Application Class Loading on SAP NetWeaver Explained – Running Hibernate Using the So-Called Heavy Resources

Applies to

Any SAP NetWeaver – Java release compliant with Java EE 5 and defining the so-called Heavy Resources, i.e. NetWeaver Composition Environment 7.1 and above.

Any Java static library set – such as Hibernate 3.x.

Summary

This document discusses the basic concepts of the **Application Class Loading** on SAP NetWeaver Application Server – Java in the following often abbreviated as SAP NetWeaver Java Engine.

It also exemplary describes how to run **Hibernate** using to so-called **Heavy Resources** of the Java Engine. This example will work with an arbitrary version of Hibernate and any version of the SAP NetWeaver Java Engine compliant with Java EE 5 and introducing the Heavy Resources, i.e. **SAP NetWeaver Composition Environment 7.1** and above.

This example is not Hibernate specific – the same pattern will work for any arbitrary Java static library set.

Author: Goran Stoiljkovski

Company: SAP AG

Created on: 26 March 2009

Author Bio

Goran Stoiljkovski is a solution architect whose main focus is on architecture and design of distributed computed systems and solutions. He is member of the global Co-Innovation Lab (COIL) team as part of the SAP Global Ecosystem and Partner Group.

Table of Contents

Application Class Loading on SAP NetWeaver Java

This section is a brief introduction to the application class loading of the NetWeaver Java Engine. Take a look at ***Figure 1*** below – sketch based on Georgi Danov's blog on SDN: https://www.sdn.sap.com/irj/scn/weblogs?blog=/pub/wlg/6447.

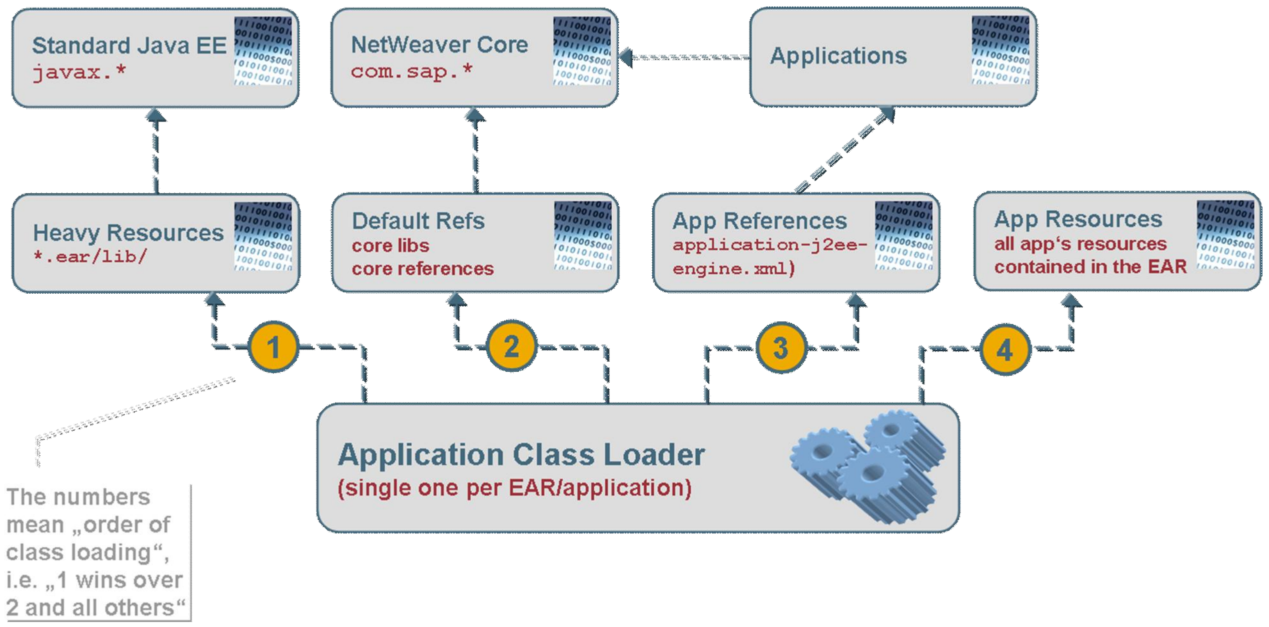

Standard Java EE
javax.*
NetWeaver Core
com.sap.*
Applications
Heavy Resources
*.ear/lib/
Default Refs
core libs
core references
App References
application-j2ee-engine.xml)
App Resources
all app's resources contained in the EAR
1
2
3
4
Application Class Loader
(single one per EAR/application)
The numbers mean „order of class loading", i.e. „1 wins over 2 and all others"

Figure 1

Let us roughly discuss this sketch for a moment:

- Applications on the NetWeaver platform are loaded with an **Application Class Loader** – and there is one and only one per application. An Enterprise Archive (EAR) can be identified with an application.
- As a basic principle, the NetWeaver follows the **parent-first** class loading pattern. As a consequence, this also means that if a class is to be loaded, NetWeaver looks for an already available instance of that class/object. The class loading hierarchy is prioritized according to ***Figure 1*** – the lower numbers in the class loading graph always win against the higher ones.
- One observes that i.e. the **Default References** win against the application references and the application resources. The Default References also hold a reference to the Core Engine classes.
- The application references are a feature of NetWeaver. As there can only be one class loader per application, NetWeaver has defined the **application references** in order to load resources (not only classes) from other applications. This app-to-app reference is set in the vendor specific **`application-j2ee-engine.xml`** descriptor of the referencing application. The reference is – so to speak – set as "referencingApp → referencedApp".
- The **Application Resources** class loader loads the resources defined in the application context – everything that is in the EAR of that application.
- The **Heavy Resources** are a class loading concept specific to the SAP NetWeaver Java and introduced just recently. They allow for defining flexible runtime class loading of resources deployed on the platform. The Heavy Resources define resources, which are loaded with a **top priority** in the class loading graph among all the resources previously mentioned. If an application is to be considered as Heavy Resource by NetWeaver, one has to **"switch it on"**. This is a property of the application (the EAR) controlled by a vendor specific descriptor called **`application-service.xml`** contained in the EAR. But note that this property is defined only for SAP Java EE 5 compliant archives/applications.

Note: One **usually and per default** deploys applications **without** switching on the heavy resources – and this is the use case that occasionally bears some disadvantages. We will discuss the answers to these dilemmas in the next section.

Deploying Hibernate within the Application

OK, that was a short introduction to the class loading architecture of the NetWeaver Java Engine. With this in mind, let us explain what happens, if one deploys Hibernate in the application context, i.e. as part of the application's EAR.

Note: This is probably the way most people would view as the most pragmatic one for running Hibernate on the SAP Java Engine – unfortunately. Deploying Hibernate within the application (as part of the application's EAR) will most likely end up with runtime errors/exceptions of/in the application, which of course is not the desired result.

