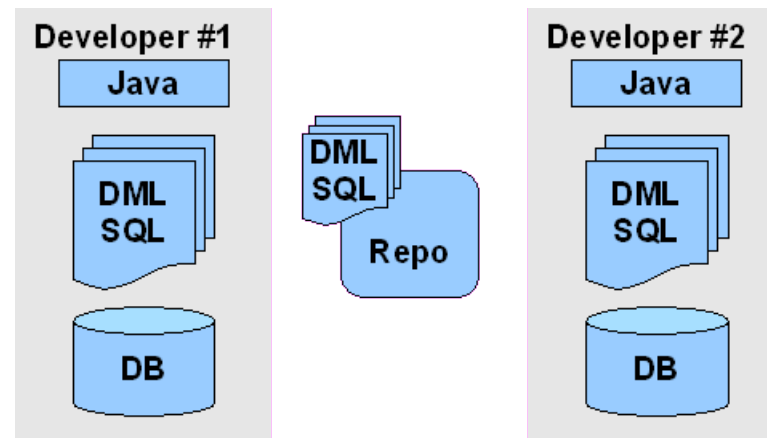


# GarinDriver with NetBeans

# Overview

This example shows how you can use GarinDriver with the NetBeans IDE to configure a project to share schema between two database instances via SVN.



Using this model each developer gets their own independent instance of the complete system database. Script for DML/SQL changes are automatically recorded by the JDBC driver and are shared between the developer workstations via SVN or a similar code repository.

# Step 1: Install

You can obtain the GarinDriver package from the GarinDriver code site.

<http://code.google.com/p/garindriver/>

The binaries for the driver is released as a single zip file.

Unzip the file to a location on your machine in order to use the driver.

If you have cygwin or are working on a unix machine, the following console commands will be helpful.

```
$ cd /opt  
$ wget http://garindriver.googlecode.com/svn/tags/v1.15/GarinDriver/garinDriver.zip  
$ unzip garinDriver.zip
```

# Step 2: Create DB

You can use any DBMS that the GarinDriver supports. MySQL, Oracle, HSQLDB, and other databases are supported.

This demo is based on PostgreSQL but everything shown here will work with any of the supported database systems.

Since each developer in for the system has their own private copy of the database everyone is free to make schema changes as needed to support development tasks. The DML and SQL files associated with these changes are stored in the code repository and automatically applied on target systems.

To create the database for this demo, use the following command any of the supported database systems.

```
CREATE DATABASE hello1;
```

