MODULO M06:
ACCESO A DATOS

Profesor: Jorge Juan Delgado

ILERNA Online

UNIDAD FORMATIVA 1

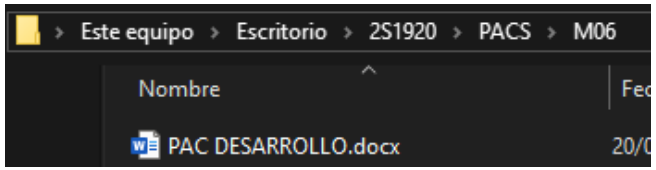
PERSISTENCIA EN FICHEROS

¿QUÉ SON LOS FICHEROS?

- Fichero (File en inglés) es tradicionalmente un lugar donde se almacenan fichas, carpetas y en general documentos
- Este concepto se usa en informática al lugar en el que se almacena información
- La naturaleza de esta información es muy diversa
- En la teoría, no hay límites para la cantidad de información que se puede almacenar
- Se pueden categorizar

> Ficheros

- Están formados por bits
- Se guardan como registros
- Identificados por ruta y nombre
- No son volátiles
- Tienen una extensión



- TIPOS DE FICHEROS

  - FICHERO ESTÁNDAR

    - Válido para todo tipo de datos (caracteres, imágenes, vídeo, etc)

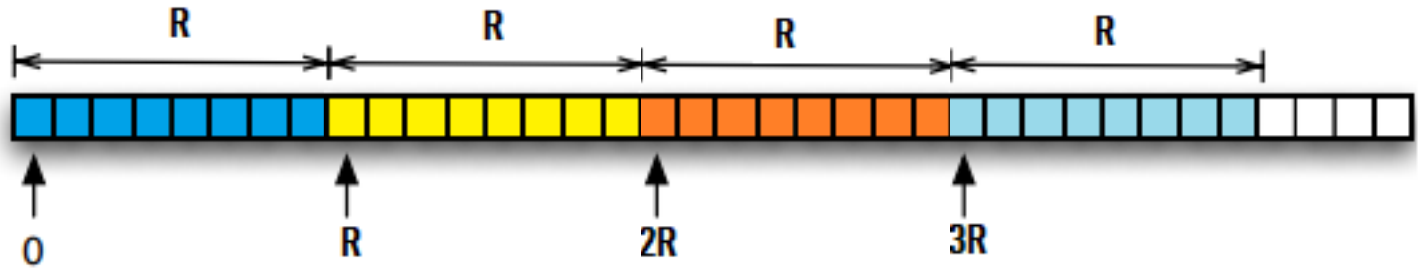
  - DIRECTORIO O CARPETA

    - Fichero que almacena más ficheros en su interior (jerarquía)

  - FICHEROS ESPECIALES

    - Del SO y se usan para controlar dispositivos o periféricos

> Registros



¿Qué son los registros?
Se puede llamar registro a cada bloque de longitud fija por el que está formado un fichero.

> Tipos de ficheros

Ficheros

Caracteres (o texto)

Creados exclusivamente con caracteres

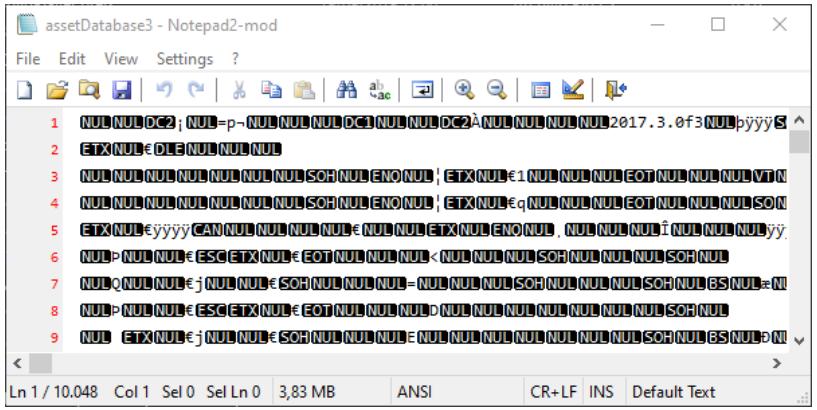
Cualquiera puede leerlo

Binarios (o bytes)

Su contenido está codificado

Formados por bytes, contienen información generalmente de tipo multimedia

Existen **2 tipos** de ficheros.



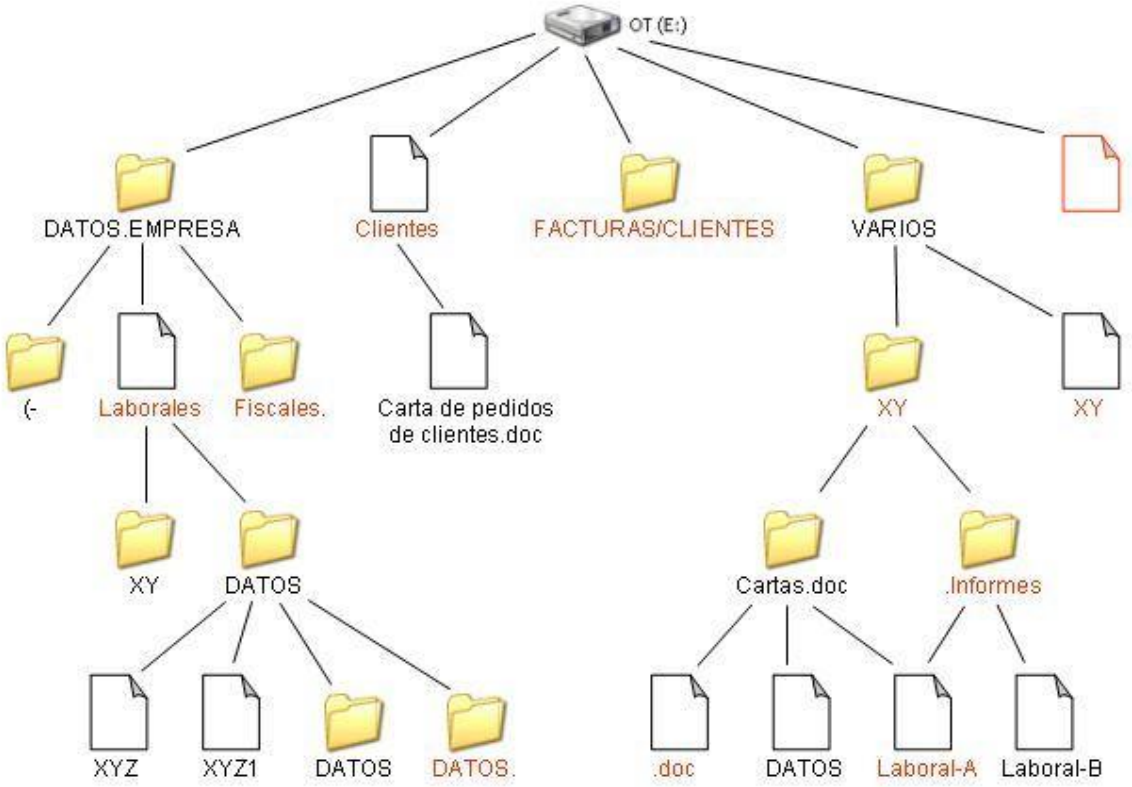
¿Cómo trabajamos en Java con ficheros?

- ACCESO A FICHEROS

- Todos los ficheros están en una posición dentro de la jerarquía de directorios de la unidad de almacenamiento de un sistema informático

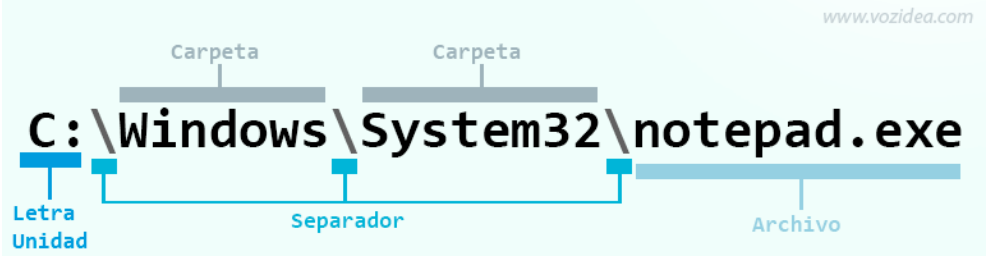
- Para tener acceso a un fichero determinado, se utiliza una **ruta**

En una ruta absoluta siempre empezamos por la unidad en la que estamos

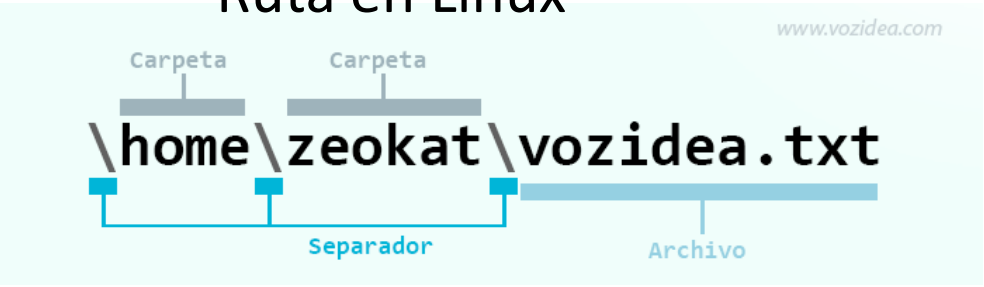


- La ruta indica la ubicación de ese fichero en nuestro sistema
- Formada por diferentes niveles jerárquicos (carpetas) separado por un símbolo barra /,
  - Windows, se utiliza la contrabarra \
  - Unix el separador es /
- Eclipse admite tanto / como \ cuando definimos la ruta.

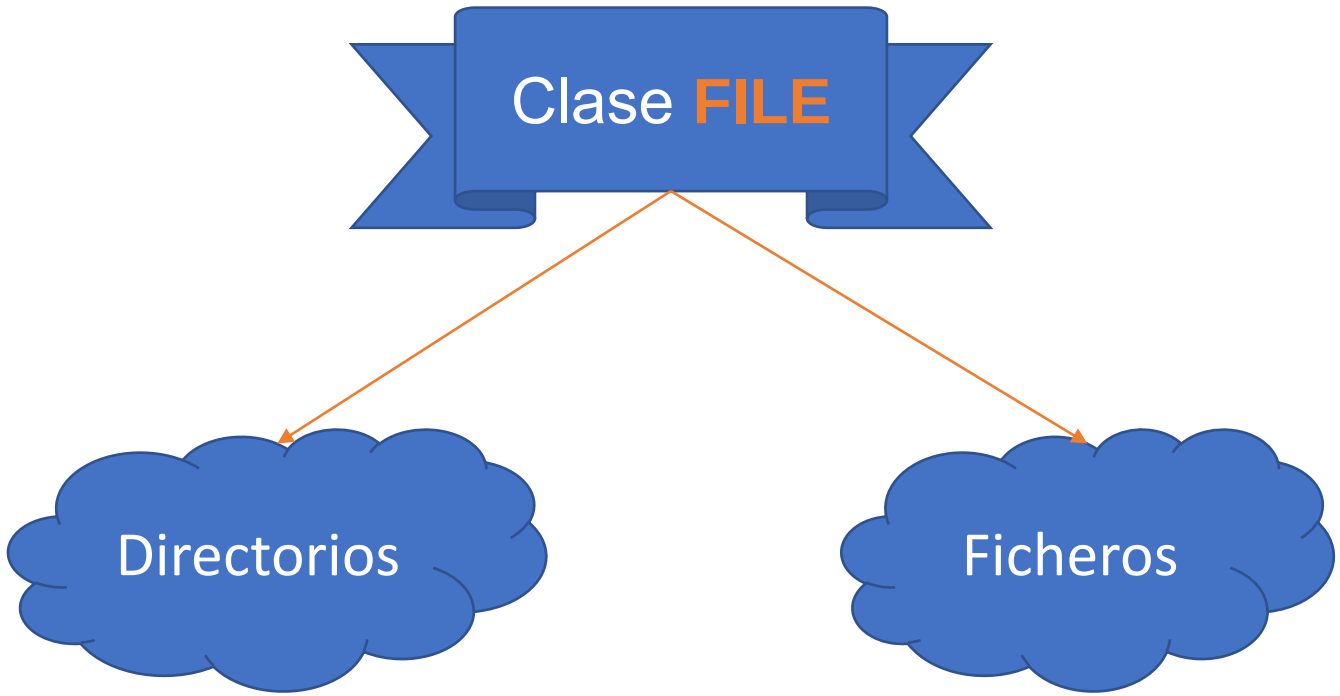
Ruta en Windows



Ruta en Linux



En la imagen está mal, en Linux se usa / no \. Aunque ya hay versiones que admiten \.



La clase File se utiliza para trabajar con ficheros y directorios.
Un objeto de esta clase puede representar a cualquiera de los dos elementos

¿Cómo se puede crear un objeto File?

> Creación de objeto File

Constructores

File (String nombre)

Solo buscara en el directorio en el que se encuentre.

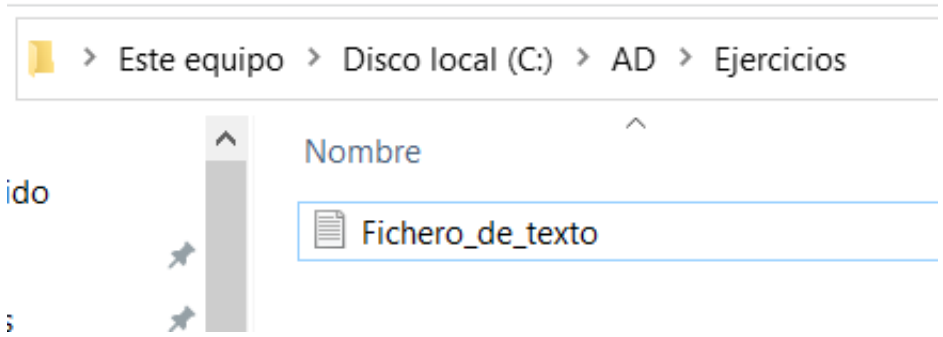
File (File ruta, String nombre)

File (String ruta, String nombre)

Indicamos la ruta del fichero y el nombre del fichero. La mejor.

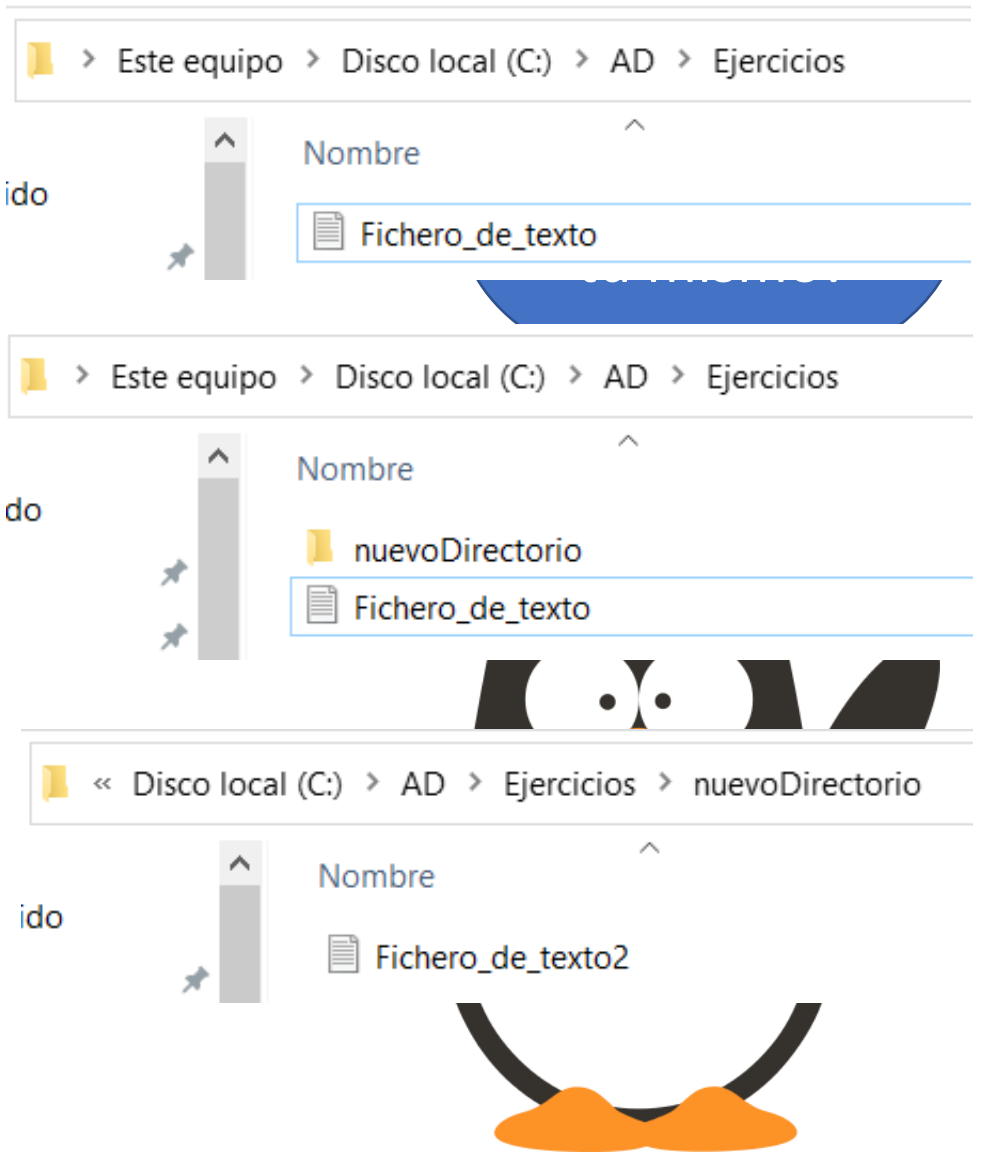
EJERCICIOS PARA PRACTICAR

1. Crea un programa en Java que cree un fichero de texto llamado miFichero.txt en la ruta C:\AD\Ejercicios



EJERCICIOS PARA PRACTICAR EN CASA

1. Añadir al ejercicio anterior un menú de opciones al programa anterior y que el usuario pueda elegir qué hacer:
    1. Crear un directorio llamado nuevoDirectorio en la ruta C:\AD\Ejercicios, dentro de él se creará un nuevo fichero de texto llamado miFichero2.txt
    2. Eliminar el fichero miFichero.txt
    3. Eliminar el directorio nuevoDirectorio
    4. Salir
2. Las opciones se podrán seguir eligiendo hasta que el usuario elija la opción de salir



```java
do {
System.out.println("__________     CREAR UN FICHERO DE TEXTO    __________");
System.out.println("                    MENÚ DE OPCIONES                   ");
System.out.println(" 1. Crear directorio nuevoDirectorio ");
System.out.println(" 2. Crear fichero miFichero2.txt en nuevoDirectorio ");
System.out.println(" 3. Eliminar fichero miFichero.txt ");
System.out.println(" 4. Eliminar directorio nuevoDirectorio ");
System.out.println(" 0. Salir ");
System.out.print("   - Elige una opción: ");
opcion = sc.nextInt();

switch(opcion) {
case 1: //Crear el directorio nuevoDirectorio
        if(miDirectorio.mkdir())
            System.out.println("Directorio creado correctamente...");
        else
            System.out.println("ERROR: El directorio no ha podido ser creado...");
            break;

case 2: //Crear fichero miFichero2.txt en nuevoDirectorio
        //Bloque try-catch para capturar excepción de E/S
        try {
            if(miFichero2.createNewFile())
                System.out.println("Fichero creado correctamente...");
            else
                System.out.println("ERROR: El fichero no ha podido ser creado...")

        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        break;
case 3: //Eliminar fichero miFichero2.txt
        if(miFichero2.delete())
            System.out.println("Fichero borrado correctamente...");
        else
            System.out.println("ERROR: El fichero no ha podido ser eliminado...");
        break;
case 4://Eliminar directorio nuevoDirectorio
        //Bloque try-catch para capturar excepción de E/S

        miDirectorio.deleteOnExit();
        System.out.println("Directorio borrado correctamente...");
        break;
case 0://Salir
        System.out.println("Gracias por usar este programa...");
        break;
default: //Opción incorrecta
    System.out.println("Opción incorrecta, inténtalo de nuevo...");

}
}while (opcion != 0);
System.out.println("____FIN DEL PROGRAMA___");
    }
}
```

Acceder a ficheros

> Acceso secuencial

El acceso es secuencial → Casete antiguo o VHS.

Data1 → Data2 → Data3

Los datos o registros se leen de forma ordenada.

Binario	FileInputStream	Caracteres	FileReader
	FileOutputStream		FileWriter

- Excepciones:
- FileReader -> FileNotFoundException
- FileWriter -> IOException
- Métodos de la clase FileReader

CLASE	FUNCIÓN
int read ()	Lee un carácter y lo devuleve
int read (char[] buf)	Lee hasta buf.length caracteres de datos de una matriz de caracteres (buf). Los caracteres leídos se almacenan en buf
int read (char[] buf, int	Lee hasta n caracteres de datos de la

- Métodos de la clase FileWriter

CLASE	FUNCIÓN
void write (int c)	Escribe un carácter
void write (char[] buf)	Escribe un array de caracteres
void write (char[] buf, int desplazamiento, int n)	Escribe n caracteres de datos en la matriz buf empezando por buf(desplazamiento)
void write (String cadena)	Escribe una cadena de caracteres
void append (char c)	Añade un carácter a un fichero Sin borrar el texto que había la útima vez que se cerró

> Ejemplo FileWrite/FileReader

```
private static void funcion2() throws IOException {
    File fichero1 = new File( pathname: "ficheros/filewriter.txt");
    FileWriter writer = new FileWriter(fichero1);
    writer.write( str: "Esto es un mensaje desde la función2");
    writer.flush();
    writer.close();
}
```

Con las dos primeras lineas (sin contar la de private) definimos el fichero sobre el que vamos a trabajar y abrimos el flujo que en este caso es de escritura para almacenar información en ese fichero. Definimos el fichero y luego el canal que llevará la información desde java al fichero.

Con writer.flush() estamos limpiando el canal que hemos utilizado. Si el usuario ha introducido un texto y enter, es posible que en la siguiente linea haya un salto de linea si no limpiamos el canal utilizado.

```
private static void funcion3() throws IOException {
    File fichero1 = new File( pathname: "ficheros/filewriter.txt");
    FileReader reader = new FileReader(fichero1);
    int valor = reader.read();
    while (valor != -1) {
        System.out.println("Char: "+(char)valor+"\t\tascii: "+valor);
        valor = reader.read();
    }
    reader.close();
}
```

```
Char: E         ascii: 69
Char: s         ascii: 115
Char: t         ascii: 116
Char: o         ascii: 111
Char:           ascii: 32
Char: e         ascii: 101
Char: s         ascii: 115
Char:           ascii: 32
                ...
```

EJERCICIOS PARA PRACTICAR

1. Crea un programa en Java que a partir de una cadena de caracteres permita almacenarlos en un fichero de texto
2. El programa añadirá un carácter * para indicar que ha llegado al final del fichero.

EJERCICIOS PARA PRACTICAR

2. Variante del ejercicio anterior: que almacene Strings a partir de un array

EJERCICIOS PARA PRACTICAR

3. Ahora el programa deberá leer el contenido del fichero de texto creado en el primer punto y lo mostrará por consola.

EJERCICIOS PARA PRACTICAR EN CASA

4. Crear un fichero en Java que almacene en un fichero de texto llamado Empleados.txt un listado con 5 empleados, indicando en cada línea el ID y el nombre de cada empleado.
5. A continuación el programa deberá leer el contenido del fichero y mostrarlo por pantalla

NOTA: los empleados y los Ids puedes introducirlos desde teclado o desde código

añadir después de IOException: , FileNotFoundException

```java
import java.io.File;
import java.io.FileReader;
import java.io.IOException;

public class Main3 {

    public static void main(String[] args) throws IOException {
        File fichero = new File ("FicheroTexto.txt");
        FileReader fr = new FileReader(fichero);

        int i; //Para almacenar los caracteres

        System.out.println("CONTENIDO DEL FICHERO: " + fichero.getName());

        while((i = fr.read()) != -1) {
            //System.out.println((char) i); //parseamos para convertir en char
            System.out.print((char) i); // Igual pero en una sola línea
        }


        fr.close();
    }

}
```

Acceder a ficheros (II)

ILERNA Online

El acceso puede ser a cualquier posición deseada → CD.

Se puede acceder a un dato sin recorrer los datos anteriores.

Aleatorio

RandomAccesFile

```
private static void randomAccesFile() {
    File fichero1 = new File( pathname: "ficheros/filewriter.txt");
    try {
        RandomAccessFile randomAccess = new RandomAccessFile(fichero1, mode: "rw");

        long pos = fichero1.length();
        randomAccess.seek(pos);
        randomAccess.writeBytes( s: "Linea desde randomAccesFile");
        randomAccess.seek(pos);
        String next;
        while ((next = randomAccess.readLine()) != null) {
            System.out.println(next + ": Tamaño = " + next.length());
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

> FICHEROS ALEATORIOS

- Dos nuevos constructores:
  - RandomAccessFile (String fichero, String modoAcceso)
  - RandomAccessFile (File f, String modoAcceso)
- Modo de Acceso: r (solo lectura), rw (lectura y escritura)
- Para leer y escribir datos usaremos clases DataInputStream y DataOutputStream
- RandomAccessFile usa puntero que indica posición actual

- Métodos RandomAccessFile

CLASE	FUNCIÓN
long getFilePointer()	Devuelve posición actual del puntero del fichero
void seek (long posicion)	Coloca puntero del fichero en posición determinada, empezando por el comienzo
long length()	Devuelve tamaño del fichero en bytes. Posición length() marca EOF
int skipBytes (int desplazamiento)	Desplaza puntero desde posición actual el número de bytes indicados en

EJERCICIOS PARA PRACTICAR

1. Crea un programa en Java que inserte datos de empleados en un fichero aleatorio, indicando apellido, departamento y salario. Para cada empleado se insertará un identificador (mayor que cero) y coincidirá con el índice + 1 que recorren los arrays
   a. Identificador = 4 bytes entero
   b. Apellido = 10 caracteres UNICODE, cada caracter 2 bytes (20 bytes en total)
   c. Departamento = 4 bytes entero
   d. Salario = 8 bytes double

EJERCICIOS PARA PRACTICAR

1. Crea un programa en Java que muestre los datos introducidos en el fichero del Ejercicio Anterior
2. El posicionamiento para empezar a recorrer es 0 y para recuperar los datos hay que añadir 36 a la variable del posicionamiento

Operaciones sobre ficheros

> ¿Qué operaciones se pueden realizar?

Altas → incluir un nuevo registro al archivo.

Modificaciones → Cambiar una parta del registro.

Bajas→ Dar de baja el registro.

Consultas → Buscar un registro.

EJERCICIOS PARA PRACTICAR

1. Crea un programa en Java que consulte los datos de un empleado en particular del fichero creado en el Ejercicio Anterior
    1. Por ejemplo: busca el empleado con ID 5

EJERCICIOS PARA PRACTICAR

1. Crea un programa en Java que inserte los datos de un empleado nuevo en fichero creado en el Anterior
    1. Puedes insertar los datos desde el código, desde la consola o como argumentos del main

```java
import java.io.*;
public class LeerFichAleatorio {
  public static void main(String[] args) throws IOException {
    File fichero = new File("AleatorioEmple.dat");
    //declaramos el fichero de acceso aleatorio
    RandomAccessFile file = new RandomAccessFile(fichero, "r");
    //
    int  id, dep, posicion;
    Double salario;
    char apellido[] = new char[10], aux;

    posicion = 0;  //para situarnos al principio

    for(;;){  //recorro el fichero    búcle for infinito
     file.seek(posicion); //nos posicionamos en posicion
     id = file.readInt();   // obtengo id de empleado

       //recorro uno a uno los caracteres del apellido
       for (int i = 0; i < apellido.length; i++) {
          aux = file.readChar();
          apellido[i] = aux;    //los voy guardando en el array
       }

       //convierto a String el array
       String apellidos = new String(apellido);
     dep = file.readInt();        //obtengo dep
     salario = file.readDouble(); //obtengo salario

     if(id >0)
         System.out.printf("ID: %s, Apellido: %s, Departamento: %d, Salario: %.2f %n",
             id,   apellidos.trim(), dep, salario);

       //me posiciono para el sig empleado, cada empleado ocupa 36 bytes
       posicion= posicion + 36;

     //Si he recorrido todos los bytes salgo del for
       if (file.getFilePointer() == file.length())break;

    }//fin bucle for
    file.close();  //cerrar fichero
    }
}
```

XML

> eXtensible Markup Language

La información que contienen está jerarquizada y estructurada.

