How-to Guide: The Migration Plug-in for SAP NetWeaver Composition Environment 7.1

Applies to:

SAP NetWeaver Composition Environment 7.1

Summary

The focus of this document is on the migration of J2EE-compliant applications to SAP NetWeaver Composition Environment (CE) from a technical point of view.

This document introduces the purpose and functionality of the migration plug-in. A sample application is used to exemplify the use of the tool as well as the basic steps to migrate an application to SAP NetWeaver CE.

Author: Markus Küfer

Company: SAP AG

Created on: June 2007

Author Bio

Markus Küfer holds a degree in medical informatics from the University of Heidelberg, Germany.

He joined SAP in 2000 and worked on Java application and technology frameworks as well as on customer specific J2EE development.

With the advent of JDO, Markus joined the initial core team that created the SAP JDO implementation and served as SAP's representative in the JDO Java Specification Request expert group (JSR 243). After some SAP J2EE Engine kernel development and design of the SAP EJB 3.0 implementation, he now imparts knowledge as a Solution Architect in the Global Ecosystem and Partner Group, Market Development Engineering team.

Table of Contents

Purpose

The Migration Plug-in is an Eclipse plug-in for the Eclipse based SAP NetWeaver Developer Studio for CE.

The plug-in allows importing third-party projects into the NetWeaver Developer Studio for CE. During the import all resources are copied into the right project structure and vendor specific deployment descriptors are converted to the corresponding SAP descriptors using XSLT transformations.

The primary focus is on the transformation of the EJB project's proprietary object-relational mapping descriptor which is one of the trickiest parts when it comes to migrating the descriptors.

The supported J2EE 1.3 and J2EE 1.4 compliant source platforms are:

- JBoss 3.x - 4.0.5
- BEA WebLogic Server 7.0 – 9.2

The generated CE Developer Studio projects are J2EE 1.4 compliant, although CE supports Java EE 5 as well.

There are several reasons to do so:

- The use of descriptors is not encouraged any more, as they are succeeded by annotations
- EJB 3 provides a standardised object-relational mapping, it is not proprietary any more as it was up to EJB 2.1.
- Not only the mapping, but also (the EJB) programming paradigm has changed dramatically. This can hardly be addressed tool-wise.

For example, some object-relational mappings are not allowed any more and can therefore not be applied to EJB 3, entities are just plain old java classes, no need for business and home interfaces any more, no more home interfaces for session beans, dependency injection simplifies coding a lot, etc.

How does the Tool work?

This will give you a more detailed understanding of how the tool works.

The subsequent sections will describe the tool's use in an exemplifying migration sample.

Transformed Descriptors

The plug-in's built-in XSLT transformation addresses the web project and ejb project descriptors. For J2EE 1.3 and 1.4 compliant projects, the proprietary descriptors like "jboss-web.xml", "weblogic.xml" for web projects or "jboss.xml", "weblogic-ejb-jar.xml" and "jbosscmp-jdbc.xml", "weblogic-cmp-rdbms-jar.xml" for ejb projects are considered.

A J2EE 1.3 compliant project additionally requires the Migration Plug-In to transform the standard descriptors "web.xml" and "ejb-jar.xml" into J2EE 1.4 compliant descriptors as well.

Both the proprietary and the standard descriptors had undergone a change in the way they are described, from doctype to schema-based descriptors. The tool handles that implicitly as well.

For your convenience, the original descriptors are copied as a reference. To avoid name collisions, the suffix ".orig" will be added to the original "web.xml" and the original "ejb-jar.xml". You can safely delete them.

The migration of web services implementations or web service related content in the descriptors described above is also out of scope of the tool. As the CE NetWeaver Developer Studio supports the latest web service standards now part of Java EE 5, it is reasonable to migrate in a "fall-forward fashion" and to make use of the standardised web service support.

Created SAP NetWeaver CE Developer Studio Projects

The Migration Plug-in creates J2EE 1.4 compliant EJB and Web projects.

For web projects, the specification of the following is required:

- Web Content Directory
- Java Source Directory
- Descriptor Directory

EJB projects require the following to be specified:

- Source Directory
- Descriptor Directory

If you realise the way the plug-in works does not suit the way the projects to be migrated are structured, you can still make use of the tool's prime feature, the (object-relational) mapping descriptors' transformation. Just do it for each of the projects, to get the descriptors, merge them and create the final SAP NetWeaver CE Developer Studio projects according to your specific needs.

Using the Migration-Plug-in: A Sample Migration

The J2EE 1.4 compliant sample application to be migrated was created by means of XDoclet for JBoss as target runtime.

It is a simple, yet complete application with regards to the object-relational mapping used.

The application deals with Employees, Departments, Projects, Products and Contact Data entity beans which are used in a so-called Services stateless session bean that serves as session-façade.

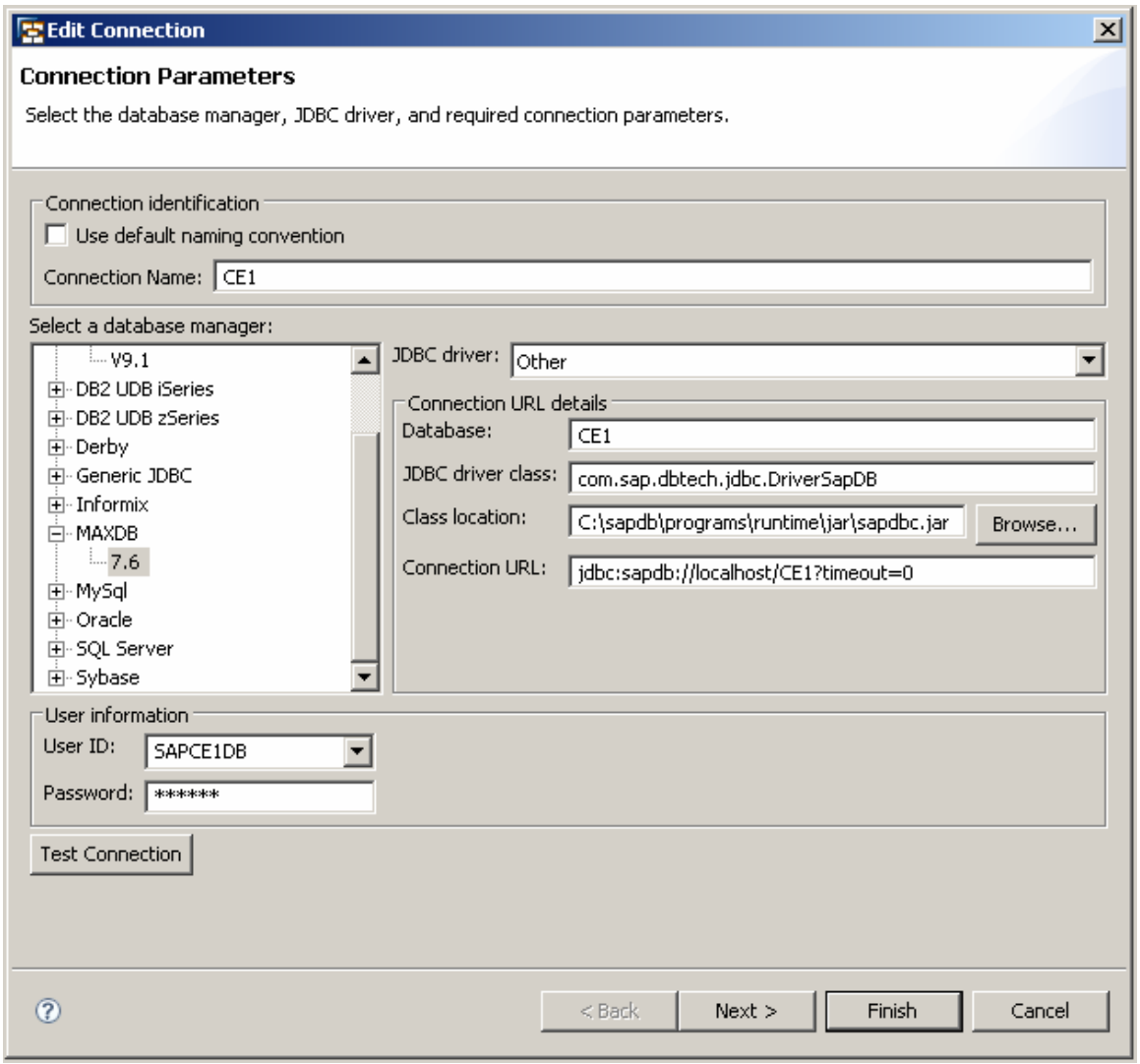
Preparation/Get the Database ready

For this sample, we create the database tables with in the Persistence Perspective of the NetWeaver Development Studio for CE.