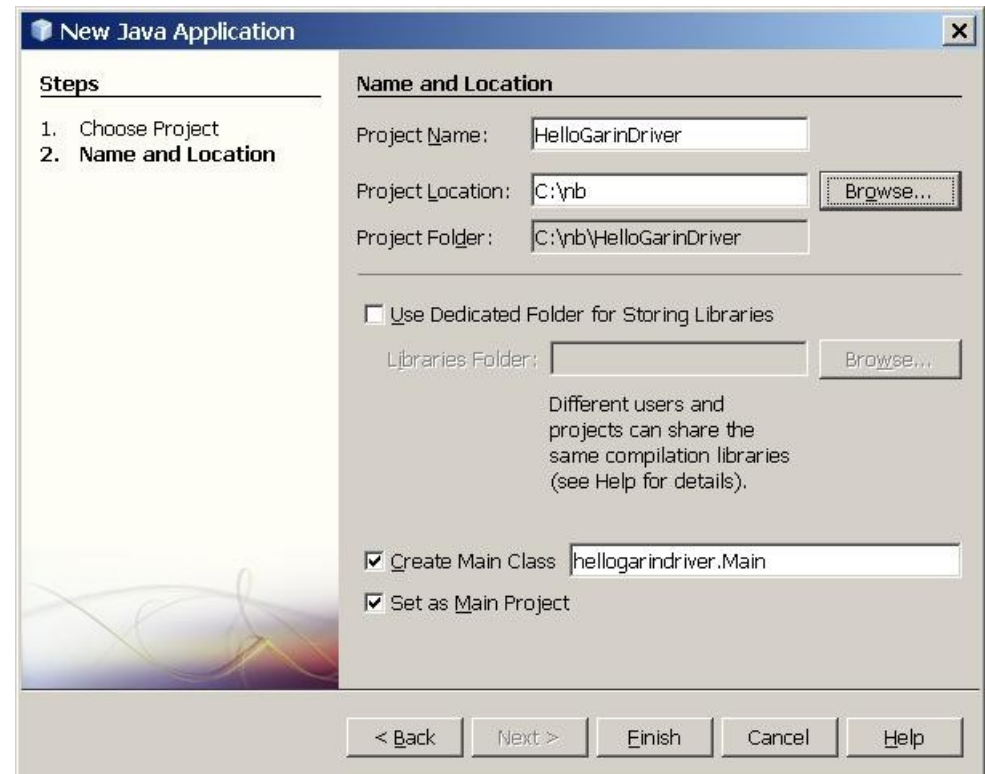
# Step 3: NetBeans Project

For the purpose of this example, we are creating a very simple hello world application from the schema and driver perspective everything works the same regardless of project type. The GarinDriver solution works for any project that connects to data via JDBC.

Make sure to keep track of the project location.

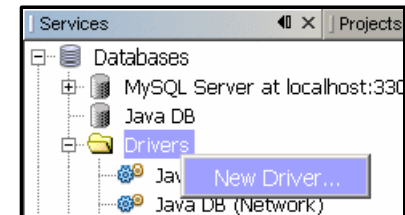
I am using "C:\nb\HelloGarinDriver".

If you use a different location, make sure you keep track of it since we will use the file system location subsequent steps when we configure our JDBC URLs for sharing schema across database instances.

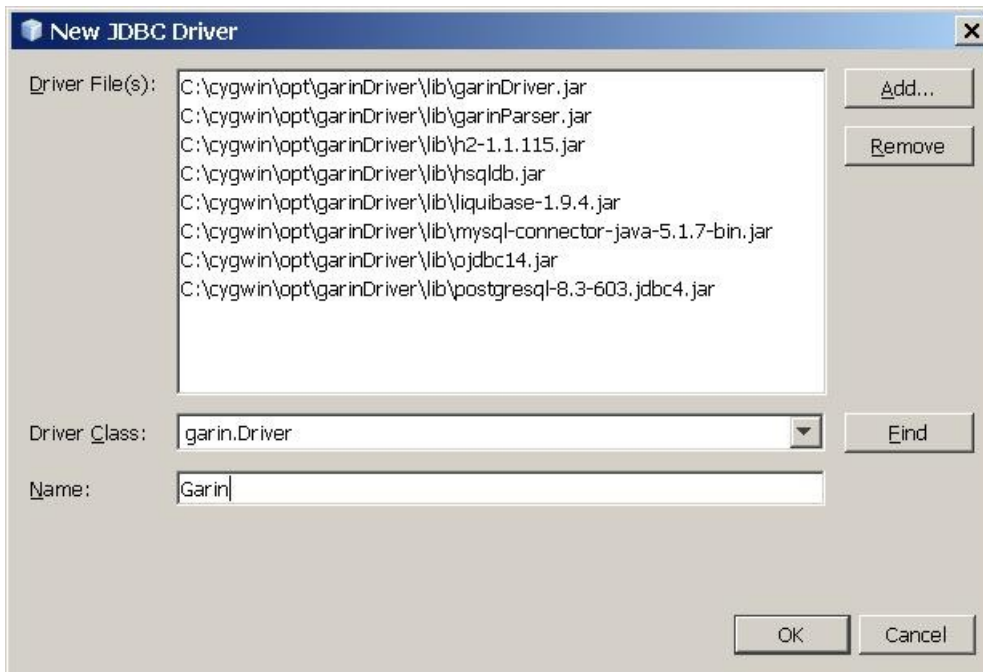


# Step 4: Configure NetBeans

Since the GarinDriver system is packaged as a JDBC driver you can use it in any context that supports the JDBC specification. A good example of this is the data tools that ship with NetBeans. In this step we will setup the driver and then connect to the database we created in step 4.



