

Grado Superior en Informática

Acceso a Datos

Unidad 3: Mapeo Objeto Relacional

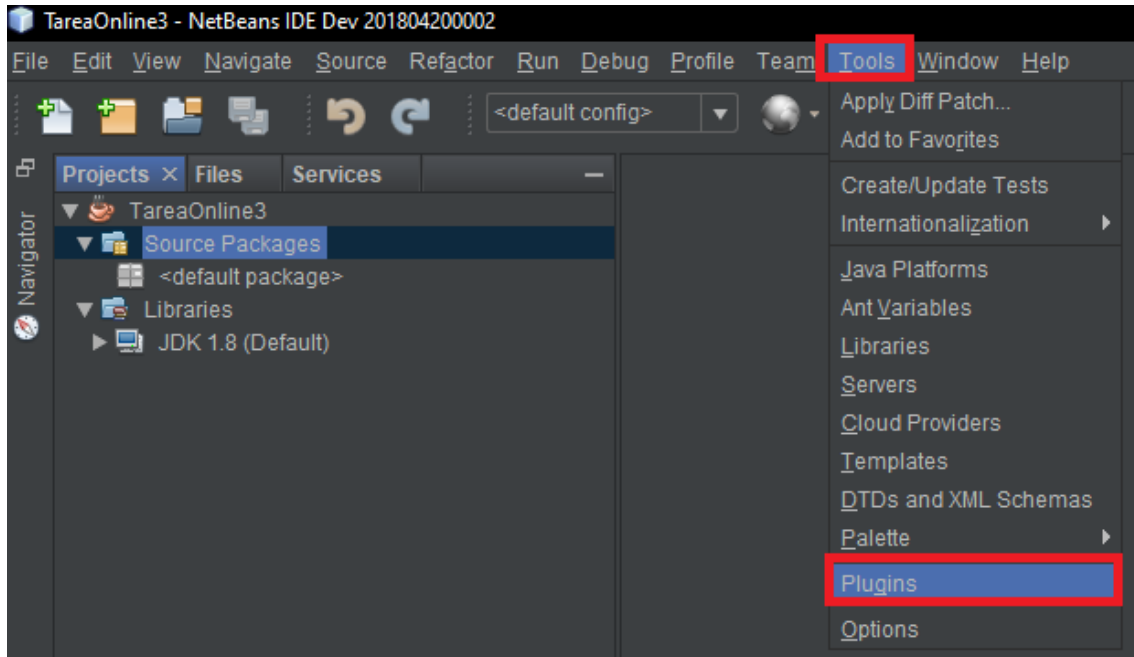
Tarea online 3



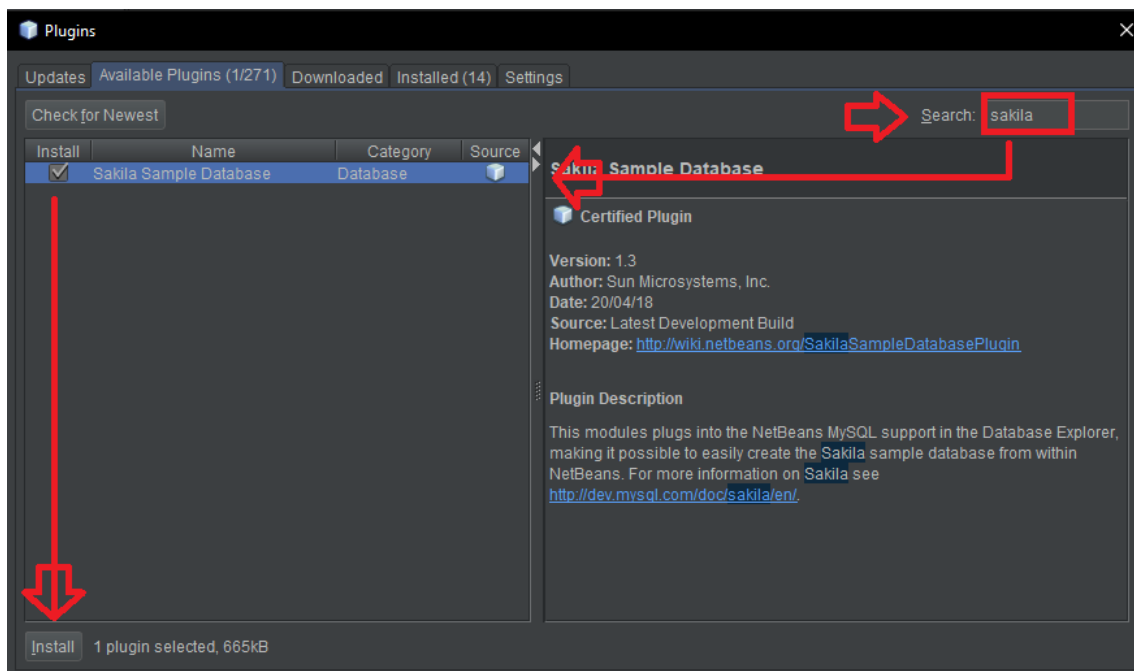
Autor: Raúl Serrano Torres.

Ejercicio 1: Instalación de la base de datos MySQL Sakila.

Para instalar la base de datos de ejemplo Sakila, tendremos que seleccionar **Tools** > **Plugins**.

