

CSRI TUTORIALS



# **The PRAXICON database Tutorial**

Athens, 2014

This document was prepared by:

Dimitris Mavroeidis, Cognitive Systems Research Institute (CSRI).

To be cited as:

The PRAXICON database Tutorial, version 1.0, CSRI Technical Reports, 2014.

This document is available:

On github as part of the PRAXICON database software<sup>1</sup>.

On the CSRI website<sup>2</sup>.

© Cognitive Systems Research Institute (CSRI) 2011-2014

---

<sup>1</sup> <https://github.com/CSRI/PraxiconDB/documentation/PraxiconDBTutorial.pdf>

<sup>2</sup> [http://www.csri.gr/.....](http://www.csri.gr/)

## Table of Contents

<b>Prerequisites .....</b>	<b>4</b>
<b>Using the PRAXICON API.....</b>	<b>5</b>
Load the database.....	5
Create project.....	5
Add Libraries .....	6
Create persistence unit .....	8
Compile and run project .....	11
<b>Retrieving data from the PRAXICON database .....</b>	<b>15</b>
With JPA.....	15
With SQL .....	15
<b>XML functionality.....</b>	<b>16</b>
Import from XML.....	16
Export to XML.....	16

# Prerequisites

Before you start using the PRAXICON database application, you need to do the following:

- Install the latest version of **Java JDK**. At the time this document is written, the latest JDK is in version 8 and you can download it from:  
<http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>.
- Install an integrated development environment (IDE) supporting Java. The PRAXICON database application can be loaded on any such IDE. In this document, we assume the use of **NetBeans**. The latest NetBeans (v.8) can be found here:  
<http://www.oracle.com/technetwork/java/javase/downloads/jdk-netbeans-jsp-142931.html>. Earlier versions like 6.9.1 or 7.2 will equally work.
- Download an implementation of **Java Persistence API v.2.1**. We use the Hibernate implementation, which –at the time this document was written– was in version 4.3.4. It can be downloaded from here: <http://hibernate.org/orm/downloads/>.
- Download and install **MySQL RDBMS**. Any release from 5.0 onwards is sufficient. The latest version can be found here: <http://dev.mysql.com/downloads/mysql/>. It is also advisable to get [MySQL Workbench](#) or [PHPMyAdmin](#) to manage your database. Other RDBMS could be used, but at your own risk.
- **Configure MySQL**, making sure that the following setting are in my.ini:
  - o `character_set_server=utf8`
  - o `lower_case_table_names=0`
- Download the **PraxiconDB library** from:  
<https://github.com/csri/praxicondb/tree/master/target/dist/PraxiconDB.jar>.
- Download the appropriate **MySQL connector** for Java. The latest one can be found at <https://dev.mysql.com/downloads/connector/j/>.
- Download the “praxicon\_csri.sql” script from  
[www.csri.gr/downloads/praxicon\\_csri.sql.7z](http://www.csri.gr/downloads/praxicon_csri.sql.7z)

