Installation Guide of the OSS Common API Reference Implementation

OSS through Java™ Initiative

Vincent Perrot, Sun Microsystem

COM-API-Ri_Installation_guide.1.5.doc

Executive Summary

The Common API offers interfaces and classes, which are common across all OSS APIs defined under OSS J Initiative. This document describes how to install and use the Reference Implementation (RI).

Table of Contents

Preface

Objectives

Installation and usage description of the OSS/J Common Reference Implementation

Audience

The target audiences are

- Developers who seek information about how the Common API can be implemented
- Developers of other OSS/J API Reference Implementations
- Developers who want to make use of these API and extend its implementation

Related Information

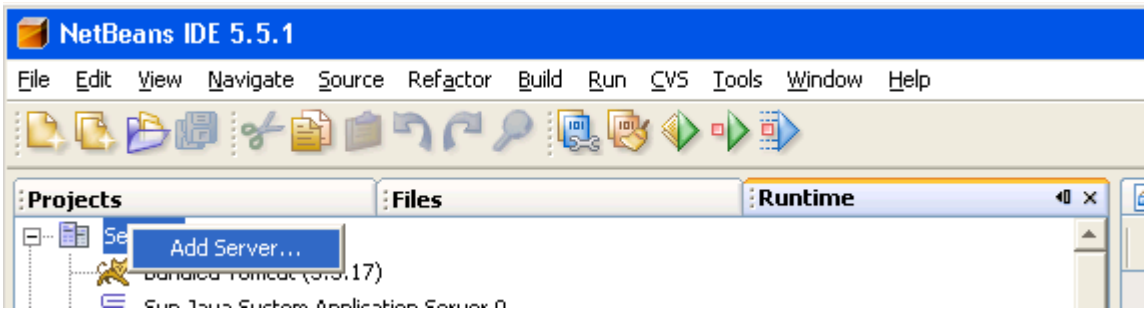
Prerequisite

NetBeans IDE 5.5.1, and glassfish v2 b50. You can download it at:

http://NetBeans.org

http://glassfish.dev.java.net

Once NetBeans and Glassfish install properly, start NetBeans go to “Runtime” and add the Glassfish application Server:



And follow instructions.

Revision History:

Date	Version	Author	State	Comments

September 2006	1.4	Vincent Perrot Sun Microsystems Inc	Maintenance Release 4	Change all the content according to the new RI design
May 2007	1.5	Vincent Perrot Sun Microsystems Inc	Maintenance Release 5	

1 Introduction

This document describes how to install, run and verify the OSS Common API Reference implementation (RI).

The RI is divided into two main parts:

- The extendable implementation of CBE and based objects
- The example demonstrating the usage and extension of the API for the three integration profiles and especially Web Services one.
- A Web Service Client example

In all the following

- the given paths are relative to the **<installation path>\ oss_common_javaeesdk-1_5-src-ri**.
- the operating system is Windows XP
- the application server bin directory is in the execution path of the environment

Note: screenshots are provided as example, version number (like “1_5” for example may be different)

2 Extendable part

This part of the RI is designed to be reused as much as possible into the other OSS through Java (OSS/J) APIs.

In the src directory you will find the java code of the classes and a build .xml file allowing compiling and creating the java archive of the Common API implementation.

This part of the RI requires the JavaEE 5 to be installed. Glassfish v2 b50 is recommended.

The common java archives of the RI are already delivered compiled into the lib directory (one jar per package name).

In order to re-compile and re-create the jar the following command could be executed:

```
> cd src\ri
> asant.bat
```

The newly recompiled archives are placed in the lib directory.

Note: “asant” is located under the bin directory of the application server installation directory.

3 Examples demonstrating the 3 integration profiles

The part of the RI is delivered as NetBeans projects starting with "common_ex" prefix.

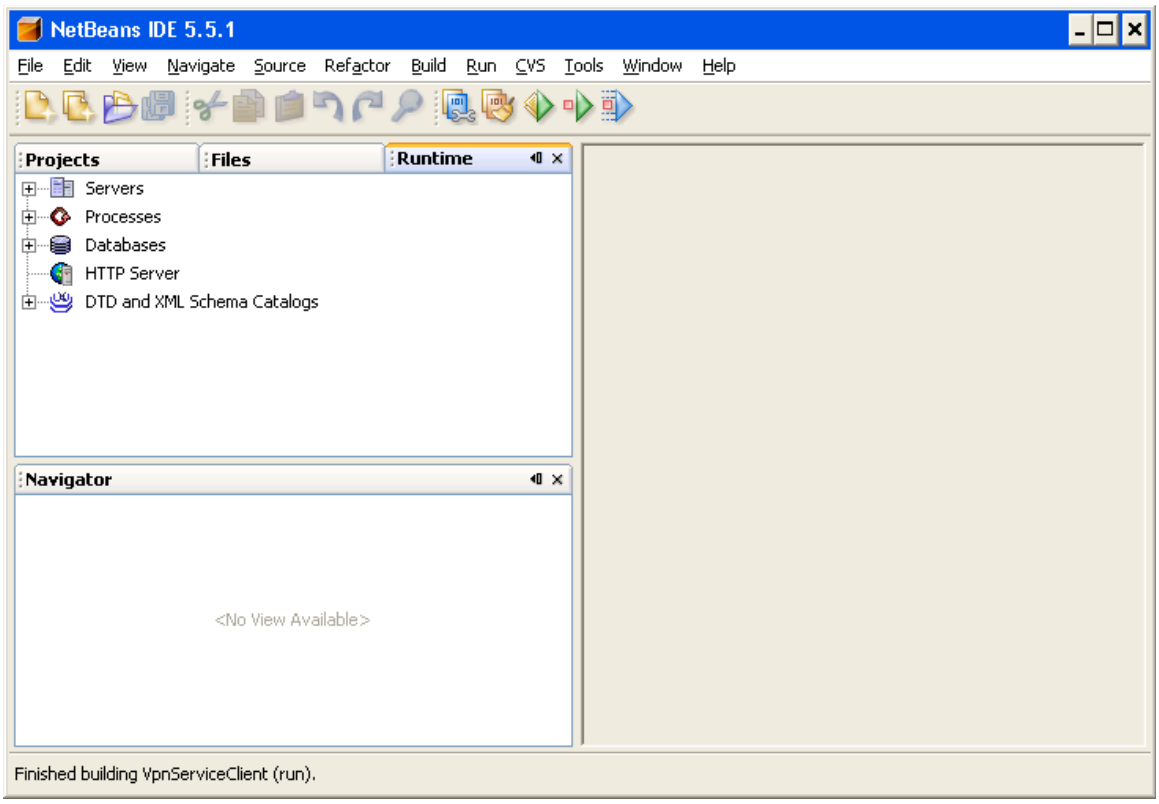
There are 5 projects (or modules):

- common_ex contains the deployable application (.ear) composed of 3 EJB modules (using the NetBeans vocabulary) below:
- common_ex_EJB contains the Java and XML profiles composed of several EJB,
- common_ex_WS contains the Web Services (WS) implementation representing the Common WS integration profile. It also demonstrates the WSDL->java tools.
- common_ex_vpn_WS contains the WS implementation representing a simplified vpn service extending the Common interface. It also demonstrate the java-> WS tools for Web Service creation
- VpnServiceClient contains an example of Web Service Client interfacing with the VpnService Web Service.