```
<fichero>
    <tema>tema1</tema>
    <autor>Jorge Juan Delgado</autor>
</fichero>
```

Funciones

- Proporcionar datos en una base de datos
- Almacenar copias de esas bases de datos
- Escribir archivos de configuración de programas
- Efectuar comandos en servidores remotos en el protocolo SOAP.

Para leer un fichero XML necesitamos hacer uso de un parser

> Parsers

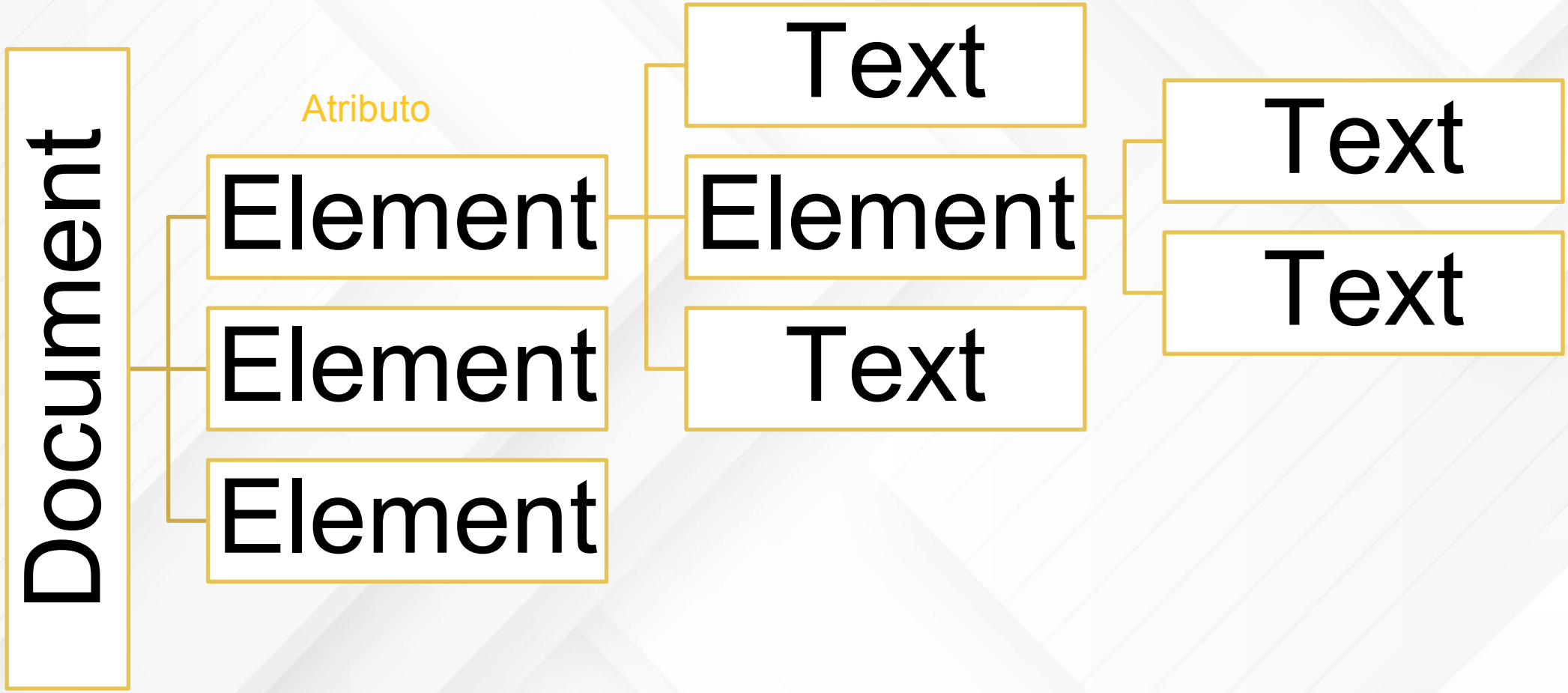
- Estructura en memoria.
- Más consumo de memoria.
- Tiene toda la información del fichero.

DOM

- Lectura secuencial.
- Menos consumo de memoria.
- Más lento porque para acceder a un nodo, hay que pasar por todos los nodos anteriores.

SAX

ILERNA Online



ILERNA Online

Lectura secuencial del documento
Callbacks según el elemento encontrado

startDocument()

startElement()

startElement()

startElement() chracters() endElement()

startElement() chracters() endElement()

endElement()

endElement()

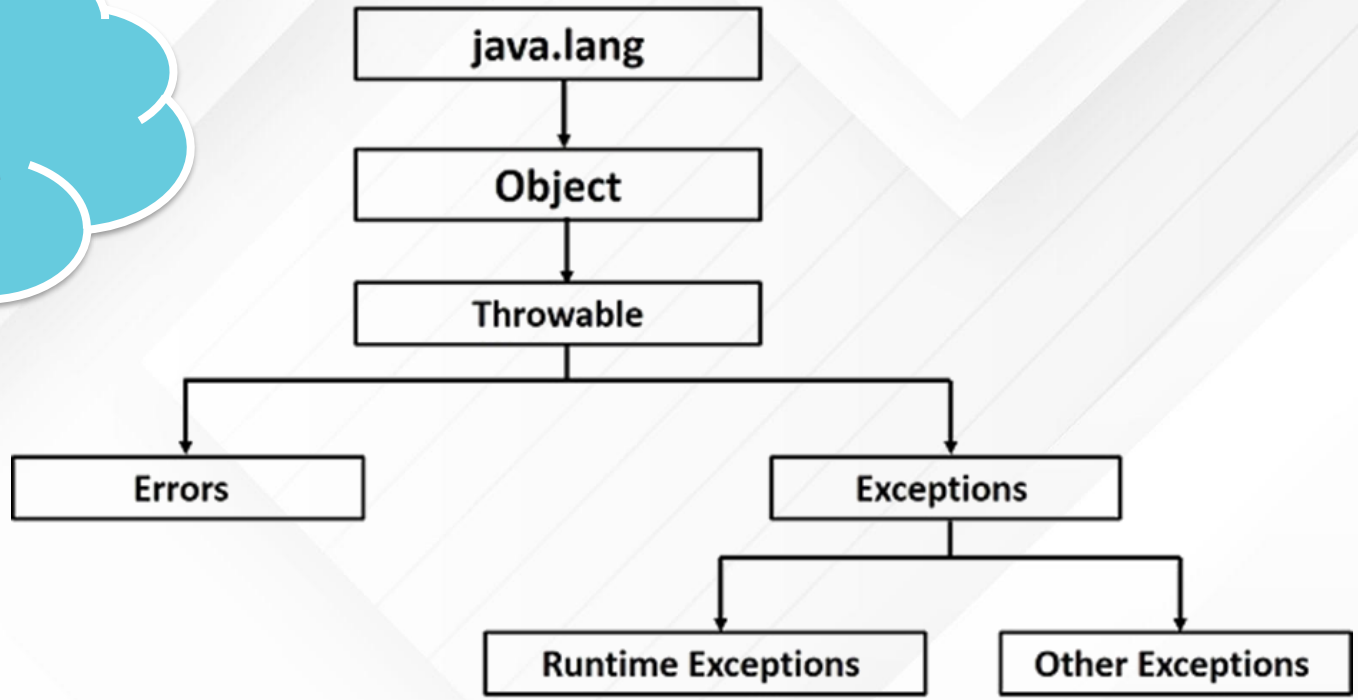
endDocument()

```
<?xml version="1.0" encoding="UTF-8"?>
<alumnos>
    <alumno>
        <nombre>Pablo</nombre>
        <apellido>López</centro>
    </alumno>
</alumnos>
```

EXCEPCIONES

ILERNA Online

➢Las Excepciones en Java son objetos que derivan de las clase Exception y que a su ver derivan de la clase Throwable



> Try.. Catch

Un bloque try catch nos permite capturar Excepciones y así de esta manera evitar que nuestro programa deje de funcionar en un momento inesperado.

```
try {
    FileOutputStream fileOutputStream = new FileOutputStream(copia, false);
    for (Integer b : bytes
    ) {
        fileOutputStream.write(b);
    }
    fileOutputStream.close();
    ;
} catch (IOException ex) {
    System.out.println("No se ha podido escribir en el fichero");
} finally {
    System.out.println("Función fileOutputStream finalizada");
}
```

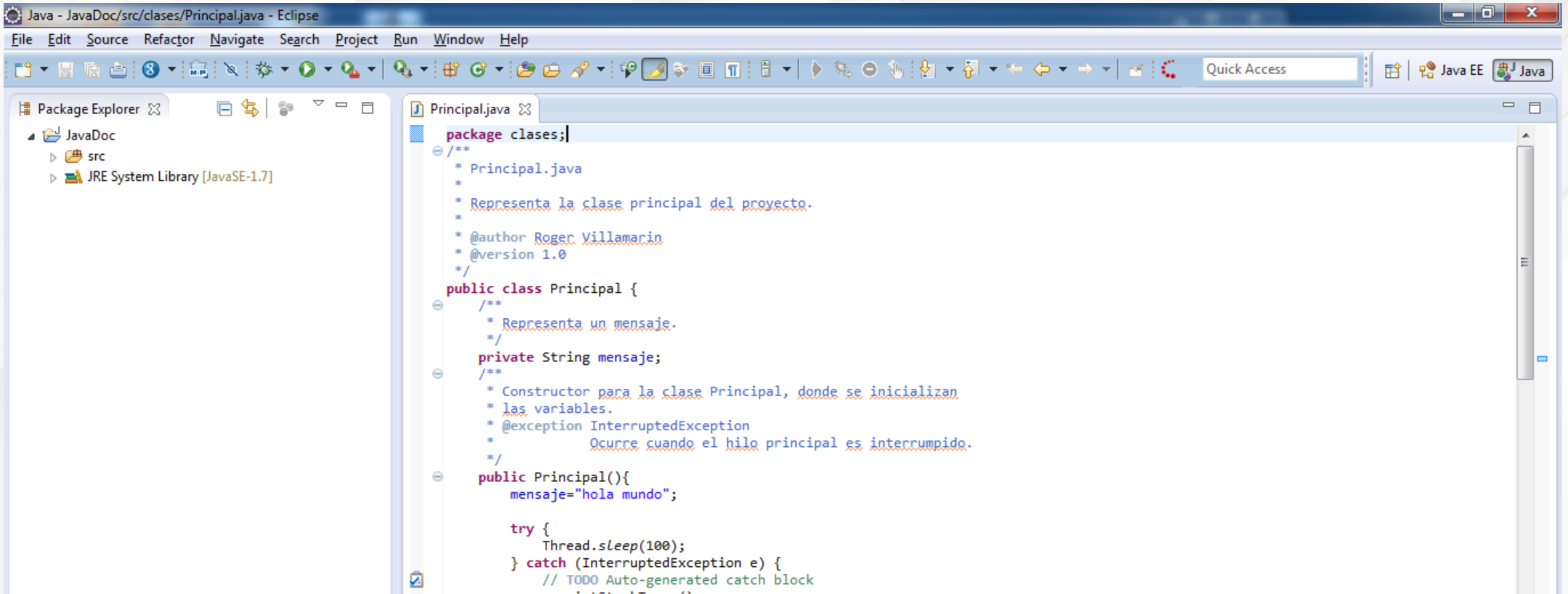
> Crear nuestras excepciones

Crear una nueva clase
Extender de la Exception deseada como:
Exception
Runtime Exception
Crear el constructor con los parámetros deseados.

```java
public class InvalidNumberException extends RuntimeException {
    public InvalidNumberException(String msg) {
        super(msg);
    }
}
```

PRUEBAS Y DOCUMENTACIÓN

- Comentar el código es garantía de calidad
- Facilita el mantenimiento
- En Java se realiza con JavaDoc

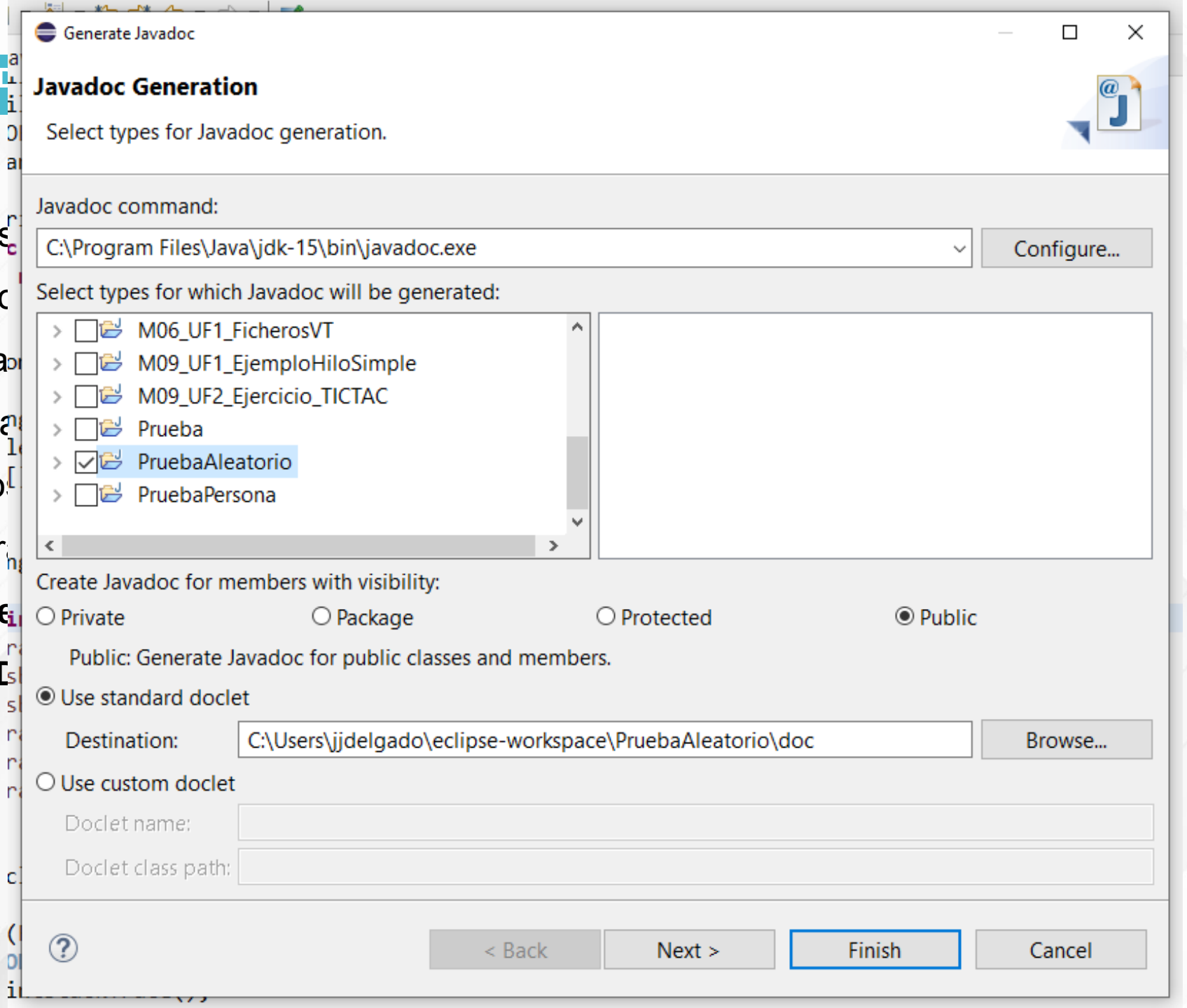


> PRUEBAS Y DOCUMENTACIÓN

- JavaDoc es una herramienta de Oracle que se usa para documentar las clases
- Podemos añadir comentarios en el código
- Permite usar etiquetas HTML dentro de los comentarios
- Para generar JavaDoc:
- **En línea de comandos: javadoc-d doc src**
- **Desde el IDE Project / Generate JavaDoc**

```
/**
 * Aquí pondremos un resumen de la finalidad de esta clase
 * Ej: Clase para documentar el funcionamiento de JavaDoc
 *
 * @author Ilerna
 *
 */
public class JavaDoc {

    /**
     * Atributo nombre de la clase javaDoc
     */
    private String nombre;

    /**
     * <p>Ejemplo de Javadoc con tags html .
     * <a href="http://www.ilerna.es">Web ilerna</a>
     * </p>
     * @param ejemplo String que pasamos por parametro
     * @return String con el resultado del metodo
     * @see <a href="http://www.ilerna.es">Ejemplos</a>
     * @since 1.0
     */
    public String ejemploMetodoJavadoc(String ejemplo) {
        // Comentario con explicación de lo que se realiza en el
    método
        return "OK";
    }


}
```

- JavaDoc es
  usa para dc
- Podemos a
- Permite usa
  comentario
- Para gener
- **En línea de**
- **Desde el ID**



ILERNA Online

- Para hacer pruebas usaremos Junit
- **Framework para hacer pruebas de código**

ILERNA Online

- Etiquetas más usadas para hacer test con JUnit

TAG	DESCRIPCIÓN
@Test	Identifica un método como un método test.
@Before	Indica qué se ejecuta antes de cada test. Se usa para preparar el entorno del test antes de ejecutarlo.
@After	Indica qué se ejecutará después de cada test. Se usa para limpiar el entorno después de la ejecución de cada test.
@BeforeClass	Se ejecuta solo una vez, antes de que se ejecuten todos los test. Se usa, por ejemplo, para conectar la base de datos.
@AfterClass	Se ejecuta solo una vez después de ejecutar todos los test. Se usa para limpiar el entorno.
@Ignore or @Ignore ("why disabled")	Marca ese test como deshabilitado.
@Rule	Establece una regla, añade funcionalidades extra a nuestro test.

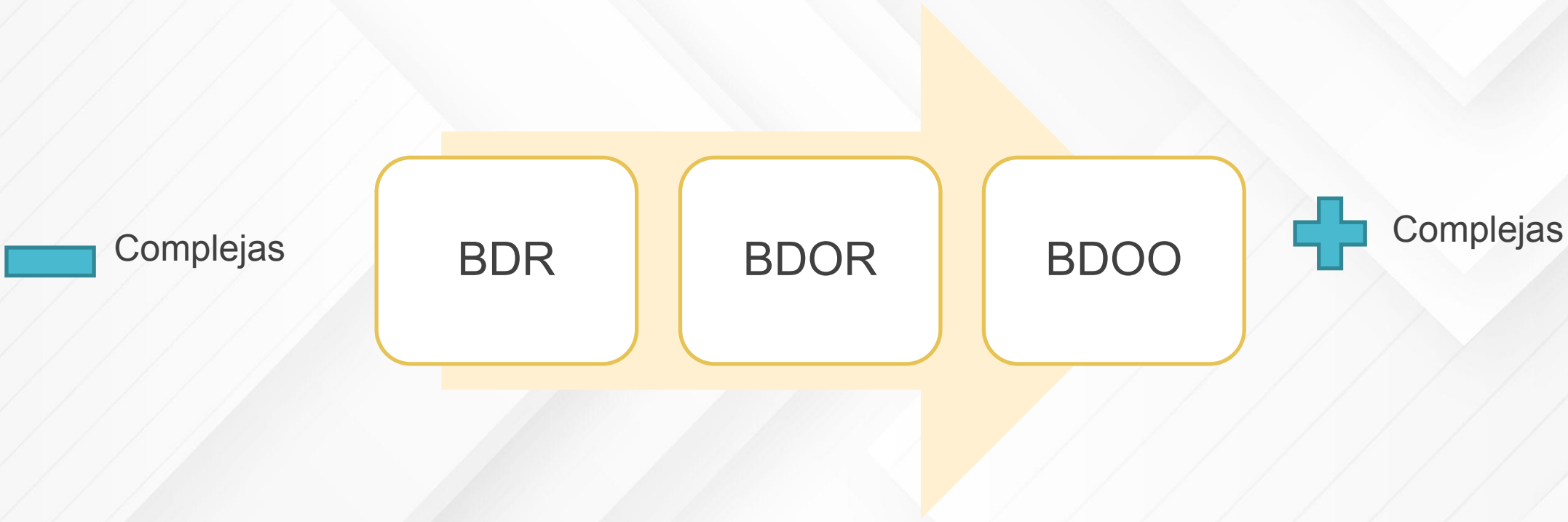
ILERNA Online

- Métodos más importantes para trabajar con JUnit
- En el material didáctico tenéis el paso a paso para configurarlo
- Acceder al repositorio para ver ejercicios resueltos con JUnit

MÉTODO	DESCRIPCIÓN
fail([message])	Se usa para hacer que el método falle y descubrir qué partes del código no se contemplan. Se suele hacer antes de desarrollar. El parámetro *message* es opcional.
assertTrue([message,]boolean condition)	Comprueba si la condición es *true*.
assertFalse([message,]boolean condition)	Comprueba si la condición es *false*.
assertFalse([message,]boolean condition)	Hace test de dos valores para comprobar si son el mismo.
assertEquals([message,] expected,actual, tolerance)	Comprueba si un *float* o *double* son iguales.
assertNull([message,] object)	Comprueba si un objeto es *null*.
assertNotNull([message,] object)	Comprueba si el objeto no es *null*.
assertSame([message,] expected, actual)	Comprueba si las dos variables hacen referencia al mismo objeto.
assertNotSame([message,] expected, actual)	Comprueba si las dos variables no hacen referencia al mismo objeto.

VIDEOTUTORÍA 04 - UF2

Persistencia en BDR-BDOR-BDOO



Existe una gran diferencia de esquemas entre los elementos que se almacenan y las características de donde se almacenan

ILERNA Online

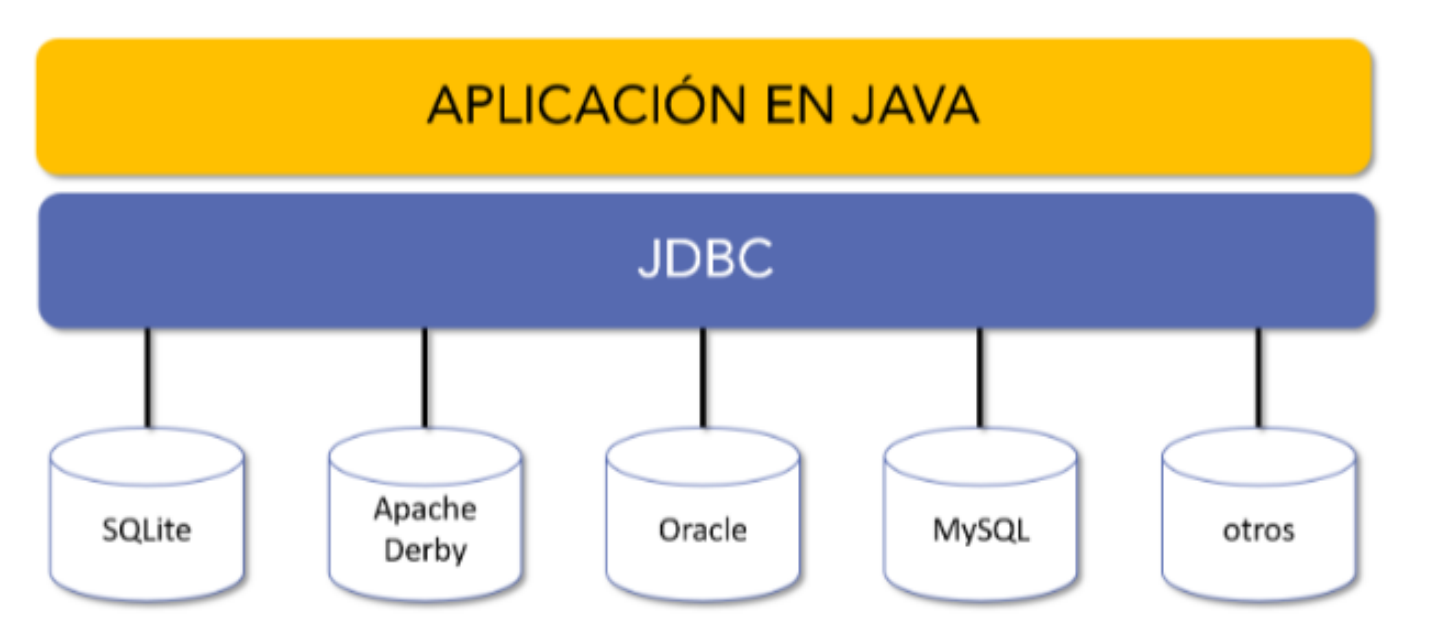
Conectores

ODBC	Especificación de Microsoft que tiene una interfaz en C y que permite el acceso desde cualquier aplicación
JDBC	Ofrece conectividad de BD con aplicaciones Java y define una API para conectarse a principalmente bases de datos de tipo relacional

ILERNA Online

Balanceo de carga: Distribuir peticiones (visitas) entre distintos servidores, útil para campañas para no sobrecargar el servidor.

JDBC define una arquitectura estándar en que los fabricantes pueden crear sus drivers para que las aplicaciones Java puedan acceder a los datos.



Tareas que realiza

- Conectar a BD
- Realizar consultar e instrucciones para actualizar la BD
- Recuperar y procesar los resultados de la BD

ILERNA Online

ACCESO A DATOS MEDIANTE JDBC

- La biblioteca JDBC incluye API para cada una de las tareas mencionadas a continuación, que están comúnmente asociadas con el uso de la base de datos
    - Haciendo una conexión a una base de datos
    - Crear sentencias SQL o MySQL
    - Ejecutando consultas SQL o MySQL en la base de datos
    - Visualización y modificación de los registros resultantes

ILERNA Online

ACCESO A DATOS MEDIANTE JDBC

- La biblioteca JDBC incluye API para cada una de las tareas mencionadas a continuación, que están comúnmente asociadas con el uso de la base de datos
    - Haciendo una conexión a una base de datos
    - Crear sentencias SQL o MySQL
    - Ejecutando consultas SQL o MySQL en la base de datos
    - Visualización y modificación de los registros resultantes

ILERNA Online

CÓMO FUNCIONA JDBC

- JDBC define varias interfaces que pueden realizar operaciones con bases de datos.
- Estas están definidas en el paquete java.sql.

CLASE E INTERFAZ	DESCRIPCIÓN
Driver	Permite conectarse a una BD: cada gestor de BD requiere un driver distinto
DriverManager	Permite gestionar todos los drivers instalados en el sistema
DriverPropiertyInfo	Proporciona información sobre un driver
Connection	Representa una conexión con una BD. Una aplicación puede tener más de una conexión
DatabaseMetadata	Proporciona información acerca de una BD (tablas que contiene, tipos de datos, etc)
Statement	Permite ejecutar sentencias SQL sin parámetros
PreparedStatement	Permite ejecutar sentencias SQL con parámetros de entrada
CallableStatement	Permite ejecutar sentencias SQL con parámetros de entrada y salida, como llamadas a procedimientos

ILERNA Online

PASOS PARA USAR JDBC

1. Importar la clase necesaria (en nuestro caso, MySQL)
2. Librería: mysql-connector-java-8.0.19.jar por ejemplo
    1. https://dev.mysql.com/downloads/connector/j/ -- LINUXç
    2. https://dev.mysql.com/downloads/windows/installer/8.0.html -- WINDOWS
    3. http://www.java2s.com/Code/Jar/m/Downloadmysqlconnectorjavajar.htm -- DRIVER

ILERNA Online

SERVIDOR LOCAL

- Para poder trabajar con una BD en local, necesitaremos un emulador de Apache y MySQL
    - XAMPP - https://www.apachefriends.org/es/download.html
    - WAMPP
    - otros

ORM - HIBERNATE

Herramienta Java, que facilita el mapeo de objetos entre las bases de datos relaciones y la de objetos mediante ficheros XML que permiten establecer sus relaciones.

Permite la persistencia de los objetos de una aplicación OO mediante una base de datos relacional

Ofrece un lenguaje propio llamado HQL (Hibernate Query Language) para simplificar las peticiones a la BD

hibernate

https://hibernate.org/orm/releases/

Para emparejar tu programa con la base de datos es necesario un intermediario que ayude a realizar dicha conexión, a eso se le llama comúnmente como mapeo.

HIBERNATE

https://hibernate.org/orm/releases/

ORM | Search | Validator | OGM | Tools | Others | Blog | Forums | Community | Follow us

Hibernate ORM

Releases

Latest stable (5.4)

Development (6.0)

- About
- Releases
- Documentation
- Books
- Migration guides
- Roadmap
- Tooling
- Envers
- Contribute
- Paid support

Interested in commercial support?
Check out Red Hat's offering .

Series

development

6.0 2020-08-20

Performance, HQL, Criteria, Type System

More info

stable

5.4 2020-11-03

EntityGraph improvements, JDK 11 support

More info

stable

5.3 2020-08-05

JPA 2.2, inheritance caching

More info

ILERNA Online

How to get it

Maven, Gradle...

Maven artifacts of Hibernate ORM are published to Maven Central and to Bintray.

You can find the Maven coordinates of all artifacts through the link below:

Maven artifacts

Zip archive

Direct download is available from SourceForge:

Download Zip archive

https://hibernate.org/orm/releases/

Hibernate ORM

Documentation - 5.4

Other versions

5.4

Guides and such

Getting Started Guide

A quickstart-style guide with tutorials. See also the Obtaining Hibernate section discussing the Hibernate artifacts and how to obtain them.

Migration Guide

Migration guide covering migration to 5.4 from the previous version

User Guide

Guide covering most user facing concepts and APIs of Hibernate

Integrations Guide

Guide covering topics of interest for developers looking to develop integrations with Hibernate.

Hibernate JavaDoc

The Hibernate JavaDocs

WildFly, updating in

Guide to update WildFly 12 to use the latest version of Hibernate ORM 5.4

JPA 2.2 JavaDoc

The JPA (2.2) JavaDocs

FAQ

ILERNA Online

Configuration

- Se configura hibernate, se especifica la ubicación de los documentos que son utilizados para el mapeo de objetos.

SessionFactory

- Esta interfaz permite la obtención de instancias Session y habitualmente solo existe una por sesión. En caso de que haya más de una BD en un proyecto, se necesitará una para cada una de ellas

Query

- Se llevan a cabo las consultas y se harán en HQL o en SQL nativo.

Transaction

- Cualquier error en la transacción producirá un fallo de la misma.

>Fichero de configuración hibernate.cfg.xml

```xml
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/HibernateConfigurationDTD//EN"
        "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <property name="dialect">org.hibernate.dialect.MySQL5Dialect</property>
        <property name="hbm2ddl.auto">create</property>
        <property name="connection.url">jdbc:mysql://127.0.0.1:3360/m06</property>
        <property name="connection.username">root</property>
        <property name="connection.password">secret</property>
        <mapping resource="alumno.hbm.xml"/>
    </session-factory>
</hibernate-configuration>
```

Dialect: Como se comunica JDBC con la BD

Hbm2ddl.auto:

validate: validate el equema.
create: crea el esquema.
create-drop: Crea y destruye el esquema.
update: actualiza el esquema.
none: No realiza cambios

> Fichero de configuración de Tabla

```xml
<?xml version='1.0' encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
        "-//Hibernate/HibernateMappingDTD5.3//EN"
        "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
    <class name="Alumno" table="Alumnos" catalog="m06">
        <id name="idAlumno">
            <column name="alumno_id"/>
            <generator class="increment"/>
        </id>
        <property name="nombre"></property>
    </class>
</hibernate-mapping>
```

Clase Alumno	Tabla Alumno
id_alumno	alumno_id
nombre	nombre

> Fichero de configuración de Tabla

