

# Hibernate en Eclipse

Desarrollo de la herramienta de Mapeo  
Objeto Relacional (ORM)

Manual de instalación y desarrollo básico del ORM  
Hibernate para eclipse.

**Aurelio Rodríguez Fernández**  
**04/02/2014**

## Contenido

1. Introducción .....	4
Base de datos relacional (Wikipedia) .....	4
Programación Orientada a Objetos (Wikipedia) .....	5
Abstracción.....	5
Encapsulamiento.....	5
Modularidad.....	5
Principio de ocultación.....	5
Polimorfismo .....	6
Herencia .....	6
Recolección de basura.....	6
2. Concepto de Mapeo Objeto Relacional (Wikipedia).....	7
Características .....	7
Herramientas ORM .....	8
3. Arquitectura de Hibernate .....	8
SessionFactory (org.hibernate.SessionFactory) .....	9
Session (org.hibernate.Session) .....	9
Objetos y colecciones persistentes .....	9
Objetos y colecciones transitorios y separados .....	9
Transaction (org.hibernate.Transaction) .....	9
ConnectionProvider (org.hibernate.connection.ConnectionProvider).....	9
TransactionFactory (org.hibernate.TransactionFactory) .....	9
Extensión Interfaces .....	9
4. Instalación y configuración de Hibernate .....	10
4.1 Eclipse y JDK .....	10
4.2 Instalación del Plugin Hibernate en Eclipse .....	11
4.3 Configuración del Driver MySQL .....	12
4.4 Configuración de Hibernate .....	15
4.4.1 Hibernate Configuration File (cfg.xml) .....	15
4.4.2 Hibernate Console Configuration.....	18
4.4.3 Hibernate Reverse Engineering File (reveng.xml).....	19
4.4.4 Generar las clases de la base de datos.....	20
4.4.5 Mapear la consola Hibernate .....	24
4.5 Instalación de la librería Hibernate (archivos .jar) .....	26
4.6 Últimos pasos de configuración .....	31

5. Código Hibernate .....	32
6. Bibliografía .....	33

## 1. Introducción

Seguiremos los puntos del libro de acceso a datos de la editorial Garceta con alguna modificación debido a las actualizaciones de las herramientas necesarias. En este manual trataremos de explicar cómo acceder a una base de datos relacional utilizando el lenguaje orientado a objetos, para pasar de la lógica relacional a la lógica de POO necesitaremos una interfaz denominada ORM (Object Relational Mapping).

Gracias a esta herramienta transformaremos las tablas de nuestra base de datos en clases de nuestra aplicación, y los registros en objetos para manejar con facilidad.

Vamos a recuperar la definición y características de una base de datos relacional así como de la programación orientada a objetos:

### [Base de datos relacional \(Wikipedia\)](#)

Una Base de Datos Relacional, es una base de datos que cumple con el modelo relacional, el cual es el modelo más utilizado en la actualidad para implementar bases de datos ya planificadas. Permiten establecer interconexiones (relaciones) entre los datos (que están guardados en tablas), y a través de dichas conexiones relacionar los datos de ambas tablas, de ahí proviene su nombre: "Modelo Relacional". Tras ser postuladas sus bases en 1970 por Edgar Frank Codd, de los laboratorios IBM en San José (California), no tardó en consolidarse como un nuevo paradigma en los modelos de base de datos

### Características

1. Una Base de Datos Relacional se compone de varias tablas o relaciones.
2. No pueden existir dos tablas con el mismo nombre ni registro.
3. Cada tabla es a su vez un conjunto de registros (filas y columnas).
4. La relación entre una tabla padre y un hijo se lleva a cabo por medio de las claves primarias y ajenas (o foráneas).
5. Las claves primarias son la clave principal de un registro dentro de una tabla y éstas deben cumplir con la integridad de datos.
6. Las claves ajenas se colocan en la tabla hija, contienen el mismo valor que la clave primaria del registro padre; por medio de éstas se hacen las relaciones.

## Programación Orientada a Objetos ([Wikipedia](#))

La programación orientada a objetos o POO (OOP según sus siglas en inglés) es un paradigma de programación que usa los objetos en sus interacciones, para diseñar aplicaciones y programas informáticos. Está basado en varias técnicas, incluyendo herencia, cohesión, abstracción, polimorfismo, acoplamiento y encapsulamiento. Su uso se popularizó a principios de la década de los años 1990. En la actualidad, existe una gran variedad de lenguajes de programación que soportan la orientación a objetos.

### Características

Existe un acuerdo acerca de qué características contempla la "orientación a objetos". Las características siguientes son las más importantes:

#### Abstracción

Denota las características esenciales de un objeto, donde se capturan sus comportamientos. Cada objeto en el sistema sirve como modelo de un "agente" abstracto que puede realizar trabajo, informar y cambiar su estado, y "comunicarse" con otros objetos en el sistema sin revelar cómo se implementan estas características. Los procesos, las funciones o los métodos pueden también ser abstraídos, y, cuando lo están, una variedad de técnicas son requeridas para ampliar una abstracción. El proceso de abstracción permite seleccionar las características relevantes dentro de un conjunto e identificar comportamientos comunes para definir nuevos tipos de entidades en el mundo real. La abstracción es clave en el proceso de análisis y diseño orientado a objetos, ya que mediante ella podemos llegar a armar un conjunto de clases que permitan modelar la realidad o el problema que se quiere atacar.

#### Encapsulamiento

Significa reunir todos los elementos que pueden considerarse pertenecientes a una misma entidad, al mismo nivel de abstracción. Esto permite aumentar la cohesión de los componentes del sistema. Algunos autores confunden este concepto con el principio de ocultación, principalmente porque se suelen emplear conjuntamente.

#### Modularidad

Se denomina modularidad a la propiedad que permite subdividir una aplicación en partes más pequeñas (llamadas módulos), cada una de las cuales debe ser tan independiente como sea posible de la aplicación en sí y de las restantes partes. Estos módulos se pueden compilar por separado, pero tienen conexiones con otros módulos. Al igual que la encapsulación, los lenguajes soportan la modularidad de diversas formas.

#### Principio de ocultación

Cada objeto está aislado del exterior, es un módulo natural, y cada tipo de objeto expone una interfaz a otros objetos que especifica cómo pueden interactuar con los objetos de la clase. El aislamiento protege a las propiedades de un objeto contra su modificación por quien no tenga derecho a acceder a ellas; solamente los propios métodos internos del objeto pueden acceder a su estado. Esto asegura que otros objetos no puedan cambiar el estado interno de un objeto de manera inesperada, eliminando efectos secundarios e interacciones inesperadas. Algunos lenguajes relajan esto, permitiendo un acceso directo a los datos internos del objeto de una manera controlada y limitando el grado de abstracción. La aplicación entera se reduce a un agregado o rompecabezas de objetos.

### Polimorfismo

Comportamientos diferentes, asociados a objetos distintos, pueden compartir el mismo nombre; al llamarlos por ese nombre se utilizará el comportamiento correspondiente al objeto que se esté usando. O, dicho de otro modo, las referencias y las colecciones de objetos pueden contener objetos de diferentes tipos, y la invocación de un comportamiento en una referencia producirá el comportamiento correcto para el tipo real del objeto referenciado. Cuando esto ocurre en "tiempo de ejecución", esta última característica se llama asignación tardía o asignación dinámica. Algunos lenguajes proporcionan medios más estáticos (en "tiempo de compilación") de polimorfismo, tales como las plantillas y la sobrecarga de operadores de C++.

### Herencia

Las clases no están aisladas, sino que se relacionan entre sí, formando una jerarquía de clasificación. Los objetos heredan las propiedades y el comportamiento de todas las clases a las que pertenecen. La herencia organiza y facilita el polimorfismo y el encapsulamiento, permitiendo a los objetos ser definidos y creados como tipos especializados de objetos preexistentes. Estos pueden compartir (y extender) su comportamiento sin tener que volver a implementarlo. Esto suele hacerse habitualmente agrupando los objetos en clases y estas en árboles o enrejados que reflejan un comportamiento común. Cuando un objeto hereda de más de una clase se dice que hay herencia múltiple.

### Recolección de basura

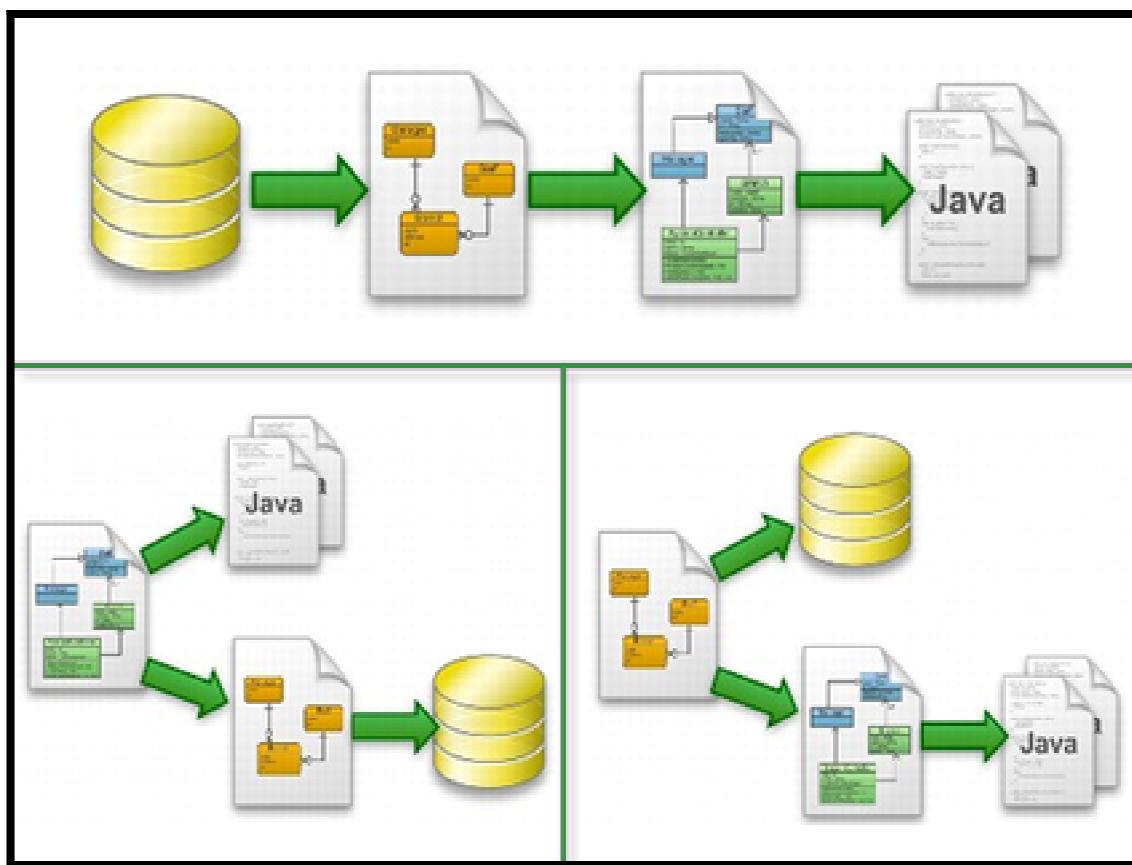
La recolección de basura o garbage collection es la técnica por la cual el entorno de objetos se encarga de destruir automáticamente, y por tanto desvincular la memoria asociada, los objetos que hayan quedado sin ninguna referencia a ellos. Esto significa que el programador no debe preocuparse por la asignación o liberación de memoria, ya que el entorno la asignará al crear un nuevo objeto y la liberará cuando nadie lo esté usando. En la mayoría de los lenguajes híbridos que se extendieron para soportar el Paradigma de Programación Orientada a Objetos como C++ u Object Pascal, esta característica no existe y la memoria debe desasignarse expresamente.

## 2. Concepto de Mapeo Objeto Relacional (Wikipedia)

El mapeo objeto-relacional (más conocido por su nombre en inglés, Object-Relational mapping, o sus siglas O/RM, ORM, y O/R mapping) es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y la utilización de una base de datos relacional, utilizando un motor de persistencia. En la práctica esto crea una base de datos orientada a objetos virtual, sobre la base de datos relacional. Esto posibilita el uso de las características propias de la orientación a objetos (básicamente herencia y polimorfismo). Hay paquetes comerciales y de uso libre disponibles que desarrollan el mapeo relacional de objetos, aunque algunos programadores prefieren crear sus propias herramientas ORM.

### Características

1. Ayudan a reducir el tiempo de desarrollo de software.
2. Abstracción de la base de datos.
3. Reutilización.
4. Permiten la producción de mejor código.
5. Son independientes de la Base de Datos.
6. Lenguaje propio para realizar las consultas.
7. Incentivan la portabilidad y escalabilidad de los programas de software.