Right click on the drivers section of the databases node in the services window to invoke the new driver dialog.

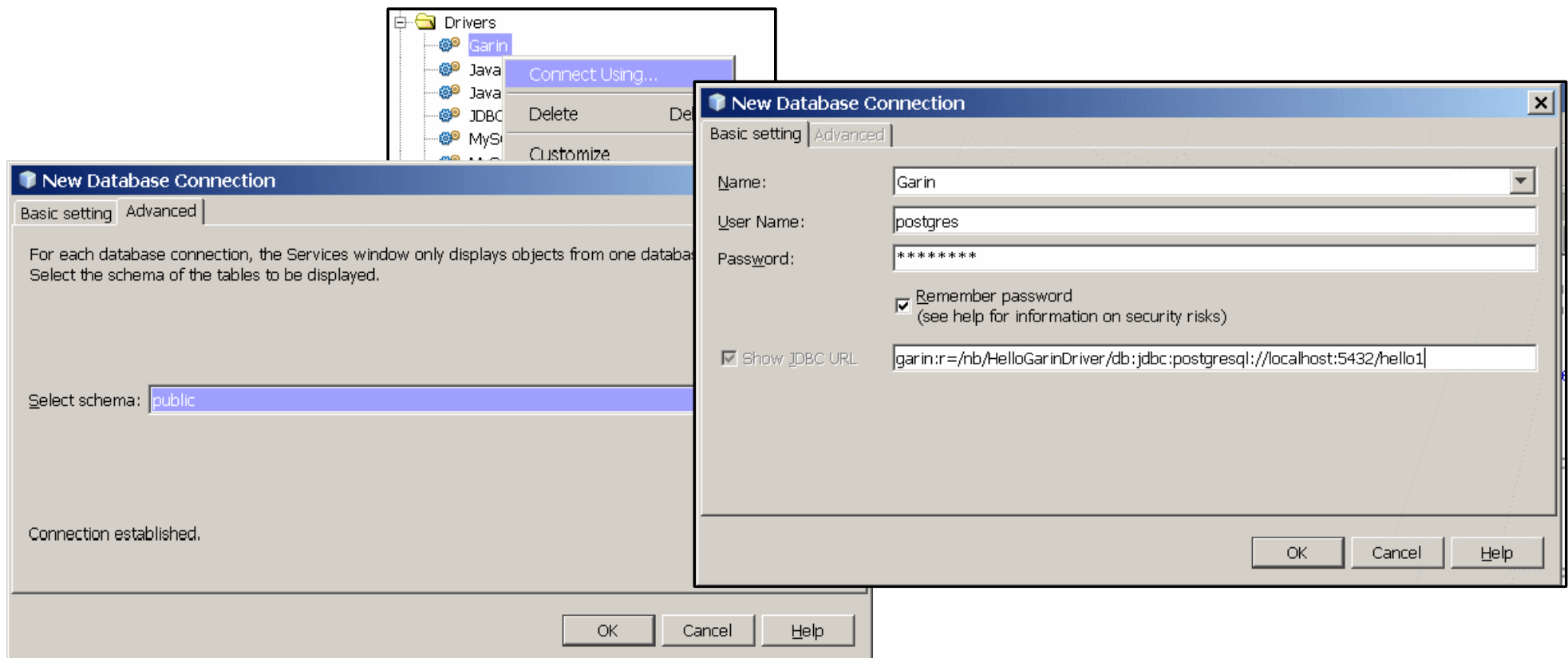


For the driver definition, we add all jars in the garinDriver/lib folder and specify garin.Driver as the JDBC driver class. We then name the driver as "Garin"

# Step 5: Connect to the DB

Right click on the driver to create a new connection that uses the GarinDriver.