```
<?xml version='1.0' encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
        "-//Hibernate/HibernateMappingDTD5.3//EN"
        "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
    <class name="Alumno" table="Alumnos" catalog="m06">
        <id name="idAlumno">
            <column name="alumno_id"/>
            <generator class="increment"/>
        </id>
        <property name="nombre"></property>
    </class>
</hibernate-mapping>
```

Property: todos los atributos que tenemos en la clase que NO son ID (nombre, edad, dirección, etc)

Type: tipo de dato que se usará para esta propiedad
En el enlace podremos encontrar los tipos disponibles

https://www.tutorialspoint.com/hibernate/hibernate_mapping_types.htm

Primitive Types

Mapping type	Java type	ANSI SQL Type
integer	int or java.lang.Integer	INTEGER
long	long or java.lang.Long	BIGINT
short	short or java.lang.Short	SMALLINT
float	float or java.lang.Float	FLOAT
double	double or java.lang.Double	DOUBLE
big_decimal	java.math.BigDecimal	NUMERIC
character	java.lang.String	CHAR(1)
string	java.lang.String	VARCHAR
byte	byte or java.lang.Byte	TINYINT
boolean	boolean or java.lang.Boolean	BIT
yes/no	boolean or java.lang.Boolean	CHAR(1) ('Y' or 'N')
true/false	boolean or java.lang.Boolean	CHAR(1) ('T' or 'F')

Date and Time Types

Mapping type	Java type	ANSI SQL Type
date	java.util.Date or java.sql.Date	DATE
time	java.util.Date or java.sql.Time	TIME
timestamp	java.util.Date or java.sql.Timestamp	TIMESTAMP
calendar	java.util.Calendar	TIMESTAMP
calendar_date	java.util.Calendar	DATE

Binary and Large Object Types

Mapping type	Java type	ANSI SQL Type

> Fichero de configuración de Tabla

¿Qué pasa si un alumno tuviera un campo que almacenara una LISTA?

Por ejemplo, que tuviera un campo que registrara el número de asignaturas en las que está matriculado. Esto se gestionaría con una Lista

Tarea de investigación para la PAC de Desarrollo

https://www.tutorialspoint.com/hibernate/hibernate_mapping_types.htm

Primitive Types

Mapping type	Java type	ANSI SQL Type
integer	int or java.lang.Integer	INTEGER
long	long or java.lang.Long	BIGINT
short	short or java.lang.Short	SMALLINT
float	float or java.lang.Float	FLOAT
double	double or java.lang.Double	DOUBLE
big_decimal	java.math.BigDecimal	NUMERIC
character	java.lang.String	CHAR(1)
string	java.lang.String	VARCHAR
byte	byte or java.lang.Byte	TINYINT
boolean	boolean or java.lang.Boolean	BIT
yes/no	boolean or java.lang.Boolean	CHAR(1) ('Y' or 'N')
true/false	boolean or java.lang.Boolean	CHAR(1) ('T' or 'F')

Date and Time Types

Mapping type	Java type	ANSI SQL Type
date	java.util.Date or java.sql.Date	DATE
time	java.util.Date or java.sql.Time	TIME
timestamp	java.util.Date or java.sql.Timestamp	TIMESTAMP
calendar	java.util.Calendar	TIMESTAMP
calendar_date	java.util.Calendar	DATE

Binary and Large Object Types

Mapping type	Java type	ANSI SQL Type

EJERCICIO PARA PRACTICAR

Crear ficheros de configuración para Hibernate

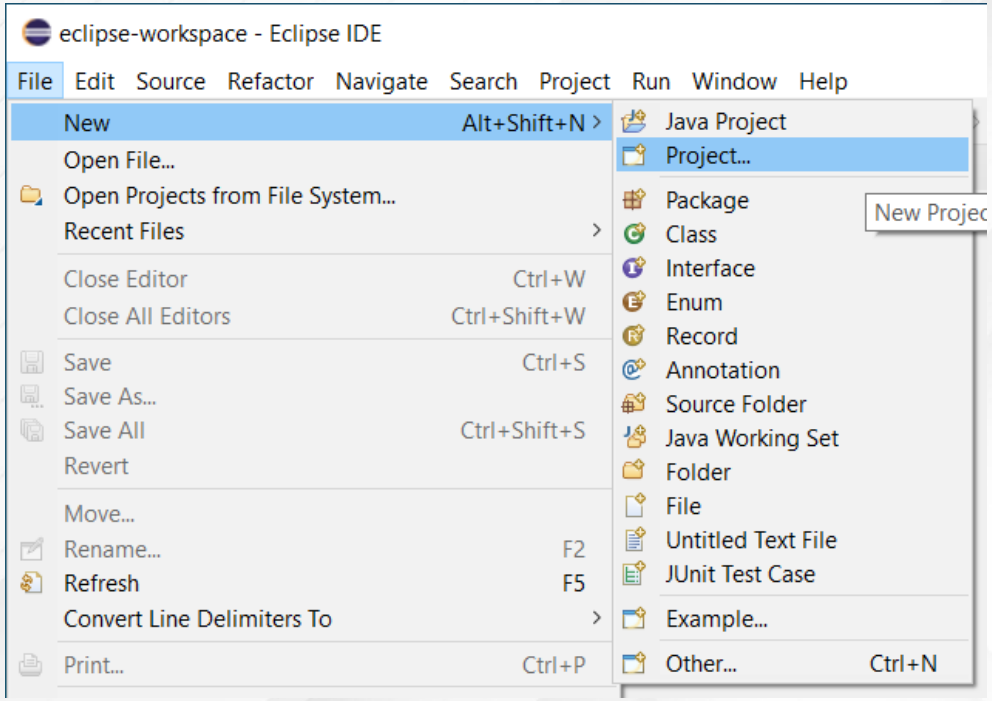
Opción A: USANDO MAVEN

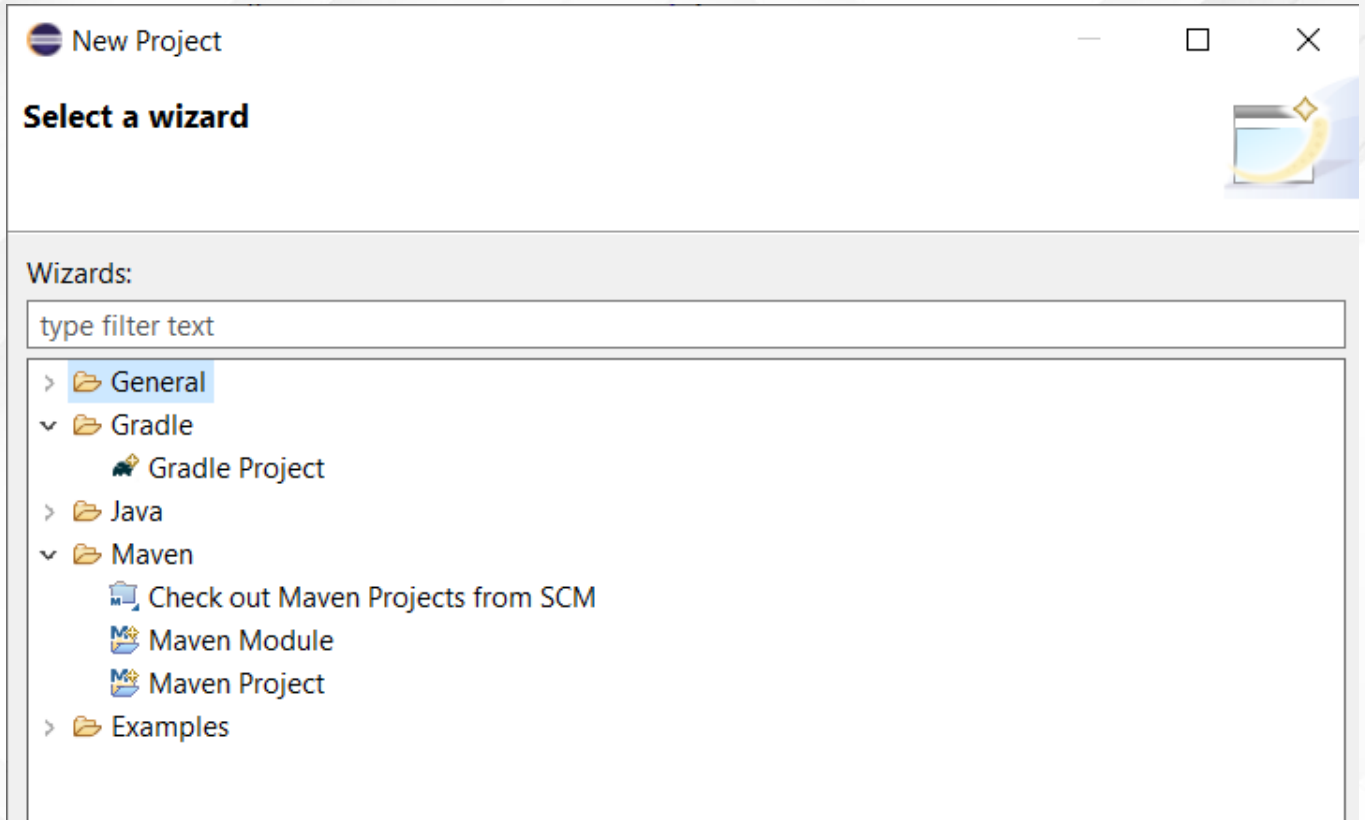
ILERNA Online

Crear un proyecto de configuración de Hibernate

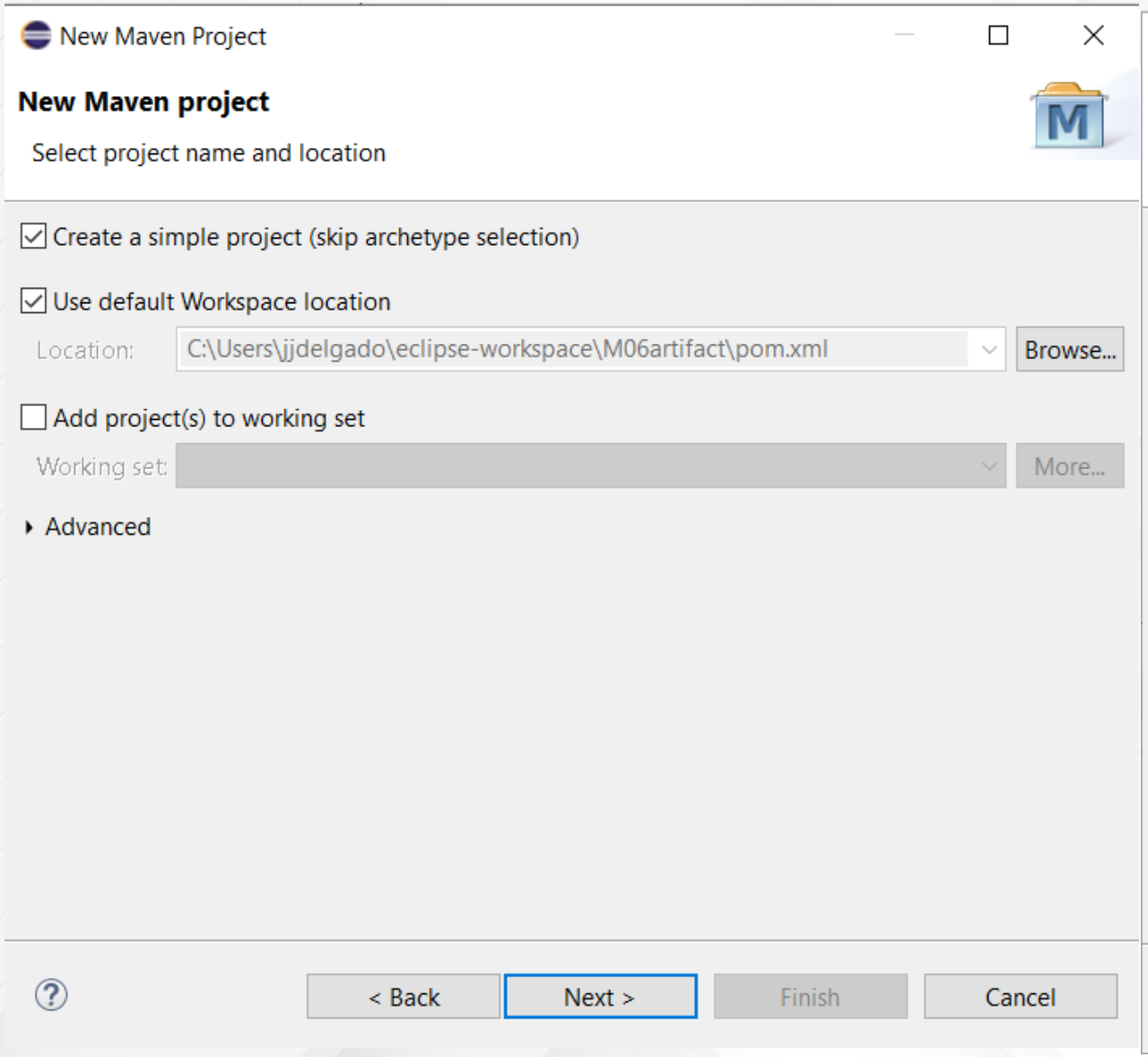
En Eclipse:

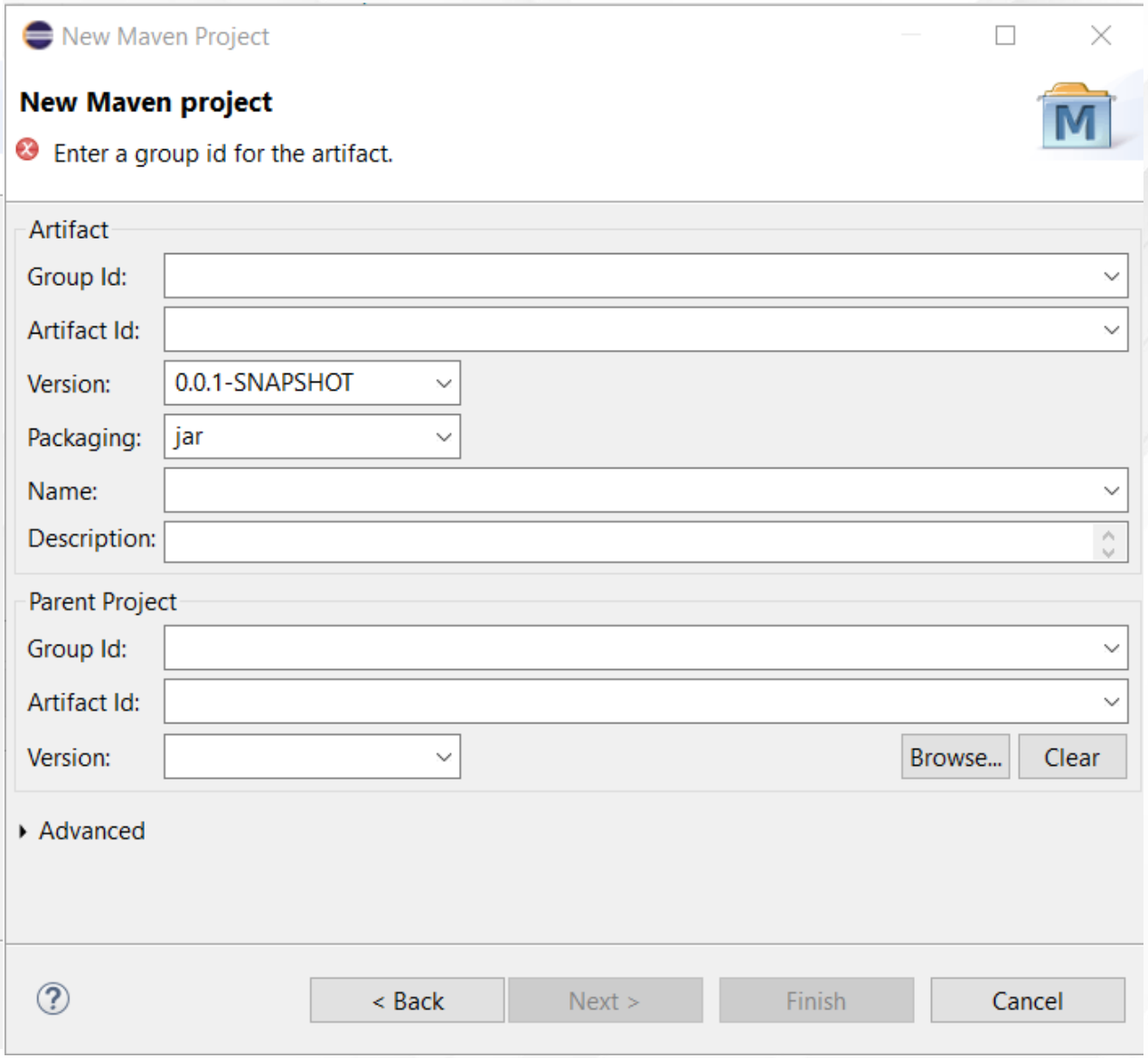
New / Project / Maven / Maven Project





> EJERCICIO PARA PRACTICAR

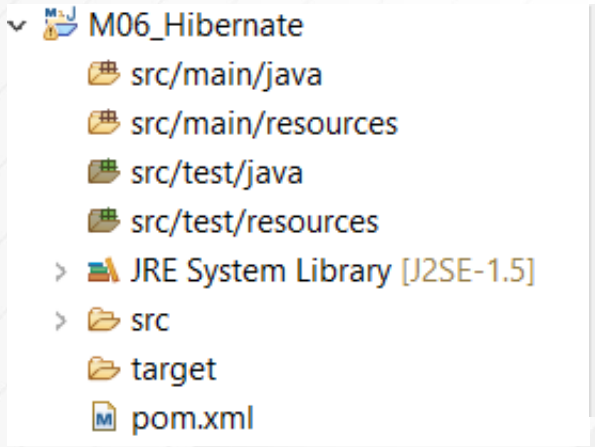




> EJERCICIO PARA PRACTICAR

ILERNA Online

Crear un proyecto de configuración de Hibernate



*M06_Hibernate/pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>M06</groupId>
  <artifactId>M06_Hibernate</artifactId>
  <version>0.0.1-SNAPSHOT</version>
</project>
```

ILERNA Online

Crear un proyecto de configuración de Hibernate

¿Cómo podemos importar nuevas librerías?

Etiqueta dependecies y dentro de ellas definir las librerías que necesitaremos

Tanto en Maven como en Gradle podemos incluir muchas librerías

Repositorio oficial de Maven
https://mvnrepository.com/artifact/org.hibernate/hibernate-core

Para lanzar una Query a la BD

```
//Recueperar los datos de la BD
tx = session.beginTransaction();

Query query = session.createQuery("FROM Alumno");
List list =  query.list();
for(int i =0; i<list.size(); i++)
{
    System.out.println(list.get(i).toString());
}
```

> EJERCICIO

Crear un proy

¿Cómo poden

Etiqueta depe
necesitaremos

Tanto en Mave

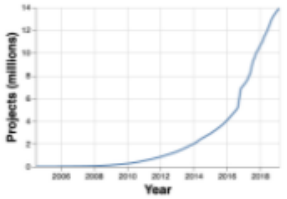
Repositorio ofi
https://mvnrep

MVNREPOSITORY

Search for groups, artifacts, categories

Search

Indexed Artifacts (18.3M)



Popular Categories

- Aspect Oriented
- Actor Frameworks
- Application Metrics
- Build Tools
- Bytecode Libraries
- Command Line Parsers
- Cache Implementations
- Cloud Computing
- Code Analyzers
- Collections
- Configuration Libraries
- Core Utilities
- Date and Time Utilities
- Dependency Injection
- Embedded SQL Databases
- HTML Parsers
- HTTP Clients
- I/O Utilities
- JDBC Extensions
- JDBC Pools
- JPA Implementations
- JSON Libraries
- JVM Languages
- Logging Frameworks

Home » org.hibernate » hibernate-core

Hibernate Core Relocation

Hibernate ORM 6.0.0.Alpha6 release. See http://hibernate.org/orm/releases/6.0

License	LGPL 2.1
Categories	Object/Relational Mapping
Tags	hibernate persistence relational mapping
Used By	3,286 artifacts

Note: This artifact was moved to:

org.hibernate.orm » hibernate-core » 6.0.0.Alpha6

Central (223) | Atlassian 3rdParty (4) | Atlassian 3rd-P Old (18) | Spring Lib Release (1) | Spring Plugins (4)
Redhat GA (44) | Redhat EA (19) | ImageJ Public (2) | ICM (5) | TU-Darmstadt (1) | Grails Core (5) | JCenter (23)
EBI (3) | Geomajas (1) | EBIPublic (1)

	Version	Repository	Usages	Date
6.0.x	6.0.0.Alpha6	Central	1	Aug, 2020
	6.0.0.Alpha5	Central	1	Apr, 2020
	6.0.0.Alpha4	Central	0	Dec, 2019
	6.0.0.Alpha3	Central	0	Nov, 2019
	6.0.0.Alpha2	Central	0	Apr, 2019
	5.4.23.Final	Central	20	Nov, 2020
	5.4.22.Final	Central	54	Sep, 2020
	5.4.21.Final	Central	86	Aug, 2020
	5.4.20.Final	Central	65	Aug, 2020
	5.4.19.Final	Central	31	Jul, 2020
	5.4.18.Final	Central	90	Jun, 2020

ILERNA Online

ILERNA Online

Crear un proyecto de configuración de Hibernate

¿Cómo podemos importar nuevas librerías?

Etiqueta de
necesitare

Tanto en M

Repositorio
https://mvr

*M06_Hibernate/pom.xml

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.Ilerna.m06</groupId>
  <artifactId>M06_Hibernate</artifactId>
  <version>0.0.1-SNAPSHOT</version>

  <dependencies>

        <!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->
        <dependency>
            <groupId>org.hibernate</groupId>
            <artifactId>hibernate-core</artifactId>
            <version>5.4.23.Final</version>
        </dependency>
  </dependencies>

