APUNTES PARCIAL ACCESO A DATOS

Tema 2

Crear un fichero DAT de acceso aleatorio

1. Hacer la clase empleado con sus getters y setters

```
public class Empleado {

    private static final int NUMERO_EMPLEADOS_FICHERO = 5;
    private static final int MIN_EMPLEADOS_FICHERO = 1;
    private static final int MAX_EMPLEADOS_FICHERO = 99;

    private int codigo;
    private String nombre;
    private String direccion;
    private float salario;
    private float comision;

    private Empleado() {
    }

    private Empleado(int codigo, String nombre, String direccion,
            float salario, float comision) {
        this.codigo = codigo;
        this.nombre = nombre;
        this.direccion = direccion;
        this.salario = salario;
        this.comision = comision;
    }

    public int getCodigo() {
        return codigo;
    }
```

1. Crear un fichero RandomAccessFile -> le pasas el nombre de fichero y los permisos
2. Fichero.seek() le pasas la longitud del fichero que ahora está vacío
3. Creas empleados aleatoriamente y devuelves una lista
4. Creas un String con los datos del empleado ¡OJO hacer new String para a final pasarle StandardCharsets.UTF_8

5. Cuando tengas el String con los datos haces fichero.writeBytes(datosEmpleados) para escribir en él

```
public RandomAccessFile crearFicheroEmpleados(String nombreFichero)
        throws FileNotFoundException, IOException {
    RandomAccessFile ficheroEmpleados;
    try (RandomAccessFile fichero = new RandomAccessFile(nombreFichero, "rw")) {
        fichero.seek(fichero.length());
        List<Empleado> empleados = crearEmpleadosAleatorios(NUMERO_EMPLEADOS_FICHERO);
        int numeroEmpleado = 1;
        for (Empleado empleado : empleados) {
            String datosEmpleado = new String(String.format("Empleado %d:\n Codigo empleado: %d \n Nombre:
                    + "Dirección: %s \n Salario: %.2f \n Comisión: %.2f \n",
                    numeroEmpleado++, empleado.getCodigo(), empleado.getNombre(),
                    empleado.getDireccion(), empleado.getSalario(),
                    empleado.getComision()).getBytes(),
                    StandardCharsets.UTF_8);
            fichero.writeBytes(datosEmpleado);
        }
        ficheroEmpleados = fichero;
    }
    return ficheroEmpleados;
}
```

```
public List<Empleado> crearEmpleadosAleatorios(int numeroEmpleados) {
    if (numeroEmpleados > MAX_EMPLEADOS_FICHERO) {
        throw new IllegalArgumentException("El máximo de empleados por fichero es 99");
    } else if (numeroEmpleados < MIN_EMPLEADOS_FICHERO) {
        throw new IllegalArgumentException("El número mínimo de empleados por fichero es 1");
    }
    List<Empleado> empleados = new ArrayList();
    String[] nombres = {"María", "Francisco", "Alejandro", "Elena", "Luismi", "Miriam"};
    String[] direcciones = {"Calle Juan Carlos I", "Calle Santa Sabina",
        "Plaza de España", "Calle falsa 123", "Fuencarral"};
    float[] comision = {10, 20, 25, 30, 33, 40, 55};
    float[] salario = {1000, 1100, 1200, 1300, 1400, 1500, 2000, 2200, 3000};
    for (int i = 1; i <= numeroEmpleados; i++) {
        Empleado empleado = new Empleado(i, nombres[getNumeroAleatorio(nombres.length) - 1],
                direcciones[getNumeroAleatorio(direcciones.length) - 1],
                salario[getNumeroAleatorio(salario.length) - 1],
                comision[getNumeroAleatorio(comision.length) - 1]);
        empleados.add(empleado);
    }
    return empleados;
}
```

Leer un fichero de acceso Aleatorio

1. File file = new File(nombreFichero) para abrir el fichero
2. Haces un try con recursos donde instancias un fichero RandomAccessFile y le pones permisos de lectura “r”
3. Te pones en la posición 0

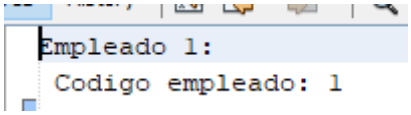
```
File file = new File(nombreFichero);
try (RandomAccessFile archivo = new RandomAccessFile(file, "r")) {
    int posicion = 0;
```

4. Haces un while infinito o un for(;;) y para salir lo consigues con (poniéndolo al final del bucle

```
            if (archivo.getFilePointer() == file.length()) {
                break;
            }

        }
```