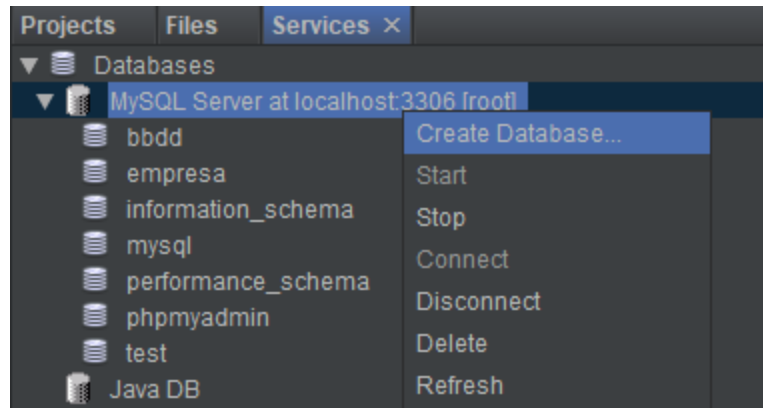


Dentro de **Plugins**, introduciremos “sakila” en el buscador. Después tendremos que seleccionar el plugin y pulsar el botón **instalar**.

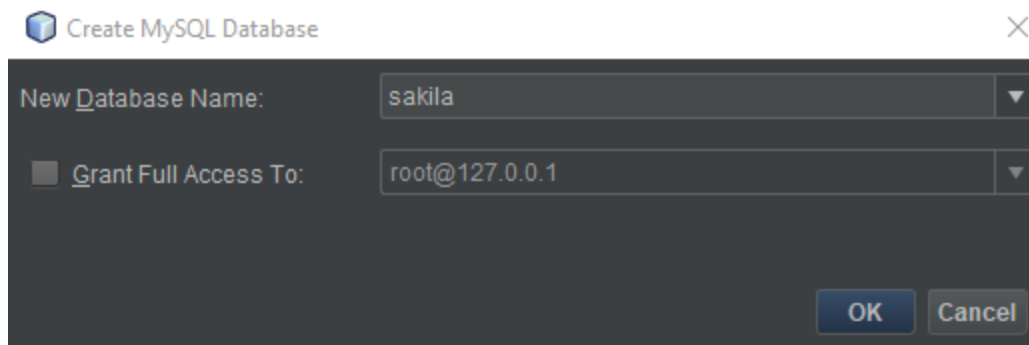


1.1 Conexión.

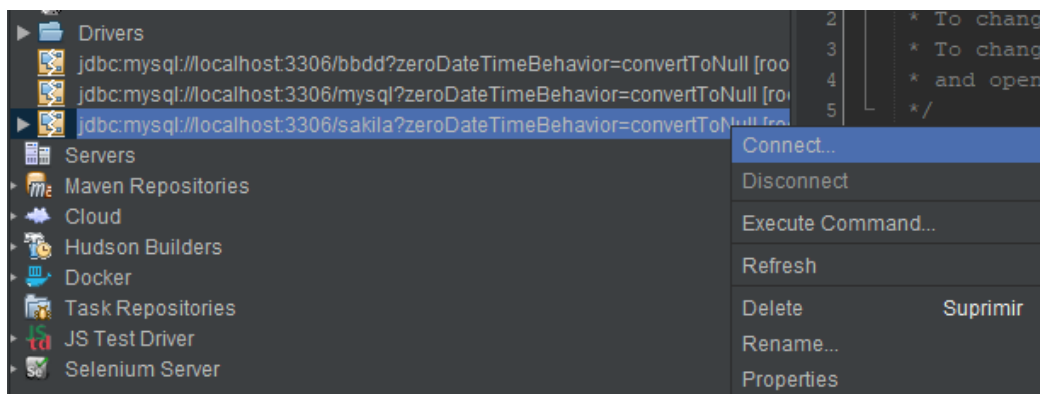
Para realizar la conexión a Sakila, tendremos que irnos a la pestaña **Servicios** de nuestro proyecto. Allí, expandiremos la pestaña **Databases** y en la pestaña **MySQL Server** haremos click izquierdo y **Create Database**.



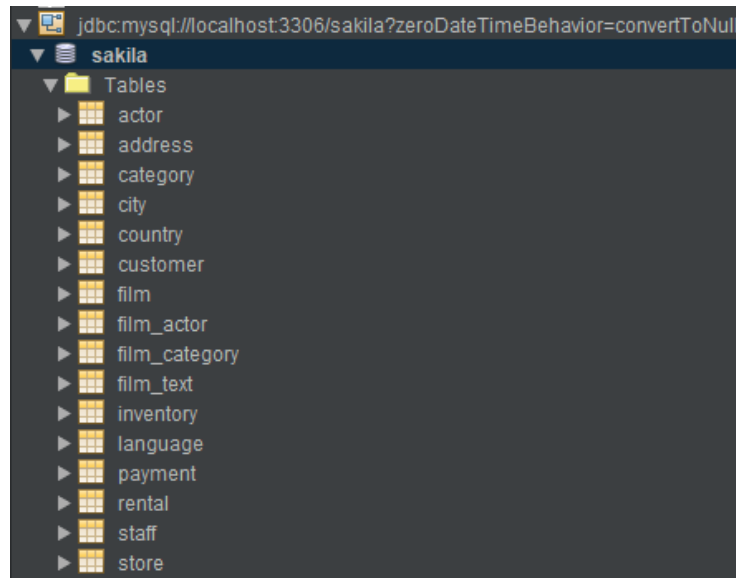
Aparecerá una ventana donde podremos elegir la base de datos que queremos crear. Elegiremos **sakila** y pulsaremos el botón **OK**.



