OpenRide Installation Guide

Table of Contents

(Re-) Deployment

Steps for deployment:

1. apply database dump initially (see below for details on setting up the database)

2. copy .ear file to %glassfish_home%/domains/domain<i>/autodeploy

3. copy a routing Graph file to %glassfish_home%/domains/domain<i>/routingGraph.hh or to %glassfish_home%/domains/domain<i>/config/routingGraph.hh   (glassfishes seem to be different)

Database Installation and Setup

At first one has to be aware of the possibility to use remote databases too. That would mean, one could jump to the next part concerned with the installation of glassfish. Otherwise follow these instructions.

http://www.postgresql.org/ : version: 8.3.7-1(http://www.enterprisedb.com/getfile.jsp?fileid=485) PostGis: http://postgis.refractions.net/

Admin tool: pgAdmin : http://www.pgadmin.org/ (should be installed during postgres installation otherwise it can be postinstalled)

After the **installation** of Postgres Version 8.3.7-1 one can use the "Stack-Builder"-GUI to install the PostGIS Plugin (1.3.6).

Database setup

- At first one has to create a new user (use pgAdmin) with the following parameters:

  o Username: „openride" (as an example – choose your own)

  o Password: „$1openride"(as an example – choose your own)

  o Katalog Ändern: Ja

  o Datenbank erzeugen:Ja

  o Kann einloggen: Ja

  o Superuser:ja

- Now one has to create the database:

  o DB name: "openride" (as an example – choose your own)

  o DB owner: "openride" (as chosen in step 1)

Restoring the database from a dump

- Connect to DB server

- Select „openride" DB (as named above)

- Right click and restore (wiederherstellen)

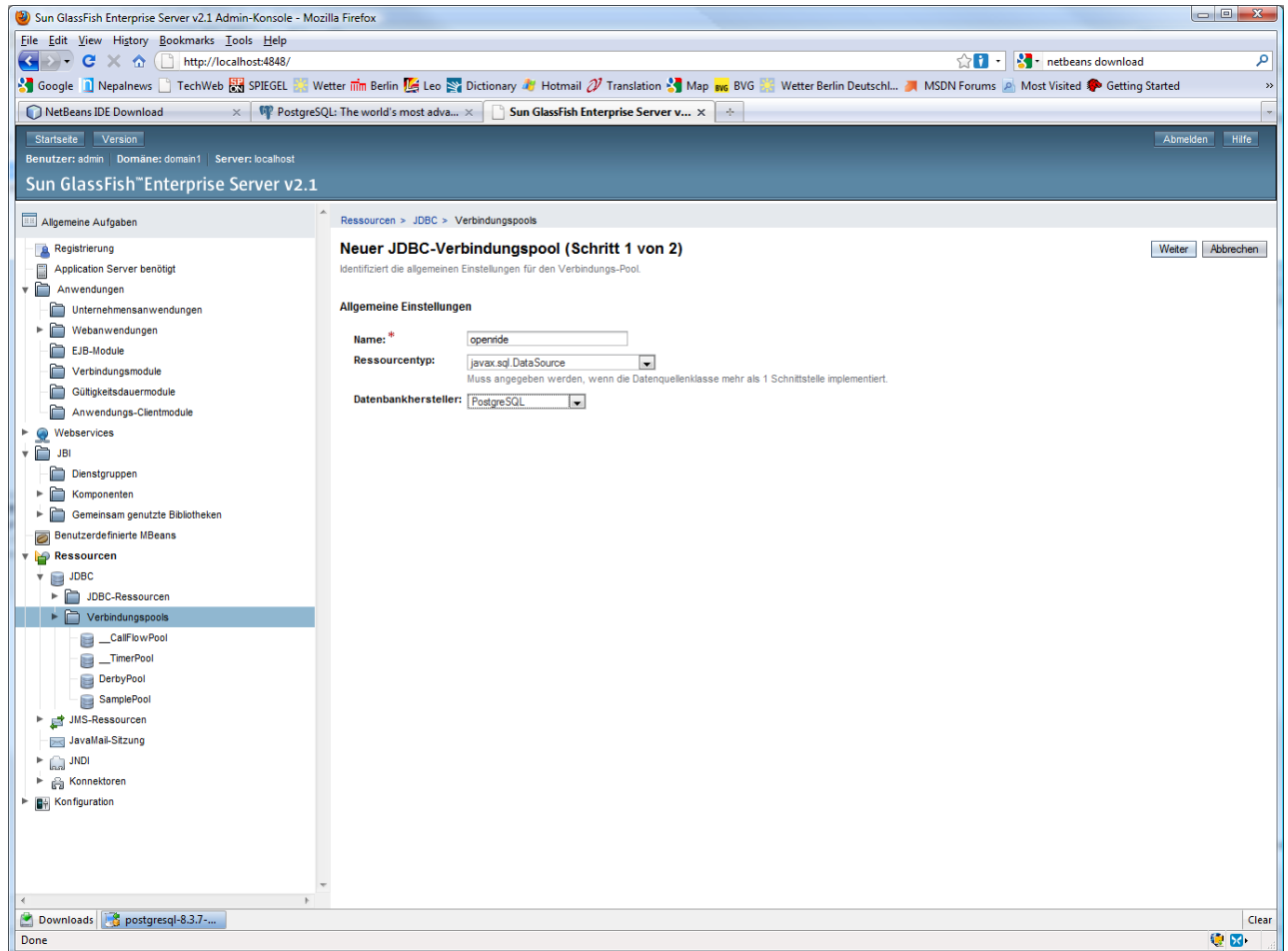- dump_openride.sql (search @: %openrideSVN%/documents/OpenRideDB/)