Therefore, we need to establish a designtime connection to the database.

We will use the Java EE engine's schema.

Switch to the Persistence Perspective. On the "Database Explorer" view, right-click "Connectons", select "New Connection…", enter the parameters according to your installation and press "Finish":

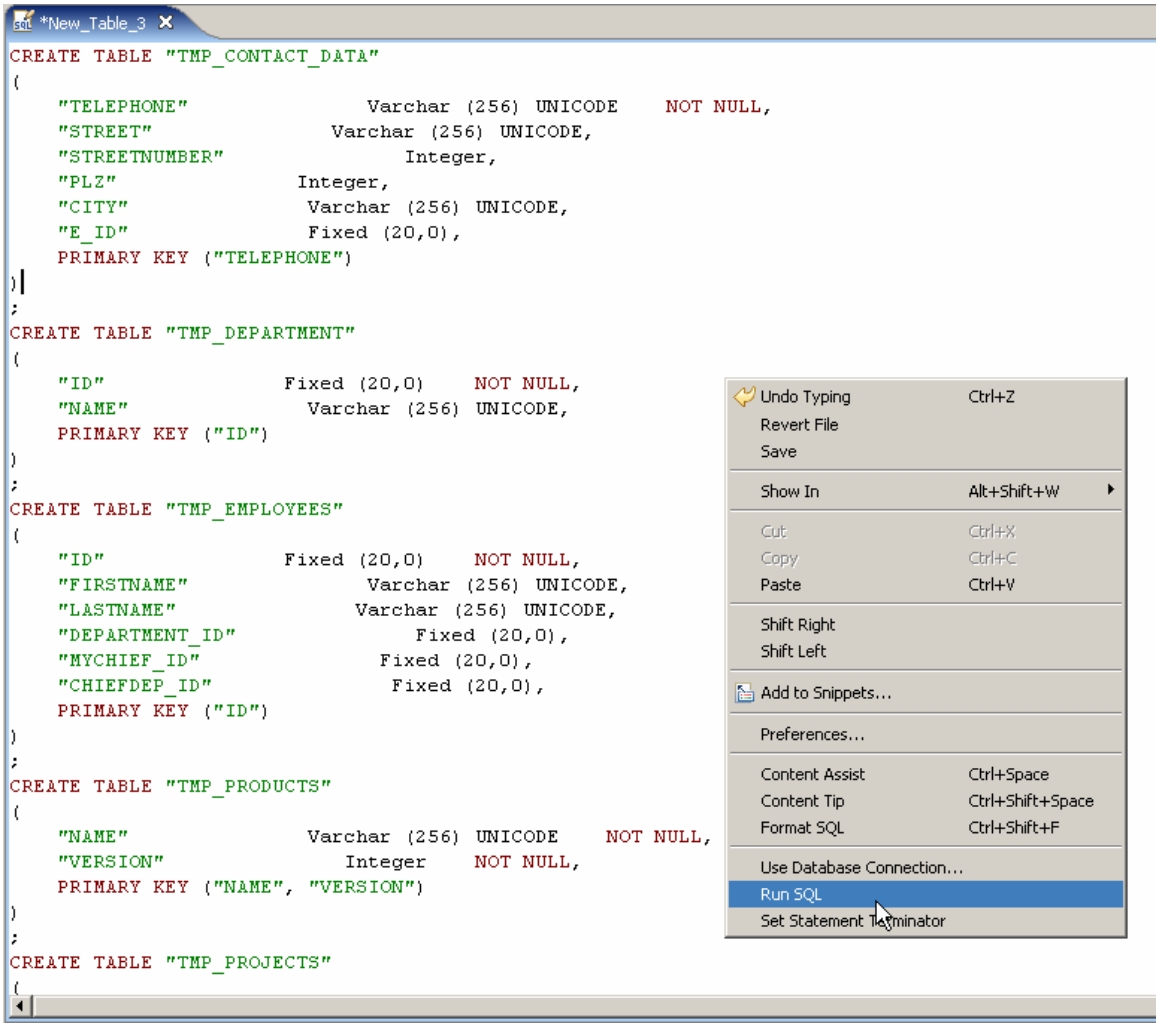


The table definitions come along with the sample code in the “JMTS” folder, file “tables.sql”.

To create the tables, navigate along “CE1→CE1→Schemas→SAPDBCE1→Tables”, right-click and select “New”→”With SQL Editor”.

A new editor tab will open. Delete its content, copy and paste the content of “tables.sql” into the editor.

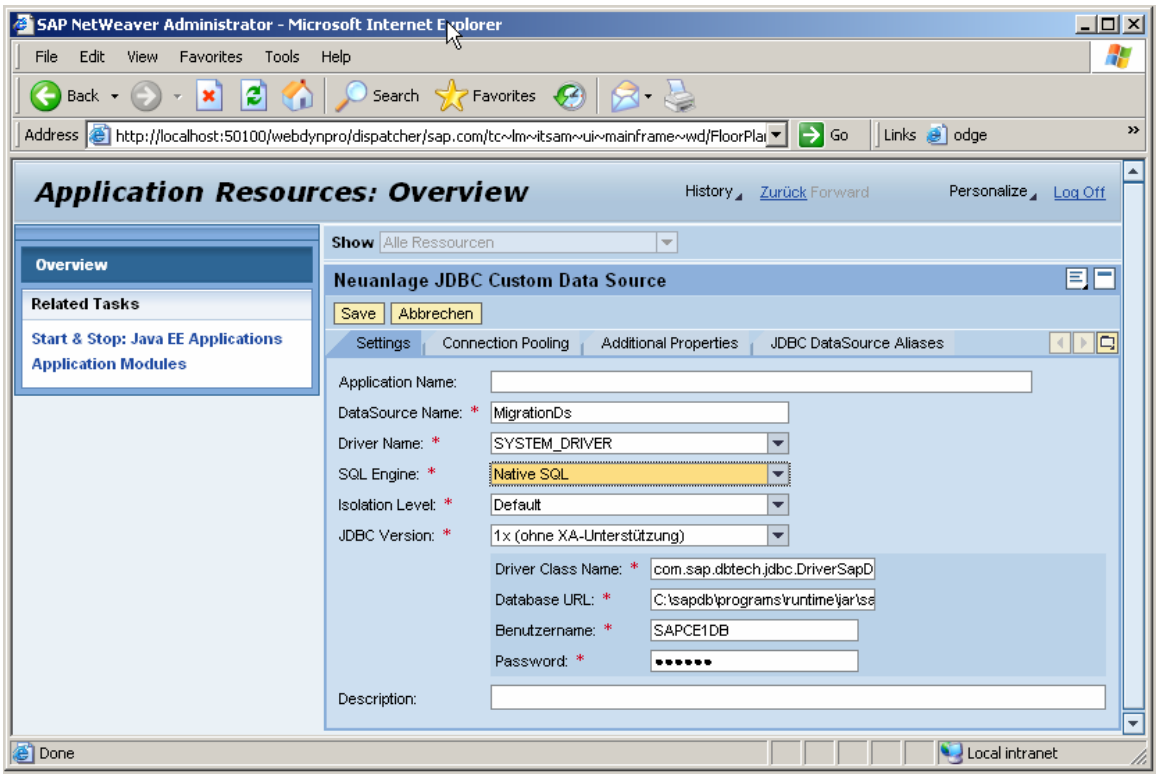
Right-click the editor’s canvas and select “Run SQL”. This will create all the necessary tables:



To access the tables at runtime, we need to create a datasource named “MigrationDS”:

Open the “NetWeaver Administrator”: http://localhost:50100/nwa in your browser.

Click “Systems→Start Stop → Java EE Applications → Application Resources” and select “Create New JDBC Custom Datasource” and enter the values as you did for the Developer Studio:



Make sure you select “Native SQL” or “Vendor SQL”.

