

HIBERNATE

RAPPEL

HIBERNATE est un projet open source visant à proposer un outil de **MAPPING** entre les objets et des données stockées dans une base de données relationnelle.

Ce projet ne repose sur aucun standard mais il est très populaire notamment à cause de ses bonnes performances et de son ouverture avec de nombreuses bases de données

Les bases de données supportées sont les principale du marché: **DB2, Oracle, MYSQL, PostgeSQL, Sybase, SQL Server**

.....

FICHE TECHNIQUE HIBERNATE

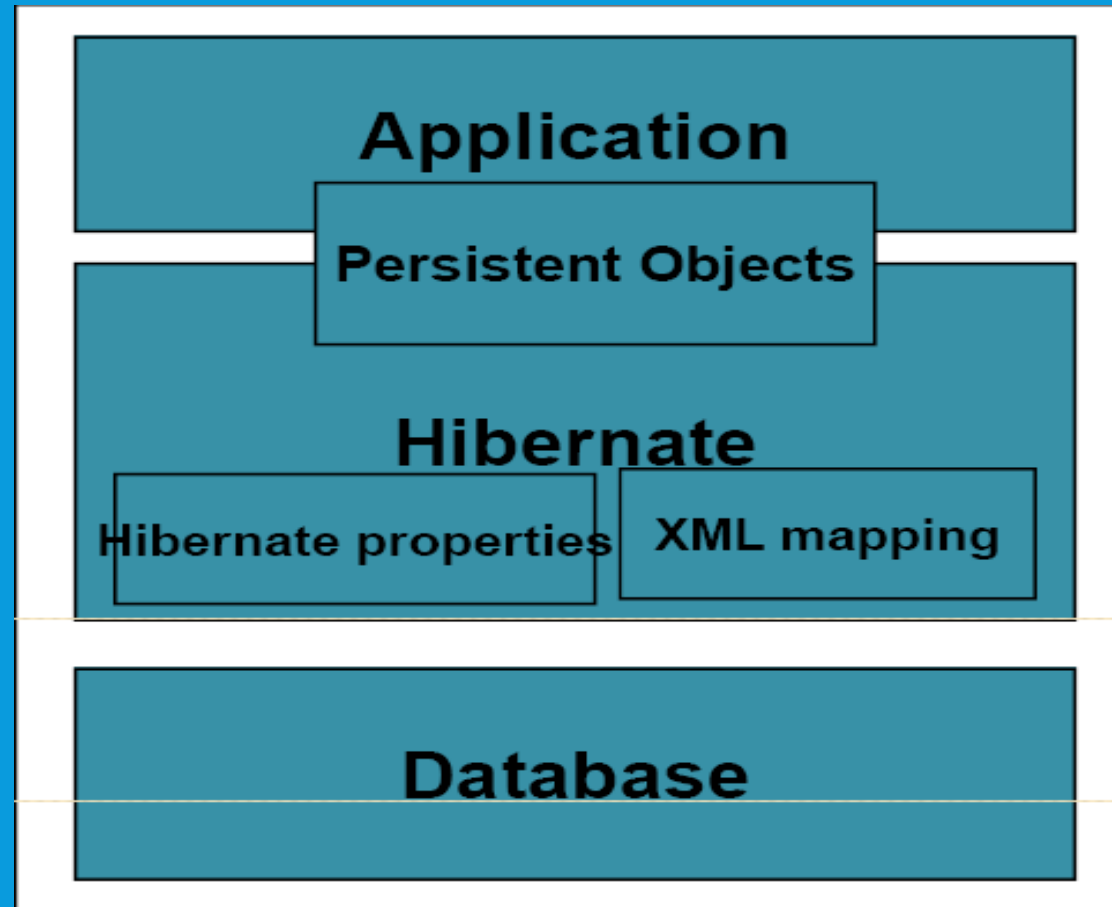


HIBERNATE

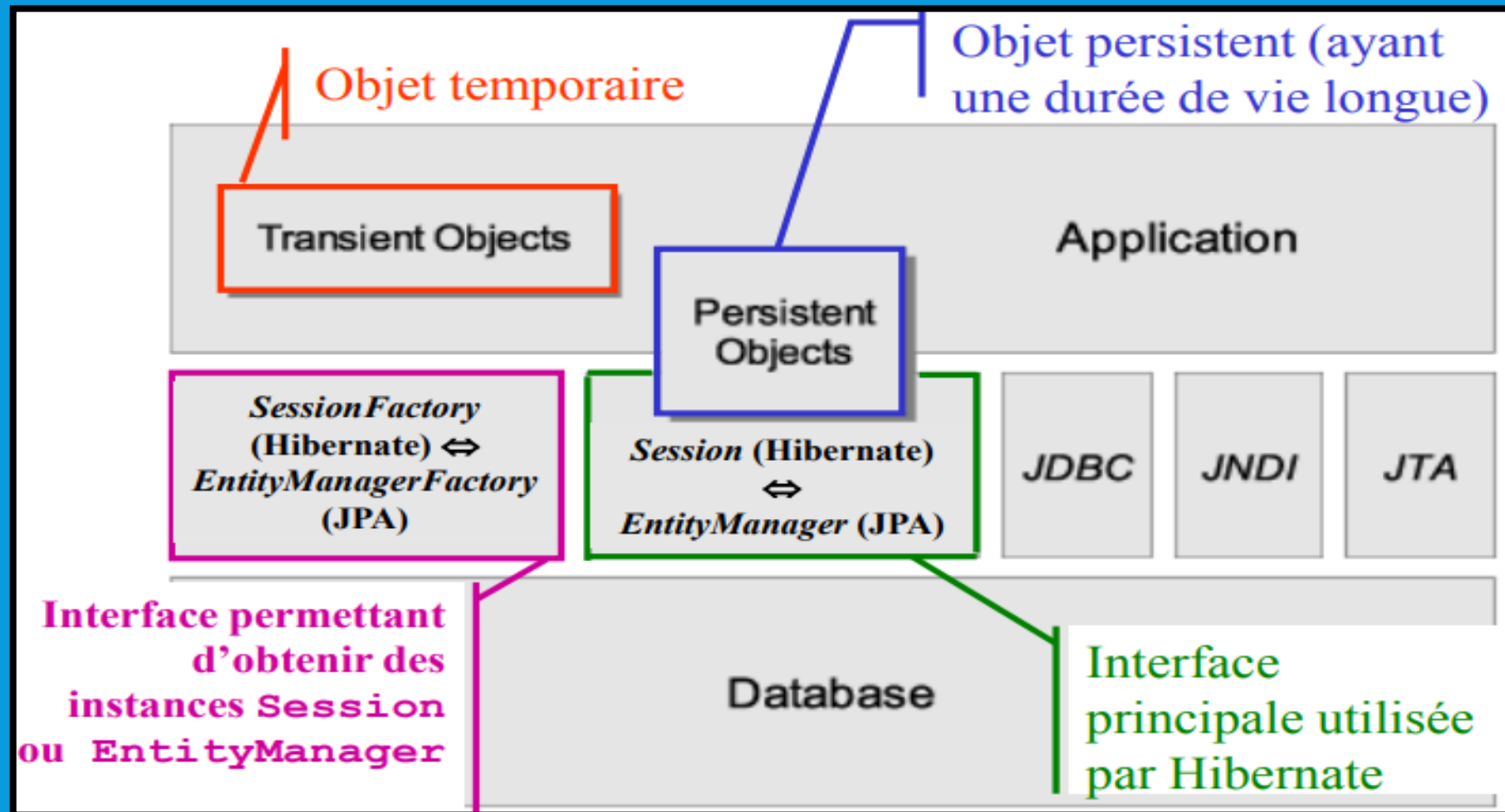
Informations

Développé par	Red Hat
Première version	23 mai 2001
Dernière version	5.4.4 (30 juillet 2019)-
Version avancée	6.0.0.Alpha2 (4 avril 2019)
Dépôt	github.com/hibernate/hibernate-orm
Écrit en	Java
Environnement	Multiplate-forme (JVM)
Langues	anglais
Type	Mapping objet-relationnel
Licence	Licence publique générale limitée GNU
Site web	hibernate.org .