## Herramientas ORM

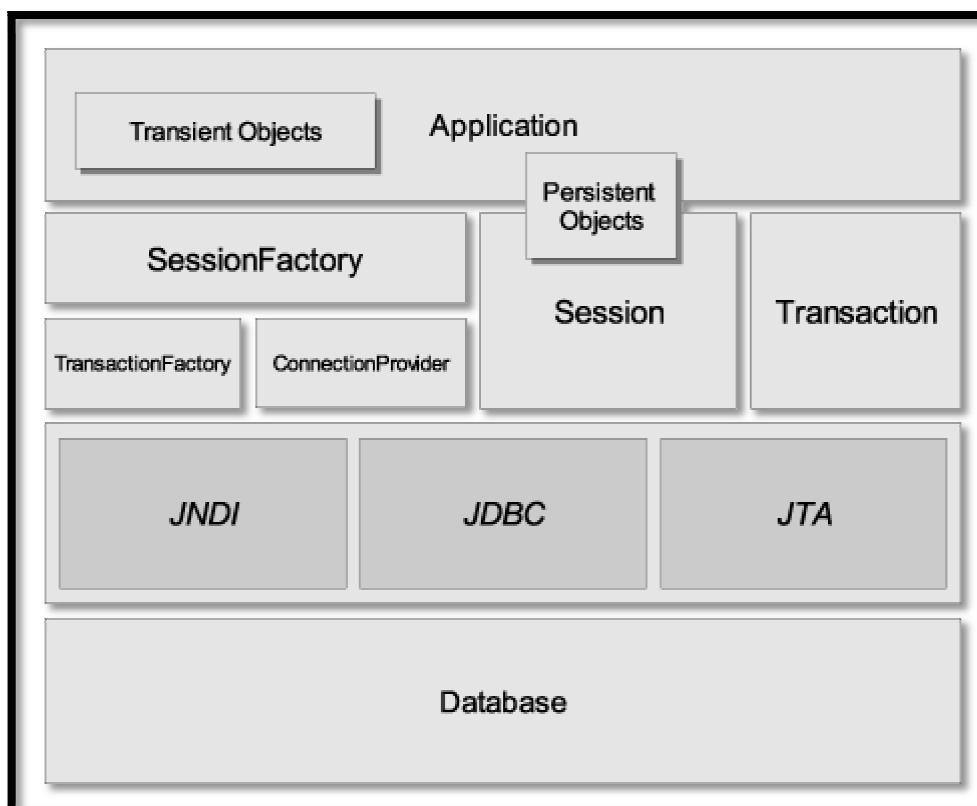
Algunas de las herramientas ORM más utilizadas son:

- Propel (PHP)
- Doctrine (PHP)
- Hibernate (Java)
- NHibernate (C#)
- ADOdb Active Record (PHP)
- LINQ (Microsoft)

En nuestro ejemplo trabajaremos con Hibernate bajo el entorno de Eclipse en una aplicación de consola.

### 3. Arquitectura de Hibernate

Hibernate parte de una filosofía de mapear objetos Java normales o más conocidos en la comunidad como "POJOs" (Plain Old Java Objects). Para almacenar y recuperar estos objetos de la base de datos, el desarrollador debe mantener una conversación con el motor de Hibernate mediante un objeto especial que es la **sesión** (clase **Session**) (equiparable al concepto de conexión de JDBC). Igual que con las conexiones JDBC hemos de crear y cerrar sesiones. La arquitectura Hibernate se muestra a continuación:



## [SessionFactory \(org.hibernate.SessionFactory\)](#)

Un caché threadsafe (inmutable) de mapeos compilados para una sola base de datos. Una fábrica de Session y un cliente de ConnectionProvider, SessionFactory puede mantener un caché opcional (de segundo nivel) de datos reusables entre transacciones a nivel de proceso o de clúster.

## [Session \(org.hibernate.Session\)](#)

Un objeto mono-hebra, de corta vida que representa una conversación entre la aplicación y el almacenamiento persistente. Envuelve una conexión JDBC y es una fábrica de Transaction. Session mantiene un caché requerido de primer nivel de objetos persistentes, que se utiliza cuando se navega el gráfico de objetos o mientras se buscan objetos por identificador.

## [Objetos y colecciones persistentes](#)

Objetos de corta vida, mono-hebra contienen un estado persistente así como una funcionalidad empresarial. Estos pueden ser JavaBeans/POJOs normales. Estos se encuentran asociados con exactamente una Session. Tan pronto como la Session se cierre, serán separados y estarán libres para utilizarlos en cualquier capa de aplicación, (por ejemplo, directamente como objetos de transferencia de datos hacia y desde la presentación).

## [Objetos y colecciones transitorios y separados](#)

Instancias de clases persistentes que no se encuentran actualmente asociadas con una Session. Pueden haber sido instanciadas por la aplicación y aún no haber sido persistidas, o pueden haber sido instanciadas por una Session cerrada.

## [Transaction \(org.hibernate.Transaction\)](#)

(Opcional) Un objeto de corta vida, mono-hebra que la aplicación utiliza para especificar unidades atómicas de trabajo. Abstacta la aplicación de las transacciones subyacentes JDBC, JTA o CORBA. En algunos casos, una Session puede extenderse sobre varias Transactiones. Sin embargo, la demarcación de la transacción, ya sea utilizando la API subyacente o Transaction, nunca es opcional.

## [ConnectionProvider \(org.hibernate.connection.ConnectionProvider\)](#)

(Opcional) Una fábrica y pool de conexiones JDBC. Abstacta a la aplicación del Datasource o DriverManager subyacente. No se expone a la aplicación, pero puede ser extendido/implementado por el desarrollador.

## [TransactionFactory \(org.hibernate.TransactionFactory\)](#)

(Opcional) Una fábrica de instancias de Transaction. No se expone a la aplicación pero puede ser extendido/implementado por el desarrollador.

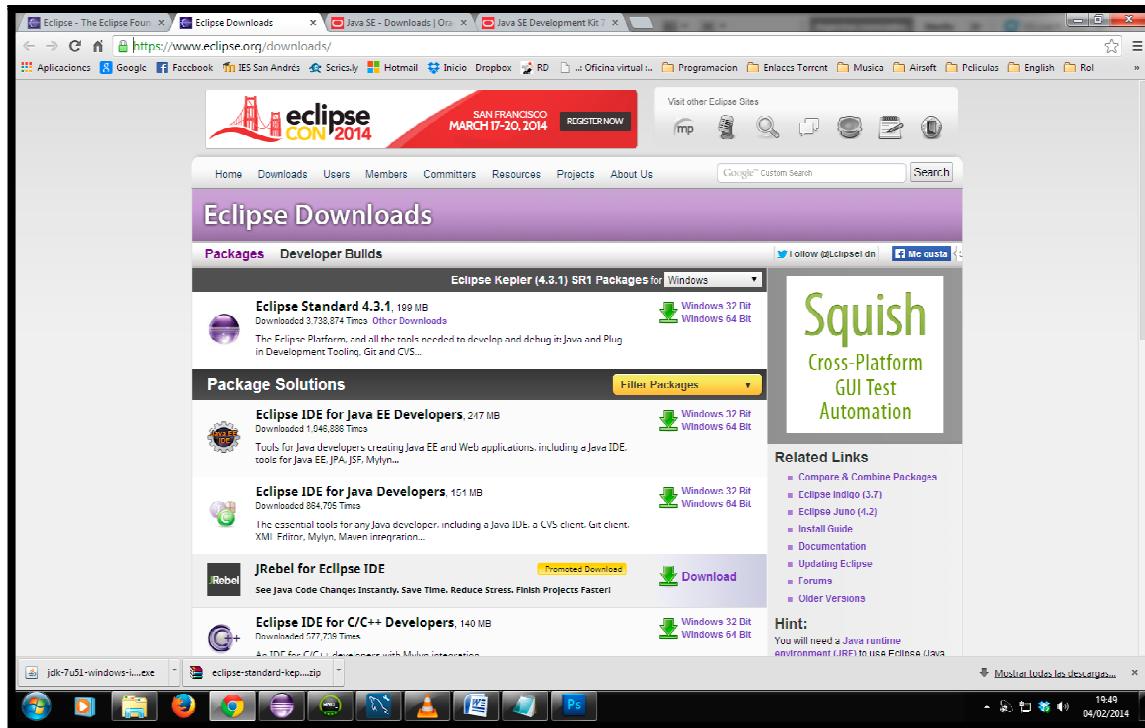
## [Extensión Interfaces](#)

Hibernate ofrece un rango de interfaces de extensión opcionales que puede implementar para personalizar el comportamiento de su capa de persistencia. Para obtener más detalles, vea la documentación de la API.

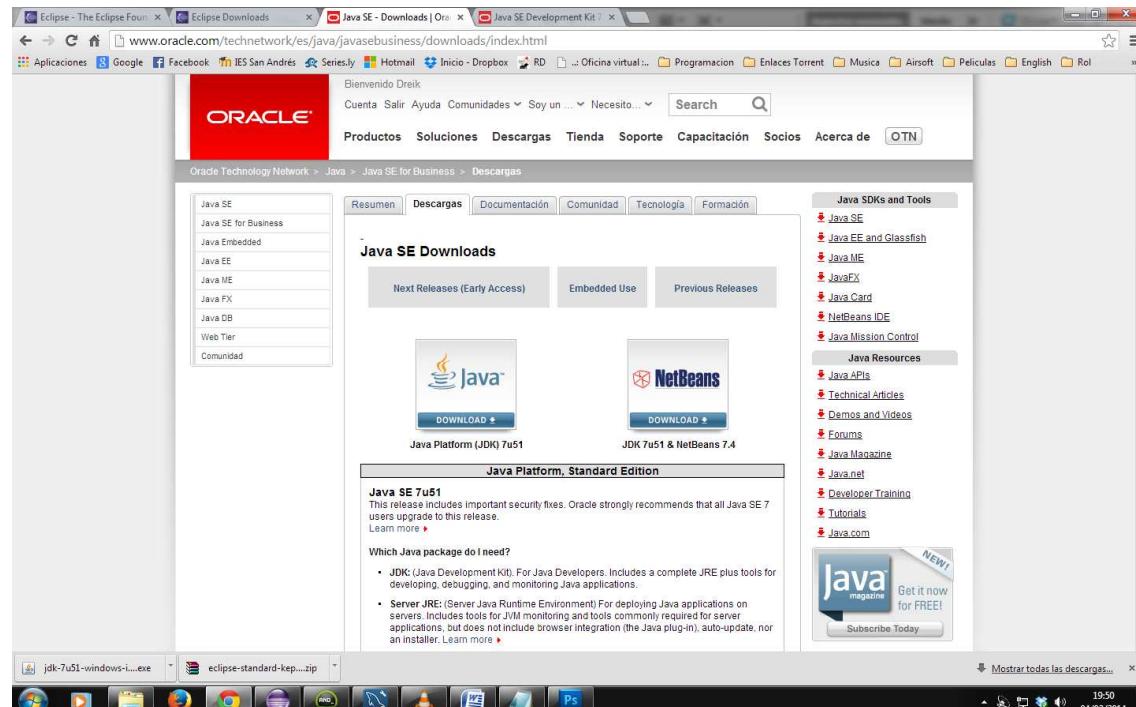
## 4. Instalación y configuración de Hibernate

### 4.1 Eclipse y JDK

Lo primero será descargar e instalar la última versión de eclipse (Kepler en estos momentos) y el JDK (Java development Kit / Kit de desarrollo para Java).



Url: [Enlace a la web de eclipse](https://www.eclipse.org/downloads/)



Url: [Enlace a la web del JDK](http://www.oracle.com/technetwork/es/java/javasebusiness/downloads/index.html)

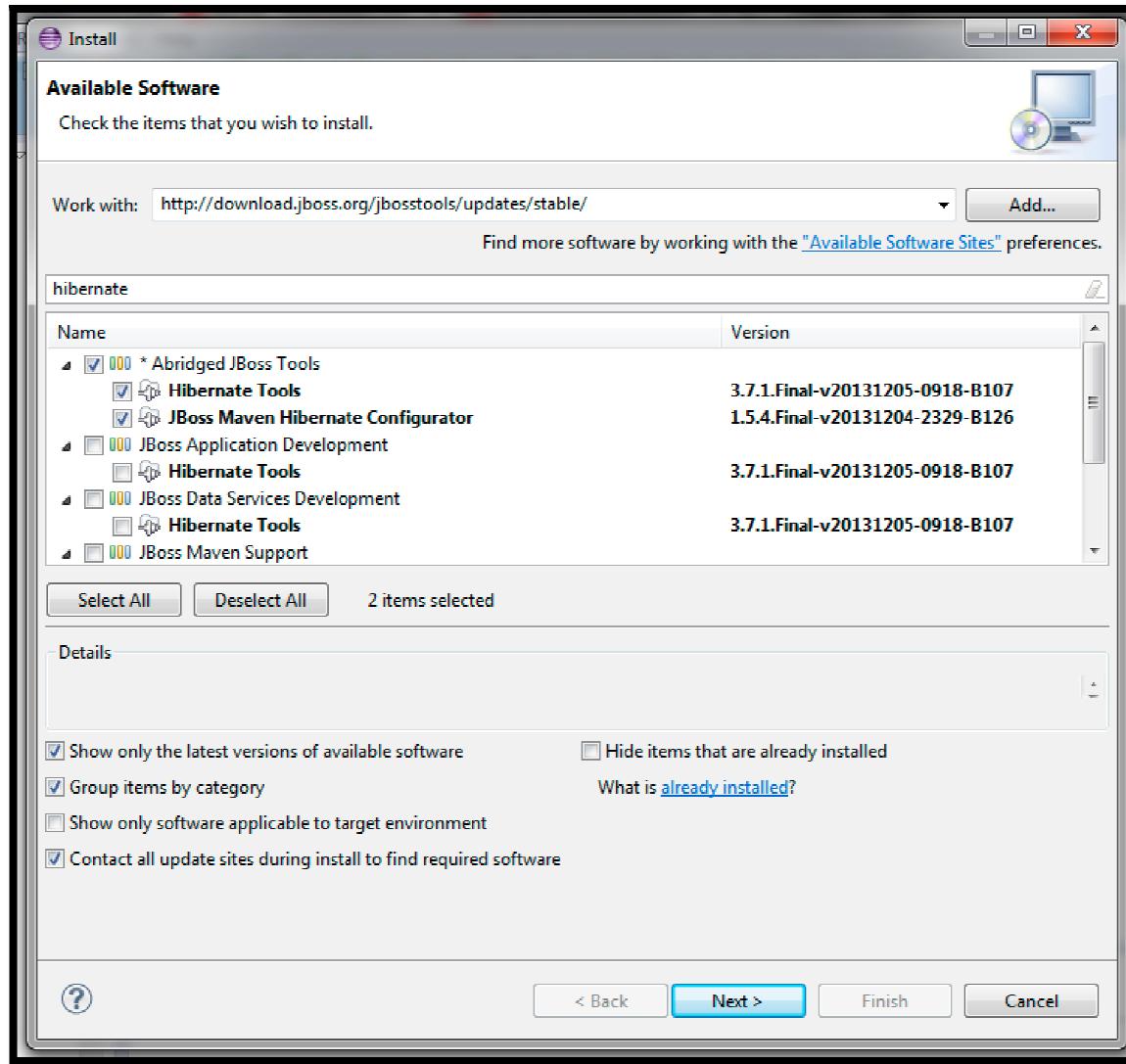
## 4.2 Instalación del Plugin Hibernate en Eclipse

Tras realizar los pasos anteriores accedemos al eclipse y nos dirigimos a:

*Help --> Install New Software*

en el menú horizontal superior de la aplicación y copiaremos en la caja de texto:

Work with: <http://download.jboss.org/jbosstools/updates/stable/>



Una vez hayamos completado la instalación del plugin ya dispondremos de la capacidad para crear ficheros Hibernate dentro de nuestras aplicaciones.