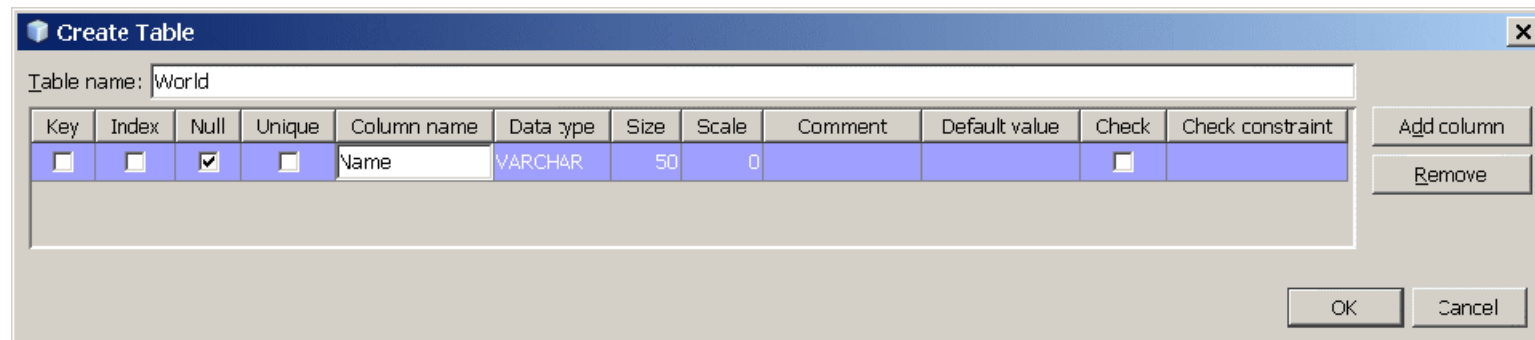
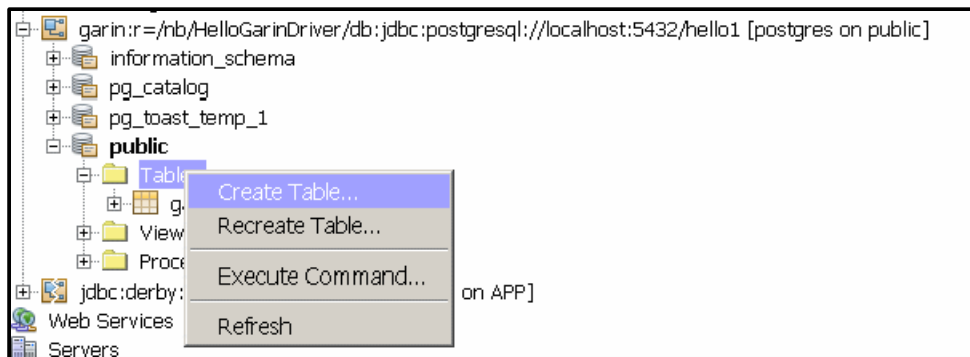
Configure the connection based on the database you created in step 2.  
“garin:r=/nb>HelloGarinDriver/db;jdbc:postgresql://localhost:5432/hello1”



# Step 6: Create Table

You can create database structures for the schema using any tool that works with JDBC.