5. Leer el fichero
    a. Archivo.seek(posición)
    b. new String(archivo.readLine().getBytes, StandardCharserts.UTF_8)
    c. Como nos interesa la línea de abajo comprobamos si la línea que hemos leído contienen "Código" y si es así avanzamos la posición con string.length() +1 para leer la línea de abajo que es la que nos interesa y como sabemos que tendrá el texto Codigo empleado y queremos el 1 sustituimos para coger el texto




```
archivo.seek(posicion);
String codigo = new String(archivo.readLine().getBytes(), StandardCharsets.UTF_8);
if (codigo.contains("Empleado ")) {
    posicion += codigo.length() + 1;
}
codigo = archivo.readLine();
if (codigo.contains("Codigo")) {
    posicion += codigo.length() + 1;
    codigo = codigo.replace("Codigo empleado: ", "");
    this.codigo = Integer.parseInt(codigo.trim());
}
```

6. Para seguir leyendo nos vamos a la posición deseada y repetimos el proceso

```
        }
        archivo.seek(posicion);
        String nombre = archivo.readLine();
        if (nombre.contains("Nombre: ")) {
            posicion += nombre.length() + 1;
            this.nombre = nombre.replace("Nombre: ", "").trim();
        }
        archivo.seek(posicion);
```

Crea un fichero XML a partir del fichero DAT

Al leer el fichero creas una lista de empleados de donde crearemos el XML

```
DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
DocumentBuilder builder = factory.newDocumentBuilder();
DOMImplementation implementation = builder.getDOMImplementation();
Document document = implementation.createDocument(null, "EMPLEADOS.XML", null);
Element raiz = document.createElement("Empleados");
document.getDocumentElement().appendChild(raiz);
```

Vas rellenando los datos a partir de la lista

```
for (Empleado empleado : empleados) {
    Element empleadoXML = document.createElement("Empleado");
    raiz.appendChild(empleadoXML);
    Element codigoEmpleado = document.createElement("Codigo");
    codigoEmpleado.setTextContent(String.valueOf(empleado.getCodigo()));
    empleadoXML.appendChild(codigoEmpleado);
    Element nombre = document.createElement("Nombre");
    nombre.setTextContent(empleado.getNombre());
    empleadoXML.appendChild(nombre);
    Element direccion = document.createElement("Dirección");
    direccion.setTextContent(empleado.getDireccion());
    empleadoXML.appendChild(direccion);
    Element salario = document.createElement("Salario");
    salario.setTextContent(String.valueOf(empleado.getSalario()));
    empleadoXML.appendChild(salario);
    Element comision = document.createElement("Comisión");
    comision.setTextContent(String.valueOf(empleado.getComision()));
    empleadoXML.appendChild(comision);

}
```

```
    TransformerFactory transformerFactory = TransformerFactory.newInstance();
    Transformer transformer = transformerFactory.newTransformer();
    DOMSource source = new DOMSource(document);
    StreamResult result = new StreamResult(new File("EMPLEADO.xml"));
    transformer.transform(source, result);
```

Visualizar las etiquetas del fichero XML usando SAX

1. Extiende de DefaultHandler

```
 */
public class SAX extends DefaultHandler{

    ...
```

2. Creamos una clase interna estática para almacenar la información del XML

```java
/**
 * Clase para almacenar los datos el XML de libros
 */
private static class Libro {

    private int year;
    private String titulo;
    private String apellido;
    private String nombre;
    private String editorial;
    private BigDecimal precio;

    /**
     * Sobreescribimos toString() para dar un formato concreto a cada libro
     * que aparece en el XML
     *
     * @return libro del XML formateado
     */
    @Override
    public String toString() {
        return String.format("Año: %d \nTítulo: %s \n"
                + "Autor: %s, %s \nEditorial: %s\n"
                + "Precio: %s€",
                year, titulo, apellido, nombre, editorial, precio);

    }
}
```

3. Booleanos para saber si tiene ese atributo o no, creamos los atributos libro y lista de libros para almacenar los datos y StringBuilder

```java
/**
 * Booleanos para comprobar si tienen esos atributos
 */
private boolean tieneTitulo;
private boolean tieneApellido;
private boolean tieneNombre;
private boolean tieneEditorial;
private boolean tienePrecio;

private List<Libro> libros;
private final Libro libro = new Libro();
private StringBuilder datos;

public List<Libro> getLibros() {
    return this.libros;
}
```

4. Sobreescribimos el método startElement y según encuentre los atributos ponemos el booleano a true y al final iniciamos el StringBuilder para que se reinicie

```java
@Override
public void startElement(String uri, String localName, String qName, Attributes attributes) {
    if (qName.equalsIgnoreCase("libro")) {
        String year = attributes.getValue("año").trim();
        this.libro.year = Integer.parseInt(year);
        if (this.libros == null) {
            this.libros = new ArrayList<>();
        }
    } else if (qName.equalsIgnoreCase("titulo")) {
        this.tieneTitulo = true;
    } else if (qName.equalsIgnoreCase("apellido")) {
        this.tieneApellido = true;
    } else if (qName.equalsIgnoreCase("nombre")) {
        this.tieneNombre = true;
    } else if (qName.equalsIgnoreCase("editorial")) {
        this.tieneEditorial = true;
    } else if (qName.equalsIgnoreCase("precio")) {
        this.tienePrecio = true;
    }
    this.datos = new StringBuilder();
}
```

