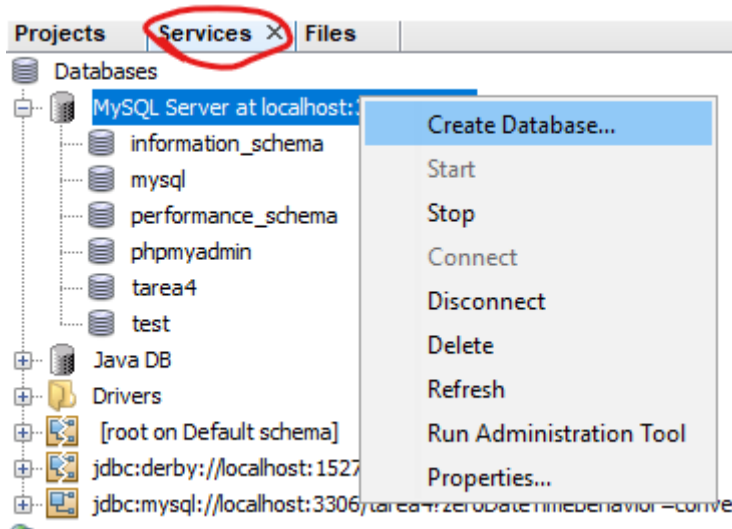


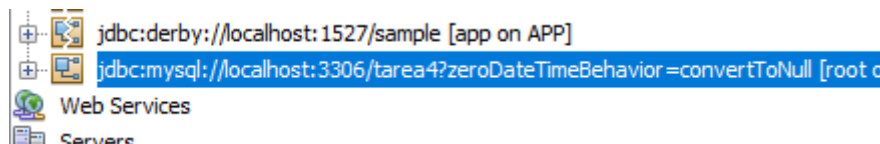
Antonio Jiménez Sevilla

AD04

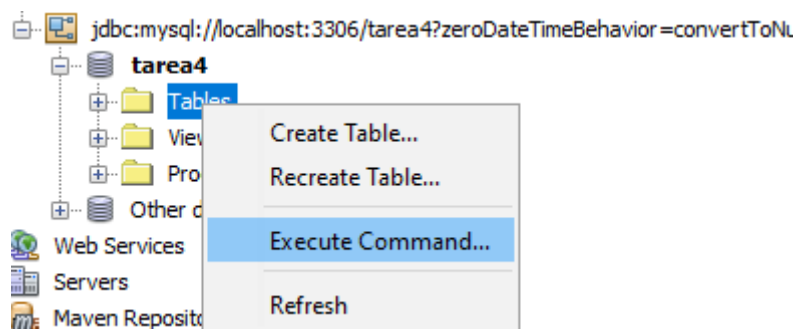
Lo primero de todo tenemos que comprobar la base de datos a la que vamos aplicar Hibernate. Para ello vamos a la pestaña **services** en netbeans, creamos una base de datos llamada tarea4, clicando con el botón derecho en MySQL server, ya que es la base de datos que vamos a usar.



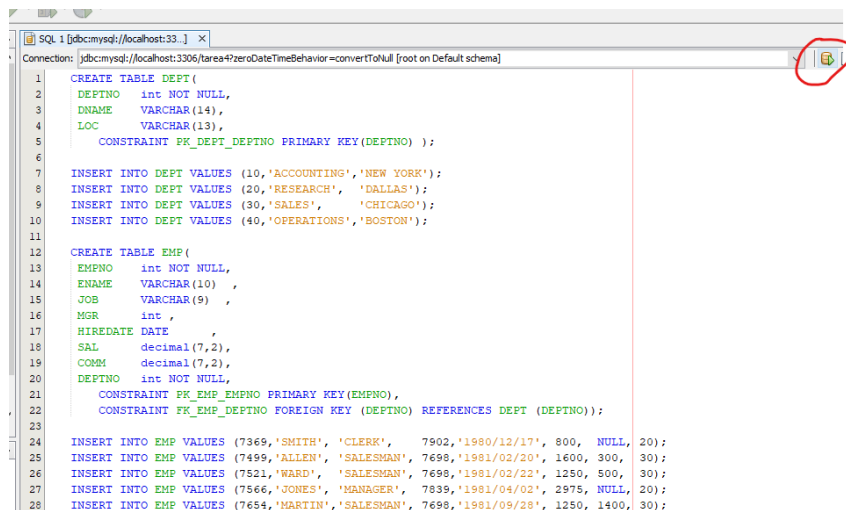
Una vez creada se nos crea un jdbc:mysql. Le damos botón derecho conectar.



Desplegamos la carpeta y con el botón derecho clicamos sobre Tables, y le damos a Execute Command.

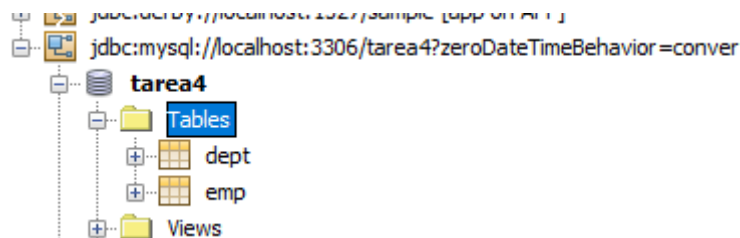


Copiamos y pegamos el fichero de las tablas del ejercicio para crear las tablas.