This is the reason why:

- The Hibernate binary package comes with a lot of helper libraries – even in the core version one finds i.e. DOM4J, ANTLR, etc.
- A version of ANTLR is also included in the core libraries of SAP NetWeaver. This version of ANTLR is loaded with the default references and is so to speak of priority 2. If one puts the Hibernate package –and thus also the accompanying ANTLR library – in the EAR, this library will be loaded with priority 4. This means that in this case the ANTLR library delivered with the server is always loaded before the ANTLR library coming along with the application (Hibernate package). The results are runtime exceptions of type `ClassCastException`, `ClassNotFoundException`, etc, or even more severe errors like `ClassDefNotFoundError` thrown by instances trying to invoke or cast to ANTLR classes.
- The way out of this dilemma is to extract the Hibernate package out of the application and deploy it as a "heavy reference application". The "actual" application then references this "heavy reference application" and everything is fine.

In the following section we will describe this process step-by-step.

Deploying Hibernate as a Heavy Resource Step-By-Step

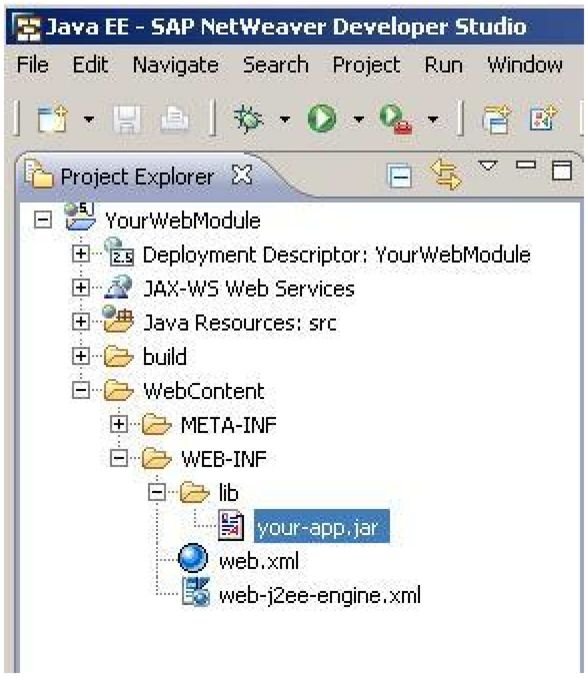
Here a detailed description of that process using the SAP NetWeaver Developer Studio. We will reassemble your application to consist of the following three archives

- An enterprise application (EAR) called ***YourApp*** – without Hibernate so to speak
- A web module (WAR) included in the YourApp's enterprise archive called ***YourWebModule***
- And a Java EE 5 compliant enterprise application (EAR) called ***Hibernate*** containing solely the Hibernate package.

Step 1 – Define a Web Module

You define a web module (in this case called ***YourWebModule***) and there you put all the application's resources – the ones for the web application as well as the application's logic. In this case you may view the `my-app.jar` as a library containing the application's logic (business or control logic).

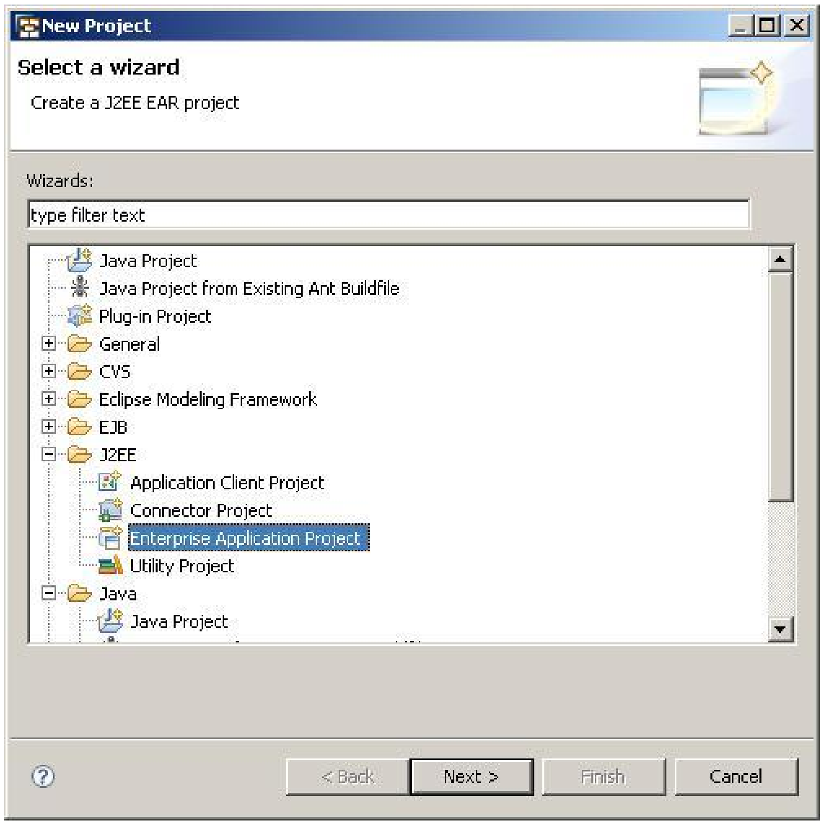
Navigation: *New → Project → Web → Dynamic Web Project.*



Step 2 – Define Application's Enterprise Archive

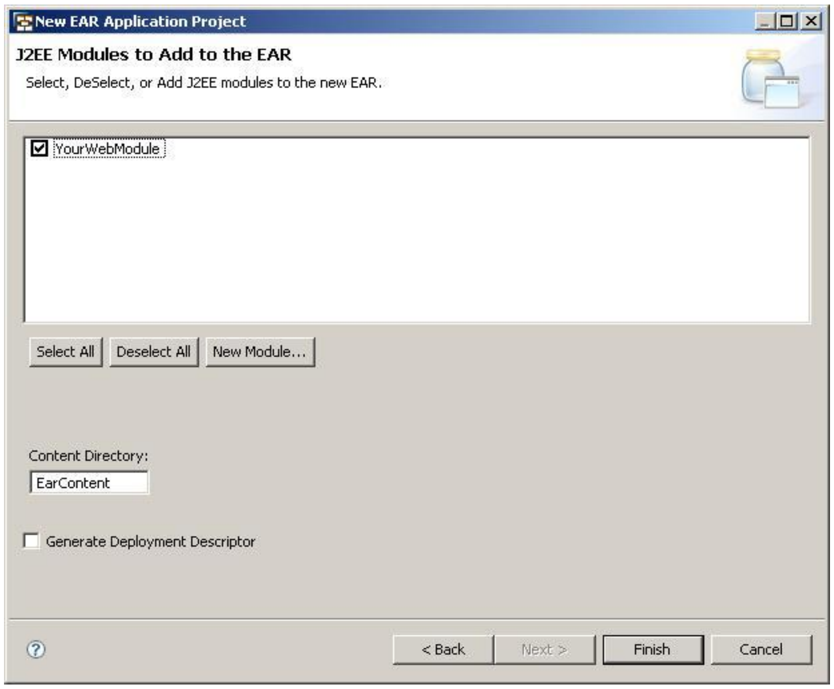
You define an enterprise application via the Enterprise Application Project in the creation wizard. The project name is the application's name ***YourApp***.