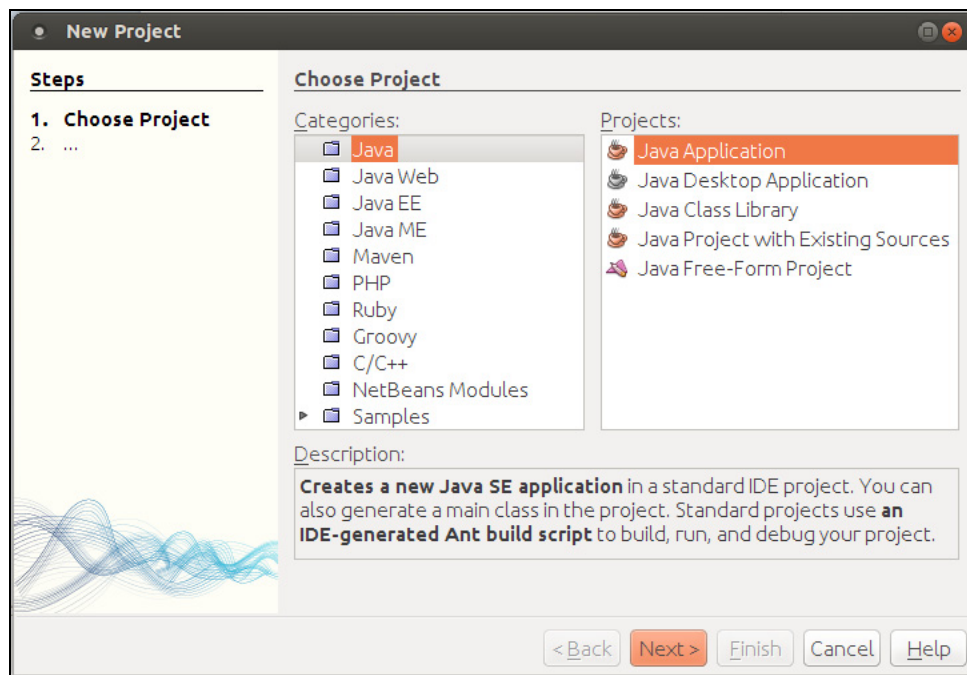
# Using the PRAXICON API

## Load the database

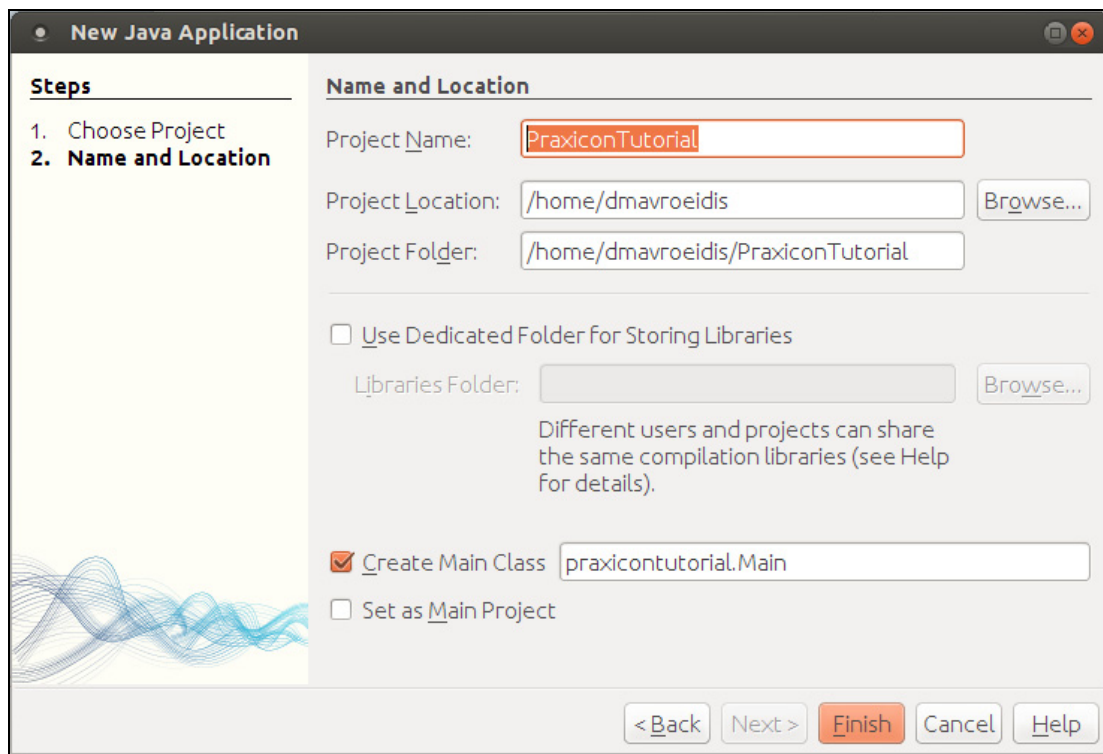
1. Create a database named “praxicon\_csri”. You can use any other valid name, as long as you are consistent throughout this tutorial.
2. Load the provided sql script “praxicon\_csri.sql”. If you changed the database name, then change the first line of this file to “USE <your\_database\_name>”.
3. Create a new database user “praxicon\_db\_user”.
4. Grant the “praxicon\_db\_user” all privileges to the “praxicon\_csri” database.

## Create project

1. Start NetBeans.
2. Create a new project (File → New Project).
3. Select “Java Application”.



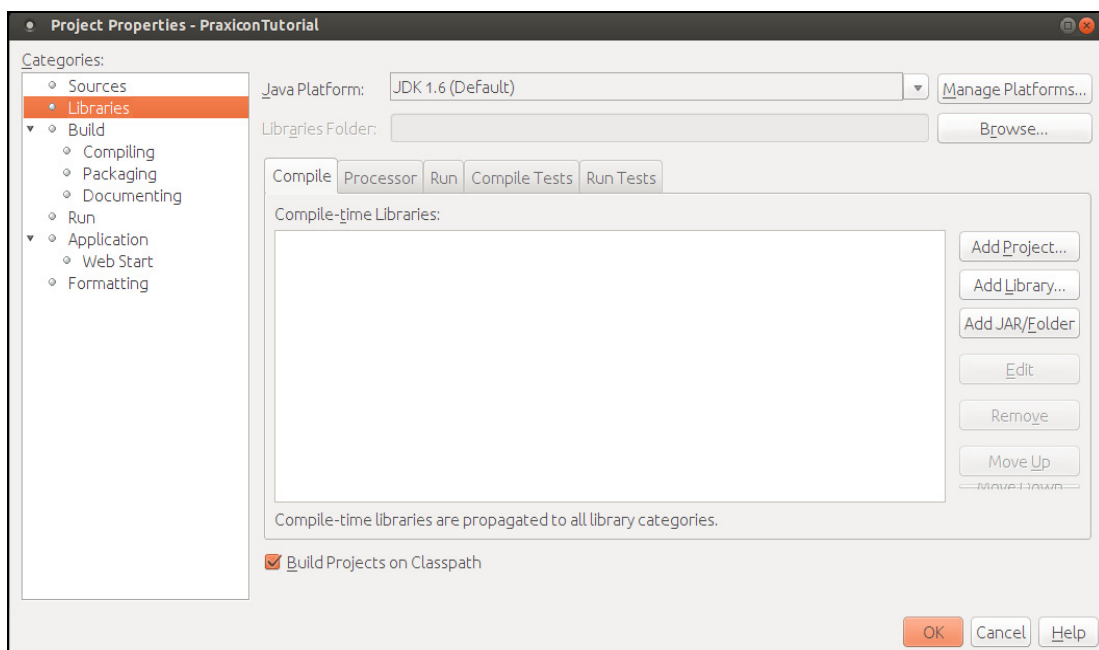
4. Provide a meaningful name for your project (e.g. “PraxiconTutorial”) and click “Finish”.