-
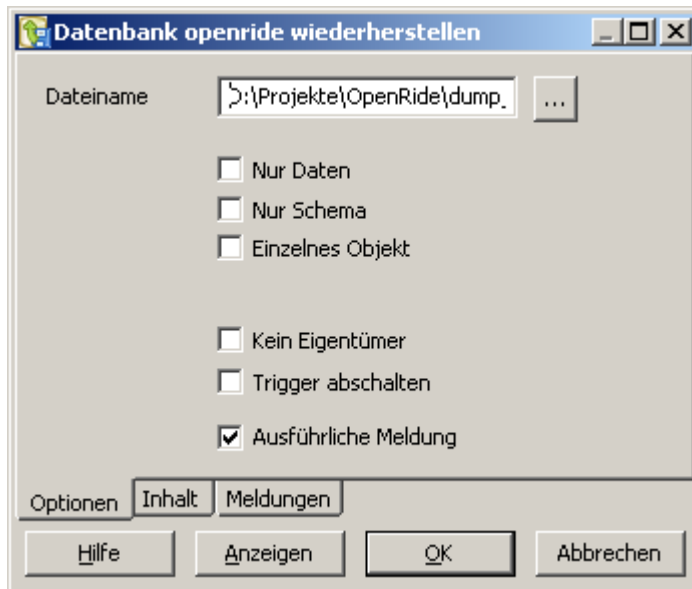
- Select sql dump & OK (Exitcode 1 may be ignored…)

- **NOTE: remove all tables if update is not correct.**

Glassfish Setup (Version 3.01)

One has to configure a Connection-Pool with the following information:

- Name: "openride" (as an example – choose your own)

- Resource type: "javax.sql.DataSource"

- Database Vendor: "PostgreSQL"

- Classname of Datasource: "org.postgresql.ds.PGSimpleDataSource"

- User: "openride" (as chosen during database setup)

- portNumber: "5432"

- databaseName: "openride" (as chosen during database setup)

- Password: "$1openride" (as chosen during database setup)

- serverName: "localhost" (if not using a remote database)

At this point it is important to be aware of the possibility to choose a **remote database.** In order to use the remote database connection it's necessary to configure the remote database to allow you to connect. For this you have to start pgAdmin on the server, connect to the server, click on the database and choose Tools -> Serverconfiguration -> pg_hba.conf. There you have to add a new entry with your ip mask. Let's say your ip is 10.147.66.XXX, and then your entry should look like this:

Host   all   all   10.147.66.0/24   md5

Save this and reload the configuration with the green play button. That's it!
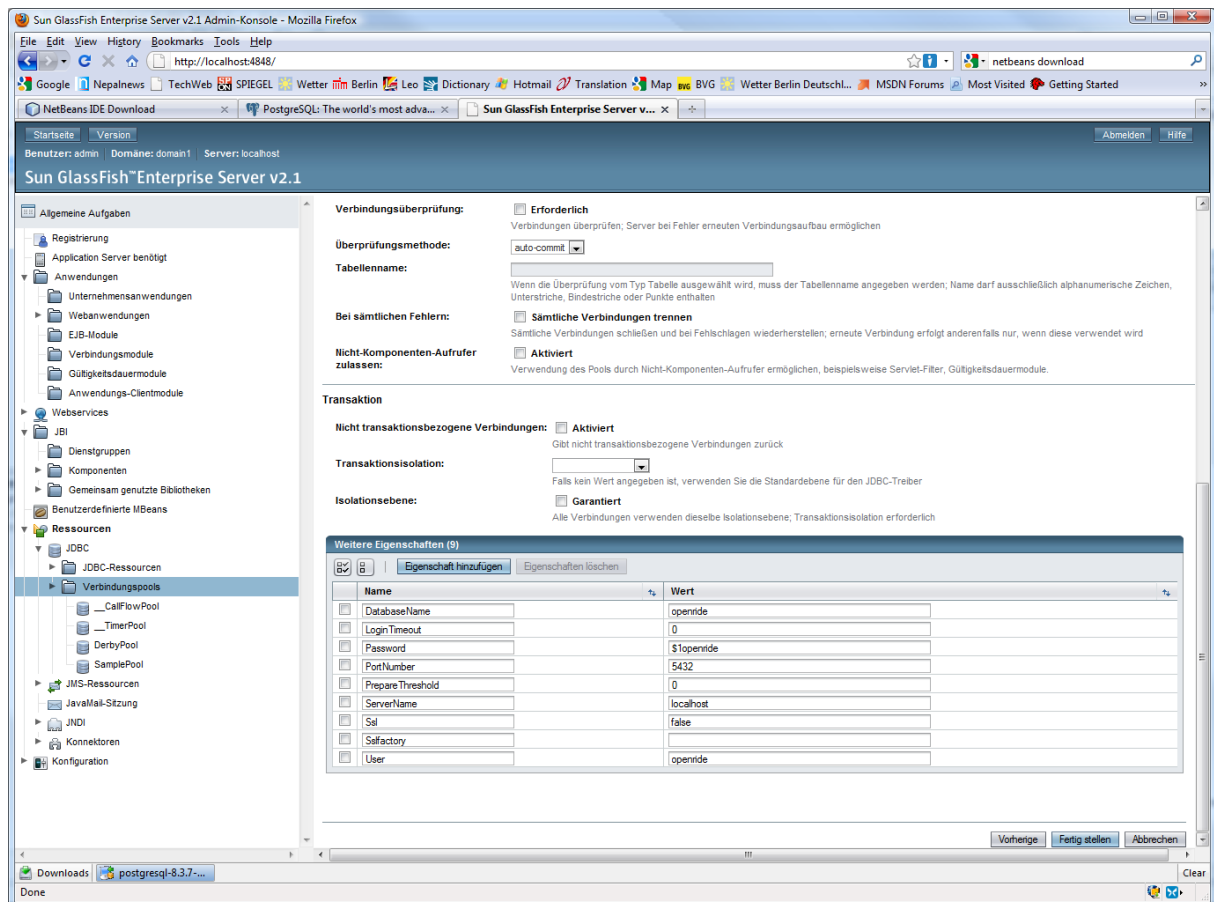
Aftwerwards one has to apply to the name of the pool to one JDBC-Ressource named "jdbc/openride".
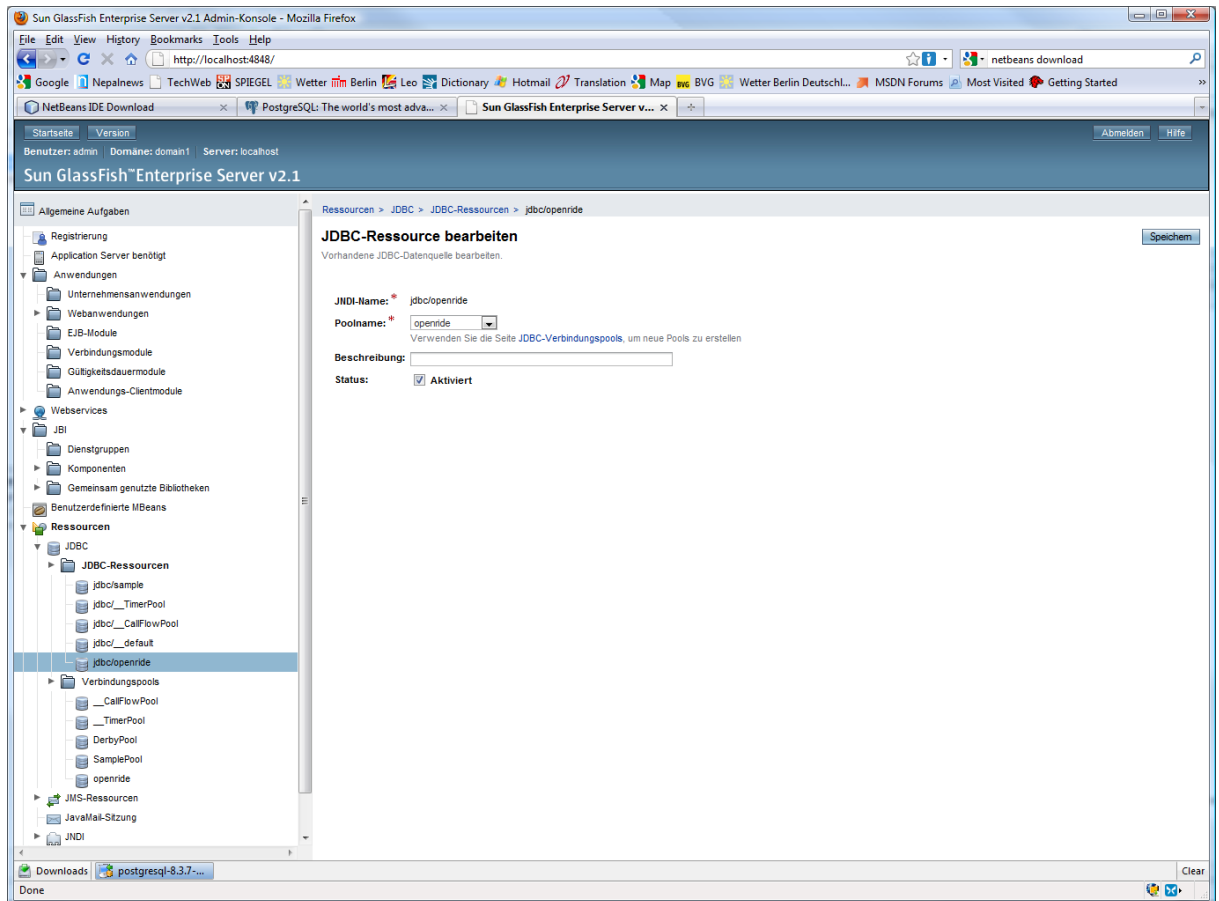
If you've done things successfully till this point you have to copy the file named "postgresql-8.3-603.jdbc3.jar" to the domains (e.g. %glassfish%/domains/domain1/lib) lib directory (this file is normally added to the ejb-Project at "root/libs" folder).