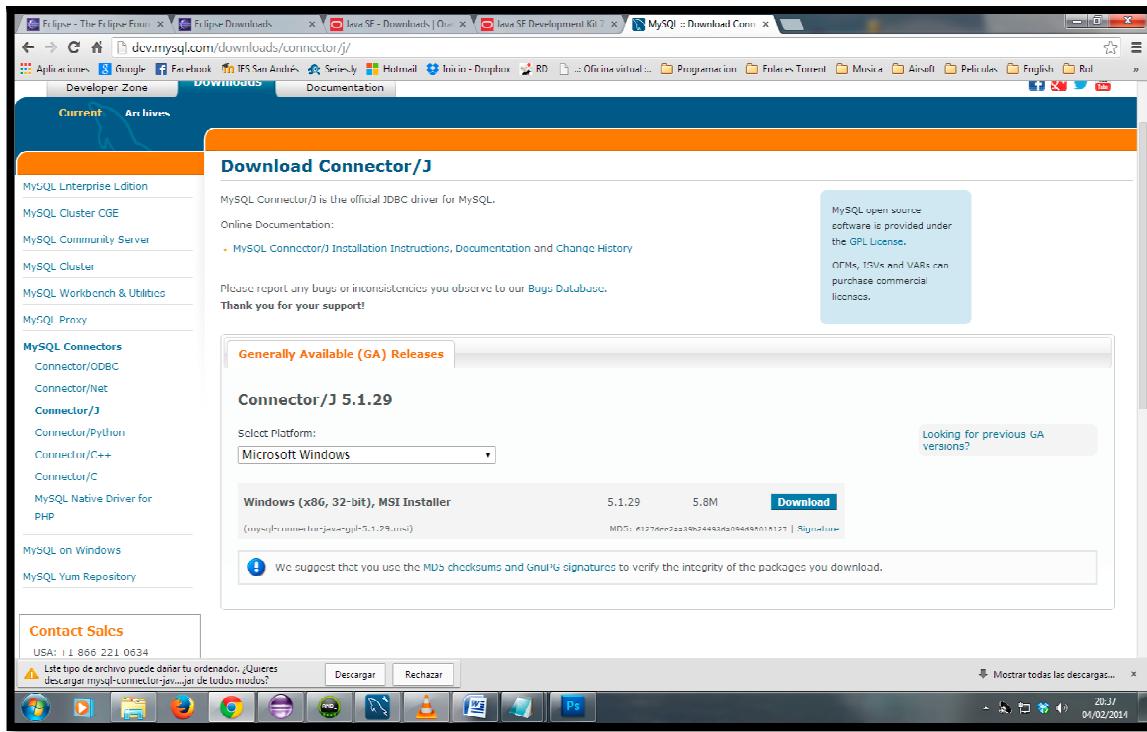
### 4.3 Configuración del Driver MySQL

Siguiendo las instrucciones del libro de acceso a datos, nuestro siguiente paso será descargar el controlador para MySQL en nuestro caso. Si queremos conectar con una base de datos diferente deberemos descargar el driver apropiado.

Para continuar, accedemos a la siguiente dirección:

Url: [Enlace para descargar el conector mysql java](http://dev.mysql.com/downloads/connector/j/)

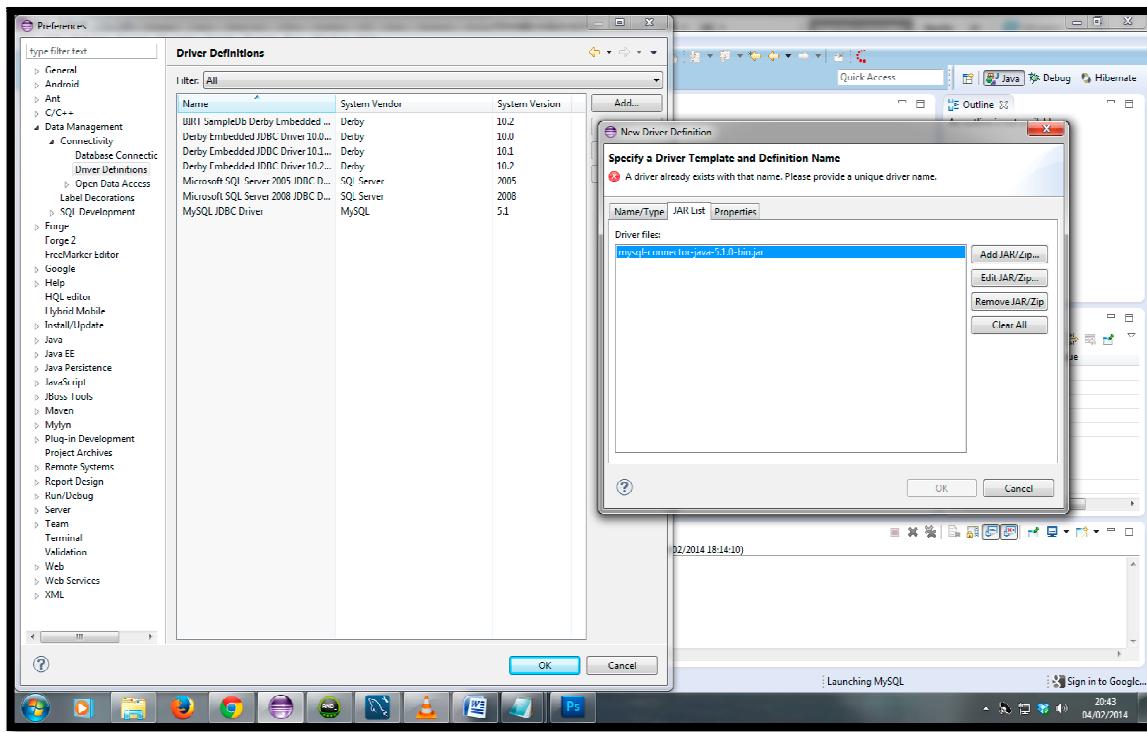
No es la dirección oficial ya que mysql pertenece a oracle y nos pedirá un usuario y contraseña para realizar las descargas.



Una vez dispongamos del archivo con la extensión .jar accedemos a la siguiente ventana en eclipse:

*Window --> Preferences --> Data Management --> Connectivity --> Driver Definitions*

Y se pulsa el botón Add

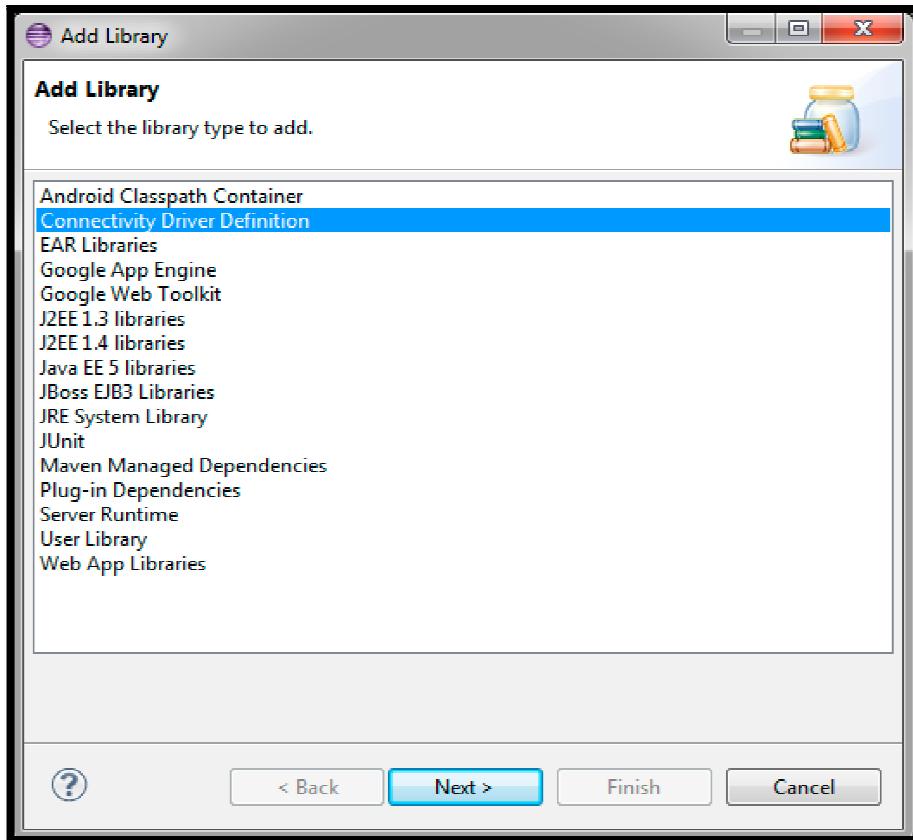


Realizaremos los siguientes pasos:

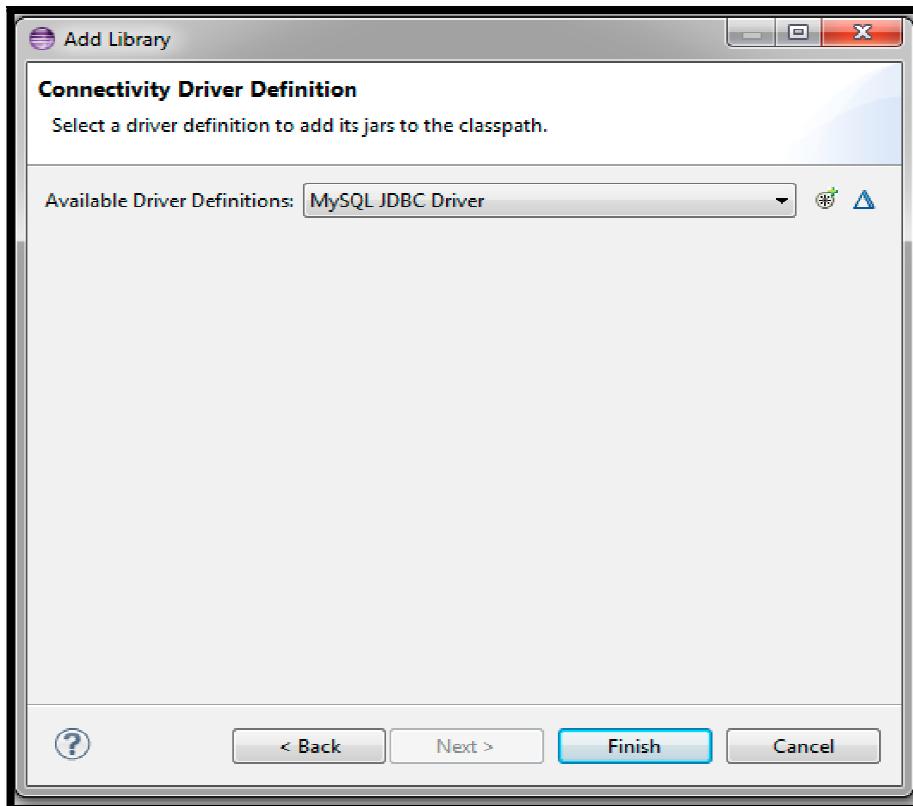
1. Desde la pestaña Name/Type se selecciona MySQL JDBC Driver en la versión 5.1.
2. Desde la pestaña Jar List pulsamos el botón Add Jar/Zip para localizar el archivo .jar  
(Nota: Deberíamos dejar el archivo en la carpeta de eclipse).
3. Pulsamos Ok y ya tendremos listo nuestro archivo de conexión.