ARCHITECTURE

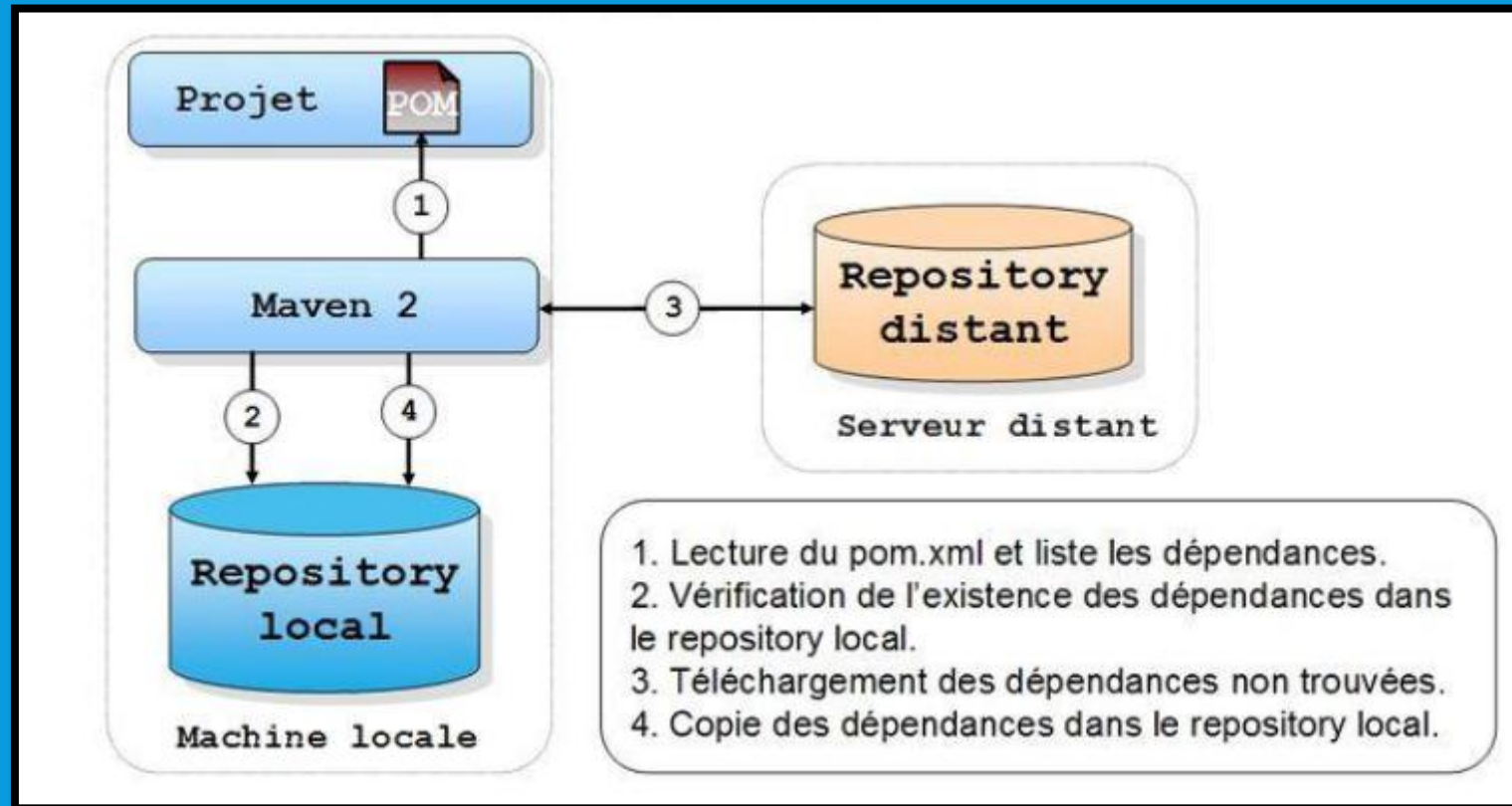


ARCHITECTURE DU NOYAU HIBERNATE

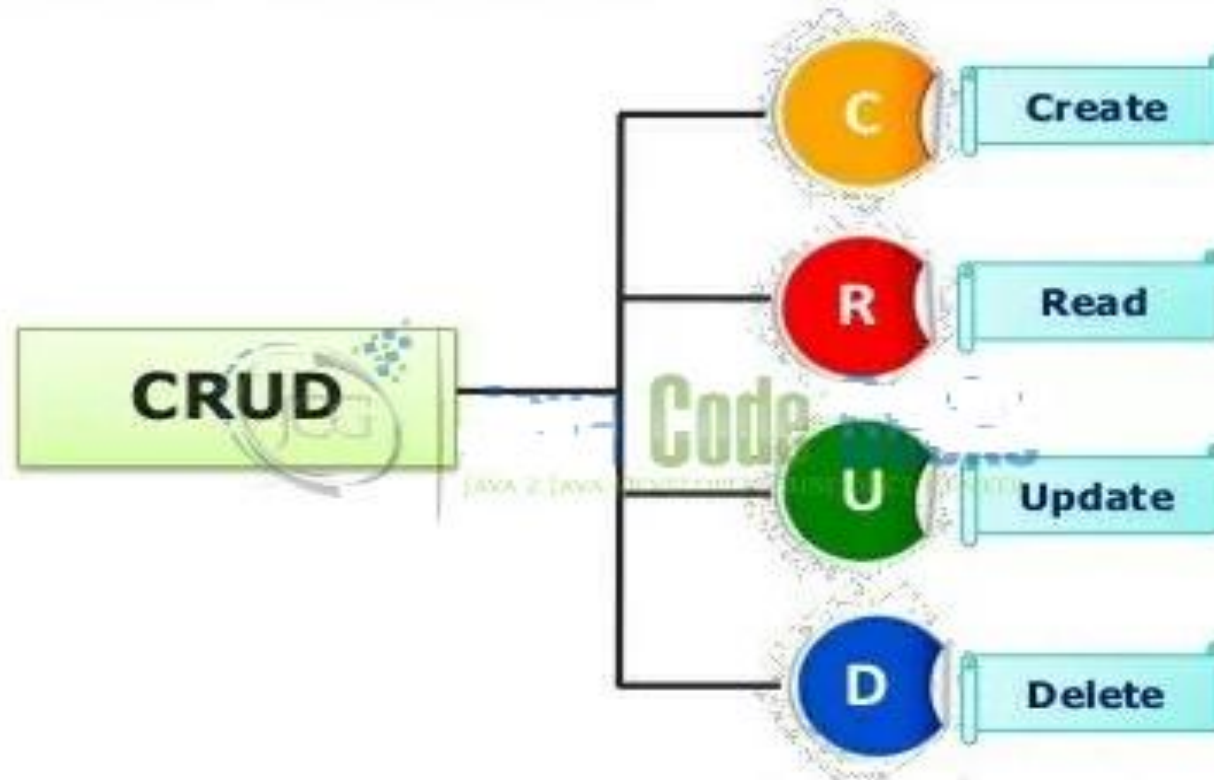


GESTION DES DÉPENDANCES PAR MAVEN

2



FINALISATION DE LA CLASSE METIER



OPERATIONS CRUD

```
session.save(e);
```

```
session.delete(e);
```

```
session.saveOrUpdate(e);
```

```
Employee emp=(Employee) session.get(Employee.class, new Integer(e.getId()));
```

```
session.createQuery("from Employee").list();
```


OPTIMISATION

```
private Session session;
```

```
private void openSession(){  
    session=HibernateUtil.getSessionFactory().openSession();  
    session.beginTransaction();  
}
```

```
private void closeSession(){  
    session.getTransaction().commit();  
    session.close();  
}
```

MAPPING À L'AIDE DE FICHIERS XML

- Soit la table Produits

Nom	Type
<u>ref</u>	int(11)
lib	varchar(255)
prixu	float

ENTITY PRODUITS

```