</project>
```

EJERCICIO PARA PRACTICAR

Crear ficheros de configuración para Hibernate

Opción B: SIN USAR MAVEN

> EJERCICIO PARA PRACTICAR

ILERNA Online

Crear un proyecto de configuración de Hibernate

How to get it

Maven, Gradle...

Maven artifacts of Hibernate ORM are published to Maven Central and to Bintray.

You can find the Maven coordinates of all artifacts through the link below:

Maven artifacts

Zip archive

Direct download is available from SourceForge:

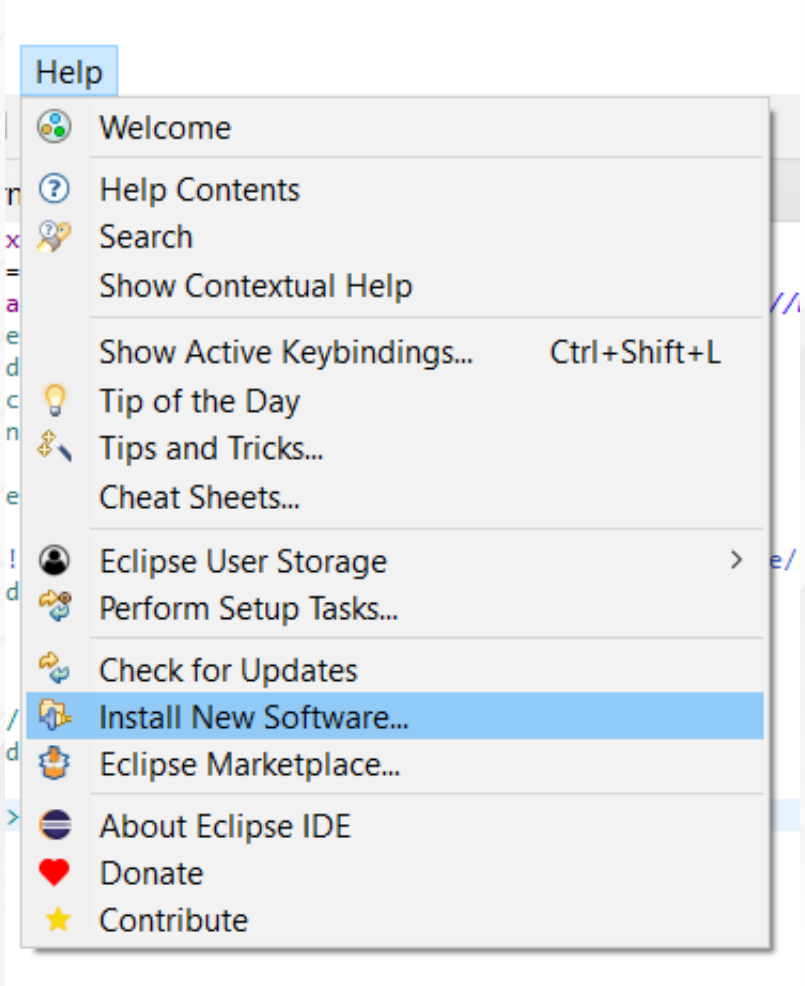
Download Zip archive

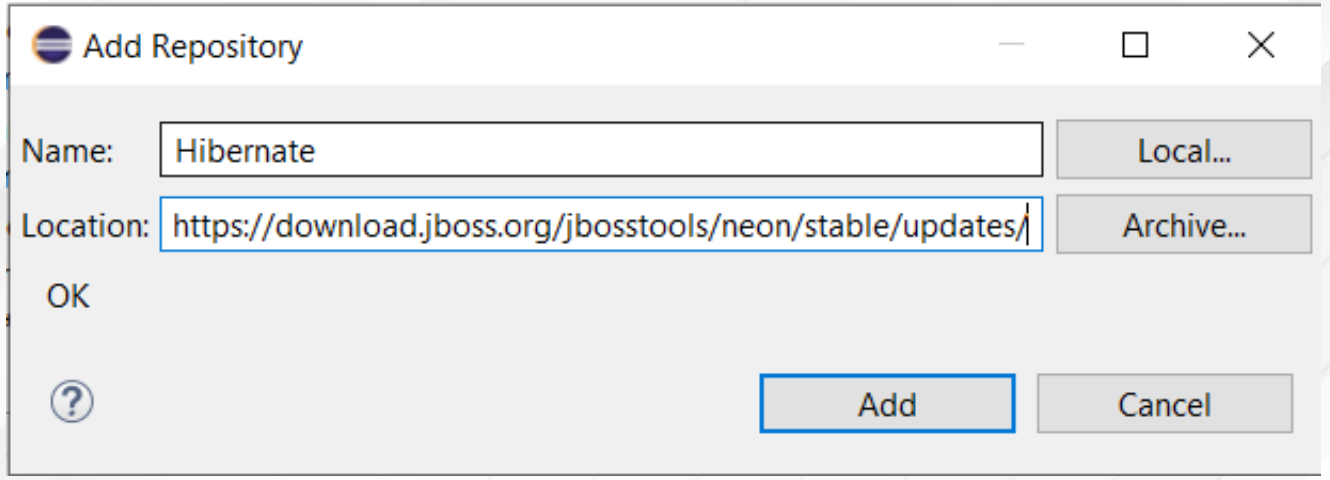
ILERNA Online

Crear un proyecto de configuración de Hibernate

En Eclipse:

Ayuda / Instalar nuevo software





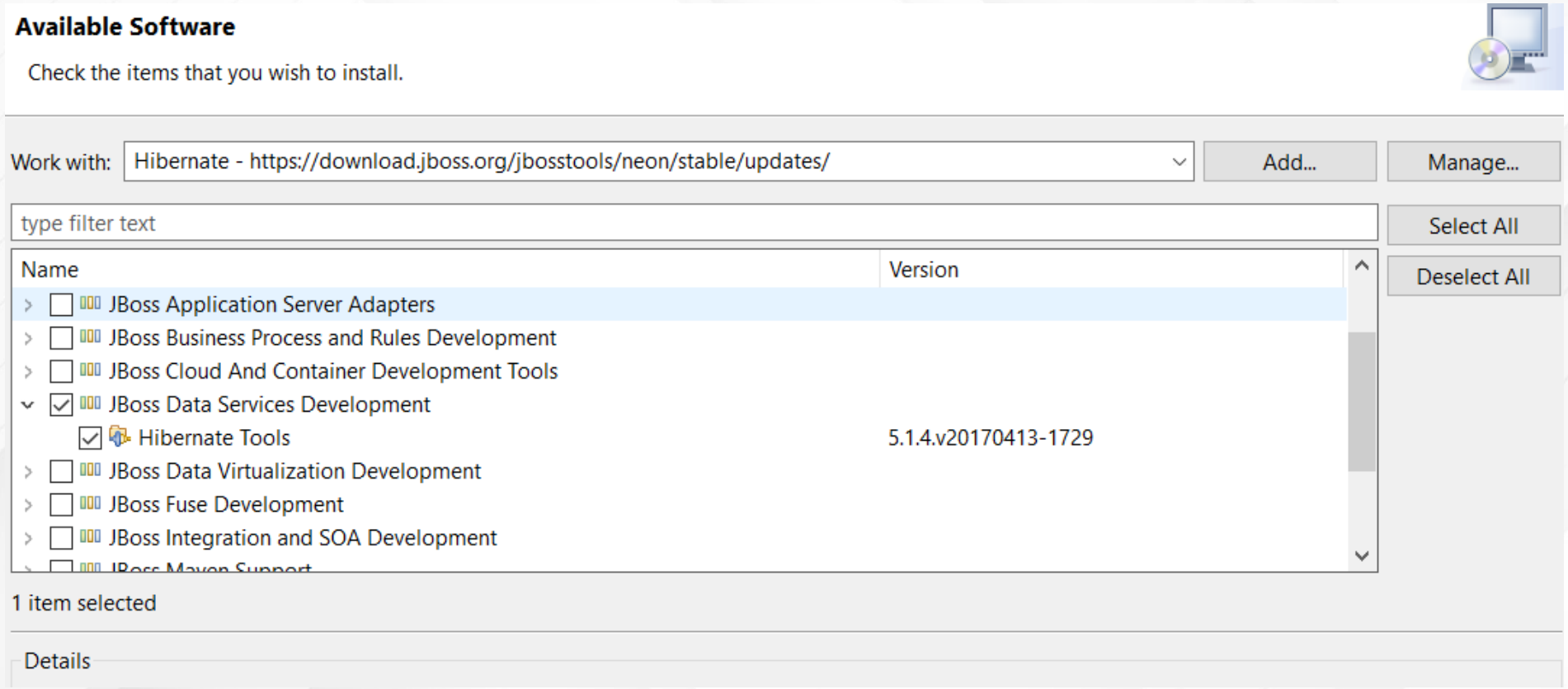
URL:
https://download.jboss.org/jbosstools/neon/stable/updates/

ILERNA Online

Crear un proyecto de configuración de Hibernate

En Eclipse:

En el listado que aparece, seleccionamos Jboss Data Services Sevelopment / HIbernate

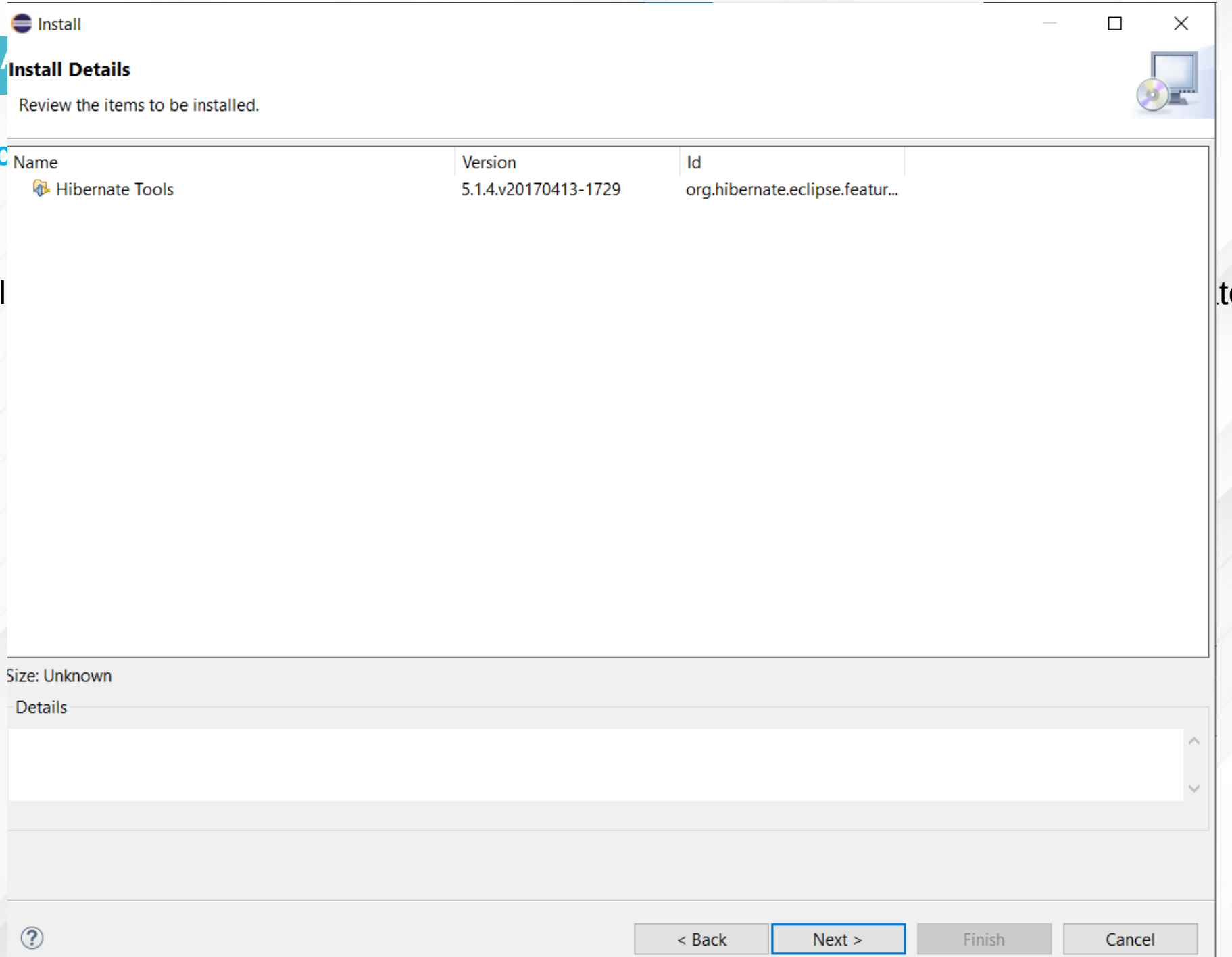


> EJERCICIO PA

Crear un proyec

En Eclipse:

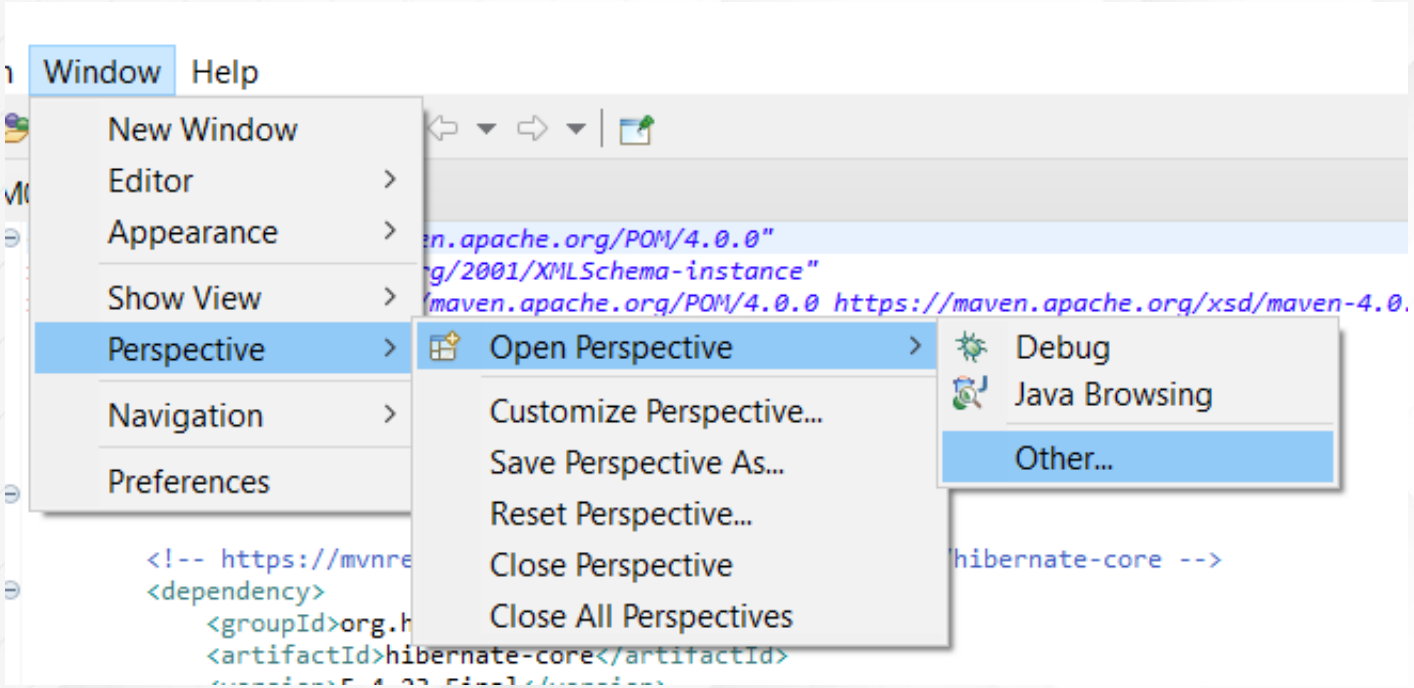
En el te

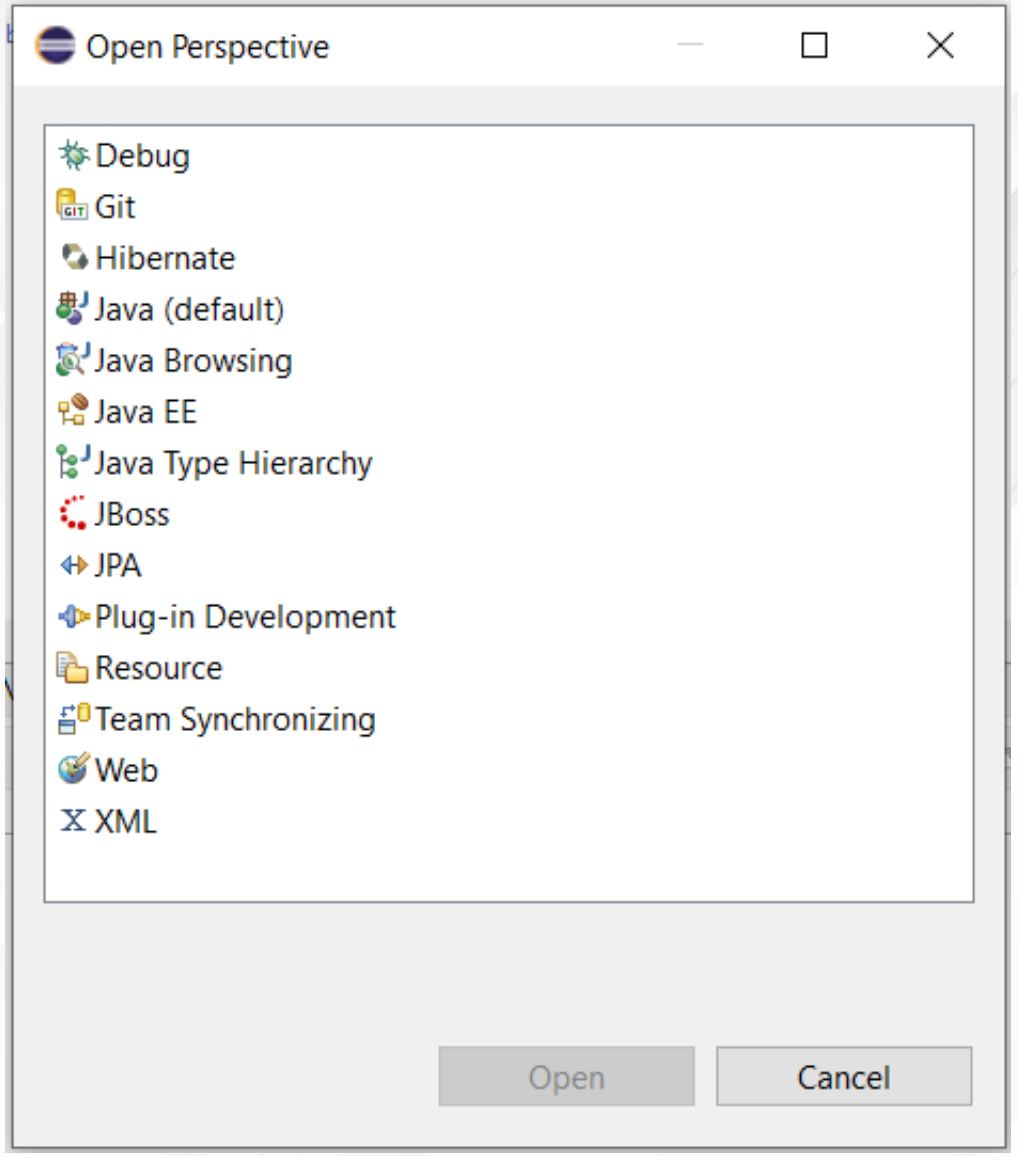


> EJERCICIO PARA PRACTICAR

Crear un proyecto de configuración de Hibernate

En Eclipse:

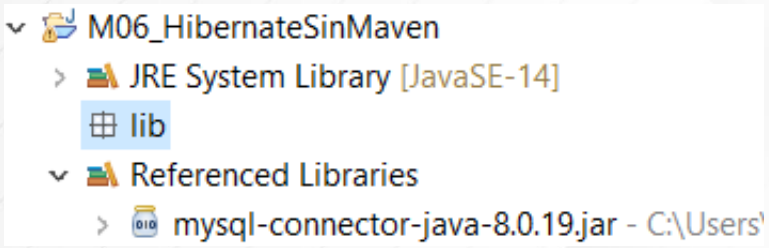


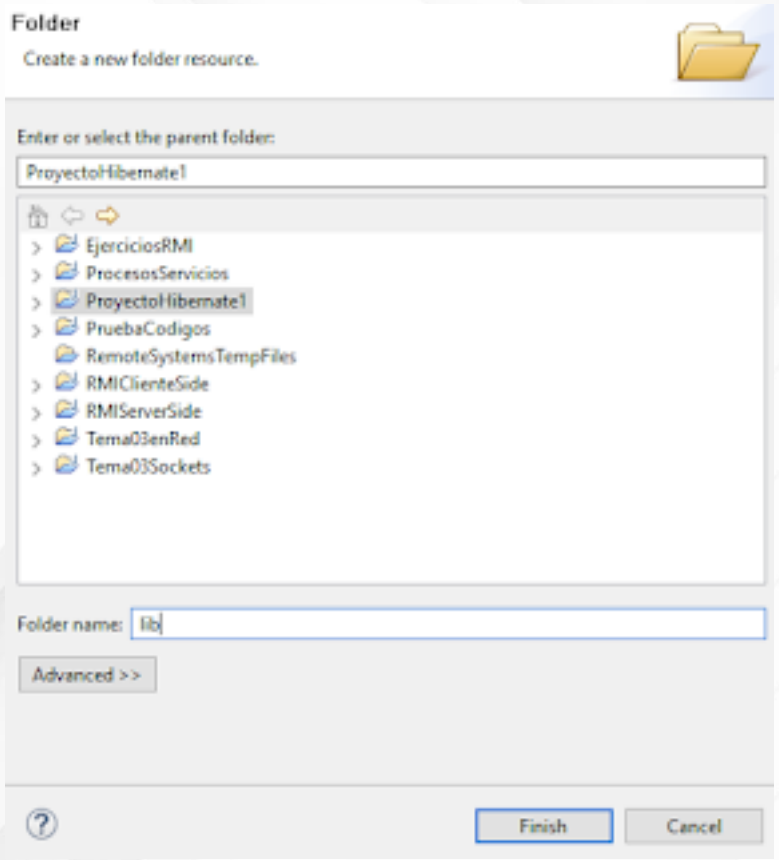


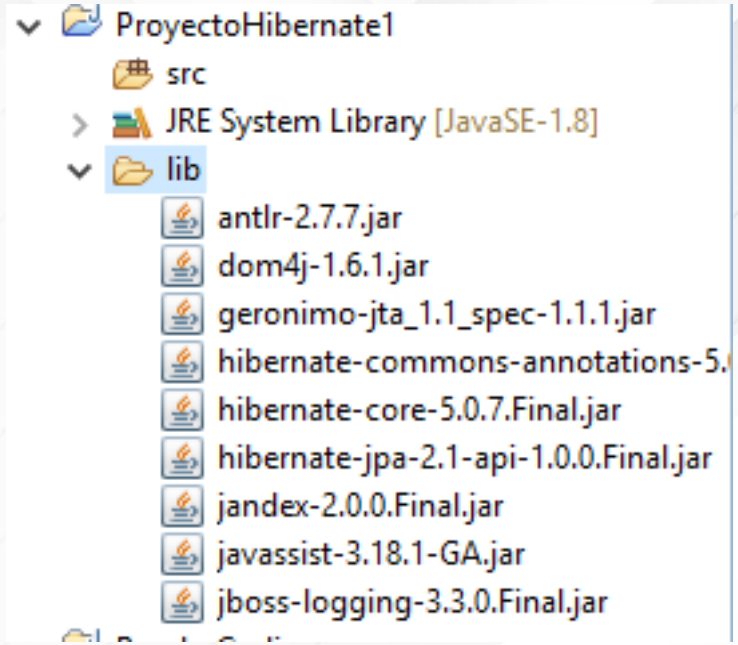
ILERNA Online

Crear un proyecto de configuración de Hibernate

En Eclipse:



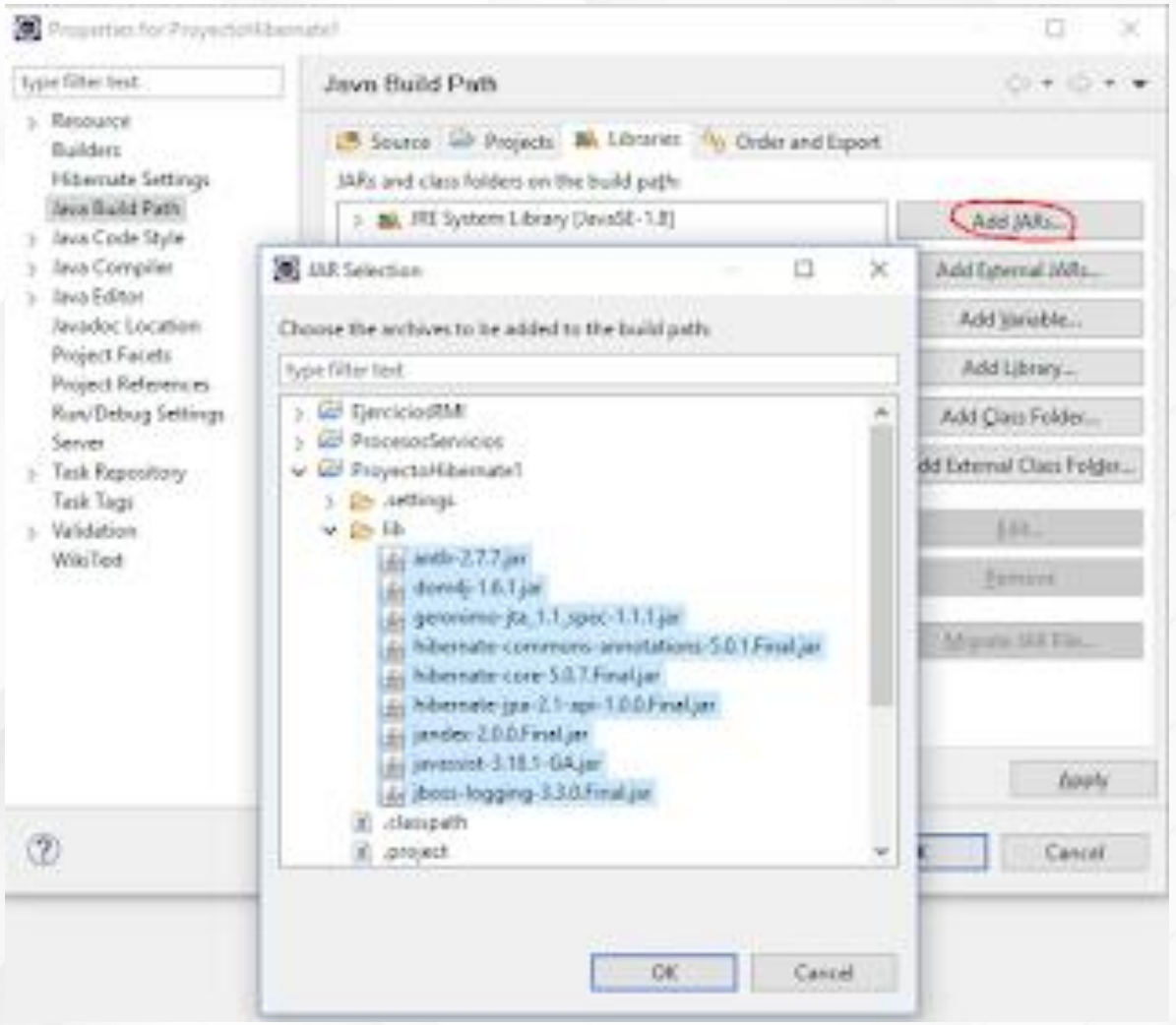




ILERNA Online

Crear un proyecto de configuración de Hibernate

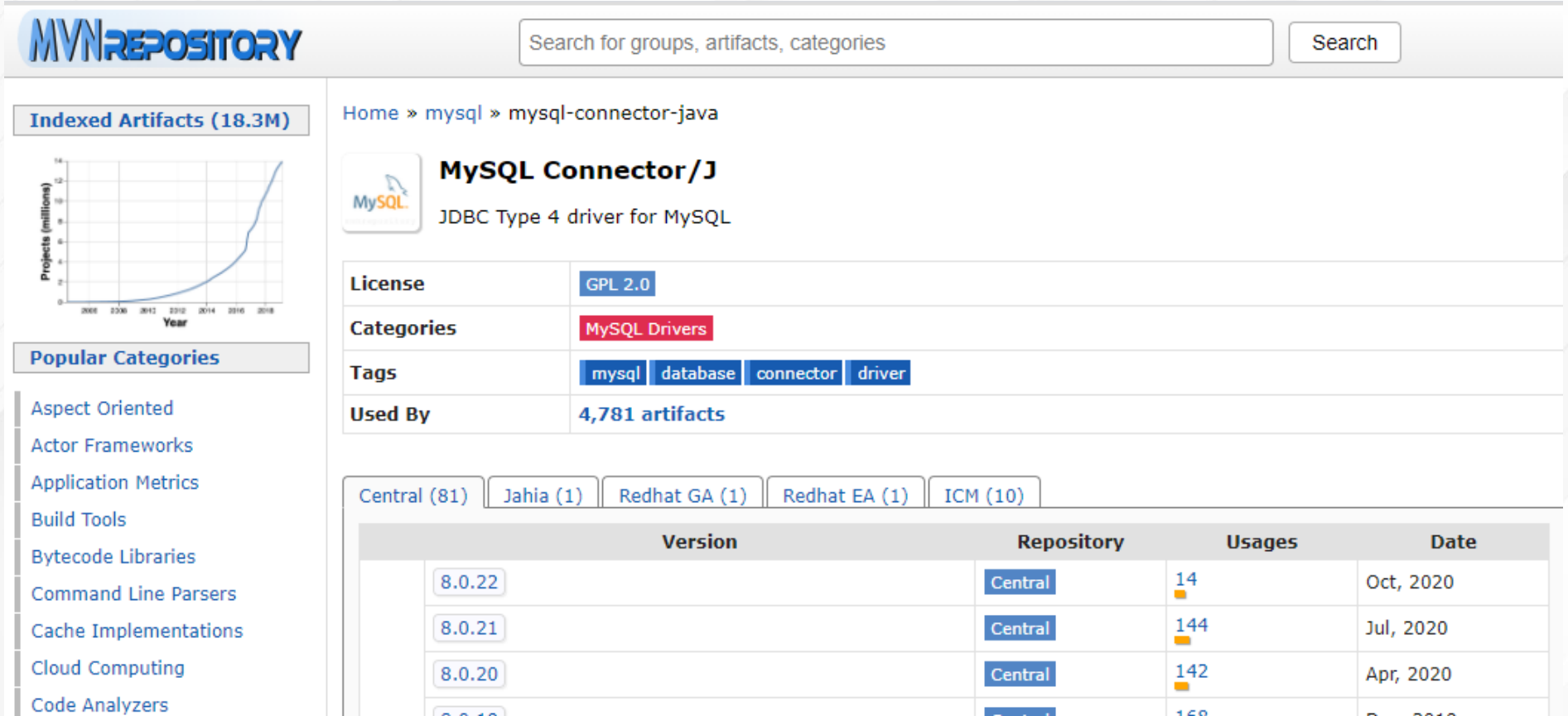
En Eclipse:



ILERNA Online

Además de Hibernate, ¿necesitamos algo más para conectarnos a la BD en MySQL?

Conector JDBC para Maven → https://mvnrepository.com/artifact/mysql/mysql-connector-java



ILERNA Online

Además d
para c

Conector JDBC

onnector-java

Home » mysql » mysql-connector-java » 8.0.22

MySQL Connector/J » 8.0.22

JDBC Type 4 driver for MySQL

License	GPL 2.0
Categories	MySQL Drivers
Organization	Oracle Corporation
HomePage	http://dev.mysql.com/doc/connector-j/en/
Date	(Oct 17, 2020)
Files	jar (2.3 MB) View All
Repositories	Central
Used By	4,781 artifacts

Maven | Gradle | SBT | Ivy | Grape | Leiningen | Buildr

```
<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.22</version>
</dependency>
```

☑ Include comment with link to declaration

ILERNA Online

Además de Hibernate, ¿necesitamos algo más para conectarnos a la BD en MySQL?

Conector JDBC

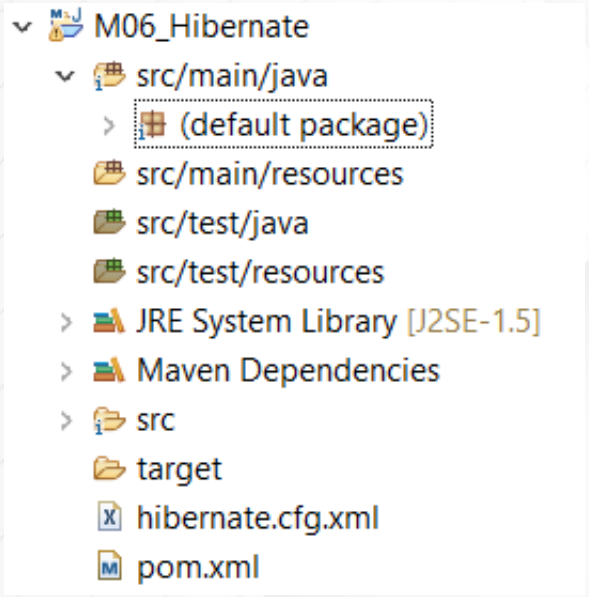
*M06_Hibernate/pom.xml

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.Ilerna.m06</groupId>
  <artifactId>M06_Hibernate</artifactId>
  <version>0.0.1-SNAPSHOT</version>

  <dependencies>

        <!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->
        <dependency>
            <groupId>org.hibernate</groupId>
            <artifactId>hibernate-core</artifactId>
            <version>5.4.23.Final</version>
        </dependency>

        <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>8.0.22</version>
        </dependency>
  </dependencies>

</project>
```

sql-connector-java

> MÁS CONFIGURACIONES

Creamos un fichero llamado: hibernate.cfg.xml y añadimos la información de configuración



M06_Hibernate/pom.xml | hibernate.cfg.xml | *Alumno.java

```
<?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="dialect">org.hibernate.dialecto.MySQL5Dialect</property>
        <property name="hbm2.ddl.auto">create</property>
        <property name="connection.url">jdbc:mysql://127.0.0.1/m06</property>
        <property name="connection.username">root</property>
        <property name="connection.password">root</property>

    </session-factory>

</hibernate-configuration>
```

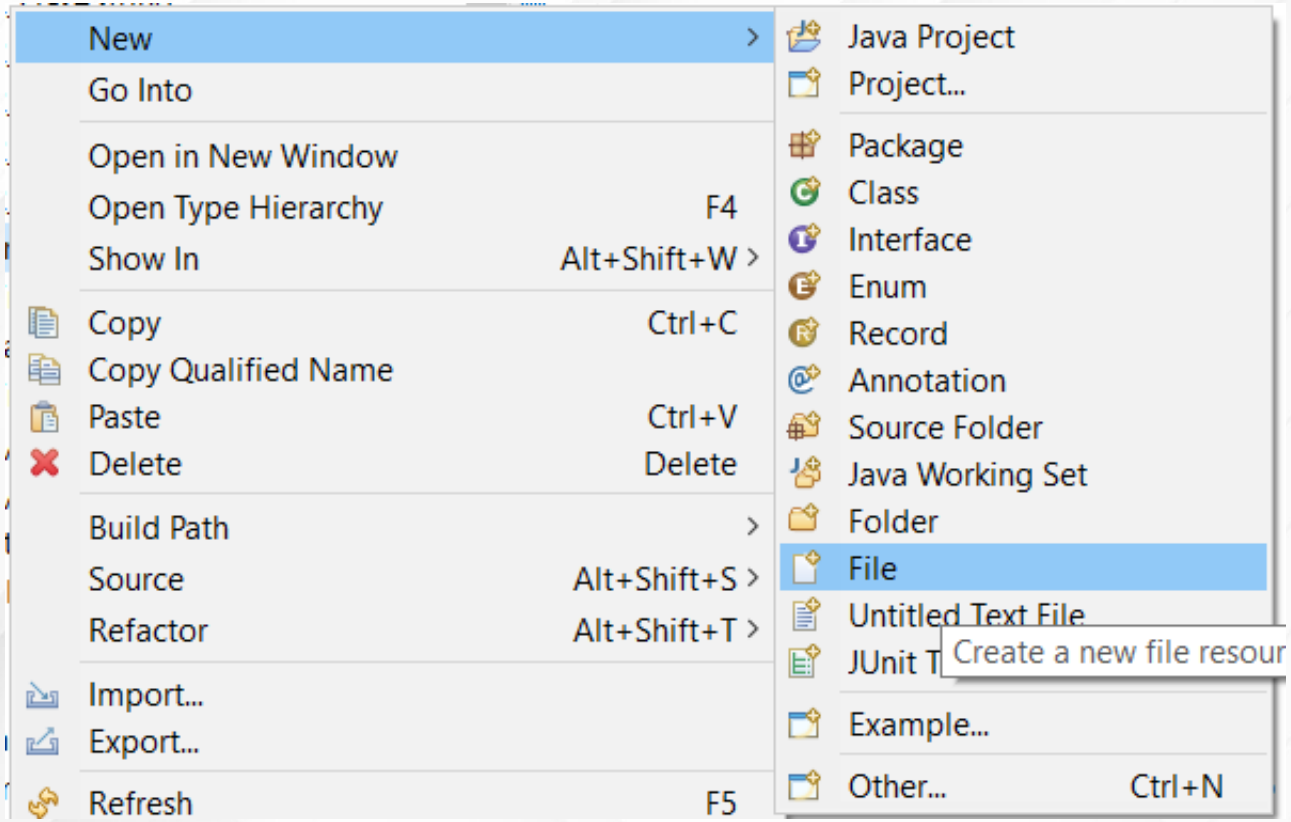
Llegados a este punto, nos faltaría definir cuál va a ser el MAPEO de las clases con las que vamos a trabajar y las tablas

En nuestro proyecto no tenemos todavía ninguna clase definida, por lo que iremos al proyecto y definiremos la CLASE ALUMNO, con sus variables y constructores, los getters, setters y toString()

```
1
2  public class Alumno {
3
4      private int idAlumno;
5      private String nombre;
6      private int edad;
7
8      public Alumno() {
9
10     }
11
12     public Alumno(int idAlumno, String nombre, int edad) {
13         super();
14         this.idAlumno = idAlumno;
15         this.nombre = nombre;
16         this.edad = edad;
17     }
18
19     public int getIdAlumno() {
20         return idAlumno;
21     }
22
23     public void setIdAlumno(int idAlumno) {
```

A partir de esta clase, vamos a crear nuestro fichero de configuración para el Mapeo de la clase Alumno en el proyecto con el que estamos trabajando

Nombre del fichero: alumno.hbm.xml



ILERNA Online

En este fichero de configuración usamos el DTD correspondiente para hacer el mapeo y agregamos la configuración para trabajar con la tabla Alumno

```xml
<?xml version='1.0' encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
        "-//Hibernate/HibernateMappingDTD5.3//EN"
        "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">

<hibernate-mapping>

    <class name="Alumno" table="Alumnos" catalog ="m06hibernate">
        <id name="idAlumno">
            <column name = "id"/>
            <generator class="increment"/>
        </id>
        <property name ="nombre"/>
        <property name="edad" type ="integer"/>

    </class>


</hibernate-mapping>
```

> MÁS CONFIGURACIONES

Añadimos este fichero alumno que acabamos de crear en el fichero de configuración de Hibernate mediante la etiqueta mapping y el atributo resource

```xml
<?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="dialect">org.hibernate.dialect.MySQL5Dialect</property>
        <property name="hbm2ddl.auto">create</property>
        <property name="connection.url">jdbc:mysql://localhost/m06hibernate</property>
        <property name="connection.username">root</property>
        <property name="connection.password"></property>

        <mapping resource = "alumno.hbm.xml"/>

    </session-factory>

</hibernate-configuration>
```

Para poder ejecutar nuestro programa y comprobar que tanto Hibernate como el proyecto funcionan correctamente, vamos a crear un método Main en el proyecto que arranque todo este proceso.