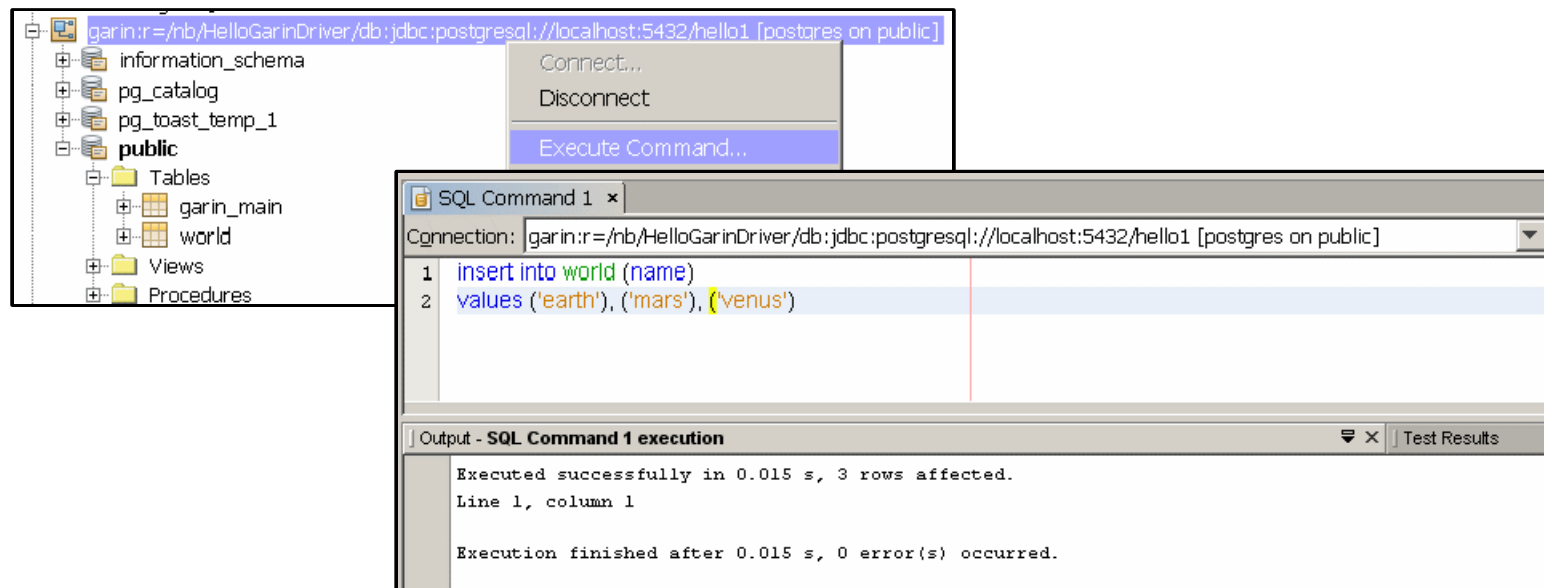
The NetBeans create table dialog is one example of such a tool.