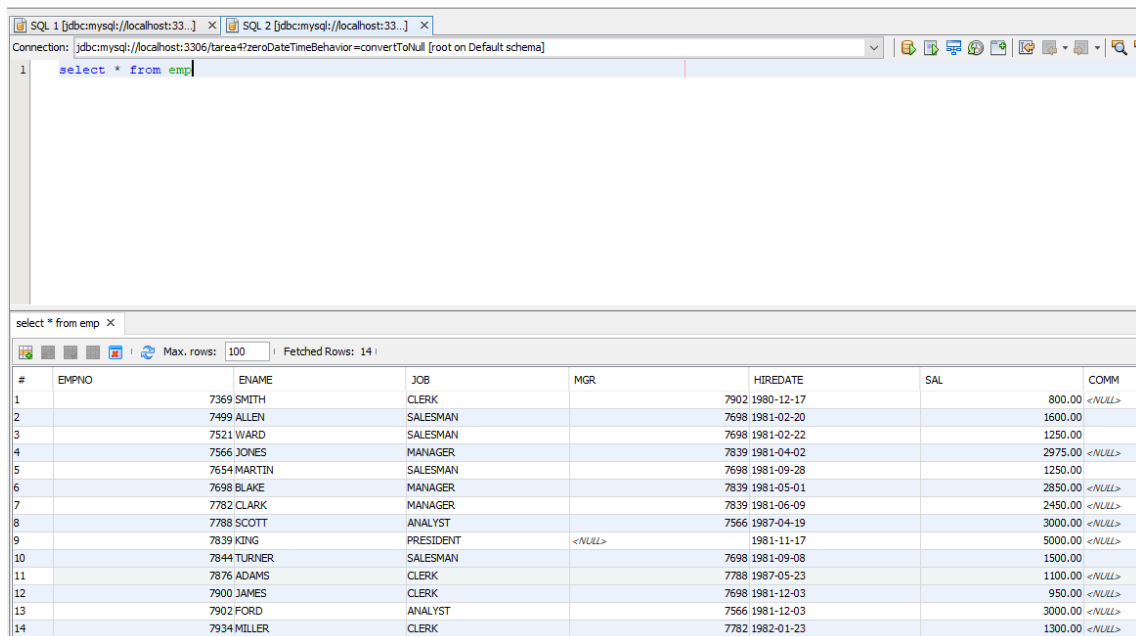


```
1 CREATE TABLE DEPT(  
2   DEPTNO int NOT NULL,  
3   DNAME  VARCHAR(14),  
4   LOC    VARCHAR(13),  
5   CONSTRAINT PK_DEPT_DEPTNO PRIMARY KEY (DEPTNO) );  
6  
7 INSERT INTO DEPT VALUES (10,'ACCOUNTING','NEW YORK');  
8 INSERT INTO DEPT VALUES (20,'RESEARCH','DALLAS');  
9 INSERT INTO DEPT VALUES (30,'SALES','CHICAGO');  
10 INSERT INTO DEPT VALUES (40,'OPERATIONS','BOSTON');  
11  
12 CREATE TABLE EMP (  
13   EMPNO int NOT NULL,  
14   ENAME  VARCHAR(10),  
15   JOB    VARCHAR(9),  
16   MGR    int,  
17   HIREDATE DATE,  
18   SAL    decimal(7,2),  
19   COMM   decimal(7,2),  
20   DEPTNO int NOT NULL,  
21   CONSTRAINT PK_EMP_EMPNO PRIMARY KEY (EMPNO),  
22   CONSTRAINT FK_EMP_DEPTNO FOREIGN KEY (DEPTNO) REFERENCES DEPT (DEPTNO));  
23  
24 INSERT INTO EMP VALUES (7369,'SMITH','CLERK',7902,'1980/12/17',800,NULL,20);  
25 INSERT INTO EMP VALUES (7499,'ALLEN','SALESMAN',7698,'1981/02/20',1600,300,30);  
26 INSERT INTO EMP VALUES (7521,'WARD','SALESMAN',7698,'1981/02/22',1250,500,30);  
27 INSERT INTO EMP VALUES (7566,'JONES','MANAGER',7839,'1981/04/02',2975,NULL,20);  
28 INSERT INTO EMP VALUES (7654,'MARTIN','SALESMAN',7698,'1981/09/28',1250,1400,30);
```

Vemos como se han creado perfectamente.



Para comprobar que funcionan perfectamente hacemos una consulta.



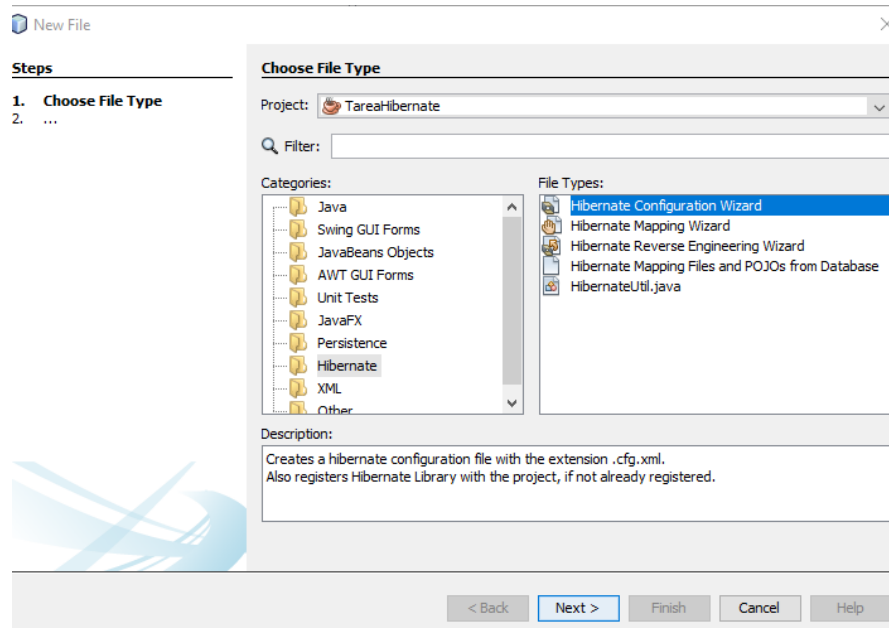
The screenshot shows the SQL Developer interface with a query window containing the statement 'select * from emp'. The results are displayed in a table with 14 rows and 8 columns: EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, and an unnamed column. The data is as follows:

#	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM
1	7369	SMITH	CLERK		1980-12-17	800.00	<NULL>
2	7499	ALLEN	SALESMAN		1981-02-20	1600.00	
3	7521	WARD	SALESMAN		1981-02-22	1250.00	
4	7566	JONES	MANAGER		1981-04-02	2975.00	<NULL>
5	7654	MARTIN	SALESMAN		1981-09-28	1250.00	
6	7698	BLAKE	MANAGER		1981-05-01	2850.00	<NULL>
7	7782	CLARK	MANAGER		1981-06-09	2450.00	<NULL>
8	7788	SCOTT	ANALYST		1987-04-19	3000.00	<NULL>
9	7839	KING	PRESIDENT	<NULL>	1981-11-17	5000.00	<NULL>
10	7844	TURNER	SALESMAN		1981-09-08	1500.00	
11	7876	ADAMS	CLERK		1987-05-23	1100.00	<NULL>
12	7900	JAMES	CLERK		1981-12-03	950.00	<NULL>
13	7902	FORD	ANALYST		1981-12-03	3000.00	<NULL>
14	7934	MILLER	CLERK		1982-01-23	1300.00	<NULL>

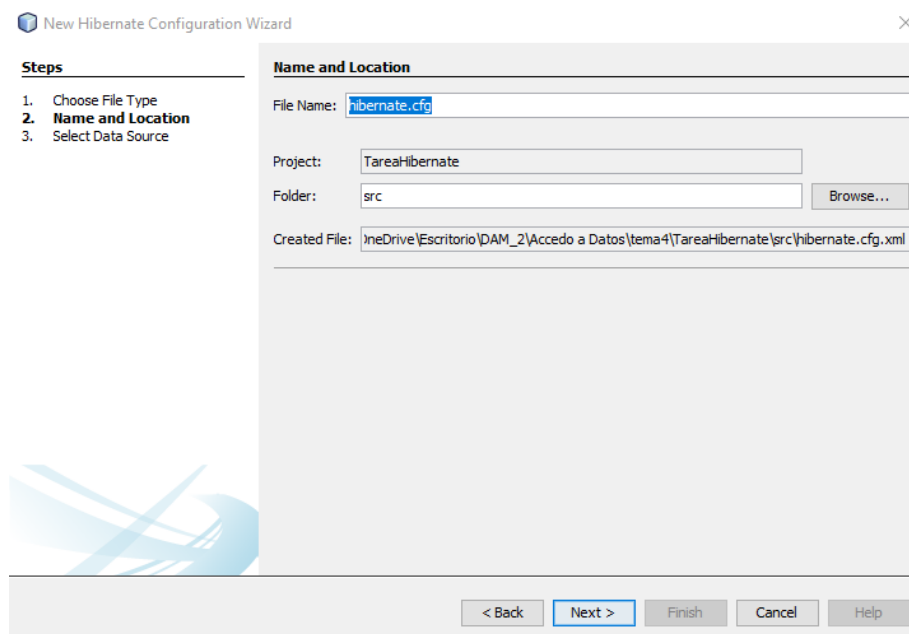
Ahora vamos a crear el fichero de configuración. Para empezar a trabajar con Hibernate es necesario configurar la herramienta para que conozca qué objetos debe recuperar de la base de datos relacional y en qué lugar los hará persistir. Por tanto, el primer paso será tener una base de datos relacional con la que poder trabajar.

Después de crear el fichero de configuración, éste puede ser editado usando el editor interactivo, o editar directamente el código XML.

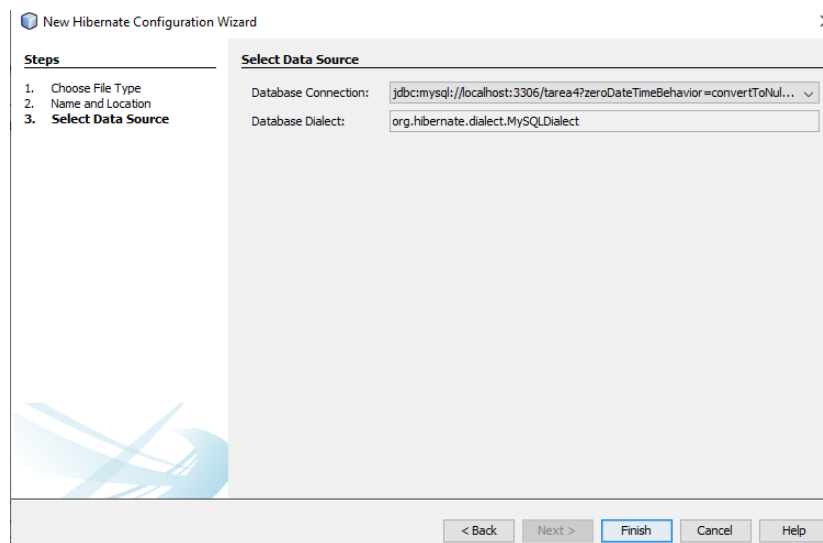
Para crear el fichero de configuración clicamos con el **botón derecho sobre el paquete, le damos a other, y seleccionamos Hibernate configuration Wizard.**