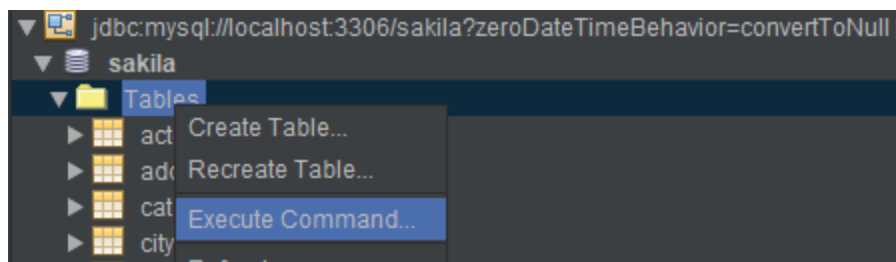
Esto añadiré el driver correspondiente para poder manipular la base de datos. Para conectarnos a ella, daremos click derecho y seleccionaremos **Connect**.



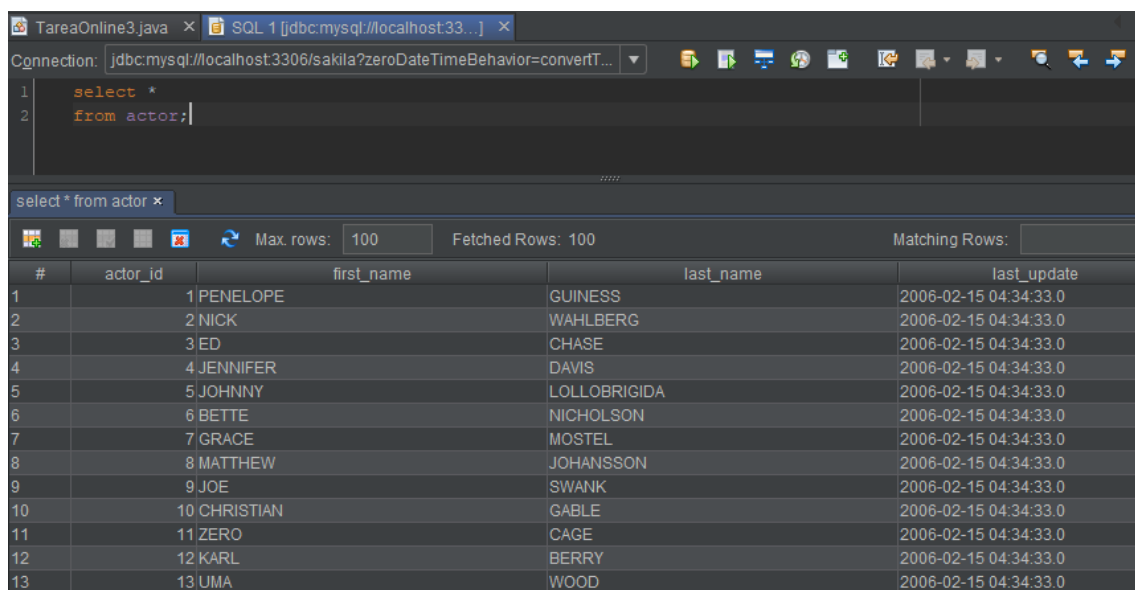
Con este último paso, ya tendremos conectada nuestra base de datos.



Si hacemos click derecho en Tables, podremos ejecutar un comando SQL para comprobar que se pueden realizar consultas.

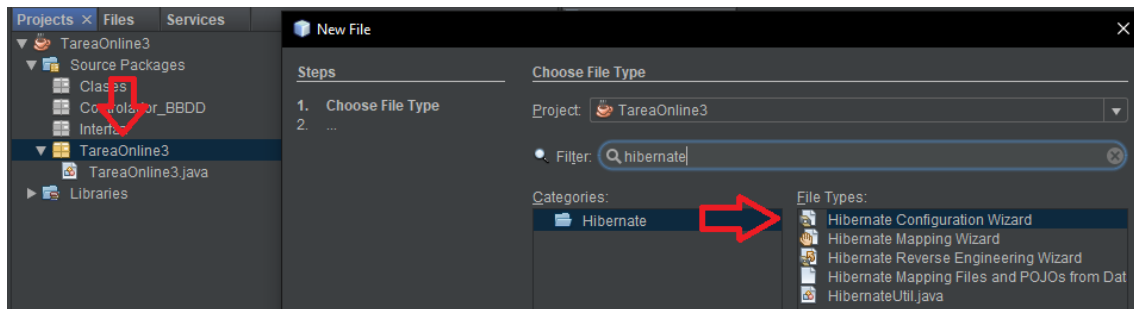


Para no complicarnos mucho, simplemente haremos que nos devuelva el contenido de la tabla actor.

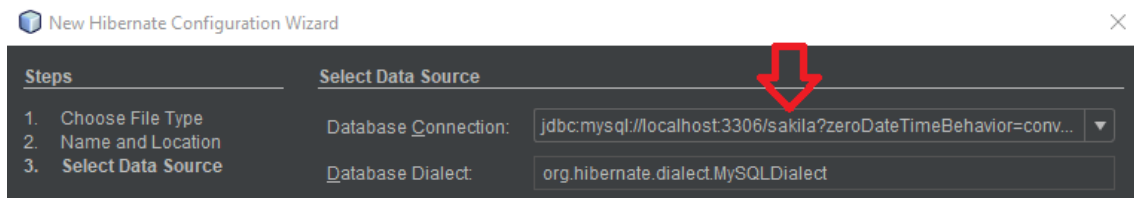


Ejercicio 2: Fichero de configuración de Hibernate.