public class Produits {  
    private int ref;  
  
    private String lib;  
  
    private float prixu;  
  
    public int getRef() {..  
  
    public void setRef(int ref) {..  
  
    public String getLib() {..  
  
    public void setLib(String lib) {..  
  
    public float getPrixu() {..  
  
    public void setPrixu(float prixu) {..  
}
```

RODUITS.HBM.XML

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<!-- Generated 21 mai 2015 04:06:49 by Hibernate Tools 3.4.0.CR1 -->
<hibernate-mapping>

    <class name="entities.Produits" table="produits">

        <id name="ref" type="int">
            <generator class="identity"></generator>
        </id>

        <property name="lib"></property>

        <property name="prixu"></property>

    </class>
</hibernate-mapping>
```

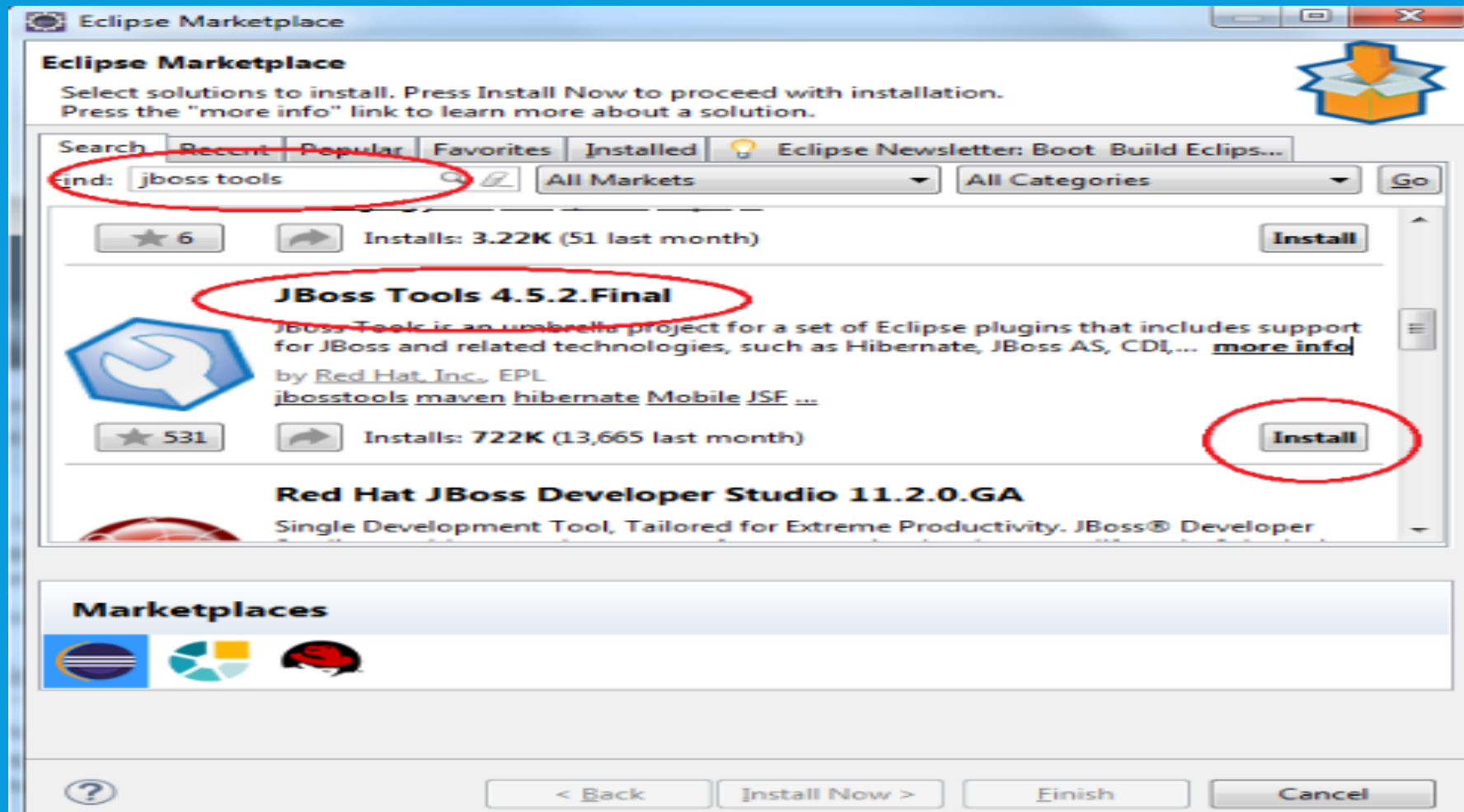
MISE À JOUR DU FICHIER DE CONFIGURATION HIBERNATE