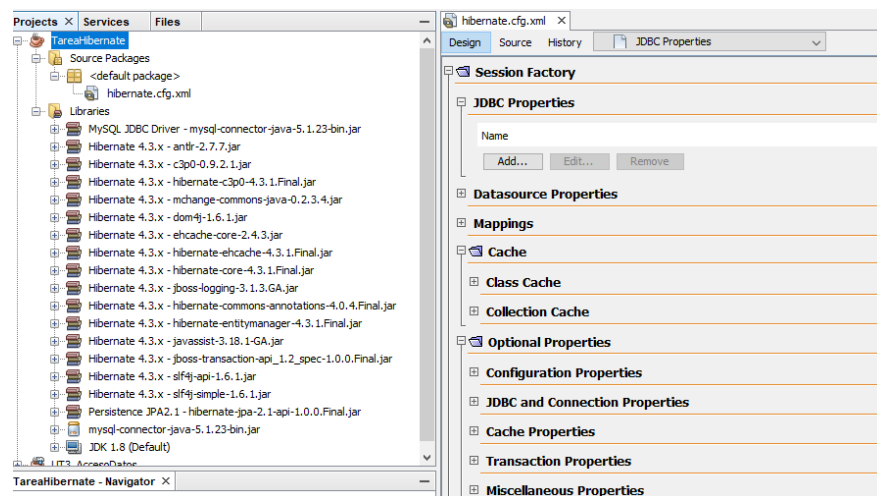
Le damos a next.



Ahora seleccionamos la base de datos creada anteriormente. En NetBeans, cuando se crea el archivo de configuración de Hibernate usando el asistente, podemos especificar la conexión a la base de datos, eligiendo de una lista de conexiones de bases de datos registradas en el IDE.



Cuando se genera el archivo de configuración, el IDE añade de forma automática detalles de la conexión e información basada en la conexión de la base de datos seleccionada. El IDE añade también las bibliotecas de Hibernate en el proyecto.



Como podemos observar en la pestaña **design** nos aparecen varias propiedades del fichero hibernate.cfg.xml que podemos cambiar o bien desde esta pestaña o escribiendo el xml.

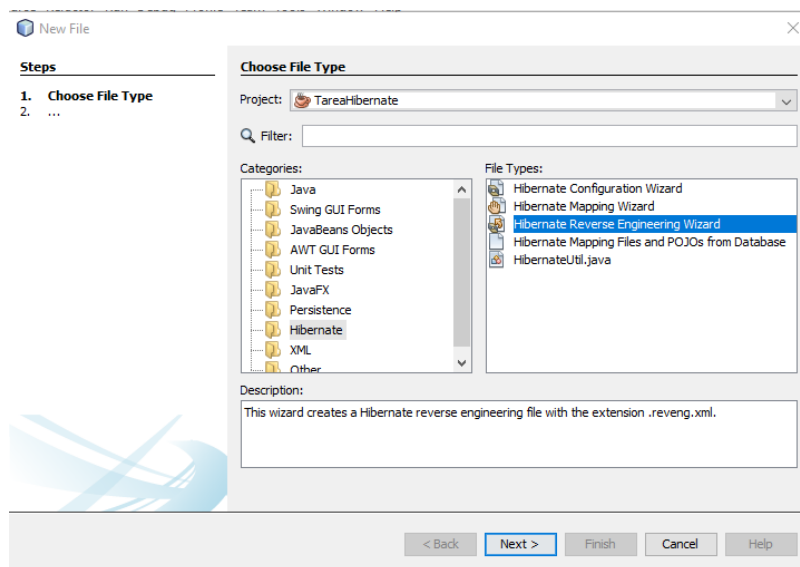
```
11  -->
12  <hibernate-configuration>
13  <session-factory>
14    <property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
15    <property name="hibernate.connection.username">root</property>
16    <property name="hibernate.query.factory_class">org.hibernate.hql.internal.ast.ASTQueryTranslatorFactory</property>
17    <property name="hibernate.show_sql">true</property>
18    <property name="hibernate.connection.url">jdbc:mysql://localhost:3306/tarea4</property>
19  </session-factory>
20  </hibernate-configuration>
21
```

Yo he cambiado:

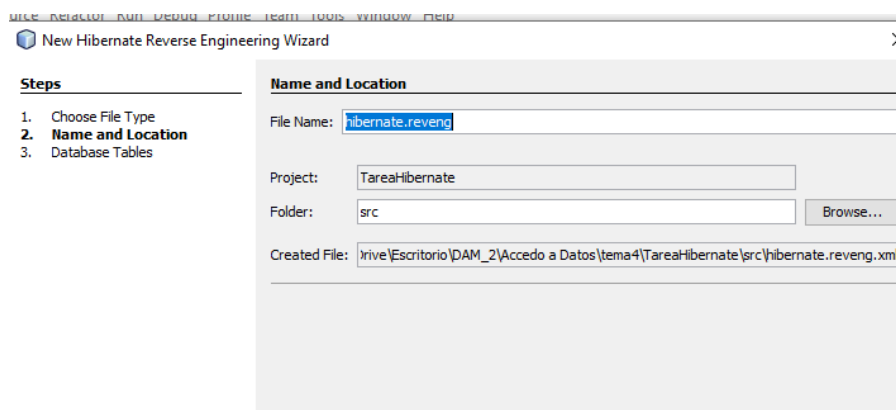
- **hibernate.connection.driver_class**: Driver utilizado para la conexión con la base de datos.
- **hibernate.connection.username**: Nombre del usuario que va a realizar la extracción de información. Por defecto, el nombre de usuario es root.
- **hibernate.query.factory_class**: Elige la implementación de análisis sintáctico *HQL*.
- **hibernate.show_sql**: Para mostrar la herramienta. Por defecto, su valor es true.
- **hibernate.connection.url**: Dirección de la base de datos con la que se va a conectar Hibernate.