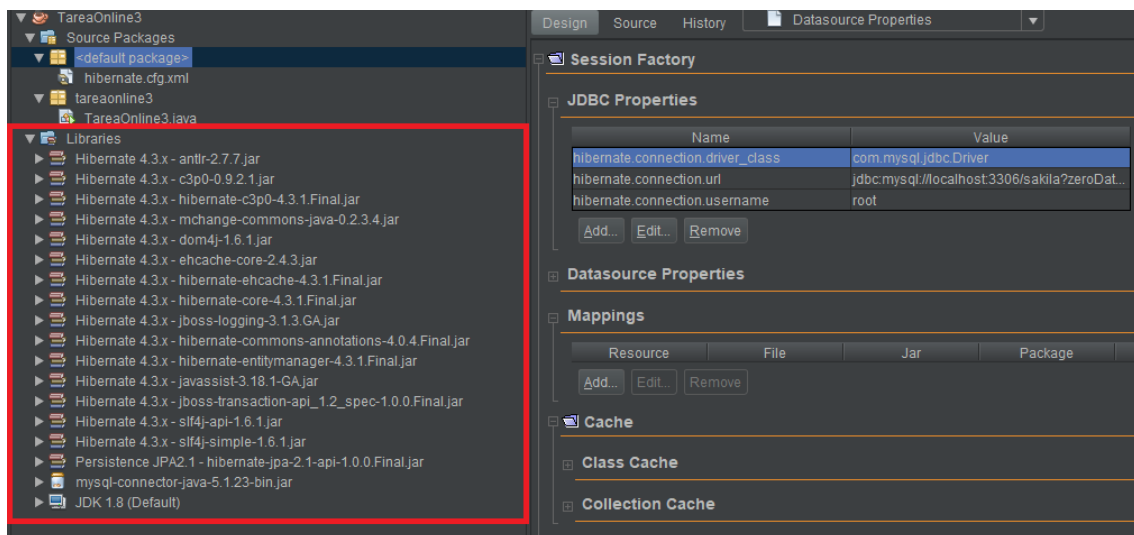
El primer paso será seleccionar el paquete por defecto de nuestro proyecto y añadiremos el archivo de configuración de hibernate.



Lo llamaremos **hibernate.cfg**. En el siguiente paso, seleccionaremos la base de datos con la que vamos a trabajar. En este caso **sakila**.

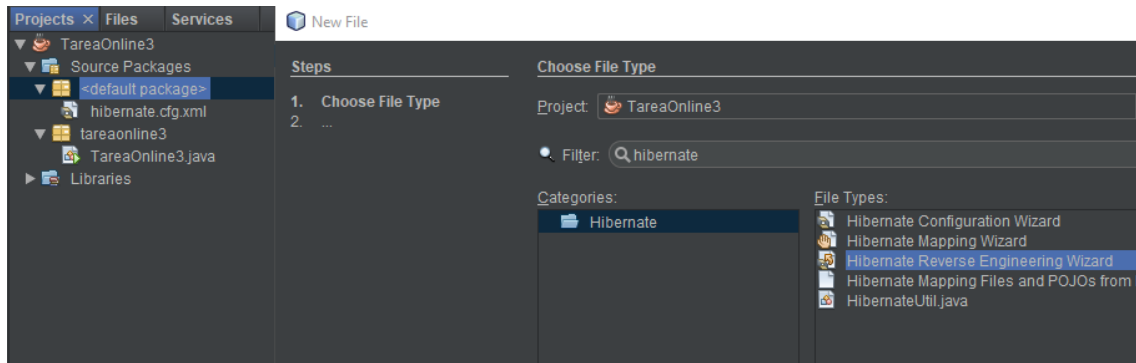


Tras esto, pulsaremos **Finish**, en la parte inferior de la ventana y se nos añadirá este archivo a un nuevo paquete (<default package>). También se añadirán las librerías necesarias para que podamos trabajar.

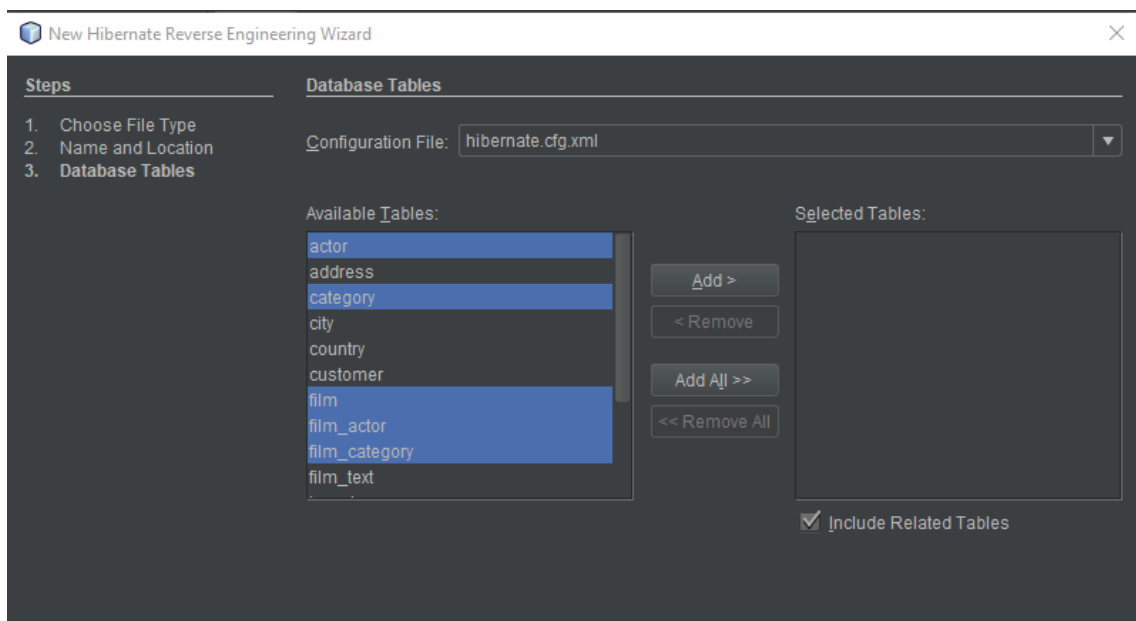


Ejercicio 3: Asistente de archivos de ingeniería inversa.