3.1 Installation instructions

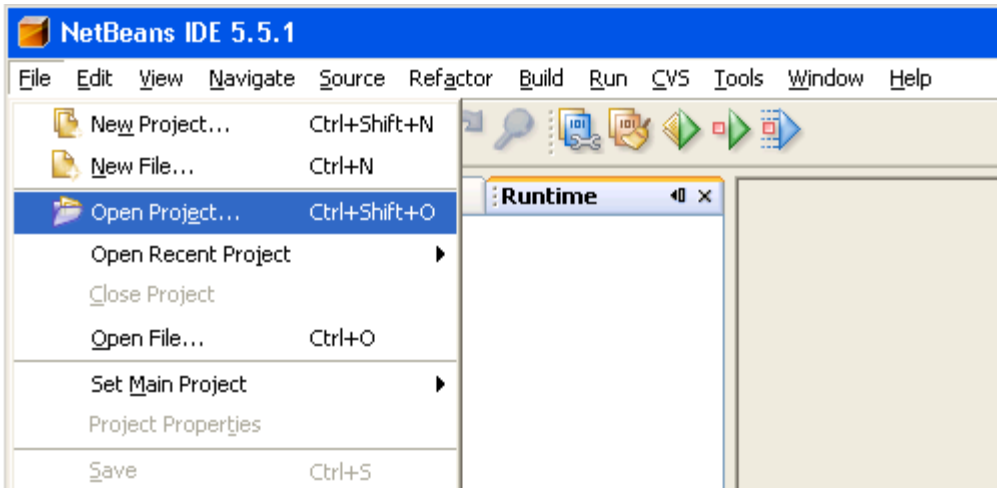
All the installation steps are performed using NetBeans.

Start NetBeans, the following window appears:

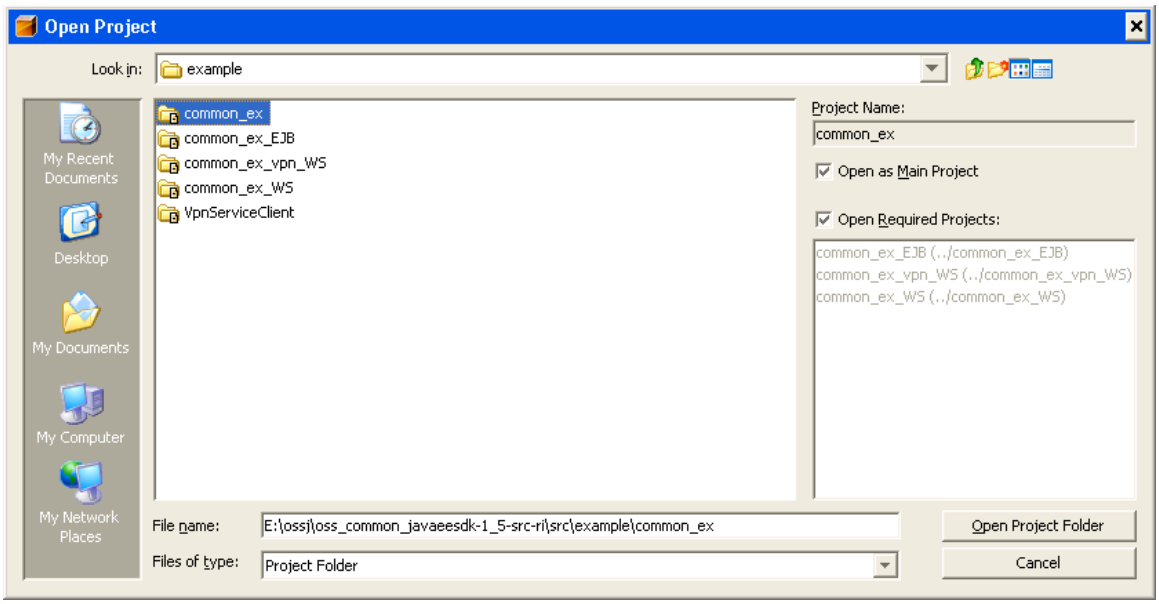


3.1.1 Load the project

Open all the common_ex (and Required Projects) project from the src\example directory



In the wizard window navigate until the example directory and selection the 4 projects:



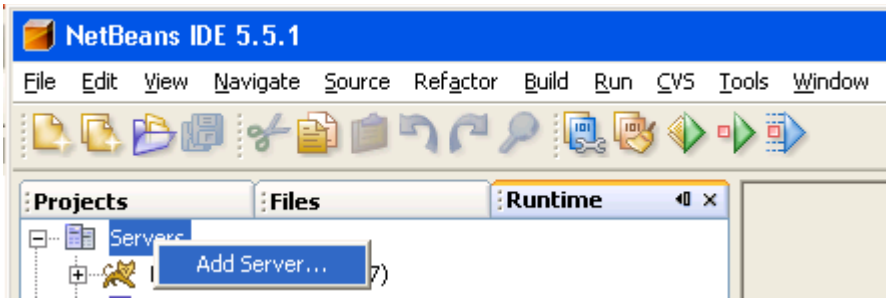
Solve any reference when missing by browsing the existing loaded projects or libraries. NetBeans will give you instructions.

3.1.2 Configure the application server

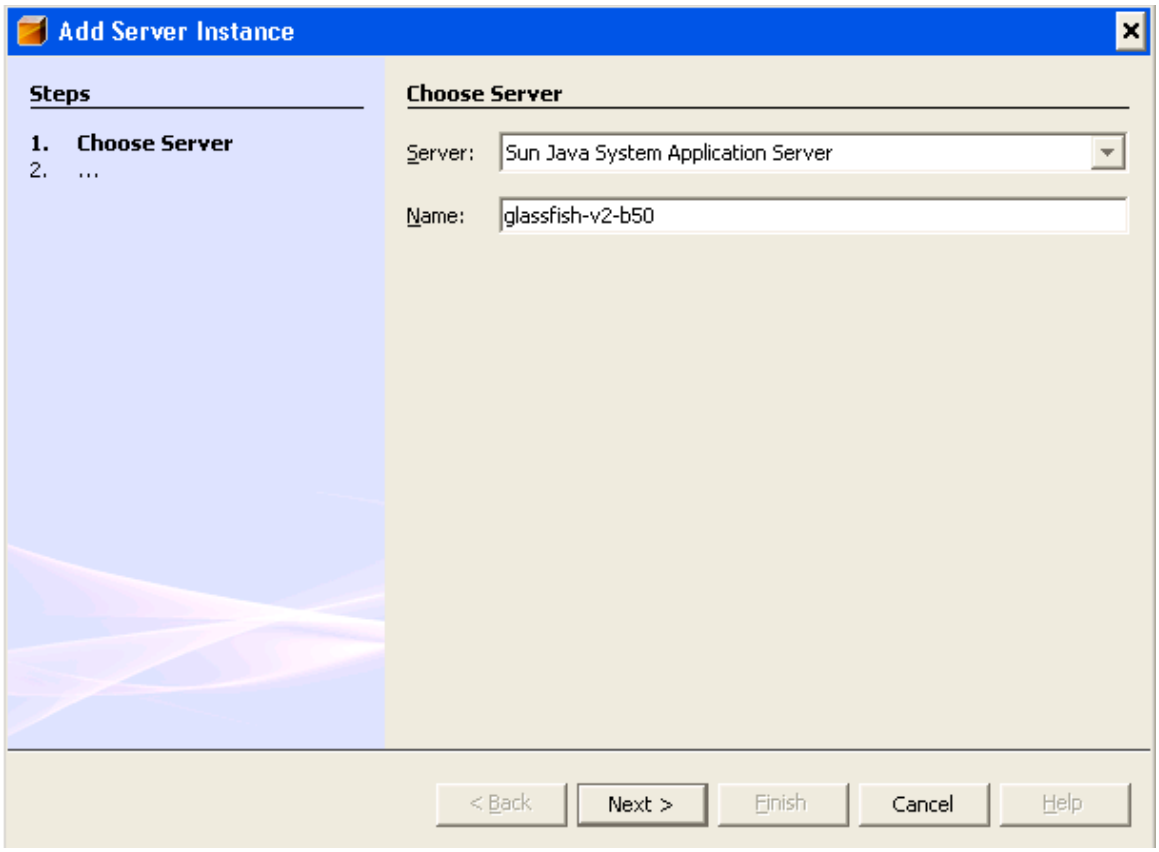
The first step is to add the GlassFish application server to NetBeans Runtime environment.