```
<mapping resource="entities/Produits.hbm.xml"/>
```

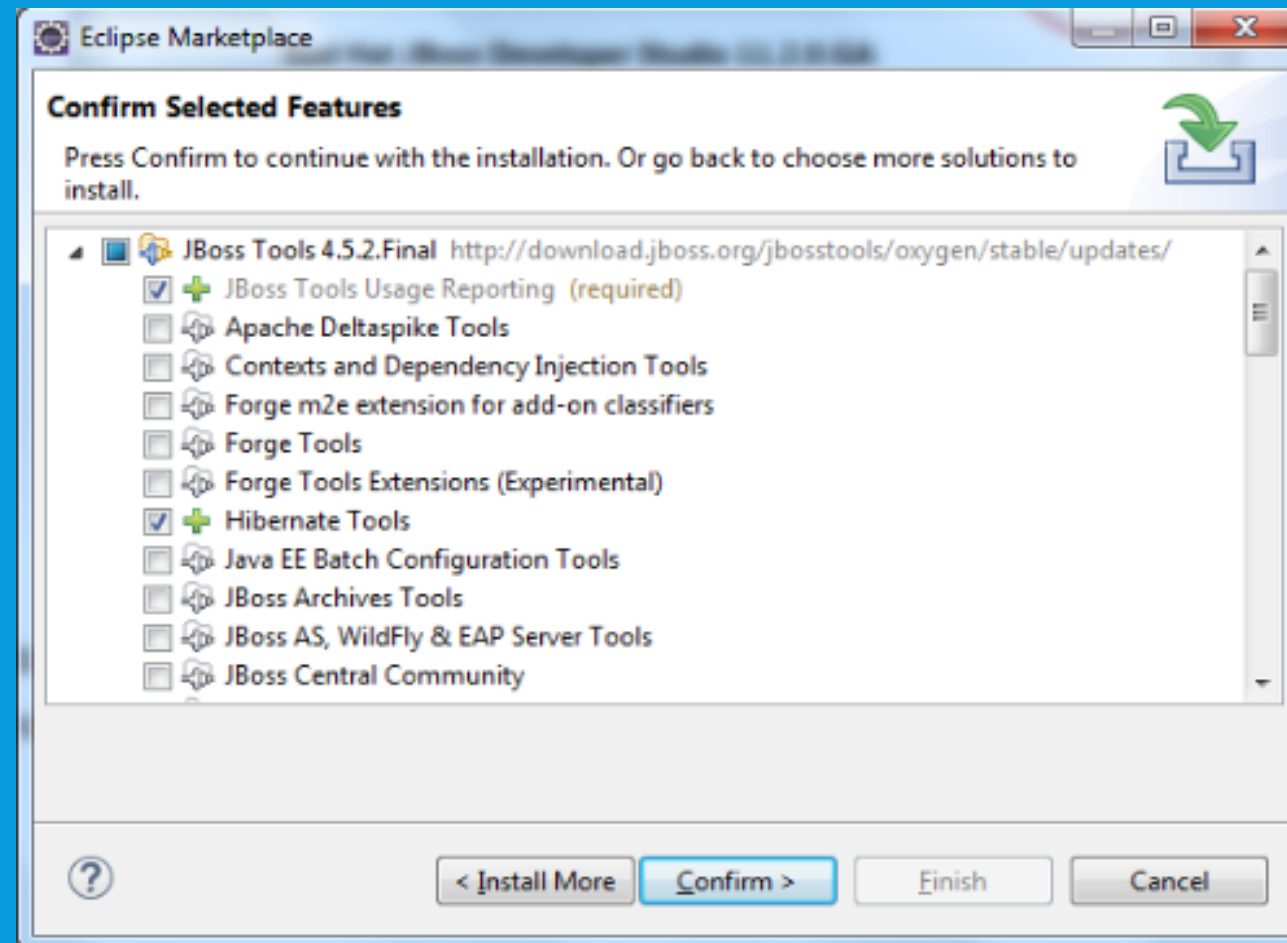
REVERSE ENGINEERING

- Comment générer les classes à partir des tables?

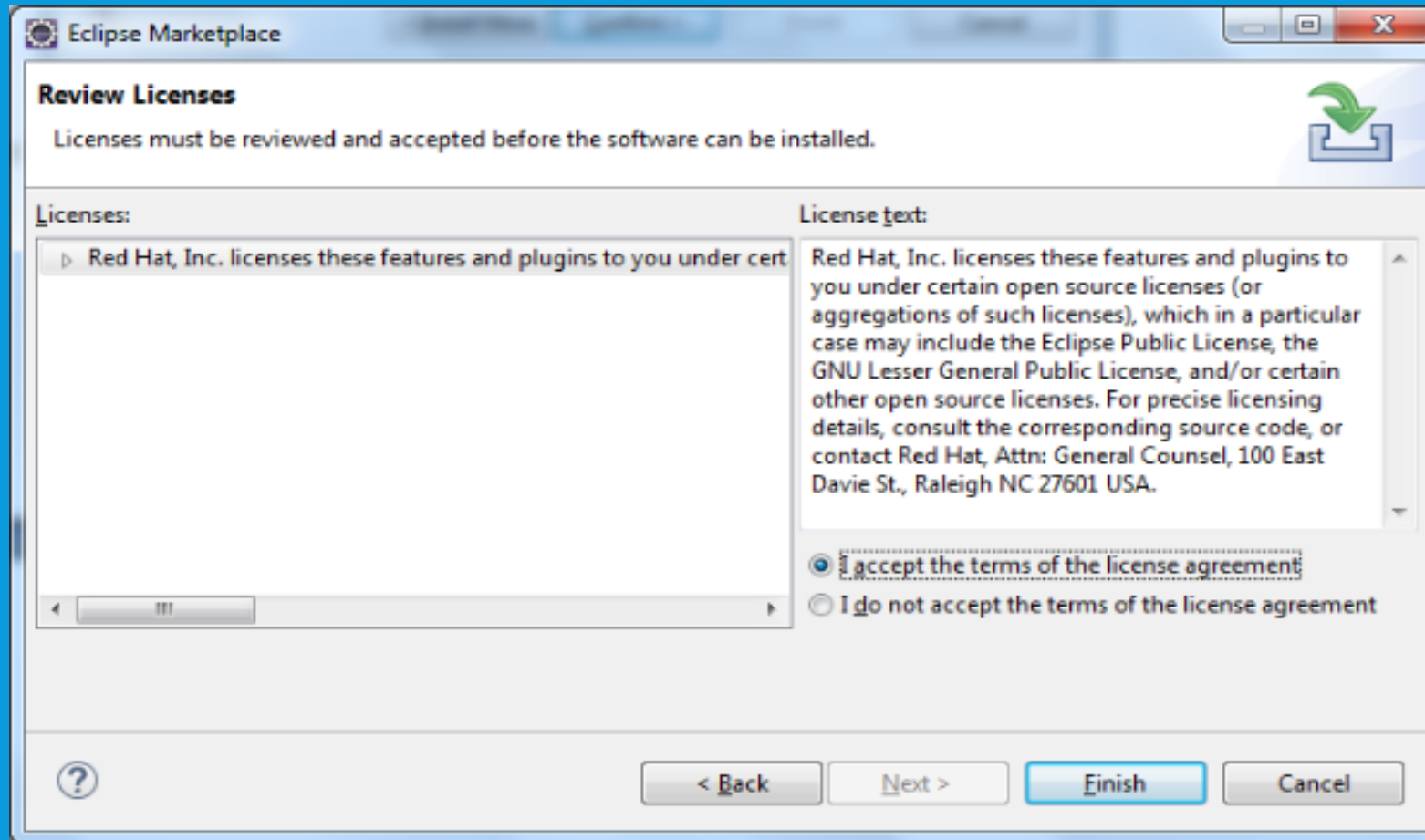
INSTALLING HIBERNATE TOOLS IN ECLIPSE IDE



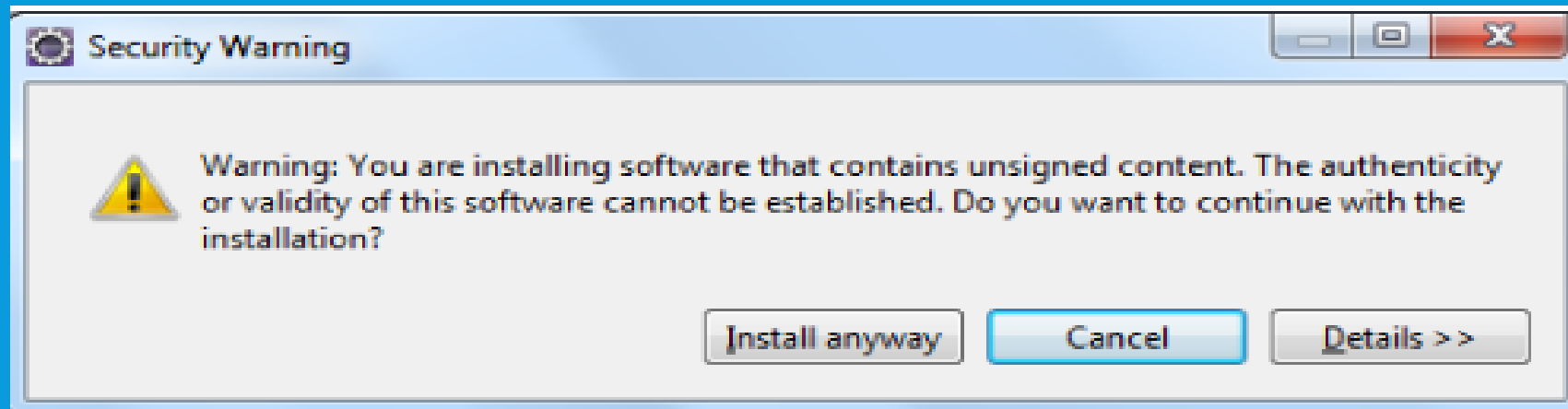
CHOOSE HIBERNATE TOOLS FROM THE LIST



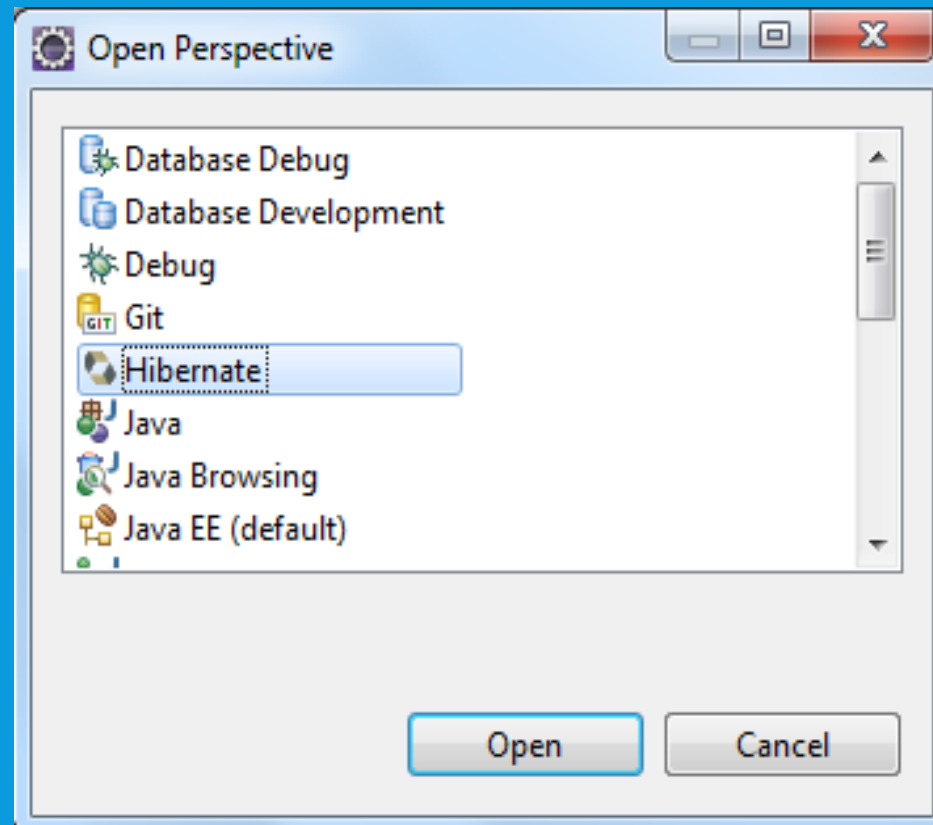