5. Sobreescribimos el método endElement y quitamos los espacios y saltos de línea. Irá entrando en los if si es ese aributo el que ha encontrado hasta crear el libro que podamos añadir a la lista

```
@Override
public void endElement(String uri, String localName, String qName) {

    //Quitamos los espacios del texto y los saltos de línea
    String textoFormateado = datos.toString().trim().replaceAll("[\t\n]",
            "");
    /**
     * Comprobamos si tiene un atributo y lo vamos rellenando. Una vez
     * asignado el atributo se pone a false para que no vuelva a entrar en
     * el if
     */
    if (this.tieneTitulo) {
        this.libro.titulo = textoFormateado;
        this.tieneTitulo = false;
    } else if (this.tieneApellido) {
        this.libro.apellido = (textoFormateado);
        this.tieneApellido = false;
    } else if (this.tieneNombre) {
        this.libro.nombre = (textoFormateado);
        this.tieneNombre = false;
    } else if (this.tieneEditorial) {
        this.libro.editorial = (textoFormateado);
        this.tieneEditorial = false;
    } else if (this.tienePrecio) {
        this.libro.precio = (new BigDecimal(textoFormateado));
        this.tienePrecio = false;
    }
    if (qName.equalsIgnoreCase("libro")) {
        this.libros.add(libro);
    }
}
```

6. Sobreescribimos el método characters

```
}

@Override
public void characters(char[] ch, int start, int length) {
    datos.append(new String(ch, start, length));
}
```

7. Abrimos el XML y creamos la lista de libros para mostrarlos por pantalla con toString()

```
public void visualizarEtiquetasXMLconSAX()
        throws SAXException, IOException {
    XMLReader xmlReader = XMLReaderFactory.createXMLReader();
    SAX sax = new SAX();
    xmlReader.setContentHandler(sax);
    InputSource inputSource = new InputSource("libros.xml");
    xmlReader.parse(inputSource);
    List<SAX.Libro> libros = sax.getLibros();
    for (int i = 0; i < libros.size(); i++) {
        System.out.printf("Libro %d: \n", i + 1);
        System.out.println(libros.get(i));
        System.out.println();
    }

}
```

Visualizar las etiquetas del fichero XML usando DOM

```
private void visualizarEtiquetasXMLconDOM() {

    // Creo una instancia de DocumentBuilderFactory
    DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
    try {
        DocumentBuilder builder = factory.newDocumentBuilder();
        // Obtengo el documento, a partir del XML
        Document documento = builder.parse(new File("libros.xml"));
        documento.getDocumentElement().normalize();
        NodeList libros = documento.getElementsByTagName("libro");
        // Recorro las etiquetas
        for (int i = 0; i < libros.getLength(); i++) {
            System.out.printf("Libro %d: \n", i + 1);
            // Cojo el nodo
            Node nodo = libros.item(i);
            //Obtengo la información de ese nodo
            getInformacionNodo(nodo);
        }

    } catch (ParserConfigurationException | SAXException | IOException e) {
        e.printStackTrace();
    }
}
```

Vas recorriendo los nodos y sacando las etiquetas

```java
private void getInformacionNodo(Node nodo) {
    //Cogemos el atributo "año" del libro
    NamedNodeMap attributes = nodo.getAttributes();
    if (attributes.getLength() > 0) {
        Node item = attributes.item(0);
        System.out.printf("%s: %s\n", item.getNodeName(), item.getTextContent());

    }
    if (nodo.getNodeType() == Node.ELEMENT_NODE) {
        Element e = (Element) nodo;
        // Obtengo sus hijos y los recorro
        NodeList hijos = e.getChildNodes();
        for (int j = 0; j < hijos.getLength(); j++) {
            Node hijo = hijos.item(j);
            if (hijo.getNodeType() == Node.ELEMENT_NODE) {
                String text = hijo.getTextContent().trim();
                //Como el autor está separado por nombre y apellido sustituimos los saltos
                //de linea y espacios para dejarlo en una línea
                if (text.contains("\n")) {
                    text = text.replace("\n", "")
                            .replaceAll("\\s+", ",");
                }
                // Muestro el contenido
                System.out.printf("%s: %s\n", hijo.getNodeName(), text);
            }

        }
        System.out.println();
    }
}
```