Tendremos que definir en él las interfaces que comentamos al inicio de este punto

Tenemos que tener xamp abierto y andando, con los servicios de http y de apache levantado.

```java
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.boot.registry.StandardServiceRegistryBuilder;
import org.hibernate.cfg.Configuration;

public class Main {

    public static void main(String[] args) {

        System.out.println("INICIO DEL PROGRAMA...");

        //Para indicar que queremos usar Hibernate definimos las interfaces
        Configuration cfg =new Configuration().configure();
        //única instancia de sessionFactory en nuestra sesión
        SessionFactory sessionFactory = cfg.buildSessionFactory(new StandardServiceRegistryBuilder().configure().build());
        Session session = sessionFactory.openSession();
        System.out.println("CONFIGURACIÓN REALIZADA");

        session.close();
        sessionFactory.close();
    }

}
```

Si todo ha ido bien, entre los mensajes de INFO veremos el siguiente:

```
nov. 04, 2020 1:01:42 P. M. org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl$PooledConnections
INFO: HHH000115: Hibernate connection pool size: 20 (min=1)
nov. 04, 2020 1:01:42 P. M. org.hibernate.dialect.Dialect <init>
INFO: HHH000400: Using dialect: org.hibernate.dialect.MySQL5Dialect
nov. 04, 2020 1:01:43 P. M. org.hibernate.engine.transaction.jta.platform.internal.JtaPlatformInitiator initiateService
INFO: HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
CONFIGURACIÓN REALIZADA
nov. 04, 2020 1:01:43 P. M. org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl$PoolState stop
INFO: HHH10001008: Cleaning up connection pool [jdbc:mysql://localhost/m06hibernate]
```

ILERNA Online

Para insertar datos en la BD

```
//Para insertar datos en la BD vamos a usar transactions
Transaction tx = session.beginTransaction();
Alumno a = new Alumno();
a.setNombre("Alberto");
a.setEdad(18);
session.save(a);

tx.commit(); //Hacemos el commit de la transacción y habremos insertado
             // los datos del primer registro de alumno
```

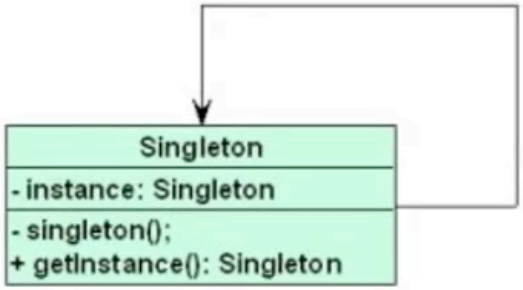
ILERNA Online

Para lanzar una Query a la BD

```
//Recueperar los datos de la BD
tx = session.beginTransaction();

Query query = session.createQuery("FROM Alumno");
List list =  query.list();
for(int i =0; i<list.size(); i++)
{
    System.out.println(list.get(i).toString());
}
```

> Singleton

Un Singleton es un patrón de diseño capaz de restringir la creación de objetos pertenecientes a una clase. Solo permite que haya una instancia por clase y un punto de acceso global.




```
Output - PatronesJava (run)
run:
El objeto ha sido creado
Ya existe el objeto
Ya existe el objeto
Ya existe el objeto
Ya existe el objeto
BUILD SUCCESSFUL (total time: 1 second)
```

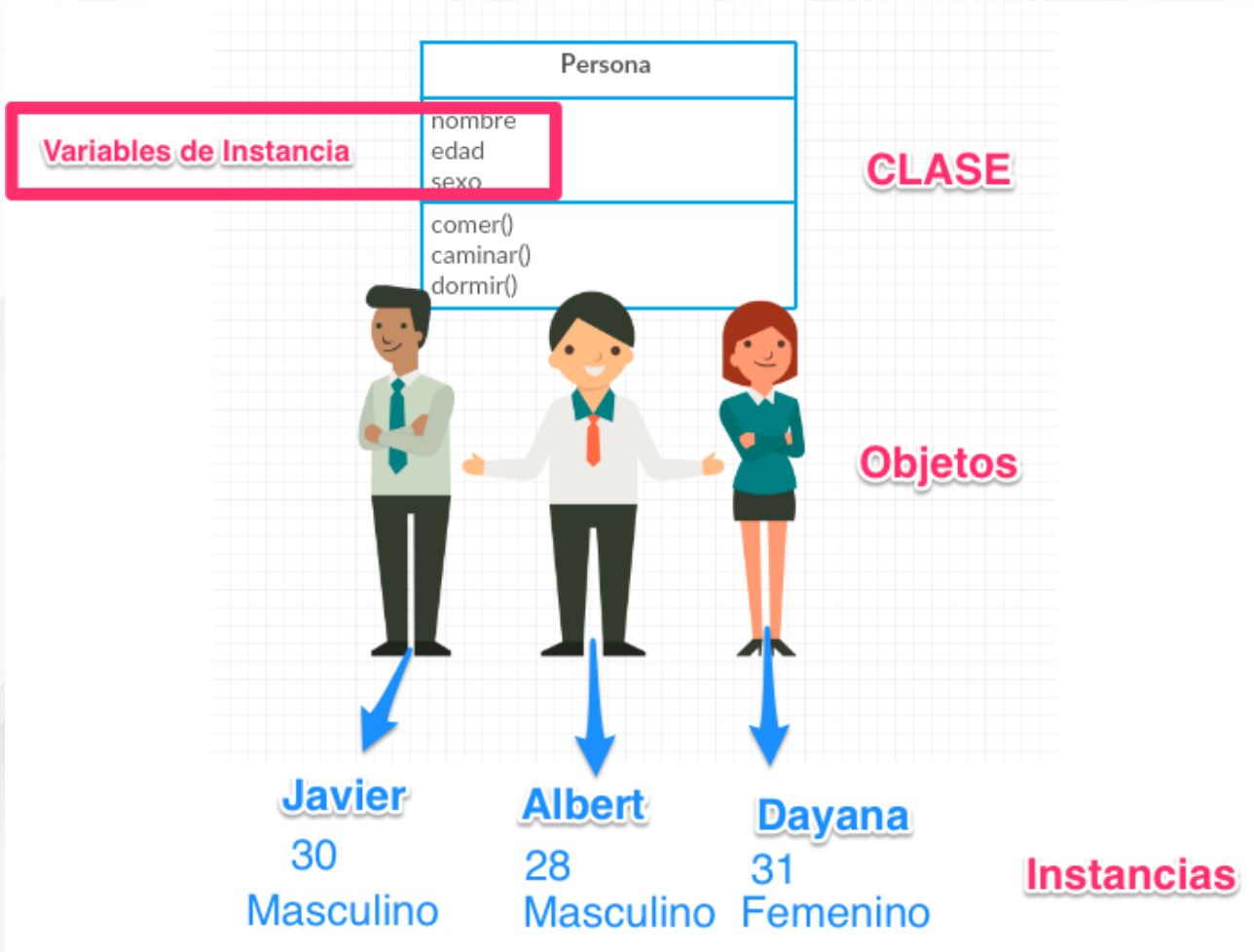
BASES DE DATOS OBJETO-RELACIONALES
BASES DE DATOS ORIENTADAS A OBJETOS

> BDOR Y BDOO

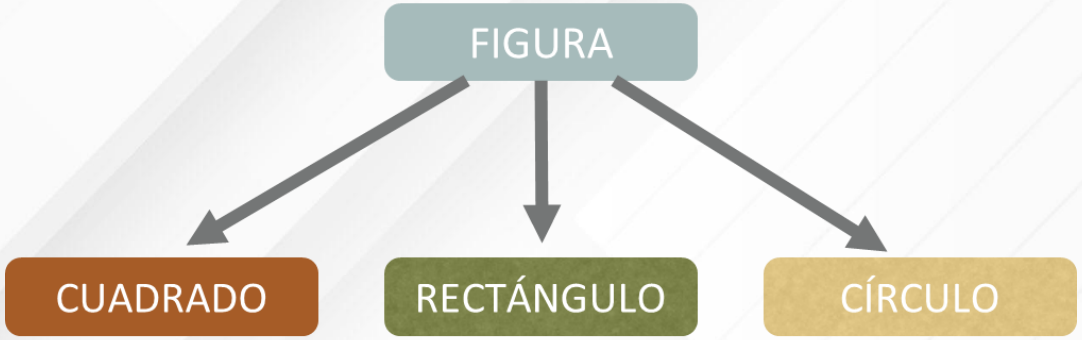
Una **base de datos objeto-relacional (BDOR)** es un tipo de BBDD que se caracteriza por haber evolucionado del modelo relacional a la orientación a objetos, convirtiéndose en una base de datos híbrida, ya que usa las dos tecnologías.

Una **base de datos orientada a objetos (BDOO)** se caracteriza por representar la información que se guarda mediante objetos, tal y como se realiza en el lenguaje de programación orientado a objetos.

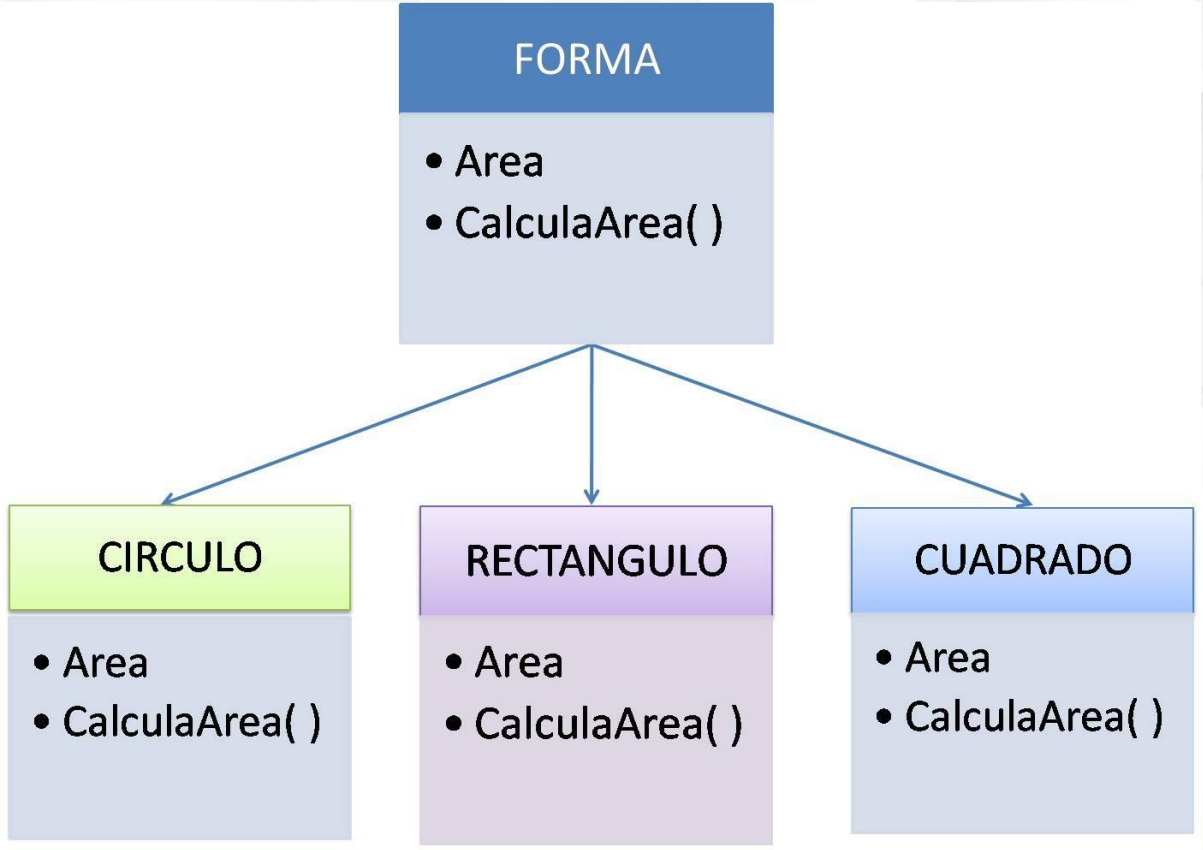
- CARACTERÍSTICAS DE LAS BD OBJETO-RELACIONALES
  - Crear objetos propios para gestionar un tipo de datos personalizado (instancia de un objeto)



- CARACTERÍSTICAS DE LAS BD OBJETO-RELACIONALES
  - Se puede disfrutar de los beneficios de la herencia (la relación entre un tipo de objeto más general y otro más específico)



- CARACTERÍSTICAS DE LAS BD OBJETO-RELACIONALES
  - De esta herencia surge la necesidad de implementar el polimorfismo y la reutilización de código
  - El **Polimorfismo** es capacidad que tiene un objeto de actuar de una forma o de otra en función de los parámetros que recibe de entrada al ser llamado



- CARACTERÍSTICAS DE LAS BD OBJETO-RELACIONALES
  - Este tipo de base de datos es la ocultación de datos de un objeto de modo que solo se permita su modificación a partir de las operaciones definidas por este objeto.
  - Este concepto se conoce como **encapsulación** y sirve para proteger los datos guardados

- El lenguaje SQL es el lenguaje característico para las operaciones con una base de datos objeto-relacional.
- Creado por IBM en 1989 y desde entonces se han ido realizando revisiones del lenguaje.

  En su primera versión, se definieron los siguientes puntos:
  - **El lenguaje de definición de datos (LDD**): se definieron las instrucciones create, alter y drop, que sirven para crear, definir o borrar bases de datos.
  - **El lenguaje de manipulación de datos (LMD):** establecieron las instrucciones de gestión de tablas: select, insert, delete, update, commit y rollback.
  - **El lenguaje de control de datos (LCD):** establecieron todas las operaciones para otorgar o revocar permisos: grant y revoke.

- Posteriormente se añadieron nuevas características:
  - Nuevos tipos de datos, como DATE, TIMESTAMP, TIME e INTERVAL, y diferentes tipos de string, como BIT, VARCHAR y NATIONAL CHARACTER.
  - Cursores.
  - Posibilidad de definir esquemas.
  - Creación de tablas temporales.
  - Transacciones.
  - Operaciones: union join, natural join.
  - Nuevas operaciones para la definición de datos: ALTER y DROP.
  - Posibilidad de controlar los privilegios de los usuarios.
  - Se aseguró que todos los estándares creados en el pasado o los que se crearan en el futuro fueran compatibles los unos con los otros

> FUNCIONALIDADES RELACIONAES (I)

- Se realizan ciertos cambios y se añaden funcionalidades al lenguaje SQL para mejorar las capacidades relacionales de este lenguaje para modelar los datos
  - Nuevos tipos de datos:
    - LOB (objeto de tamaño grande)
    - CLOB (guarda largas cadenas de carcteres)
    - BLOB (tipo de dato binario de gran tamaño)
    - Boolean: tipos de datos BIT (1 ó 0)
    - Array: guardar colección de datos en una columna de la tabla
      - DIAS VARCHAR(10) ARRAY(8)
    - Colecciones: almacenar colecciones enteras de datos (básicos y estructurados)

> FUNCIONALIDADES RELACIONAES (II)

- Se realizan ciertos cambios y se añaden funcionalidades al lenguaje SQL para mejorar las capacidades relacionales de este lenguaje para modelar los datos
  - Nuevos predicados (expresión básica de SQL)
    - DISTINCT: elimina registros repetidos de una consulta SQL
    - LIKE: Para hacer una comparación
      - select * from estudiante where apellido LIKE ‘Garcia’;
  - Cambio en semántica SQL
  - Más seguridad: con permisos y roles que tengan diferentes permisos
  - TRIGGERS: objetos de la base de datos que ejecutan acciones al producirse un evento

> USANDO OBJETOS EN SQL

ILERNA Online

- OBJETOS: Tipos de datos definidos por el usuario
- Tienen una estructura y un comportamiento común de datos de las aplicaciones
- Categorías:
  - Tipos de objetos (Object type)
  - Tipos para colecciones (Collection type)
- Representan una entidad del mundo real
- Formados por:
  - Atributos (de tipo básico o de un tipo definido por usuario)
  - Métodos: En PL/SQL o C

- SINTAXIS
  - Creación de un Tipo de Objeto

```
-- Creación de un tipo (un objeto)
CREATE OR REPLACE TYPE DIRECCION AS OBJECT
(
CALLE VARCHAR2(25), -- Tipo texto
CIUDAD VARCHAR2(20),
CODIGO_POST NUMBER(5) -- Tipo numérico
);
```

  - Declaración de un tipo que incluye otro en su interior

```
-- Creación de un tipo que incluya otro en su interior
CREATE OR REPLACE TYPE PERSONA AS OBJECT
(
CODIGO NUMBER,
NOMBRE VARCHAR2(35),
DIREC DIRECCION, -- Hacemos referencia al tipo de objeto creado anteriormente
FECHA_NAC DATE -- Tipo fecha
);
```

- SINTAXIS
  - Declaración de métodos

```
-- Cuando creamos un tipo podemos asignarle como campo un método (operación)
CREATE OR REPLACE TYPE DIRECCION AS OBJECT
(
CALLE VARCHAR2(25),
CIUDAD VARCHAR2(20),
CODIGO_POST NUMBER(5),
/* Un procedimiento realizará una operación sin retornar un resultado
   En este caso llamando a set_calle el campo calle será igual al parámetro C */
MEMBER PROCEDURE SET_CALLE(C VARCHAR2),
/* Una función realizará una operación y devolverá un resultado.
   En este caso, retornará el valor del campo calle */
MEMBER FUNCTION GET_CALLE RETURN VARCHAR2 -- Indicamos el tipo de objeto a devolver
);
```

- SINTAXIS
  - Declaración de métodos

```
-- Por ejemplo, vamos a crear el cuerpo del tipo DIRECCION
CREATE OR REPLACE TYPE BODY DIRECCION AS
-- Procedimiento que da valor a la calle
MEMBER PROCEDURE SET_CALLE(C VARCHAR2) IS
BEGIN
  CALLE:=C; -- Simplemente igualamos el campo al parámetro
END;
-- Función que devuelve el nombre de la calle
MEMBER FUNCTION GET_CALLE RETURN VARCHAR2 IS
BEGIN
  RETURN CALLE; -- Devuelvo el campo seleccionado
END;
END;
```

- SINTAXIS
  - Creación de una tabla de objetos
    - Se usan para almacenar los daots de los objetos en la base de datos

```
-- Creo una tabla basada en el objeto PERSONA
CREATE TABLE ALUMNOS OF PERSONA (
CODIGO PRIMARY KEY -- Establezco la clave primaria
);
```

    - Para insertar los datos de una PERSONA que contiene un objeto DIRECCION

```
INSERT INTO ALUMNOS VALUES (1, 'Juan Pérez', DIRECCION('C/ Los manantiales 5', 'GUADALAJARA', 19005), '18/12/1991');
INSERT INTO ALUMNOS (CODIGO, NOMBRE, DIREC, FECHA_NAC) VALUES (2, 'Julia Breña', DIRECCION('c/ Los espartales 25', 'GUADALAJARA', 19004), '18/12/1987');
```

- SINTAXIS
  - Referenciar campos de la dirección de una persona, en acciones como consultar, eliminar o actualizar datos

```
-- SELECT a todos los campos con una condicion de ciudad de Guadalajara
SELECT * FROM ALUMNOS A WHERE A.DIREC.CIUDAD='GUADALAJARA';
-- SELECT a campos en concreto de PERSONA
SELECT CODIGO, A.DIREC FROM ALUMNOS A;
-- SELECT a métodos definidos en objeto DIRECCION
SELECT NOMBRE, A.DIREC.GET_CALLE() FROM ALUMNOS A;
```

```
-- Consulta de actualización
UPDATE ALUMNOS A
  SET A.DIREC.CIUDAD=LOWER(A.DIREC.CIUDAD)
  WHERE A.DIREC.CIUDAD='GUADALAJARA';
```

```
-- Consulta de borrado
DELETE ALUMNOS A WHERE A.DIREC.CIUDAD='guadalajara';
```

ILERNA Online

- VARARRAYS
  - Son arrays de un tipo de objeto y de un tamaño definido por nosotros
  - Si se a actualiza un campo de un array con varias posiciones, debemos volver a guardar también las otras posiciones
  - Lo vemos con un ejemplo:

ILERNA Online

- VARARRAYS

```
-- Creo un array de tres posiciones de cadenas de texto con máx. nueve caracteres
CREATE TYPE TELEFONO AS VARRAY(3) OF VARCHAR2(9);
-- Hago un nuevo tipo incluyendo el varray anterior
CREATE TABLE AGENDA
(
NOMBRE VARCHAR2(15),
TELEF TELEFONO
);
-- Para realizar inserciones llamaremos al constructor del varray.
-- Hay que tener en cuenta que no hace falta pasarle tres teléfonos, pero
-- si queremos añadir un segundo telefono más tarde, deberemos volver a escribir todos
-- los anteriores
INSERT INTO AGENDA VALUES ('MANUEL', TELEFONO('656008876','927986655','639883300'));
INSERT INTO AGENDA (NOMBRE, TELEF) VALUES ('MARTA', TELEFONO('649500800'));
-- Las consultas funcionan como si llamasemos a propiedades de tipos
SELECT A.TELEF FROM AGENDA A;
SELECT A. NOMBRE, A.TELEF FROM AGENDA A;
-- Este es el problema de los varrays antes mencionado
-- Para actualizar un varray, hay que volver a escribir todos los campos que lo componen
UPDATE AGENDA SET TELEF = TELEFONO('649500800', '659222222')
WHERE NOMBRE='MARTA';
```

ILERNA Online

- VARARRAYS

```
-- Vamos a crear un cursos para sacar la tabla por pantalla
SET SERVEROUTPUT ON -- Activo la salida de texto
DECLARE
  CURSOR C1 IS SELECT * FROM AGENDA; -- Creo el cursor para la agenda
  CAD VARCHAR2(50); -- Y una cadena que concatene el texto
BEGIN
  FOR I IN C1 LOOP -- Inicio recorrido del cursor
    DBMS_OUTPUT.PUT_LINE(I.NOMBRE || ', Número de Teléfonos: ' || I.TELEF.COUNT); -- Escribo la cabecera
    CAD:='*'; -- Y un separador
    FOR J IN 1..I.TELEF.COUNT LOOP -- Bucle que recorra los telefonos que estén escritos
      CAD:= CAD || I.TELEF(J) || '*'; -- Y los concatene
    END LOOP; -- Fin de bucle
    DBMS_OUTPUT.PUT_LINE(CAD); -- Imprimo la cadena
  END LOOP; -- Fin del cursor
END; -- Fin programa
```

```
-- Procedimiento para hacer inserción pasando los campos de nombre y varray de telfs.
CREATE OR REPLACE PROCEDURE INSERTAR_AGENDA(N VARCHAR2, T TELEFONO) AS
BEGIN
  INSERT INTO AGENDA VALUES(N,T); -- Inserción normal llamando a los tipos
END;
```

```
-- Y la inserción metiendola también dentro de un cuerpo
BEGIN
  INSERTAR_AGENDA('LUIS', TELEFONO('949009977'));
  INSERTAR_AGENDA('MIGUEL', TELEFONO('949004020', '678905400'));
  COMMIT;
END;
```

- HERENCIA DE TIPOS
- La herencia de tipos funciona igual que la herencia de objetos
- Ejemplo:
  - Objeto base, por ejemplo ANIMALES
    - Característica: VIVIR
  - Clases Heredadas: los tipos VERTEBRADOS e INVERTEBRADOS, que además de sus propiedades llevarán incorporada la de VIVIR del tipo que han heredado. Y a su vez estos pueden ser campos que hereden otros subcampos.
    - MAMIFEROS heredará de VERTEBRADOS
    - Clase final: (tras varias herencias) PERRO

> USANDO OBJETOS EN SQL

- EJEMPLO CLASE PERSONA

```
-- Se define el tipo persona
CREATE OR REPLACE TYPE TIPO_PERSONA AS OBJECT(
DNI VARCHAR2(10),
NOMBRE VARCHAR2(25),
FEC_NAC DATE,
MEMBER FUNCTION EDAD RETURN NUMBER,
FINAL MEMBER FUNCTION GET_DNI RETURN VARCHAR2,
MEMBER FUNCTION GET_NOMBRE RETURN VARCHAR2,
MEMBER PROCEDURE VER_DATOS
) NOT FINAL

-- Cuerpo del tipo persona
CREATE OR REPLACE TYPE BODY TIPO_PERSONA AS
-- Función que calcula la edad
MEMBER FUNCTION EDAD RETURN NUMBER IS
  ED NUMBER;
  BEGIN
    ED:=TO_CHAR(SYSDATE,'YYYY') - TO_CHAR(FEC_NAC, 'YYYY');
    RETURN ED;
  END;
-- Devuelve el DNI
FINAL MEMBER FUNCTION GET_DNI RETURN VARCHAR2 IS
  BEGIN
    RETURN DNI;
  END;
-- Devuelve el nombre
MEMBER FUNCTION GET_NOMBRE RETURN VARCHAR2 IS
  BEGIN
    RETURN NOMBRE;
  END;
-- Imprime los datos de esa persona
MEMBER PROCEDURE VER_DATOS IS
BEGIN
  DBMS_OUTPUT.PUT_LINE(DNI || '*' || NOMBRE || '*' || EDAD());
END;
END; -- Fin Cuerpo
```

> USANDO OBJETOS EN SQL

- EJEMPLO CLASE PERSONA
- Ahora creamos la CLASE ALUMNO
  - Hereda de PERSONA

```
-- Se define el tipo alumno
-- Under tipo a heredar es la forma de indicar de que tipo
-- se recibirá la herencia
CREATE OR REPLACE TYPE TIPO_ALUMNO UNDER TIPO_PERSONA(
CURSO VARCHAR2(10),
NOTA_FINAL NUMBER,
MEMBER FUNCTION NOTA RETURN NUMBER,
-- Sobrescribimo el procedimiento VER_DATOS
-- del tipo persona. En tipo alumno cambiará
OVERRIDING MEMBER PROCEDURE VER_DATOS
);

-- Cuerpo del tipo alumno
CREATE OR REPLACE TYPE BODY TIPO_ALUMNO AS
  -- Devuelve la nota final
  MEMBER FUNCTION NOTA RETURN NUMBER IS
  BEGIN
    RETURN NOTA_FINAL;
  END;
  -- Nueva versión de ver datos
  OVERRIDING MEMBER PROCEDURE VER_DATOS IS
  BEGIN
    DBMS_OUTPUT.PUT_LINE(CURSO || '*' || NOTA_FINAL);
  END;
END;
```

> USANDO OBJETOS EN SQL

- EJEMPLO CLASE PERSONA
- Ahora creamos la CLASE ALUMNO
  - Hereda de PERSONA

```
SET SERVEROUTPUT ON; -- Activo salida por pantalla
DECLARE
  A1 TIPO_ALUMNO:=TIPO_ALUMNO(NULL,NULL,NULL,NULL,NULL);
  A2 TIPO_ALUMNO:=TIPO_ALUMNO('871234533A', 'PEDRO','12/12/1996', 'SEGUNDO', 7);
  NOM A1.NOMBRE%TYPE;
  DNI A1.DNI%TYPE;
  NOTAF A1.NOTA_FINAL%TYPE;
BEGIN
  A1.NOTA_FINAL:=8;
  A1.CURSO:='PRIMERO';
  A1.NOMBRE:='JUAN';
  A1.FEC_NAC:='20/10/1997';
  A1.VER_DATOS;
  NOM :=A2.GET_NOMBRE();
  DNI:=A2.GET_DNI();
  NOTAF:=A2.NOTA();
  A2.VER_DATOS();
  DBMS_OUTPUT.PUT_LINE(A1.EDAD());
  DBMS_OUTPUT.PUT_LINE(A2.EDAD());
END;

-- La creación de tabla e inserción de elementos no difiere de lo que ya sabiamos
CREATE TABLE TALUMNOS OF TIPO_ALUMNO (DNI PRIMARY KEY);
INSERT INTO TALUMNOS VALUES ('871234533A', 'PEDRO','12/12/1996', 'SEGUNDO', 7);
INSERT INTO TALUMNOS VALUES ('809004534B', 'MANUEL','12/12/1997', 'TERCERO', 8);
-- Y las consultas de campos tampoco
SELECT * FROM TALUMNOS;
SELECT DNI, NOMBRE, CURSO, NOTA_FINAL FROM TALUMNOS;
SELECT P.GET_DNI(), P.GET_NOMBRE(), P.EDAD(), P.NOTA() FROM TALUMNOS P;
```

BD XML

Tipo	Descripción	Ejemplo
Raíz	Raíz del árbol	/
Elemento	Cualquier elemento	<alumno></alumno>
Texto	Caracteres entre las etiquetas	Alvaro
Atributo	Propiedades añadidas a los elementos	modulo="mo06"
Comentario	Texto no procesado	<!– Comentario -->
Espacio de	para proporcionar elementos y atributos con	Xmlns="www.prueb

> Documento XML

```
1  <?xml version='1.0' encoding='UTF-8' ?>
2  <!DOCTYPE hibernate-configuration PUBLIC
3          "-//Hibernate/HibernateConfigurationDTD//EN"
4          "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
5  <hibernate-configuration>
6      <session-factory>
7          <property name="dialect">org.hibernate.dialect.MySQL5Dialect</property>
8          <property name="hbm2ddl.auto">create</property>
9          <property name="connection.url">jdbc:mysql://127.0.0.1:3360/m06</property>
10         <property name="connection.username">root</property>
11         <property name="connection.password">secret</property>
12         <mapping resource="alumno.hbm.xml"/>
13     </session-factory>
14 </hibernate-configuration>
```