Navigation: *New → Project → J2EE → Enterprise Application Project*



Step 3 – Build References for the Application

Set a "build" reference to your web application, so the newest web module is included in the application module every time you build the enterprise application.


New EAR Application Project
J2EE Modules to Add to the EAR
Select, DeSelect, or Add J2EE modules to the new EAR.
Select All
Deselect All
New Module...
< Back
Next >
Finish
Cancel

☑ YourWebModule

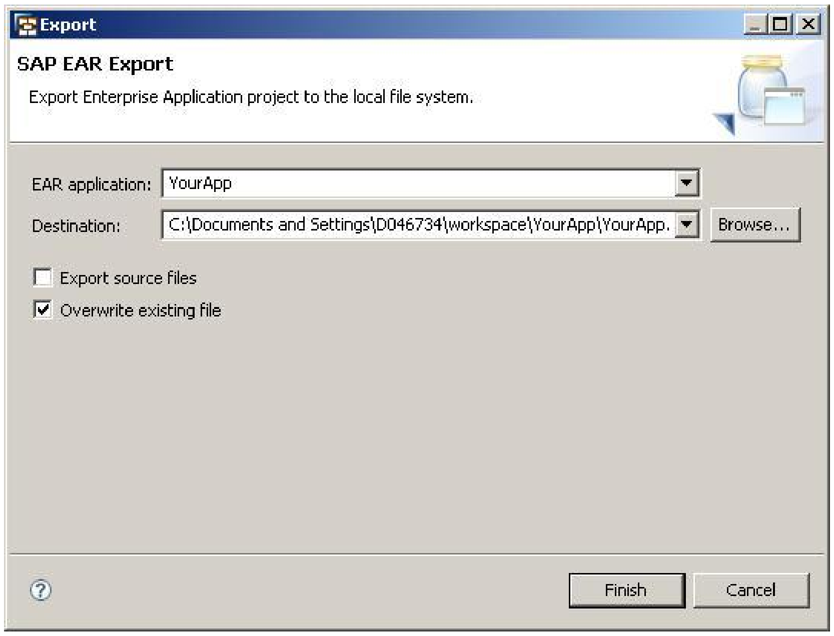
Content Directory: EarContent

☐ Generate Deployment Descriptor

Step 4 – Check the Correctness of the Build Reference

You can check if this reference is set correctly by exporting the ear file to the file system. Unzip the `*.ear` file to check if the `*.war` file is included.

Navigation: *New* → *Export* → *SAP EAR File*.


Export
SAP EAR Export
Export Enterprise Application project to the local file system.
Browse...
Finish
Cancel

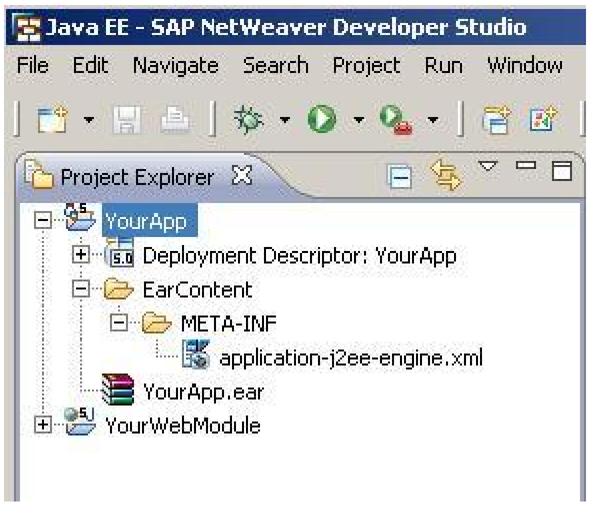
EAR application: YourApp

Destination: C:\Documents and Settings\D046734\workspace\YourApp\YourApp.

☐ Export source files

☑ Overwrite existing file

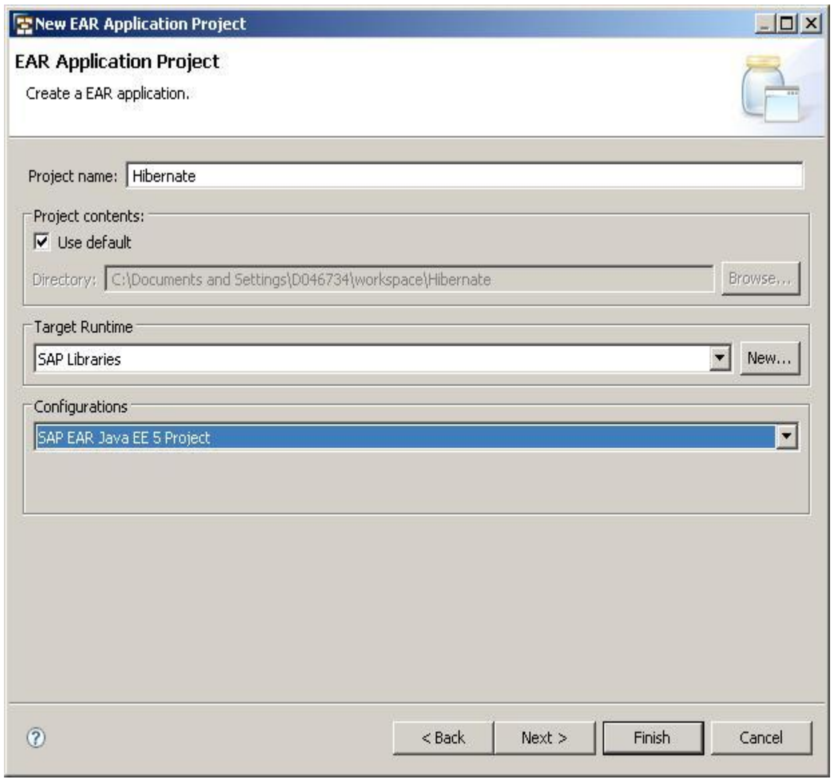
If you export the file on the root level of your enterprise application, you will also see it in the IDE (NetWeaver Developer Studio)


Java EE - SAP NetWeaver Developer Studio
File Edit Navigate Search Project Run Window
Project Explorer
YourApp
Deployment Descriptor: YourApp
EarContent
META-INF
application-j2ee-engine.xml
YourApp.ear
YourWebModule

Step 5 – Create the Heavy Resource Application

Create your second enterprise application project (**Hibernate**). In this case you have to make sure you define this project as **“SAP EAR Java EE 5 Project”**.

IMPORTANT: This is necessary since the “heavy references” work only for Java EE 5 applications!



New EAR Application Project

EAR Application Project

Create a EAR application.