Glassfish installation instruction needs to be completed before executing the following:

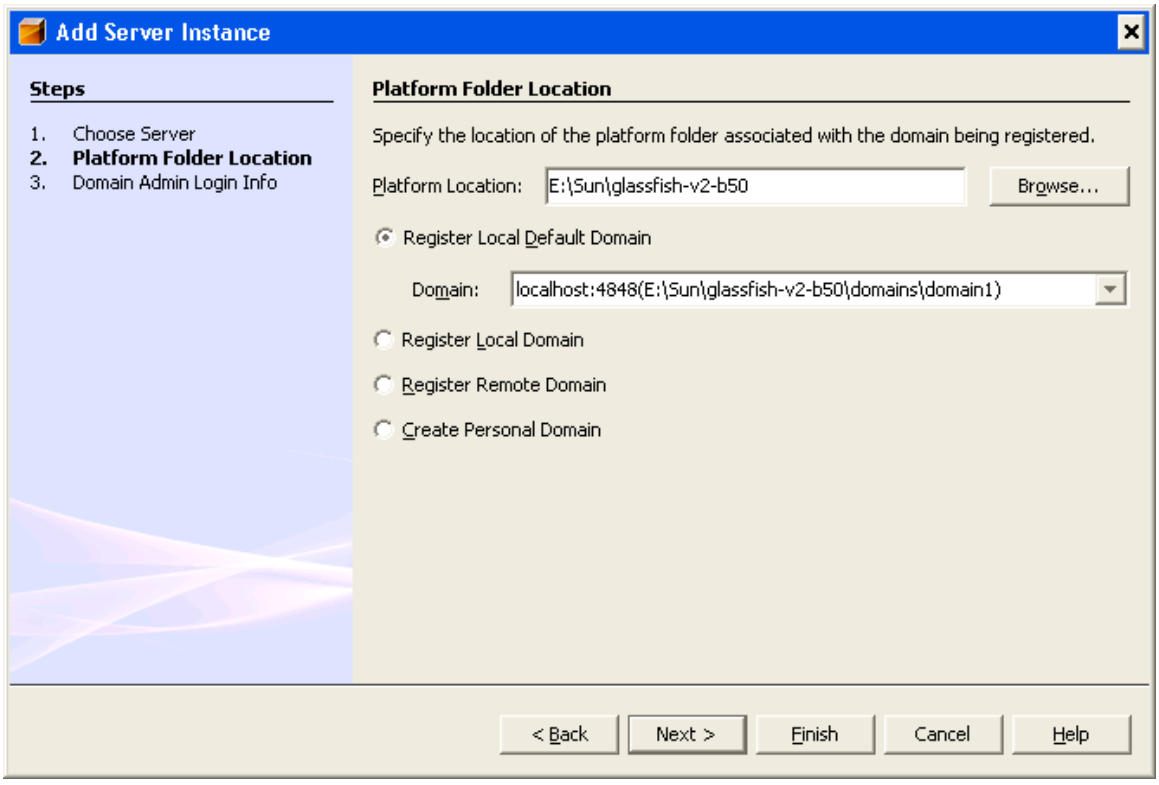
Go to the “Run Time section and add the Server:



Complete the wizard as follow:



And then

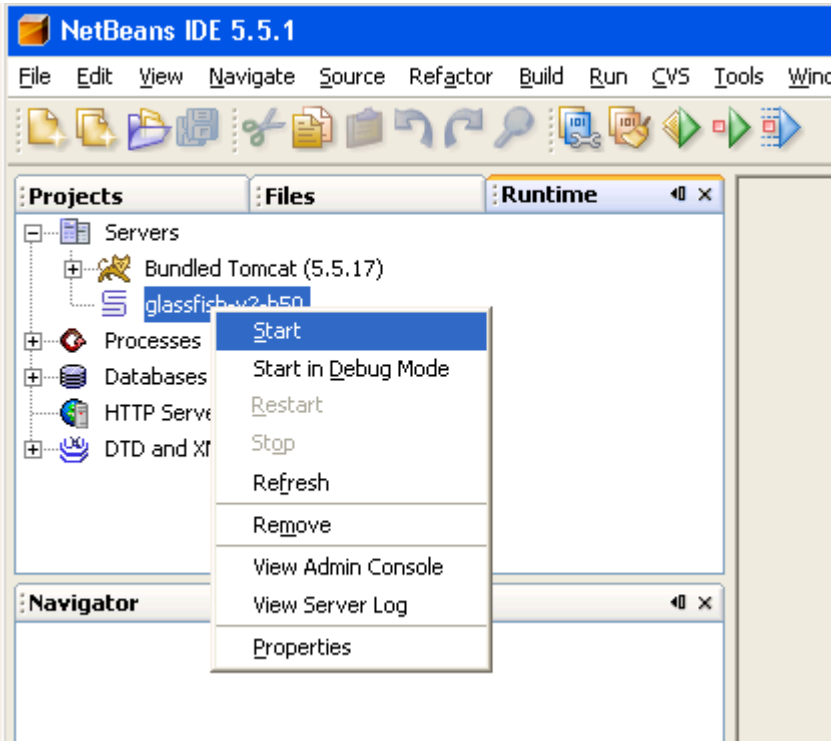


Finally, in last wizard use the defaults.

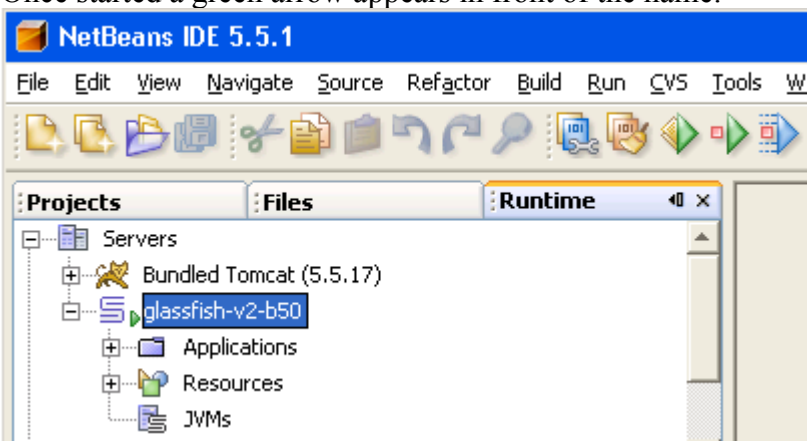
Start the application server:

Note:

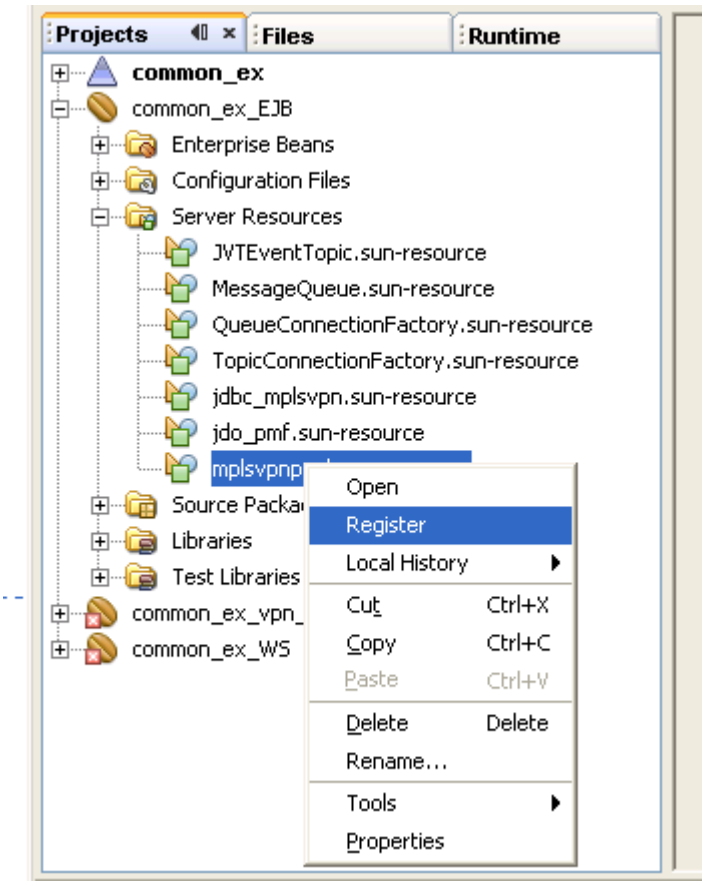
Be sure that your firewall is correctly configured to allow Java virtual machine, NetBeans, etc to access the network.