CONFIRM



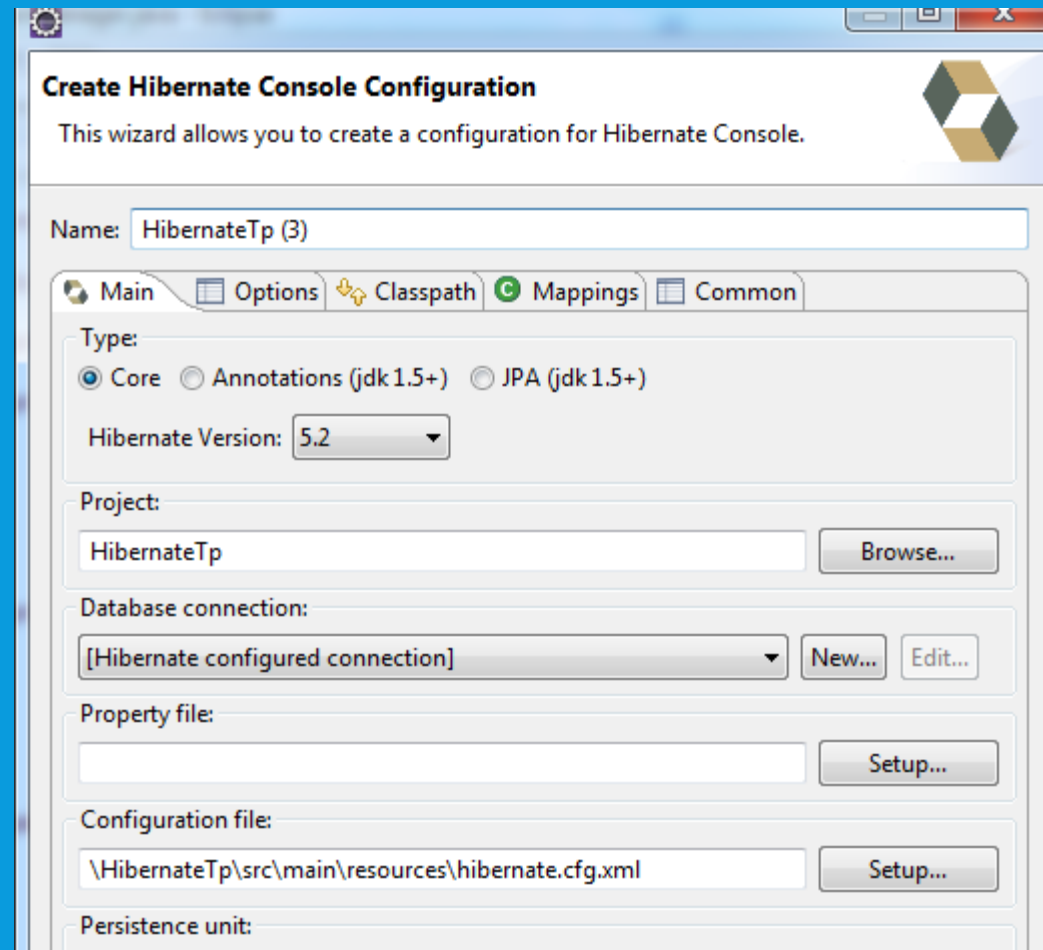
RESTART ECLIPSE



CLICK WINDOW > PERSPECTIVE > OPEN
PERSPECTIVE... AND CHOOSE HIBERNATE



CREATING A NEW HIBERNATE CONSOLE CONFIGURATION



Create Hibernate Console Configuration

This wizard allows you to create a configuration for Hibernate Console.

Name:

Main Options Classpath Mappings Common