Since you've done that you can start the **deployment** of the "OpenRideServer.ear" file. For example by putting the file in the autodeploy directory or going via commandline to %glassfish%/bin using the asadmin tool.

Hints

ATTENTION: If you get the following error:

```
{
        CLI171 Command deploy failed : Deploying application in domain failed;
        Internal Exception: java.sql.SQLException: This pool is not registered with the
        runtime environment : null
        Error Code: 0
}
```

This means, you have not set the right Connection-Pool in the ejb-Project at the persistence.xml file. Normally this should be jdbc/openride. Only in case of deployment on robusta as developer Version you should use jdbc/openride_dev, to separate developer and customer versions of the application.

Hint:
To start glassfish in verbose-mode (see all the logs on the console), type the following to the console:
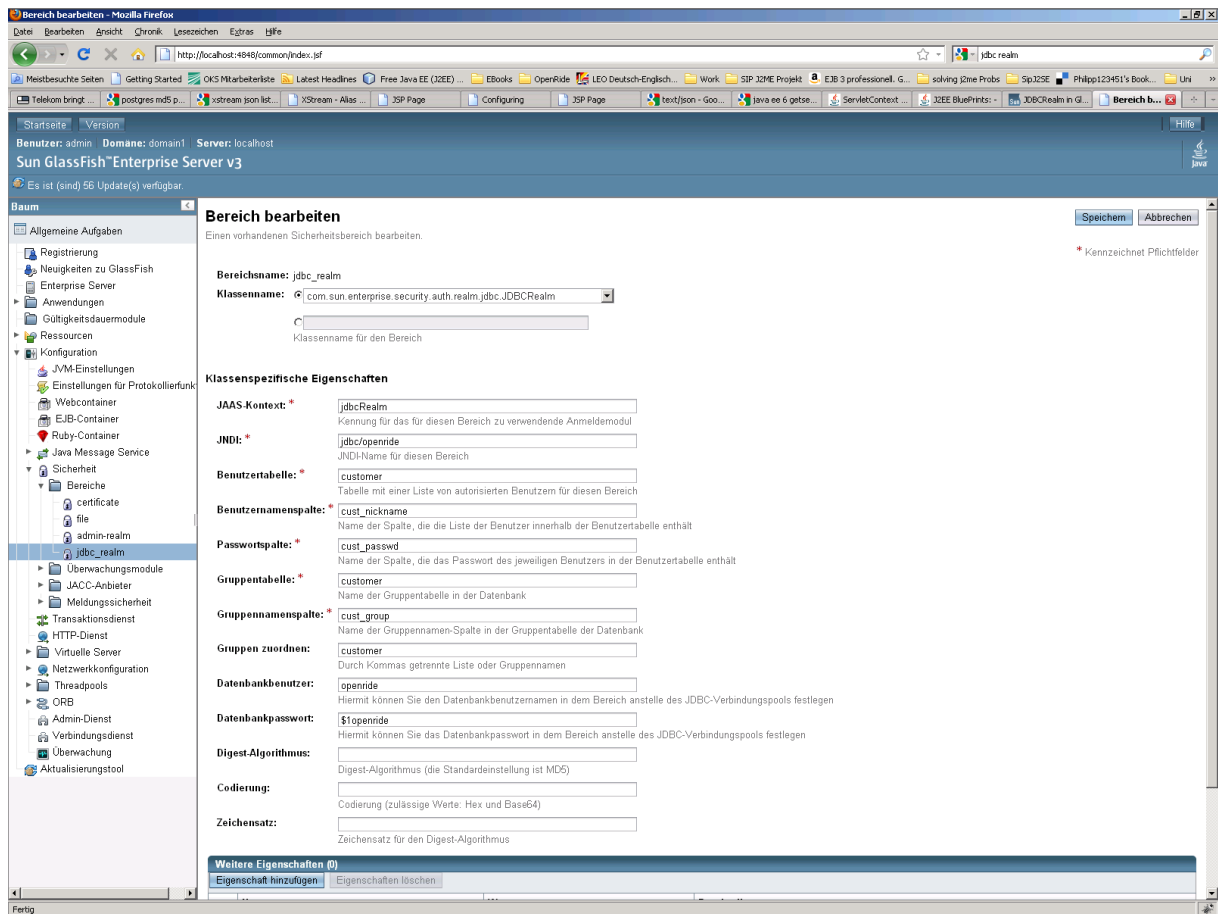   *asadmin start-domain –verbose domain2* (e.g.)
To stop glassfish, type:
        asadmin stop-domain domain2 (e.g.)
If the application gets slower, stop the domain, restart the db and restart the domain afterwards.