Una vez realizado esto crearemos el archivo de ingeniería inversa, este fichero deberá estar en el mismo paquete que el **hibernate.cfg.xml**, la clase a persistir.

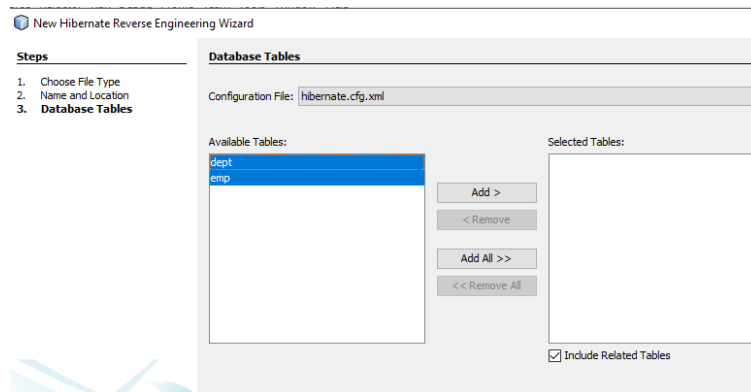
Para ello clicamos con el botón derecho en el paquete por defecto y esta vez seleccionaremos **Hibernate Reverse Engineering Wizard**.



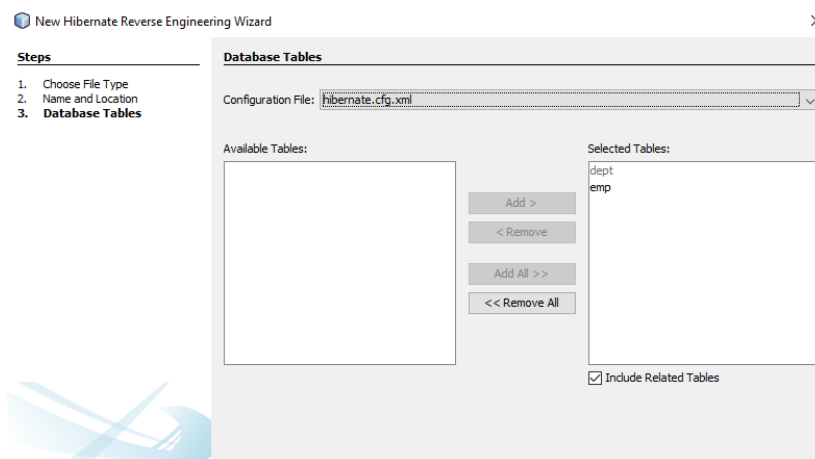
Indicamos el nombre dl fichero y su ruta.



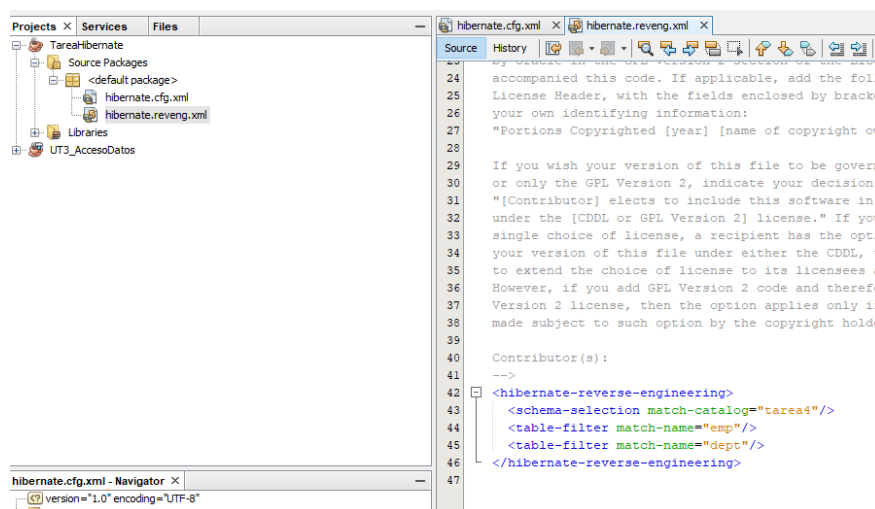
En este paso, vemos como en la parte superior, aparece el fichero de configuración de Hibernate creado anteriormente. Si todo es correcto, nos aparecen las tablas que forman la base de datos.



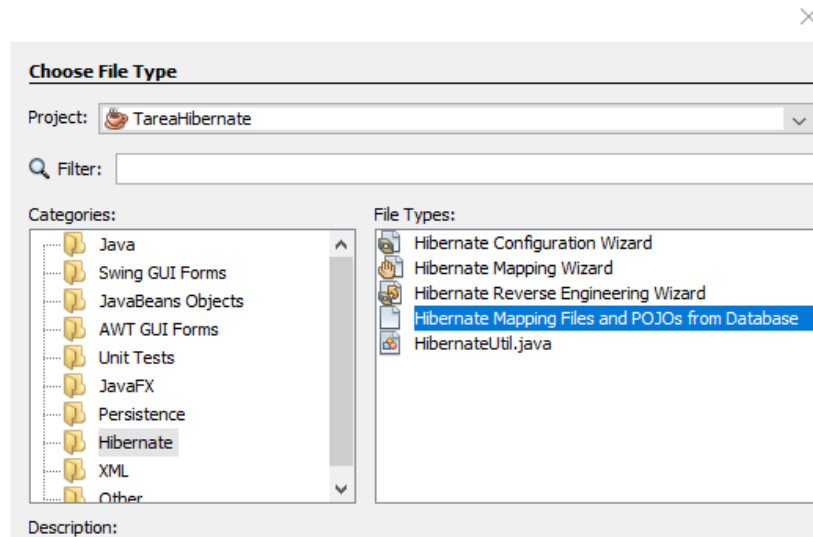
Seleccionamos las tablas y las añadimos a la columna de selected tables.