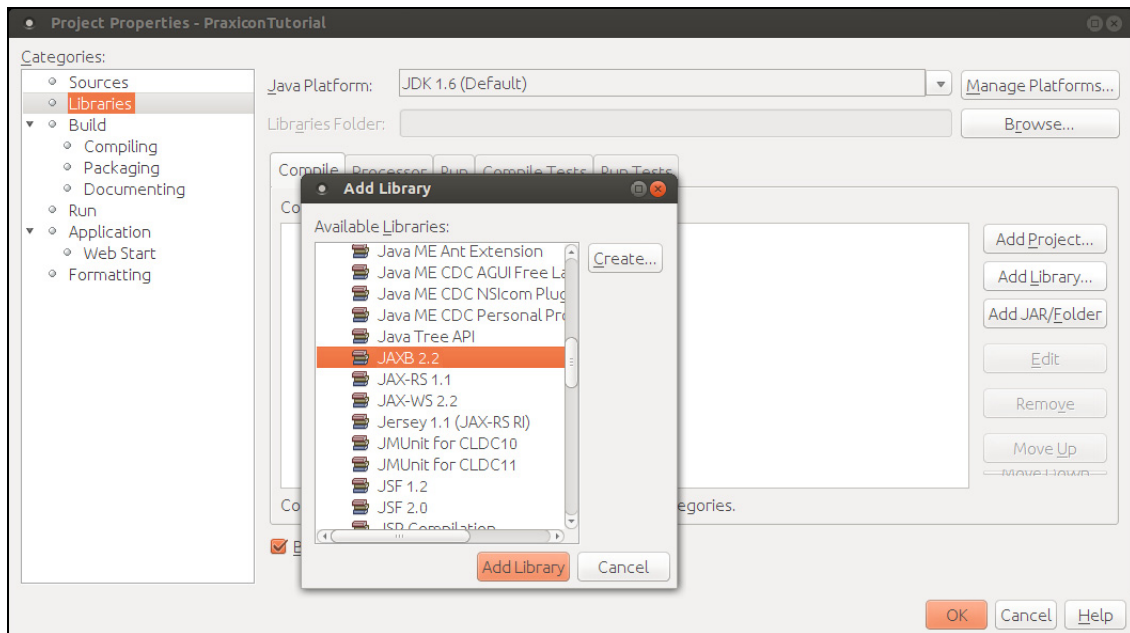
5. Right-click on the project's name and select "Properties".