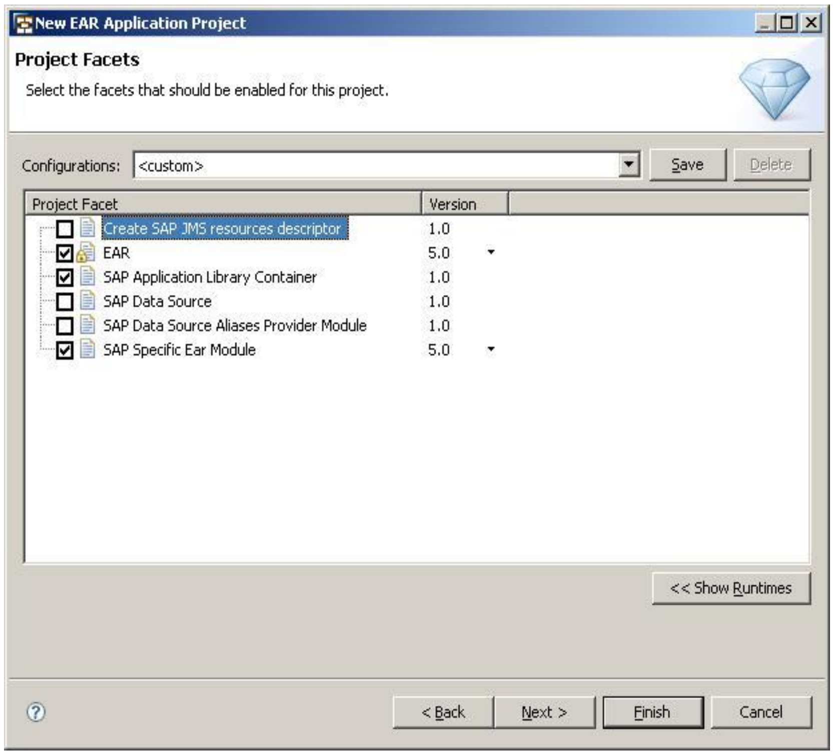
Project name: Hibernate

Project contents:
☑ Use default
Directory: C:\Documents and Settings\D046734\workspace\Hibernate

Target Runtime: SAP Libraries

Configurations: SAP EAR Java EE 5 Project

Make sure that the property **“SAP Application Library Container”** is checked. This is a property for SAP’s deployment and runtime management. This way the deployed enterprise application (in this case Hibernate) is marked as a “heavy resource”.



New EAR Application Project

Project Facets

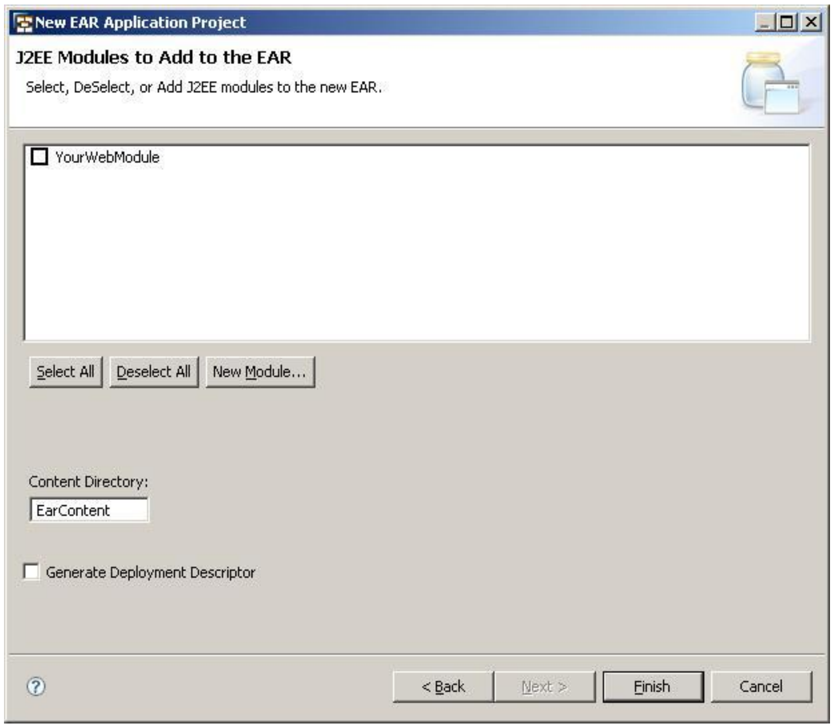
Select the facets that should be enabled for this project.

Configurations: <custom>

Project Facet	Version
☐ Create SAP JMS resources descriptor	1.0
☑ EAR	5.0
☑ SAP Application Library Container	1.0
☐ SAP Data Source	1.0
☐ SAP Data Source Aliases Provider Module	1.0
☑ SAP Specific Ear Module	5.0

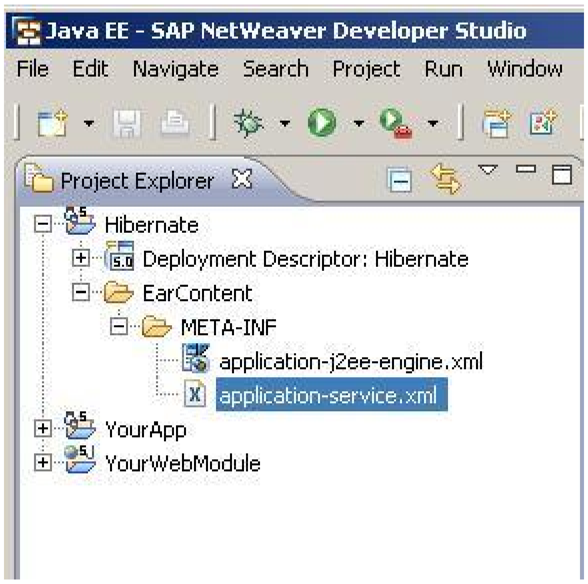
Please **do not add** any web modules in this application – even though the IDE will recognize the web modules in your open workspace and will ask if you want to add them to your enterprise application.

Name your application ***"Hibernate"***.


New EAR Application Project
J2EE Modules to Add to the EAR
Select, DeSelect, or Add J2EE modules to the new EAR.
YourWebModule
Select All
Deselect All
New Module...
Content Directory:
EarContent
Generate Deployment Descriptor
< Back
Next >
Finish
Cancel

This is the result viewed in the IDE.

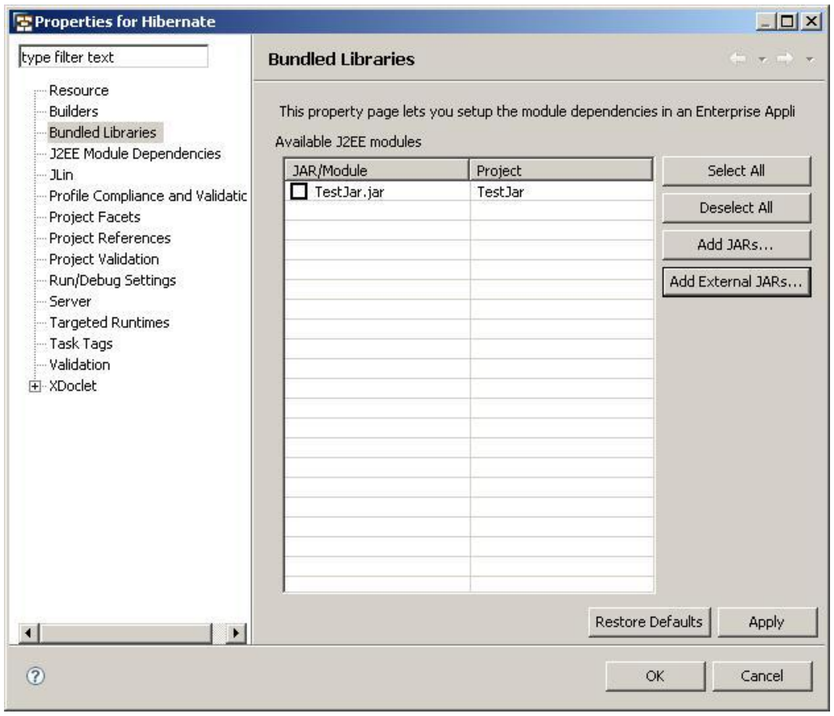
You can also see the SAP Java Engine specific descriptor `application-service.xml` needed for the application to be marked as a heavy resource.