# Step 7: Insert Rows

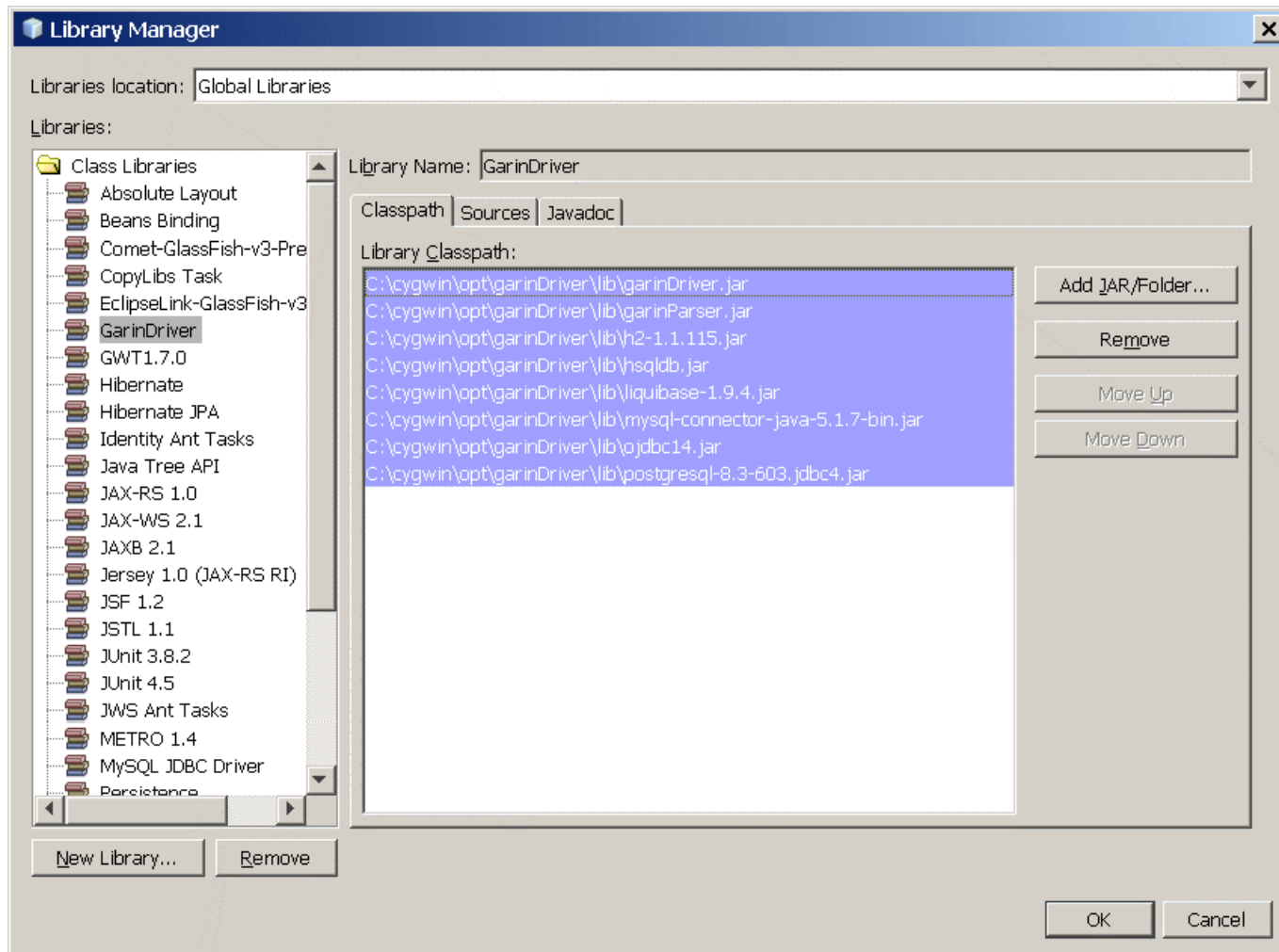
In NetBeans you can use a SQL command window to issue SQL commands such as insert, update, or select statements.

Open the command window and issue a insert statements to create rows for Earth, Mars, and Venus.