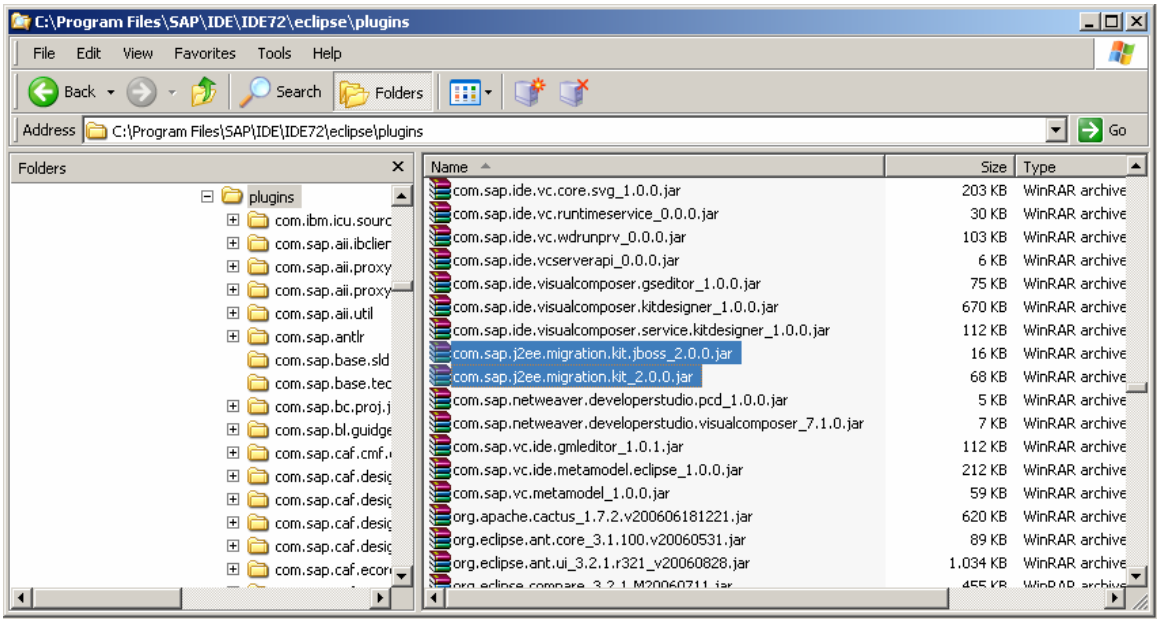
Press “Save” to create the datasource finally.[1]

[1] You may want the MigrationDS to be started automatically on server start-up. Therefore, switch to “Start & Stop Java EE Applicatins“, filter the list by “MigrationDS“, select it and press “Start“ → “On All Instances And Set Started“ As Initial State“.

Installing the Migration Plug-in

Unpack the jar files contained in the zip file for the Migration Plug-in you downloaded.

Straightforwardly copy those plug-in jar files into the eclipse plug-in folder[2]:

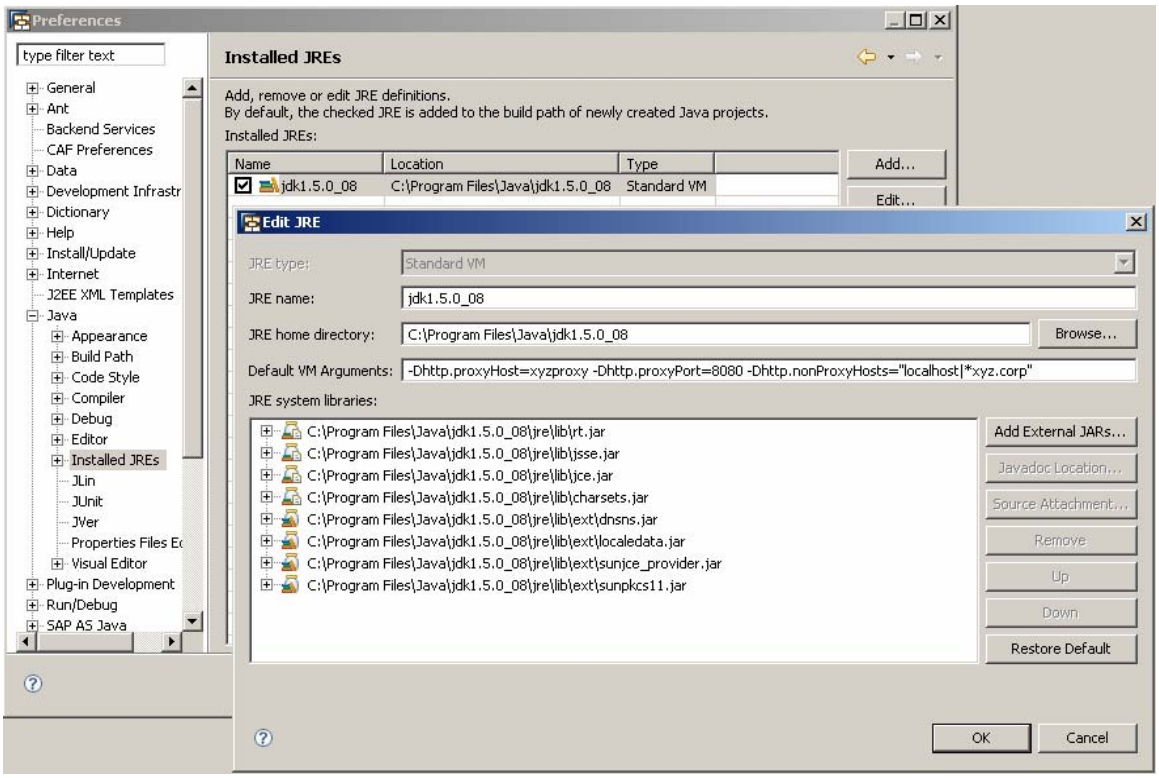


Now start the NetWeaver Developer Studio for CE. If you started it before you copied the jars, please restart it.

If you connect to the internet through a firewall, please do the proxy settings accordingly. It is the most reliable way is to do this on the IDE VM level[3]:

Open “Window”→”Preferences…”→”Java”→”Installed JREs”, select your JRE, click “Edit…” and enter your proxy settings accordingly:

[2] Copying the jar for BEA is fine as well. If you migrate a BEA project, you have to copy it.
[3] The Internet Proxy settings on the Preferences page did not work for me.

