



HIBERNATE

Persistencia de datos con Hibernate utilizando una
base de datos SQL Server

Software Factory
2016-1

Roy Alejo Taza Rojas
U201213734@upc.edu.pe

Contenido

INTRODUCCIÓN	2
REQUISITOS	3
PREPARACIÓN	4
Base de datos	4
JDBC Driver para SQL Server	4
IMPLEMENTACIÓN	5
Creación del Proyecto	5
Dependencias	5
Configuración de Hibernate	6
Creando el archivo HibernateUtil.....	11
Creando Hibernate Revenge Engineering Wizard.....	12
Creando los Archivos de configuración de Hibernate y POJO.....	14
Utilizando sentencias HQL Query.....	20
Helper de Productos.....	21
PROBANDO LA FUNCIONALIDAD	23
Listar	24
Obtener	24
Insertar	24
Actualizar.....	24
Eliminar	24

INTRODUCCIÓN

En este tutorial, se creará aplicación que muestra los datos de una base de dato. La aplicación utilizará el framework de Hibernate para la persistencia de datos, recuperar y almacenar objetos (Plain Old Java POJO) a una base de datos relacional SQL Server.

Hibernate proporciona herramientas para mapeo objeto-relacional (ORM). El tutorial muestra cómo agregar soporte para el marco de hibernación al IDE de Netbeans y crear los archivos necesarios de Hibernate.

REQUISITOS

Para la presente guía se utilizó:

- Netbeans IDE 8.0.2
- JDK 8
- SQL Server 2008 R2

PREPARACIÓN

Base de datos

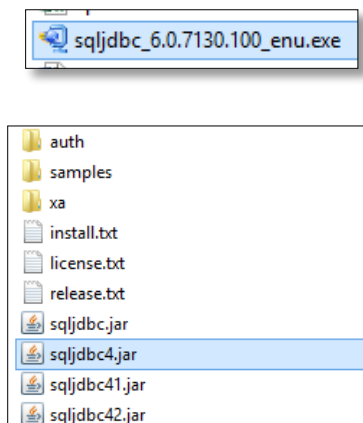
Para esta guía utilizaremos la base de datos Northwind, la cual podemos descargar y restauraremos en nuestra instancia de base de datos:

<https://northwinddatabase.codeplex.com/>

The screenshot shows the Northwind database website. The main content area features a 'Project Description' section stating that the database can be downloaded as an executable SQL script (285 Kb) or as a backup that can be restored with sample data. It also mentions compatibility with SQL Server 2005 and 2008. Below this is a 'The database diagram for Northwind' section with a link to download an A4 size printout. The diagram itself is a complex ER model showing tables like Products, Order Details, EmployeeTerritories, Orders, Customers, Suppliers, Categories, Employees, Territories, Region, Shippers, CustomerCustomerDemo, and CustomerDemographics. To the right of the diagram is a 'download' button and a table with metadata: CURRENT (northwind database backup-Sql Server 2005 and 2008), DATE (Thu Aug 11, 2011 at 2:00 AM), STATUS (Alpha), DOWNLOADS (348.866), and RATING (4 stars, 8 ratings). There is also a 'MOST HELPFUL REVIEWS' section with two reviews and a 'View all reviews' link. At the bottom right is an 'ACTIVITY' section.

JDBC Driver para SQL Server

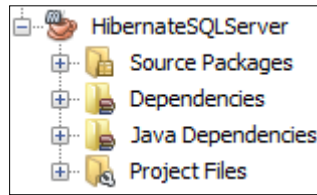
Para conectarnos con SQL Server necesitamos tener el driver JDBC correspondiente. Descargamos el driver de la siguiente página, teniendo en cuenta el JDK que tengamos: <https://www.microsoft.com/en-us/download/details.aspx?displaylang=en&id=11774>. En esta guía estamos utilizando JDK 8, por ello descargaremos sqljdbc_6.0.7130.100_enu:



IMPLEMENTACIÓN

Creación del Proyecto

Creamos un proyecto Maven -> Java Application y pondremos de nombre **HibernateSQLServer**.

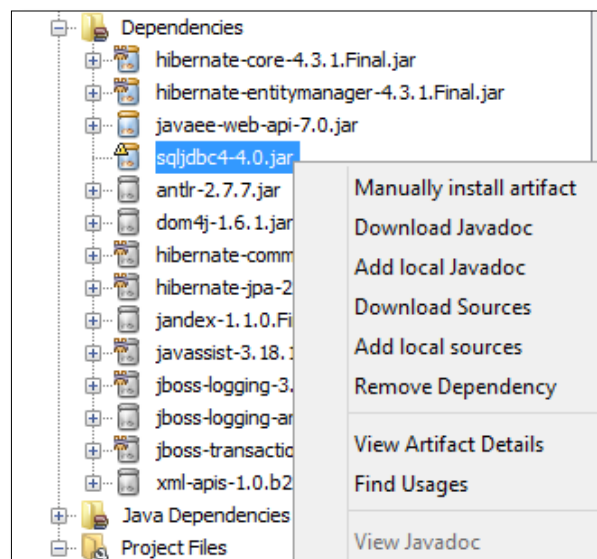


Dependencias

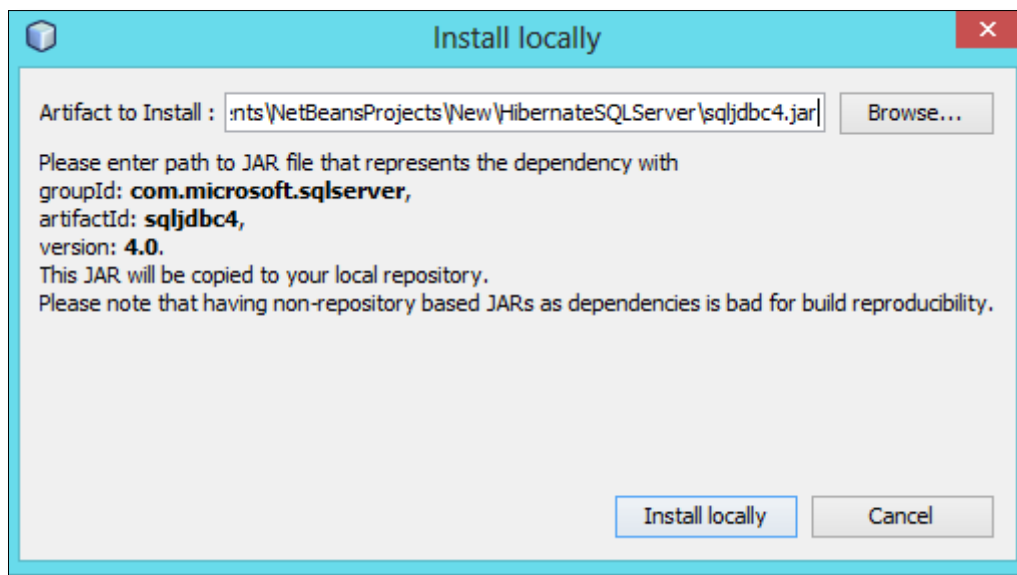
Añadimos la siguiente dependencia a nuestro proyecto editando el archivo pom.xml

```
<dependency>
  <groupId>org.hibernate</groupId>
  <artifactId>hibernate-core</artifactId>
  <version>4.3.1.Final</version>
</dependency>
<dependency>
  <groupId>org.hibernate</groupId>
  <artifactId>hibernate-entitymanager</artifactId>
  <version>4.3.1.Final</version>
</dependency>
<dependency>
  <groupId>com.microsoft.sqlserver</groupId>
  <artifactId>sqljdbc4</artifactId>
  <version>4.0</version>
</dependency>
```

Expandiremos la carpeta dependencies de nuestro proyecto y veremos que **sqljdbc4** no fue descargado, esto es porque Microsoft no da permiso a Maven que publique su JDBC Driver, por ello lo haremos manualmente. Haremos clic derecho a la dependencia y seleccionaremos la opción **Manually install artifact**

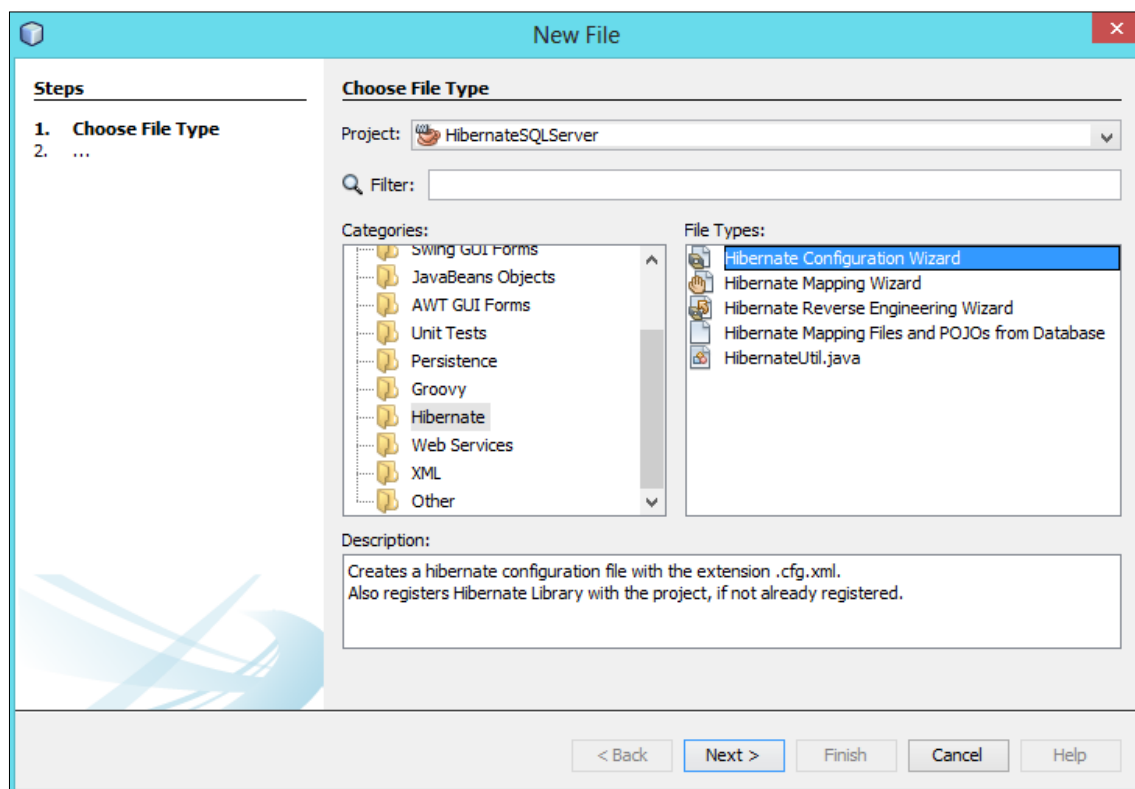


Seleccionamos el jar del driver JDBC SQL Server correspondiente y lo instalamos:

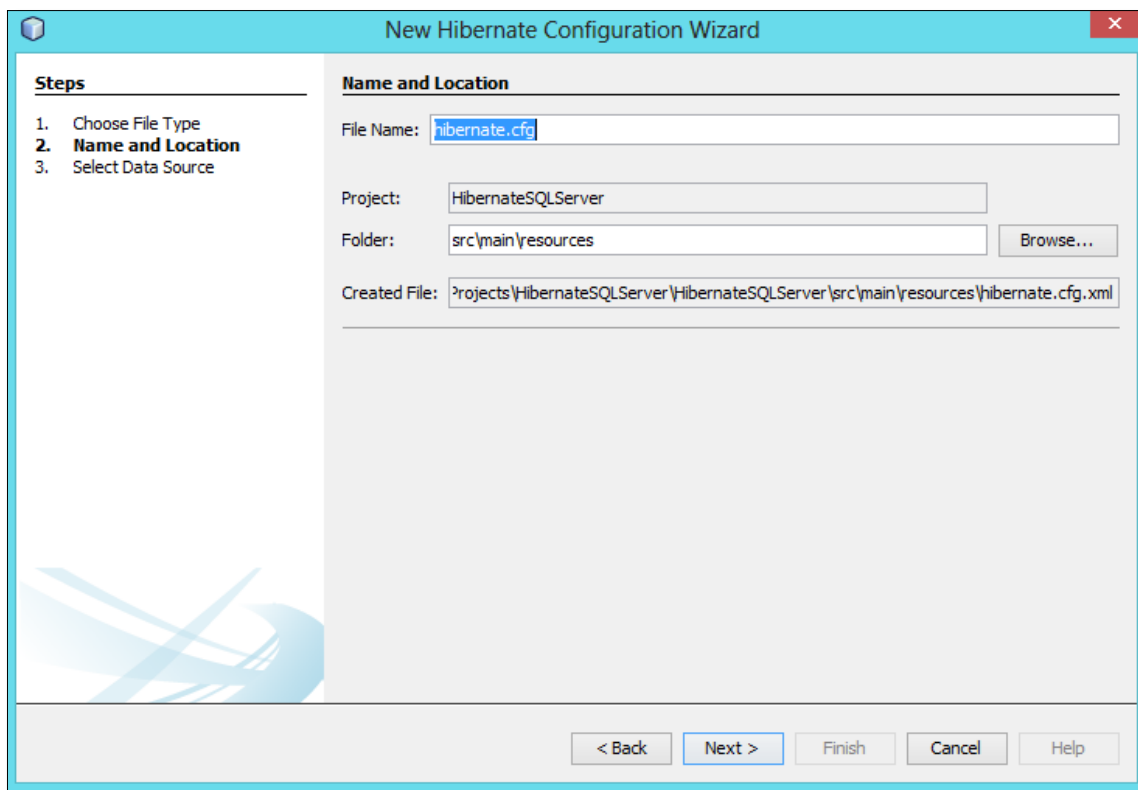


Configuración de Hibernate

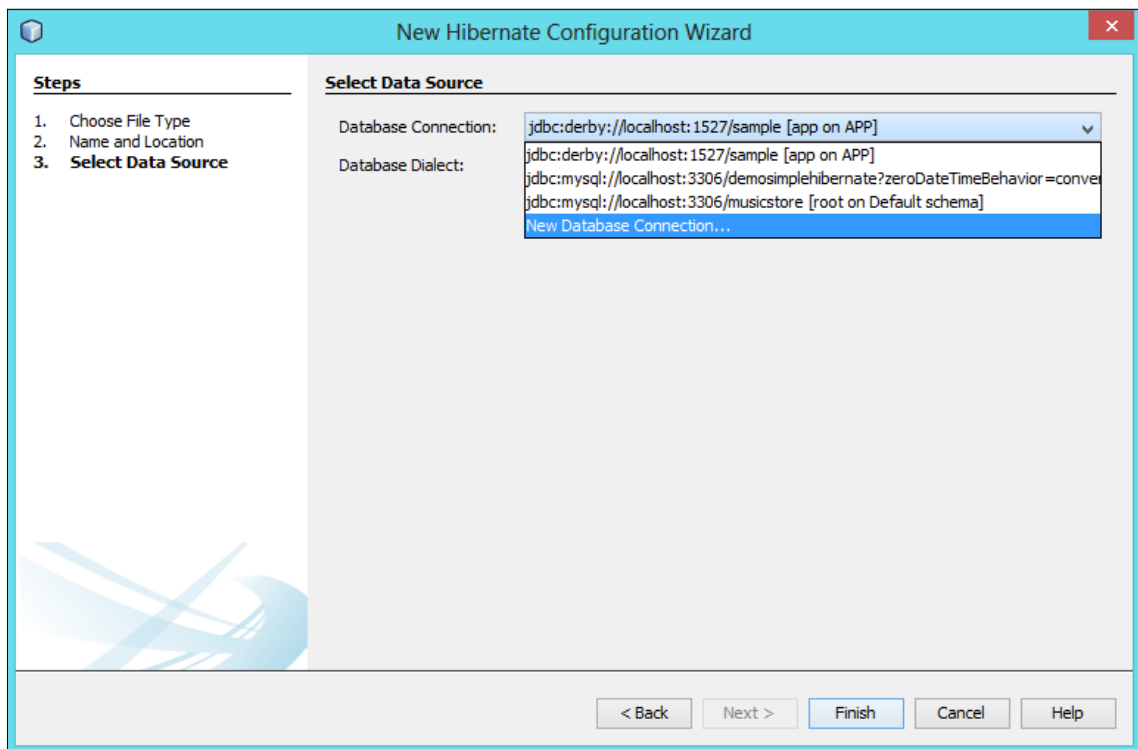
Clic derecho en nuestro proyecto y añadimos un archivo Hibernate Configuration Wizard:



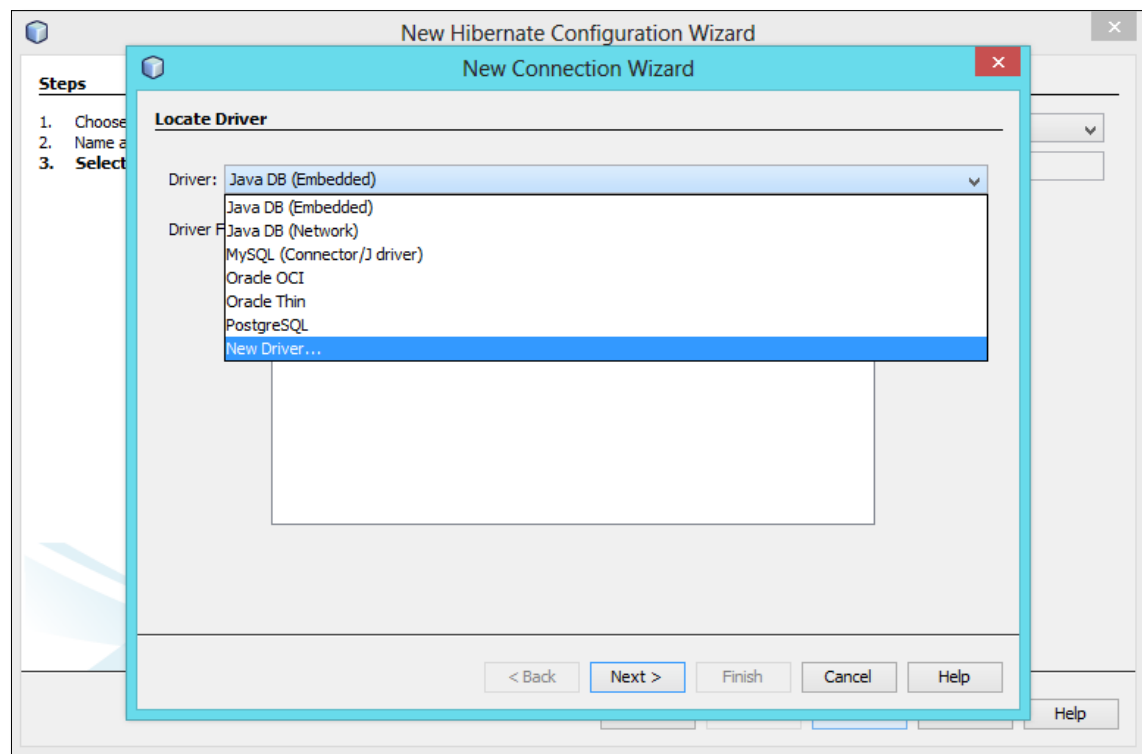
Lo creamos en la raíz del proyecto:



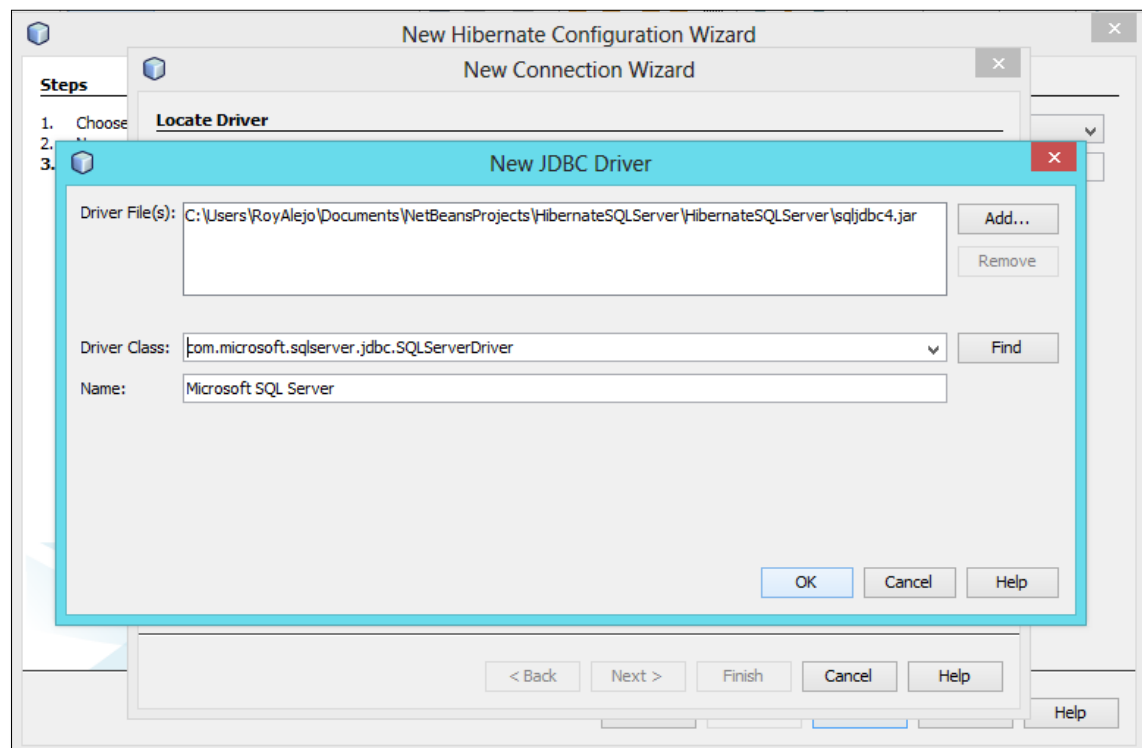
Seleccionamos una nueva conexión de base de datos:



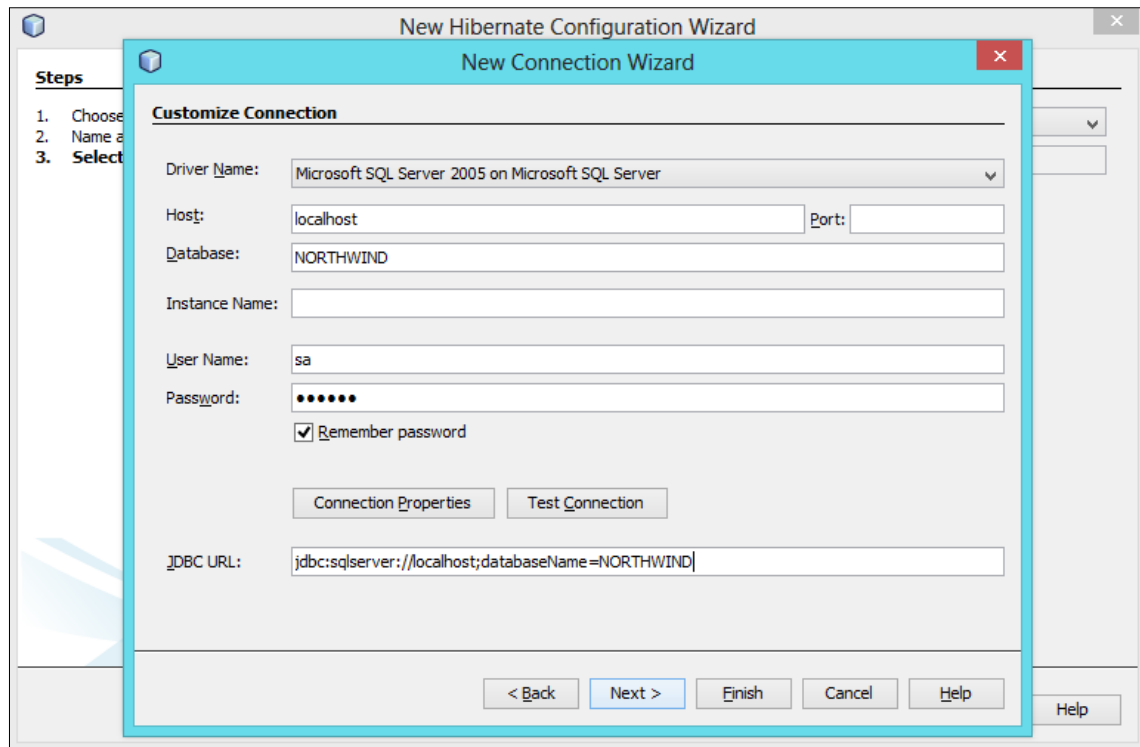
Seleccionamos nuevo driver:



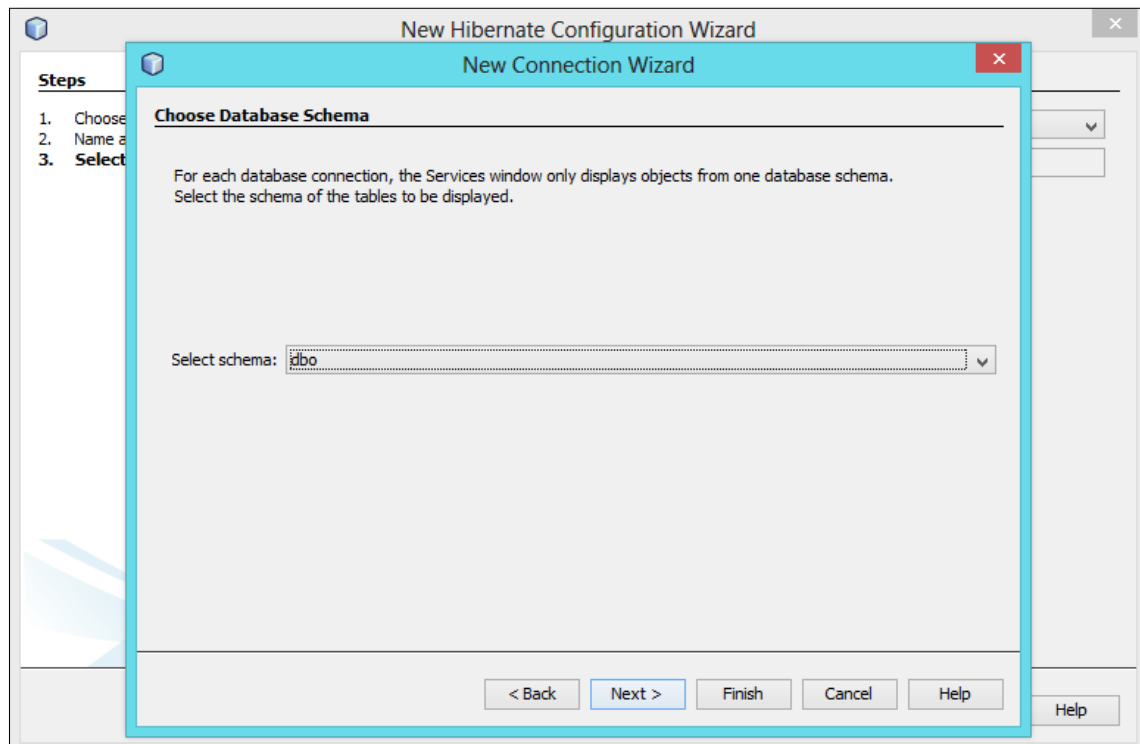
Seleccionamos el JDBC driver para SQL Server:



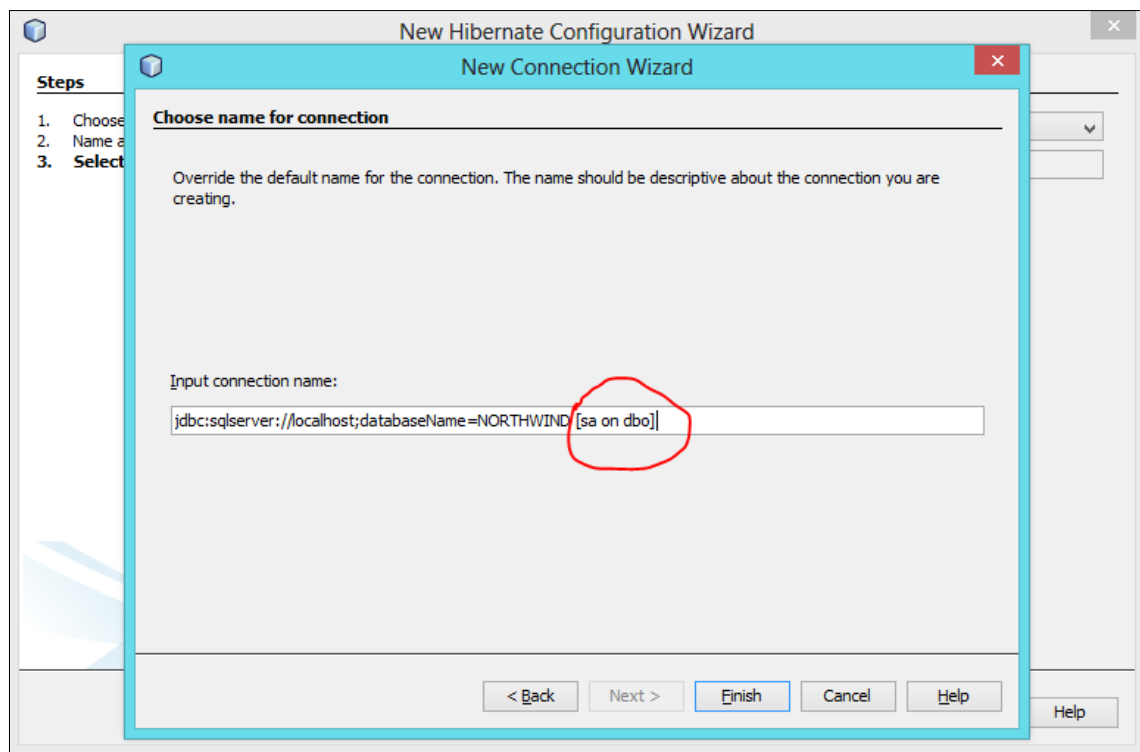
Configuramos la conexión a la base de datos Northwind



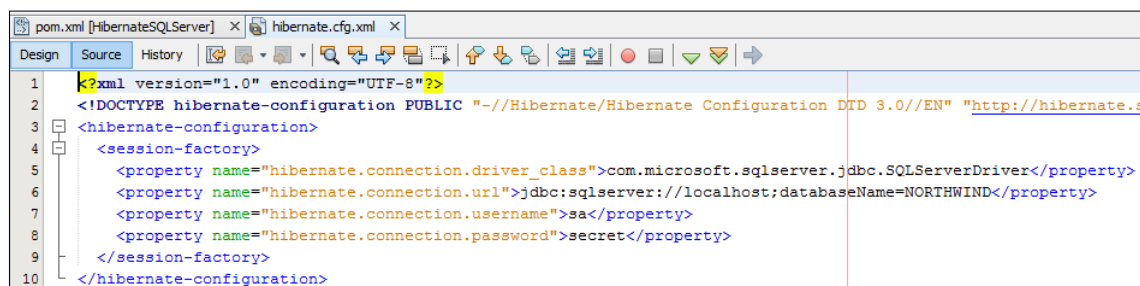
Escogemos el esquema dbo:



Nos aseguramos que diga [sa on dbo]



Finish -> Finish. El archivo de configuración de Hibernate se habrá creado:



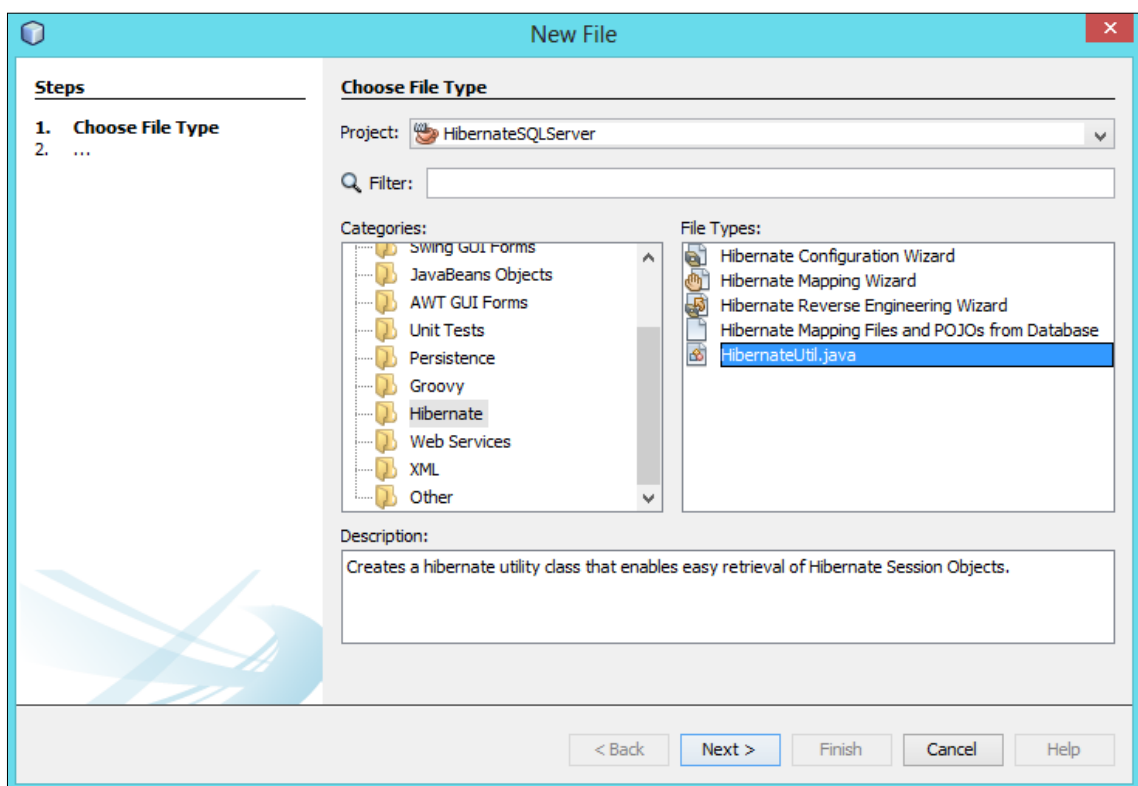
Agregamos el dialecto SQL Server 2008 para nuestro caso, y algunas propiedades más. El archivo de configuración quedará así:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD
3.0//EN" "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
  <session-factory>
    <property
name="hibernate.connection.driver_class">com.microsoft.sqlserver.jdbc.SQLServerDriver</
property>
    <property
name="hibernate.connection.url">jdbc:sqlserver://localhost;databaseName=NORTHWIND<
/property>
    <property name="hibernate.connection.username">sa</property>
```

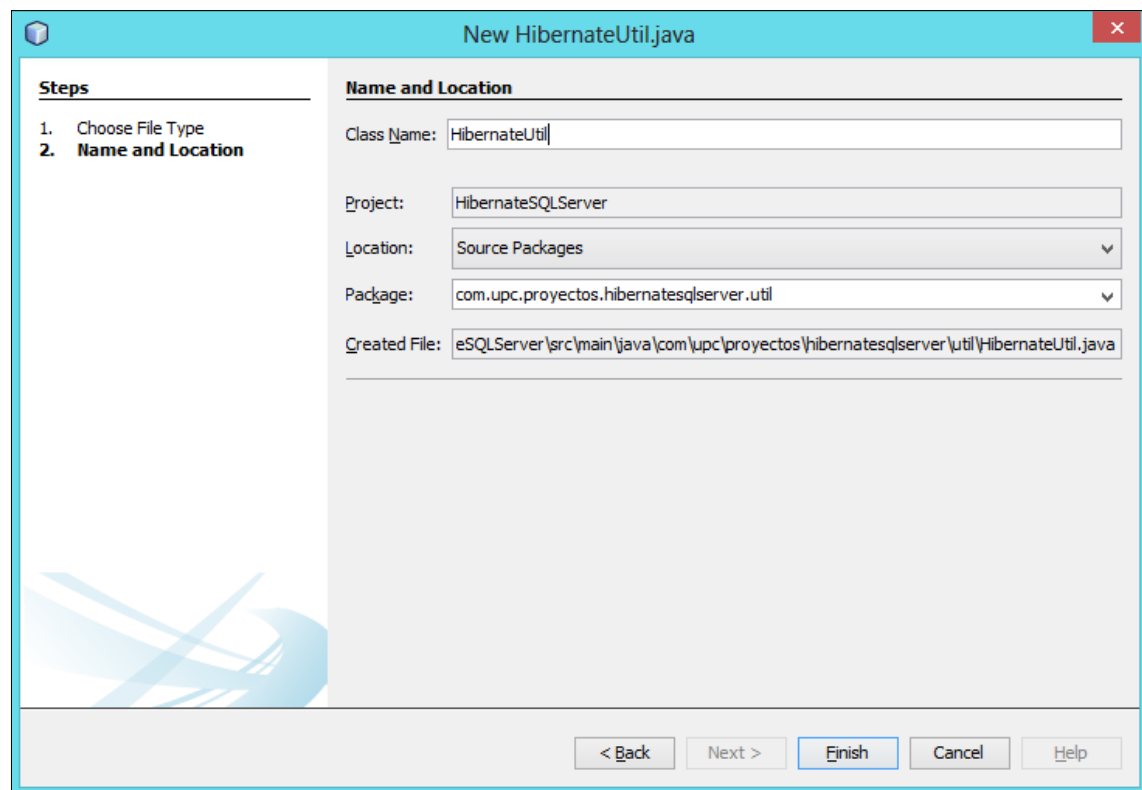
```
<property name="hibernate.connection.password">secret</property>
<!-- JDBC connection pool (use the built-in) -->
<property name="connection.pool_size">1</property>
<!-- SQL dialect -->
<property name="dialect">org.hibernate.dialect.SQLServer2008Dialect</property>
<!-- Echo all executed SQL to stdout -->
<property name="show_sql">true</property>
</session-factory>
</hibernate-configuration>
```

Creando el archivo HibernateUtil

Este archivo inicializa la sesión a la base de datos. Clic derecho en nuestro proyecto y creamos un archivo **HibernateUtil.java**

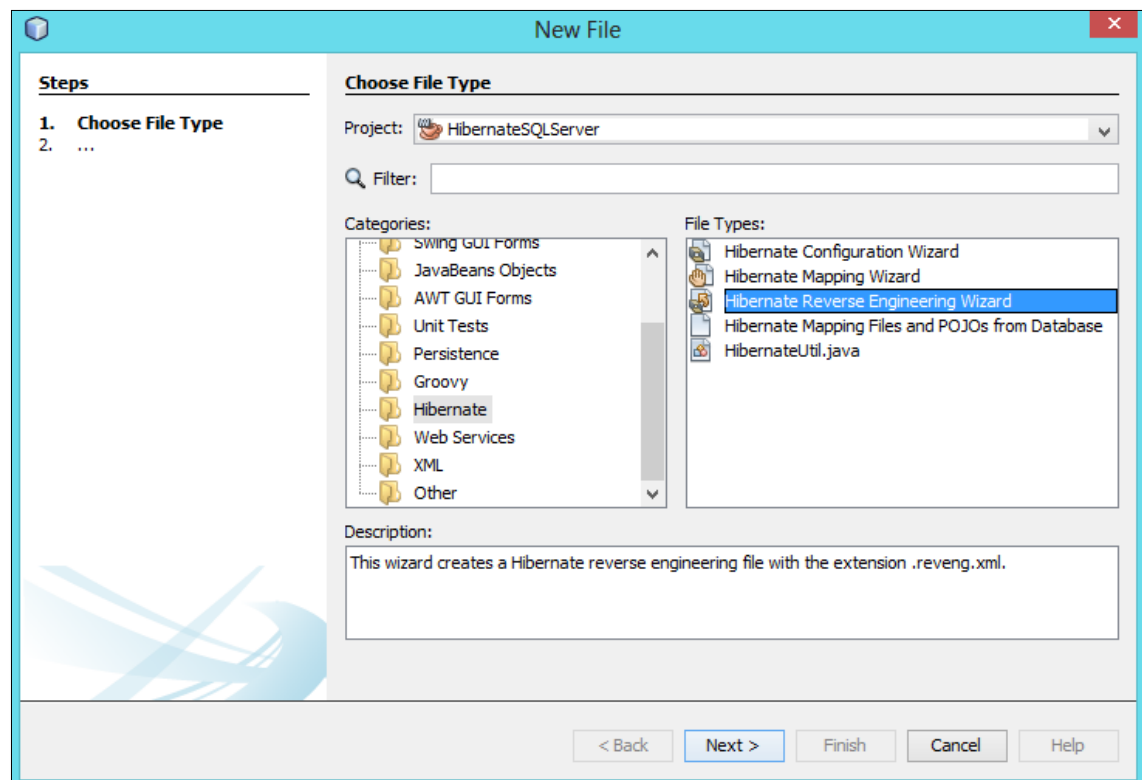


Lo pondremos en un paquete **útil** de nuestro proyecto:

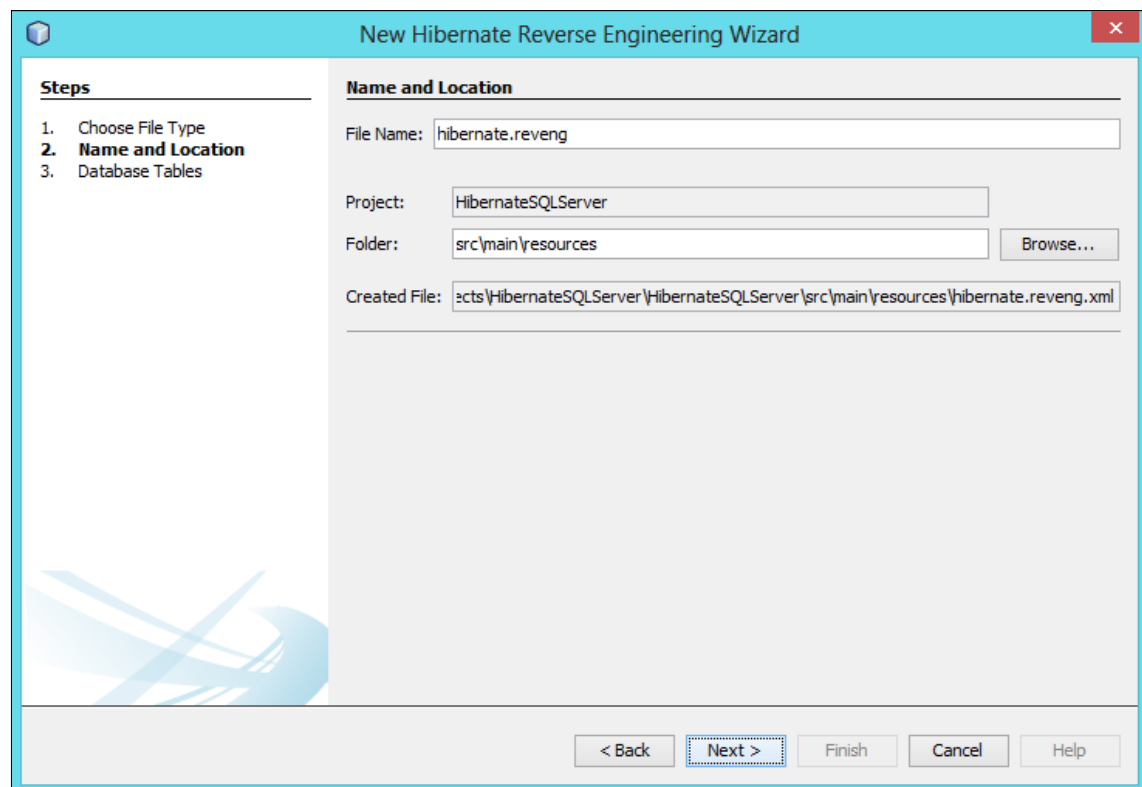


Creando Hibernate Revenge Engineering Wizard

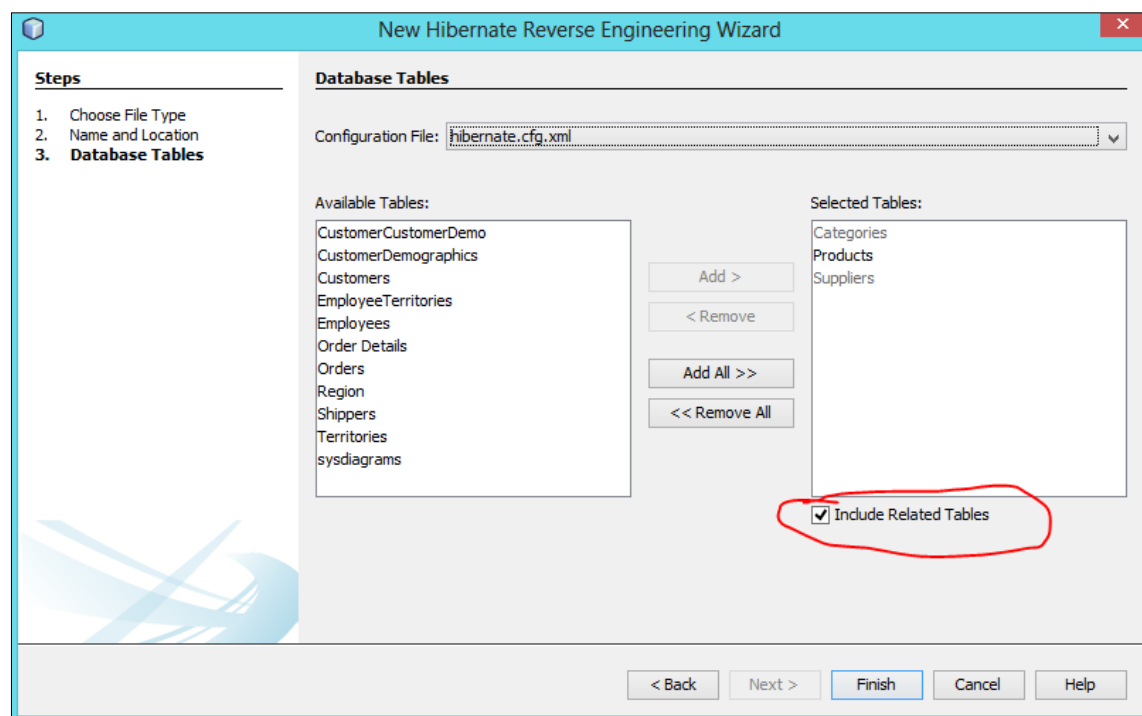
Este archivo de ingeniería inversa permite tener un mayor control sobre la estrategia de mapeo de base de datos. Clic derecho en nuestro proyecto y seleccionamos lo siguiente:



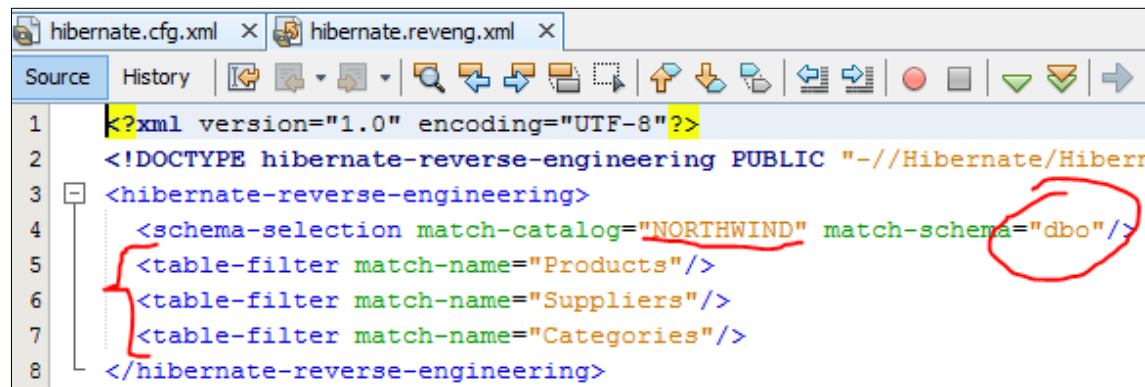
Lo pondremos en la raíz del proyecto:



Tomará el archivo de configuración de hibernate y nos mostrará las tablas. Seleccionaremos para esta guía la tabla Products junto con sus dependencias.



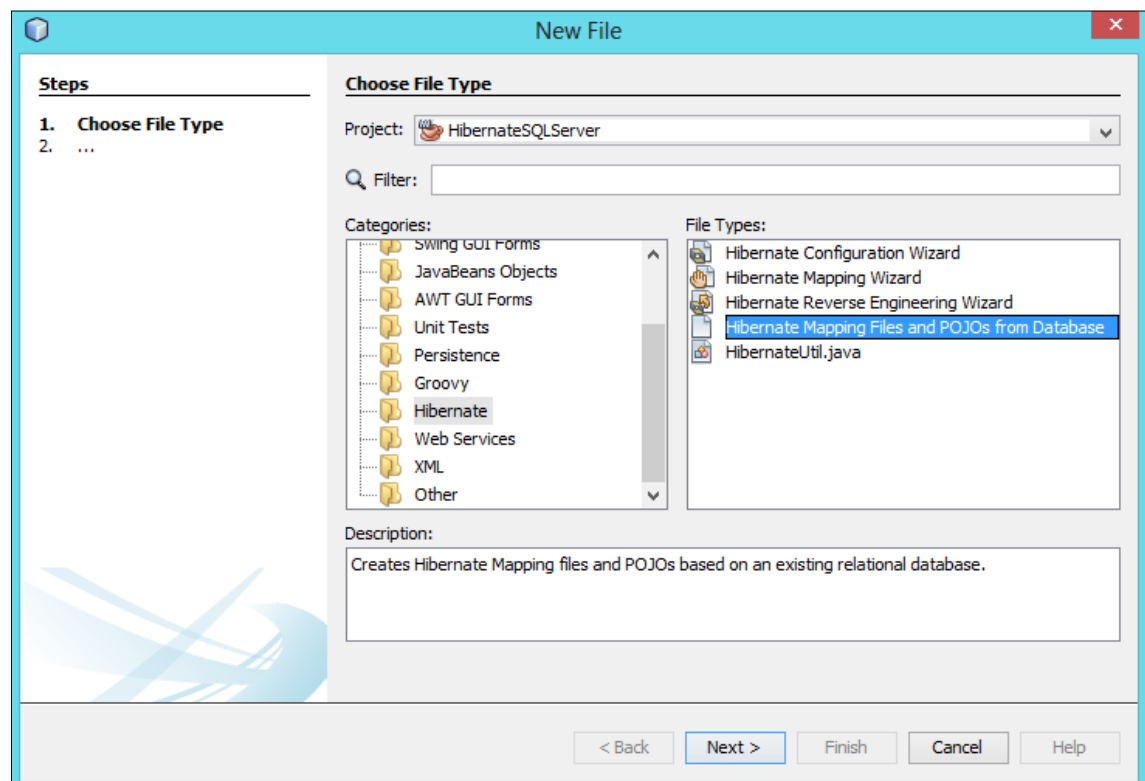
Se habrá creado el archivo de ingeniería inversa. Cerciorémonos que el esquema sea **dbo**, la base **Northwind** y las tablas que seleccionamos:



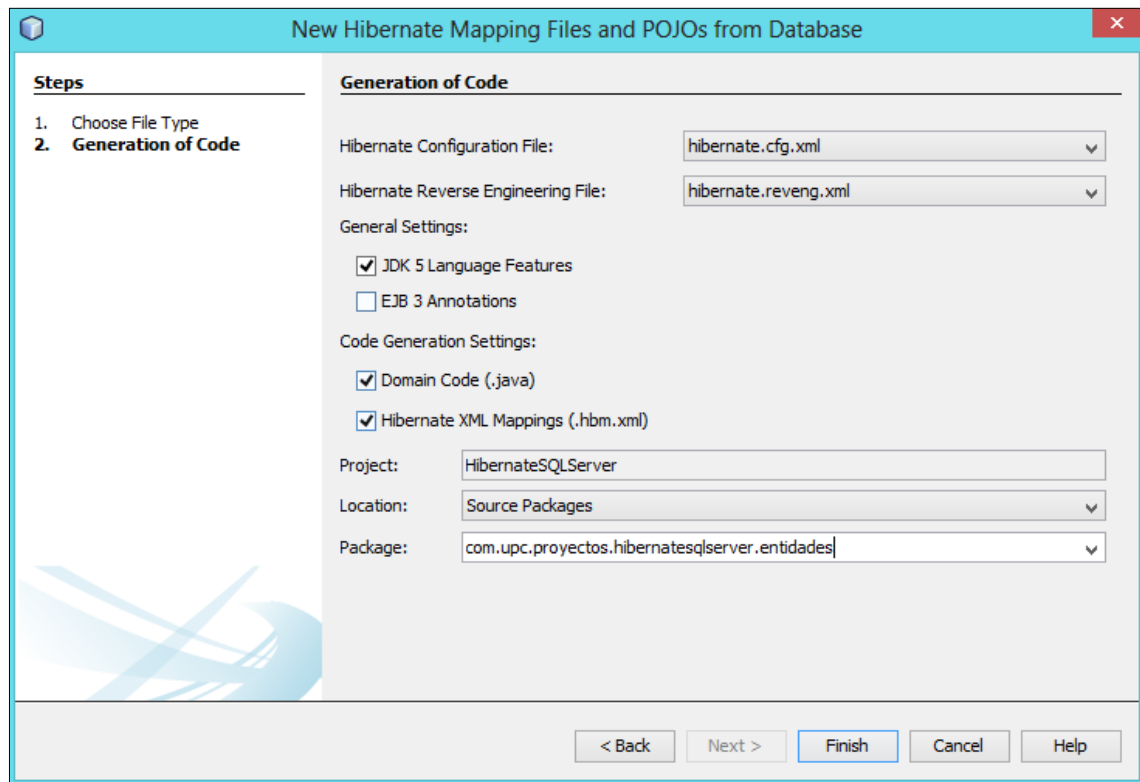
```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE hibernate-reverse-engineering PUBLIC "-//Hibernate/Hibernate
3 <hibernate-reverse-engineering>
4   <schema-selection match-catalog="NORTHWIND" match-schema="dbo"/>
5   <table-filter match-name="Products"/>
6   <table-filter match-name="Suppliers"/>
7   <table-filter match-name="Categories"/>
8 </hibernate-reverse-engineering>
```

Creando los Archivos de configuración de Hibernate y POJO

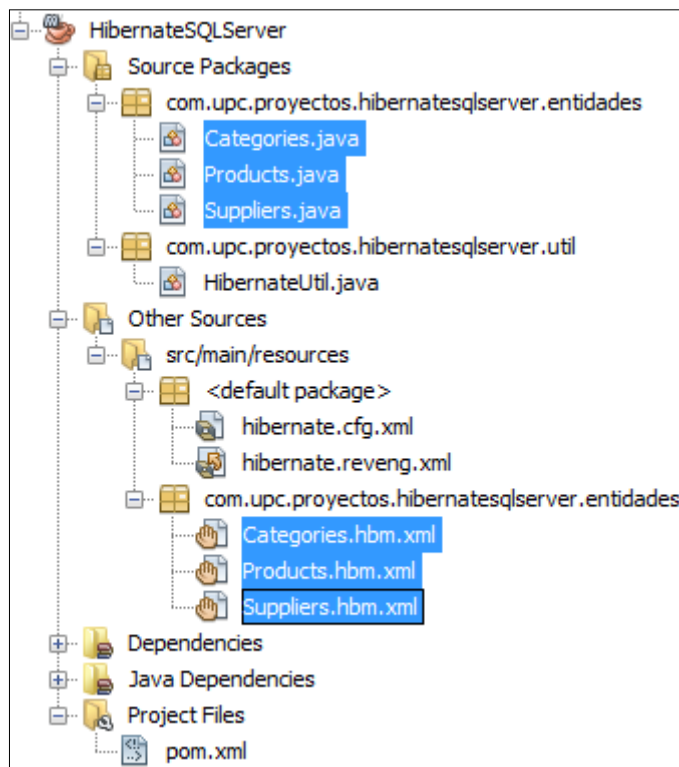
Crearemos el mapeo de las tablas de nuestra base de datos. Clic derecho en nuestro proyecto y seleccionamos lo siguiente:



Veremos que utiliza el archivo de configuración e ingeniería inversa de Hibernate. Elegimos las siguientes opciones:



Se agregará lo siguiente al proyecto:



Se habrá mapeado las entidades junto a sus relaciones, reduciendo este engorroso proceso.

Ya que estamos trabajando con SQL Server, tenemos que verificar los tipos de datos de cada propiedad de las clases mapeadas. Por ejemplo, en la clase Suppliers se habrá mapeado campos de tipo Serializable cuando en realidad son String. Al hacer el reemplazo quedará así:

ANTES	DESPUES
<pre> public class Suppliers implements java.io.Serializable { private int supplierId; private Serializable companyName; private Serializable contactName; private Serializable contactTitle; private Serializable address; private Serializable city; private Serializable region; private Serializable postalCode; private Serializable country; private Serializable phone; private Serializable fax; private Serializable homePage; private Set<Products> productses = new HashSet<Products>(0); public Suppliers() { } public Suppliers(int supplierId, Serializable companyName) { this.supplierId = supplierId; this.companyName = companyName; } public Suppliers(int supplierId, Serializable companyName, Serializable contactName, Serializable contactTitle, Serializable address, Serializable city, Serializable region, Serializable postalCode, Serializable country, Serializable phone, Serializable fax, Serializable homePage, Set<Products> productses) { this.supplierId = supplierId; this.companyName = companyName; this.contactName = contactName; this.contactTitle = contactTitle; this.address = address; this.city = city; this.region = region; this.postalCode = postalCode; this.country = country; this.phone = phone; this.fax = fax; this.homePage = homePage; this.productses = productses; } public int getSupplierId() { return this.supplierId; } </pre>	<pre> public class Suppliers implements java.io.Serializable { private int supplierId; private String companyName; private String contactName; private String contactTitle; private String address; private String city; private String region; private String postalCode; private String country; private String phone; private String fax; private String homePage; private Set<Products> productses = new HashSet<Products>(0); public Suppliers() { } public Suppliers(int supplierId, String companyName) { this.supplierId = supplierId; this.companyName = companyName; } public Suppliers(int supplierId, String companyName, String contactName, String contactTitle, String address, String city, String region, String postalCode, String country, String phone, String fax, String homePage, Set<Products> productses) { this.supplierId = supplierId; this.companyName = companyName; this.contactName = contactName; this.contactTitle = contactTitle; this.address = address; this.city = city; this.region = region; this.postalCode = postalCode; this.country = country; this.phone = phone; this.fax = fax; this.homePage = homePage; this.productses = productses; } public int getSupplierId() { return this.supplierId; } </pre>