Java EE - SAP NetWeaver Developer Studio
File Edit Navigate Search Project Run Window
Project Explorer
Hibernate
Deployment Descriptor: Hibernate
EarContent
META-INF
application-j2ee-engine.xml
application-service.xml
YourApp
YourWebModule

Step 6 – Define Heavy Resource's Specific Libraries

Now you have to add the Hibernate specific libraries to this EAR.

Go to the context menu of the Hibernate project and call the Preferences. Click on the item *"Bundled Libraries"* and then *"Add External Jars"* from the file system where they are located.

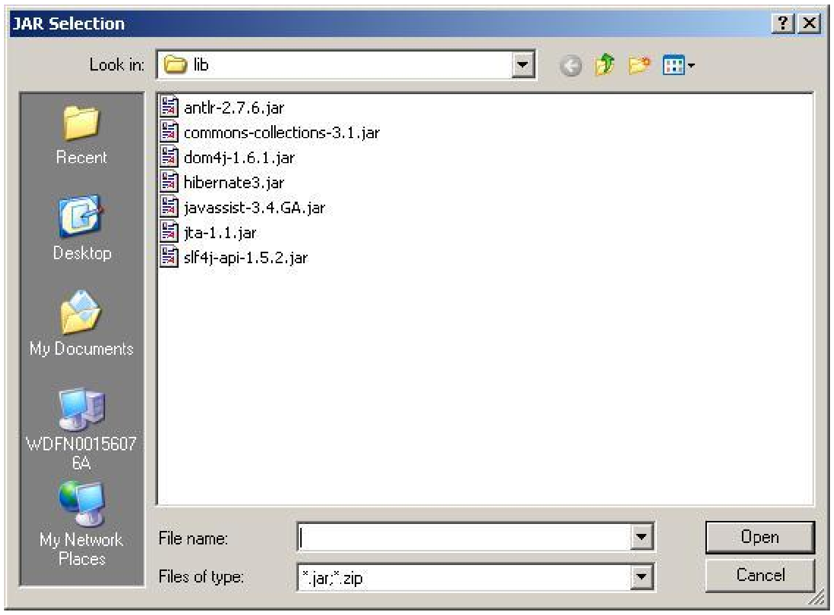


This property page lets you setup the module dependencies in an Enterprise Appli

Available J2EE modules

JAR/Module	Project
☐ TestJar.jar	TestJar

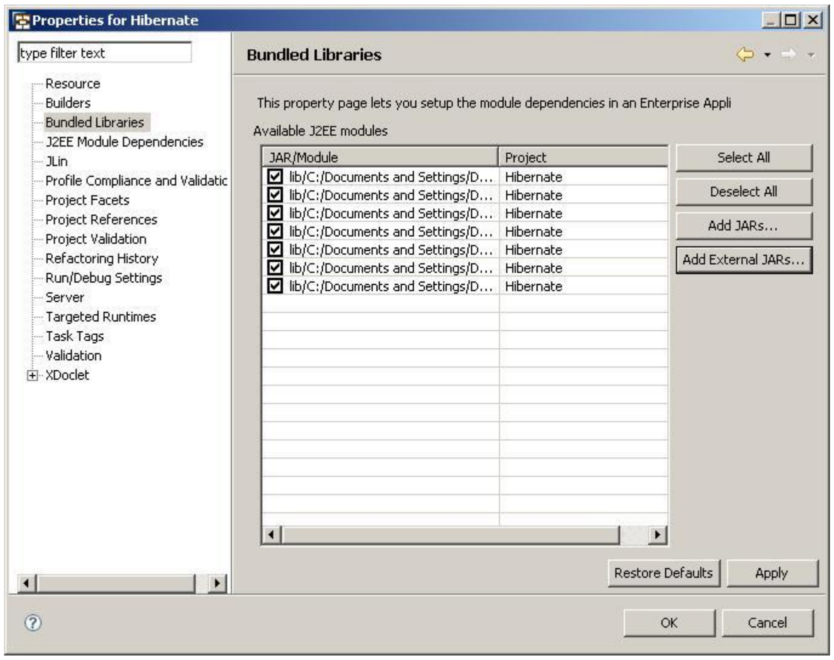
This is a list of JARs we add to our Hibernate project. In your case this list may be much longer depending on Hibernate's flavor you are using in your application. We exemplarily used the basic version.



antlr-2.7.6.jar
commons-collections-3.1.jar
dom4j-1.6.1.jar
hibernate3.jar
javassist-3.4.GA.jar
jta-1.1.jar
slf4j-api-1.5.2.jar

.jar;.zip

This is what you will see after this step. Complete the import by saying “OK”.



This property page lets you setup the module dependencies in an Enterprise Appli

Available J2EE modules

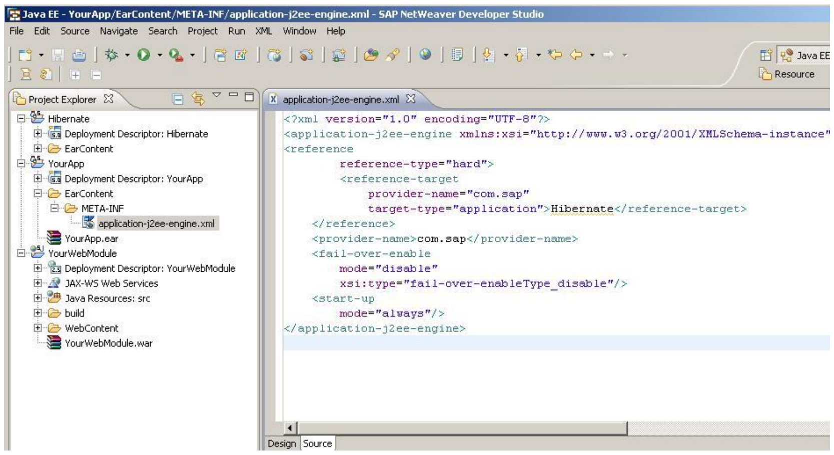
JAR/Module	Project
☑ lib/C:/Documents and Settings/D...	Hibernate
☑ lib/C:/Documents and Settings/D...	Hibernate
☑ lib/C:/Documents and Settings/D...	Hibernate
☑ lib/C:/Documents and Settings/D...	Hibernate
☑ lib/C:/Documents and Settings/D...	Hibernate
☑ lib/C:/Documents and Settings/D...	Hibernate
☑ lib/C:/Documents and Settings/D...	Hibernate

Step 7 – Set Applications' Class Loading References

Now you have to set a reference from your application **YourApp** to the **Hibernate** application. This is done by editing the descriptor `application-j2ee-engine.xml` of the referencing application and that is **YourApp**.