JDBC-Realm

Configure the JDBC-Realm as follows:

Afterwards set the Realm as default:

Routing Graph

Copy a routingGraph file to %glassfish_home%/domains/domain<i>/routingGraph.hh or to %glassfish_home%/domains/domain<i>/config/routingGraph.hh   (glassfishes seem to be different)

OpenRide

Setting up the development environment

Netbeans:
Version 6.9.1

www.netbeans.org

Install the Netbeans IDE. Then connect your Netbeans IDE with your Glassfish domain. To do this click on the "server" tab on the left side an then make a right-click on "Servers". In the upcoming windows choose "Glassfish Server 3" and click next. Fill in the Glassfish home directory and click next. Register domain1 and finish the connection. That's it.

After checking out the root folder of the project (%OpenRide SVN%/src/OpenRideServer/OpenRideServer) create a new project as enterprise project with existing sources.
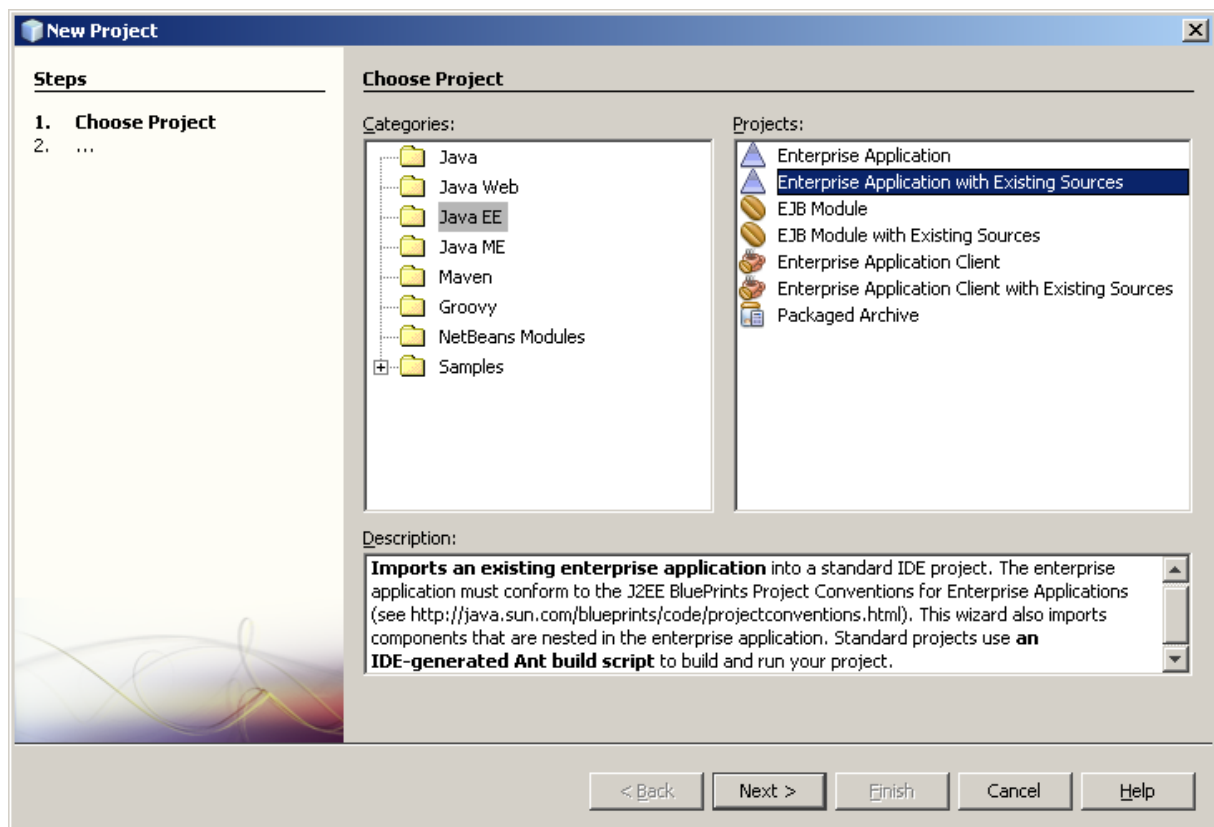
Choose the connected Glassfish Server as Application Container.

Then do a right click on the project and click „open required projects".

Required projects are: OpenRideServer-RS, OpenRideServerWeb, OpenRideAdmin.

Create another project with existing sources in the .../OpenRideServer/OpenRideServer-ejb directory. Select OpenRideServer for the Enterprise Application.
(If asked for configuration, source and test package folders, specify "config", "src", and "test" within OpenRideServer-ejb.)

Delete the setup/sun-resources.xml in the projects OpenRideServer-RS and OpenRideServer-ejb.