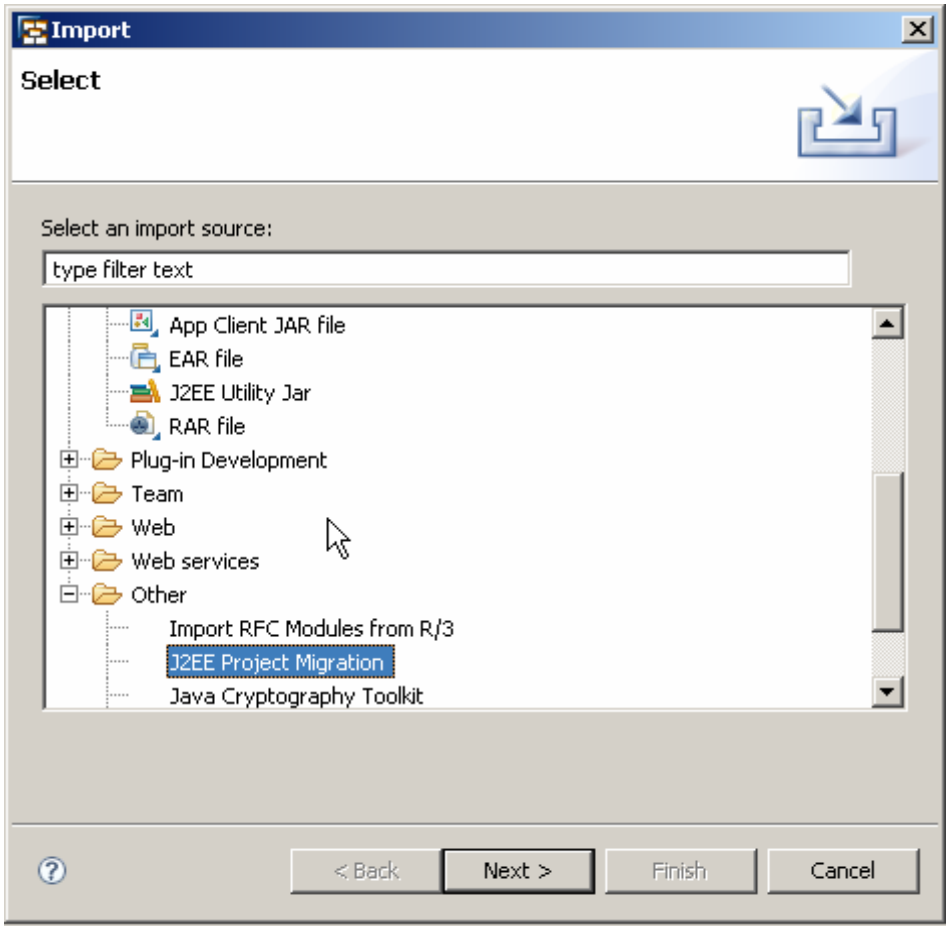

Preferences
type filter text
General
Ant
Backend Services
CAF Preferences
Data
Development Infrastr
Dictionary
Help
Install/Update
Internet
J2EE XML Templates
Java
Appearance
Build Path
Code Style
Compiler
Debug
Editor
Installed JREs
JLin
JUnit
JWer
Properties Files Ec
Visual Editor
Plug-in Development
Run/Debug
SAP AS Java
Installed JREs
Add, remove or edit JRE definitions.
By default, the checked JRE is added to the build path of newly created Java projects.
Installed JREs:
Name
Location
Type
jdk1.5.0_08
C:\Program Files\Java\jdk1.5.0_08
Standard VM
Add...
Edit...
Edit JRE
JRE type:
Standard VM
JRE name:
jdk1.5.0_08
JRE home directory:
C:\Program Files\Java\jdk1.5.0_08
Browse...
Default VM Arguments:
-Dhttp.proxyHost=xyzproxy -Dhttp.proxyPort=8080 -Dhttp.nonProxyHosts="localhost|*xyz.corp"
JRE system libraries:
C:\Program Files\Java\jdk1.5.0_08\jre\lib\rt.jar
C:\Program Files\Java\jdk1.5.0_08\jre\lib\jsse.jar
C:\Program Files\Java\jdk1.5.0_08\jre\lib\jce.jar
C:\Program Files\Java\jdk1.5.0_08\jre\lib\charsets.jar
C:\Program Files\Java\jdk1.5.0_08\jre\lib\ext\dnsns.jar
C:\Program Files\Java\jdk1.5.0_08\jre\lib\ext\localedata.jar
C:\Program Files\Java\jdk1.5.0_08\jre\lib\ext\sunjce_provider.jar
C:\Program Files\Java\jdk1.5.0_08\jre\lib\ext\sunpkcs11.jar
Add External JARs...
Javadoc Location...
Source Attachment...
Remove
Up
Down
Restore Default
OK
Cancel

Create the IDE Projects and transform the Descriptors – Using the Migration Plug-in

Unpack the .zip file that contains the sample application.

It consists of an EJB part and a web part.

Open “File”→”Import…”→”Other”→”J2EE Project Migration”:



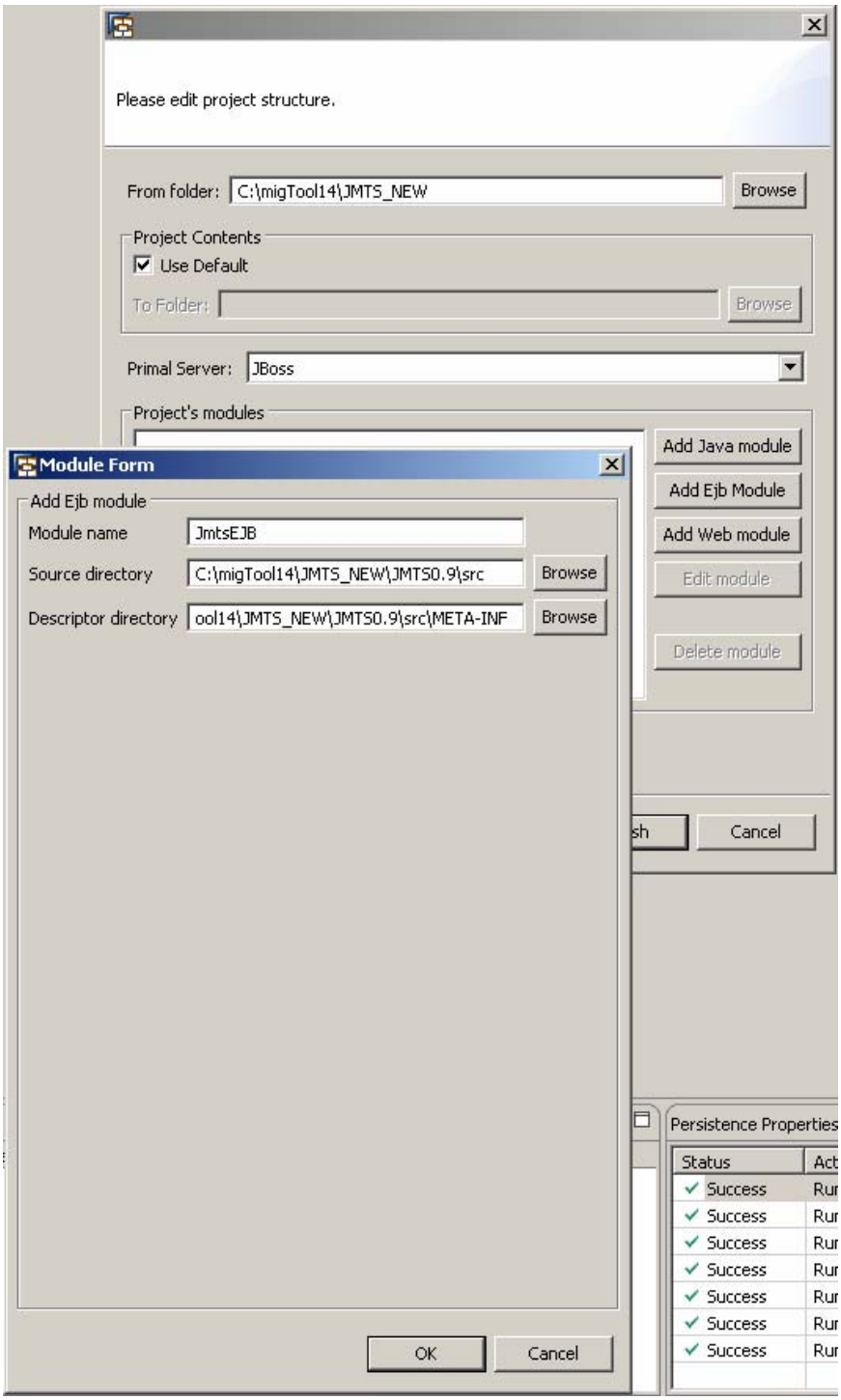
The window you will see next is supposed to collect all the information needed to migrate the sample application.

The “From folder” just specifies a convenient default folder to start from that is used when you add a Java, EJB or Web project.

Choose the folder you unpacked the sample application to.

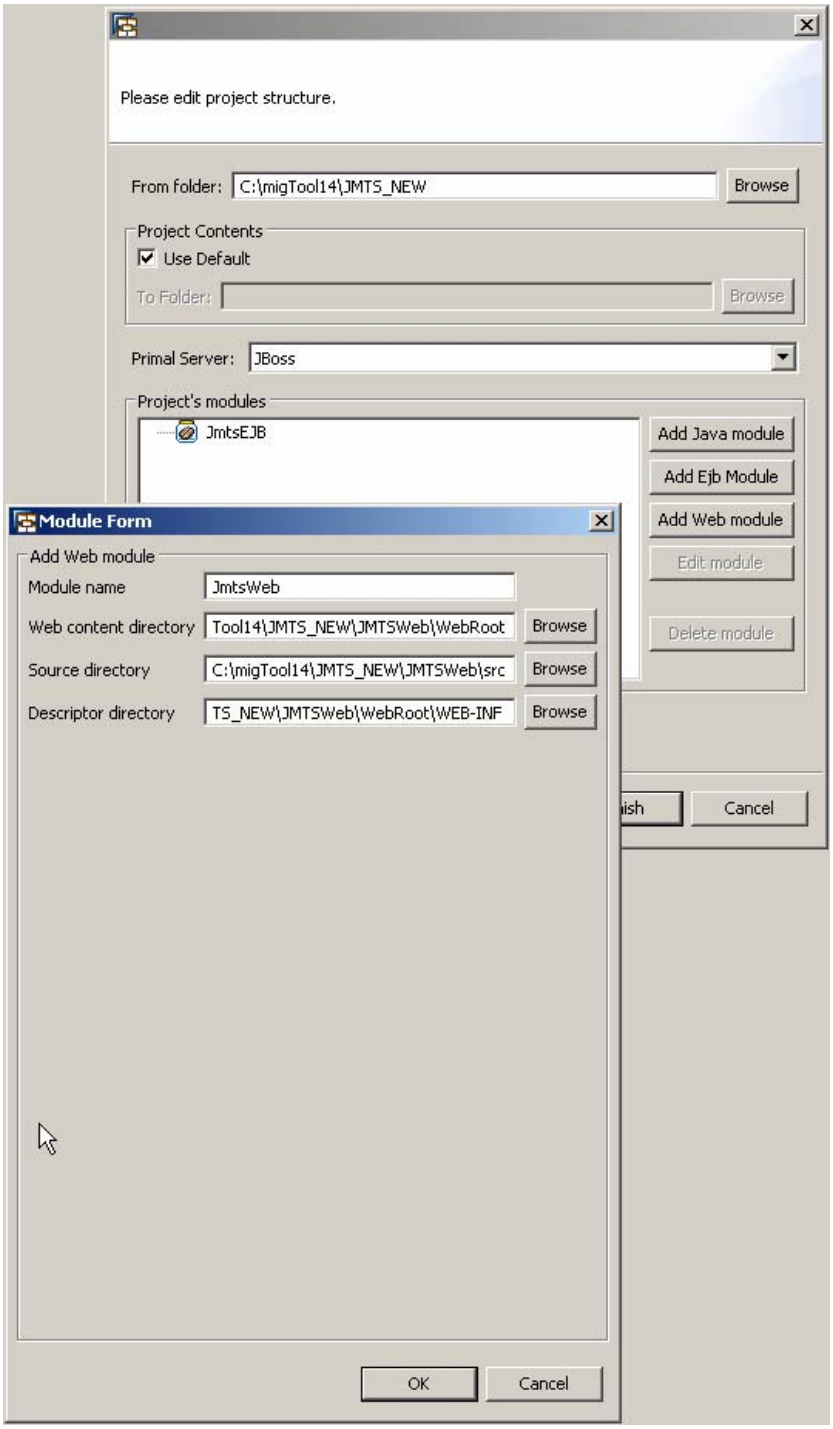
The “Primal Server” is JBoss in this case, as we migrate an app that formerly ran on JBoss.

