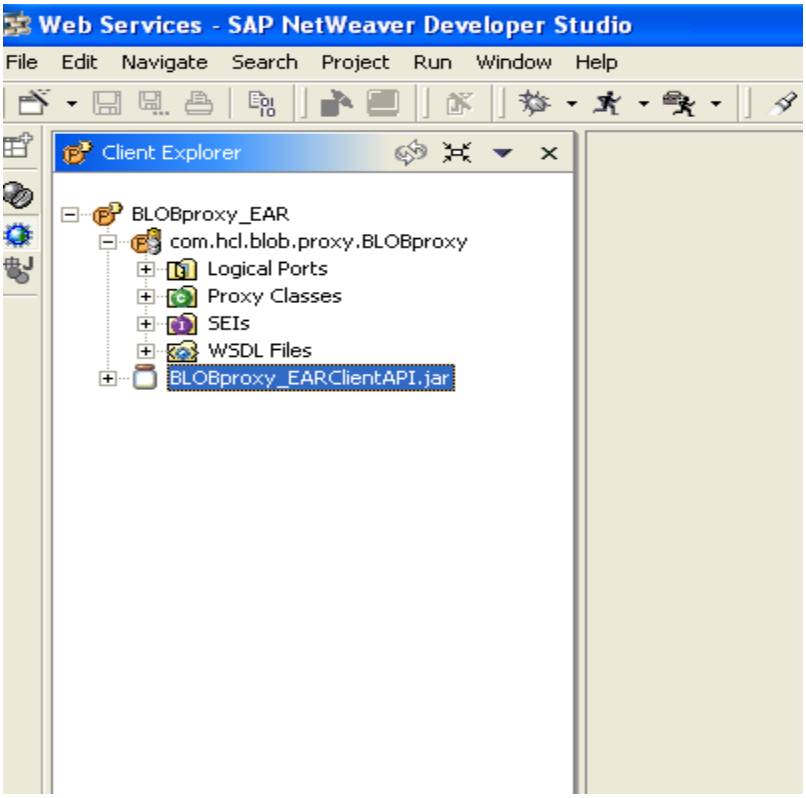


7. Select the web service and click next.

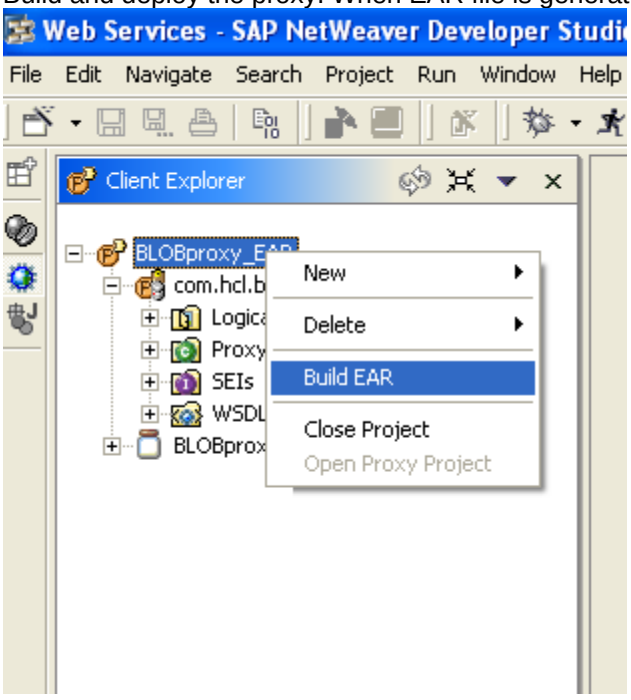


8. Click finish.
9. Now we can see a jar file in Client Explorer window.

   [Note: If we can't see the jar file then follows the steps from 5 to 8 and creates a temporary client proxy definition. Then we can see the jar file. Now delete this temporary client proxy.]