Instrucciones

Espacios de Nombre

Etiquetas

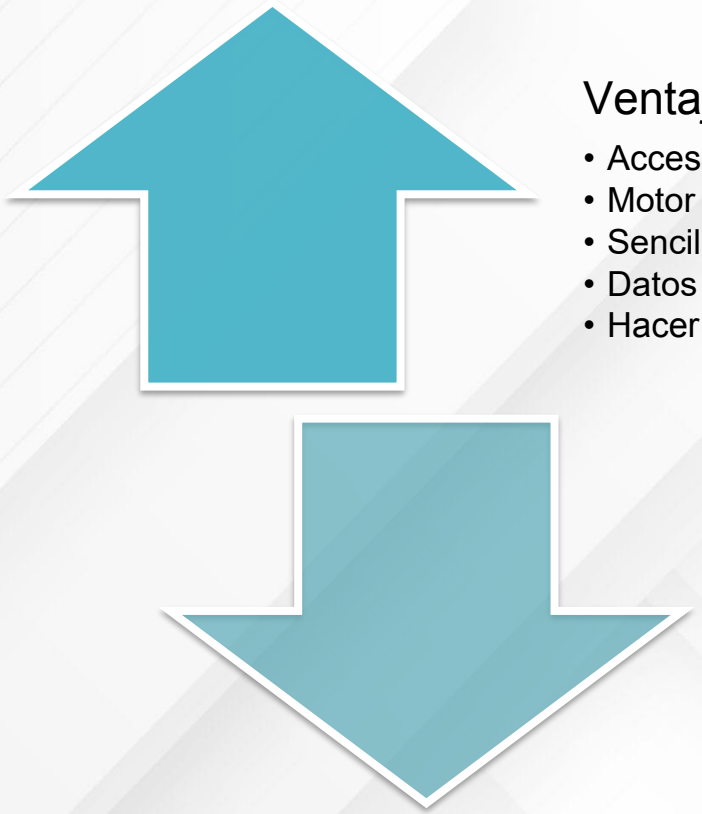
Atributos

Texto

> Documento XML

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Ilerna>
    <Ciclos>
        <Ciclo abreviacion="DAM">
            <Nombre>Desarrollo de Aplicaciones Multiplataforma</Nombre>
            <Modulos>
                <M01>
                    <Alumnos>
                        <Alumno>
                            <nombre>Pablo Garcia</nombre>
                        </Alumno>
                        <Alumno>
                            <nombre>David Perez</nombre>
                        </Alumno>
                        ...
                    </Alumnos>
                </M01>
                ...
            </Modulos>
        </Ciclo>
    </Ciclos>
</Ilerna>
```

Las bases de datos basadas en XML se diferencian del resto ya que su unidad minina será el mismo documento XML. Los cuales XML no han de almacenarse necesariamente en formato de texto



Ventajas

- Acceso y almacenamiento en formato XML
- Motor de búsqueda de alto rendimiento.
- Sencillez al añadir nuevos documentos XML.
- Datos heterogéneos.
- Hacer b´´usquedas muy rápidas

Inconvenientes:

- Complicación al indexar documentos para realizar búsquedas (datos heterogéneos).
- No ofrece funciones de agregaciones. (en SQL tenemos SUM, COUNT, AVG)
- Almacenamiento como documento o nodo; resulta complicado formar nuevas estructuras.

Son bases de datos, y como tales soportan transacciones, acceso multi-usuario, lenguajes de consulta, etc.

Están diseñadas especialmente para almacenar documentos XML

ILERNA Online

eXist

eXist es un SGDB libre de código abierto en el que se almacenan datos XML con un motor de BD, escrito en Java que es capaz de soportar los estándares de consulta:

- Xpath → lenguaje basado en rutas de XML y se utiliza para navegar de forma jerárquica en un XML, considera un documento XML como un árbol de nodos

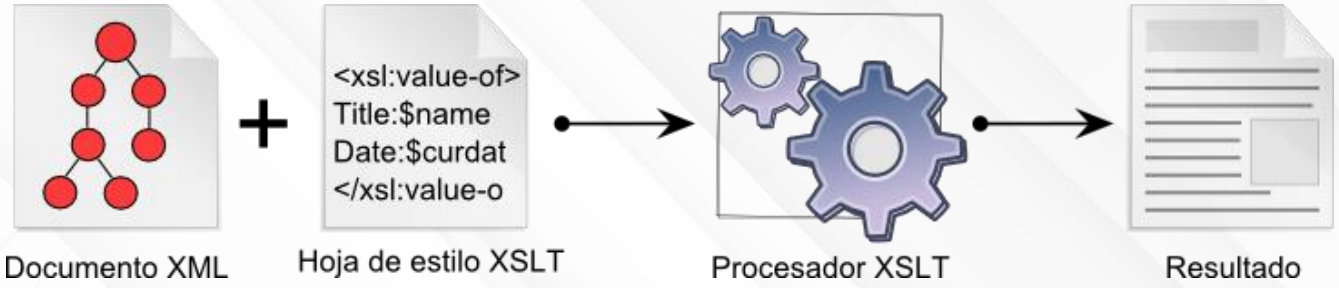
- Xquery → Lenguaje de consulta de documentos XML,

- XSLT → Transformaciones XSL (de XML a otro diferente)

eXist almacena los documentos en colecciones y además estas podrán estar anidadas.
Cada colección sería una carpeta y los documentos XML dentro de ellas.

ILERNA Online

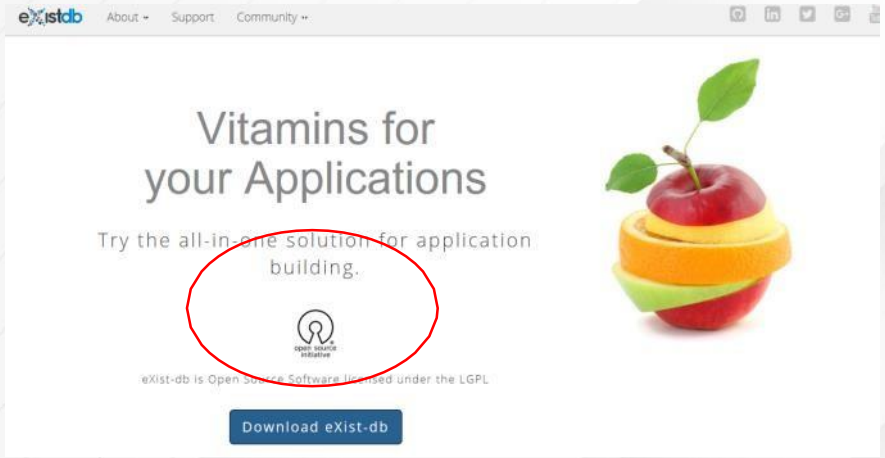
eXist

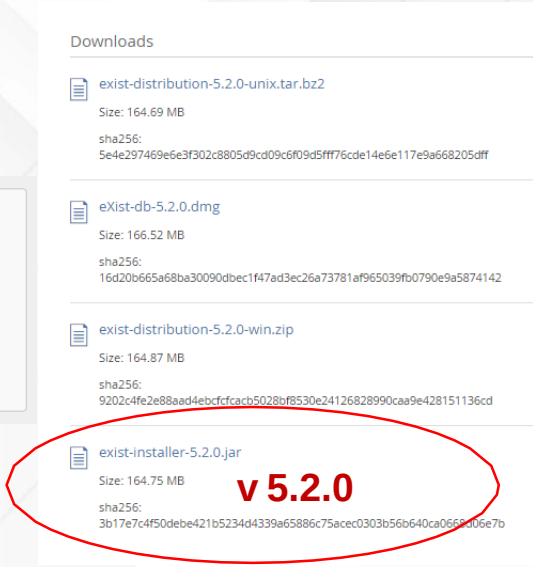


- El lenguaje XSLT describe un lenguaje basado en XML para transformar documentos XML a cualquier otro formato
- Usaremos XSLT para transformar documentos entre esquemas XML que permitan su procesamiento por distintos sistemas.
- También utilizaremos XSLT para transformar documentos XML en HTML, WML, o cualquier otro formato que facilite su presentación en la pantalla de un ordenador o en impresora.
- La transformación de XML a HTML es el principal uso que se hace de XSLT

> Instalación eXist

Para descargar la ultima versión de la base de datos: http://exist-db.org/exist/apps/homepage/index.html

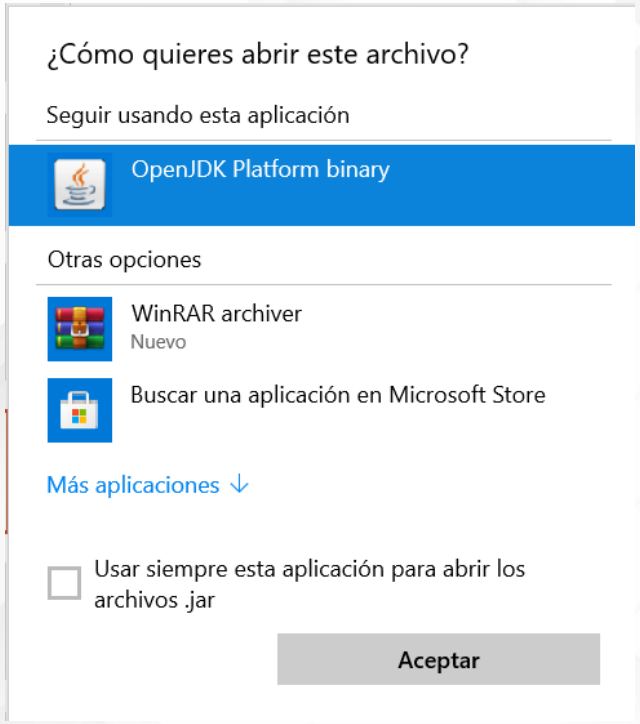




Y por ultimo ejecutamos el archivo .jar y siguiendo el asistente. Atención del directorio de instalación y el password del administrador (recomendable poner **admin**)

> Instalación eXist

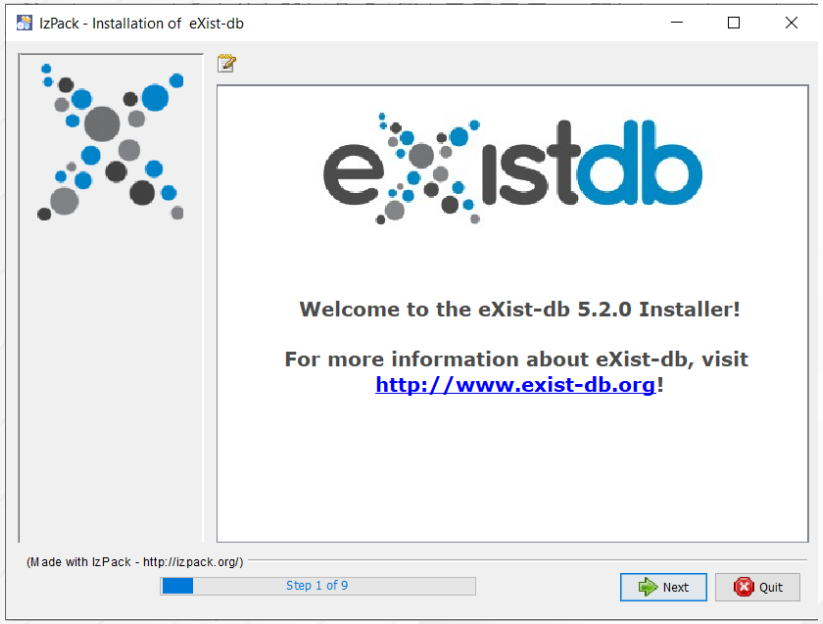
Para descargar la ultima versión de la base de datos: http://exist-db.org/exist/apps/homepage/index.html



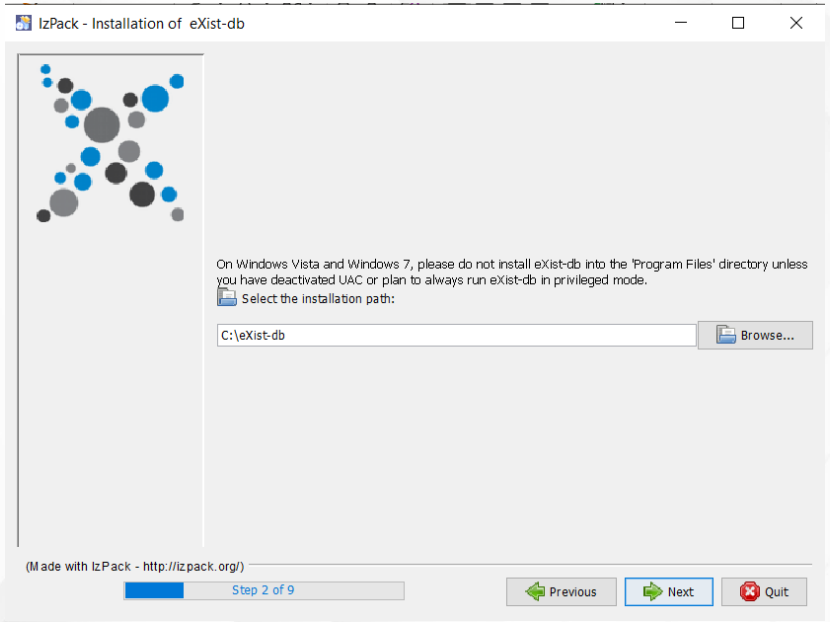
Si nos aparece este aviso,elegimos la opción OpenJDK Platform binary y aceptamos

Y por ultimo ejecutamos el archivo .jar y siguiendo el asistente. Atención del directorio de instalación y el password del administrador (recomendable poner **admin**)

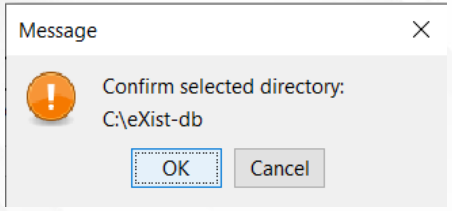
> Instalación eXist

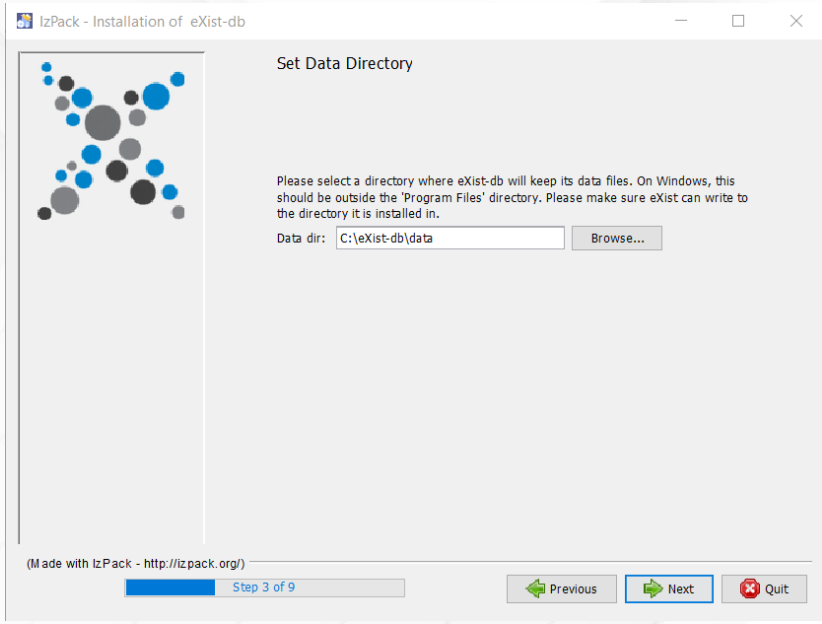


Pantalla de bienvenida. Pulsamos Next



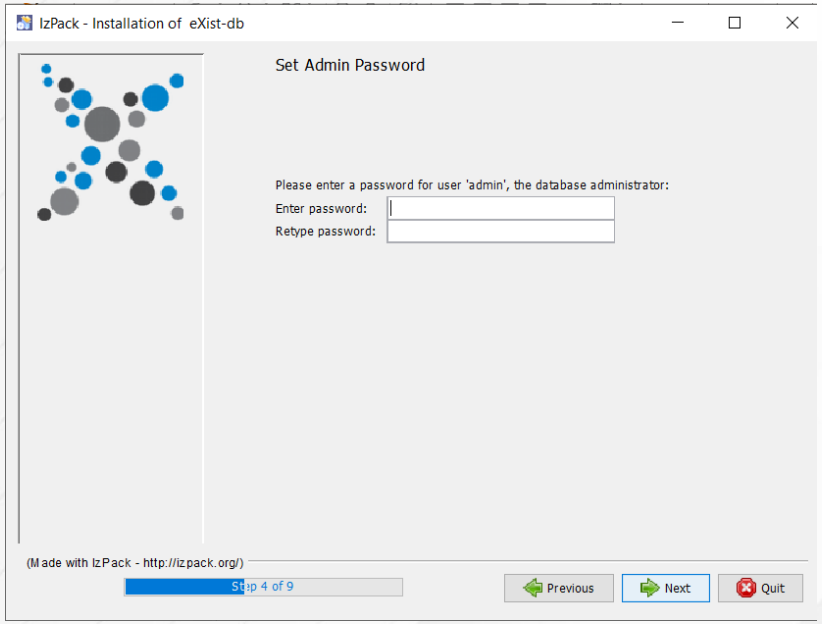
Indicar la ruta donde queremos instalar eXist.
Dejamos c:\exist-db
Confirmamos y pulsamos Next



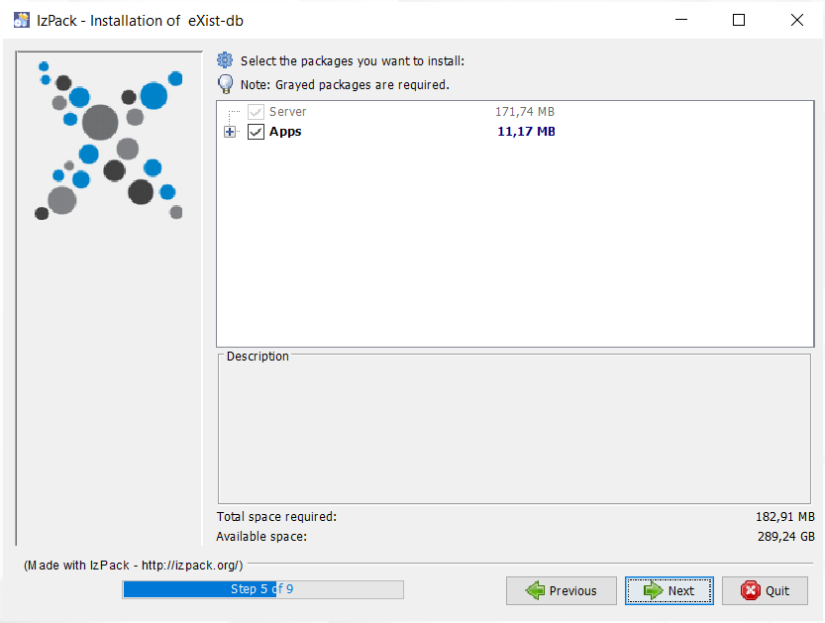


Indicar la ruta donde queremos guardar los datos de eXist
Dejamos la ruta C:\eXist-db\data
Confirmamos y pulsamos Next

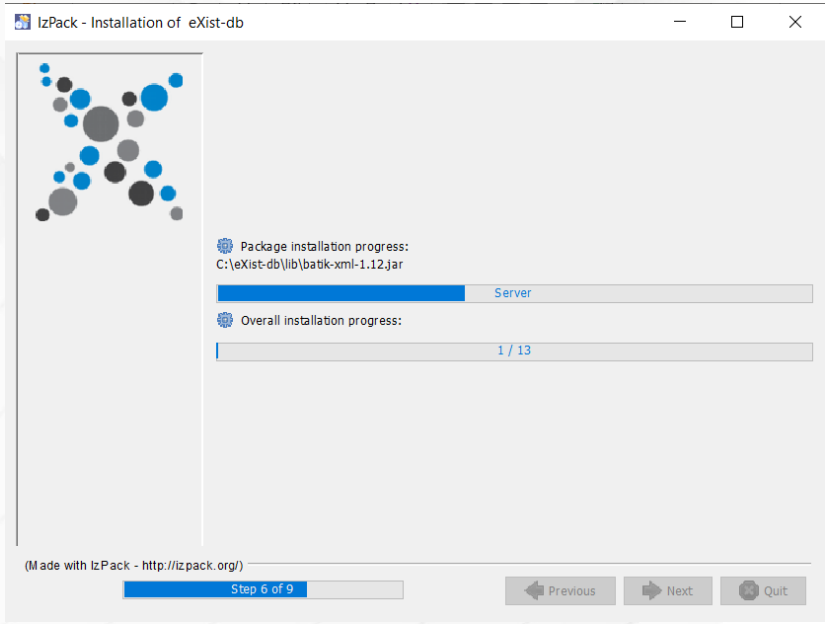
> Instalación eXist



Nos pide un usuario y una contraseña para el administrador de eXist
Usuario: admin
Contraseña: root (por ejemplo)
.Pulsamos Next

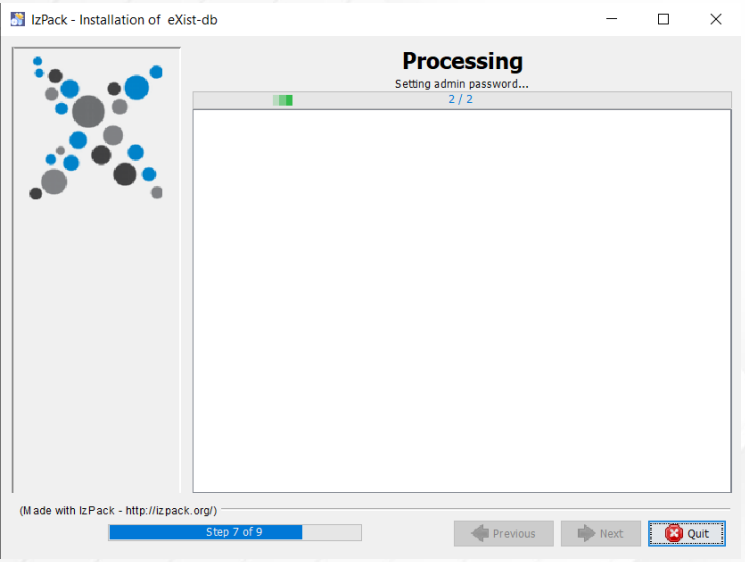


Elegimos lo que queremos instalar, en nuestro caso, el servidor y las aplicaciones.
Pulsamos en Next con todo marcado

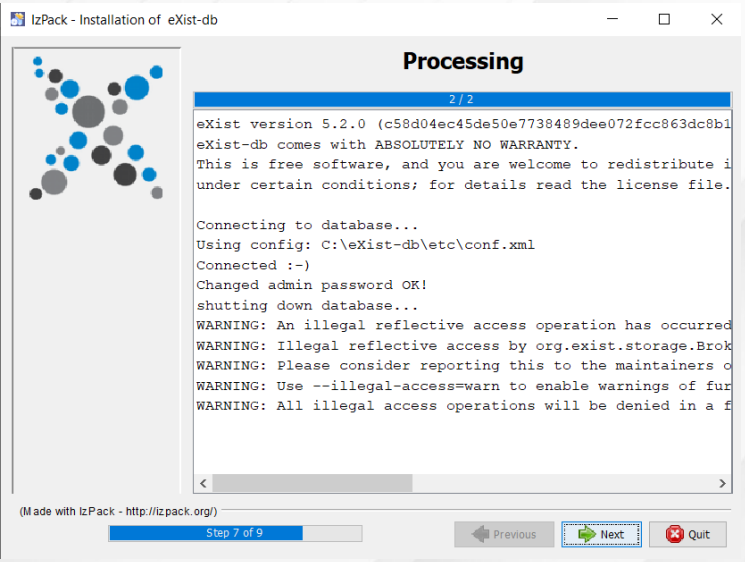


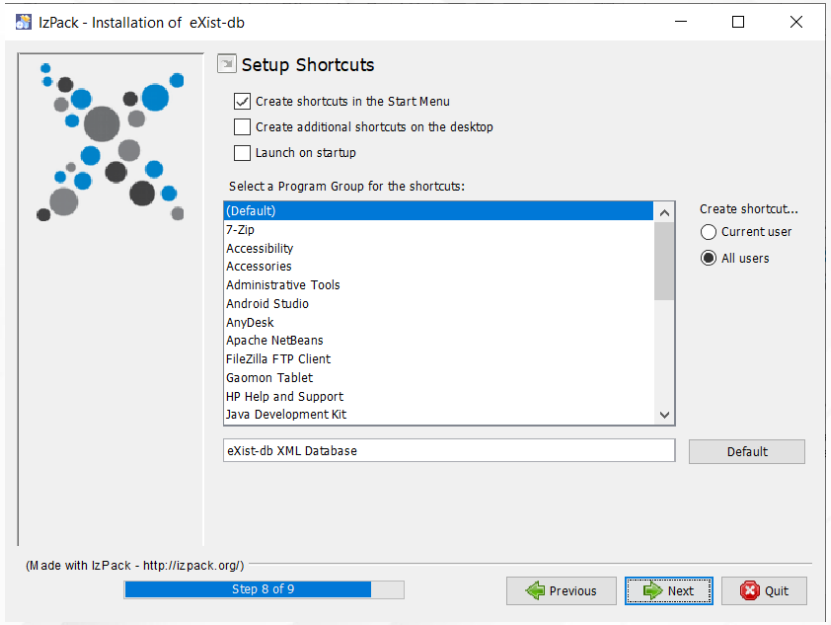
Hace la instalación del servidor de BD nativo
Esperamos a que termine la instalación y pulsamos Next

> Instalación eXist

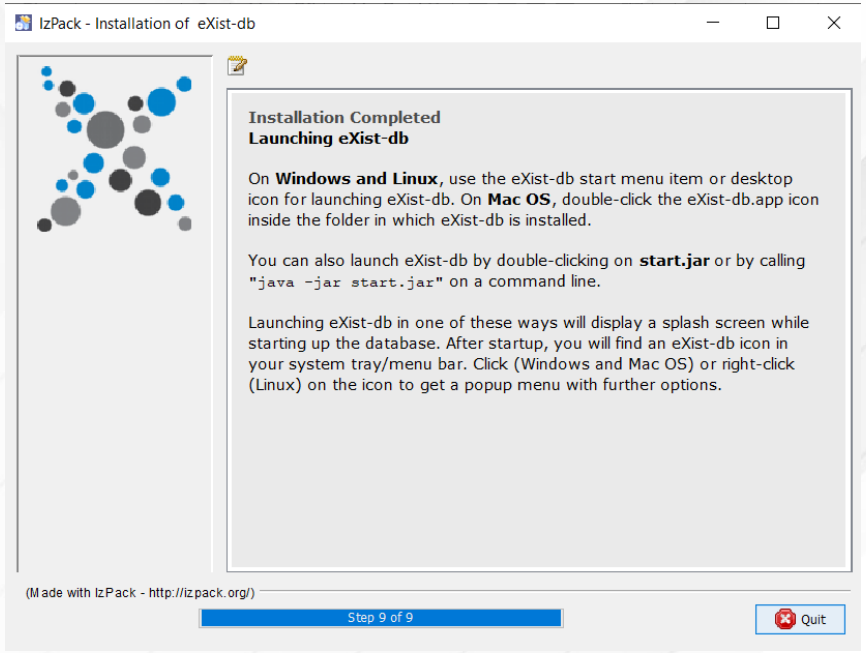


Procesa la configuración, esperamos unos segundos. Pulsamos Next





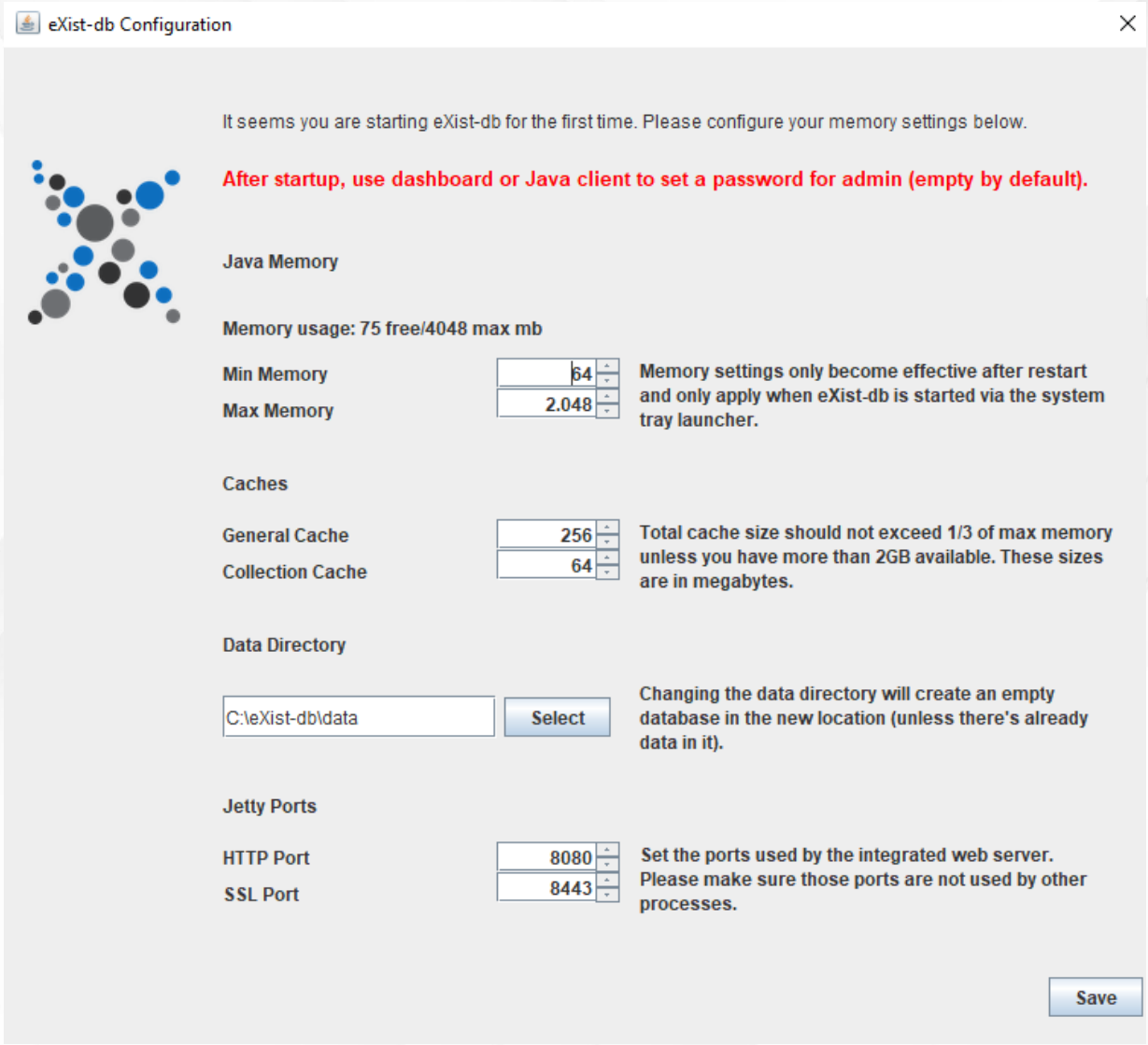
Configuramos aspectos como accesos directos en el Escritorio, etc.
Pulsamos Next

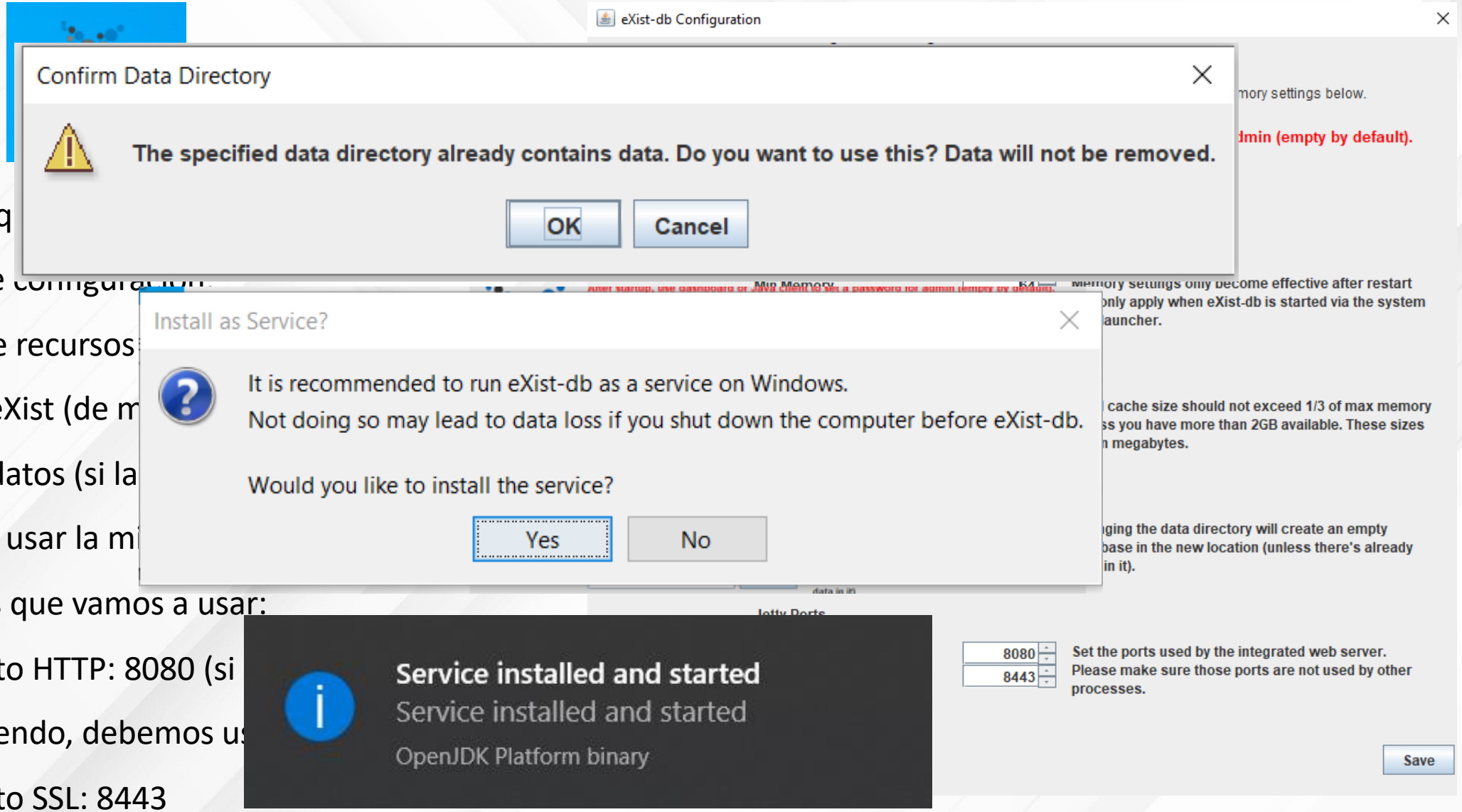


Instalación Finalizada
Pulsamos Quit

La primera vez que iniciamos eXist, nos aparecerá esta ventana de configuración:

- Cantidad de recursos que queremos que se dedique a eXist (de memoria y de caché)
- La ruta de datos (si la cambiamos en la instalación, usar la misma que pusimos
- Los puertos que vamos a usar:
    - Puerto HTTP: 8080 (si hay otro servidor corriendo, debemos usar otro)
    - Puerto SSL: 8443





La primera vez q

esta ventana de

- Cantidad de recursos

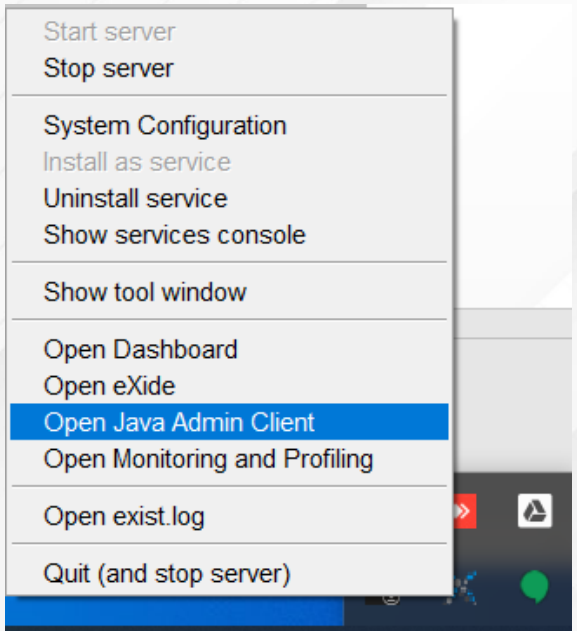
  dedique a eXist (de m
- La ruta de datos (si la

  instalación, usar la mi
- Los puertos que vamos a usar:
  - Puerto HTTP: 8080 (si

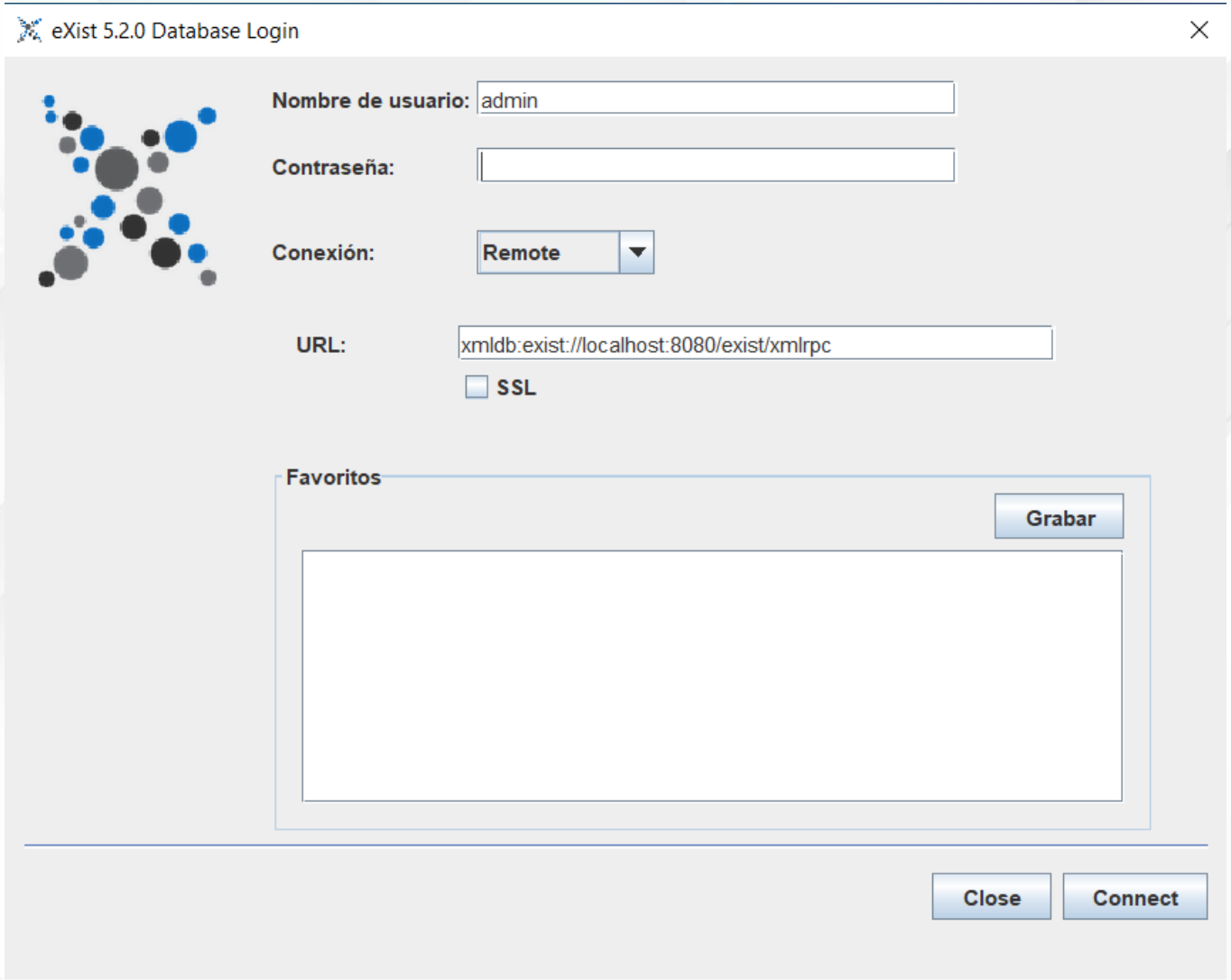
    corriendo, debemos u
  - Puerto SSL: 8443

Una vez instalado, en el área de notificaciones tendremos un icono de eXist.

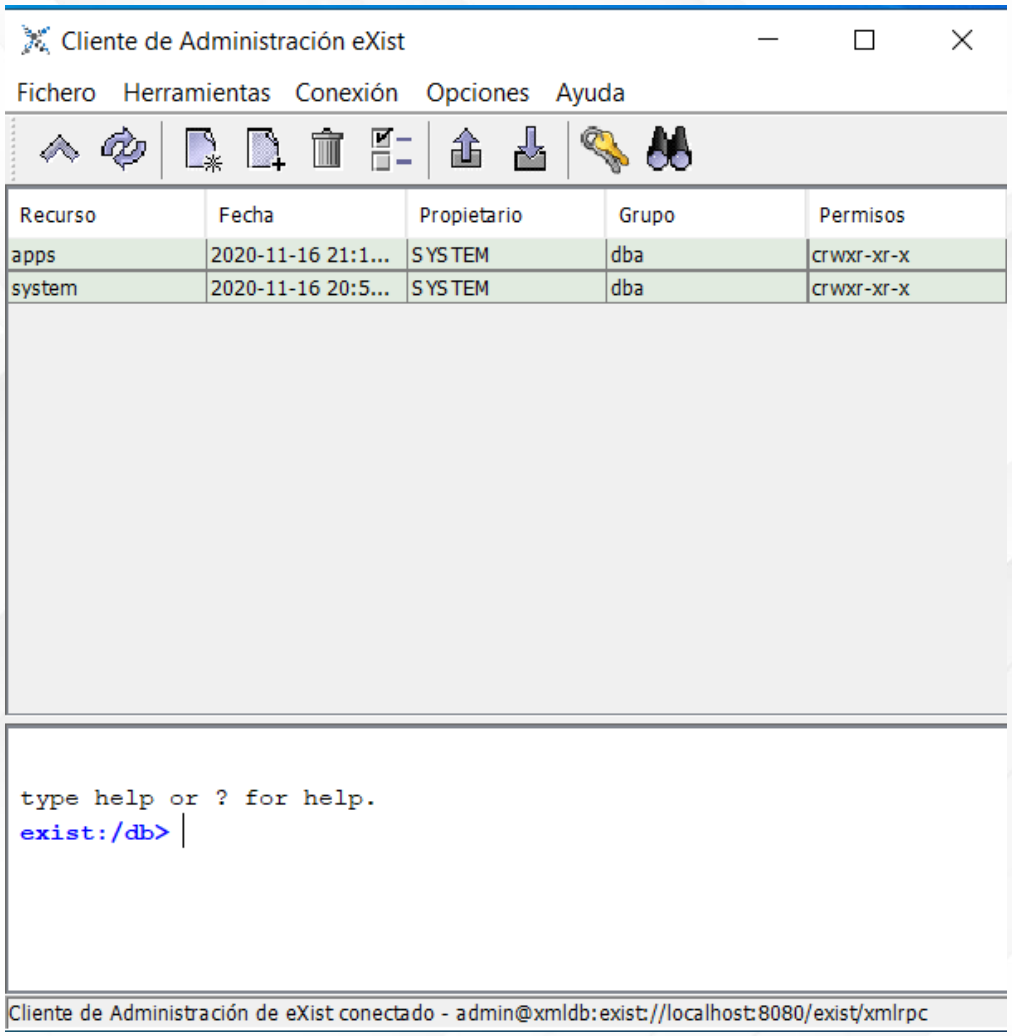
Para empezar a trabajar, haremos clic con el botón derecho sobre él y seleccionaremos Open Java Admin Client



Y se abrirá esta ventana. Introducimos la contraseña y pincharemos en Connect



Así se nos abre el cliente de administración para eXist.

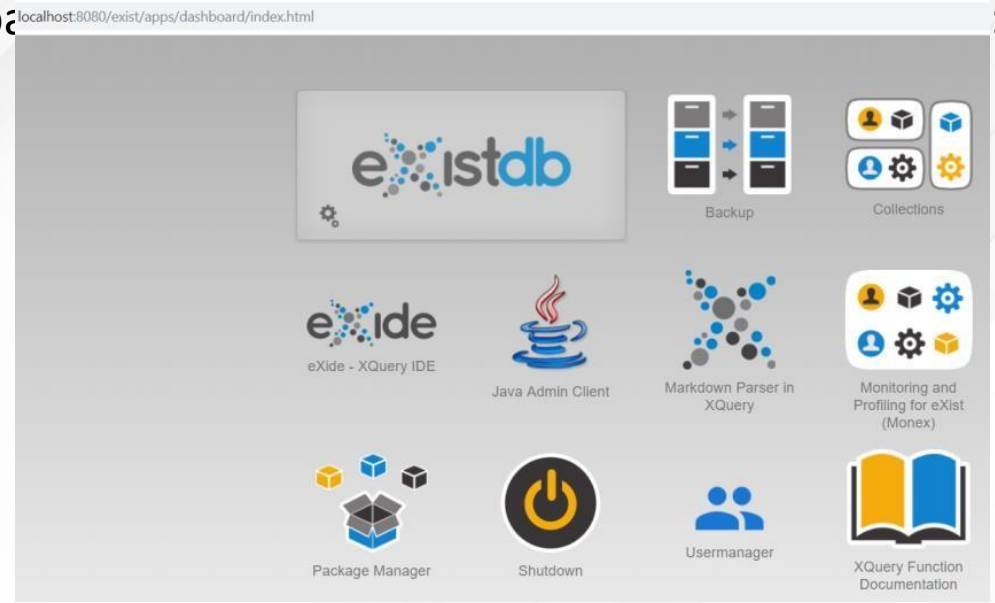


También tenemos otras formas de abrirlo, como es el caso de haber arrancado la base de datos. Entonces se podrá iniciar directamente desde el navegador escribiendo en la barra de direcciones:

http://localhost:8080/exist/

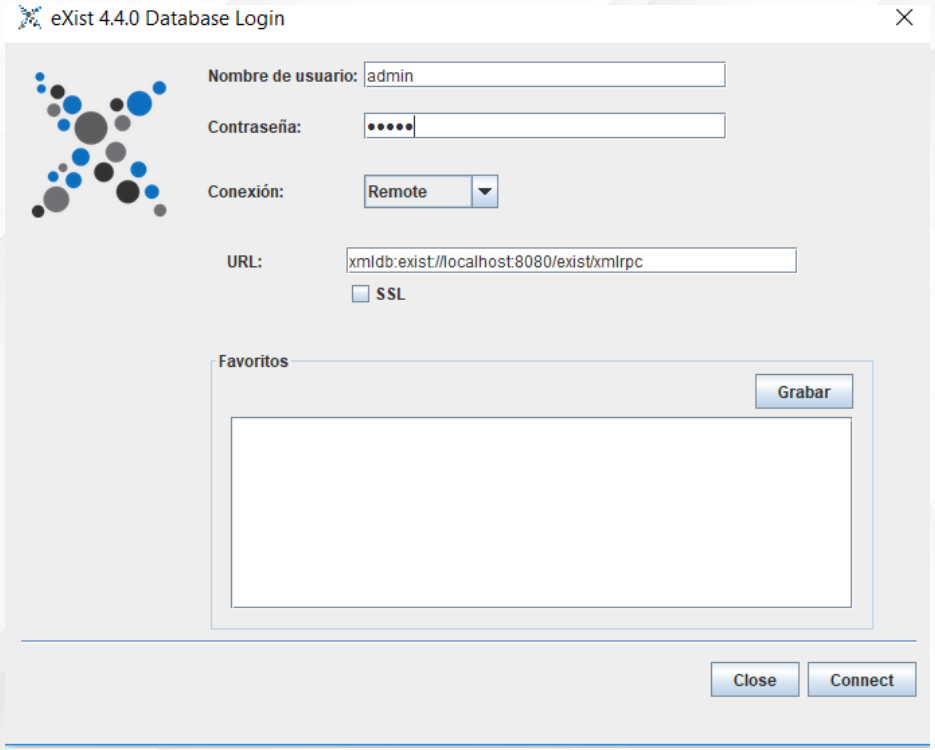
A la hora de realizar consultas y de trabajar con las ba:

- **Desde el eXide (Open eXide)**
- **Desde el *dashboard* (Open dashboard)**
- **Desde el cliente java (Open Java Admin Client)**



Primeros pasos con eXist

- **Desde el cliente java (Open Java Admin Client)**

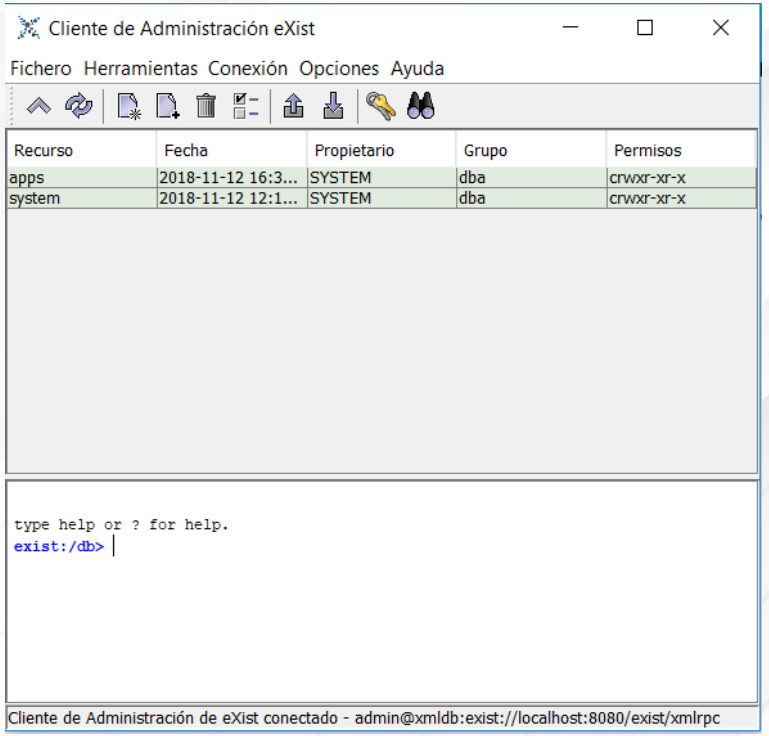


El cliente es la herramienta que utilizaremos a lo largo del tema para hacer consultas. El cliente nos pedirá la conexión, con el usuario y la contraseña y hay que asegurarse de poner bien la URL

xmldb:exist://localhost:8080/exist/xmlrpc

> El cliente de administración de eXist

Una vez conectado el usuario se muestra la ventana del Cliente de Administración eXist, donde se podrán realizar todo tipo de operaciones sobre la BD.

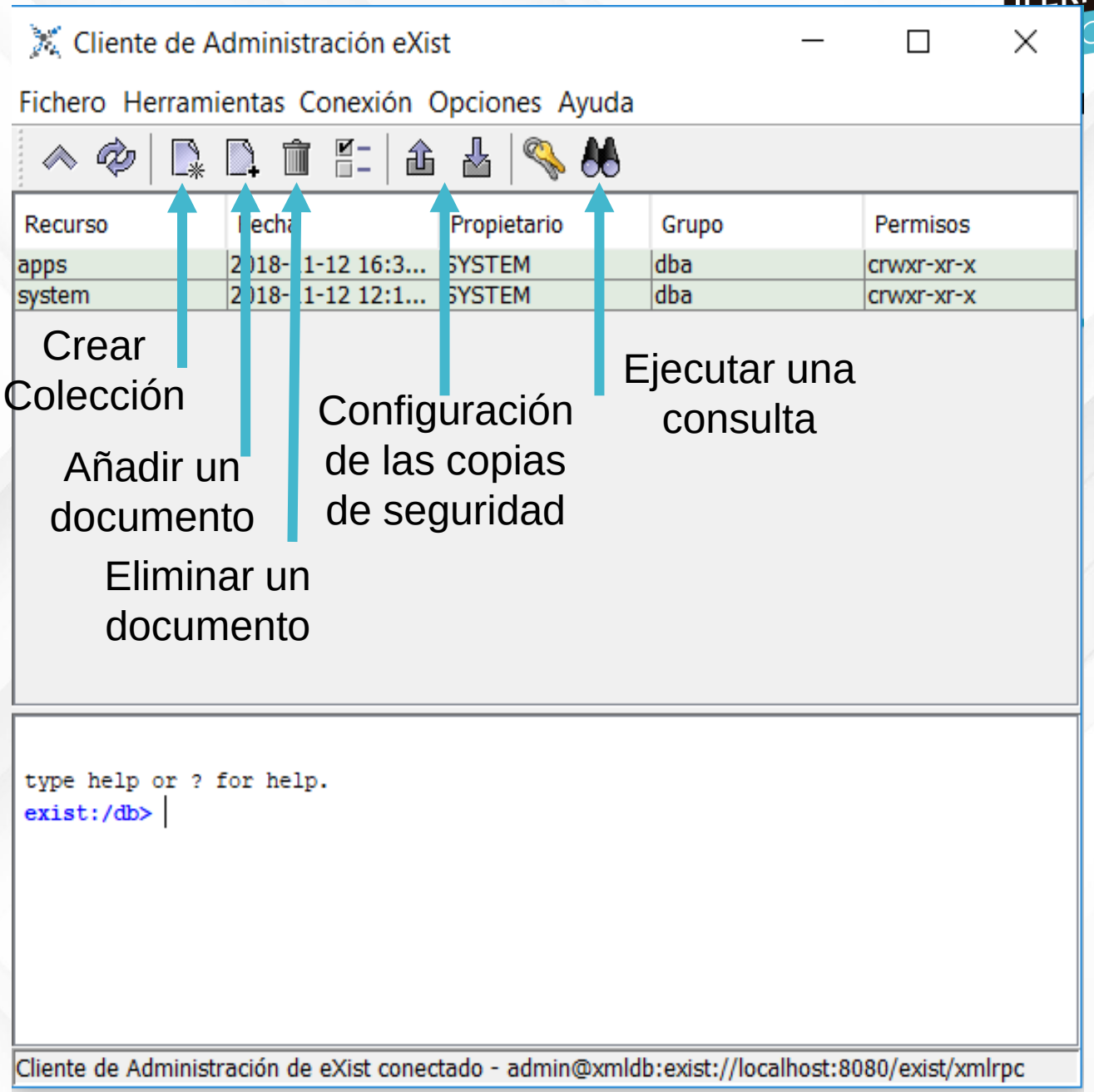


1. Creación y borrado de colecciones.
2. Incluir y eliminar documentos a las colecciones.
3. Modificar documentos.
4. Creación de copias de seguridad y restauración de las mismas.
5. Administración de usuarios.
6. Realización de consultas Xpath.
7. Navegar por colecciones y escoger un contexto a través del cual se ejecutarán las consultas.

Si se hace doble clic en un documento este se abrirá, y también se podrán realizar cambios en ellos.

Al pulsar sobre el icono de los prismáticos se abrirá una ventana llamada ***Diálogo de consulta***, desde la cual se pueden realizar consultas en la parte superior y, tras pulsar en el botón ***Ejecutar***, aparecerá el resultado de dicha consulta en la parte inferior. Las consultas realizadas pueden ser guardadas en archivos externos.

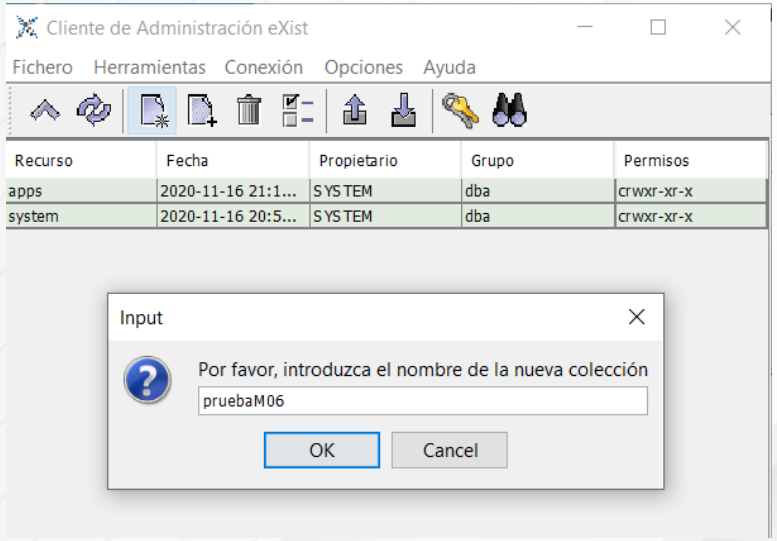
> El cliente de administración de eXist



>

Para crear una colección nueva (una BD):

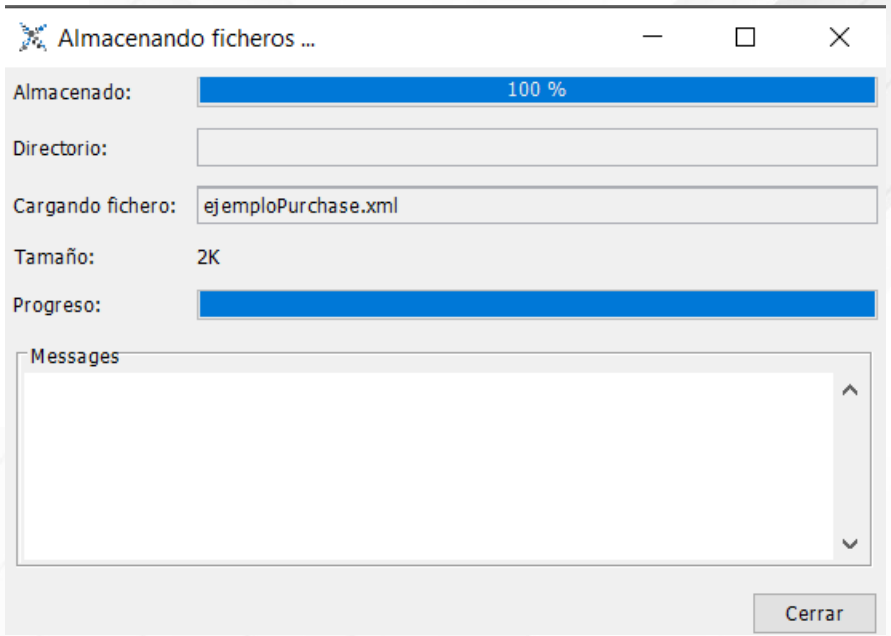
- Clic en icono con *
- Le damos un nombre a la nueva colección



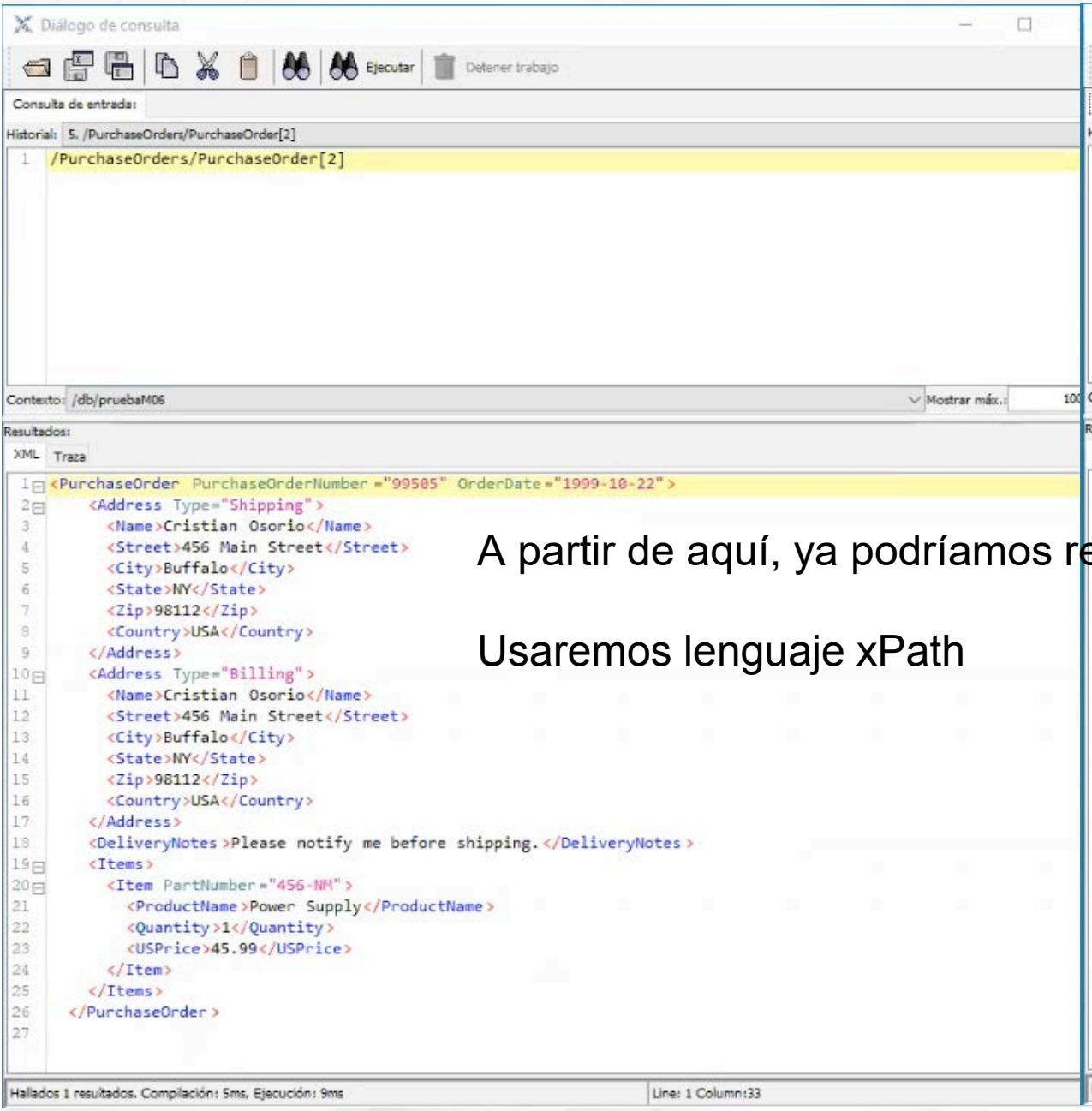
Guardamos un nuevo fichero en esta colección: ejemploPurchase.xml

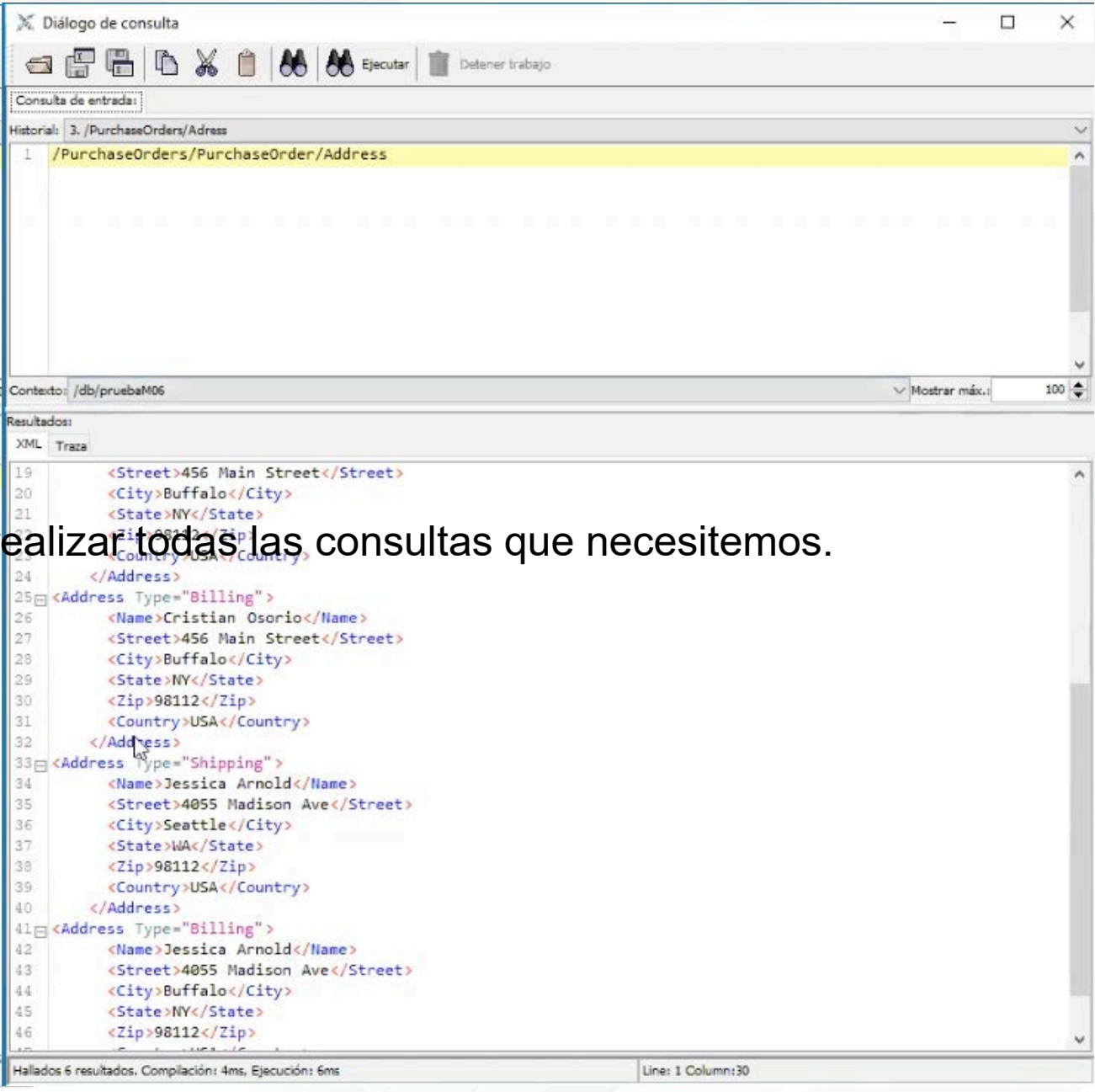
```xml
<?xml version="1.0"?>
<PurchaseOrder PurchaseOrderNumber="99503" OrderDate="1999-10-20">
  <Address Type="Shipping">
    <Name>Ellen Adams</Name>
    <Street>123 Maple Street</Street>
    <City>Mill Valley</City>
    <State>CA</State>
    <Zip>10999</Zip>
    <Country>USA</Country>
  </Address>
  <Address Type="Billing">
    <Name>Tai Yee</Name>
    <Street>8 Oak Avenue</Street>
    <City>Old Town</City>
    <State>PA</State>
    <Zip>95819</Zip>
    <Country>USA</Country>
```

Para agregarlo, pinchamos en el icono con el + y seleccionamos el archivo xml