## Add Libraries

6. Select "Libraries" on the left and press the "Add Library" button.



7. Select "JAXB2.2" and click the "Add Library" button.

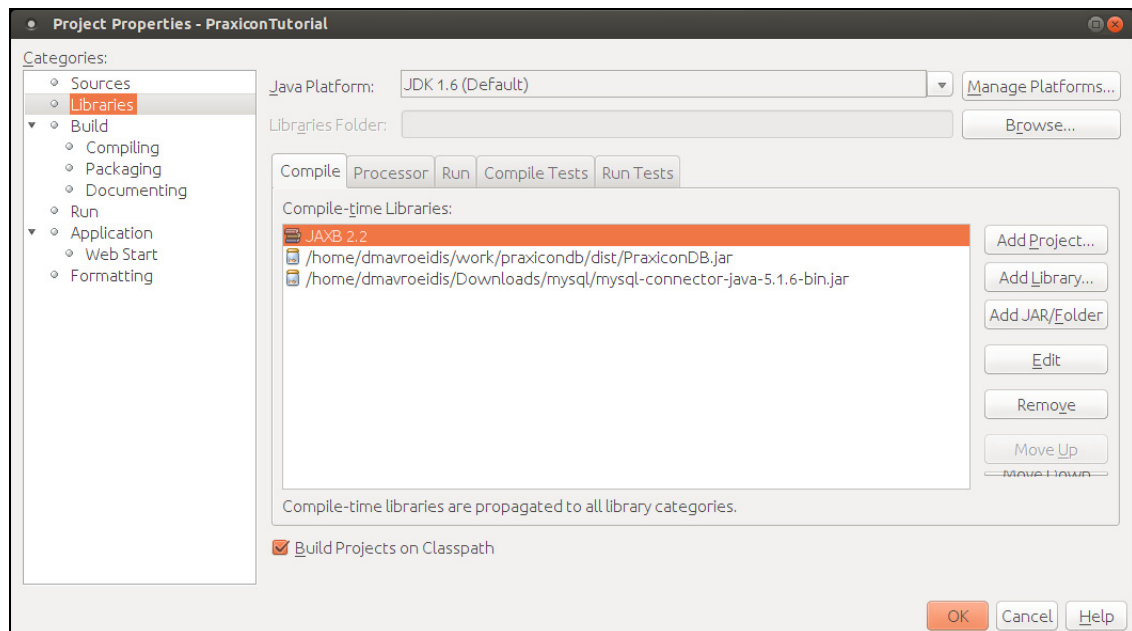


8. Press the “Add JAR/Folder” button.

9. Select the “PraxiconDB.jar” file that you downloaded earlier and press “OK”.

10. Select the mysql connector library file (e.g. “mysql-connector-java-5.1.27-bin.jar”) and press “OK”.

11. Now, you should have three items in the “Compile” tab, like below:

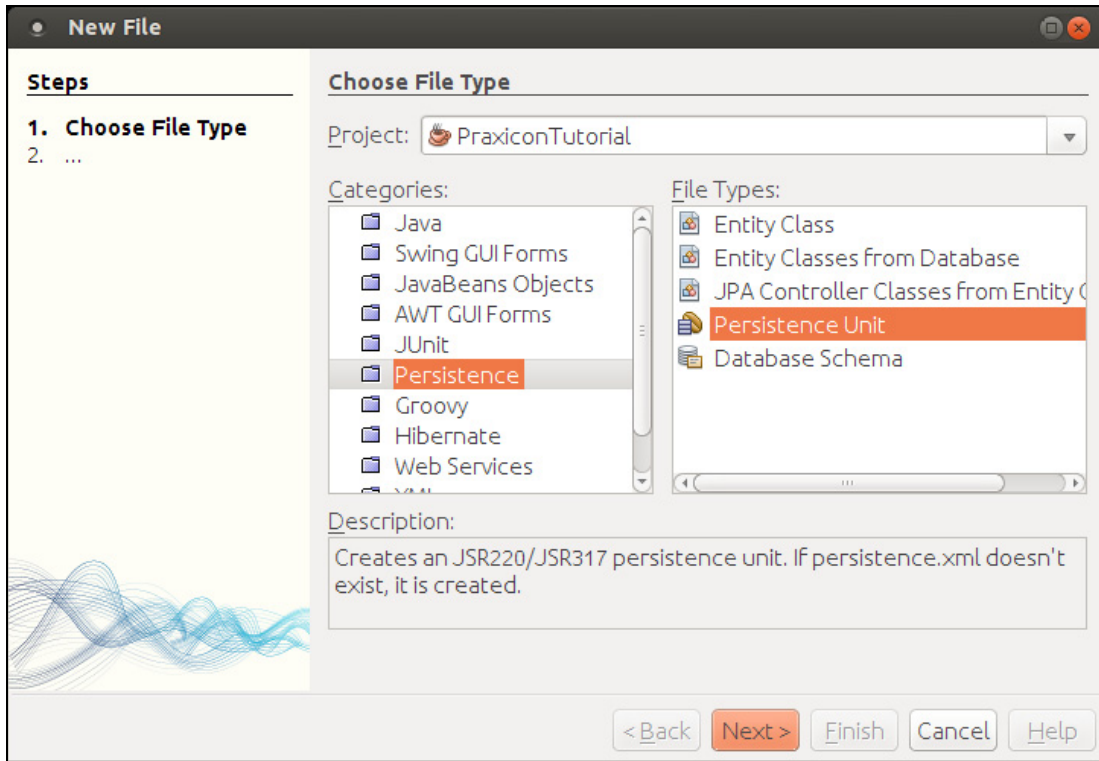


12. Press the “OK” button.

## Create persistence unit

13. Right-click on the project name and select “New→Other”

14. Select the “Persistence” category and “Persistence Unit” file type and press “Next”.

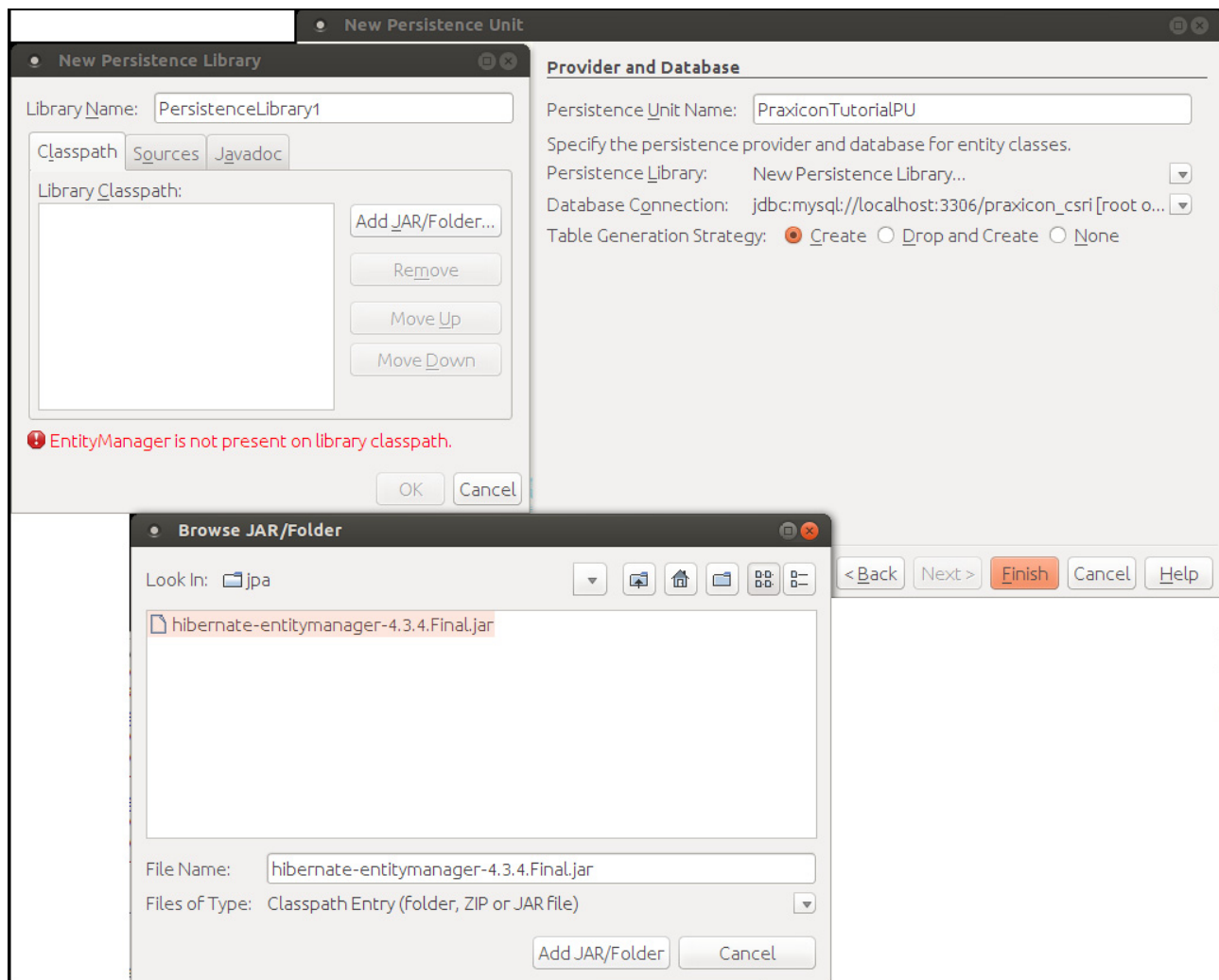


15. Provide a name for the persistence unit (e.g. PraxiconTutorialPU).

16. In the “Persistence Library” field, select “Hibernate(JPA 2.0)” (which is actually JPA2.1 compliant) or any JPA2.1-compliant persistence library. If the selection is not there, do the following:

- Select the “Persistence Library” field.
- Select “New Persistence Library”.
- Click on “Add JAR/Folder”.
- Select the JPA2.1-compliant library of your choice (preferably Hibernate).
- Click “OK”.