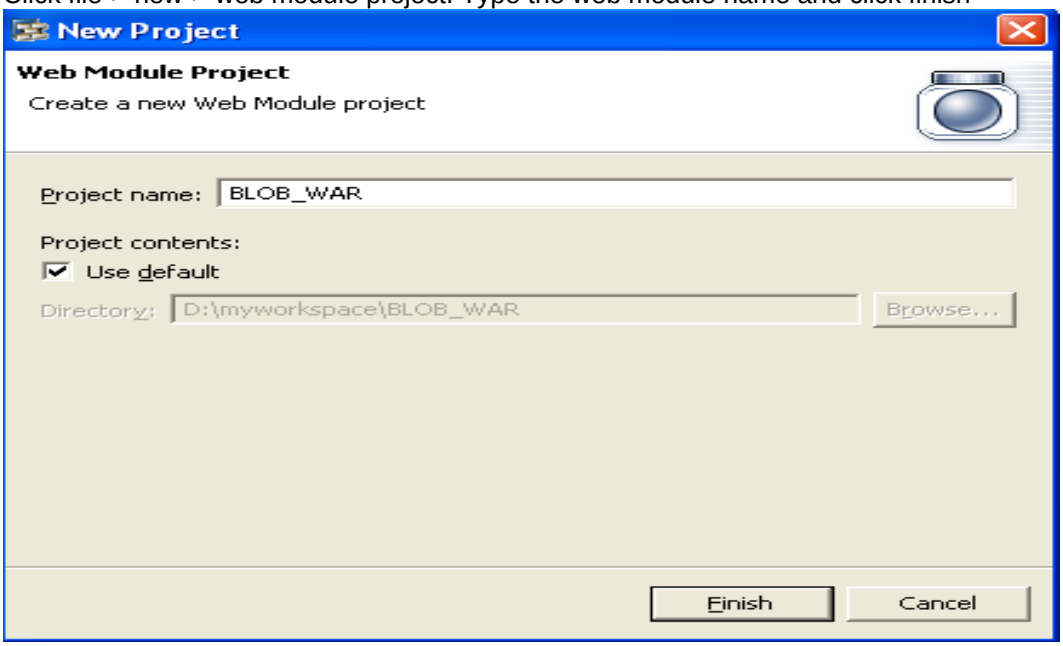
10. Build and deploy the proxy. When EAR file is generated, then select it and deploy it.