Add the OpenRideServer-ejb project as dependency to OpenRideServer-RS, OpenRideServerWeb, OpenRideAdmin (you may need to remove any existing references to the OpenRideServer-ejb project first, then add them again). Also add all libraries of the OpenRideServer-ejb/libs directory as
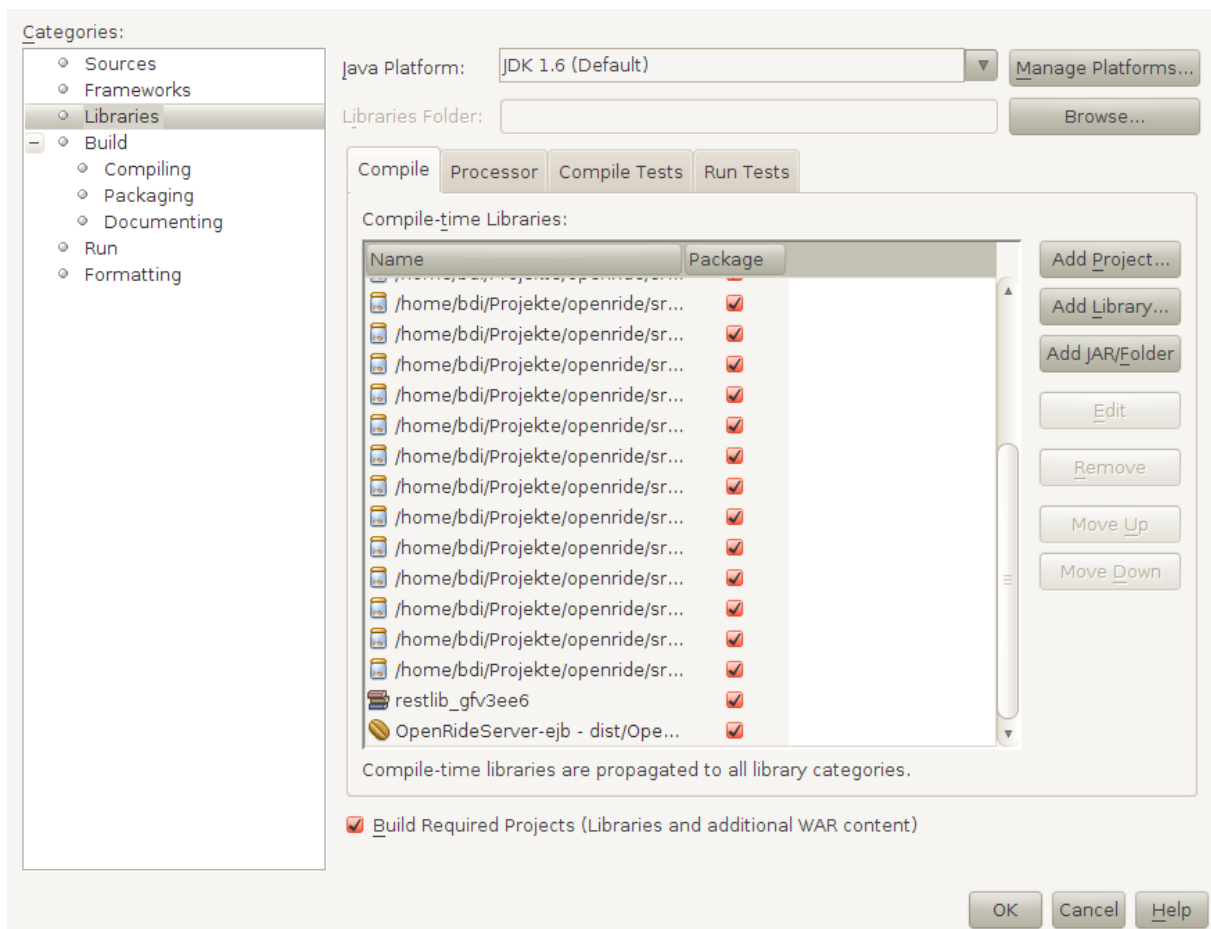
dependencies to OpenRideServer-RS, OpenRideServerWeb, OpenRideAdmin.

Testing frameworks and tools

JUnitEE

JUnitEE (http://www.junitee.org/) allows testing enterprise beans running in their container through a servlet interface. A short overview can also be found here: http://www.ibm.com/developerworks/rational/library/07/0102_woldemichael/

To add JUnitEE support conduct the following steps:

Add **junit.jar** to the EJB module. Add both **junit.jar** and **junitee.jar** to the Web Client module. The **web.xml** file in the WEB-INF folder already contains the **JUnitEE TestServlet**. Please note that there seems so far to be no support for JUnit 4.x in JUnitEE, only JUnit 3.x is supported.

The TestServlet is now locally accessable through the URL:

http://localhost:8080/OpenRideServer-war/TestServletEE

Type the name of a JUnit compliant test class (e.g. *de.fhg.fokus.openride.rating.junittests.RatingBeanTest*) and hit "Run". The result of the test is listed for each of the test methods in the test class.

In the EJB module JUnit tests can be generated automatically like this: In the Properties dialog / Sources add the location of a test package folder (usually *test*). You should select JUnit 3.x because 4.x is not supported by JUnitEE so far. Now right click on a source code package or a class -> choose Tools -> choose Create JUnit Tests. Unfortunately the generated code will not be packaged into the ejb jar (or is it possible and how?) and will not be accessible to the test servlet. Therefore it is necessary to manually copy the generated code into the source folder of your project. The test code should be placed into the appropriate package extended with .junittests, e.g. *de.fhg.fokus.openride.rating.junittests.RatingBeanTest*.