Ahora para añadir la librería de conexión a nuestro entorno, pulsaremos el botón derecho del ratón sobre el mismo y nos dirigimos a:

*Build Path --> Add Libraries*



Y seleccionamos el driver que configuramos previamente en nuestro entorno de eclipse:



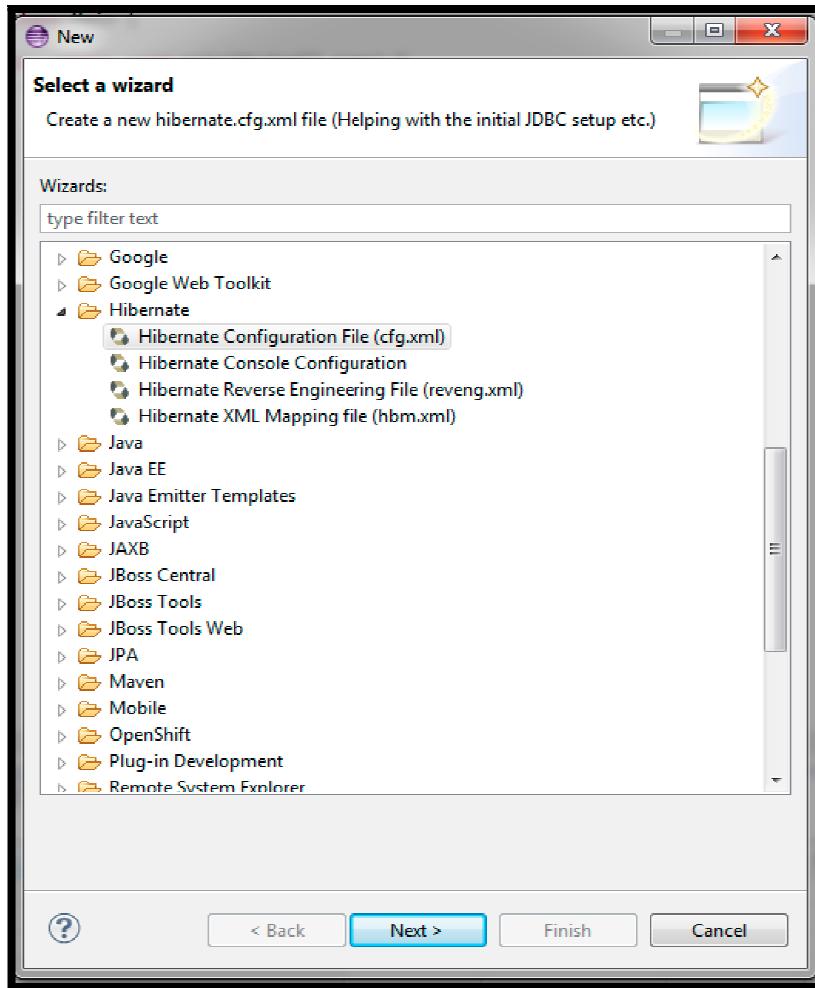
## 4.4 Configuración de Hibernate

### 4.4.1 Hibernate Configuration File (cfg.xml)

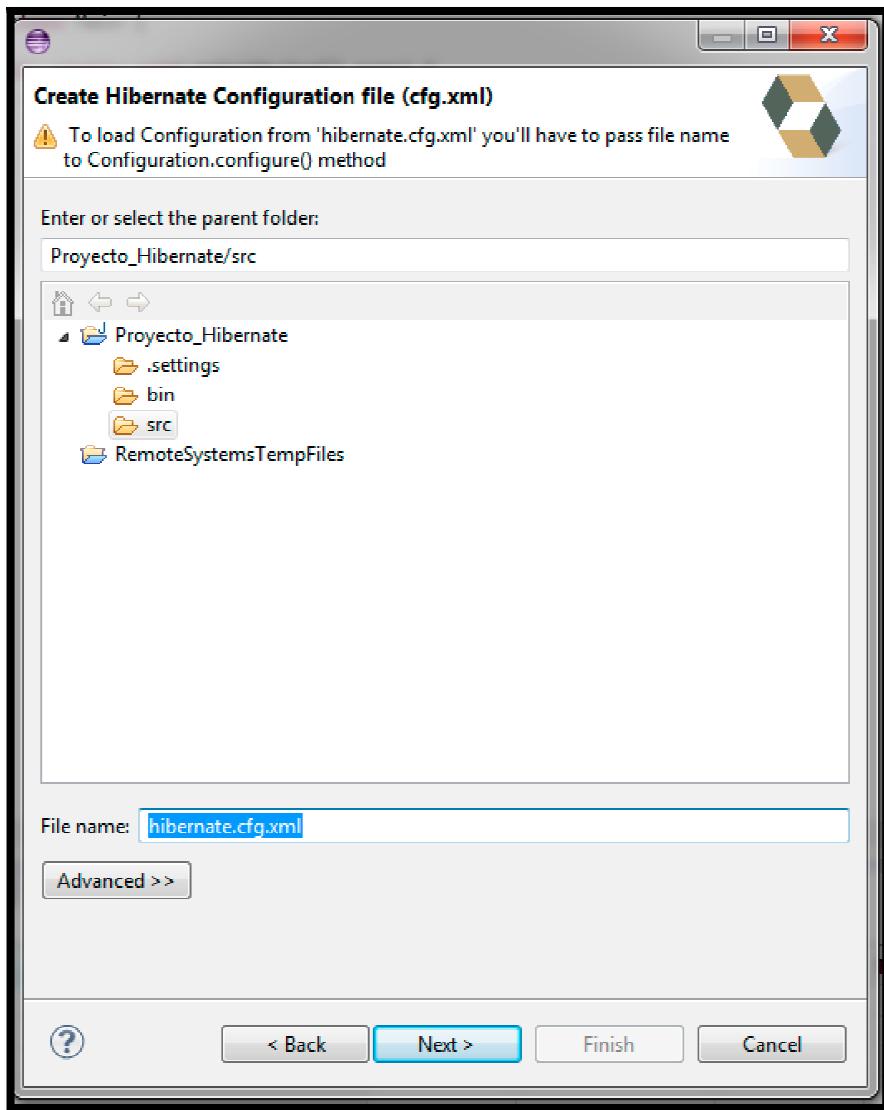
Una vez tenemos el driver MySQL en nuestro proyecto hemos de crear el fichero de configuración de Hibernate hibernate.cfg.xml. Seleccionamos nuestro proyecto, pulsamos el botón derecho del ratón y pulsamos sobre:

New --> Other --> Hibernate --> Hibernate Configuration File (cfg.xml)

Este fichero tiene los datos necesarios para realizar nuestra conexión a la base de datos.

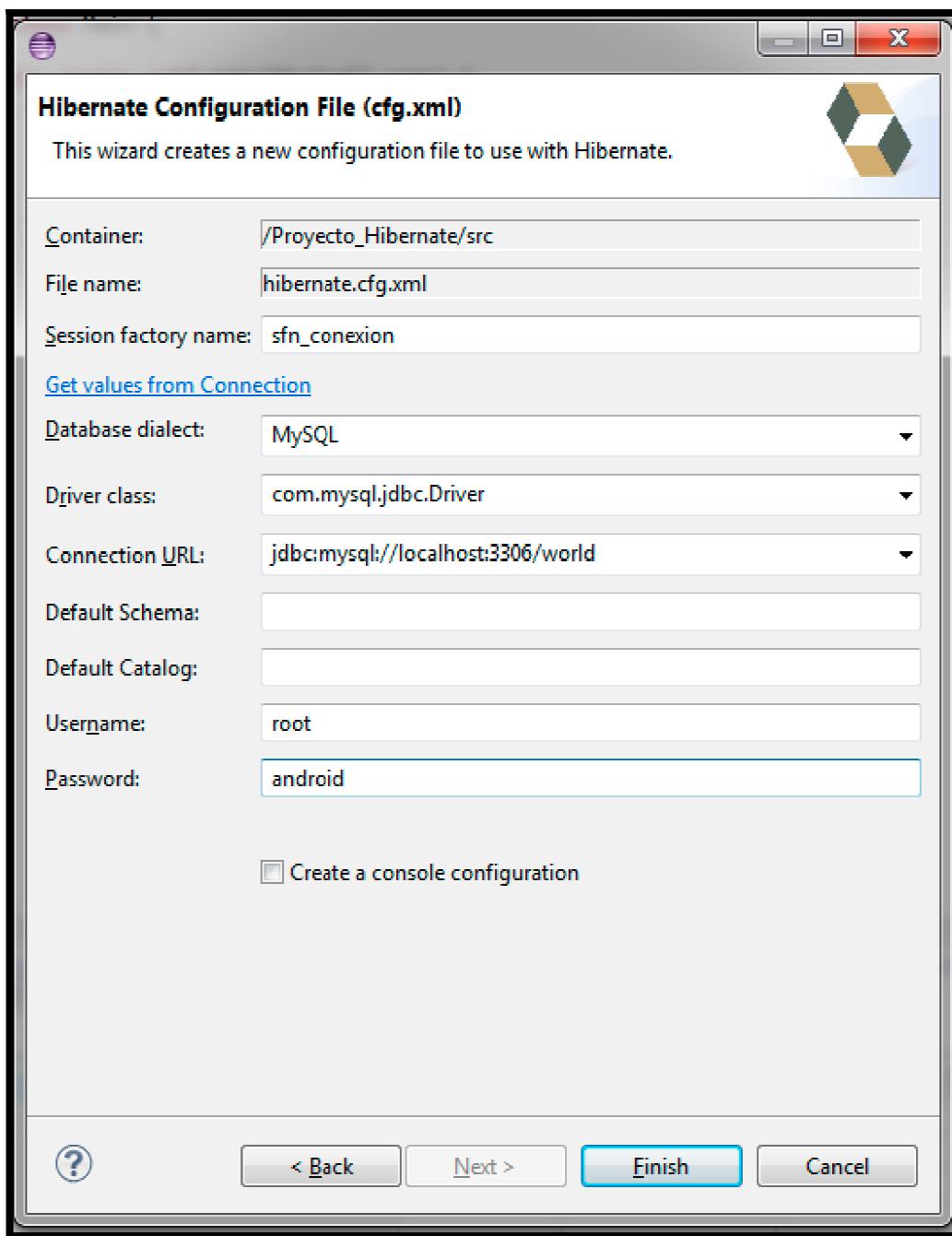


Seleccionamos el nombre del fichero de configuración, como normal se deja el nombre por defecto.



Configuramos los parámetros del mismo:

- Session factory name: Nombre del archivo session factory
- Database dialect: MySQL (Nuestra base de datos en el ejemplo)
- Driver class: com.mysql.jdbc.Driver (Esta cadena se selecciona)
- Connection URL: jdbc:mysql://localhost:3306/world  
La cadena de conexión con la dirección, puerto y nombre de la base de datos
- Username: root  
El nombre de usuario con permisos de la base, en el ejemplo usaremos root pero no es aconsejable utilizar nunca el superusuario para este tipo de conexiones.
- Password: android (Clave generada para el superusuario en la base de datos)



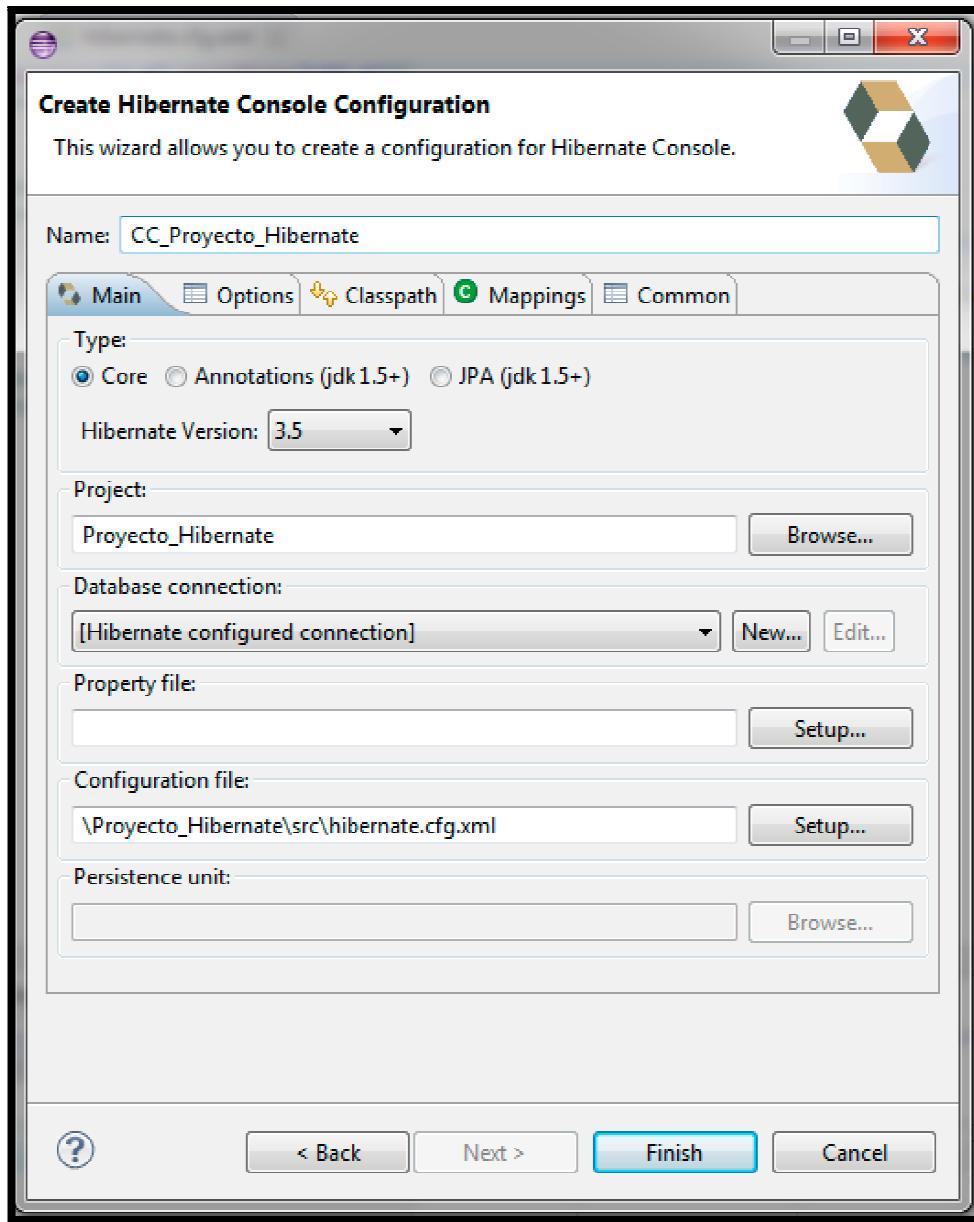
#### 4.4.2 Hibernate Console Configuration

Ahora crearemos la consola de configuración para realizar nuestras pruebas bajo el entorno Hibernate:

Para ello, accedemos como antes a:

New --> Other --> Hibernate --> Hibernate Console Configuration

En el archivo de configuración seleccionamos el hibernate.cfg.xml que generamos previamente.