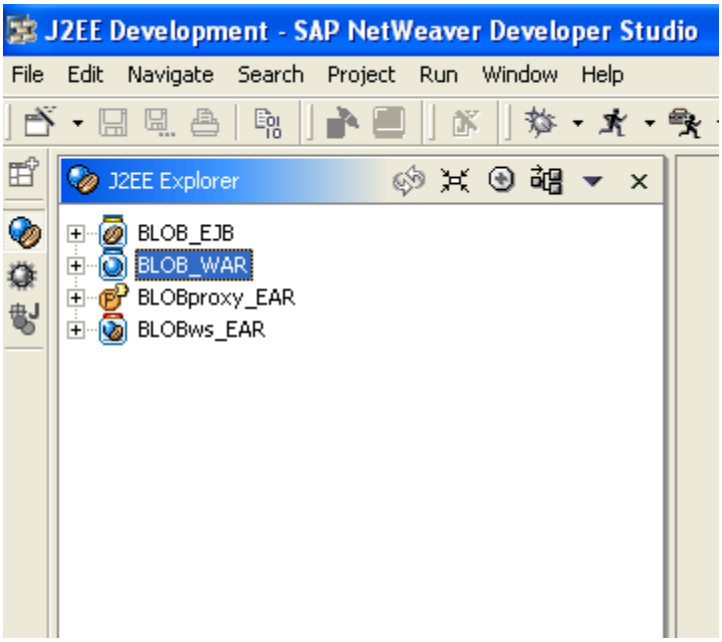


11. Deployable Proxy has been created.

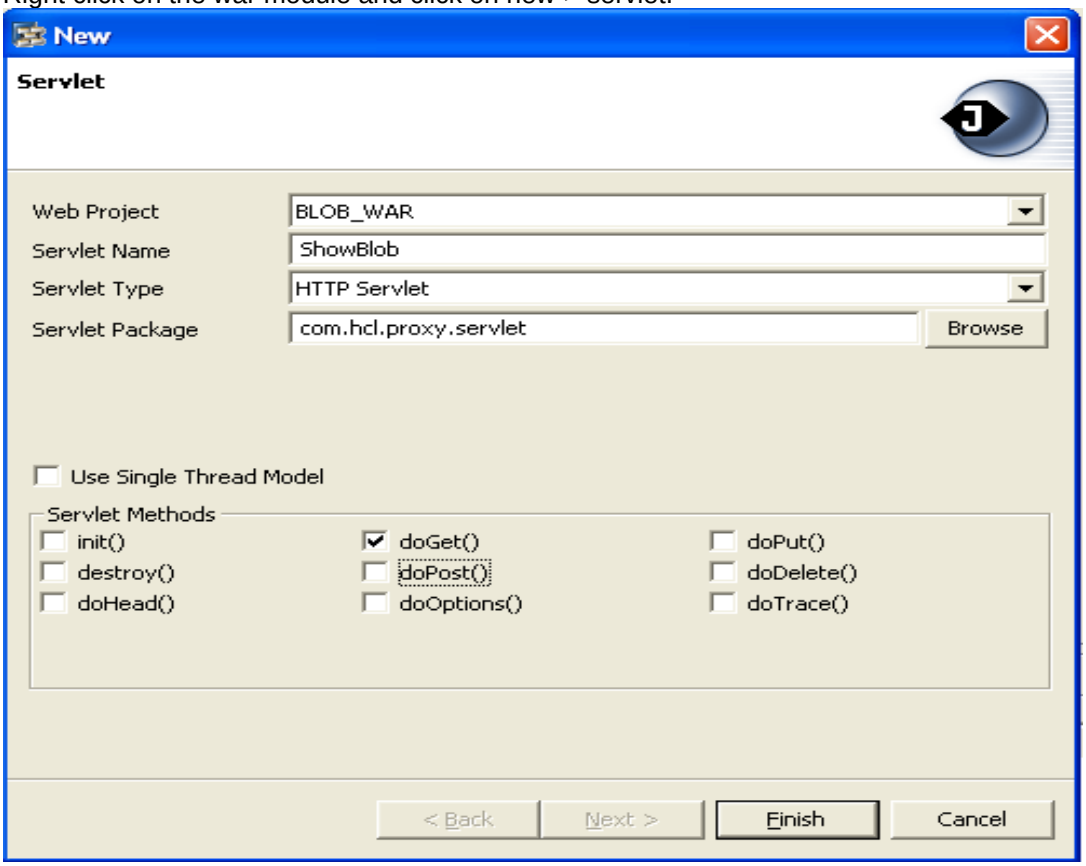
Web module that Calls Proxy

1. Click file-> new-> web module project. Type the web module name and click finish

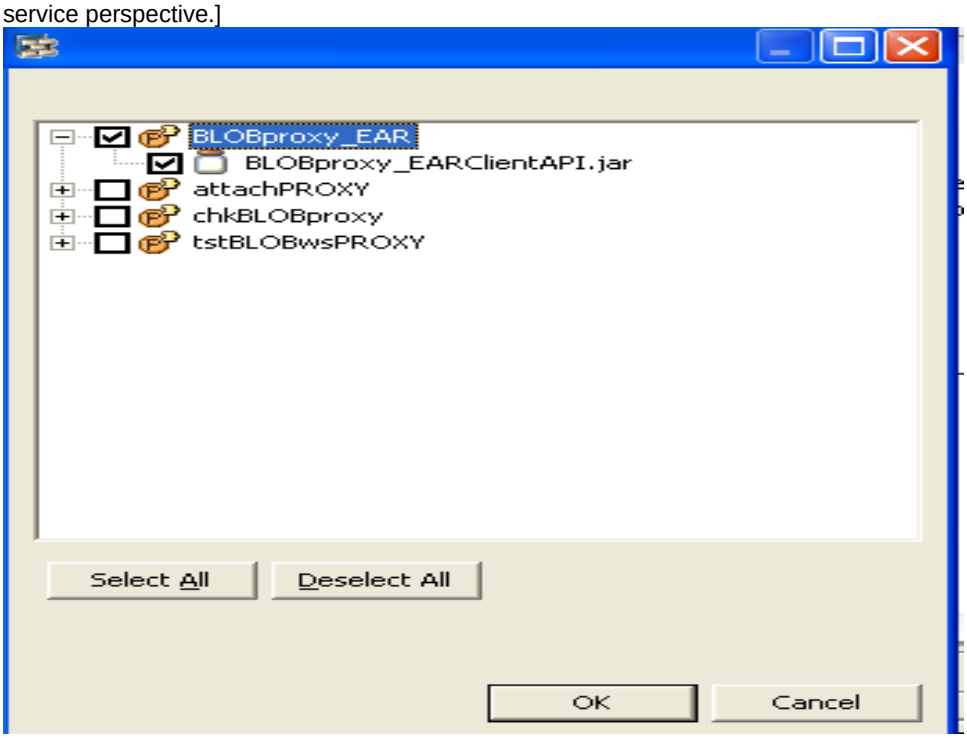




2. Right click on the war module and click on new-> servlet.



3. Type the servlet name, package name and click finish.
4. Right click on the war module and click Add Web Service Client API Library.
[Note: This option will not be enabling until the jar file of client deployable proxy is showing in the web service perspective.]