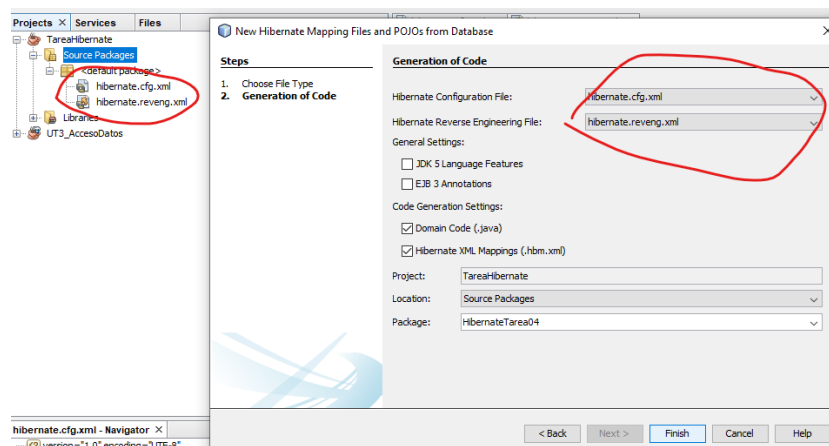
Una vez finalizado el asistente, ya tendríamos creado el archivo **Hibernate de ingeniería inversa** de Hibernate.



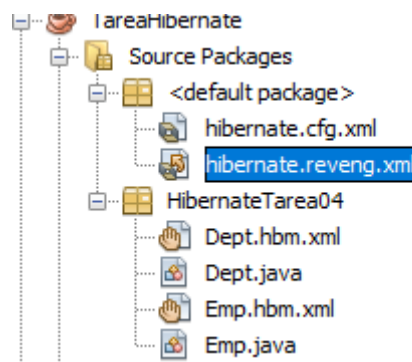
Ahora crearemos un **archivo de mapas de Hibernate y POJOs basados en una base de datos relacional existente**, es decir, generaremos las clases java, correspondientes a las tablas de la base de datos. Creamos el fichero botón de-
recho en el paquete.



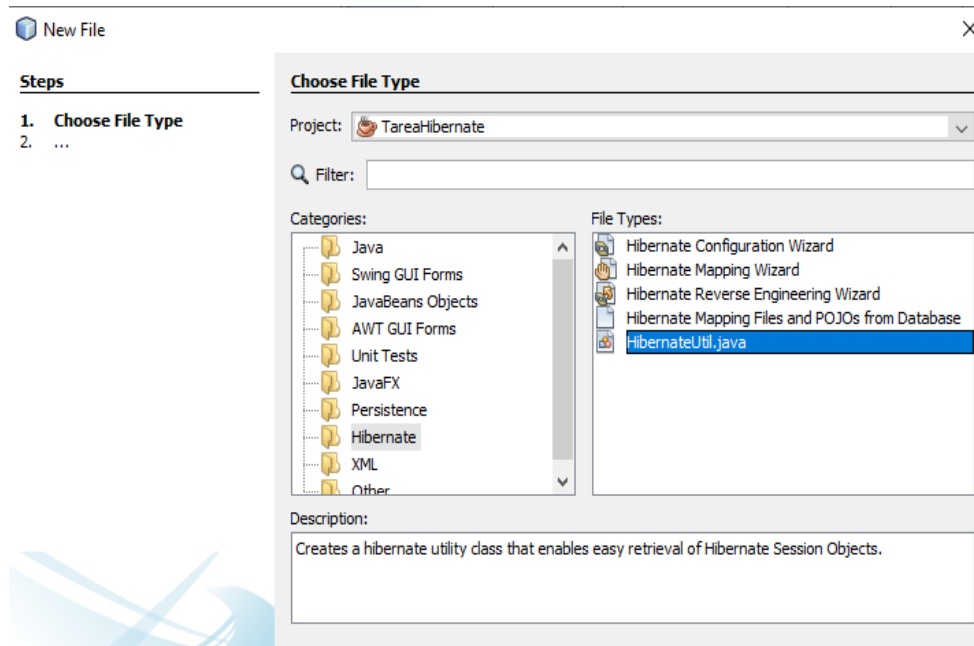
Proporcionamos un nombre al paquete y comprobamos que coincide la información del hibernate configuration file y el Hibernate Reverse.



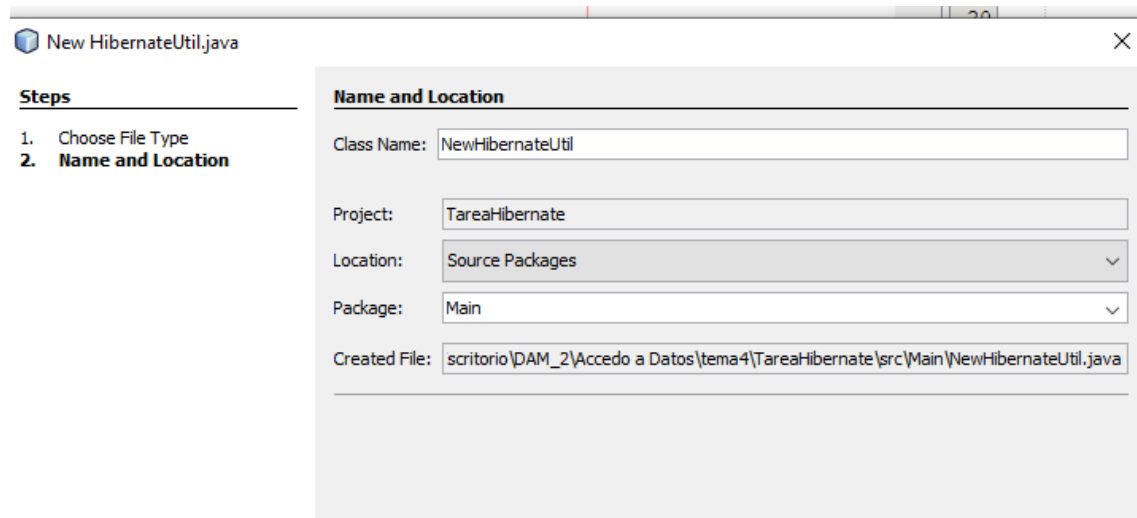
Una vez comprobado le damos a finalizar, generando las clases Java correspondientes. Así es como tenemos nuestro proyecto hasta ahora.