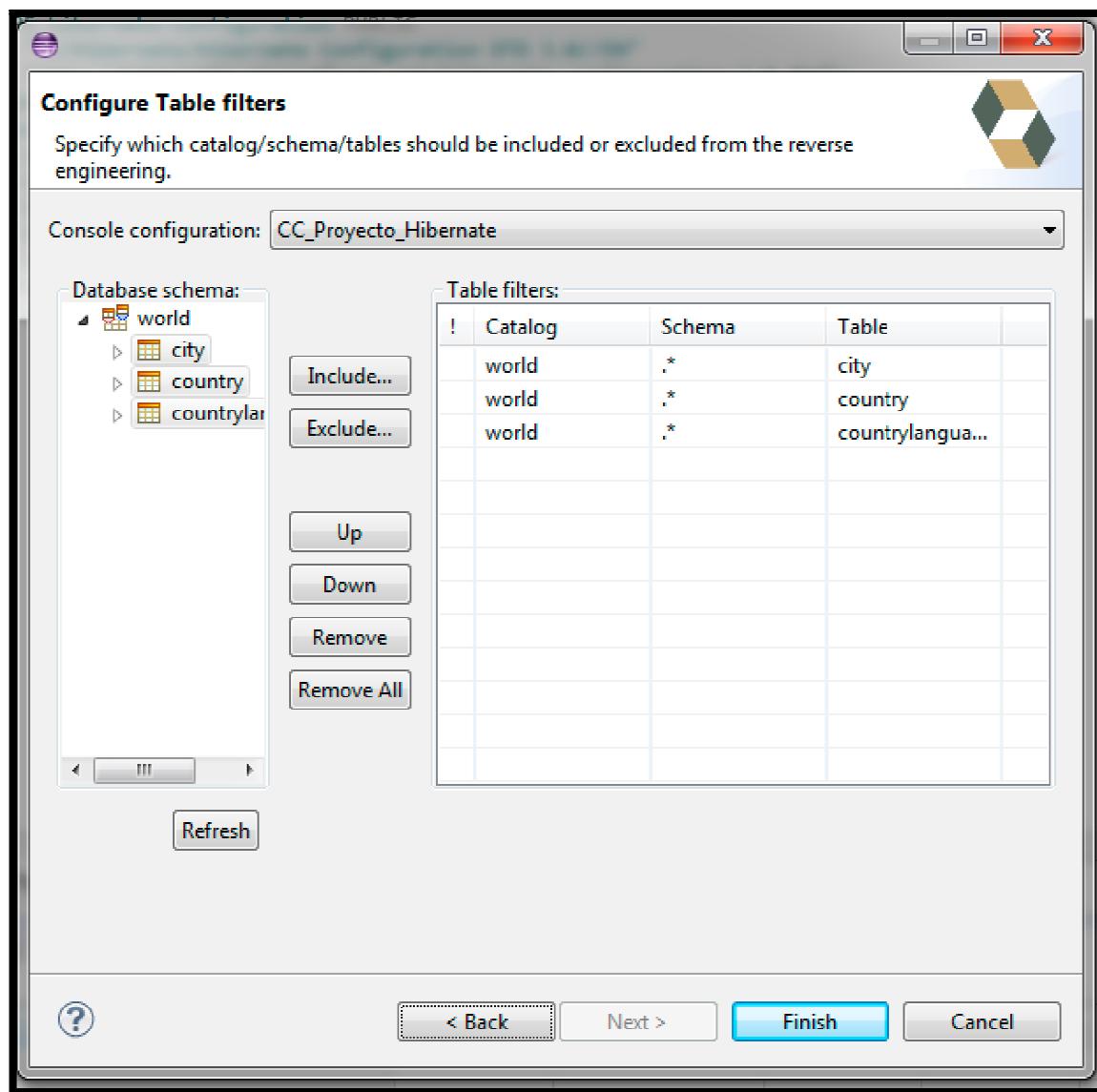


#### 4.4.3 Hibernate Reverse Engineering File (reveng.xml)

Por último, hemos de crear el fichero XML Hibernate Reverse Engineering que es el encargado de crear las clases de nuestras tablas MySQL. Pulsamos botón derecho del ratón y accedemos a:

New --> Other --> Hibernate --> Hibernate Reverse Engineering

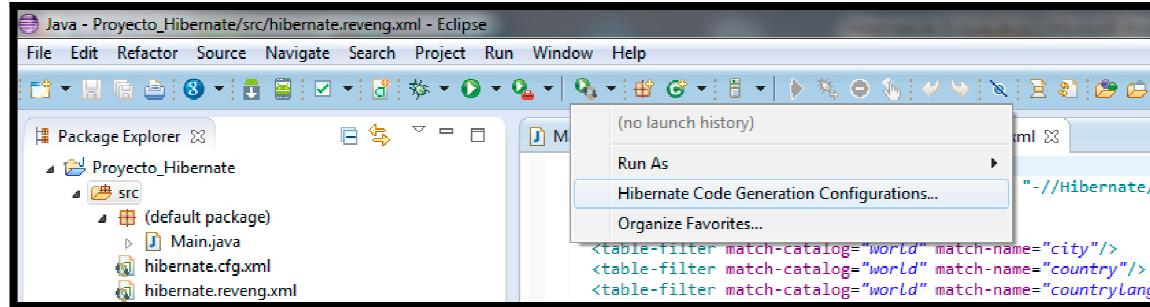
Nos aparecerá la siguiente ventana, en ella seleccionamos la consola de configuración que creamos en el paso anterior y las tablas de nuestra base para incluir en el archivo reveng.xml).



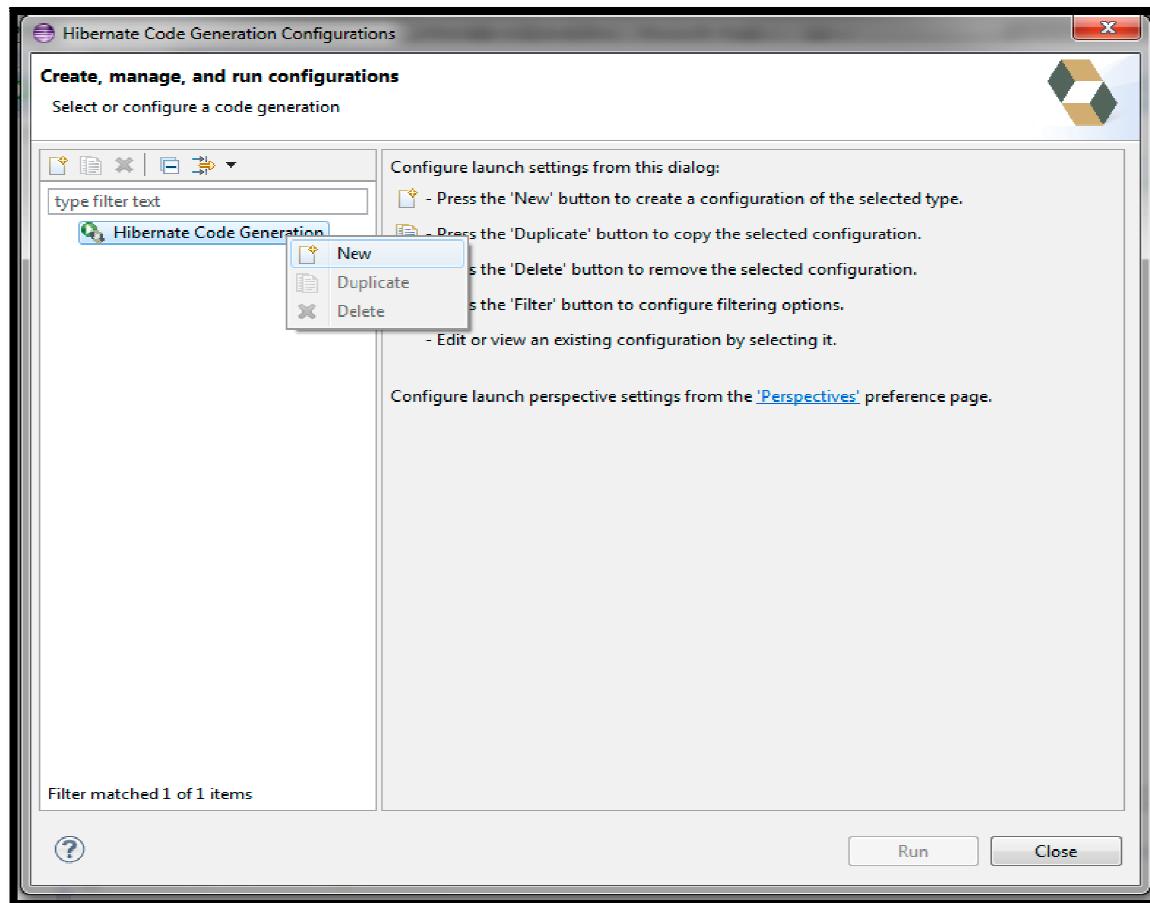
#### 4.4.4 Generar las clases de la base de datos

En este paso generamos las clases de nuestras tablas de la base de datos "world" de nuestro MySQL.

Seleccionamos la opción: Hibernate Code Generation Configurations...

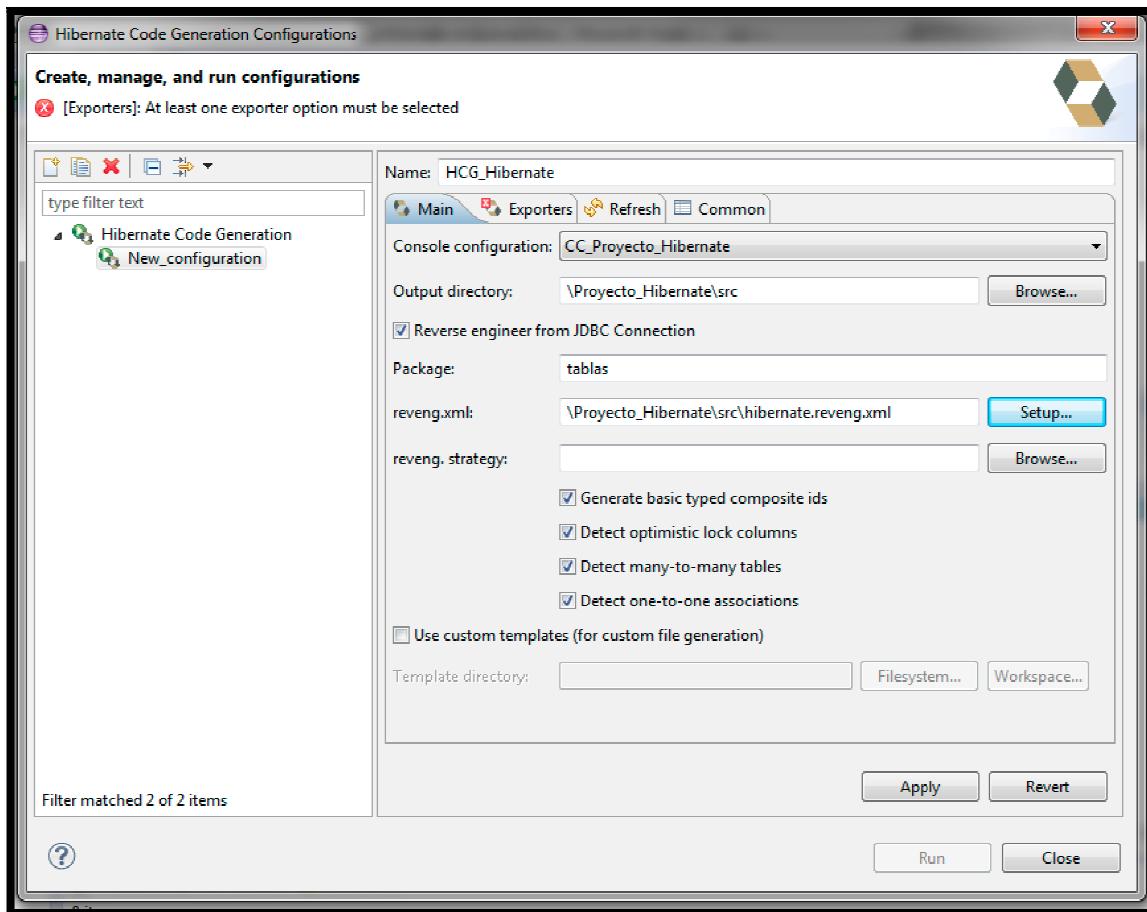


Una vez seleccionado, pulsamos botón derecho sobre "Hibernate Code Generation" y pulsamos en "New".