Tema 3. Manejo de conectores

Conector Oracle XE

Jdbc:oracle:thin@localhost:1521:XE

Clase que conecta con base de datos con patrón singleton

```
public class BaseDatos {

	private static BaseDatos miInstancia = null;
	private static boolean permitirInstancianueva;
	private String cadenaConexion = "jdbc:oracle:thin:@localhost:1521:XE";
	private String usuario = "ad03";
	private String password = "ad03";
	private Connection conn;
	private Statement stmt;

	BaseDatos() throws Exception {
		if (!permitirInstancianueva)
			throw new Exception("No se puede crear la instancia, usa getInstance");
	}

	public static BaseDatos getInstance() {
		if (miInstancia == null) {
			permitirInstancianueva = true;
			try {
				miInstancia = new BaseDatos();
			} catch (Exception e) {
				e.printStackTrace();
			}
			permitirInstancianueva = false;
		}
		return miInstancia;
	}
```

Consulta a una base de datos

```
public ResultSet consultaBD(String consulta) throws SQLException {
	DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());

	conn = DriverManager.getConnection(cadenaConexion, usuario, password);

	stmt = conn.createStatement();
	ResultSet rset = stmt.executeQuery(consulta);

	return rset;
}
```

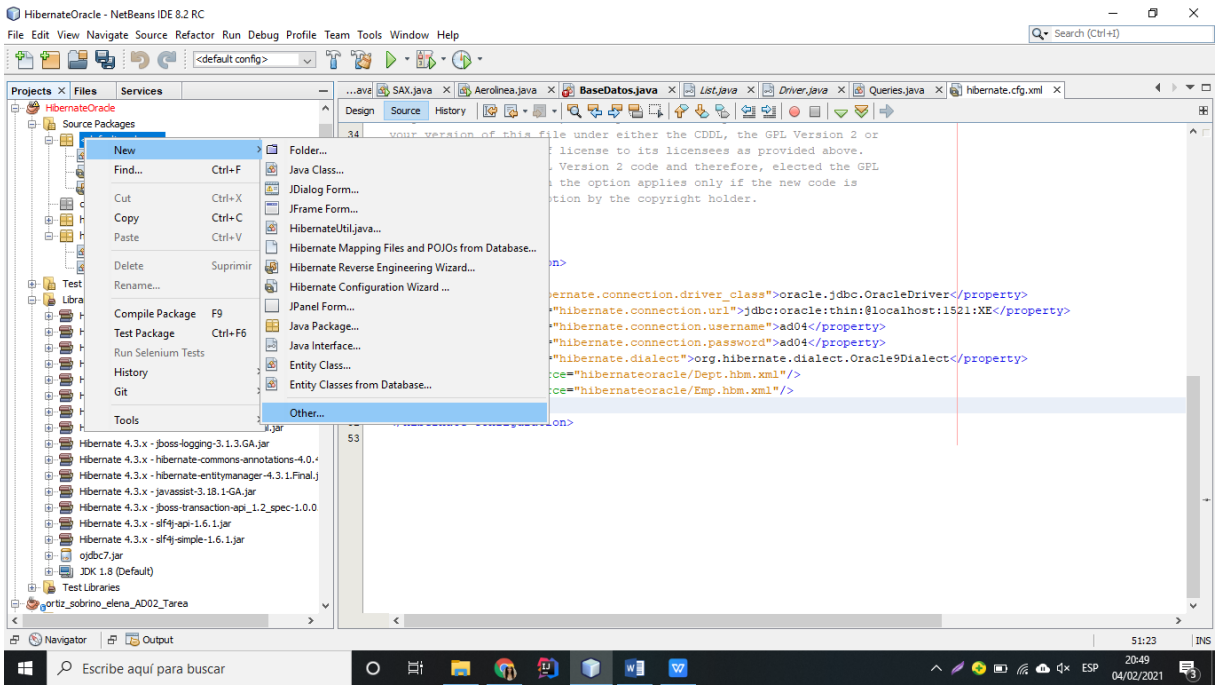
Tema 4.Mapeo Objeto Relacional

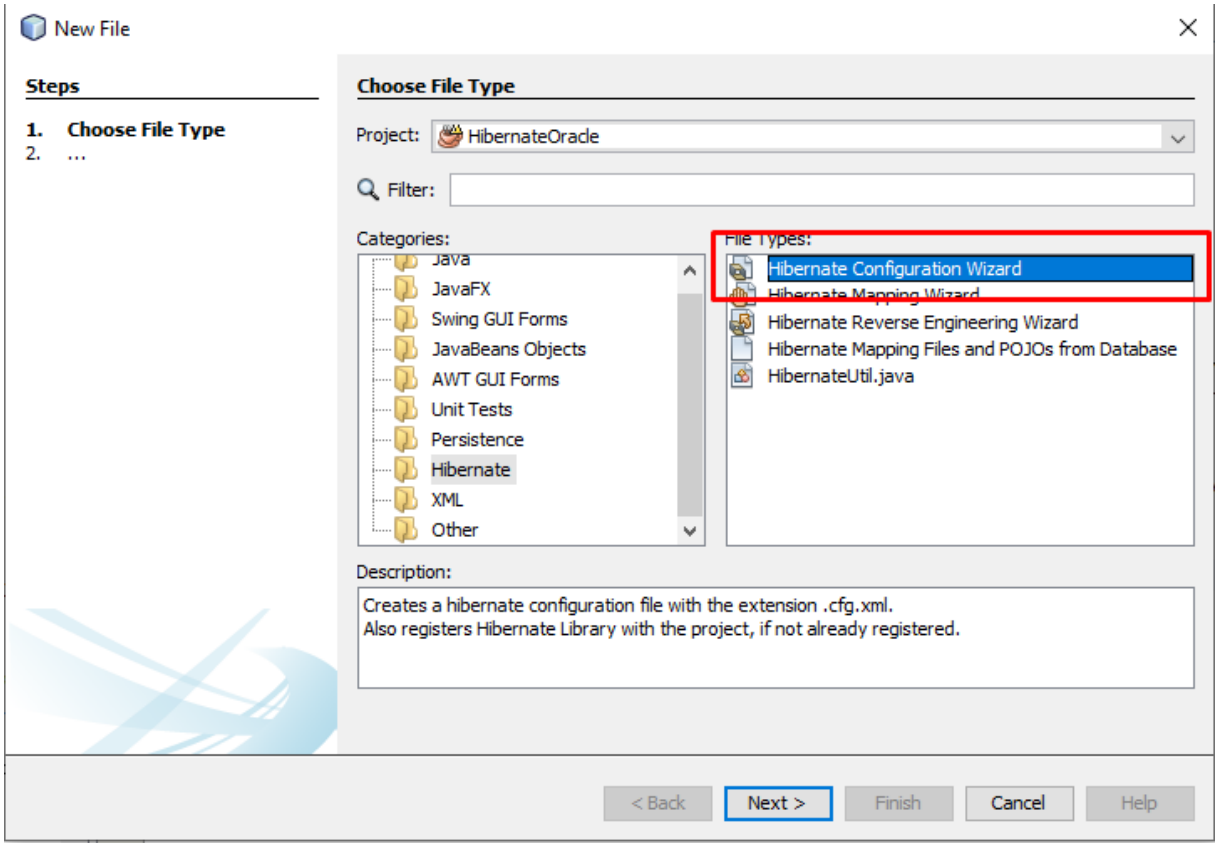
Mapeo con hibernate

Descargar hibernate

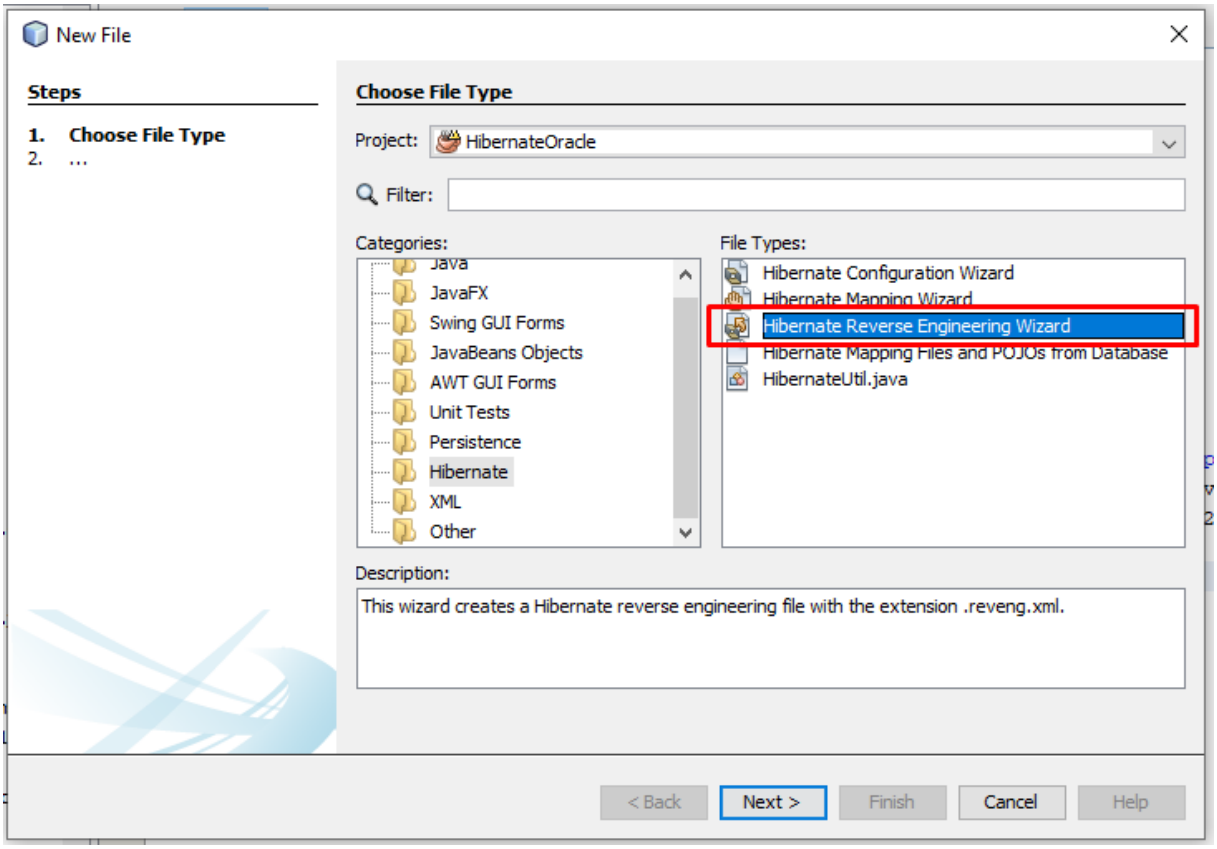
1. Descargas Hibernate JBoss
2. Copiar todos los ficheros jar que vienen dentro de lib