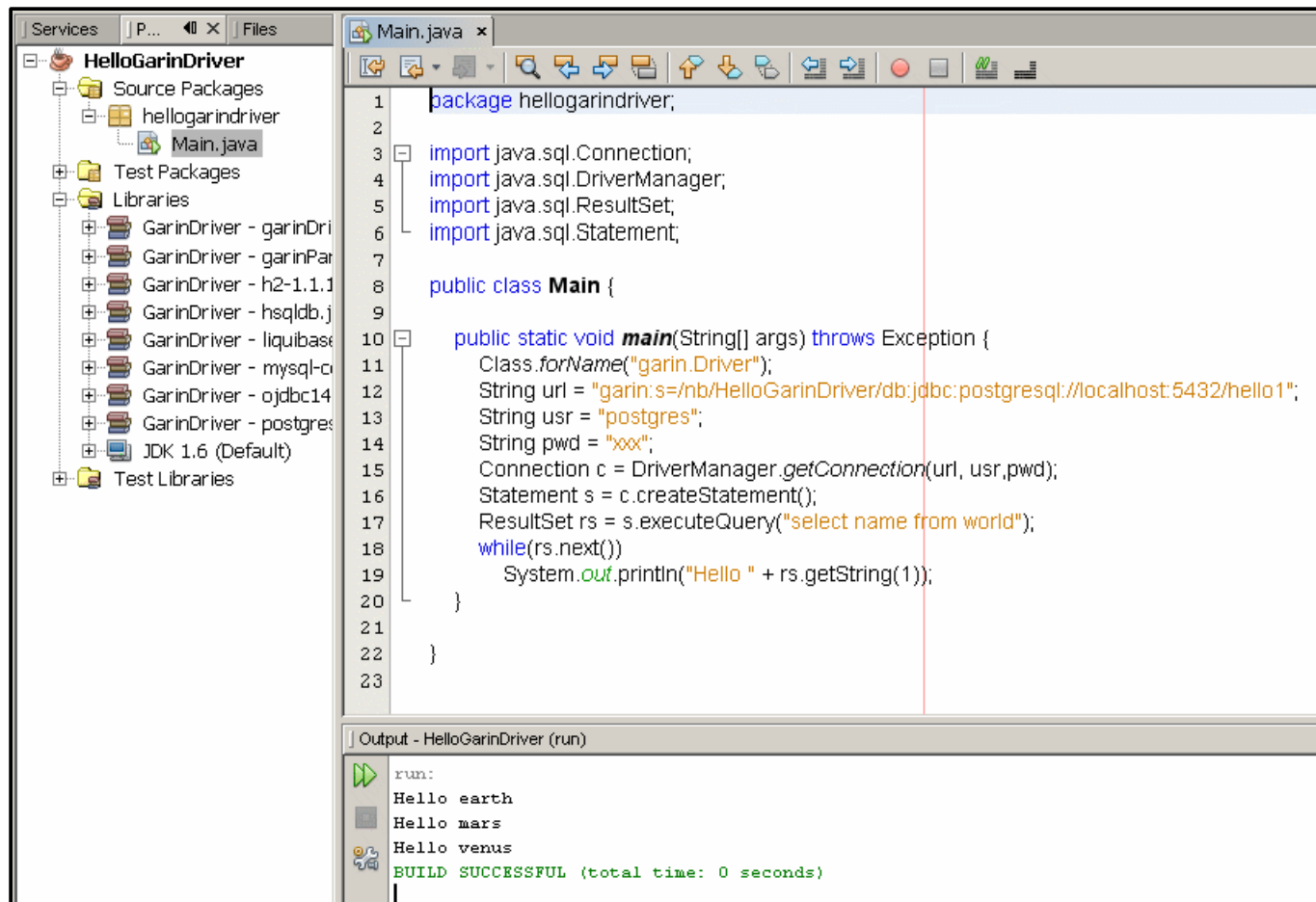
# Step 8: Define Library

Use the Library Manager to define a new library for GarinDriver.



# Step 9: Create Main Class

Create a very simple java main method to produce output based on the database content.