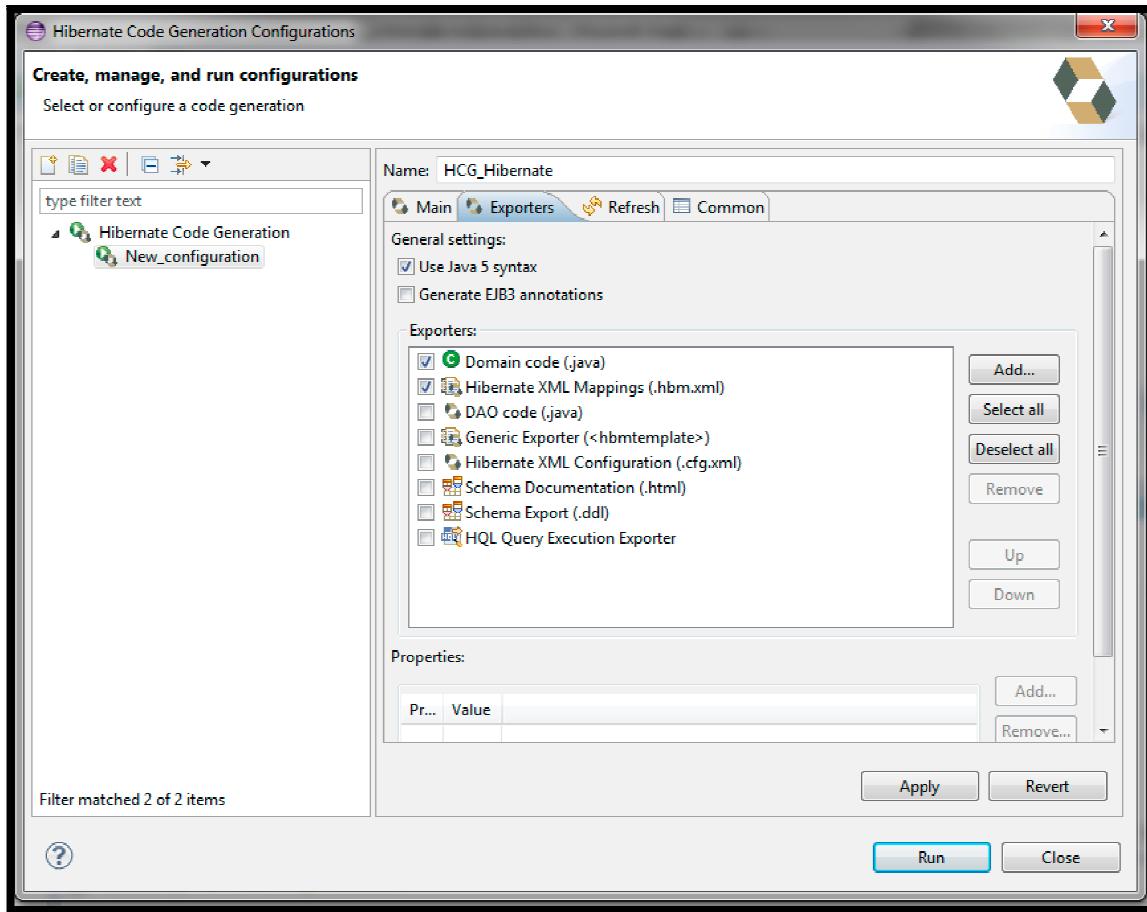
Configuramos:

- Name: Nombre de nuestro generador de código
- Console Configuration: La creada previamente
- Output directory: En esta parte manuales recomiendan dejar la ruta del src por defecto
- Marcamos el checkbox (Reverse engineer from JDBC Connection)
- Package: nombre del paquete para nuestras clases
- Reveng.xml: archivo de ingeniería inversa

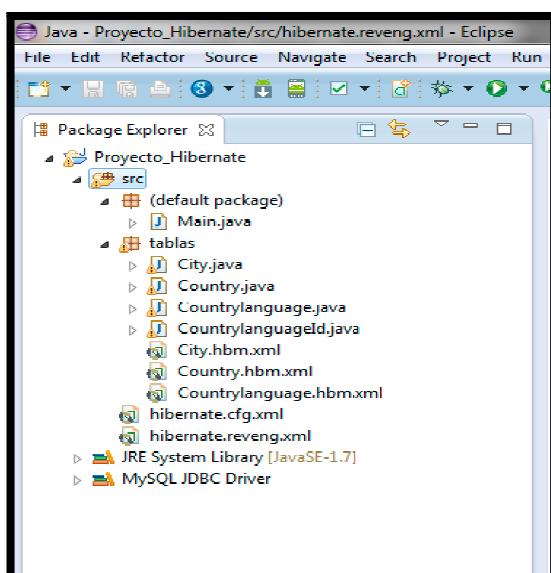


Seleccionamos la pestaña Exporters y seguimos la configuración:

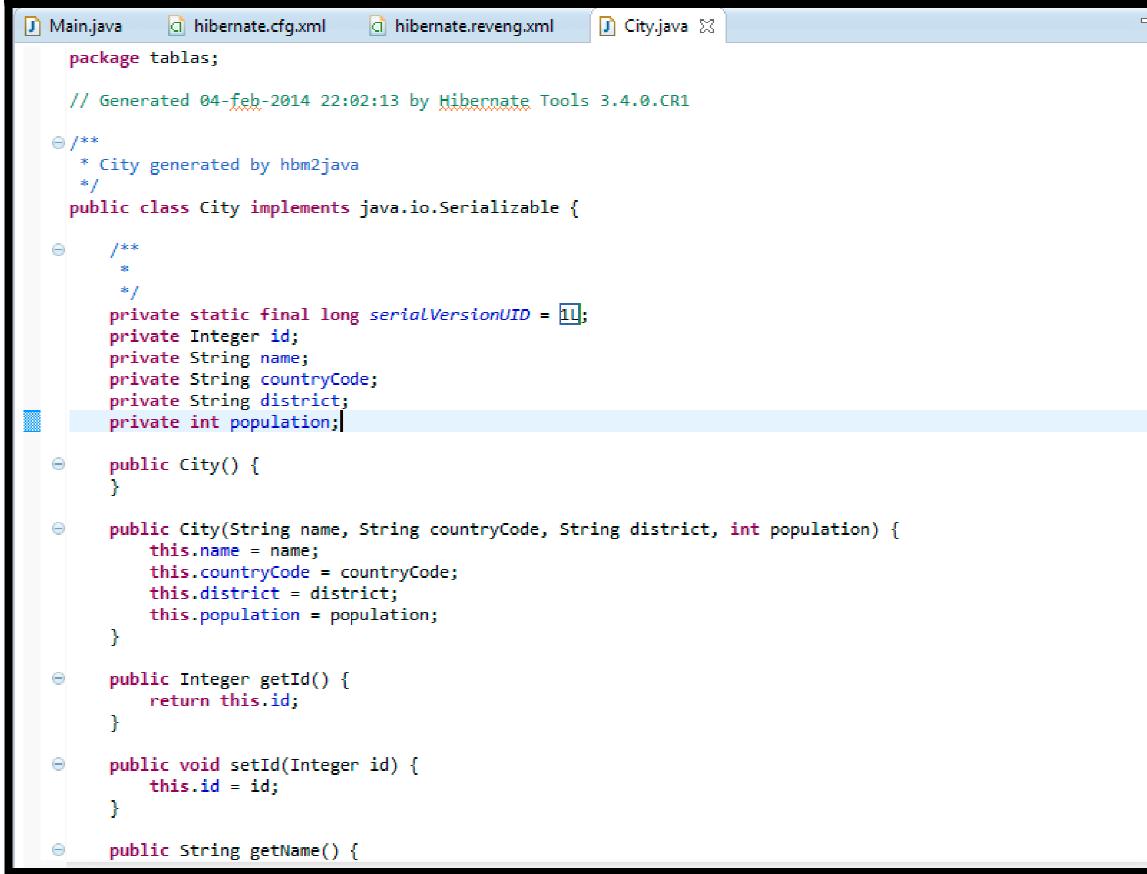
- Marcamos el checkbox (Use Java 5 syntax)
- Domain code(.java)
- Hibernate XML Mappings (.hbm.xml)



Al finalizar pulsamos "Apply" y "Run" en ese orden, así obtendremos nuestras clases de las tablas de la Base de datos.



El último paso en este apartado será añadir la opción serializable a todas nuestras clases generadas de forma automática. Ejemplo:



```
package tablas;

// Generated 04-feb-2014 22:02:13 by Hibernate Tools 3.4.0.CR1

/**
 * City generated by hbm2java
 */
public class City implements java.io.Serializable {

    /**
     *
     */
    private static final long serialVersionUID = 1L;
    private Integer id;
    private String name;
    private String countryCode;
    private String district;
    private int population;

    public City() {
    }

    public City(String name, String countryCode, String district, int population) {
        this.name = name;
        this.countryCode = countryCode;
        this.district = district;
        this.population = population;
    }

    public Integer getId() {
        return this.id;
    }

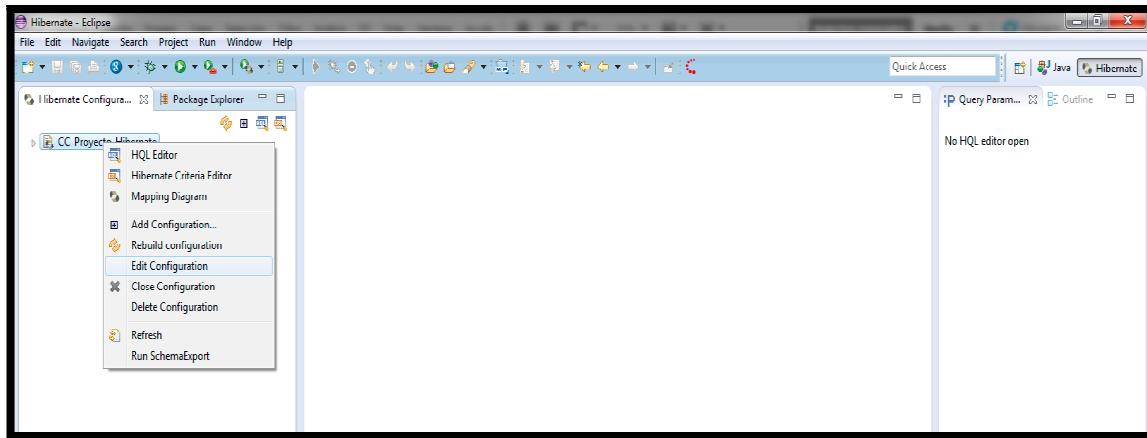
    public void setId(Integer id) {
        this.id = id;
    }

    public String getName() {
```

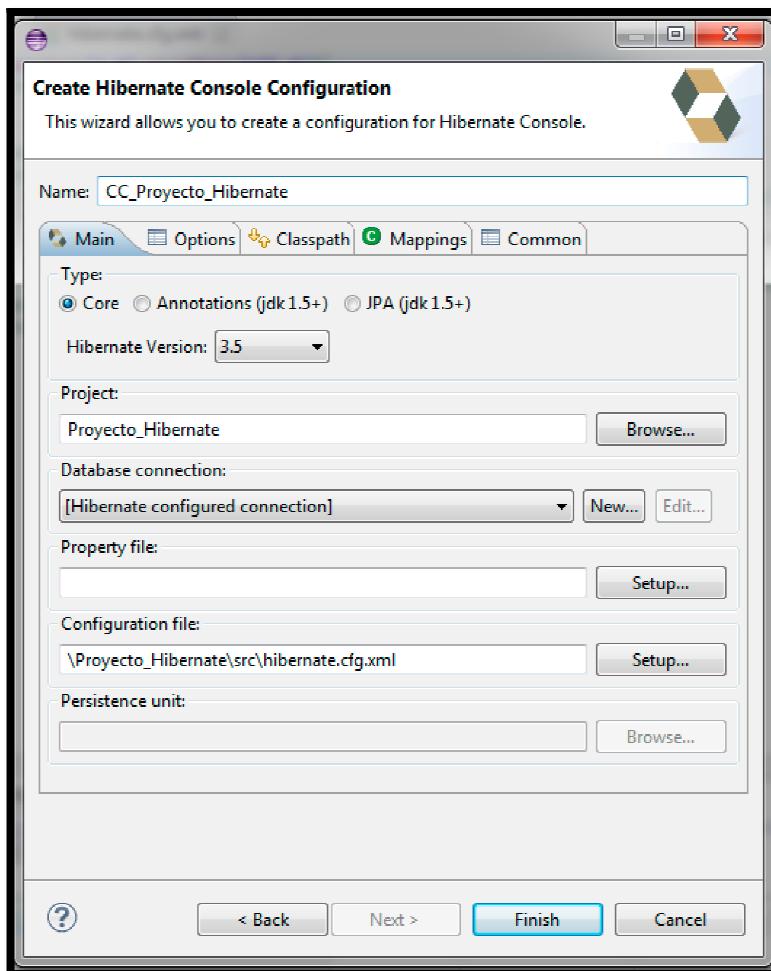
#### 4.4.5 Mapear la consola Hibernate

Este paso es importante ya que el libro se lo salta y requiere mapear la Consola Hibernate para que pueda realizar las consultas a nuestra base de datos bajo la perspectiva Hibernate .