17. Select a “Database connection” from the list. If the database that holds the “praxicon\_csri” database is not in the list, then:

- a. Select “New Database Connection...”.
- b. Fill in the connection details as per the following screenshot.
- c. Press the “OK” button.

The “New Database Connection” window should look like this:

**New Database Connection**

Basic setting | Advanced

Data Input Mode: ☒ Field Entry ☐ Direct URL Entry

Driver Name: MySQL (Connector/J driver)

Host: localhost

Port: 3306

Database: praxicon\_csri

User Name: praxicon\_db\_user

Password: \*\*\*\*\*

Display Name (Optional):

☐ Remember password  
(see help for information on security risks)

Additional Props:

☐ Show JDBC URL

OK Cancel Help

18. In “Table Generation Strategy”, select “None”.

19. You should now be able to see the full picture of the persistence unit. Check that your settings are similar to the following screenshot.

**New Persistence Unit**

**Steps**

1. Choose File Type
2. **Provider and Database**

**Provider and Database**

Persistence Unit Name: PraxiconTutorialPU

Specify the persistence provider and database for entity classes.

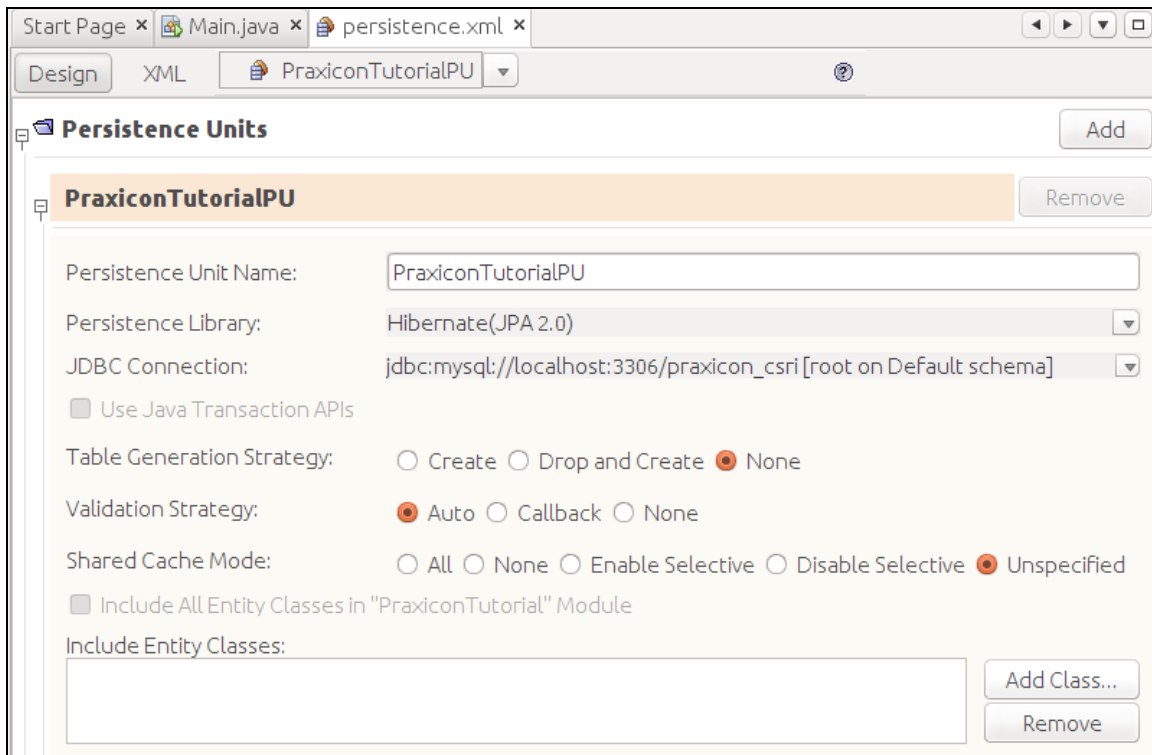
Persistence Library: Hibernate(JPA 2.0)

Database Connection: jdbc:mysql://localhost:3306/praxicon\_csri[...]