Type:
☒ Core ☐ Annotations (jdk 1.5+) ☐ JPA (jdk 1.5+)

Hibernate Version:

Project:

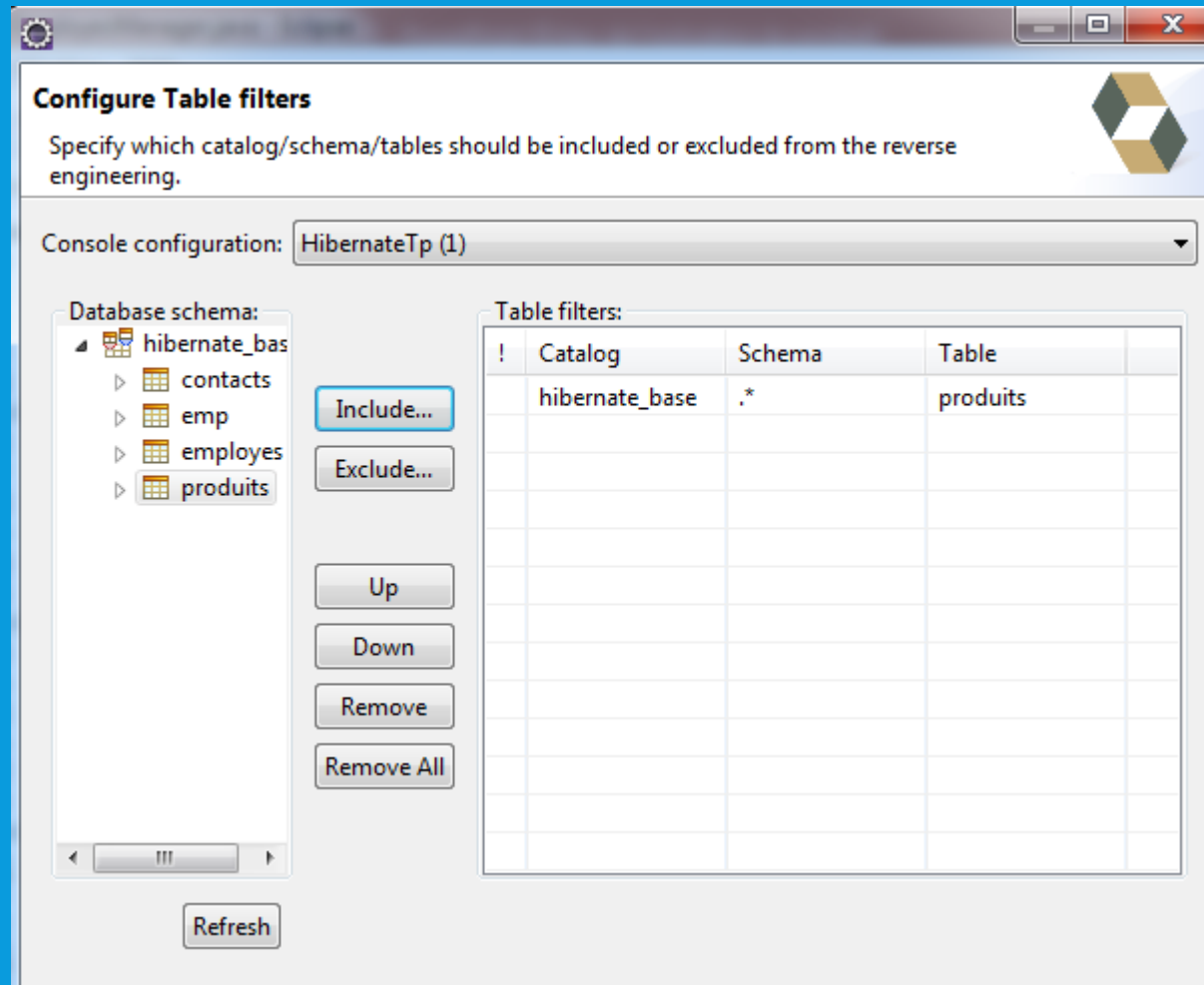
Database connection:

Property file:

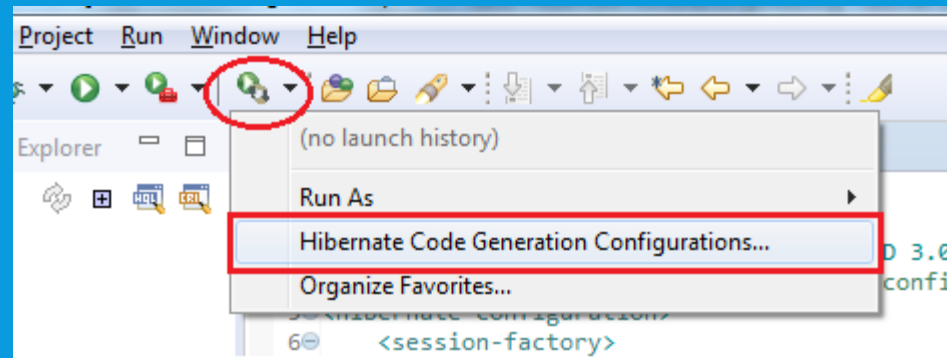
Configuration file:

Persistence unit:

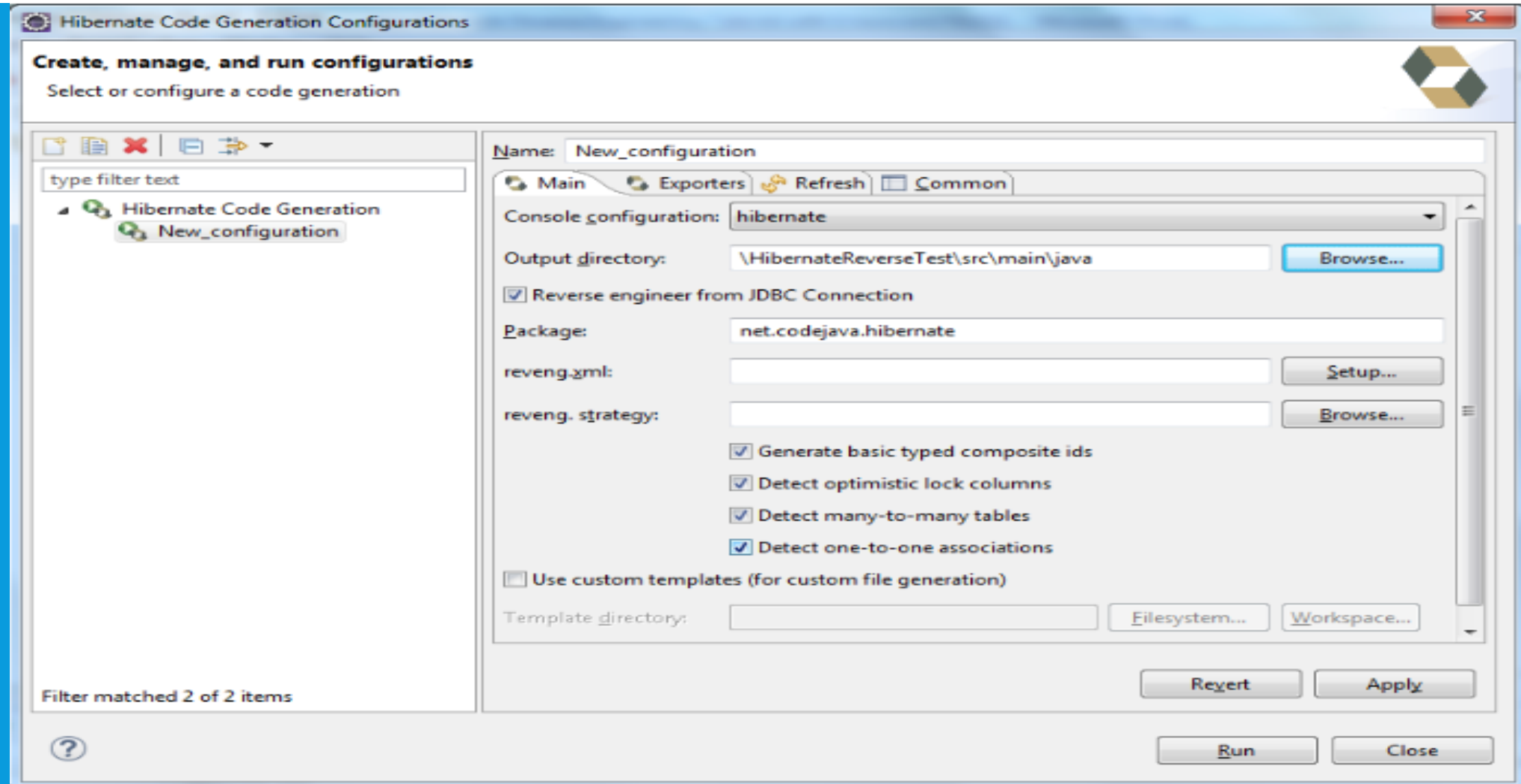
CREATE HIBERNATE REVERSE ENGINEERING



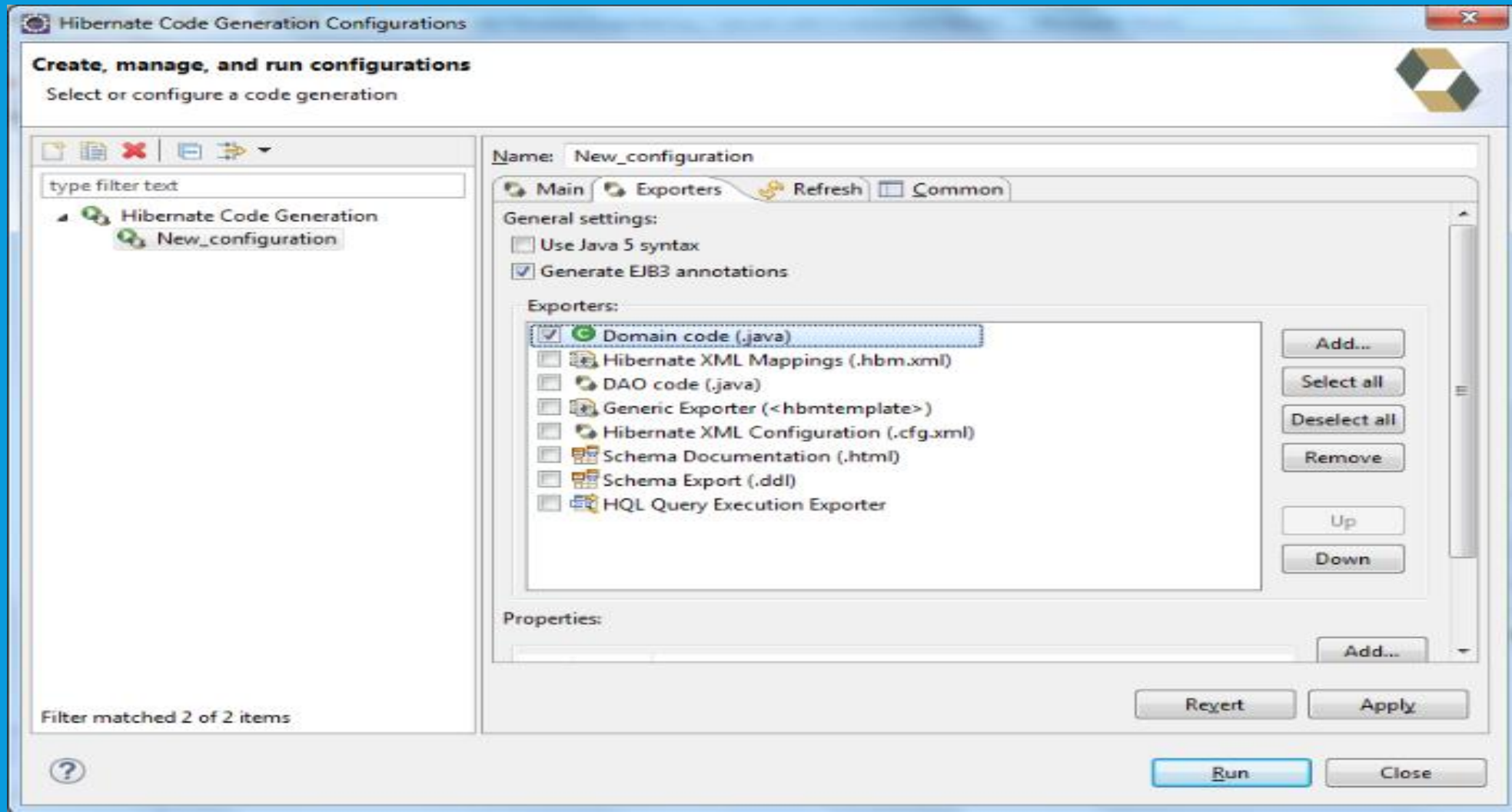
GENERATING MODEL CLASSES



MODEL CLASSES PROPERTIES



GENERATE EJB₃ ANNOTATIONS AND DOMAIN CODE (.JAVA)



TROIS TYPES DE REQUÊTES SELECT

- SQL
 - `sess.createSQLQuery().list()`
- HQL
 - `sess.createQuery().list()`
- Criteria
 - `sess.createCriteria().list()`

SQL NATIF

- La requête SQL la plus basique permet de récupérer une liste de (valeurs) scalaires.