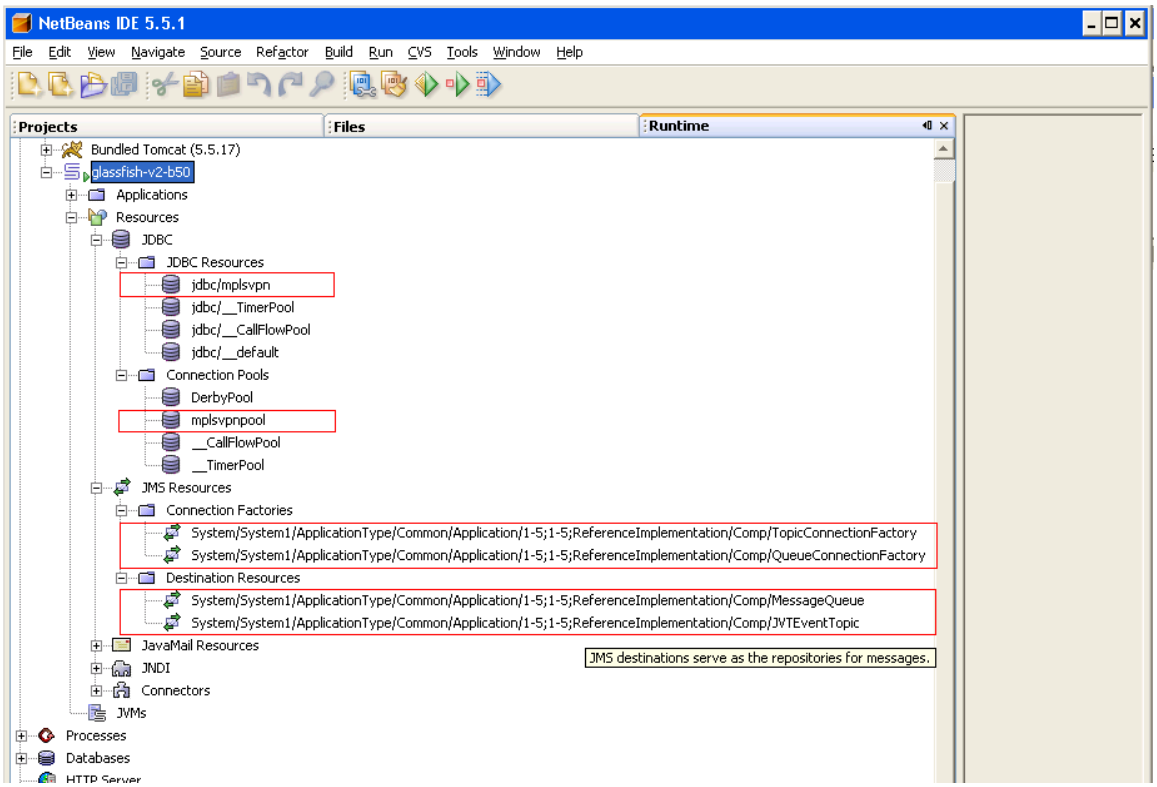
Once started a green arrow appears in front of the name:



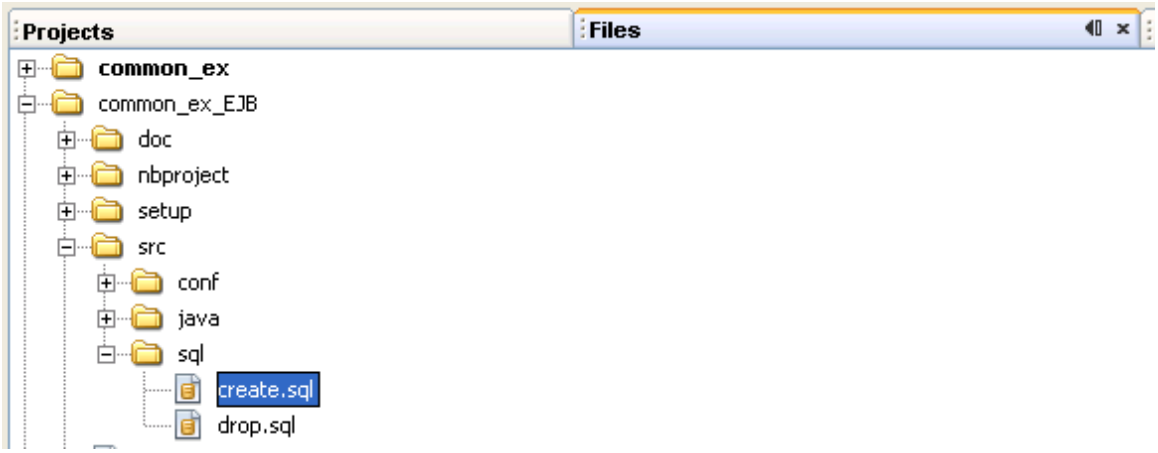
Open the common_ex_EJB project and “register” the resources starting with the mplsvpnpool, jdbc_mplsvpn first and then the others moving up (note: JMS factories need to be registered before the queue or topic):



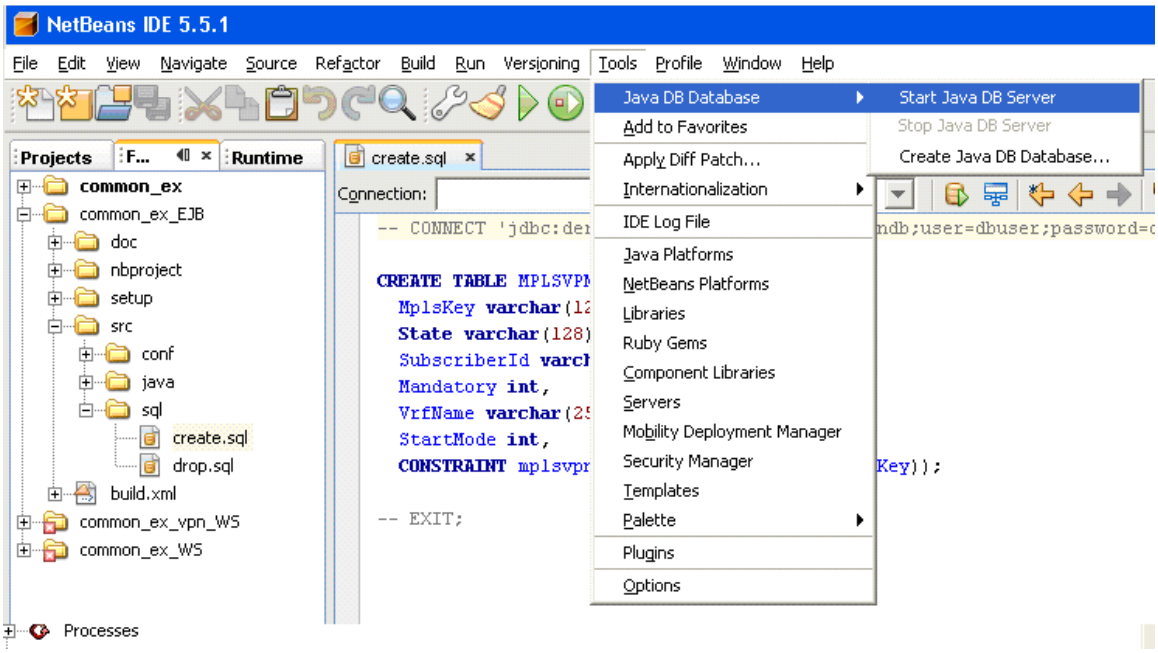
Once all the resources registered, verify the configuration into the “Runtime” tab:



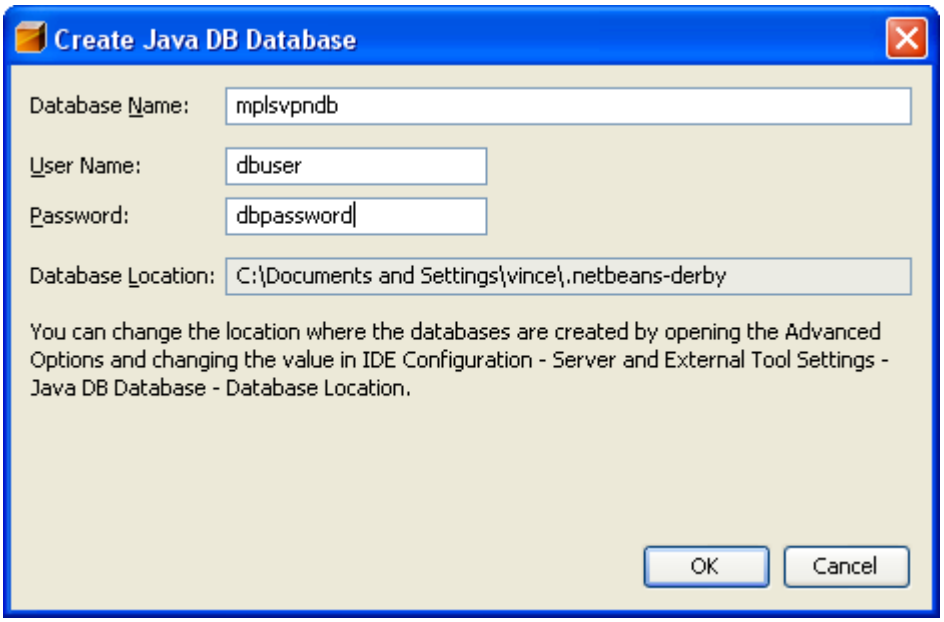
Open and create the table into the database, all needed information is located in the create.sql file of the common_ex_EJB project:



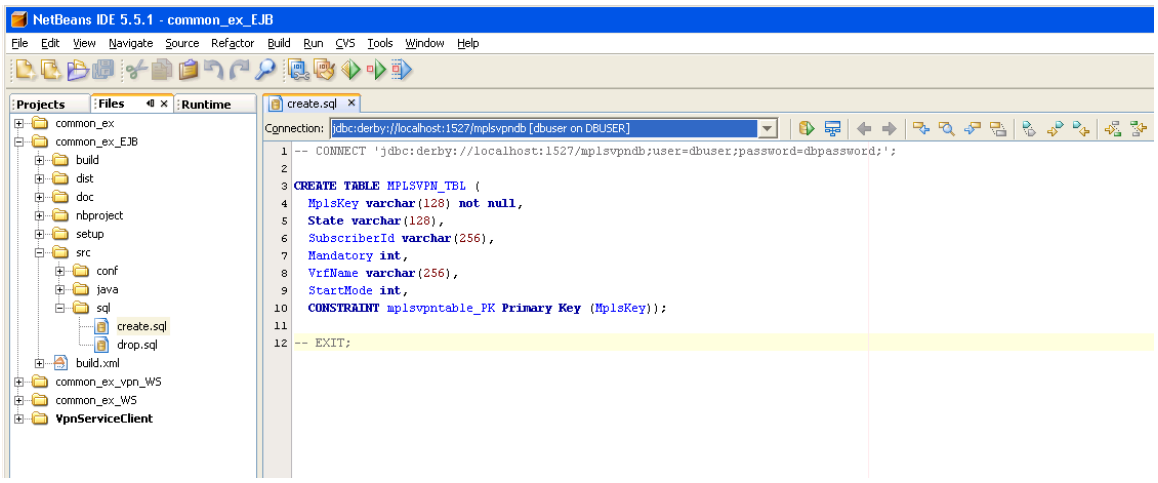
Create a new DB:



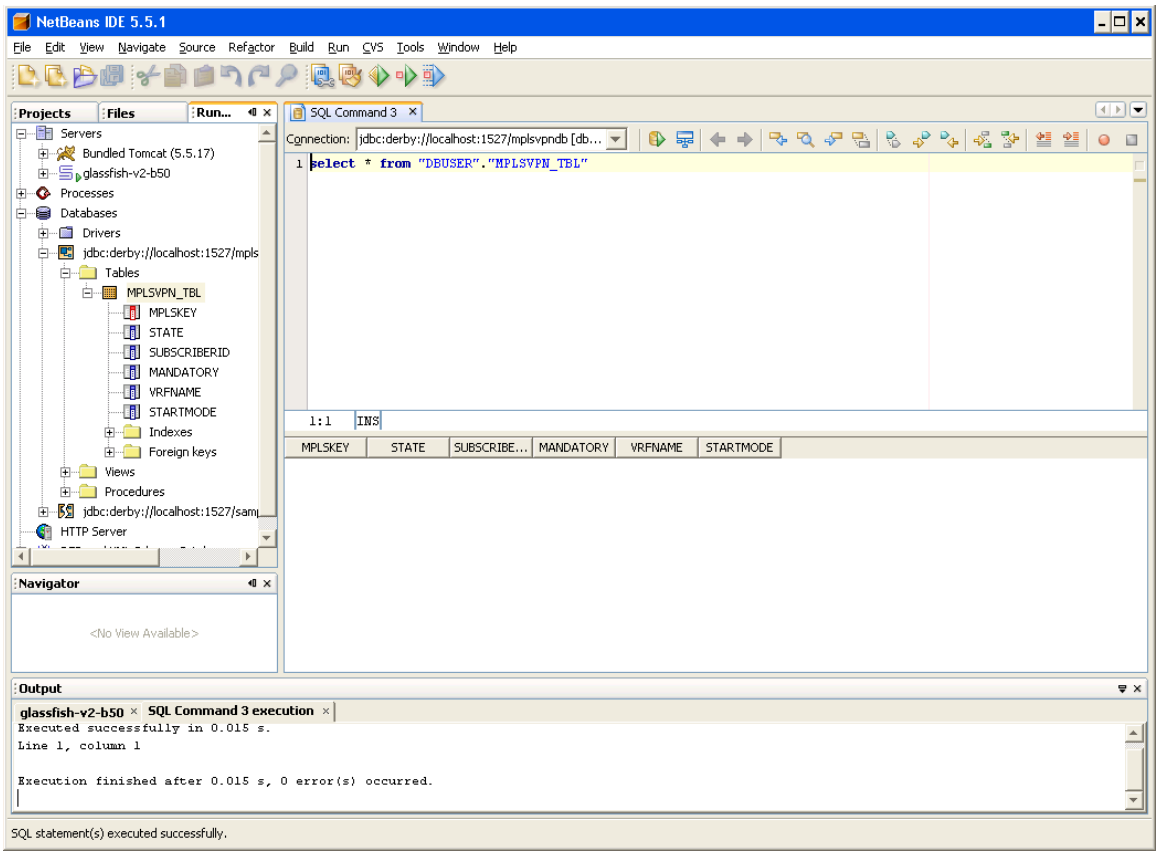
From the wizard:



Once create execute the create.sql from the edition windows once the correct DB selected:



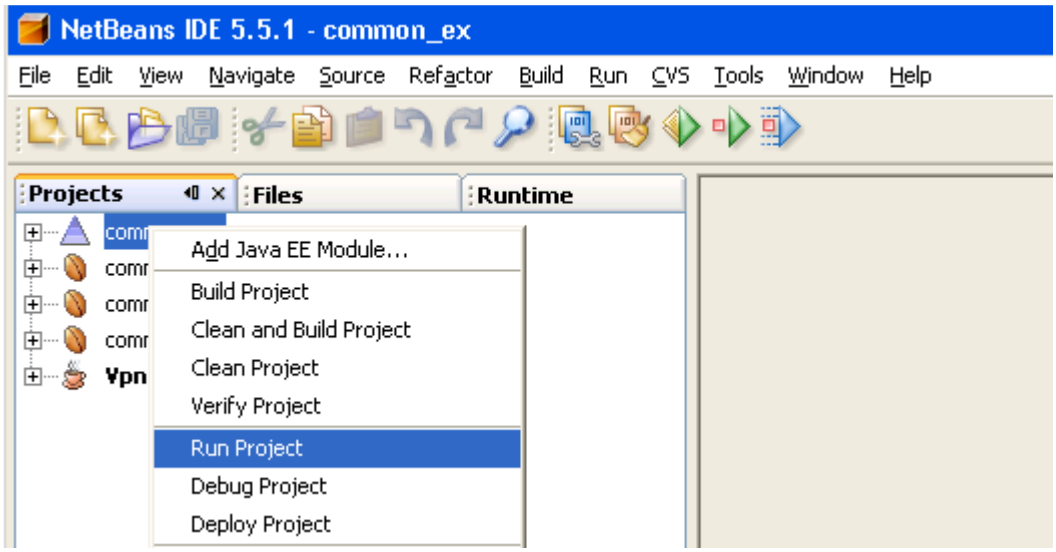
Verify the Table creation from the Runtime tab:



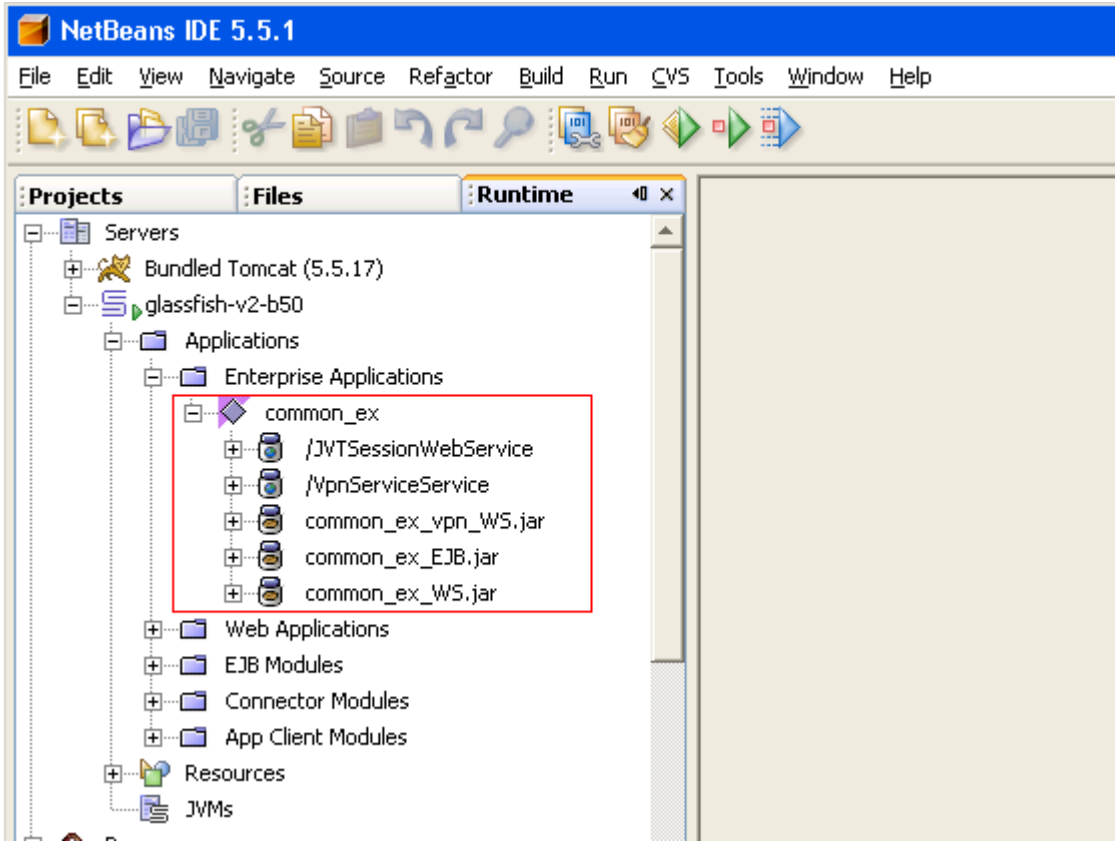
Note: If the table doesn't appear, "disconnect" and re-"connect" from the DB.

3.1.3 Deploy the application

From the "Projects" tab, right-click on the common_ex project and select the "Run Project" item:



Verify the correct deployment from the "Runtime" tab:



The five lines in the red box indicate that the 2 WS and the 3 jars have been correctly deployed (In case use "refresh" to actualize the representation).

(If necessary, the traces of the Application server could be checked at the bottom of the NetBeans window)

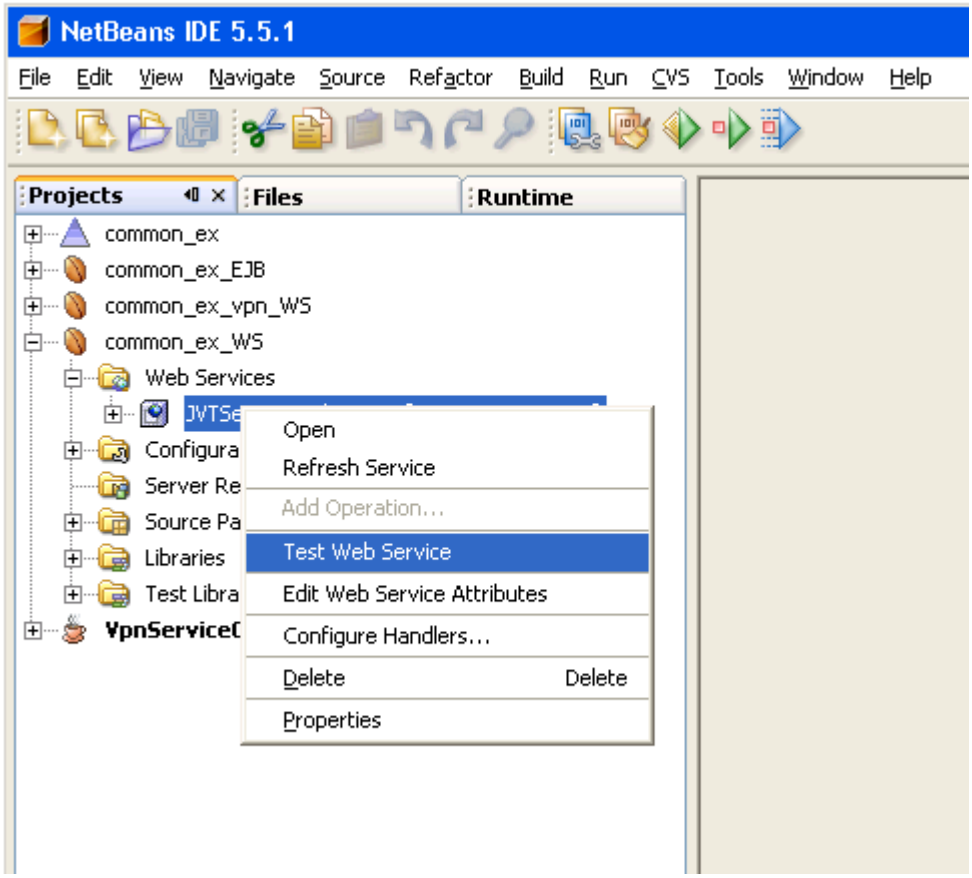
The application is now ready and could be used.