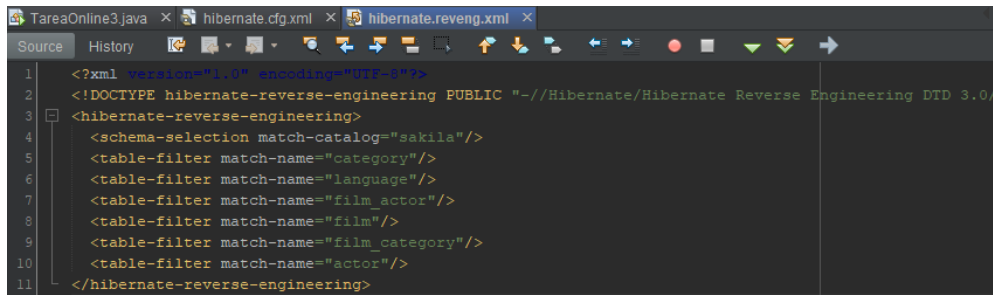
Haremos click izquierdo en el paquete que se generó en el paso anterior, el mismo donde tenemos el archivo de configuración de hibernate. Le añadiremos un archivo **Hibernate Reverse Engineering Wizard**.



Lo llamaremos **hibernate.reveng**. A continuación, seleccionaremos las tablas con las que trabajaremos posteriormente. En este caso actor, film, film_actor, film_category y category.

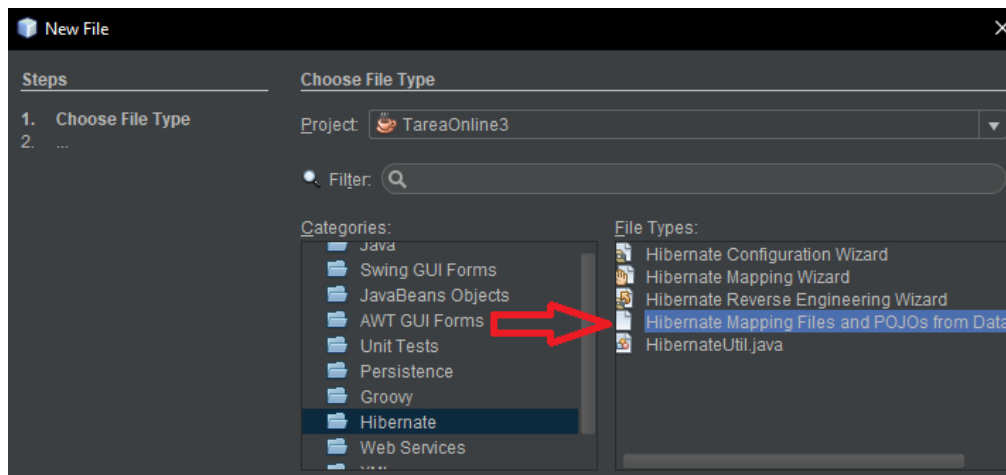


Las añadiremos pulsando el botón **Add>**. Cuando hayamos terminado, pulsaremos el botón **Finish** para añadir el fichero xml .

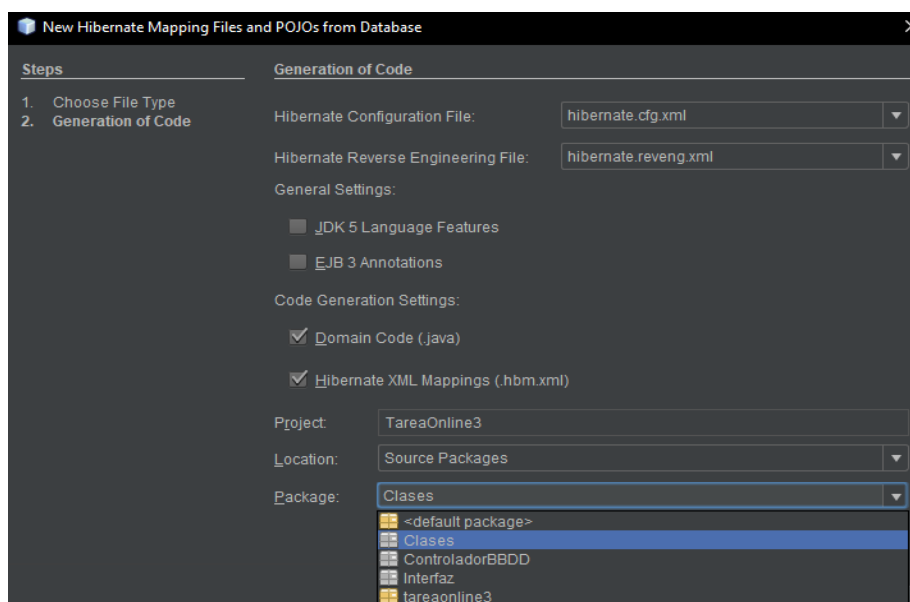


```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE hibernate-reverse-engineering PUBLIC "-//Hibernate/Hibernate Reverse Engineering DTD 3.0/"
3 <hibernate-reverse-engineering>
4   <schema-selection match-catalog="sakila"/>
5   <table-filter match-name="category"/>
6   <table-filter match-name="language"/>
7   <table-filter match-name="film_actor"/>
8   <table-filter match-name="film"/>
9   <table-filter match-name="film_category"/>
10  <table-filter match-name="actor"/>
11 </hibernate-reverse-engineering>
```