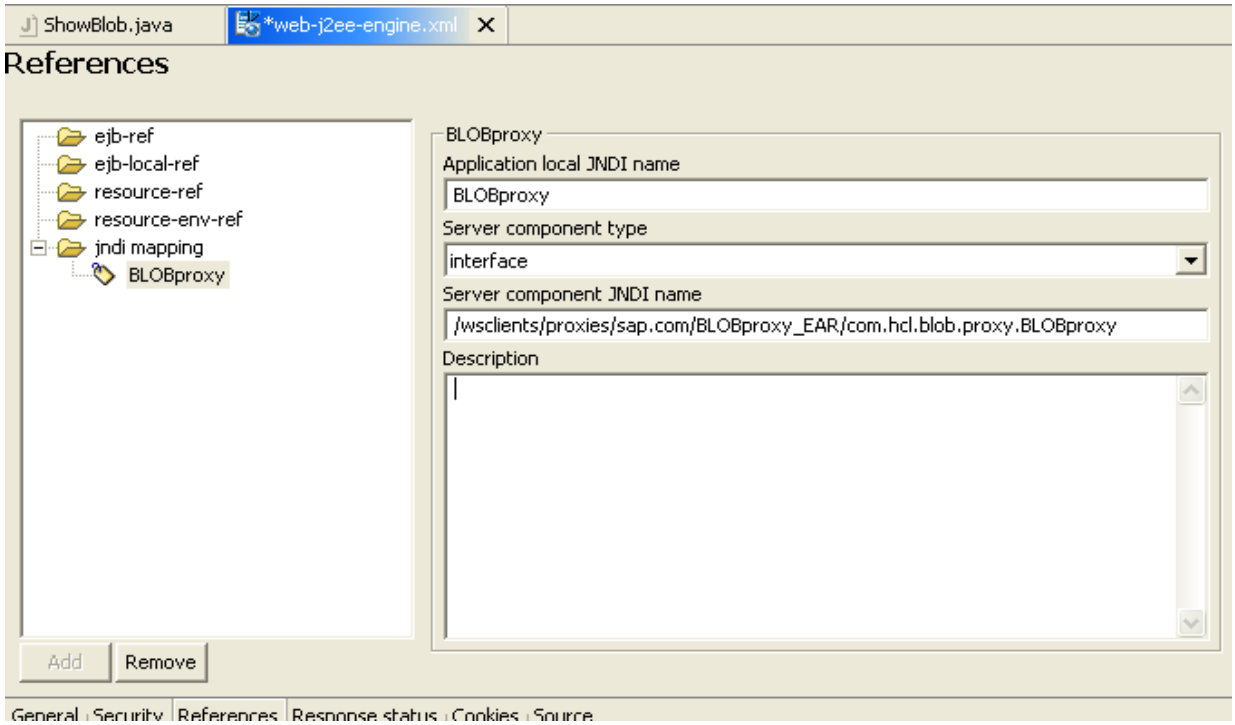


5. Select the client deployable proxy and click ok.

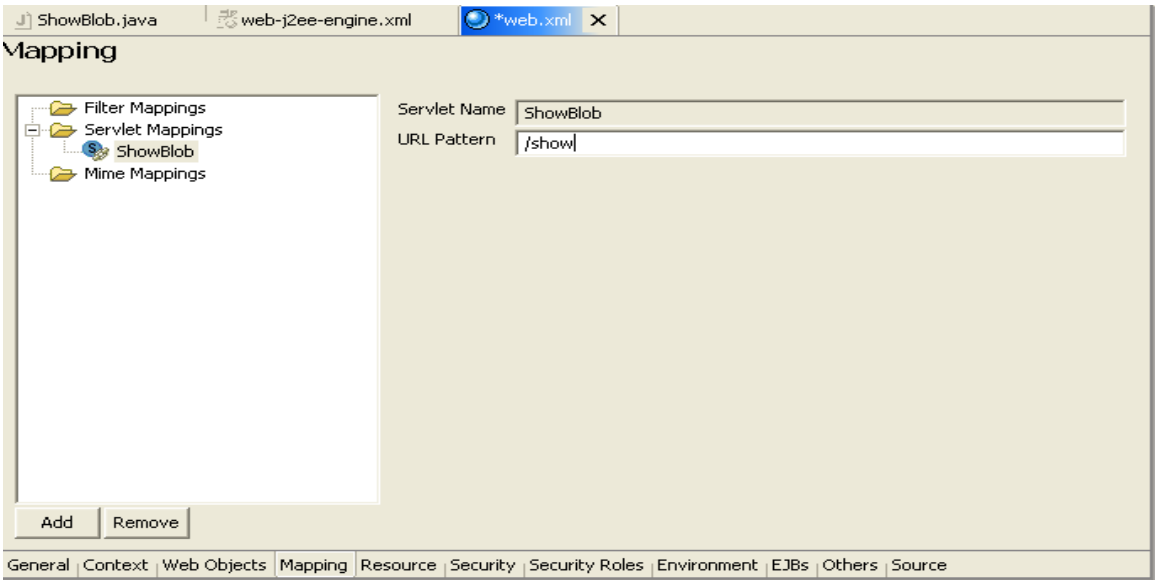
6. Open the servlet file and type the following code to call the web service method.

```
        PrintWriter pw = null;
   InitialContext ic = new InitialContext();
   BLOB_WS blbws = (BLOB_WS) ic.lookup
                                        ("java:comp/env/BLOBproxy");
   BLOB_WSViDocument vi = blbws.getLogicalPort ();