<pre> } public void setSupplierId(int supplierId) { this.supplierId = supplierId; } public Serializable getCompanyName() { return this.companyName; } public void setCompanyName(Serializable companyName) { this.companyName = companyName; } public Serializable getContactName() { return this.contactName; } public void setContactName(Serializable contactName) { this.contactName = contactName; } public Serializable getContactTitle() { return this.contactTitle; } public void setContactTitle(Serializable contactTitle) { this.contactTitle = contactTitle; } public Serializable getAddress() { return this.address; } public void setAddress(Serializable address) { this.address = address; } public Serializable getCity() { return this.city; } public void setCity(Serializable city) { this.city = city; } public Serializable getRegion() { return this.region; } public void setRegion(Serializable region) { this.region = region; } </pre>	<pre> public void setSupplierId(int supplierId) { this.supplierId = supplierId; } public String getCompanyName() { return this.companyName; } public void setCompanyName(String companyName) { this.companyName = companyName; } public String getContactName() { return this.contactName; } public void setContactName(String contactName) { this.contactName = contactName; } public String getContactTitle() { return this.contactTitle; } public void setContactTitle(String contactTitle) { this.contactTitle = contactTitle; } public String getAddress() { return this.address; } public void setAddress(String address) { this.address = address; } public String getCity() { return this.city; } public void setCity(String city) { this.city = city; } public String getRegion() { return this.region; } public void setRegion(String region) { this.region = region; } public String getPostalCode() { return this.postalCode; } </pre>
---	---

<pre> public Serializable getPostalCode() { return this.postalCode; } public void setPostalCode(Serializable postalCode) { this.postalCode = postalCode; } public Serializable getCountry() { return this.country; } public void setCountry(Serializable country) { this.country = country; } public Serializable getPhone() { return this.phone; } public void setPhone(Serializable phone) { this.phone = phone; } public Serializable getFax() { return this.fax; } public void setFax(Serializable fax) { this.fax = fax; } public Serializable getHomePage() { return this.homePage; } public void setHomePage(Serializable homePage) { this.homePage = homePage; } public Set<Products> getProductses() { return this.productses; } public void setProductses(Set<Products> productses) { this.productses = productses; } </pre>	<pre> } public void setPostalCode(String postalCode) { this.postalCode = postalCode; } public String getCountry() { return this.country; } public void setCountry(String country) { this.country = country; } public String getPhone() { return this.phone; } public void setPhone(String phone) { this.phone = phone; } public String getFax() { return this.fax; } public void setFax(String fax) { this.fax = fax; } public String getHomePage() { return this.homePage; } public void setHomePage(String homePage) { this.homePage = homePage; } public Set<Products> getProductses() { return this.productses; } public void setProductses(Set<Products> productses) { this.productses = productses; } </pre>
---	---

Hacemos lo mismo con Suppliers.hbm.xml

ANTES	DESPUES
<pre> <?xml version="1.0"?> <!DOCTYPE hibernate-mapping PUBLIC "-// Hibernate/Hibernate Mapping DTD 3.0//EN" </pre>	<pre> <?xml version="1.0"?> <!DOCTYPE hibernate-mapping PUBLIC "-// Hibernate/Hibernate Mapping DTD 3.0//EN" </pre>

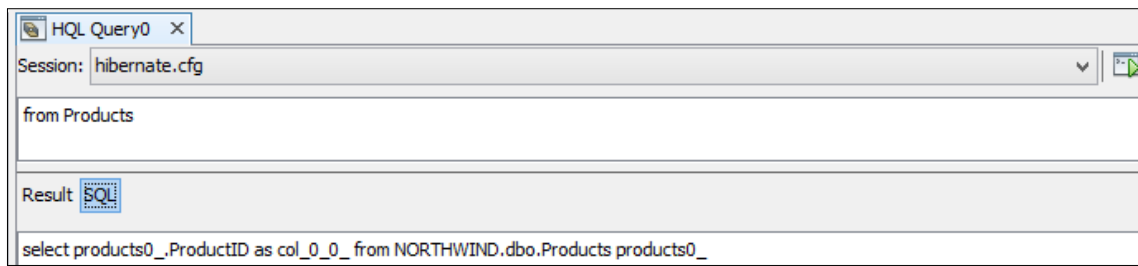
<pre> "http://www.hibernate.org/dtd/hibernate- mapping-3.0.dtd"> <!-- Generated 05/06/2016 06:13:26 PM by Hibernate Tools 4.3.1 --> <hibernate-mapping> <class name="com.upc.proyectos.hibernateqlserver.en tidades.Suppliers" table="Suppliers" schema="dbo" catalog="NORTHWIND" optimistic-lock="version"> <id name="supplierId" type="int"> <column name="SupplierID" /> <generator class="assigned" /> </id> <property name="companyName" type="serializable"> <column name="CompanyName" not- null="true" /> </property> <property name="contactName" type="serializable"> <column name="ContactName" /> </property> <property name="contactTitle" type="serializable"> <column name="ContactTitle" /> </property> <property name="address" type="serializable"> <column name="Address" /> </property> <property name="city" type="serializable"> <column name="City" /> </property> <property name="region" type="serializable"> <column name="Region" /> </property> <property name="postalCode" type="serializable"> <column name="PostalCode" /> </property> <property name="country" type="serializable"> <column name="Country" /> </property> <property name="phone" type="serializable"> <column name="Phone" /> </property> <property name="fax" type="serializable"> <column name="Fax" /> </property> <property name="homePage" type="serializable"> <column name="HomePage" /> </property> </pre>	<pre> "http://www.hibernate.org/dtd/hibernate- mapping-3.0.dtd"> <!