Y así habremos importado la información





A partir de aquí, ya podríamos realizar todas las consultas que necesitemos.

Usaremos lenguaje xPath

> Colecciones y documentos

En una colección se engloban un conjunto de documentos **XML** que forman una estructura en árbol. En esta estructura en árbol cada recurso o documento pertenece a una colección. Las colecciones puedes ser vistas como carpetas en las cuales se almacenan recursos.

La estructura en árbol permite establecer caminos que referencien los recursos a los que se quiere acceder. Para acceder a los datos de estos recursos se trabajarán con dos tipos de estándares que permiten acceder y obtener datos desde documentos XML: **XPath** y **XQuery**.

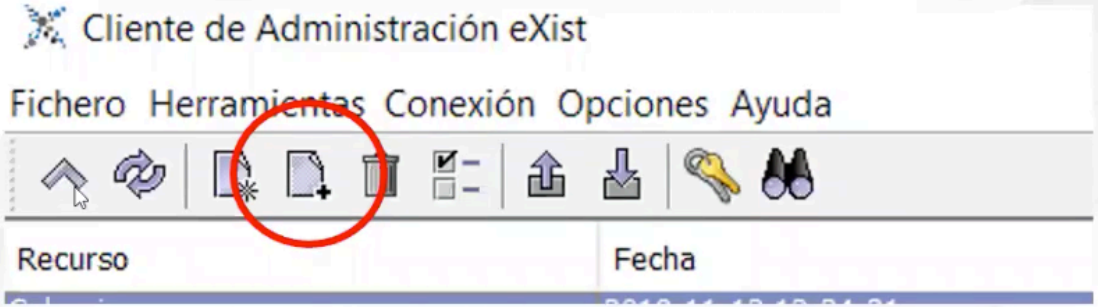
XPath es un lenguaje basado en rutas de **XML** y se utiliza para navegar de forma jerárquica en un XML.

XQuery es a Xml lo mismo que Sql es a las bases de datos relacionales. Es un lenguaje de consulta de documentos XML. Abarca tanto ficheros XML como base de datos relacionales con funciones para obtener XML a partir de registros. XQuery contiene a XPath.

XQuery tiene almacenada en sí misma a **XPath** por lo que cualquier consulta en XPath es válida también en XQuery; adicionalmente, XQuery permite mayor funcionalidad.

> Subir una colección a la base de datos

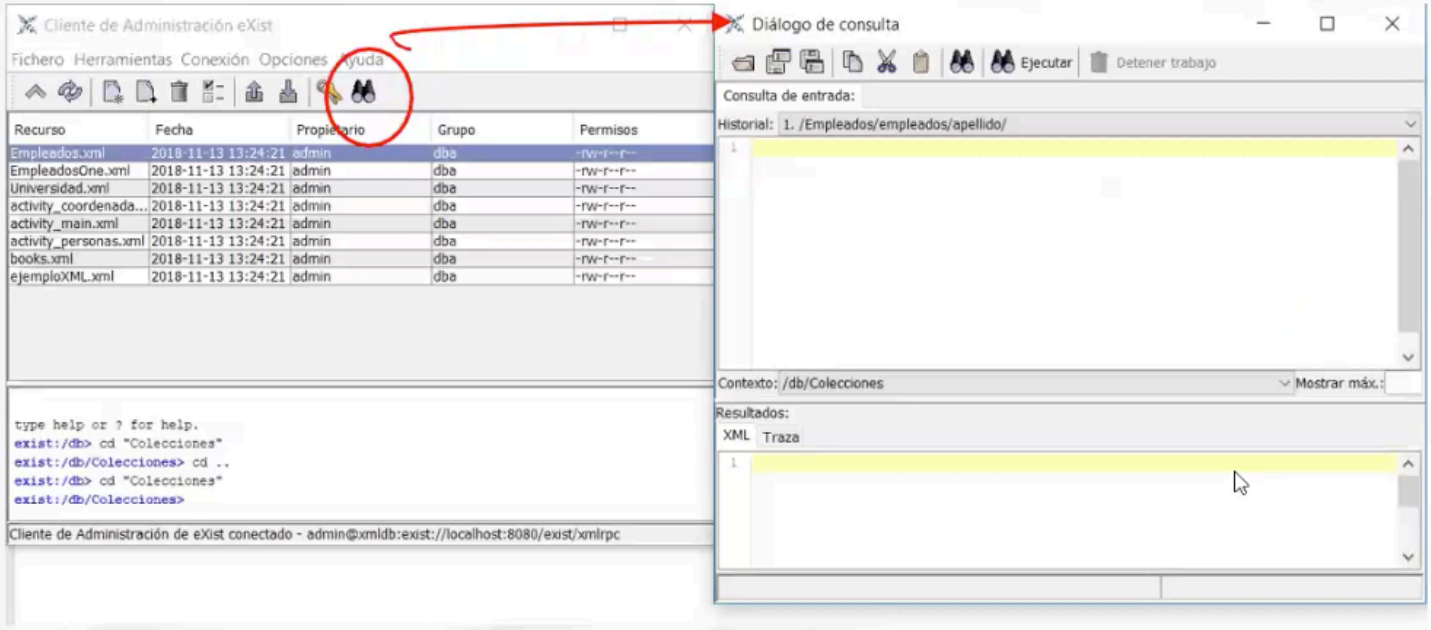
Vamos a subir una colección a la base de datos desde el cliente. Nos conectaremos como admin/admin.
En la ventana del cliente pulsamos el botón



Así se creará la colección con los documentos en la BD. En principio se subirán dentro del contexto ***exist:/db>.***

Si queremos hacer consultas a esa colección, haremos doble clic sobre ella para seleccionarla. Una vez estemos dentro de la colección (es decir, estamos en el contexto a utilizar, por ejemplo ***exist:/db/Colecciones>***) pulsamos los prismáticos (Consultar la base de datos usando XPath).

> Subir una colección a la base de datos



Se abrirá la ventana de Diálogo de Consultas, donde además de ejecutar las consultas, podremos guardarlas.

Si la consulta se realiza mal, aparecerá una ventana Java informando del error.

> Expresiones XPath

El lenguaje Xpath nos permite seleccionar nodos de un documento XML y obtener valores a partir de su contenido.

La forma en que Xpath selecciona partes del documento XML se basa en la representación arbórea que se genera del documento. Cuando recorremos un árbol XML podemos encontrarnos con algunos de los siguientes tipos de nodos:

- Nodo raíz → se representa por "/" y es la raíz del árbol.
- Nodos elementos → son las etiquetas del árbol, es cualquier elemento de un documento XML.
- Nodos texto → los caracteres que están entre las etiqueta
- Nodos comentario → las etiquetas de comentarios

Acciones sobre nodo

Acción	Descripción
/nombreDelNodo	Para seleccionar un nodo
Prefix:*	Selecciona nodos de un espacio de nombre determinados
Text()	Selecciona el contenido del elemento
Node()	Slecciona todos los nodos (Elementos y texto)
Processing-instruction()	Selecciona los nodos que son instrucciones de proceso.
Comment()	Selecciona los nodos de tipo comentario

> XPath

https://docs.microsoft.com/sl-si/dotnet/standard/linq/sampl
xml-file-multiple-purchase-orders

Ejemplo de cómo realizar búsquedas con XPath

/ - Recupera a partir de la raíz del documento

//Elemento - Recupera todos los elementos a partir de Elemento

/PurchaseOrders - Accede a la raíz del documento y no tiene nada en su interior

/PurchaseOrders/PurchaseOrder - Accede a todos los pedidos

/PurchaseOrders/PurchaseOrder[2] - Accede al pedido número 2

/PurchaseOrders/PurchaseOrder[@PurchaseOrderNumber = "99505"] - Recuperar un PurchaseOrder con un atributo concreto, por ejemplo 99505

/PurchaseOrders/PurchaseOrder[@PurchaseOrderNumber = "99505"]/Address[1]/Name
- Recuperar el nombre de la primera dirección (Cristian Osorio)

> Expresiones XPath

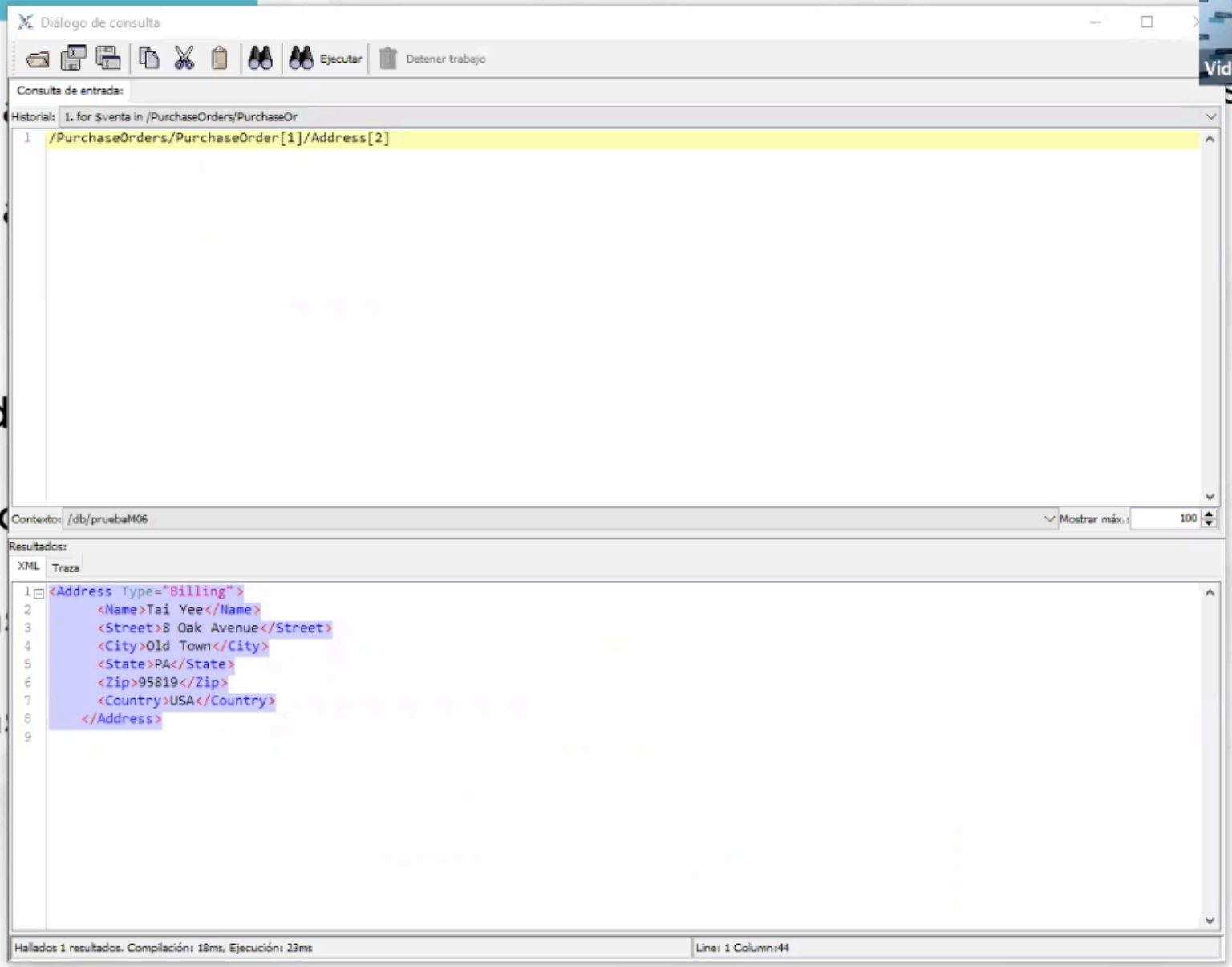
La sintaxis de Xpath es muy parecida a la del direccionamiento de ficheros. Se utiliza descriptores de ruta o de camino que sirven para marcar los nodos o elemtos que se encuentran en cierta ruta en el documento.

Vamos a realizar consultas a nuestra colección → Colecciones

Para hacer consultas Xpath se pueden escribir tanto de forma abreviada, o de una forma más compleja.

> EJERCICIO

- Recupera los Purch
- Recupera los Purch
- Recupera todas las
- Recupera la segund
- Recupera la direcci
- Recupera el Purcha
- Recupera el Purcha
- Recupera todos los
- Recupera todas las
- Recupera todos los códigos postales



> eXist

- Recupera los PurchaseOrders

/PurchaseOrders

- Recupera los PurchaseOrder

/PurchaseOrders/PurchaseOrder

- Recupera todas las direcciones

/PurchaseOrders/PurchaseOrder/Address

- Recupera la segunda dirección del 1r PurchaseOrder

/PurchaseOrders/PurchaseOrder[1]/Address[2]

- Recupera la dirección con el tipo Shipping (Type Shipping)

/PurchaseOrders/PurchaseOrder[1]/Address[@Type='Shipping']

- Recupera el PurchaseOrderNumber 99503

/PurchaseOrders/PurchaseOrder[@PurchaseOrderNumber="99503"]

- Recupera el PurchaseOrderNumber 99503 ítem n° 2

/PurchaseOrders/PurchaseOrder[@PurchaseOrderNumber="99503"]/Items/Item[2]

- Recupera todos los nombres de Productos (ProductName)

//ProductName

- Recupera todas las direcciones de Facturación (Type Billing)

//Address[@Type = "Billing"]

- Recupera todos los códigos postales

//Zip/text()

XQuery

ILERNA Online

Las consultas pueden usar las siguientes funciones	Collection(“/ruta”)	Se indica la ruta a la que hace referencia
	Doc(“/ruta/documento.xml)	Se indica la ruta de un documento de una colección y el nombre de dicho documento.

El lenguaje XQuery es una expresión capaz de leer datos de uno o más documentos en XML

> Expresión FLWOR

- Las consultas se pueden realizar mediante la expresión FLWOR:
  - **F**or → Se utiliza para almacenar nodos en una variable
  - **L**et → Se asignan valores resultantes de expresiones Xpath, para que sea más simplificado, sin hacer uso de las rutas constantemente
  - **W**here → Filtro de resultados para obtener solo los deseados.
  - **O**rder by → Se utiliza para ordenar los resultados obtenidos
  - **R**eturn → Se construye el documento XML

OPERADORES EN XQUERY

Matemáticos

+ - * div idiv(división entera) mod

Comparación

= != < > <= >= not()

Secuencia

Union, intersect, except

Redondeo

Floor(): Devuelve el valor más grande sin superar determinado tope.
Ceiling(): Valor más pequeño al que se puede acceder sin tener en cuenta la parte decimal.
Round(): Redondea el valor al entero más cercano.

ILERNA Online

Funciones de agrupación

Count() (para contar los elementos), min() (para obtener el mínimo), max() (para obtener el máximo), avg() (para obtener la media aritmética de unos datos), sum() (devuelve la suma de todos los elementos que le pasemos)

Funciones de cadena

Concat() (para concatenar), string-lenght(), start-with(), ends-with(), substring()(extraer subcadenas de texto), upper-case(), lower-case(), string()

> Ejemplo XQuery

Para extraer usamos el simbolo $ y el nombre de una variable para crear dicha variable.

- xquery version "3.1";
  - for $alumno in /alumnos/alumno Sirve para sustituir la ruta por el nombre de la variable.
  - let $nombre := ($alumno/nombre/text()) Asignamos el contenido de la etiqueta texto. Como un where.
  - let $nombreOnliner := concat($nombre," Onliner")
  - order by $nombre
  - return <alumno>{$nombreOnliner}</alumno>

ILERNA Online

Indica qué sentencia necesitas para obtener estos resultados

Usa exist

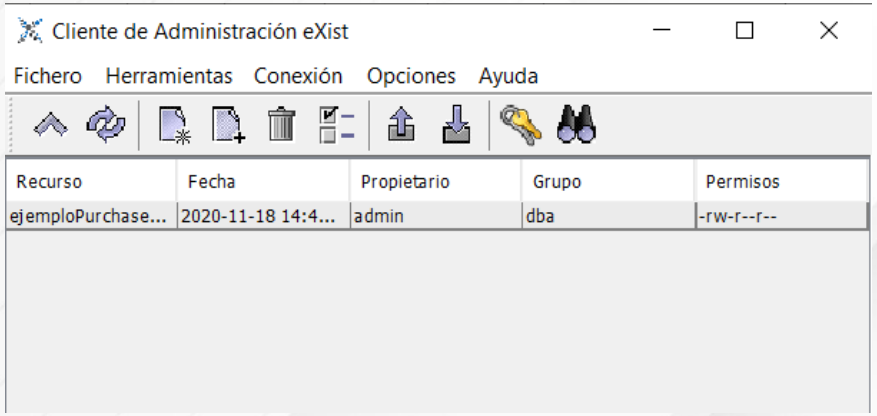
- Muestra todos los purchaseOrder bajo la etiqueta venta
- Muestra todos los purchaseOrder que sean de Seattle
- Muestra todos los purchaseOrder con USPrice superior a 50
- Muestra todos los purchaseOrder con con mas de dos artículos (aunque sean el mismo)
- Modifica el nombre del Nodo PurchaseOrder a Ventas
- Modifica la quantity del primer Venta, el 1r ítem a 33
- Elimina el 3r PurchaseOrder
- Cuenta todos los PurchaseOrder

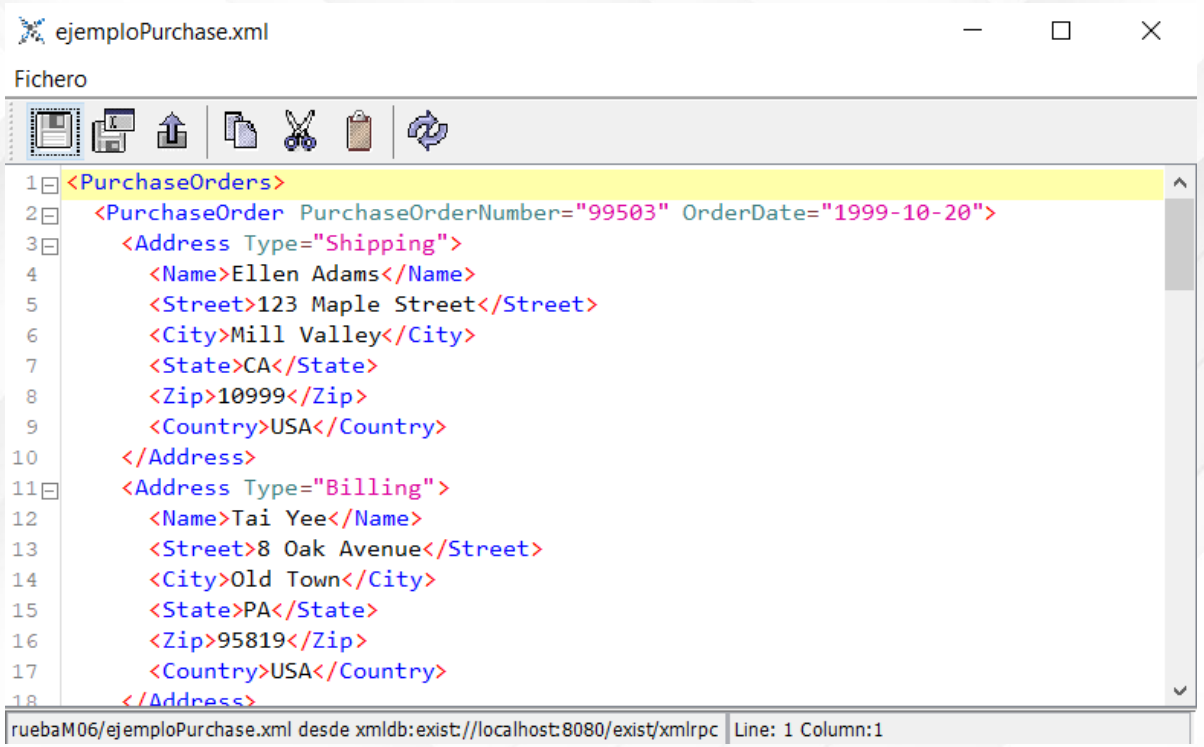
> Ejercicios XQuery

- Muestra todos los purchaseOrder bajo la etiqueta venta

> Ejercicios XQuery

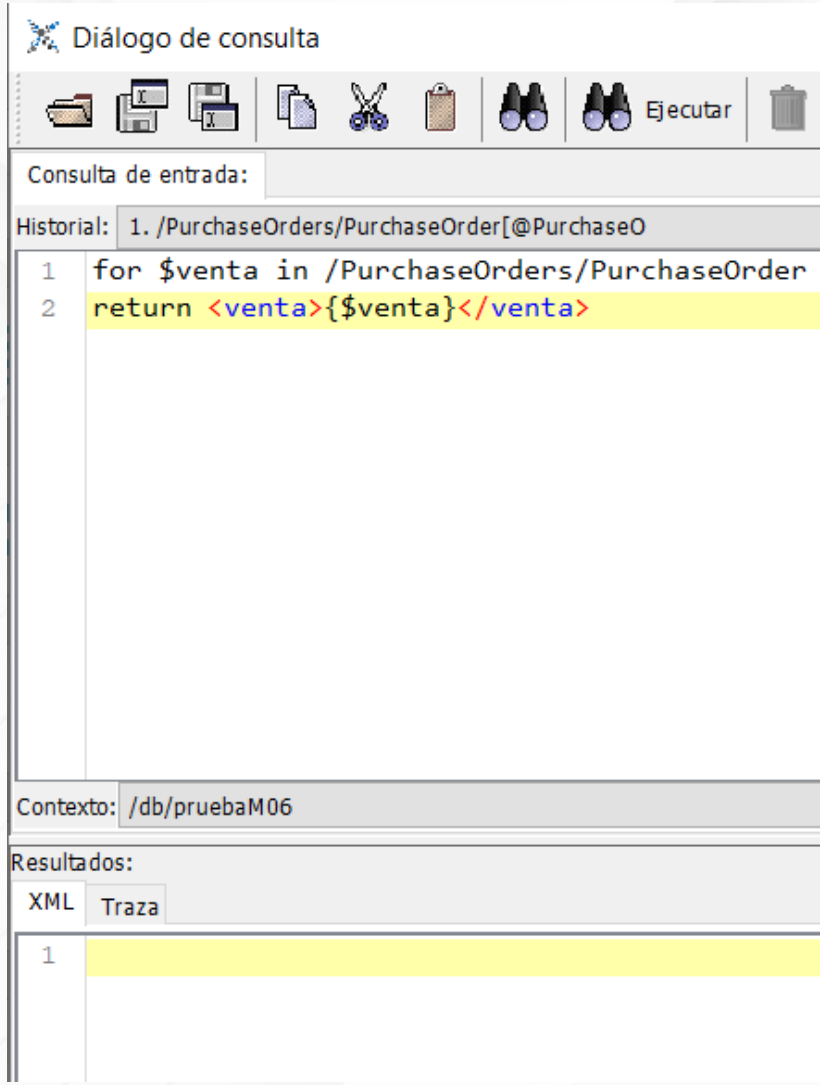
- Muestra todos los purchaseOrder bajo la etiqueta venta





> Ejercicios XQuery

- Muestra todos los purchaseOrder bajo la etiqueta venta
- Creamos la variable venta para guardar todos los PurchaseOrder que encontramos en la ruta /PurchaseOrders/PurchaseOrder
- Como no queremos modificar nada, solo nos queda devolverlos, pero dentro de las etiquetas ventas y /ventas
- Lo ponemos entre llaves para que nos lo muestre de forma correcta
- Si no, mostraría el texto automáticamente



> Ejercicios XQue

- Muestra tod
- Creamos la
  PurchaseOr
  /PurchaseO
- Como no qu
  devolverlos
- Lo ponemos
  correcta
- Si no, mostr