Table Generation Strategy: ☐ Create ☐ Drop and Create ☒ None

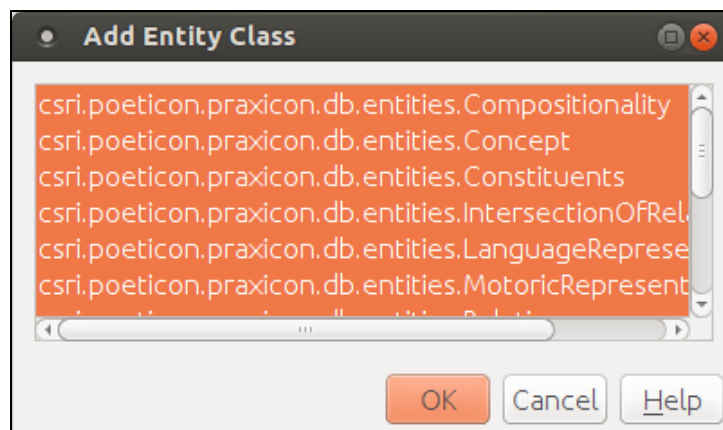
< Back Next > Finish Cancel Help

20. When you click on “Finish”, a new file called “persistence.xml” will be created at the root of your project and opened by default.



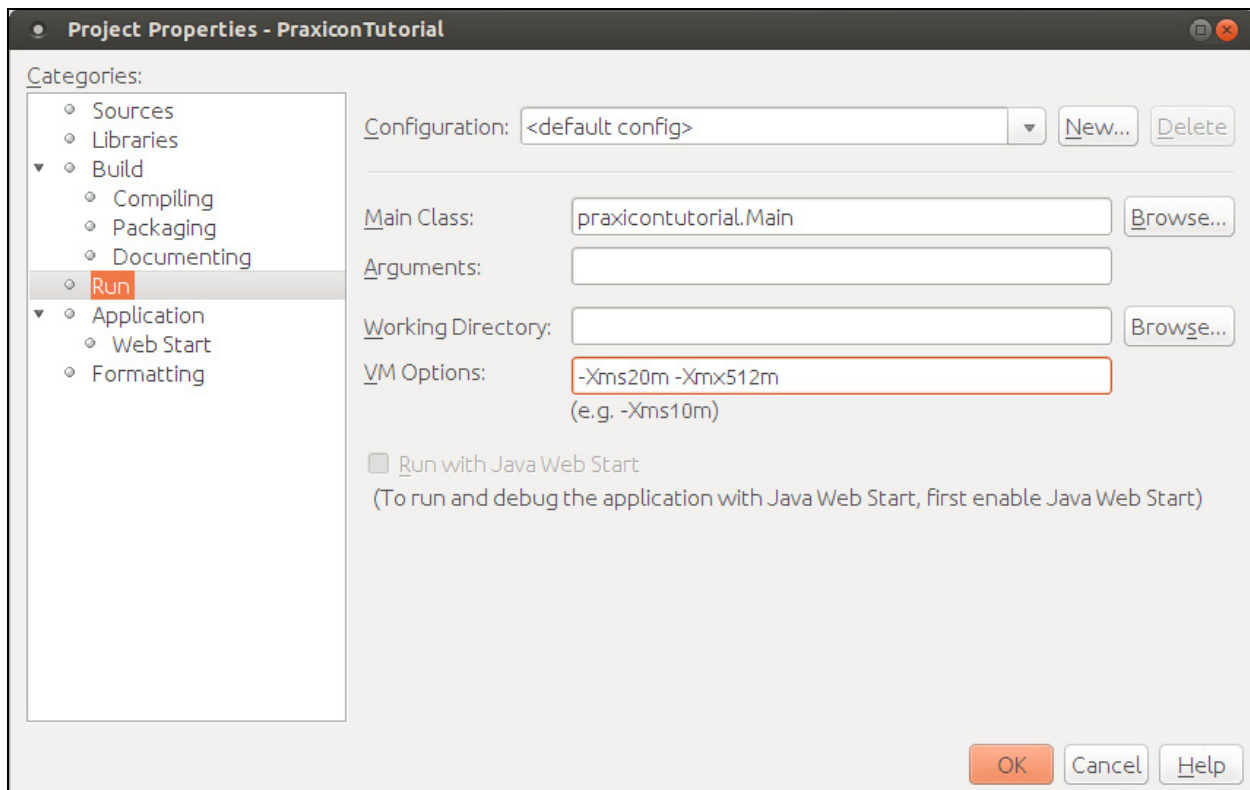
21. Press the “Add Class” button.

22. Select all classes and press the “OK” button.



## Compile and run project

23. Now, right-click on the project name, select “Properties → Run” and add the following to “VM Options”: “-Xms20m -Xmx512m”.