Por último, debemos crear un HibernateUtil, para poder establecer la conexión.



En este paso, únicamente informamos, el nombre del fichero y el paquete donde crearemos, los ficheros. Una vez, finalizado el asistente, ya tenemos creado nuestro **HibernateUtil**. Lo he metido en la carpeta Main que es donde voy a crear la clase principal.



Ahora para comprobar que todo está correcto, pinchamos con el botón derecho en el fichero de configuración **Hibernate.cfg.xml** y seleccionamos **run HQL**. Escribimos la consulta **from Emp** por ejemplo y nos sale lo que tiene la tabla Emp. (Este era el paso con el que me he estado peleando dos semanas, porque no me salían las tablas, me daba error de mapeo)

from Emp

<

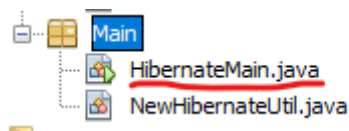
Result SQL

Set Ma

0 row(s) updated.; 14 row(s) selected.

Job	Dept	Sal	Comm	Mgr	Hiredate	Ename	Empno
CLERK	HibernateTarea04.Dept@5c340c63	800.00	NULL	7902	1980-12-17	SMITH	7369
SALESMAN	HibernateTarea04.Dept@2b796949	1600.00	300.00	7698	1981-02-20	ALLEN	7499
SALESMAN	HibernateTarea04.Dept@2b796949	1250.00	500.00	7698	1981-02-22	WARD	7521
MANAGER	HibernateTarea04.Dept@5c340c63	2975.00	NULL	7839	1981-04-02	JONES	7566
SALESMAN	HibernateTarea04.Dept@2b796949	1250.00	1400.00	7698	1981-09-28	MARTIN	7654
MANAGER	HibernateTarea04.Dept@2b796949	2850.00	NULL	7839	1981-05-01	BLAKE	7698
MANAGER	HibernateTarea04.Dept@179d3394	2450.00	NULL	7839	1981-06-09	CLARK	7782
ANALYST	HibernateTarea04.Dept@5c340c63	3000.00	NULL	7566	1987-04-19	SCOTT	7788
PRESIDENT	HibernateTarea04.Dept@179d3394	5000.00	NULL	NULL	1981-11-17	KING	7839
SALESMAN	HibernateTarea04.Dept@2b796949	1500.00	0.00	7698	1981-09-08	TURNER	7844
CLERK	HibernateTarea04.Dept@5c340c63	1100.00	NULL	7788	1987-05-23	ADAMS	7876
CLERK	HibernateTarea04.Dept@2b796949	950.00	NULL	7698	1981-12-03	JAMES	7900
ANALYST	HibernateTarea04.Dept@5c340c63	3000.00	NULL	7566	1981-12-03	FORD	7902
CLERK	HibernateTarea04.Dept@179d3394	1300.00	NULL	7782	1982-01-23	MILLER	7934

Creamos una clase principal en la carpeta Main donde está el **NewHibernateUtil**, y comenzamos a realizar la conexión con Hibernate, y realizar los insertos, los borrados y las consultas del ejercicio.



Para la realización de insertado y borrado he creado una clase llamada Queries donde he realizado los métodos para llamarlos en la clase principal. Intente hacerlo como venia en los apuntes, pero me daba muchos fallos.

```

public static void main(String[] args) throws ParseException {
    //creamos la conexion y la abrimos
    Session session = NewHibernateUtil.getSessionFactory().openSession();
    Transaction tx = session.beginTransaction();
    //
    Queries queries = new Queries();
    Query q = session.createQuery("from Dept");
    List<Dept> deps = q.list();
    Dept departament = deps.get(0);
    SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
    String stringFecha = "2021-01-15";
    Date fecha = sdf.parse(stringFecha);
    Emp e = new Emp();

    e.setJob("Informatico");
    e.setSal(new BigDecimal("2000"));
    e.setEmpno(7568);
    e.setEname("Antonio");
    e.setHiredate(fecha);
    e.setDept(departament);
    e.setMgr(7896);
    e.setComm(new BigDecimal("150"));

    session.save(e);
    tx.commit();
    session.close();
}
  
```

He creado la clase Queries, donde para hacer las consultas get se utiliza el patrón singleton donde solo se abre la sesión si es nula porque daba excepción al abrir sesión la segunda vez.

He creado un List, con el objeto query, para poder crear las consultas en SQL

```
public Session getSession() {
    //para poder obtener la session tube que mirar en internet para hacerlo
    //por que me daba excepcion al abrir por segunda vez
    if (session == null) {
        SessionFactory sessionFactory = NewHibernateUtil.getSessionFactory();
        session = sessionFactory.openSession();
    }
    return session;
}

public <T> List<T> consulta(String ConsultaSQL) {
    Query query = getSession().createQuery(ConsultaSQL);
    return query.list();
}
```

Para la inserción de los datos he creado estos dos métodos, uno para insertar un empleado y otro para borrarlo.