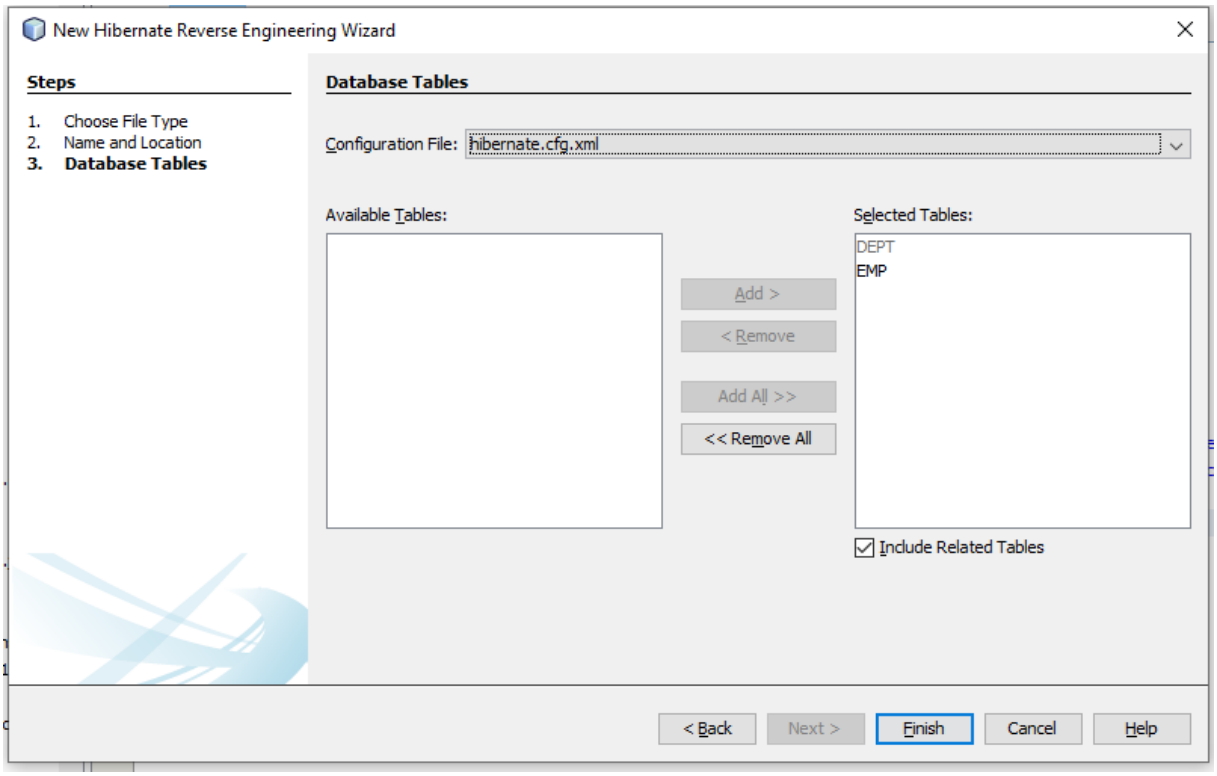
Creación del fichero hibernate.cgf.xml



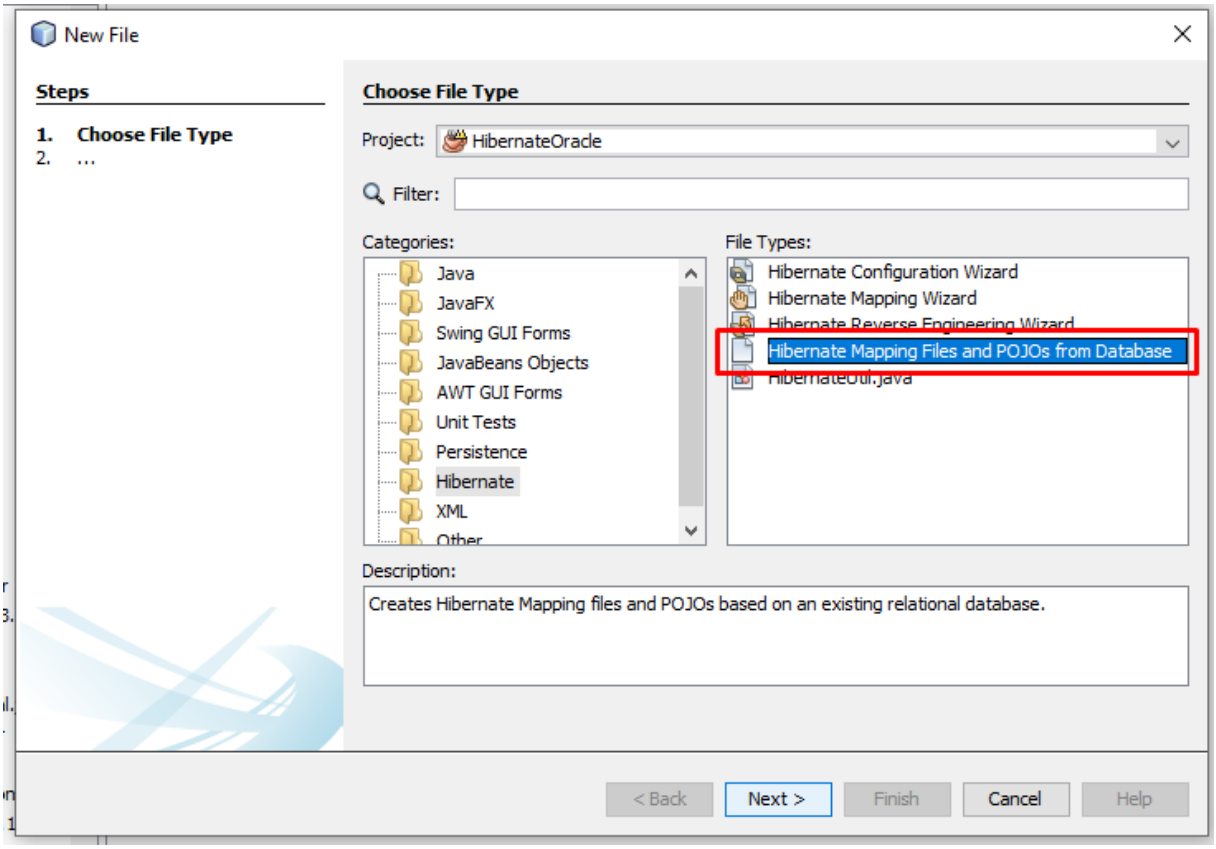


Crear el fichero hibernate.reveng.xml

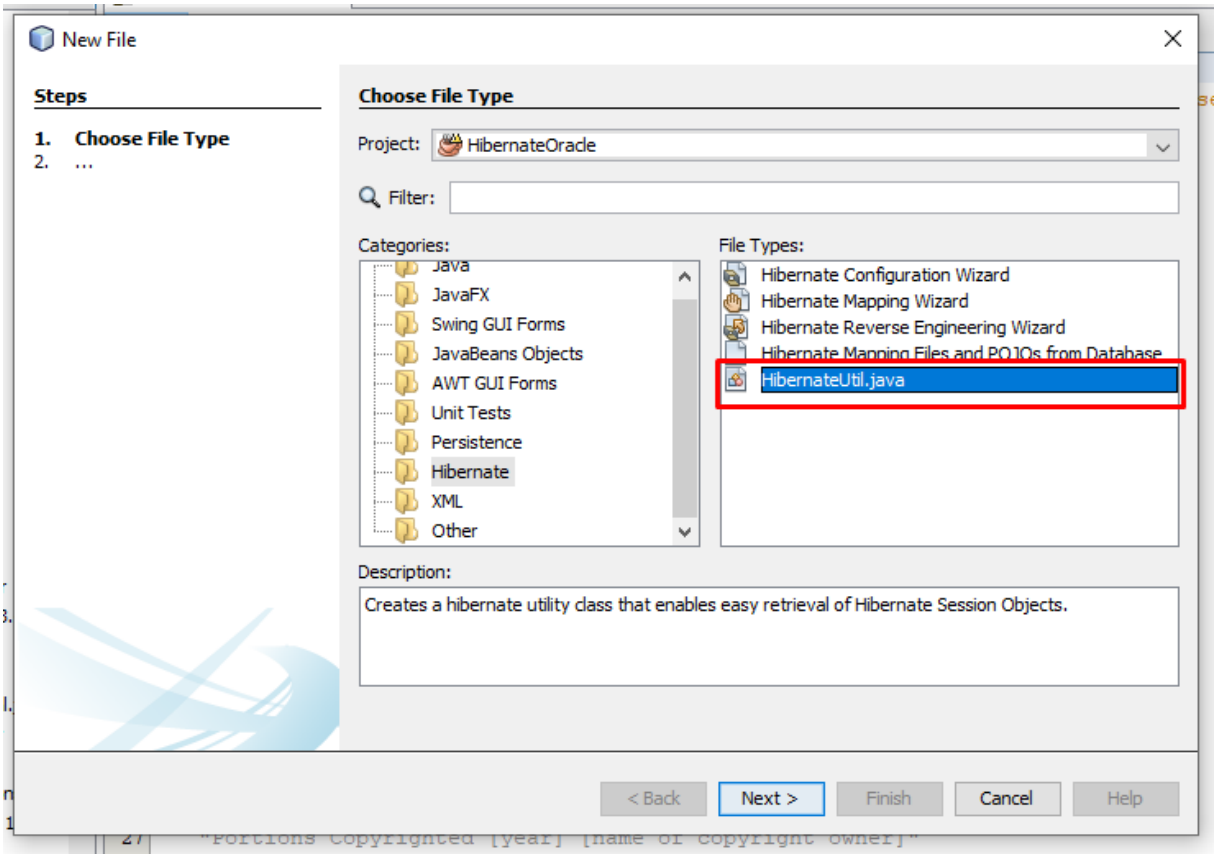




Crear las clases



HibernateUtil



Otras utilities

Consulta genérica

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

Controlar sesiones

```
private Session session;
SessionFactory sessionFactory;

public Queries() {
}

public Session getSession() {

    if (session == null) {
        SessionFactory sessionFactory = HibernateUtil.getSessionFactory();
        session = sessionFactory.openSession();
    }
    return session;
}
```

Ejemplo post

```
public void altaEmpleado(Emp empleado) {
    SessionFactory sesion = HibernateUtil.getSessionFactory();
    Session session = sesion.openSession();
    Transaction tx = session.beginTransaction();
    session.save(empleado);
    tx.commit();
    session.close();
}
```

Ejemplo delete

```
public void bajaEmpleado(Emp empleado) {