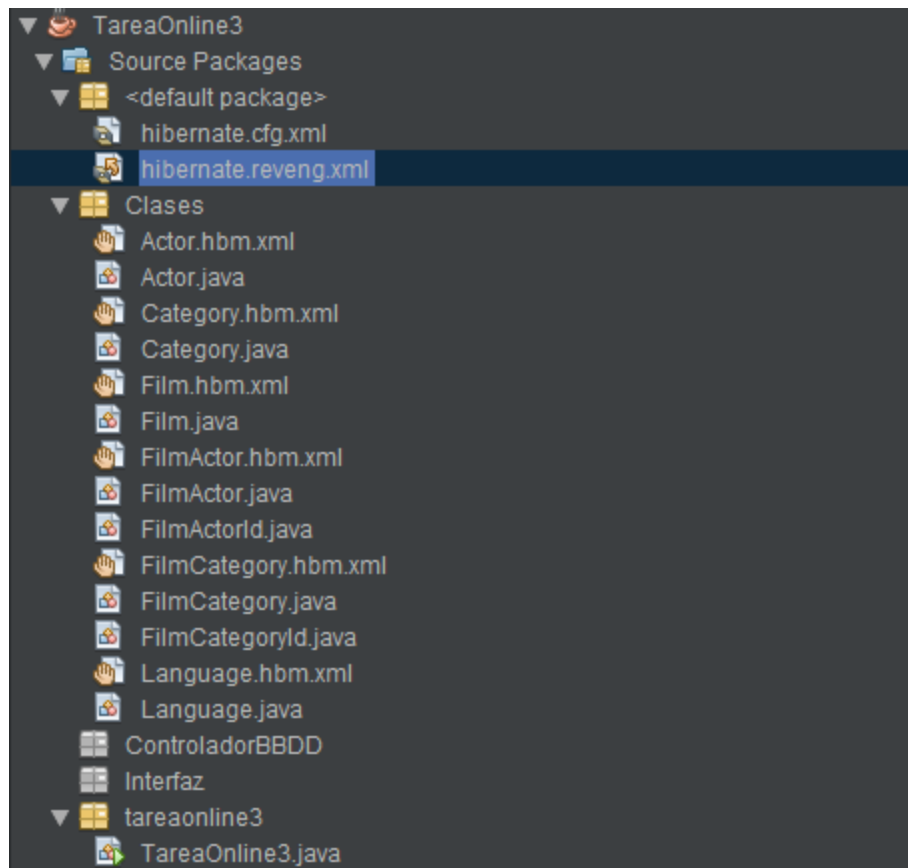
Por último, añadiremos las clases. Para ello, seleccionaremos cualquier paquete de nuestro proyecto y añadiremos **Hibernate Mapping Files and POJOS from Database**.



Tan solo tendremos que elegir el paquete donde irán los ficheros con las clases. Elegiremos el paquete **Clases** que debió ser creado anteriormente.



Hecho esto, pulsaremos el botón **Finish** y se añadirán las clases correspondientes a las tablas que elegimos en el ejercicio anterior.



Ejercicio 4. Implementación en Swing de consultas.

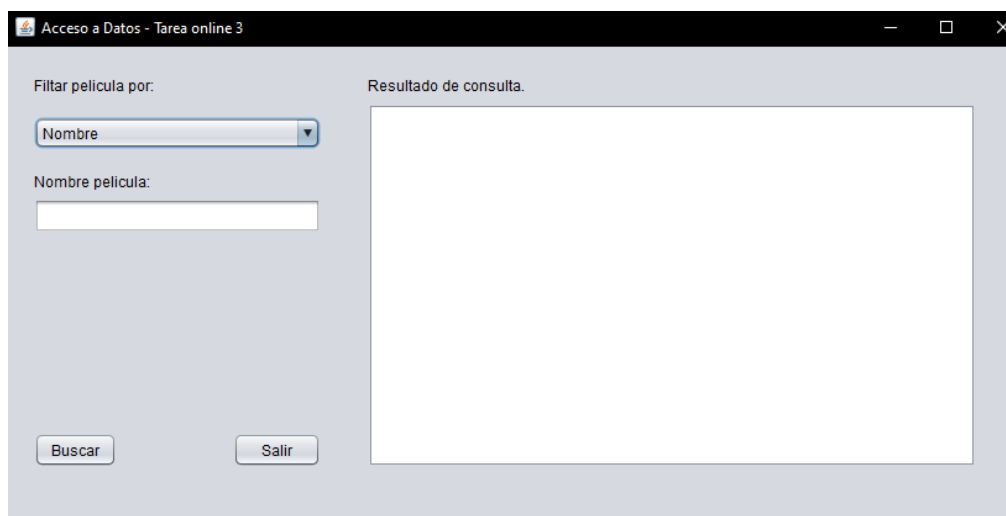
4.1 Interfaz.

La interfaz consta de los siguientes componentes.