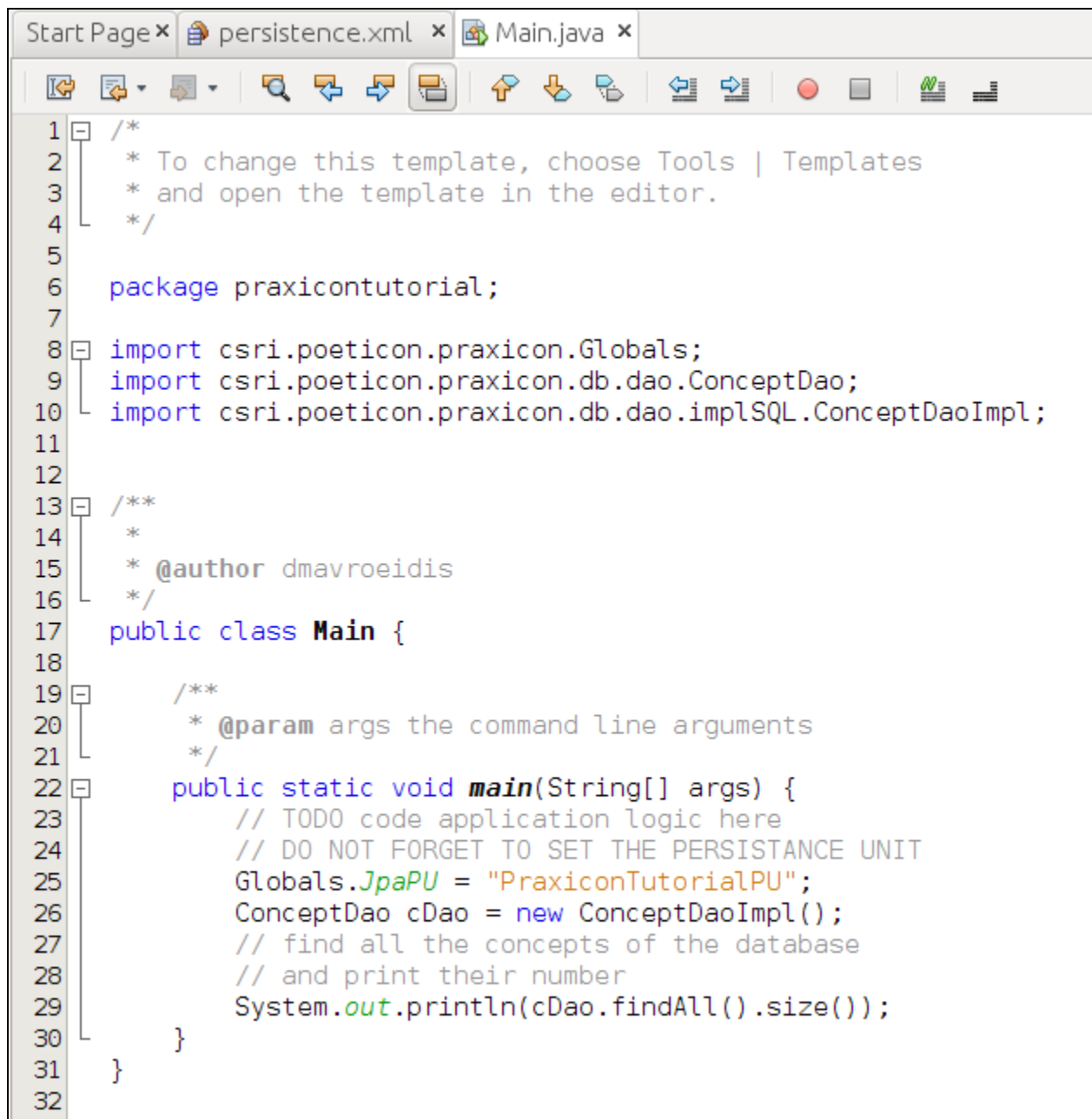
24. Now, open the “Main.java” file and add the following in the “Main” section:

```
Globals.JpaPU = "PraxiconTutorialPU";
ConceptDao cDao = new ConceptDaoImpl();
// find all the concepts of the database
// and print their number
System.out.println(cDao.findAll().size());
```

25. You will notice that the first two lines of the code seem to cause some errors. We need to import a few libraries:

```
import csri.poeticon.praxicon.Globals;
import csri.poeticon.praxicon.db.dao.ConceptDao;
import csri.poeticon.praxicon.db.dao.implSQL.ConceptDaoImpl;
```

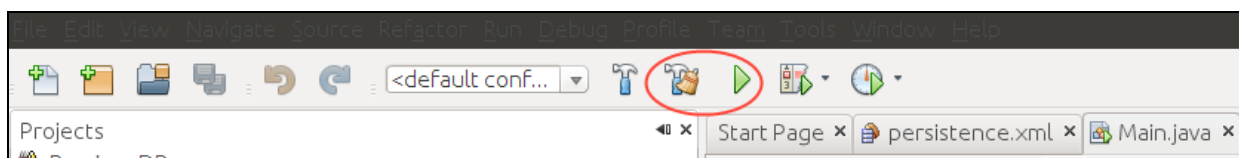
These imports should go under the package declaration at the beginning of the “Main.java” file like below.







```
1  /*
2   * To change this template, choose Tools | Templates
3   * and open the template in the editor.
4   */
5
6  package praxicontutorial;
7
8  import csri.poeticon.praxicon.Globals;
9  import csri.poeticon.praxicon.db.dao.ConceptDao;
10 import csri.poeticon.praxicon.db.dao.implSQL.ConceptDaoImpl;
11
12
13 /**
14  *
15  * @author dmavroeidis
16  */
17 public class Main {
18
19     /**
20      * @param args the command line arguments
21      */
22     public static void main(String[] args) {
23         // TODO code application logic here
24         // DO NOT FORGET TO SET THE PERSISTENCE UNIT
25         Globals.JpaPU = "PraxiconTutorialPU";
26         ConceptDao cDao = new ConceptDaoImpl();
27         // find all the concepts of the database
28         // and print their number
29         System.out.println(cDao.findAll().size());
30     }
31 }
32
```

26. After importing these libraries, we can build and run the application. This application will output all concepts that are in the “praxicon\_csri” database.