response.reset();
   response.setContentType("image/jpeg");
   response.getOutputStream ().write (vi.getBLOBData ());
response.flushBuffer ();
```

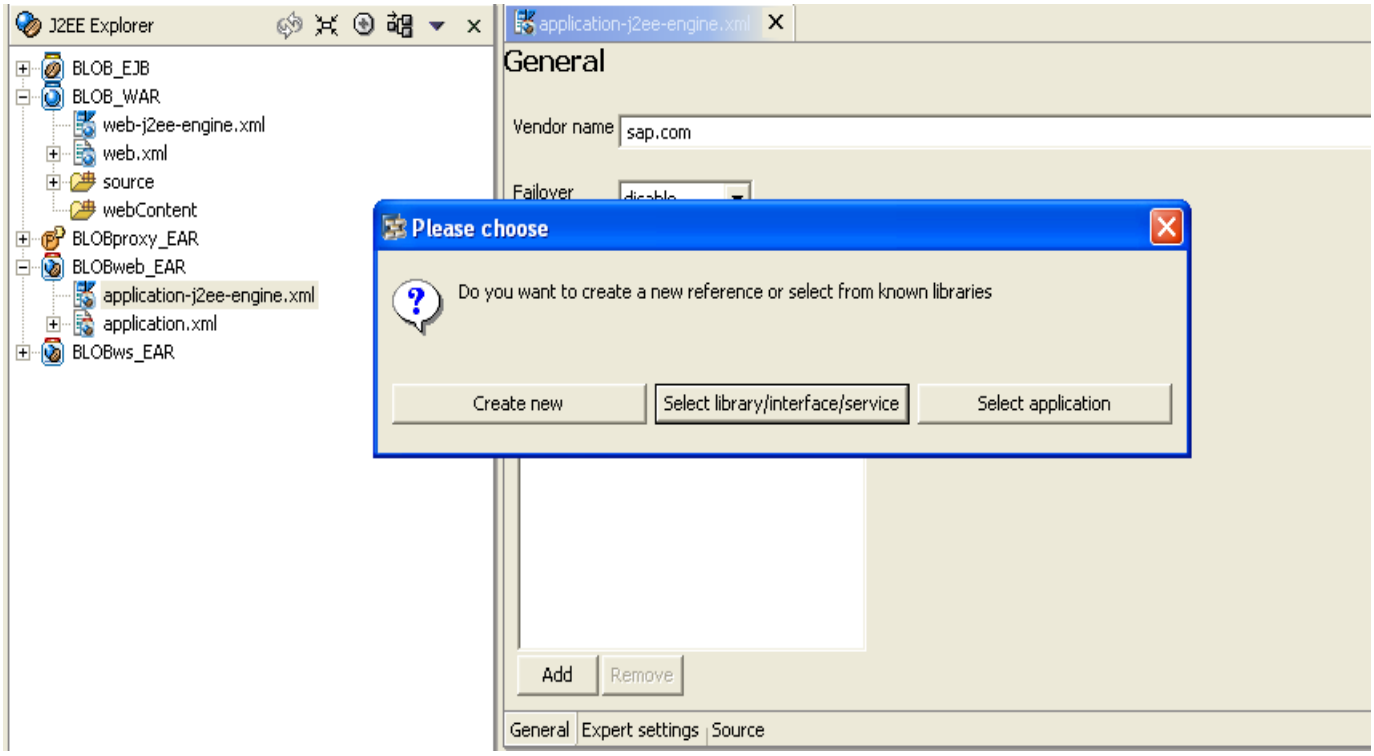
7. Add JNDI mapping.
   - Open web-j2ee-engine.xml file and choose references. Select JNDI mapping and click add.
   - Mention the application JNDI name (BLOBproxy) and server component JNDI name (/wsclients/proxies/sap.com/BLOBproxy_EAR/com.hcl.blob.proxy.BLOBproxy)