- **ComoBox con opciones de filtrado** (cmbFiltro) para buscar películas (Film). Estas opciones son:
 - Nombre: filtra por título de la película.
 - Categoría: filtra por categoría de la película.
 - Actor: permite buscar una película en función de un actor que aparezca en ella.
 - Características especiales: permite realizar una consulta en función de una característica especial
 - Longitud: realiza una consulta que devuelve las películas cuya longitud es menos que 100.
- En función de la **opción** seleccionada en el comboBox:
 - **TextField txtFiltro**: Permite introducir el título de una **película** o el nombre de un **actor** para realizar una consulta (para las opciones Nombre y Actor)
 - **ComboBox comboElemento**: Permite seleccionar la **categoría** y la **característica especial** que se quiere usar para realizar una consulta (Para las opciones Categoría y Características especiales).

* En el caso de la opción **Longitud < 100**, se ocultan la segunda opción de filtrado, pues la consulta a realizar ya viene con los parámetros incluidos.

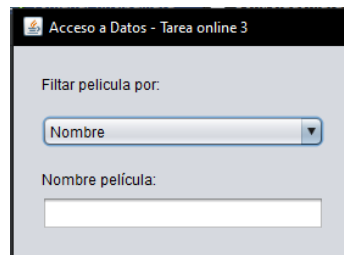
- **Botón Buscar**: Confirma la opción de búsqueda para realizar la consulta e imprime esta en el JList de la parte derecha.
- **Botón Salir**: Sale de la aplicación
- **JList**: Zona donde se cargarán las consultas realizadas cuando se pulse el botón Buscar.



4.1.2 Funcionamiento de la interfaz.

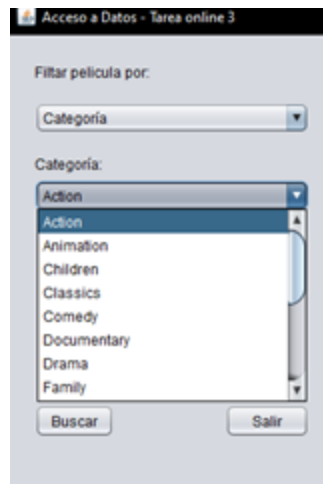
La interfaz funciona de la siguiente forma:

1. Se elige la opción de filtrado.
2. En función de la opción seleccionada:
 - a. Nombre o Actor: se rellenará el JTextField de la zona inferior con el nombre de la película o del actor que se usará para realizar la consulta. Esta opción oculta el comboBox para elegir categoría o característica especial.



- b. Categoría o característica especial: se elegirá del comboBox que aparecerá la categoría o la característica que se usará para

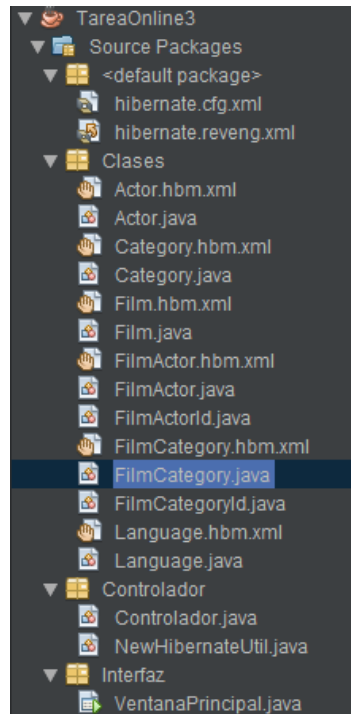
realizar la consulta. Esta opción oculta el JTextField para escribir el nombre o el actor de la película.



- c. Longitud < 100: Oculta los elementos que se utilizan para la segunda opción de filtrado.
3. Se pulsa el botón Buscar para realizar la consulta y cargarla en el JList.
4. Se realiza una nueva búsqueda o se sale del programa con el botón Salir.

4.2 Código.

A continuación, se expondrá el código de la aplicación, comenzando por los paquetes que la componen:



Paquetes:

- <defaultPackage>: Contiene el fichero de **configuración** de hibernate (hibernate.cfg.xml) y el fichero de **ingeniería inversa** de hibernate (hibernate.reveng.xml).
- Clases: Contiene las clases, creadas con hibernate, que se corresponden con las tablas de la base de datos sakila que se usa en el ejercicio.
- Controlador: Contiene la clase **Controlador**, que contiene los métodos que realizan consultas a la base de datos. También contiene el fichero **NewHibernateUtil**, que es necesario para crear las sesiones que se utilizan en la clase Controlador.
- Interfaz: Contiene la interfaz de la aplicación en fichero VentanaPrincipal.

Métodos de la clase controlador:

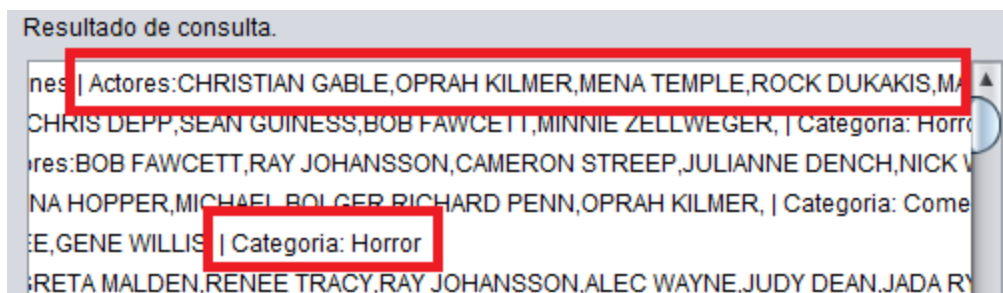
- public DefaultComboBoxModel btenerCategorias(): Este método es usado para cargar el comboBox **comboElemento**. Realiza una consulta en la base de datos que devuelve los distintos tipos de categorías que hay. Pasa estas a un ArrayList que se utiliza para rellenar un modelo de ComboBox el cual se pasará a la interfaz para que esta lo cargue.
- public DefaultListModel buscarPorNombre(String titulo): Desde la **VentanaPrincipal** se llama a este método, al cual se le pasa un String con el nombre del título de la película que se quiere buscar. Si se envía una parte del título buscará todas las películas que empiecen de la misma forma.