JUnitEE how to test EJB's + DB

The whole project consists of the following projects.

-OpenRideServer+

       |
      -> OpenRideServer-RS
    |
    -> OpenRideServer-ejb
    |
    -> OpenRideWeb
    |
    -> OpenRideAdmin
    |
    ->OpenRideTest-war

The WebProject: OpenRideTest-war contains all needed configuration files and is structured so that one can add new JUnit-Tests or JUnit-Testsuites via the following steps using Netbeans 6.8:

1. right-click on the package/class in the OPENRIDESERVER-EJB project you want to test. Select Tools->Create JUnit Tests.

2. The generated Tests can be found in a directory of the OPENRIDESERVER-EJB PROJECT named TEST.

3. One can edit(!!) the files now.

4. Copy the test(s) to the OPENRIDETEST-WAR/SOURCE-PACKAGE.

5. Now you have to add the packagename+classname of the new test class to the so called testCases.txt, which is located at OPENRIDETEST-WAR/WEB-PAGES/WEB-INF/TESTCASES.TXT.

6. If you now deploy the application a Test-Servlet can be found under the following URL: <host>:<port>/OpenRideTest-war/TestServletEE. This is the place where you can start the tests.

DBUnit

DBUnit is a JUnit extension targeted at database-driven projects that, among other things, puts your database into a known state between test runs: http://www.dbunit.org/howto.html

Libs necessary: dbunit.jar, slf4j-api-*.jar, slf4j-nop-*.jar

To run DBUnit inside the same Application container as the EJBS, DBUnit tests can be run via OpenRideTest-war/TestServletEE as described in the JUnitEE section. Like the JUnitEE Testcases, DbUnit-Testcases should be located in OpenRideTest-war project.

In many (Test)Cases it is required to put the Database to a known initial state. This can be done e.g. via import of an xml file holding the initial state. These Xmls can be coded by hand or be exported a database. For exporting, use the class 'dbunit. DatasetExporter'. I suggest locating all xml exports required by TestCases in the subtree of 'OpenRideTest-war/res'. The project will be configured to include the files into the deployable war-file.

Setting up the Project:

1. Create a new Database named 'openride_test' and import an actual dump from svn

2. Add Connection-Pool and Jdbc Resource to Glassfish using 'jdbc/openride_test' as jndi-name

3. Configure 'OpenRideTest-ejb' to use the Resource 'jdbc/openride_test' instead of 'jdbc/openride'

4. Setup packaging
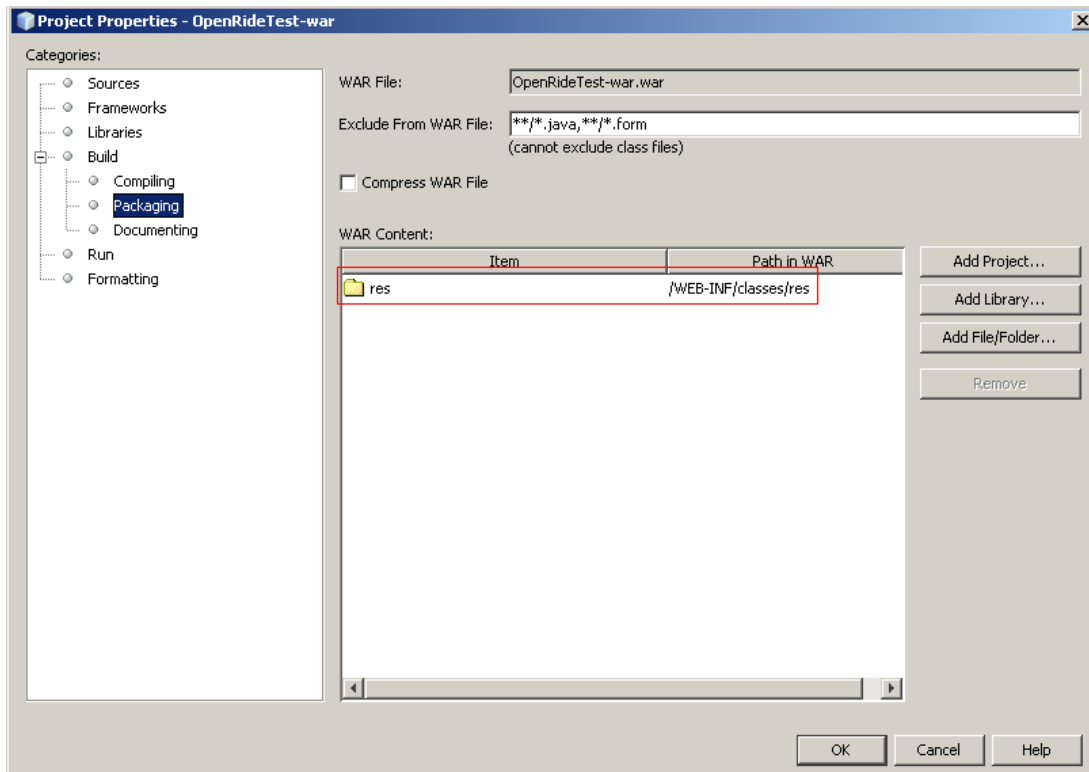
Implementing a DBJunit Test Case :

1. If steps are not clear, check : 'dbunit.sample. CustomerControllerTest' or '*de.fhg.fokus.openride.rating.junittests.RatingBeanDBTest*'