3.1.4 Exercise the Web Services profile

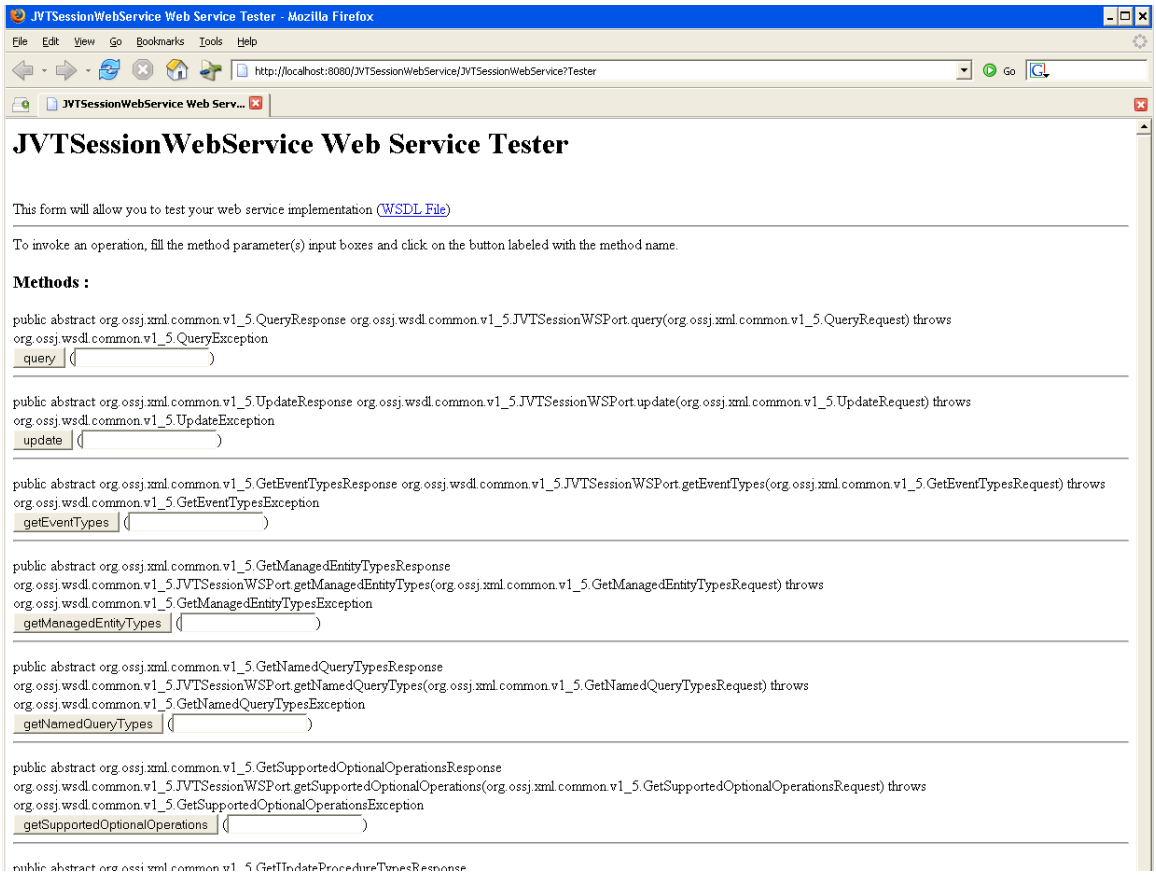
3.1.4.1 Common WS profile

The Common WS profile could be tested from your favorite Web browser.

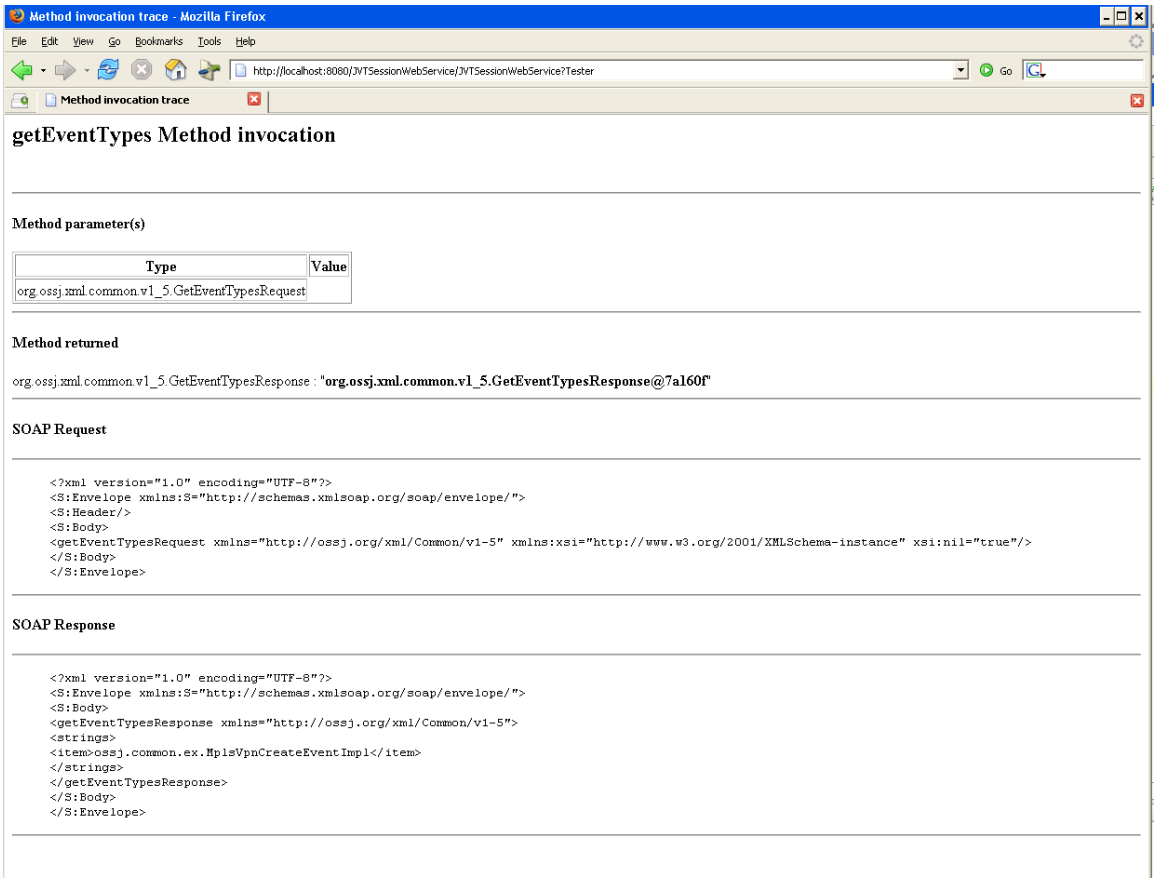
In the common_ex_WS Project section, right-click on the WS name and select "Test Web Service":



Your favorite web browser will pops up the web services page from where you can exercise one of the OSS Common API features:



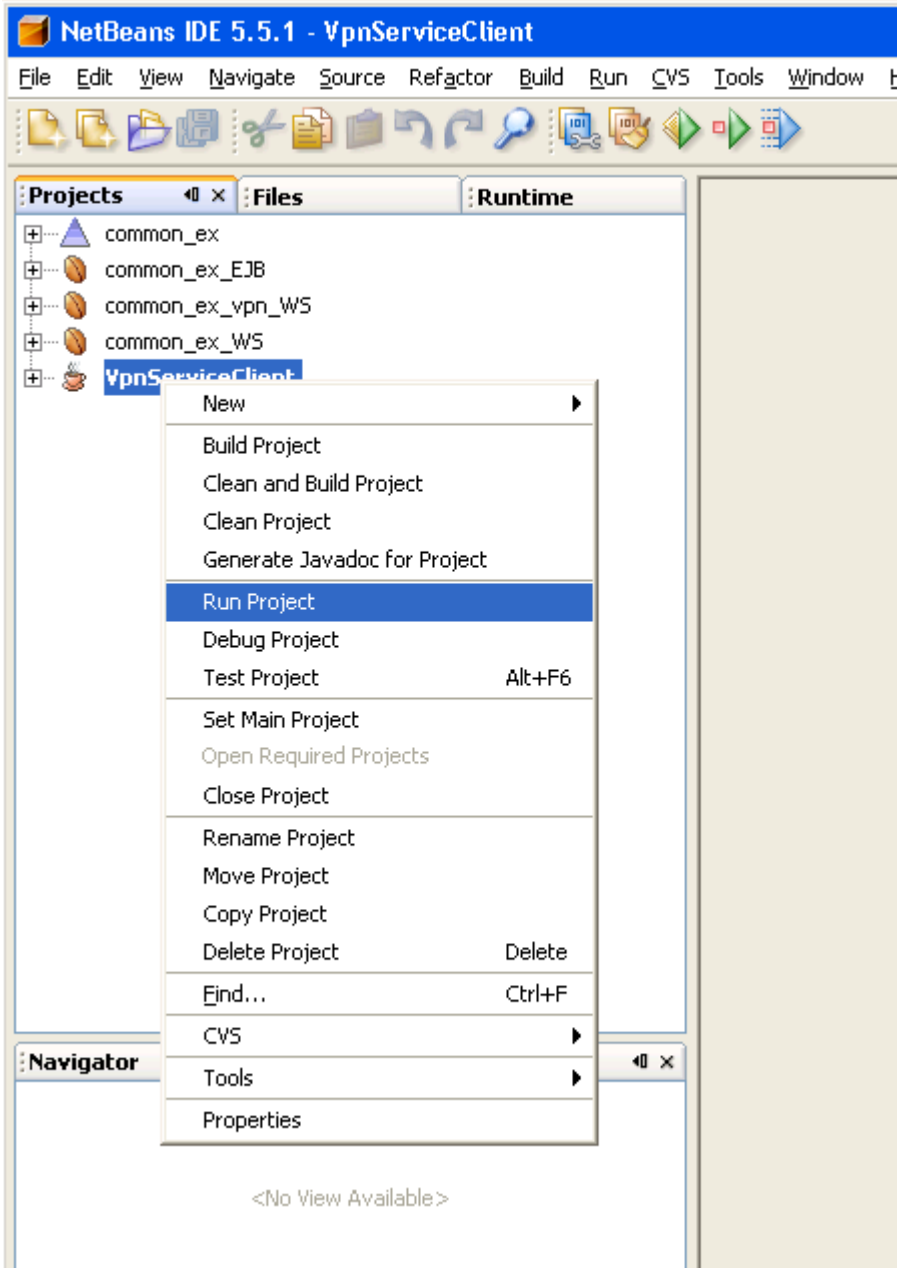
For example, pushing the “getEventTypes” button will return:



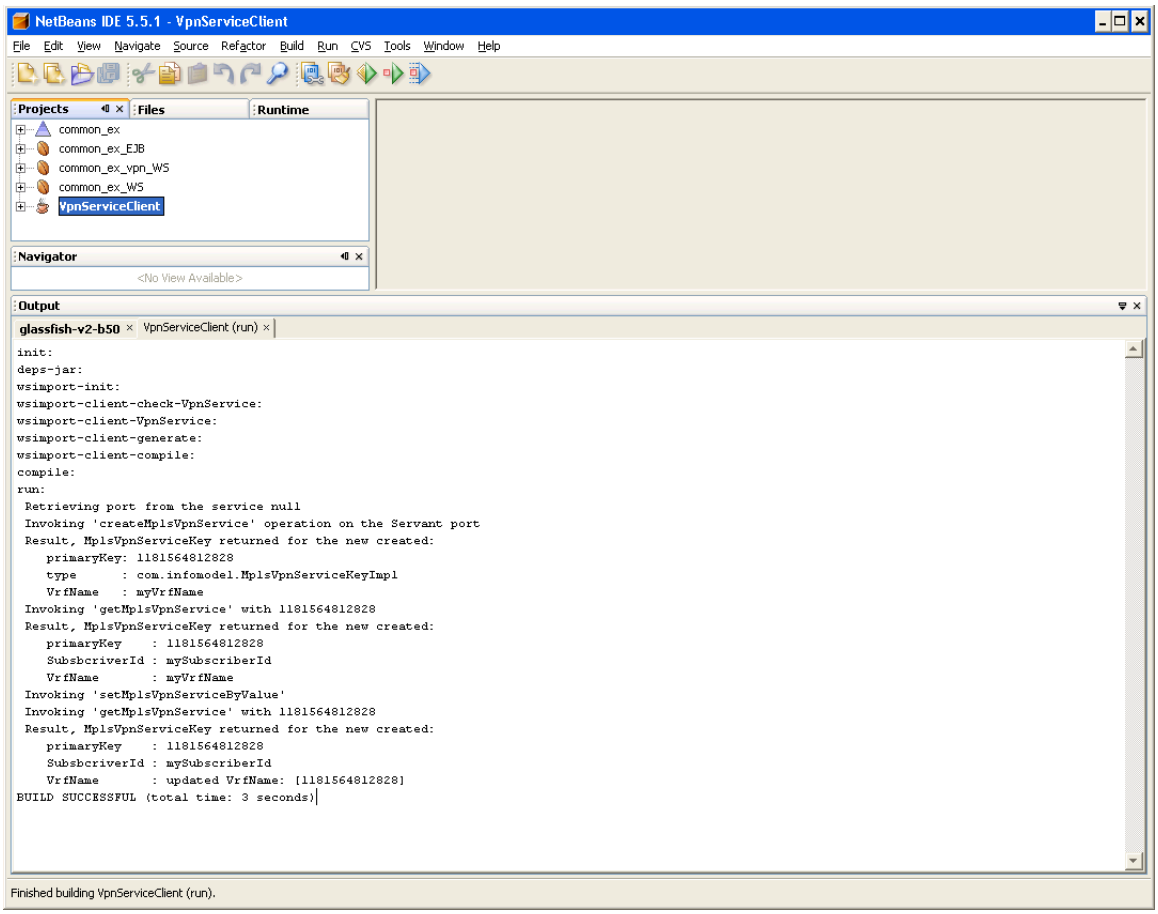
3.1.4.2 VPN Web Service (common_ex_vpn_WS project)

The VPN web service could be exercised using the same method as previously or using the provided client example. The small application benefits from the NetBeans environment to be executed.

In order to compile and execute the client, open the VpnServiceClient project and “Run Project”.



At the bottom of Netbeans windows you will get the execution results as follow:


NetBeans IDE 5.5.1 - VpnServiceClient
File Edit View Navigate Source Refactor Build Run CVS Tools Window Help
Projects
Files
Runtime
common_ex
common_ex_EJB
common_ex_vpn_WS
common_ex_WS
VpnServiceClient
Navigator
<No View Available>
Output
glassfish-v2-b50
VpnServiceClient (run)
init:
deps-jar:
wsimport-init:
wsimport-client-check-VpnService:
wsimport-client-VpnService:
wsimport-client-generate:
wsimport-client-compile:
compile:
run:
Retrieving port from the service null
Invoking 'createMplsVpnService' operation on the Servant port
Result, MplsVpnServiceKey returned for the new created:
primaryKey: 1181564812828
type : com.infomodel.MplsVpnServiceKeyImpl
VrfName : myVrfName
Invoking 'getMplsVpnService' with 1181564812828
Result, MplsVpnServiceKey returned for the new created:
primaryKey : 1181564812828
SubsbcriverId : mySubscriberId
VrfName : myVrfName
Invoking 'setMplsVpnServiceByValue'
Invoking 'getMplsVpnService' with 1181564812828
Result, MplsVpnServiceKey returned for the new created:
primaryKey : 1181564812828
SubsbcriverId : mySubscriberId
VrfName : updated VrfName: [1181564812828]
BUILD SUCCESSFUL (total time: 3 seconds)
Finished building VpnServiceClient (run).

Appendix A: Glossary and References

References