Devuelve un modelo de JList, el cual será directamente cargado en el JList que muestra las consultas.

- `public DefaultListModel buscarPorActor(String actor):` Funciona exactamente igual que **buscarPorNombre()**.
- `public DefaultListModel buscarPorCaracteristicas(String caracteristica):` Se le pasa un String, extraído de la opción seleccionada en **comboElemento**. Este String se usará como parámetro para realizar una consulta de las películas que cuentan con la característica que contiene dicho String. Se devuelve un modelo de JList que se cargará en el JList que muestra las consultas.
- `public DefaultListModel buscarPorCategoria(String categoria):` funciona de forma similar a `buscarPorCaracteristicas`.
- `public DefaultListModel buscarPorLongitud():` Este método no recibe parámetros. Devuelve un modelo de JList con todas las películas cuya longitud es inferior a 100.

Clases:

Las clases se han dejado tal y como las ha creado hibernate. Tan solo se han modificado los `toString()` de las clases **Film**, **Actor**, **FilmActor**, **Category** y **filmCategory** para realizar una correcta impresión de los atributos categoría y actor de la clase/tabla Film, pues antes en lugar de los actores y la categoría de las películas devueltas se cargaba su id. Ahora se muestra de forma mucho más clara:



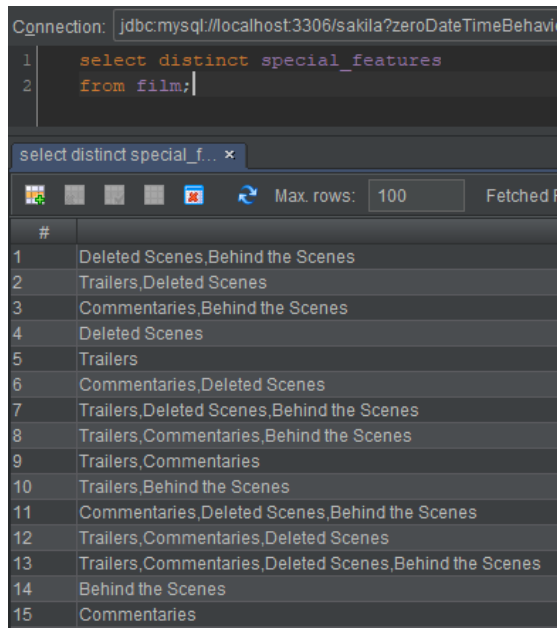
También se ha eliminado la información que se ha considerado menos relevante para la mejor lectura de las consultas.

Interfaz:

De los métodos creados en la interfaz destacan tres:

- `cargaCaracteristicasEspeciales():` Carga el `comboElemento` con las características especiales. Se cargan directamente desde la interfaz en lugar de llamando a un método porque al realizar la consulta para obtener las `special_features` se encontró que no se podían extraer de una en una, ya que

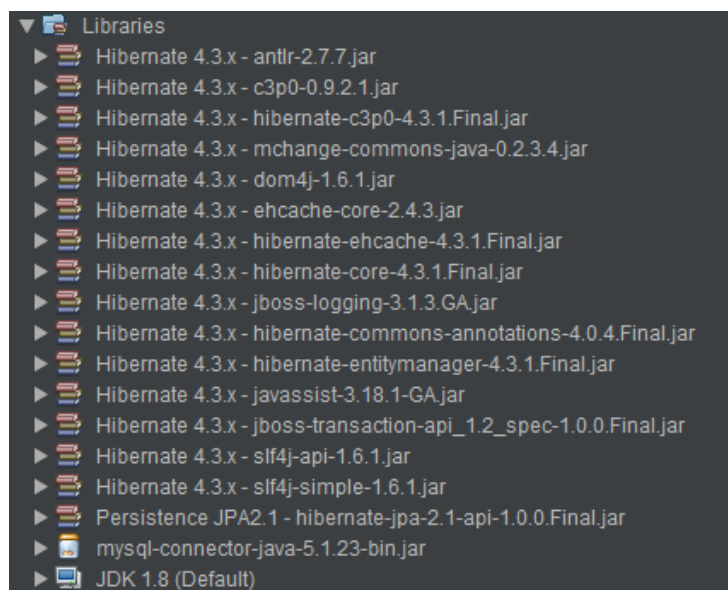
en ese atributo de Film se pueden encontrar varias a la vez como se muestra en la siguiente imagen:



- actualizarInterfaz(): En función del elemento seleccionado en **cmbFiltro** muestra unos elementos u otros.
- btnBuscarActionPerformed(): Llama a los distintos métodos de la clase controlador en función de la opción seleccionada en **cmbFiltro**.

Librerías usadas:

Estas son las librerías que son necesarias incluir en el proyecto para que el mismo funcione las cuales han sido eliminadas de la entrega para poder subir el archivo:



Deberán ser incluidas dentro de la carpeta **dist>lib**.