You will find the listing of this file at the end of this document.

For the meaning of the XML tags/elements and their attributes please refer to the online documentation on http://help.sap.com

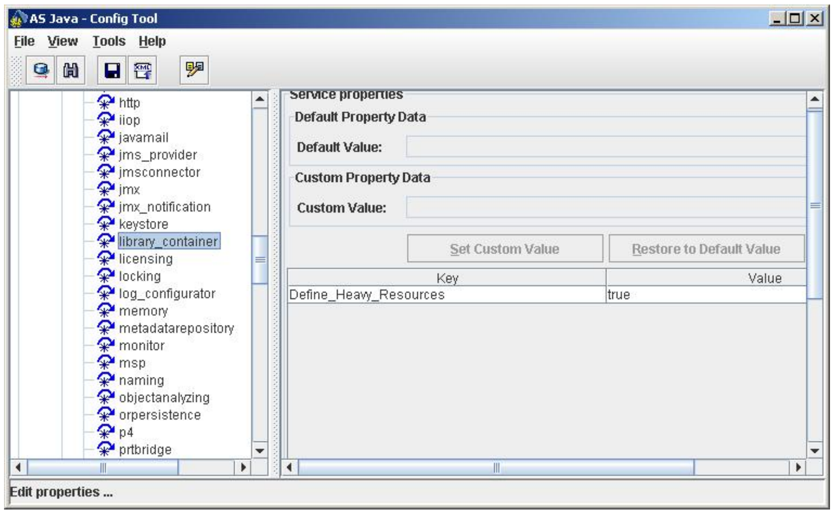

Java EE - YourApp/EarContent/META-INF/application-j2ee-engine.xml - SAP NetWeaver Developer Studio
File Edit Source Navigate Search Project Run XML Window Help
Java EE
Resource
Project Explorer
Hibernate
Deployment Descriptor: Hibernate
EarContent
YourApp
Deployment Descriptor: YourApp
EarContent
META-INF
application-j2ee-engine.xml
YourApp.ear
YourWebModule
Deployment Descriptor: YourWebModule
JAX-WS Web Services
Java Resources: src
build
WebContent
YourWebModule.war
application-j2ee-engine.xml
Design
Source

```
<?xml version="1.0" encoding="UTF-8"?>
<application-j2ee-engine xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
<reference
        reference-type="hard">
        <reference-target
            provider-name="com.sap"
            target-type="application">Hibernate</reference-target>
    </reference>
    <provider-name>com.sap</provider-name>
    <fail-over-enable
        mode="disable"
        xsi:type="fail-over-enableType_disable"/>
    <start-up
        mode="always"/>
</application-j2ee-engine>
```

Step 8 – Check the Heavy Resources Configuration

Before you deploy your applications to the Java Engine – and especially the Hibernate application – you should make sure that the version of the Engine you are running is configured for deployment of the heavy resources. This configuration is a default in newest versions of the engine, so most likely you will not have to change anything.

Open the offline **Config Tool** located in the <install-home>\<instance-name>\j2ee\configtool and start the `configtool.bat`. Connect to the system database and go to *Services → library_container*. You should set the **"Define_Heavy_Resources"** property to "**true"** if not already set.

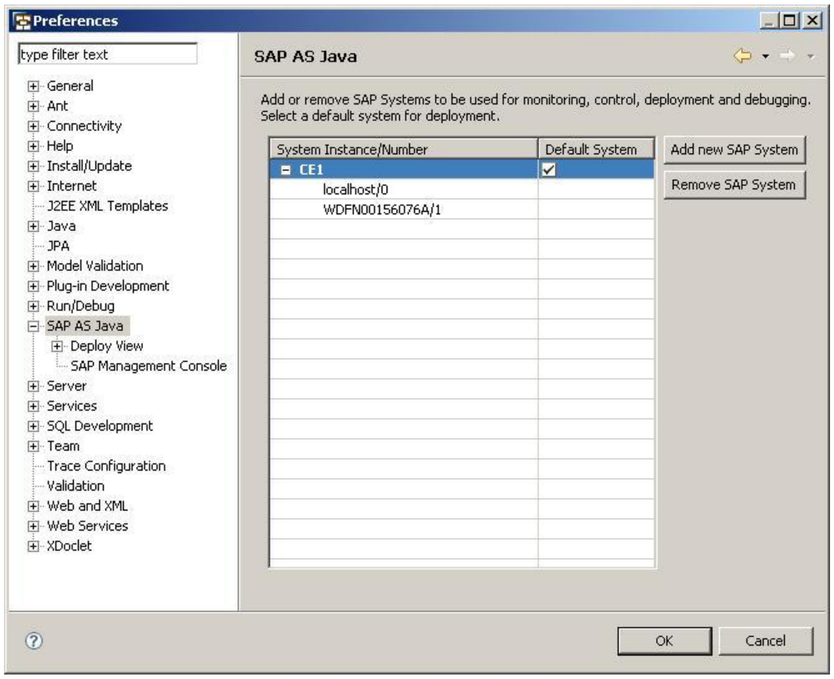


Key	Value
Define_Heavy_Resources	true

Step 9 – Deployment

Now you have to deploy your applications. Remember that you have two applications: **Hibernate** and **YourApp**.

Make sure that your IDE knows the server you are deploying to. You can check that in *Window → Preferences → SAP AS Java*.