-- Generated 05/06/2016 06:13:26 PM by Hibernate Tools 4.3.1 --> <hibernate-mapping> <class name="com.upc.proyectos.hibernateqlserver.en tidades.Suppliers" table="Suppliers" schema="dbo" catalog="NORTHWIND" optimistic-lock="version"> <id name="supplierId" type="int"> <column name="SupplierID" /> <generator class="assigned" /> </id> <property name="companyName" type="string"> <column name="CompanyName" not- null="true" /> </property> <property name="contactName" type="string"> <column name="ContactName" /> </property> <property name="contactTitle" type="string"> <column name="ContactTitle" /> </property> <property name="address" type="string"> <column name="Address" /> </property> <property name="city" type="string"> <column name="City" /> </property> <property name="region" type="string"> <column name="Region" /> </property> <property name="postalCode" type="string"> <column name="PostalCode" /> </property> <property name="country" type="string"> <column name="Country" /> </property> <property name="phone" type="string"> <column name="Phone" /> </property> <property name="fax" type="string"> <column name="Fax" /> </property> <property name="homePage" type="string"> <column name="HomePage" /> </property> <set name="productses" table="Products" inverse="true" lazy="true" fetch="select"> <key> <column name="SupplierID" /> </key> </pre>
---	---

<pre> <set name="productses" table="Products" inverse="true" lazy="true" fetch="select"> <key> <column name="SupplierID" /> </key> <one-to-many class="com.upc.proyectos.hibernateqlserver.ent idades.Products" /> </set> </class> </hibernate-mapping> </pre>	<pre> <one-to-many class="com.upc.proyectos.hibernateqlserver.ent idades.Products" /> </set> </class> </hibernate-mapping> </pre>
--	---

Utilizando sentencias HQL Query

Construimos el proyecto. Clic derecho en hibernate.cfg.xml y seleccionamos **Run SQL Query**. Escribiremos **from Products** y veremos esto:

HQL Query0 x								
Session: hibernate.cfg								
from Products								
Result SQL								
0 row(s) updated.; 77 row(s) selected.								
Categories	ProductId	ReorderLevel	ProductName	UnitsInStock	UnitPrice	UnitsOnOr...	Suppliers	QuantityP...
com.upc.pro...	1	10	Chai	39	18.0000	0	com.upc.pro...	10 boxes x ...
com.upc.pro...	2	25	Chang	17	19.0000	40	com.upc.pro...	24 - 12 oz b...
com.upc.pro...	3	25	Aniseed Syrup	13	10.0000	70	com.upc.pro...	12 - 550 ml ...
com.upc.pro...	4	0	Chef Anton'...	53	22.0000	0	com.upc.pro...	48 - 6 oz jars
com.upc.pro...	5	0	Chef Anton'...	0	21.3500	0	com.upc.pro...	36 boxes
com.upc.pro...	6	25	Grandma's B...	120	25.0000	0	com.upc.pro...	12 - 8 oz jars
com.upc.pro...	7	10	Uncle Bob's ...	15	30.0000	0	com.upc.pro...	12 - 1 lb pkgs.
com.upc.pro...	8	0	Northwoods...	6	40.0000	0	com.upc.pro...	12 - 12 oz jars
com.upc.pro...	9	0	Mishi Kobe N...	29	97.0000	0	com.upc.pro...	18 - 500 g p...
com.upc.pro...	10	0	Ikura	31	31.0000	0	com.upc.pro...	12 - 200 ml j...
com.upc.pro...	11	30	Queso Cabr...	22	21.0000	30	com.upc.pro...	1 kg pkg.
com.upc.pro...	12	0	Queso Manc...	86	38.0000	0	com.upc.pro...	10 - 500 g p...
com.upc.pro...	13	5	Konbu	24	6.0000	0	com.upc.pro...	2 kg box
com.upc.pro...	14	0	Tofu	35	23.2500	0	com.upc.pro...	40 - 100 g p...
com.upc.pro...	15	5	Genen Shouyu	39	15.5000	0	com.upc.pro...	24 - 250 ml ...
com.upc.pro...	16	10	Pavlova	29	17.4500	0	com.upc.pro...	32 - 500 g b...
com.upc.pro...	17	0	Alice Mutton	0	39.0000	0	com.upc.pro...	20 - 1 kg tins
com.upc.pro...	18	0	Carnarvon T...	42	62.5000	0	com.upc.pro...	16 kg pkg.
com.upc.pro...	19	5	Teatime Cho...	25	9.2000	0	com.upc.pro...	10 boxes x ...
com.upc.pro...	20	0	Sir Rodney's...	40	81.0000	0	com.upc.pro...	30 gift boxes
com.upc.pro...	21	5	Sir Rodney's...	3	10.0000	40	com.upc.pro...	24 pkgs. x 4...
com.upc.pro...	22	25	Gustaf's Knä...	104	21.0000	0	com.upc.pro...	24 - 500 g p...
com.upc.pro...	23	25	Tunnbröd	61	9.0000	0	com.upc.pro...	12 - 250 g p...
com.upc.pro...	24	0	Guaraná Fa...	20	4.5000	0	com.upc.pro...	12 - 355 ml ...
com.upc.pro...	25	30	NuNuCa Nu...	76	14.0000	0	com.upc.pro...	20 - 450 g gl...
com.upc.pro...	26	0	Gumbär Gu...	15	31.2300	0	com.upc.pro...	100 - 250 g ...
com.upc.pro...	27	30	Schoggi Sch...	49	43.9000	0	com.upc.pro...	100 - 100 g ...
com.upc.pro...	28	0	Rössle Saue...	26	45.6000	0	com.upc.pro...	25 - 825 g c...



Con estas sentencias HQL procederemos a construir las clases Helper para consumir los recursos de la base de datos.

Helper de Productos

Creamos una clase ProductHelper y ponemos lo siguiente, con lo cual se iniciará una sólo instancia para productos:

```
private Session sesion;
private Transaction tx;

public ProductHelper() {
}

private void iniciaOperacion() throws HibernateException {
    sesion = HibernateUtil.getSessionFactory().openSession();
    tx = sesion.beginTransaction();
}

private void manejaExcepcion(HibernateException he) throws HibernateException {
    tx.rollback();
    throw new HibernateException("Ocurrió un error en la capa de acceso a datos", he);
}
```

Añadiremos el método para obtener un producto por ID:

```
public Products getProductById(int productId) {
    Products product = null;
    try {
        iniciaOperacion();
        product = (Products) sesion.get(Products.class, productId);
    } finally {
        sesion.close();
    }
    return product;
}
```

Método para obtener todos los productos:

```
public List getProducts() {
    List<Products> productList = null;
    try {
        iniciaOperacion();
        productList = sesion.createQuery("from Products").list();
    } finally {
        sesion.close();
    }
}
```

```
    }  
    return productList;  
}
```

Método para insertar un nuevo producto:

```
public Products saveProduct(Products product) throws HibernateException {  
    int id = 0;  
    try {  
        iniciaOperacion();  
        id = (int) sesion.save(product);  
        tx.commit();  
        product.setProductId(id);  
    } catch (HibernateException he) {  
        manejaExcepcion(he);  
        throw he;  
    } finally {  
        sesion.close();  
    }  
    return product;  
}
```

Actualizar un producto:

```
public Products updateProduct(Products product) throws HibernateException {  
    try {  
        iniciaOperacion();  
        sesion.update(product);  
        tx.commit();  
    } catch (HibernateException he) {  
        manejaExcepcion(he);  
        throw he;  
    } finally {  
        sesion.close();  
    }  
    return product;  
}
```

Eliminar un producto:

```
public void deleteProduct(Products product) throws HibernateException {  
    try {  
        iniciaOperacion();  
        sesion.delete(product);  
        tx.commit();  
    } catch (HibernateException he) {  
        manejaExcepcion(he);  
        throw he;  
    } finally {  
        sesion.close();  
    }  
}
```

PROBANDO LA FUNCIONALIDAD

Crearemos un MainClass y lo siguiente probará las funcionalidades:

```
public static void main(String[] args) {
    ProductHelper productHelper = new ProductHelper();

    //Listar todos los productos
    List<Products> listProducts = productHelper.getProducts();
    System.out.println("====LISTAR====");
    for (Products products : listProducts) {
        System.out.println(products.getId() + ">" + products.getName());
    }

    //Obtener un producto por id
    System.out.println("====OBTENER====");
    Products product = productHelper.getProductById(3);
    System.out.println(product.getId() + ">" + product.getName());

    //Insertar un producto
    System.out.println("====INSERTAR====");
    product = new Products();
    product.setName("Nuevo Producto");
    product.setDiscontinued(false);
    product = productHelper.saveProduct(product);
    System.out.println(product.getId() + ">" + product.getName());

    //Actualizar un producto
    System.out.println("====ACTUALIZAR====");
    product.setName("Nombre Actualizado Producto");
    product = productHelper.updateProduct(product);
    System.out.println(product.getId() + ">" + product.getName());

    //Listar todos los productos nuevamente
    System.out.println("====LISTAR NUEVAMENTE====");
    listProducts = productHelper.getProducts();
    for (Products products : listProducts) {
        System.out.println(products.getId() + ">" + products.getName());
    }

    //Eliminar un producto
    System.out.println("====ELIMINAR====");
    productHelper.deleteProduct(product);

    //Listar todos los productos nuevamente
    System.out.println("====LISTAR NUEVAMENTE====");
    listProducts = productHelper.getProducts();
    for (Products products : listProducts) {
        System.out.println(products.getId() + ">" + products.getName());
    }
}
```


Veamos los resultados.

Listar

```
Hibernate: select products0_.ProductID as ProductID_1_0_,
=====LISTAR=====
1=>Chai
2=>Chang
3=>Aniseed Syrup
4=>Chef Anton's Cajun Seasoning
5=>Chef Anton's Gumbo Mix
6=>Grandma's Boysenberry Spread
7=>Uncle Bob's Organic Dried Pears
8=>Northwoods Cranberry Sauce
9=>Mishi Kobe Niku
10=>Ikura
11=>Queso Cabrales
12=>Queso Manchego La Pastora
13=>Konbu
14=>Tofu
15=>Genen Shouyu
16=>Pavlova
17=>Alice Mutton
18=>Carnarvon Tigers
19=>Teatime Chocolate Biscuits
20=>Sir Rodney's Marmalade
```

Obtener

```
=====OBTENER=====
Hibernate: select products0_.ProductID as ProductID_1_0_,
3=>Aniseed Syrup
```

Insertar

```
=====INSERTAR=====
Hibernate: insert into NORTHWIND.dbo.Products
78=>Nuevo Producto
```

Actualizar

```
=====ACTUALIZAR=====
Hibernate: update NORTHWIND.dbo.Products set CategoryID=?,
78=>Nombre Actualizado Producto
```

Eliminar

```
=====ELIMINAR=====
Hibernate: delete from NORTHWIND.dbo.Products where ProductID=?
```