We first specify where the EJB module’s sources and descriptors can be found. Select “Add Ejb Module”, specify how you want the generated Eclipse project to be named and browse the folders appropriately as show below:


Please edit project structure.
From folder: C:\migTool14\JMTS_NEW
Browse
Project Contents
Use Default
To Folder:
Browse
Primal Server: JBoss
Project's modules
Add Java module
Add Ejb Module
Add Web module
Edit module
Delete module
Cancel
Module Form
Add Ejb module
Module name JmtsEJB
Source directory C:\migTool14\JMTS_NEW\JMTS0.9\src
Browse
Descriptor directory ool14\JMTS_NEW\JMTS0.9\src\META-INF
Browse
OK
Cancel
Persistence Properties
Status
Success

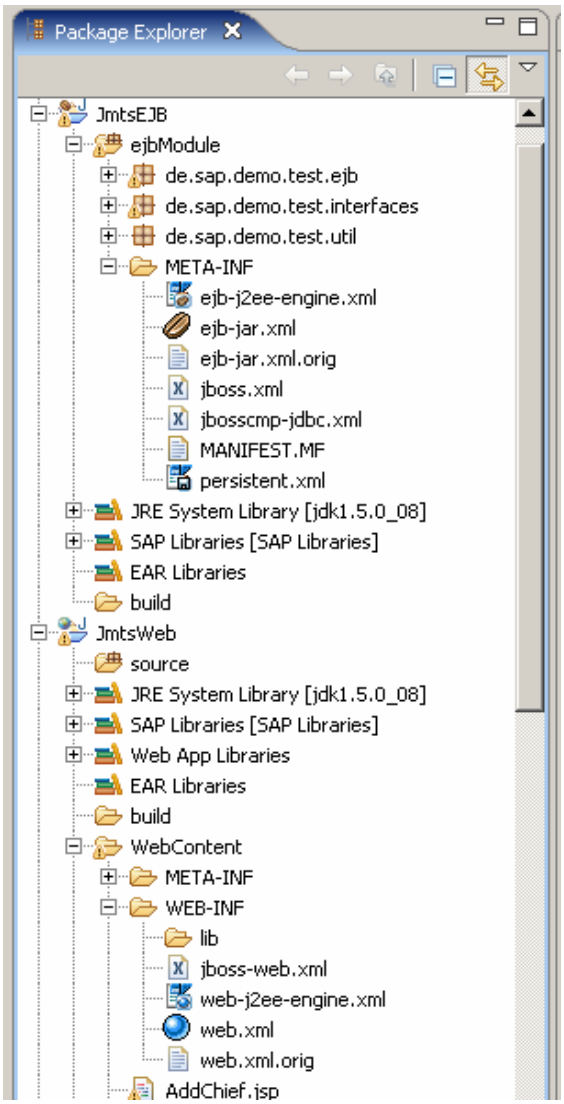
We will now collect all the information needed for the web module:

Select “Add Web Module” ”, specify how you want the generated Eclipse project to be named and browse the folders appropriately as show below:



Press “OK” to return and “Finish” to start the project creations and descriptor transformations.

Your workspace will look similar to this:



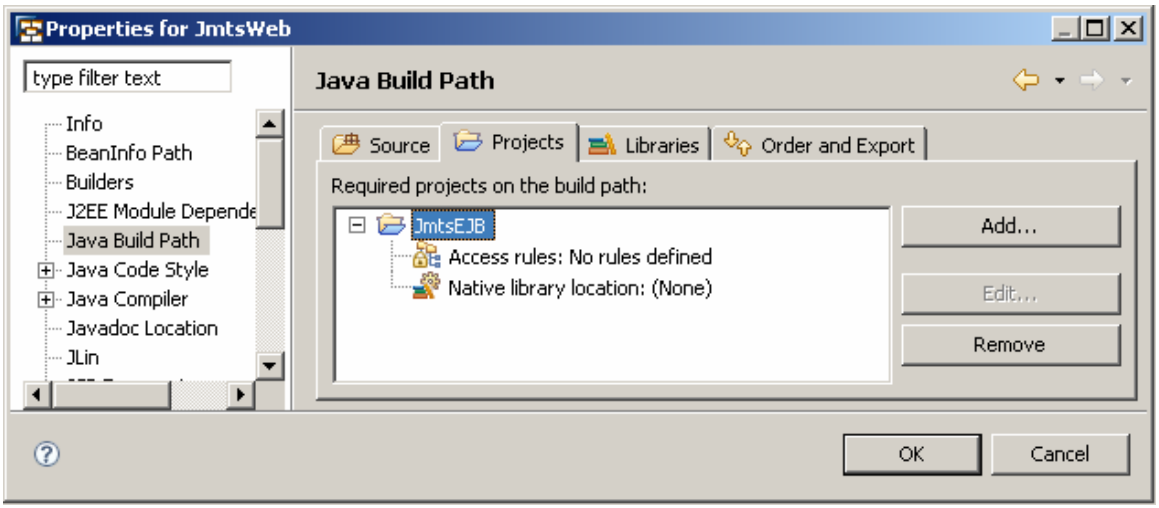
Project Structure Completion

We complete the project structure by the creation of an ear project that contains “JmtsEJB” and “JmtsWeb”:

Select “File” → “New” → “Project…”→ “J2EE” → “Enterprise Application Project”, enter a project name, “JmtsEAR”, for example, press “Next >” two times, select the appropriate web and EJB project (“JmtsWeb” and “JmtsEJB”) and press “Finish”.

To make the EJB classes known in the web project, a reference to the EJB project (“JmtsEJB”) has to be set in the web project (“JmtsWeb”):

Right-click the web project (“JmtsWeb”), select “Properties”, goto “Java Build Path” and switch to the tab “Projects”. Press “Add…” and select the appropriate EJB project (“JmtsEJB”) and press “OK” two times.



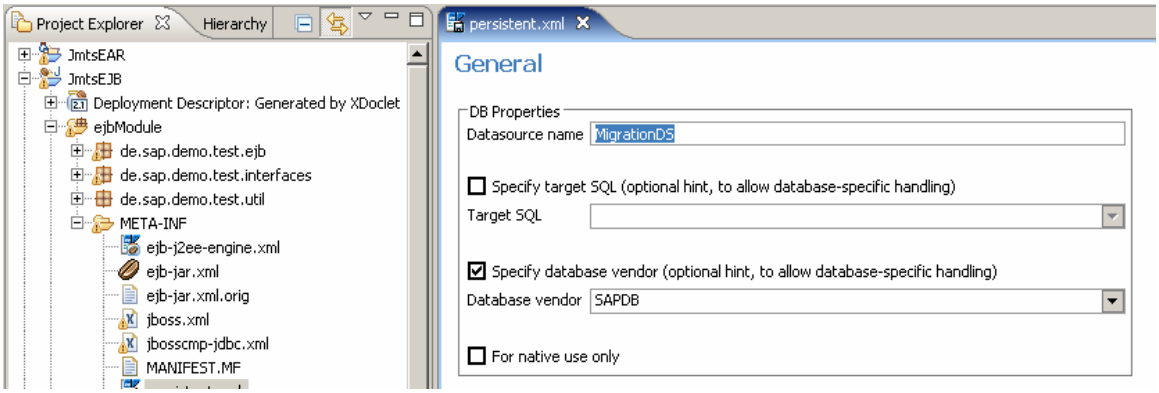
Rebuild your projects to resolve any classpath related issues.