2. Extend 'dbunit.AbstractOpenRideDbUnitTestCase'

   a. or use a subclass of DatabaseTestCase e.g. JndiBasedDBTestCase

   b. or subclass DatabaseTestCase directly

3. To implement the Method getDataSet() (represents the initial db state)

   a. Create a XML by hand or by using the exporter class: 'dbunit. DatasetExporter'

   b. Copy the XML to a subdir of 'OpenRideTest-war/res'

   c. Return a FlatXmlDataset using the respective xml file.

4. Add the class to 'TestCases.txt' located at WEB-INF

5. Implement DBUnit related Methods e.g. 'getSetUpOperation()'

6. Implement test Methods analog to JUnit test cases

7. Run Test case via OpenRideTest-war/TestServletEE

TODO:

- How to export / import non standard sql datatypes like postgis geometries?

soapUI – Netbeans integration

Install the Netbeans soapUI plugin as drescibed here:

http://www.soapui.org/netbeans/installation.html

(Downloadable from http://sourceforge.net/projects/soapui/files/soapui-netbeans-plugin/3.5/com-eviware-soapui-netbeans-module-3.5.nbm/download)

Once the plugin has been installed, the project „OpenRideServer-RS-Testing" will be recognized and may be opened as a soapUI project.

RESTful Webservices

Open the OpenRideServer-RS project and Netbeans will ask you how the RESTful Services shall be configured. You should choose the first alternative: Netbeans will generate a subclass from "…" . Afterwards you can deploy the project and the webservices will be available.

More informations on the type of services can be found in the SVN in the document Actionnames.pdf

Android Client Installation

- eclipse installieren

- android sdk unter

http://developer.android.com/sdk/download.html?v=android-sdk_r05-windows.zip

runterladen und in einen beliebigen ordner entpacken.

Im entpackten ordner die datei „SDK setup.exe" ausführen, es erfolgen dann einige updates.

- Dann eclipse öffnen(e.g. Galileo version), unter windows->preferences->android den pfad zum sdk eingeben.

- eclipse-plugin: in eclipse unter help->install new software->add einen beliebigen namen und die url https://dl-ssl.google.com/android/eclipse/ eingeben, installation des plugins abschließen.

- treiberinstallation htc hero:

das programm htcsynch unter http://www.htc.com/de/SupportDownload.aspx?p_id=283&cat=3&dl_id=852 downloaden und die exe installieren → reboot. Falls da bei der installation ein fehler kommt bezgl xp-treiber nicht gefunden, keine ahnung, worans liegt, einfach ignorieren. Danach (htc sollte sich nach dem neustart selbst gestartet haben, sonst htc starten) mal ein gerät per usb anschließen. Falls htc das gerät nicht erkennt oder die windows-treiber-installation kommt, für den windows installationsdialog den pfad zum ordner „usb_driver" im hauptordner des android sdk angeben, htc sollte das gerät nun erkennen.

Im projekt im ordner res->values in der datei strings.xml den key (tag debug_apikey) durch den eigenen ersetzen. wenn man ihn nicht mehr hat: http://code.google.com/intl/de-DE/android/add-ons/google-apis/mapkey.html.

wenn man beim offenen eclipse nun ein gerät anschließt und dann auf rechtsklick->run as->run configurations (hier im tab "target" auf radiobutton "manual" klicken)->run geht, sollte ein dialog kommen, wo man das gerät sieht, dann ok und er sollte es auf dem gerät installieren und starten.

Help with Android devices

HTC HERO rooting, backup and costom ROMs

This chapter is for information on how to root a HTC HERO.

First of all, read this: http://forum.xda-developers.com/showthread.php?t=622228

Then download a custom ROM of your choice: http://forum.xda-developers.com/forumdisplay.php?f=512

→ Follow the installation instructions of the custom ROM

Kundendeployment

Standortbezogene Default Favoriten

Für jeden Kunden sollen standordbezogene Favoriten vorhanden sein, die jedem Benutzer direkt nach der Anmeldung zur Verfügung stehen. Hierzu gibt es für jedes Kundensystem einen Template-Benutzer. Für diesen Benutzer werden vor dem Rollout die entsprechenden Favoriten eingetragen. Anschließend bekommt jeder neue Benutzer bei seiner Registrierung automatisch auch alle Favoriten des Template-Benutzers. Die Favoriten des Template-Benutzers können im nachhinein geändert werden. Diese Änderungen gelten dann für alle neuen Benutzer, welche sich nach der Änderung registrieren. Der Template-Benutzer sollte als erster Nutzer mit folgenden Daten registriert werden.

Logindaten des Template-Benutzers:

- Name: template_user

- Passwort: $or_template_user1

Siehe auch Dokument Customization.