27. To build the application, you can press the “Clean and Build” (the hammer and broom) button. To run it you can use the “Run” (the green play) button.



Output - PraxiconTutorial (clean,jar) Usages Search Results

 init:  
 deps-clean:  
Updating property file: /home/dmavroeidis/PraxiconTutorial/build/built-clean.properties  
 Deleting directory /home/dmavroeidis/PraxiconTutorial/build  
clean:  
 init:  
deps-jar:  
Created dir: /home/dmavroeidis/PraxiconTutorial/build  
Updating property file: /home/dmavroeidis/PraxiconTutorial/build/built-jar.properties  
Created dir: /home/dmavroeidis/PraxiconTutorial/build/classes  
Created dir: /home/dmavroeidis/PraxiconTutorial/build/classes/META-INF  
Copying 1 file to /home/dmavroeidis/PraxiconTutorial/build/classes/META-INF  
Created dir: /home/dmavroeidis/PraxiconTutorial/build/empty  
Created dir: /home/dmavroeidis/PraxiconTutorial/build/generated-sources/ap-source-output  
Compiling 1 source file to /home/dmavroeidis/PraxiconTutorial/build/classes  
compile:  
Created dir: /home/dmavroeidis/PraxiconTutorial/dist  
Copy libraries to /home/dmavroeidis/PraxiconTutorial/dist/lib.  
Building jar: /home/dmavroeidis/PraxiconTutorial/dist/PraxiconTutorial.jar  
To run this application from the command line without Ant, try:  
java -jar "/home/dmavroeidis/PraxiconTutorial/dist/PraxiconTutorial.jar"  
jar:  
BUILD SUCCESSFUL (total time: 1 second)

## Retrieving data from the PRAXICON database

Before attempting to retrieve any data from the PRAXICON database, we suggest that you read the documentation which is available at the project's github page<sup>3</sup>.

### With JPA

We will describe a complex example of retrieving data from the PRAXICON database using JPA. We will use the "Main.java" we used in the previous section.

We want to retrieve all concepts that are objects in the relations of type "HAS\_INSTANCE". We first create a method to retrieve all concepts given a relation type:

```
public static List<Concept> getObjectsOfRelation(String type_of_relation)
{
    //Get the appropriate RelationName object
    RelationName t = RelationType.RelationName.valueOf(type_of_relation);
    //create the JPQL query
    Query q = EntityMngFactory.getEntityManager().createQuery(
        "SELECT DISTINCT r.object FROM Relation r, RelationType rtype" +
        "WHERE r.type = rtype.id AND rtype.forwardName = ?1");
    q.setParameter(1, t);

    //Get the results
    return q.getResultList();
}
```

Then, we need to call it from the "Main.java", like that:

```
List<Concept> objects_of_relation_type;
objects_of_relation_type = getObjectsOfRelation("HAS_INSTANCE");
System.out.println(objects_of_relation_type.size());
```

### With SQL

We will use an SQL query to retrieve the same data.

1. Start the MySQL Workbench
2. Connect to the MySQL server that contains the "praxicon\_csri" database.

---

<sup>3</sup> <https://github.com/csri/praxicondb/documentation/PraxiconDBDocumentation.pdf>

3. Enter the following sql query in the query window:

```
USE praxicon_csri;
SELECT
    co.NAME
FROM
    Relation r
    INNER JOIN
    RelationType ON RelationType.Id = r.ID
    INNER JOIN
    Concept co ON co.Id = r.Object
WHERE
    RelationType.ForwardName = "HAS_INSTANCE"
;
```

4. Running it will return all concepts that serve as objects involved in a relation.

You should study the database schema (in the database documentation) in order to create queries that satisfy your needs.

## XML functionality

The PRAXICON database application has an “XMLUtils” module which allows XML files to be loaded to the database. You can also create an XML file containing data retrieved from a JPA query.

### Import from XML

There are several ways to import an XML file. The simplest one is to run

```
java -jar PraxiconDB.jar concepts.xml
```

from the command line, assuming “concepts.xml” contain the data you want to import.

### Export to XML

The “XMLUtils” class contains a “saveToXML” method. To export all concepts having “knife” as the language representation, you need to run the following code:

```
// DO NOT FORGET TO SET THE PERSISTENCE UNIT
Globals.JpaPU = "PraxiconTutorialPU";
ConceptDao cDao = new ConceptDaoImpl();
// get a list of concepts that their LR contains the string knife
List<Concepts> knifeConcepts = cDao.findAllByLanguageRepresentation("knife");
// save it as an XML called "ConceptsKnives.xml"
XML_Utils.saveToXML(knifeConcepts, "ConceptsKnives.xml");
```