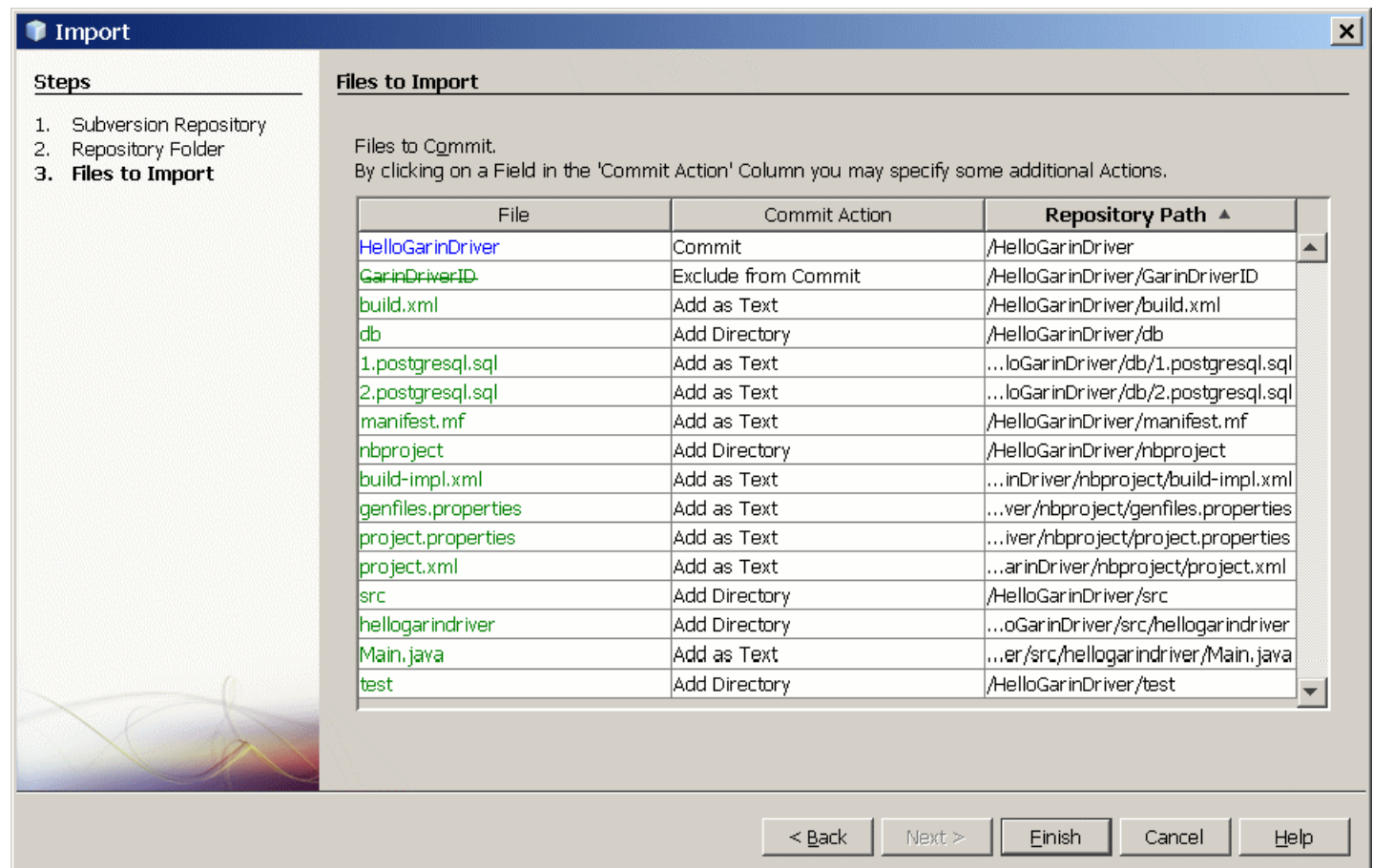
```
1 package hellogarindriver;
2
3 import java.sql.Connection;
4 import java.sql.DriverManager;
5 import java.sql.ResultSet;
6 import java.sql.Statement;
7
8 public class Main {
9
10     public static void main(String[] args) throws Exception {
11         Class.forName("garin.Driver");
12         String url = "garin:s=/nb/HelloGarinDriver/db;jdbc:postgresql://localhost:5432/hello1";
13         String usr = "postgres";
14         String pwd = "xxx";
15         Connection c = DriverManager.getConnection(url, usr,pwd);
16         Statement s = c.createStatement();
17         ResultSet rs = s.executeQuery("select name from world");
18         while(rs.next())
19             System.out.println("Hello " + rs.getString(1));
20     }
21
22 }
23
```

Output - HelloGarinDriver (run)

```
run:
Hello earth
Hello mars
Hello venus
BUILD SUCCESSFUL (total time: 0 seconds)
```

# Step 10: Share (part 1)

When you share the project make sure to exclude GarinDriverID from the commit and add this file as part of your SVN ignore list. The file identifies the instance ID for a driver runtime location and should not be shared with others.



# Step 11: Share (part 2)

On another developer's machine get the project and configure your environment but do not create any tables or insert any records. When you connect to the database for the first time GarinDriver will automatically initialize the database by running the scripts to create the table and insert the records.

Since all schema changes recorded by the driver are stored in your project location the exchange of schema information is managed using the normal SVN workflow.