El objeto **Session** se obtiene a partir de un objeto **SessionFactory**, invocando el método **openSession**. Un objeto **SessionFactory** representa una configuración particular de un conjunto de metadatos de mapping objeto/relaciona. Cuando se crea el objeto **Session**, se le asigna la conexión de la base de datos que va a utilizar. Una vez obtenido el objeto **Session**, se crea una nueva unidad de trabajo (**Transaction**) utilizando el método **beginTransaction**. Dentro del contexto de la transacción creada, se pueden invocar los métodos de gestión de persistencia proporcionados por el objeto **Session**, para recuperar, añadir, eliminar o modificar el estado de instancias de clases persistentes.

Se utiliza el objeto creado con **Session** lo usamos para, salvar actualizar, borrar, cerrar etc... Si las operaciones de persistencia no han producido ninguna excepción, se invoca el método **commit** de la unidad de trabajo para confirmar los cambios realizados.

```
public void instertarEmpleado(Emp empleado) {
    //abrimos la conexion con hibernate
    SessionFactory session = NewHibernateUtil.getSessionFactory();
    Session session = session.openSession();
    //para que se modifique en la base de datos
    Transaction tx = session.beginTransaction();
    session.save(empleado); //guardamos la inserccion
    //no se por que de repente me daba fallo, si no lo comento ejecuta bien
    //pero claro como no he llamado a la tansaccion no hay cambios en la base de datos
    tx.commit();
    session.close(); //se cierra la sesión
}

public void borrarEmpleado(Emp empleado) {
    SessionFactory session = NewHibernateUtil.getSessionFactory();
    Session session = session.openSession();
    Transaction tx = session.beginTransaction();
    session.delete(empleado); //borramos la inserción
    tx.commit();
    session.close();
}
```

Para obtener el listado de las tablas EMP y DEPT, he realizado la siguiente consulta en la clase principal.

He llamado a la interfaz **createQuery** por que lo he realizado en, sino hubiera sido **createSQLQuery**. Para retornar la lista de la consulta, he creado una lista de objetos, y un bucle for para listar las tablas.

Importantísimo cerra la sesión con Hibernate.

```
System.out.println("-----"
    + "\nLista de tablas EMP y DEPT que visualice empno,ename,dname,sal y loc con HQL\n"
    + "-----");

//Llamamos a la interfaz SQLQuery o createQuery dependiendo de si queremos hacer la consulta en HQL o SQL
//Creo que esta vez está bien la consulta
Query q1 = session.createQuery("FROM Emp as empleado INNER JOIN empleado.dept as departamento");
//Esta consulta retornarán una lista de objetos arrays (Object[]) con valores
//escalares para cada columna en la tabla EMP
List<Object[]> lista = q1.list();//lista de objetos
for (Object[] objeto : lista) {//bucle para listar las tablas
    Emp empleado = (Emp) objeto[0];
    Dept departamento = (Dept) objeto[1];
    System.out.println("Nombre: " + empleado.getEname() + ", número: " + empleado.getEmpno()
        + ", salario: " + empleado.getSal() + ", departamento: " + departamento.getDname()
        + ", localización departamento: " + departamento.getLoc());
}

session.close();//se cierra la sesion

//IMPORTANTE CERRAR LA SESION
NewHibernateUtil.getSessionFactory().close();
}
```

```
-----
Lista de tablas EMP y DEPT que visualice empno,ename,dname,sal y loc
-----
Hibernate: select emp0_EMPNO as EMPNO1_0_, dept1_DEPTNO as DEPTNO1_0_1_, emp0_DEPTNO as DEPTNO1_0_1_, emp0_ENAME as ENAMES1_0_, emp0_JOB as JOB4_1_0_, emp0_MGR as MGR5_1_0_, emp0_HIREDATE as HIREDATE6_1_0_, emp0_
Nombre: SMITH, número: 7369, salario: 800.00 , departamento: RESEARCH, localización departamento: DALLAS
Nombre: ALLEN, número: 7499, salario: 1600.00 , departamento: SALES, localización departamento: CHICAGO
Nombre: WARD, número: 7521, salario: 1250.00 , departamento: SALES, localización departamento: CHICAGO
Nombre: JONES, número: 7566, salario: 2975.00 , departamento: RESEARCH, localización departamento: DALLAS
Nombre: MARTIN, número: 7654, salario: 1250.00 , departamento: SALES, localización departamento: CHICAGO
Nombre: BLAKE, número: 7698, salario: 2850.00 , departamento: SALES, localización departamento: CHICAGO
Nombre: CLARK, número: 7782, salario: 2450.00 , departamento: ACCOUNTING, localización departamento: NEW YORK
Nombre: SCOTT, número: 7788, salario: 3000.00 , departamento: RESEARCH, localización departamento: DALLAS
Nombre: KING, número: 7835, salario: 5000.00 , departamento: ACCOUNTING, localización departamento: NEW YORK
Nombre: TURNER, número: 7844, salario: 1600.00 , departamento: SALES, localización departamento: CHICAGO
Nombre: ADAMS, número: 7876, salario: 1100.00 , departamento: RESEARCH, localización departamento: DALLAS
dic 11, 2011 8:58:57 AM org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionFactoryImpl stop
Nombre: JAMES, número: 7900, salario: 950.00 , departamento: SALES, localización departamento: CHICAGO
Nombre: FORD, número: 7902, salario: 3000.00 , departamento: RESEARCH, localización departamento: DALLAS
Nombre: MILLER, número: 7934, salario: 1300.00 , departamento: ACCOUNTING, localización departamento: NEW YORK
INFO: HH000000: Cleaning up connection pool [jdbc:mysql://localhost:3306/saare?useDate=TimeBehavior=convertToNull]
BUILD SUCCESSFUL (total time: 0 seconds)
```