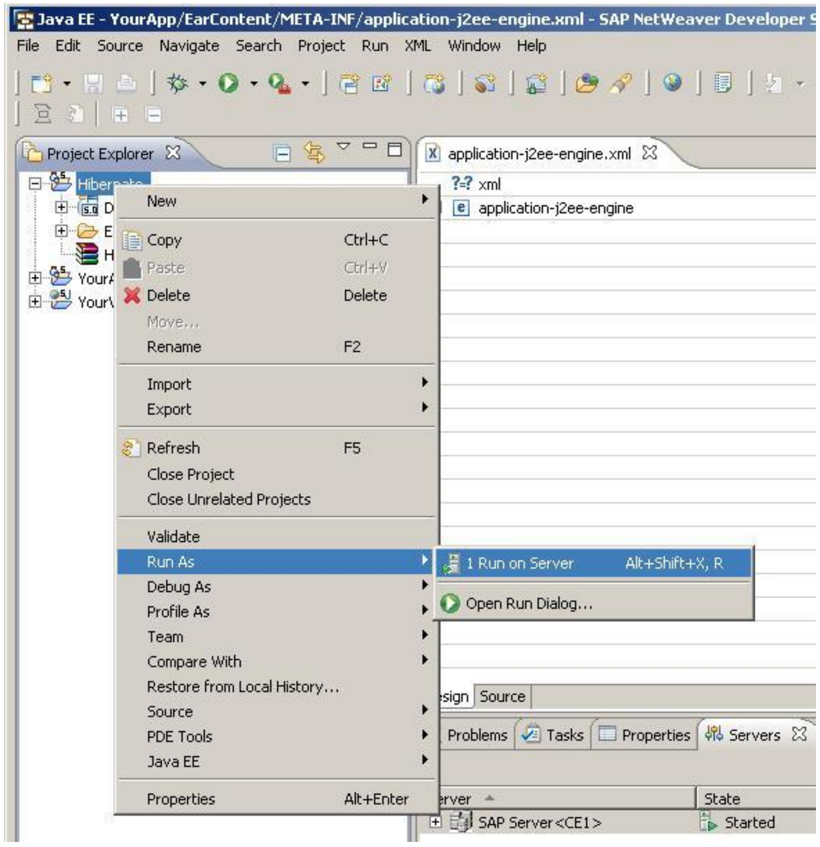
Add or remove SAP Systems to be used for monitoring, control, deployment and debugging. Select a default system for deployment.

System Instance/Number	Default System
CE1	✓
localhost/0	
WDFN00156076A/1	

Since YourApp is has a hard reference to the Hibernate application, you will have to deploy **Hibernate** first.

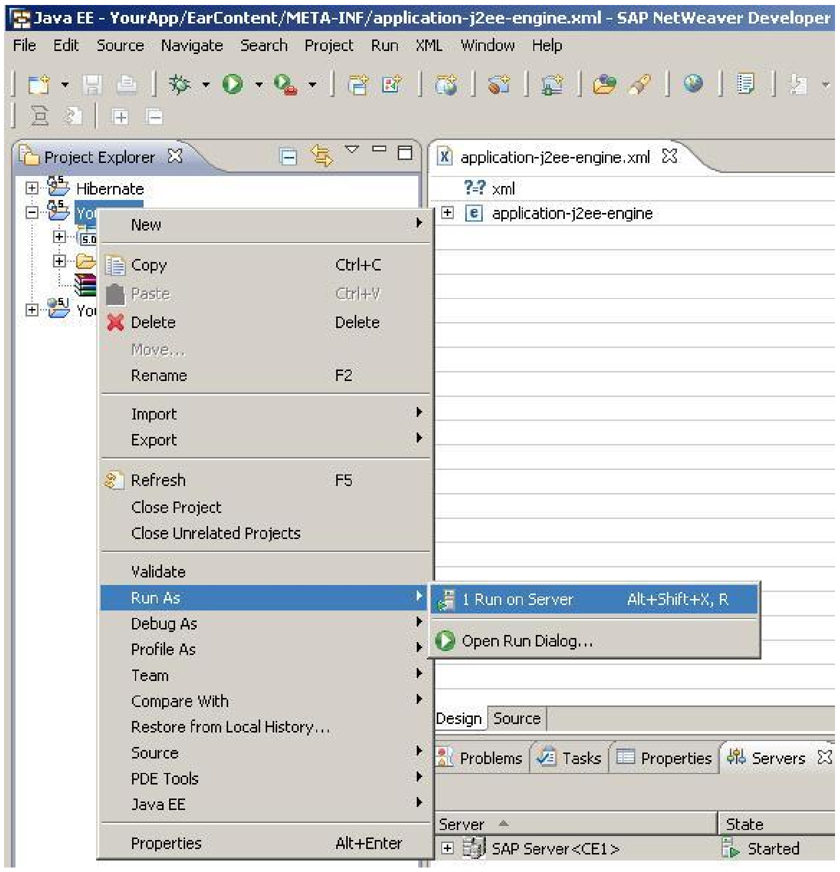
In the IDE right-click the Hibernate project and go to *Run As → Run on Server.*

You will get a message (a pop-up window) about the successful deployment of your **Hibernate** application.



In the same way deploy the **YouApp** application.

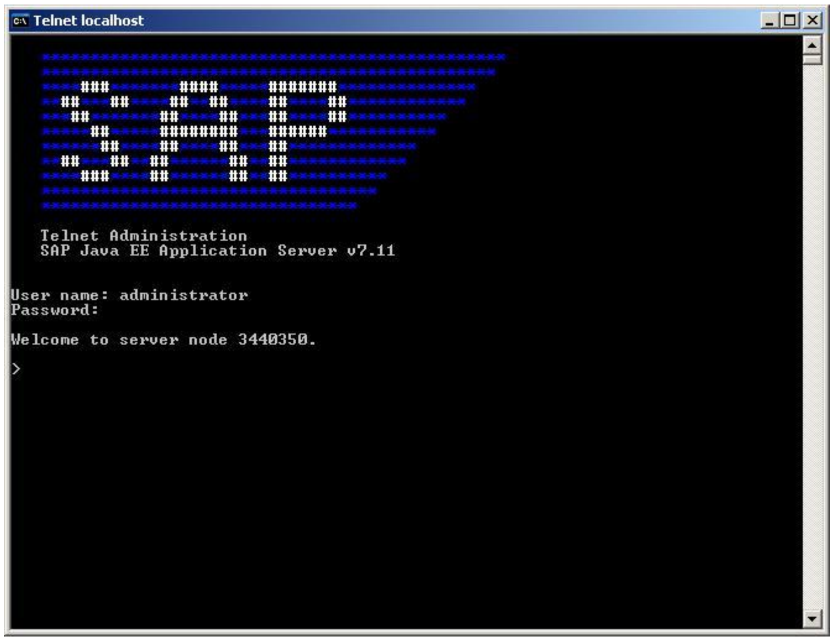
That's it! You have successfully performed the deployment of both applications.



Step 10 – Final Check

Now you should check if everything is in place – especially if the Hibernate application was successfully deployed as a Heavy Resource.

Logon to the server via telnet. Bear in mind that telnet does not work remotely but only on the local machine, where your Java Engine is running.