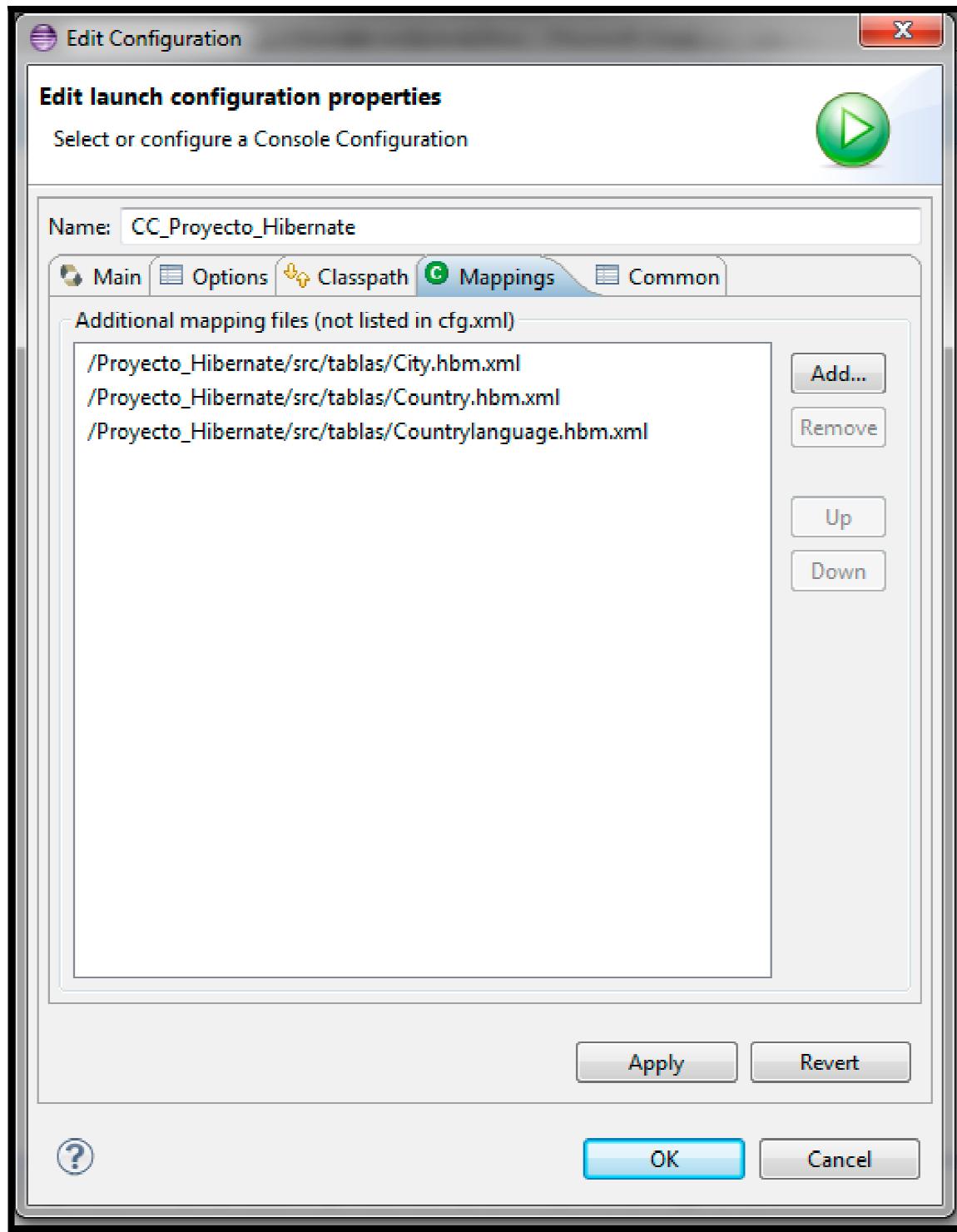
Para esto, mostraremos la perspectiva hibernate en eclipse y pulsaremos con el botón derecho del ratón sobre nuestra consola para acceder a "Edit Configuration".



Nos aparecerá la ventana de la consola que configuramos y seleccionamos la pestaña mappings:



Una vez accedemos a "Mappings" añadiremos los archivos con extension hbm.xml generados en nuestro package "tablas" para relacionar las tablas de la base de datos con nuestras clases:

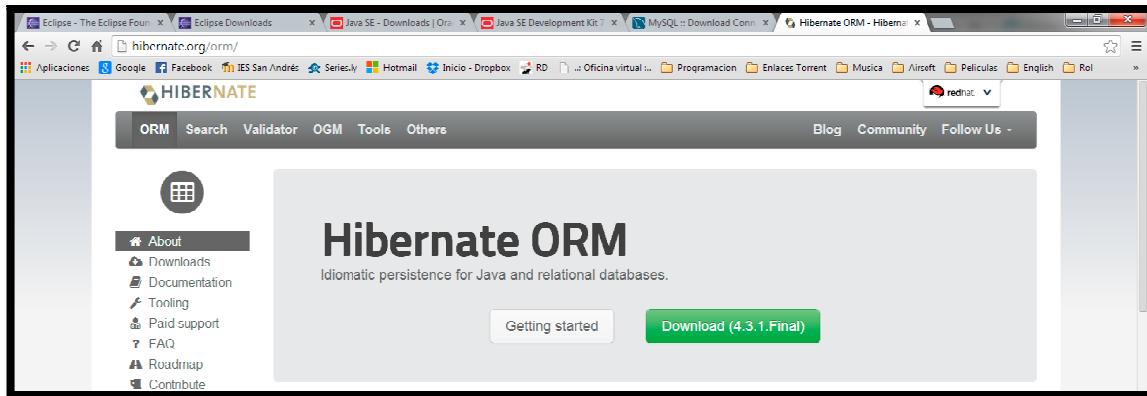


Finalizaremos el proceso pulsando "Apply" y "Ok" en ese orden, así tendremos configurada la consola al 100% con conexión directa a nuestras tablas de la base de datos.

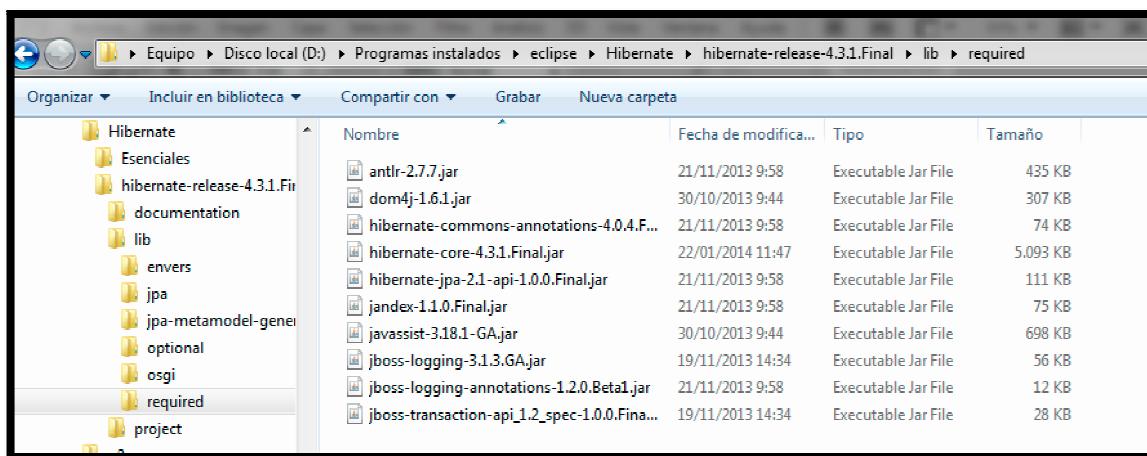
## 4.5 Instalación de la librería Hibernate (archivos .jar)

Para instalar las ultimas librerías necesarias para ejecutar nuestras aplicaciones Hibernate accedemos a la dirección:

URL: [Enlace para acceder a las librerías de hibernate](http://hibernate.org/orm/)

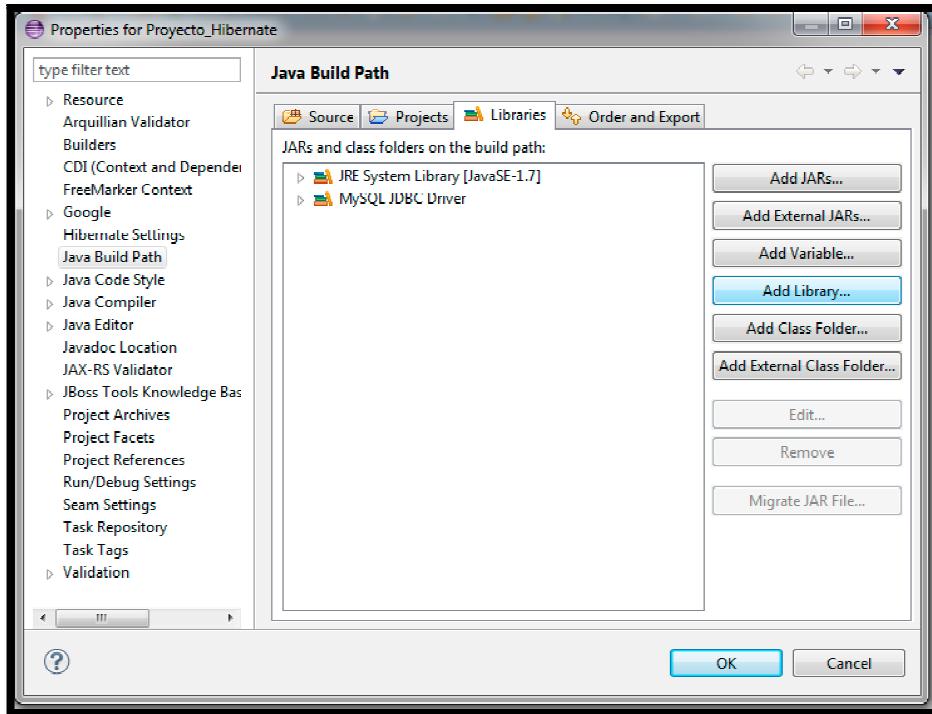


Deberemos crear una carpeta hibernate en el directorio raíz de eclipse y descomprimir el archivo en ella:

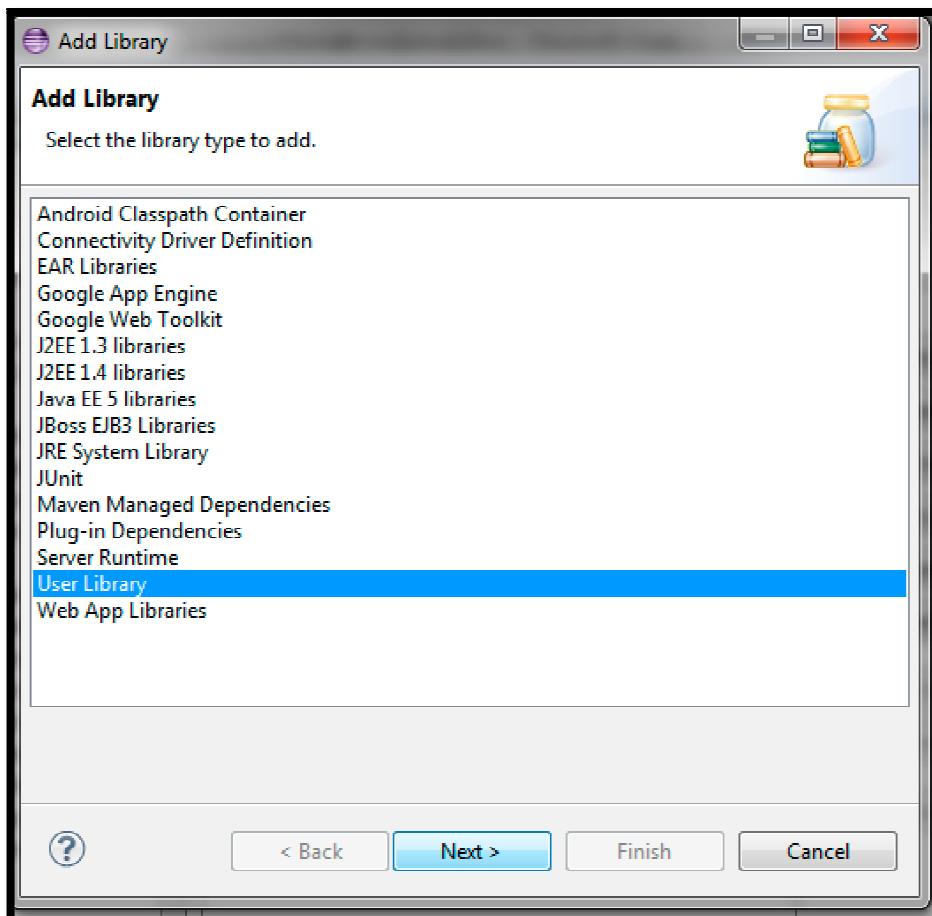


Una vez tengamos la estructura de archivos deseada, procederemos a instalar las librerías de la ruta lib/required de la imagen.

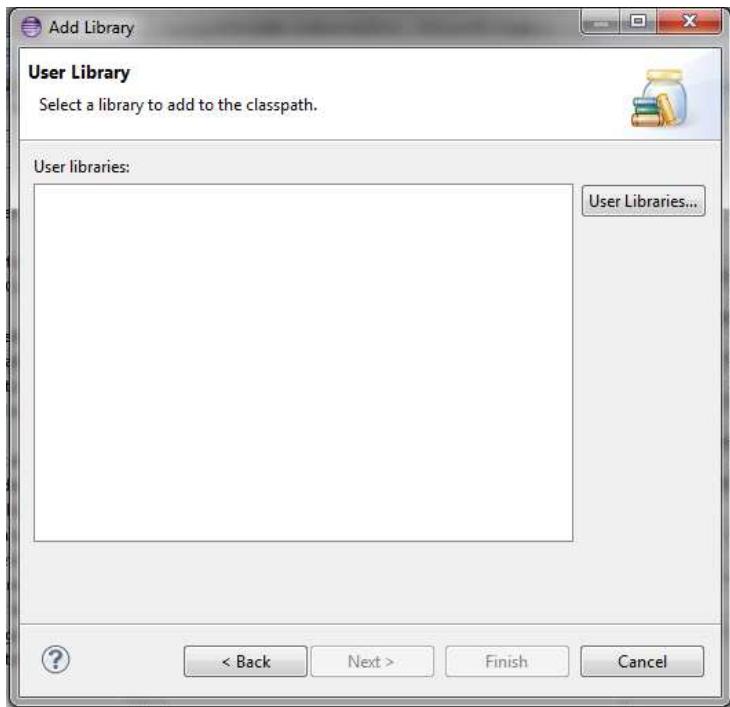
Pulsamos sobre Add Library:



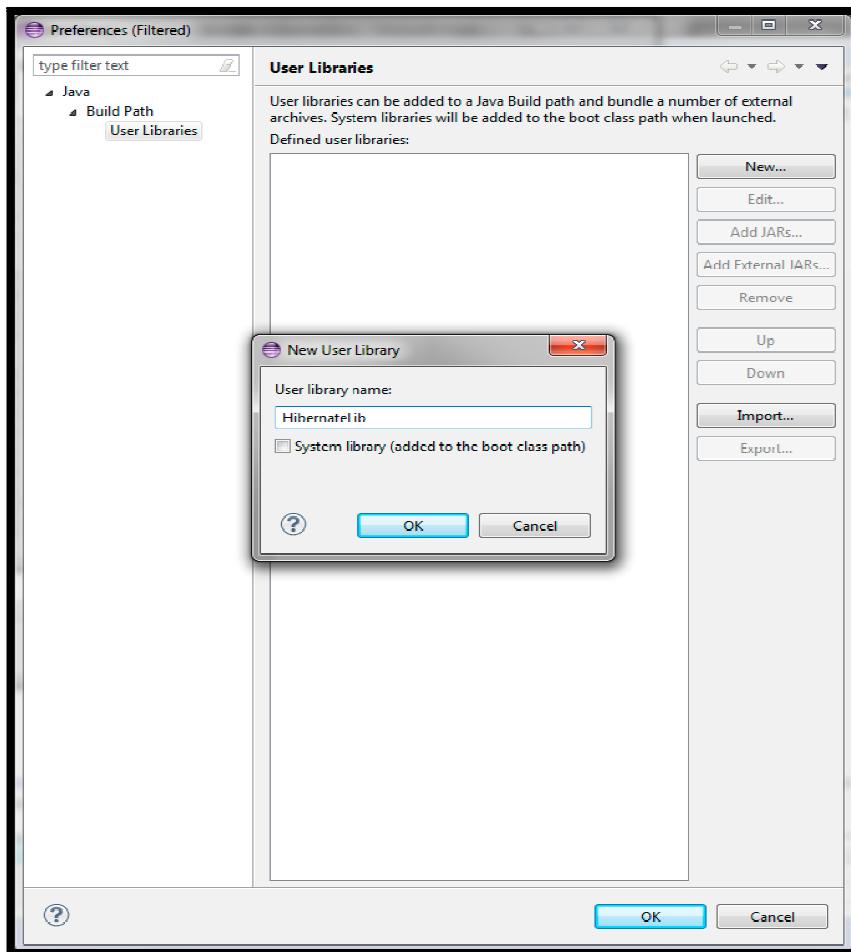
Y seleccionamos



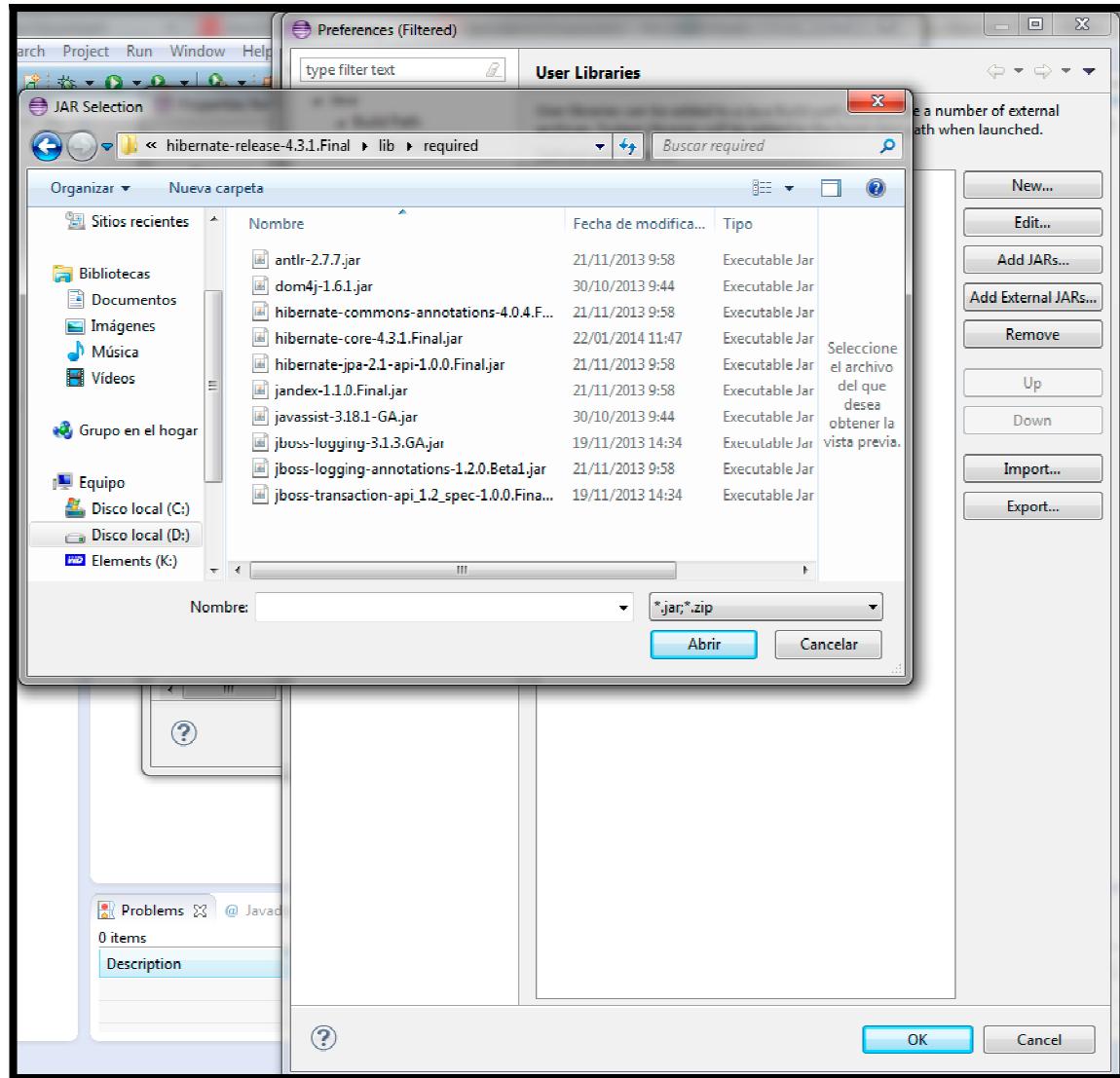
Pulsamos sobre User Libraries...:



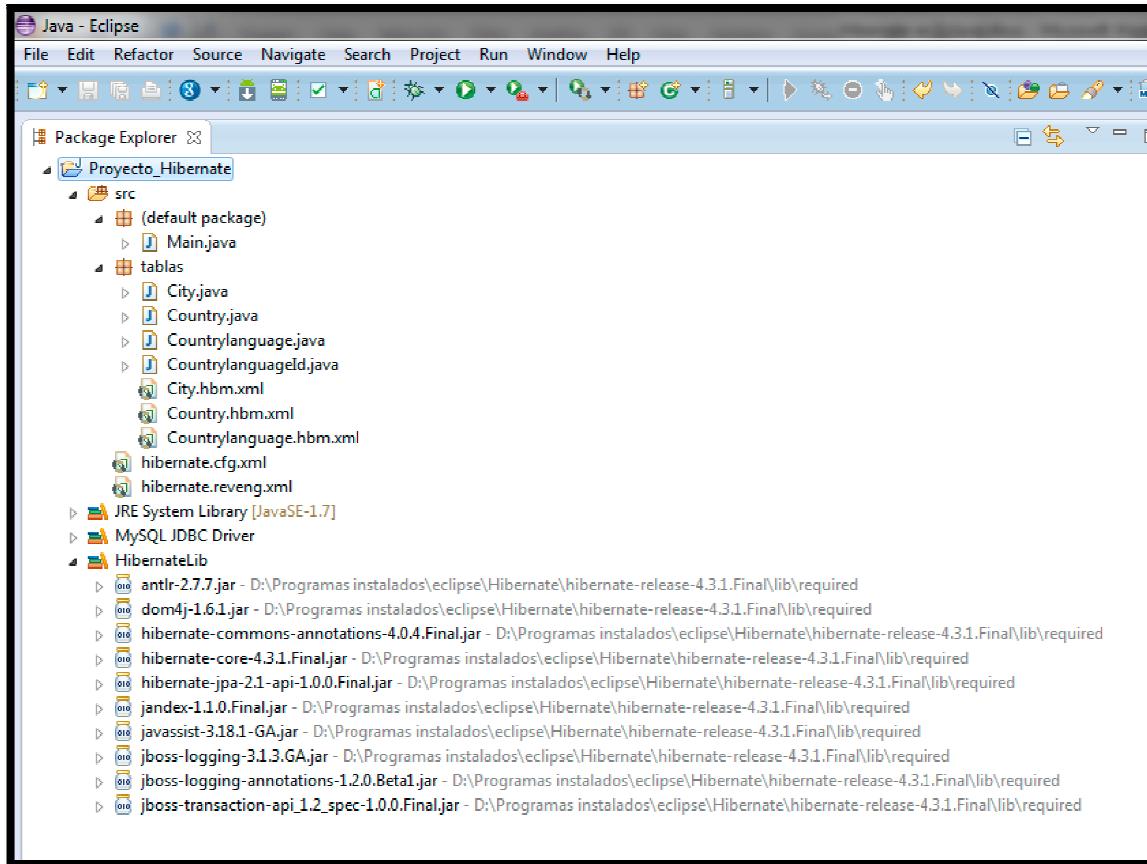
New y escribimos el nombre de nuestra librería para finalizar en Ok:



Finalizamos añadiendo los archivos jar que descomprimimos en los pasos anteriores a esta librería:



Finalizado este paso el proyecto mostrará el siguiente aspecto:



Con la librería Hibernate completa para añadir a cualquier proyecto futuro de nuestro entorno eclipse.

#### 4.6 Últimos pasos de configuración

Para utilizar el método **singleton** debemos eliminar del archivo "hibernate.cfg.xml" la etiqueta:

```
<session-factory="sfnConexion">
```

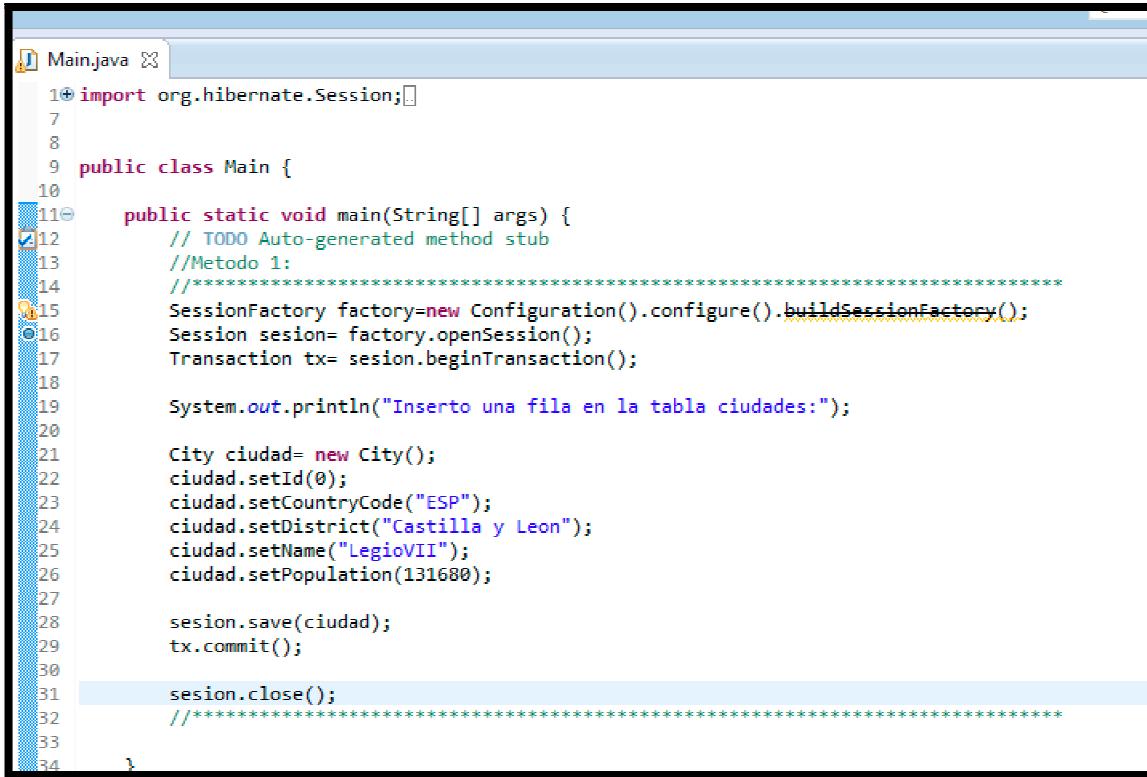
Por

```
<session-factory>
```

Una vez modificada hay que pulsar con el botón derecho en el nombre del proyecto y luego en la opción **Refresh**.

## 5. Código Hibernate

En esta parte vamos a ver un ejemplo muy sencillo del manejo de los objetos hibernate en nuestra aplicación.



```
1+ import org.hibernate.Session;[]
2
3
4 public class Main {
5
6     public static void main(String[] args) {
7         // TODO Auto-generated method stub
8         //Metodo 1:
9         //*****
10        SessionFactory factory=new Configuration().configure().buildSessionFactory();
11        Session sesion= factory.openSession();
12        Transaction tx= sesion.beginTransaction();
13
14        System.out.println("Inserto una fila en la tabla ciudades:");
15
16        City ciudad= new City();
17        ciudad.setId(0);
18        ciudad.setCountryCode("ESP");
19        ciudad.setDistrict("Castilla y Leon");
20        ciudad.setName("LegioVII");
21        ciudad.setPopulation(131680);
22
23        sesion.save(ciudad);
24        tx.commit();
25
26        sesion.close();
27        //*****
28    }
29
30 }
```

## 6. Bibliografía

1. Acceso a Datos (Editorial Garceta)
2. Web: <http://hibernate.org/>
3. Manual Hibernate (adjunto en la documentación)  
autor: Ismael Núñez