    SessionFactory sesion = HibernateUtil.getSessionFactory();
    Session session = sesion.openSession();
    Transaction tx = session.beginTransaction();
    session.delete(empleado);
    tx.commit();
    session.close();
}
```

HQL

//TODO: mirarse mejor las consultas HQL (¿) ver después del examen

Select

```
Queries queries = new Queries();
List<Dept> deps = queries.consulta("FROM Dept");
Dept departament = deps.get(0);
```

Select con join

Coges el objeto 0 y el 1 porque el Object[] está formado por object[0] que es el empleado y el object[1] que es el departamento

```
List<Object[]> lista = queries.consulta("FROM Emp as empleado INNER JOIN empleado.dept as departamento");
for (Object[] objecto : lista) {
    Emp empleado = (Emp) objecto[0];
    Dept departamento = (Dept) objecto[1];
    System.out.println("Nombre: " + empleado.getEname() + ", número: " + empleado.getEmpno()
            + ", salario: " + empleado.getSal() + " , departamento: " + departamento.getDname()
            + ", localización departamento: " + departamento.getLoc());
}
```

Ejemplo fichero hibernate.cgf.xml

```
<hibernate-configuration>
    <session-factory>
    <property name="hibernate.connection.driver_class">oracle.jdbc.OracleDriver</property>
        <property name="hibernate.connection.url">jdbc:oracle:thin:@localhost:1521:XE</property>
        <property name="hibernate.connection.username">ad04</property>
        <property name="hibernate.connection.password">ad04</property>
        <property name="hibernate.dialect">org.hibernate.dialect.Oracle9Dialect</property>
        <mapping resource="hibernateoracle/Dept.hbm.xml"/>
        <mapping resource="hibernateoracle/Emp.hbm.xml"/>
    </session-factory>
</hibernate-configuration>
```

Utilities

Crear usuarios

Abrir Run SQL Command Line

1. Connectarte como admin y poner tu contraseña

   connect sys/root as sysdba;

2. Crear un usuario

   create user ad03 identified by ad03

default tablespace users

quota unlimited on users;

3. Conceder permisos al usuario creado

grant connect, resource to ad03;