```
sess.createSQLQuery("SELECT * FROM CATS").list();  
sess.createSQLQuery("SELECT ID, NAME, BIRTHDATE FROM CATS").list();
```

- Ces deux requêtes retourneront un tableau d'objets (Object[]) avec les valeurs scalaires de chacune des colonnes de la table CATS. **Hibernate** utilisera le **ResultSetMetadata** pour déduire l'ordre final et le type des valeurs scalaires retournées.

SQL NATIF (SUITE)

- Pour éviter l'overhead lié à **ResultSetMetaData** ou simplement pour être implicite dans ce qui est retournée , vous pouvez utiliser **addScalar()**.

```
sess.createSQLQuery("SELECT * FROM CATS")  
    .addScalar("ID", Hibernate.LONG)  
    .addScalar("NAME", Hibernate.STRING)  
    .addScalar("BIRTHDATE", Hibernate.DATE)
```

- Cette requête spécifie :
 - la chaîne de requêtes SQL
 - les colonnes et les types retournés

HQL

- Hibernate fournit un langage d'interrogation extrêmement puissant qui ressemble au SQL. HQL est totalement orienté objet, cernant des notions comme l'héritage, le polymorphisme et les associations.
- Il est sensible à la casse

HQL EXAMPLE

```
Query query = session.createQuery("from Stock where stockCode = :code ");  
query.setParameter("code", "7277");  
List list = query.list();
```

CRITERIA

- API d'interrogation par critères.
- Définition de critères de recherche

```
List resultat = session.createCriteria(Person.class)
    .add( Restrictions.like("name", "Manou%") )
    .add( Restrictions.or(
        Restrictions.eq( "age", new Integer(0) ),
        Restrictions.isNull("age")
    ) )
    .list();
```