Datasource Adjustments

The sample application is not yet migrated completely. The sample application should run on the datasource named “MigrationDS” we created at the very beginning.

Therefore, it needs to be introduced in the newly generated persistent.xml.

Open the “persistent.xml” (in the J2EE perspective), navigate to the “General” tab and enter “MigrationDS” as the Datasource name and save the “persistent.xml”:



One of the challenges that every migration encounters is the adaptation of the JNDI naming. The JBoss sample application is no exception to that.

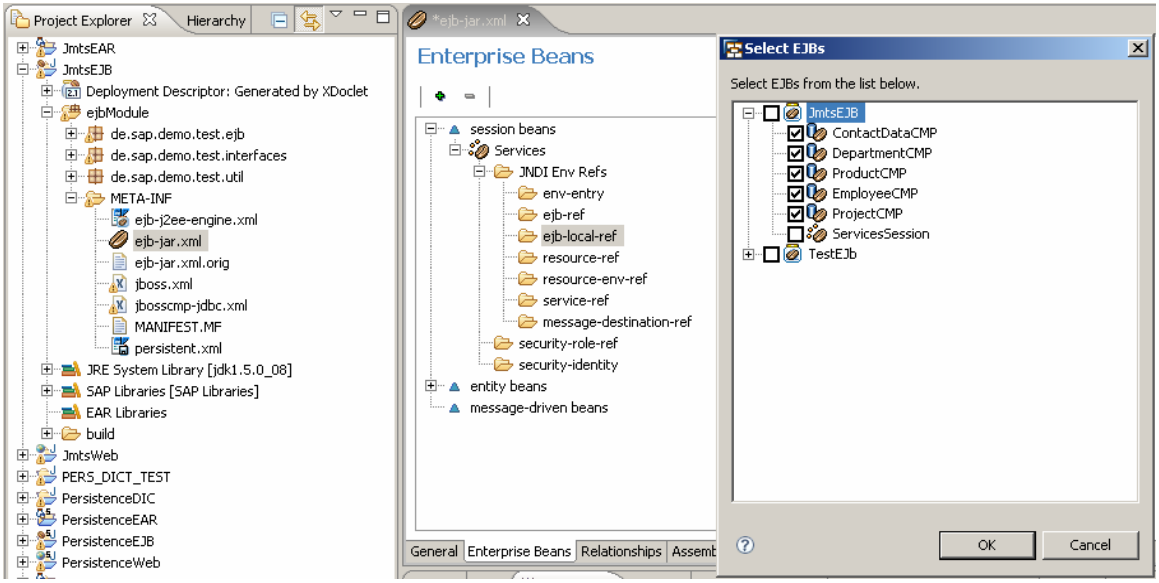
JNDI Adjustments

It is common practice and good style to use local JNDI references and lookup as it is encouraged by the Java EE specs to support portability.

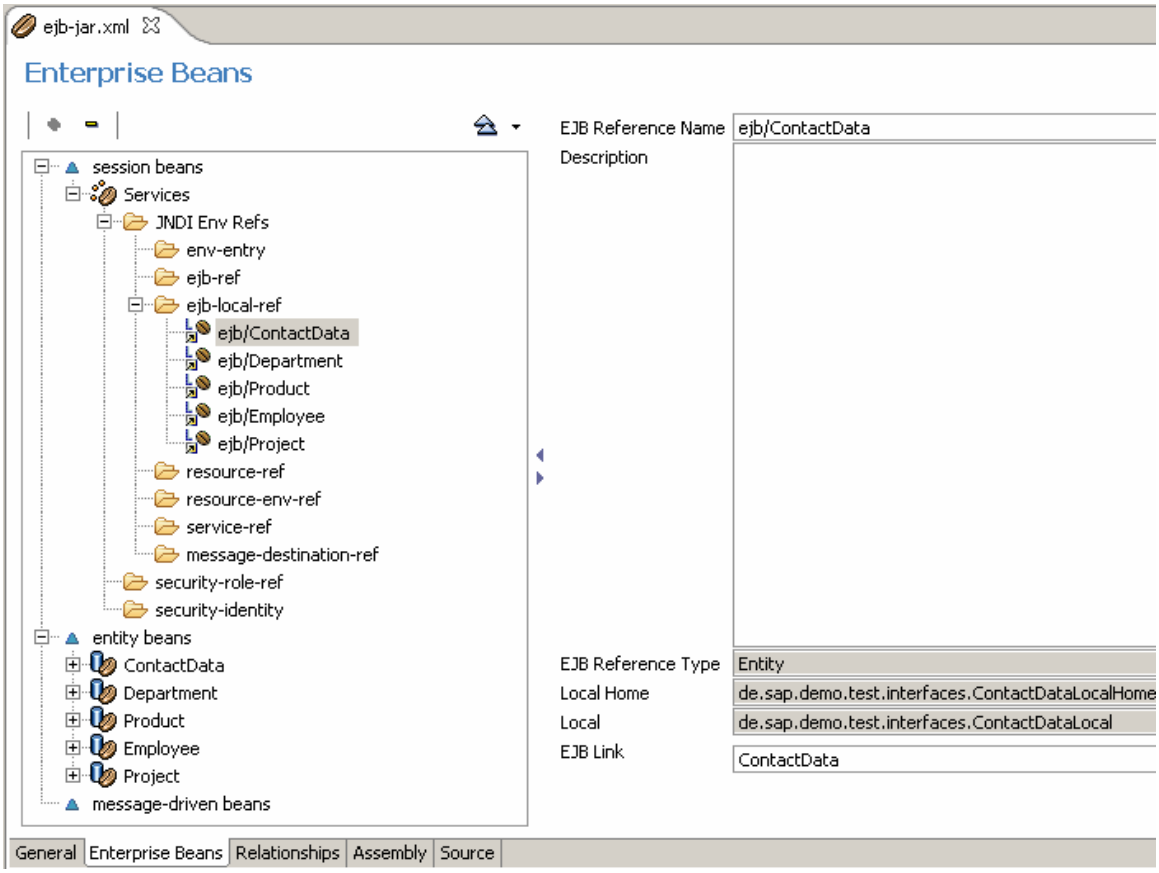
Open the “ejb-jar.xml” in the “META-INF” folder of the EJB project (“JmtsEJB”) and you will find that no local references are declared from the Services session bean to all its used entity beans. The JNDI lookups in the Services session beans are all done by using their respective global JNDI lookup name.

The next step is to use local JNDI references instead:

Switch to the “Enterprise Beans” tab, open “session beans”, then expand the “Services” session bean → “JNDI Env Refs” → “ejb local ref”, right-click and chose “Add”, select all entries beside the “ServicesSession” for your EJB project (“JmtsEJB”) and click “OK”:



Save the “ejb-jar.xml” file. Your references should look like this now:



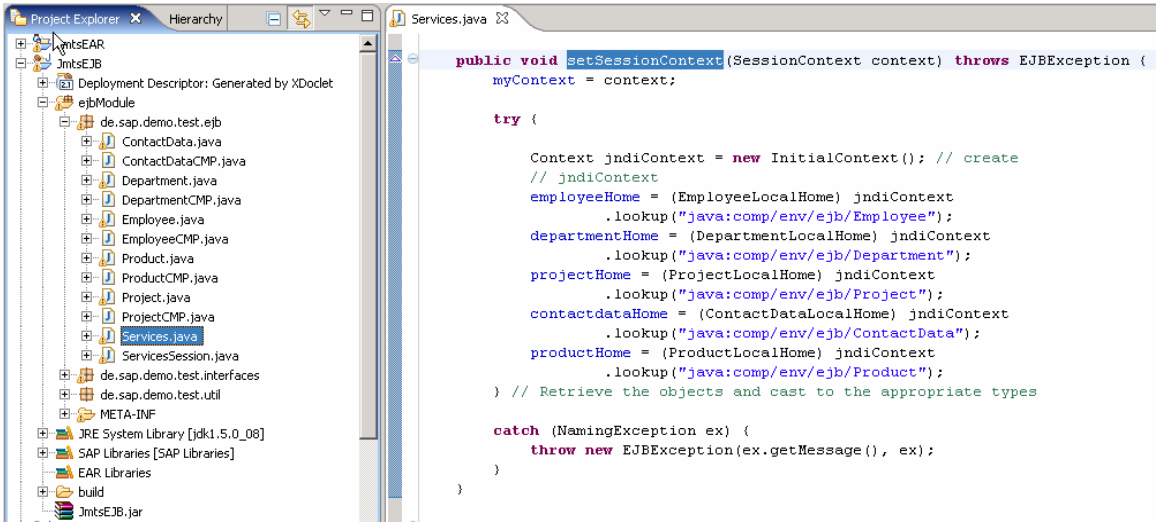
The next step is to check how the “Services” session bean looks up the entity beans it uses and to use the local JNDI references we just declared.