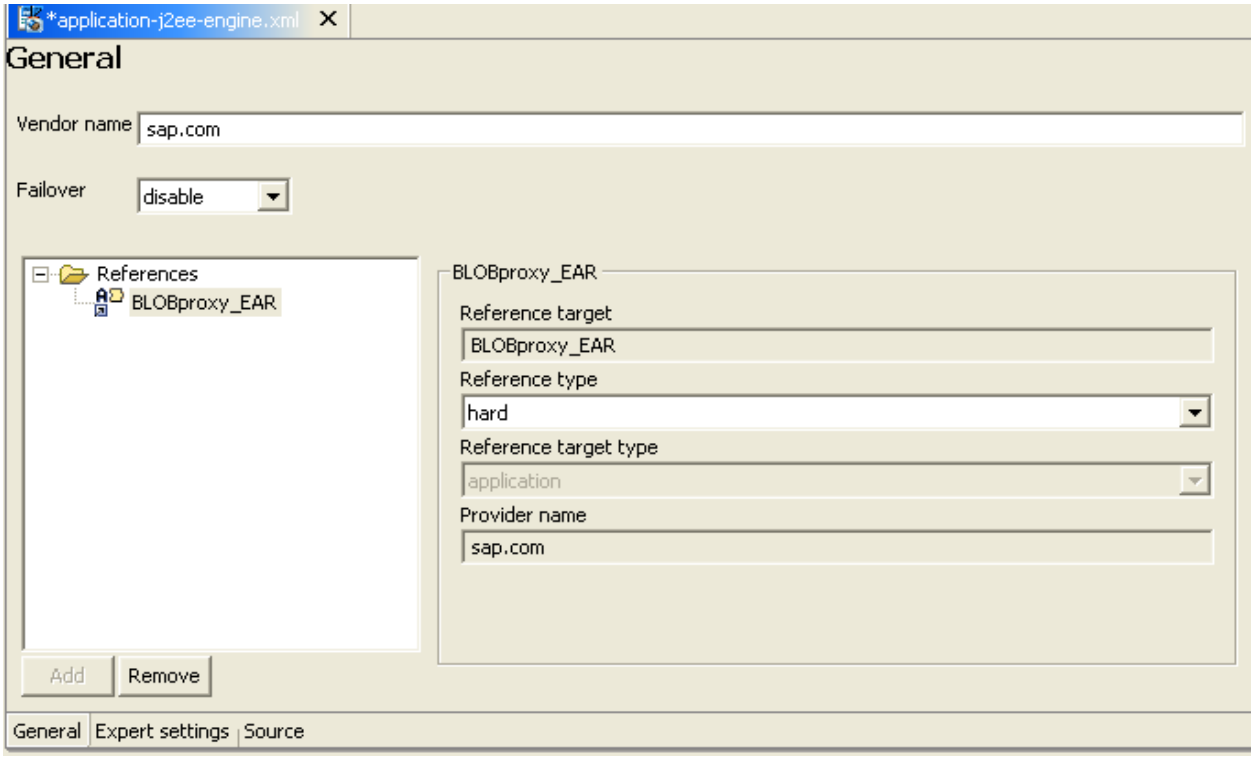
8. Add the servlet to the URL pattern.
    - Open web.xml, choose mapping-> select servlet mapping -> click on add -> check the servlet name and click on ok.
    - Type the URL Pattern.