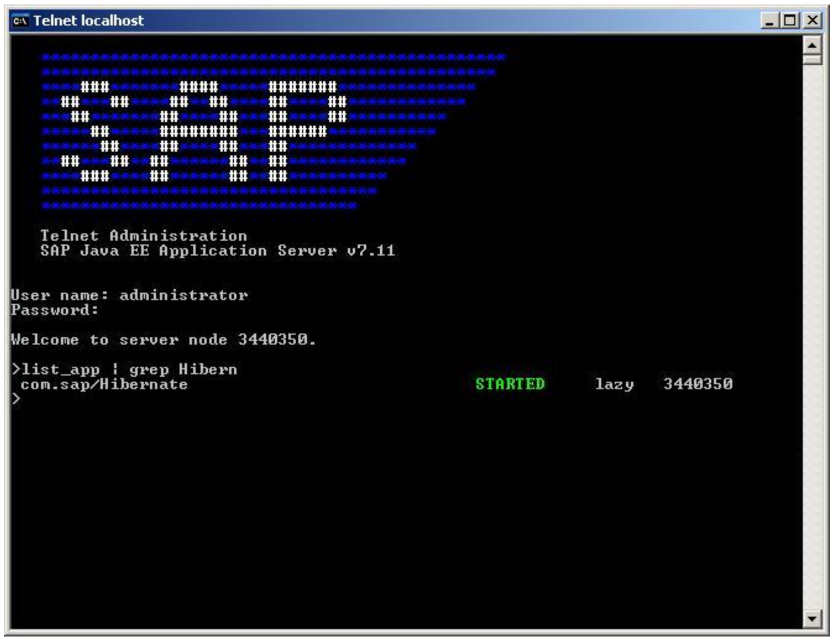


```
Telnet localhost

   Telnet Administration
   SAP Java EE Application Server v7.11

User name: administrator
Password:

Welcome to server node 3440350.

>
```

List your Hibernate application by executing the **`list_app`** command. You can **`grep`** for your Hibernate application and thus avoid listing all the applications deployed on the engine:

`list_app | grep Hibern`

You will see the status of the application, which should be "started".

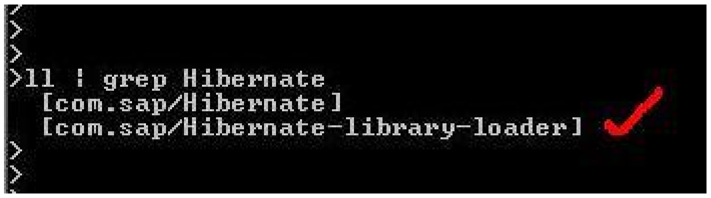


```
Telnet localhost

   Telnet Administration
   SAP Java EE Application Server v7.11

User name: administrator
Password:

Welcome to server node 3440350.

>list_app | grep Hibern
 com.sap/Hibernate                                STARTED     lazy    3440350
>
```

Now you can see if the Hibernate application was deployed as a heavy resource. Execute the command (ll = list libraries)

`ll | grep Hibernate`

The last line confirms that Hibernate was deployed as a heavy resource.



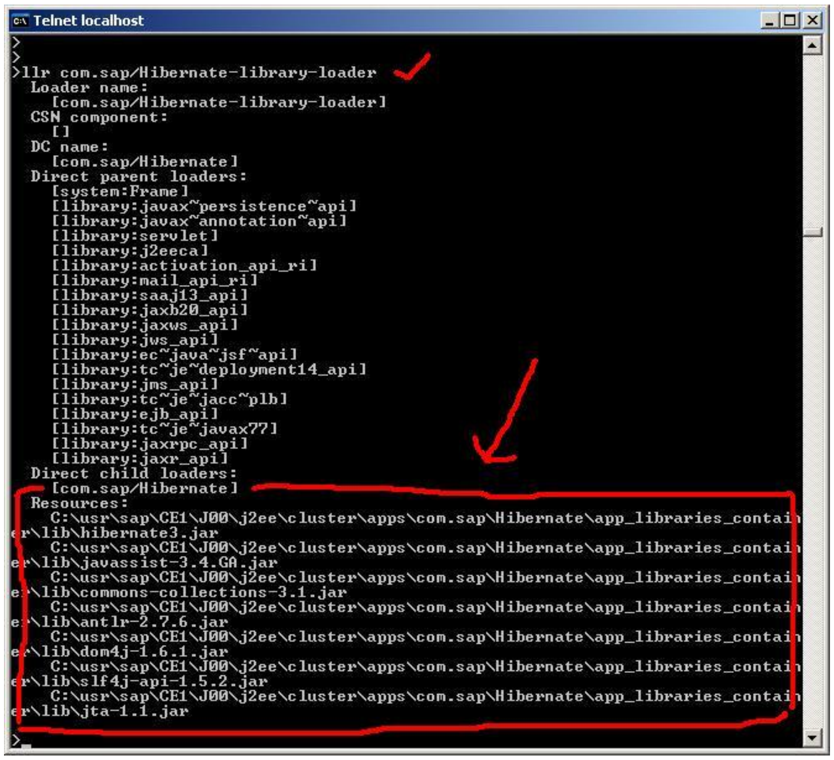
```
>
>
>ll | grep Hibernate
  [com.sap/Hibernate]
  [com.sap/Hibernate-library-loader]
>
>
```

Now you should check if all the desired jars are part or the Hibernate application

`llr com.sap/Hibernate-library-loader`

This command lists the associated resources (JARs) with the heavy resource Hibernate.

That is all.



```
>
>
>llr com.sap/Hibernate-library-loader
  Loader name:
    [com.sap/Hibernate-library-loader]
  CSN component:
    []
  DC name:
    [com.sap/Hibernate]
  Direct parent loaders:
    [system:Frame]
    [library:javax~persistence~api]
    [library:javax~annotation~api]
    [library:servlet]
    [library:j2eeca]
    [library:activation_api_ri]
    [library:mail_api_ri]
    [library:saaj13_api]
    [library:jaxb20_api]
    [library:jaxws_api]
    [library:jws_api]
    [library:ec~java~jsf~api]
    [library:tc~je~deployment14_api]
    [library:jms_api]
    [library:tc~je~jacc~plb]
    [library:ejb_api]
    [library:tc~je~javax77]
    [library:jaxrpc_api]
    [library:jaxr_api]
  Direct child loaders:
    [com.sap/Hibernate]
  Resources:
    C:\usr\sap\CE1\J00\j2ee\cluster\apps\com.sap\Hibernate\app_libraries_contai
er\lib\hibernate3.jar
    C:\usr\sap\CE1\J00\j2ee\cluster\apps\com.sap\Hibernate\app_libraries_contai
er\lib\javassist-3.4.GA.jar
    C:\usr\sap\CE1\J00\j2ee\cluster\apps\com.sap\Hibernate\app_libraries_contai
er\lib\commons-collections-3.1.jar
    C:\usr\sap\CE1\J00\j2ee\cluster\apps\com.sap\Hibernate\app_libraries_contai
er\lib\antlr-2.7.6.jar
    C:\usr\sap\CE1\J00\j2ee\cluster\apps\com.sap\Hibernate\app_libraries_contai
er\lib\dom4j-1.6.1.jar
    C:\usr\sap\CE1\J00\j2ee\cluster\apps\com.sap\Hibernate\app_libraries_contai
er\lib\slf4j-api-1.5.2.jar
    C:\usr\sap\CE1\J00\j2ee\cluster\apps\com.sap\Hibernate\app_libraries_contai
er\lib\jta-1.1.jar

>_
```

Listing of Application-j2ee-engine.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<application-j2ee-engine xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="application-j2ee-engine.xsd">
          <reference reference-type="hard">
          <reference-target provider-name="com.sap"
             target-type="application">Hibernate</reference-target>
          </reference>
          <provider-name>com.sap</provider-name>
          <fail-over-enable
             mode="disable"
             xsi:type="fail-over-enableType_disable"/>
          <start-up
             mode="always"/>
</application-j2ee-engine>
```

Copyright