Open the “Services.java” file which resides in your EJB project (“JmtsEJB”), package “de.sap.demo.test.ejb”, and navigate to the “setSessionContext” method.

The creation of the initial context does not need any property parameters because the lookup happens inside the server and will reference objects that reside on the same server, too. Therefore, delete the “properties” declaration and remove its use as parameter to the “InitialContext” creation.

Change each lookup string to the reference declared in “ejb-jar.xml”. For example, change "/ejb/EmployeeLocalHome" to “java:comp/env/ejb/Employee”.

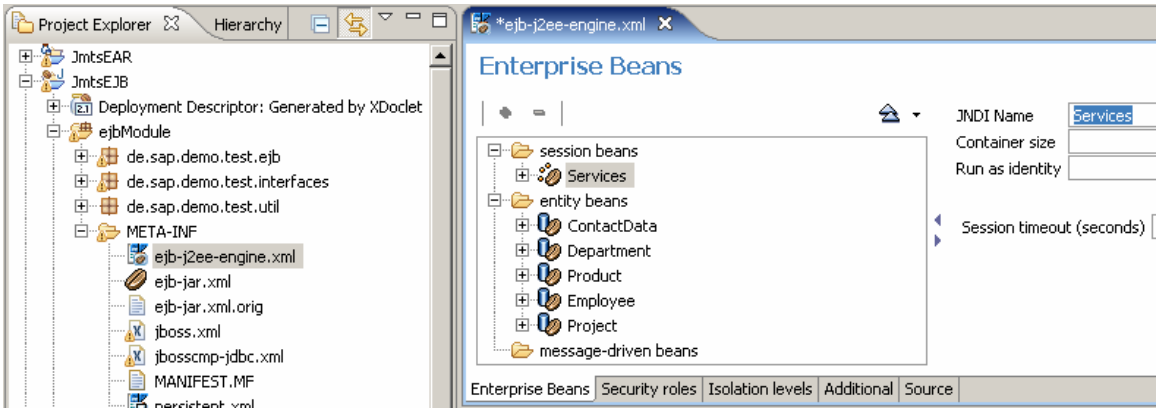
Proceed with all the other lookups in the “setSessionContext” method in the same manner and save “Services.java”:



Now have a look at the “ejb-j2ee-engine.xml” (and the original “jboss.xml”, too. It was copied to the newly created CE EJB project as well.). As it is common practice with JBoss, the EJBs’ JNDI names start with “ejb/”.

With NetWeaver CE, the JNDI entry “ejb” is already used, as there is a reserved top level naming entry for each of the Java EE engine’s services. Therefore, this leading “ejb/” prefix has to be removed from the JNDI name of all session and entity beans. This will cause the beans to be registered underneath the “ejbContexts” folder in the JNDI tree.

Open the “ejb-j2ee-engine.xml” in the “META-INF” folder of the EJB project (“JmtsEJB”). In the “Entity Beans” tab, open both “session beans” and “entity beans” and correct *each bean’s* JNDI name by removing “ejb/”:

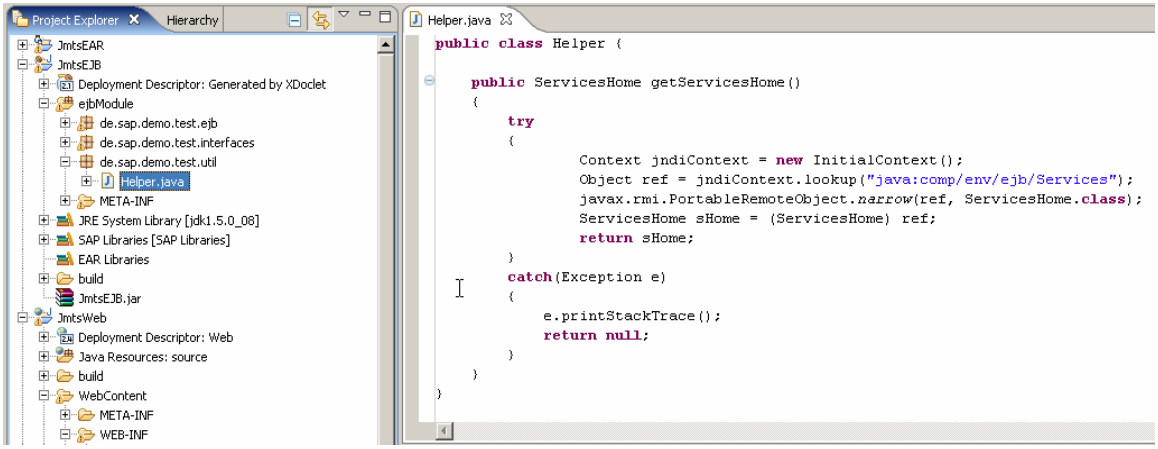


Do not forget to save the “ejb-j2ee-engine.xml” file.

Note that we do not make use of the JNDI names declared in “ejb-j2ee-engine.xml” within the sample application, but they could be used for lookups from non-J2EE applications, for example.[4]

Now we have to check the actual JNDI lookups for the “Services” session bean in the EJB project’s coding:

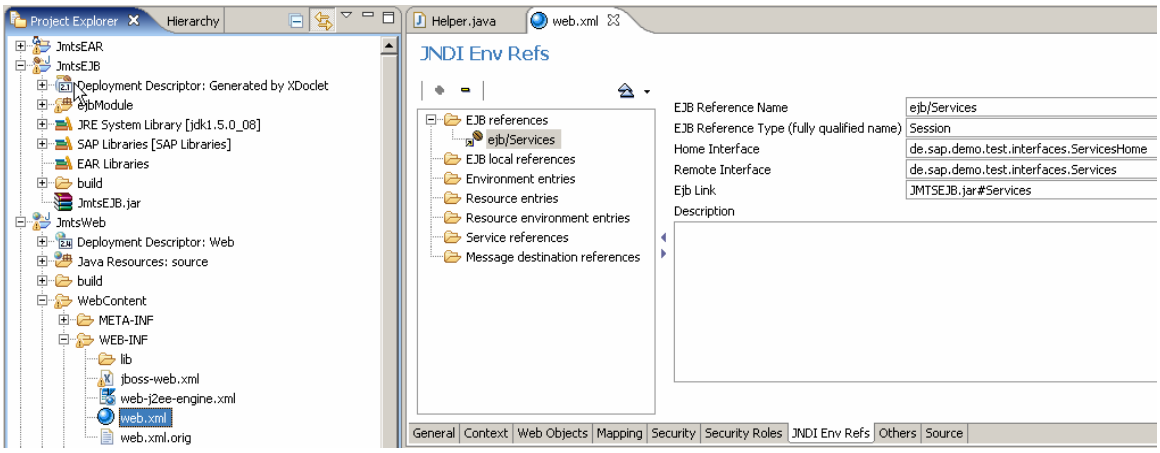
There is a “Helper” class in the package “de.sap.demo.test.util” that implements the JNDI lookup of the “Services” session bean. It is used by the web project (“JmtsWeb”) only:[5]



The “Services” session bean’s lookup is “java:comp/env/ejb/Services”.

As it is used by the web project (“JmtsWeb”) only, check its “web.xml” to see if the lookup string of the Helper class is according to the EJB reference name declared there:

[4] Even better then, use the newly introduced EJB lookup scheme for more fexibilty between the client lookup string and the real JNDI binding.
[5] Preferably move it to the web project.