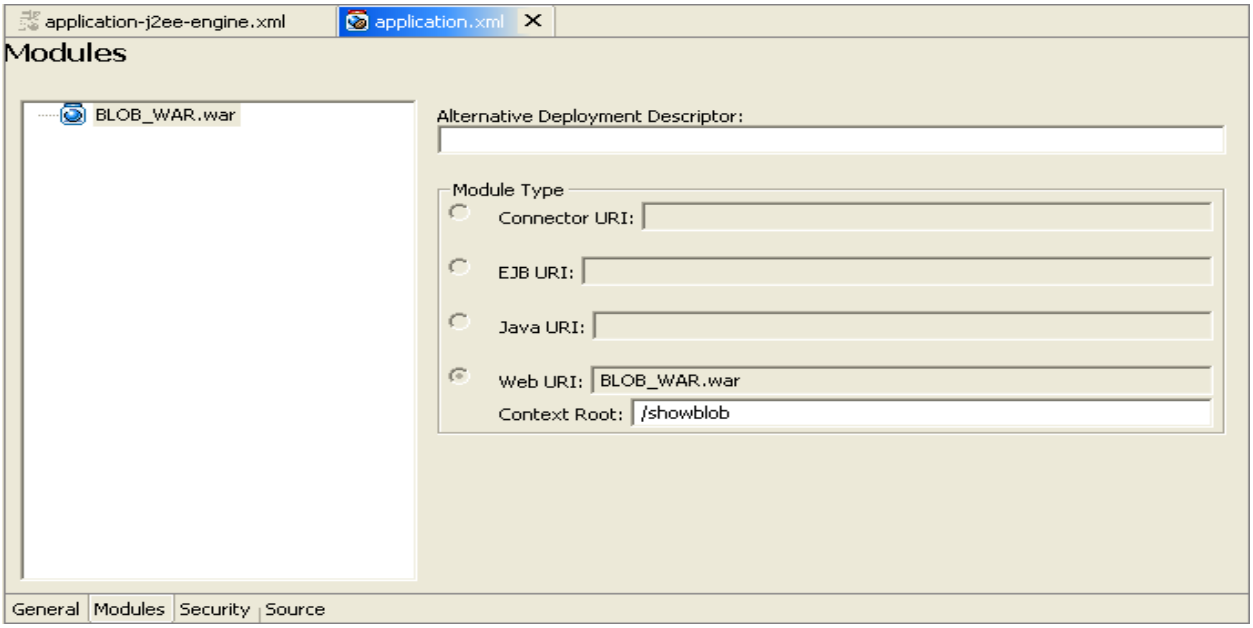
9. Create an EAR module (BLOBweb_EAR) and add the WEB module (BLOB_WAR).
10. Create a reference to the deployable proxy application.
    - Open the application-j2ee-engine.xml.
    - Choose general->reference-> add



- Click on create new.
- Mention the Reference target (BLOBproxy_EAR), Reference target type (application) and
- Provider name (sap.com).


*application-j2ee-engine.xml
General
Vendor name
sap.com
Failover
disable
References
BLOBproxy_EAR
BLOBproxy_EAR
Reference target
BLOBproxy_EAR
Reference type
hard
Reference target type
application
Provider name
sap.com
Add
Remove
General
Expert settings
Source

11. Mention the context root in application.xml.
    - Open application.xml file
    - Choose modules-> select the war module -> type the context root-> save.



12. Build the war and ear module.
13. Deploy the ear module.
14. Run the application.

    http://<host>:<port>/ showblob/show

Related Content

- http://help.sap.com/saphelp_nw04s/helpdata/en/d6/f9bc3d52f39d33e10000000a11405a/frameset.htm

Disclaimer and Liability Notice

This document may discuss sample coding or other information that does not include SAP official interfaces and therefore is not supported by SAP. Changes made based on this information are not supported and can be overwritten during an upgrade.

SAP will not be held liable for any damages caused by using or misusing the information, code or methods suggested in this document, and anyone using these methods does so at his/her own risk.

SAP offers no guarantees and assumes no responsibility or liability of any type with respect to the content of this technical article or code sample, including any liability resulting from incompatibility between the content within this document and the materials and services offered by SAP. You agree that you will not hold, or seek to hold, SAP responsible or liable with respect to the content of this document.