The EJB reference name is “ejb/Services”, everything is fine.[6]

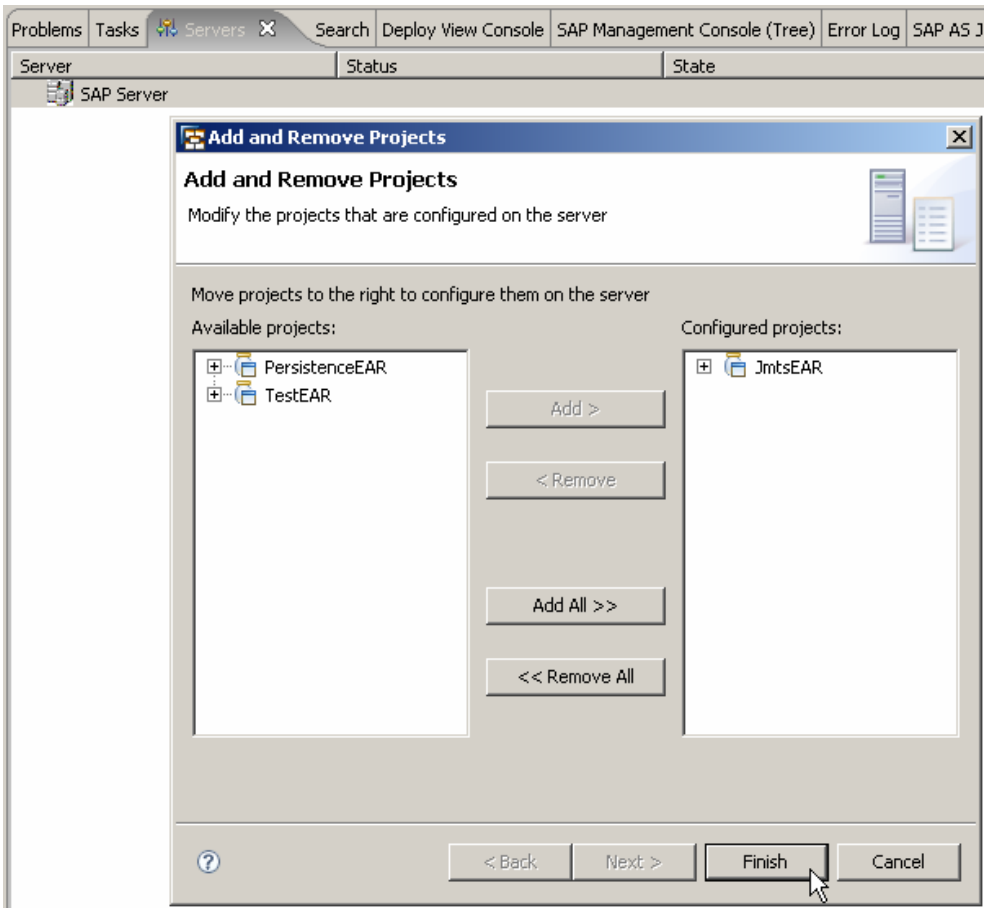
Deploy and run the Application

This is the time to check whether the application is ready to run.

To deploy it, switch to the J2EE perspective, select the “Server” view and right-click the “SAP Server” icon and select “Add and Remove Projects…”.

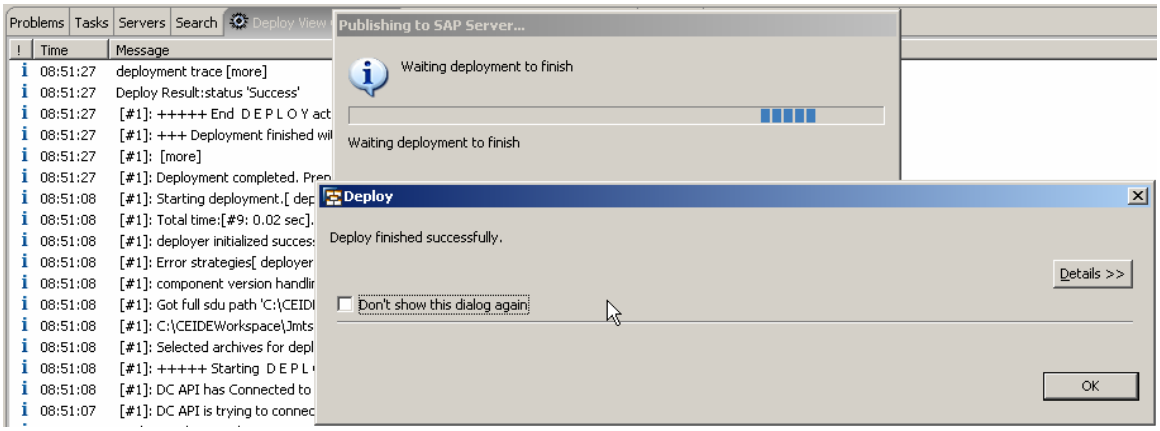
Now add your EAR project (“JmtsEAR”) by selecting it, pressing “Add >” and “Finish”:

[6] The “ListProjectsE.jsp” does the lookup on its own, without use of the Helper class; it already uses the correct lookup string, too.



You will then be asked to enter your user and password for the deployment:

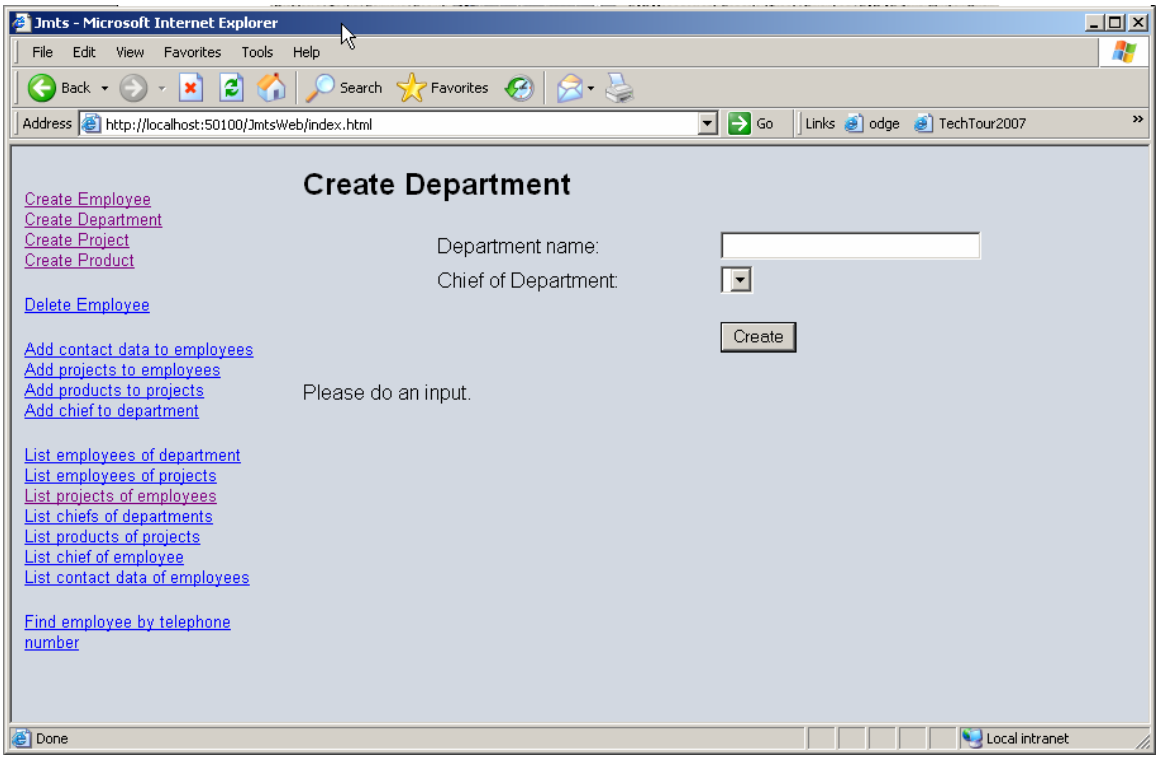
This should be the deployment's result:



Now let's run the application.

The URL should be something like http://localhost:50100/JmtsWeb/index.html. Change it according to your server, port and web project name (JmtsWeb, in this case).

This should be the result:



To give it a try, start by creating a Department and a Project.

Related Content

- The Migration Plug-in for SAP NetWeaver Composition Environment 7.1
- Sample application used in this how-to guide
- Screen cam that shows how to work with the Migration-Plug-in

Copyright