```
<venta><PurchaseOrder PurchaseOrderNumber="99503" OrderDate="1999-10-20">
    <Address Type="Shipping">
      <Name>Ellen Adams</Name>
      <Street>123 Maple Street</Street>
      <City>Mill Valley</City>
      <State>CA</State>
      <Zip>10999</Zip>
      <Country>USA</Country>
    </Address>
    <Address Type="Billing">
      <Name>Tai Yee</Name>
      <Street>8 Oak Avenue</Street>
      <City>Old Town</City>
      <State>PA</State>
      <Zip>95819</Zip>
      <Country>USA</Country>
    </Address>
    <DeliveryNotes>Please leave packages in shed by driveway.</DeliveryNotes>
    <Items>
      <Item PartNumber="872-AA">
        <ProductName>Lawnmower</ProductName>
        <Quantity>1</Quantity>
        <USPrice>148.95</USPrice>
        <Comment>Confirm this is electric</Comment>
      </Item>
      <Item PartNumber="926-AA">
        <ProductName>Baby Monitor</ProductName>
        <Quantity>2</Quantity>
        <USPrice>39.98</USPrice>
        <ShipDate>1999-05-21</ShipDate>
      </Item>
    </Items>
  </PurchaseOrder></venta>
<venta><PurchaseOrder PurchaseOrderNumber="99505" OrderDate="1999-10-22">
```

ILERNA Online

sulta

Ejecutar

Orders/PurchaseOrder[@PurchaseO

in /PurchaseOrders/PurchaseOrder

nta>{$venta}</venta>

06

> Ejercicios XQuery

- Muestra todos los purchaseOrder que sean de Seattle

> Ejercicios XQuery

- Usamos la variable venta para que busque en la ruta /PurchaseOrders/PurchaseOrder y especificamos la condición que debe cumplirse
- Con where fijamos la condición que el texto que tengamos en la etiqueta Address sea Seatle
- Devolvemos el contenido de la variable venta

Consulta de entrada:

Historial: 1. /PurchaseOrders/PurchaseOrder[@PurchaseO

```
1  for $venta in /PurchaseOrders/PurchaseOrder
2  where $venta/Address/City/text()  = 'Seattle'
3  return $venta
4
```

"Seattle" no 'Seattle'

> Ejercicios XQuery

- Usamos la varia
  /PurchaseOrder
  condición que d
- Con where fijam
  tengamos en la
- Devolvemos el d

```
1  <PurchaseOrder PurchaseOrderNumber="99504" OrderDate="1999-10-22">
2      <Address Type="Shipping">
3        <Name>Jessica Arnold</Name>
4        <Street>4055 Madison Ave</Street>
5        <City>Seattle</City>
6        <State>WA</State>
7        <Zip>98112</Zip>
8        <Country>USA</Country>
9      </Address>
10     <Address Type="Billing">
11       <Name>Jessica Arnold</Name>
12       <Street>4055 Madison Ave</Street>
13       <City>Buffalo</City>
14       <State>NY</State>
15       <Zip>98112</Zip>
16       <Country>USA</Country>
17     </Address>
18     <Items>
19       <Item PartNumber="898-AZ">
20         <ProductName>Computer Keyboard</ProductName>
21         <Quantity>1</Quantity>
22         <USPrice>29.99</USPrice>
23       </Item>
24       <Item PartNumber="898-AM">
25         <ProductName>Wireless Mouse</ProductName>
26         <Quantity>1</Quantity>
27         <USPrice>14.99</USPrice>
28       </Item>
29     </Items>
30   </PurchaseOrder>
```

```
haseOrder[@PurchaseO
chaseOrders/PurchaseOrder
ss/City/text()  = 'Seattle'
```

> Ejercicios XQuery

- Muestra todos los purchaseOrder con USPrice superior a 50

ILERNA Online

- Usamos la variable venta para que busque en la ruta /PurchaseOrders/PurchaseOrder y especificamos la condición que debe cumplirse
- En este caso, queremos que la condición de que el precio de los productos sea superior a 50$
- Se obtienen todos los elementos con sus precios, pero se añade la condición de que el precio sea mayor que 50

Consulta de entrada:

Historial: 1. /PurchaseOrders/PurchaseOrder[@PurchaseO

```
1  for $venta in /PurchaseOrders/PurchaseOrder
2  let $price := sum($venta/Items/Item/USPrice)
3  where $price > 50
4  return $venta
```

Si en lugar de return $venta usamos return $precio devuelve el valor de la suma.

> Ejercicios XQuery

- Usamos la variable ve
  /PurchaseOrders/Purc
  condición que debe cu
- En este caso, querem
  precio de los producto
- Se obtienen todos los
  pero se añade la cond
  mayor que 50

Resultados:

XML | Traza

```
1  <PurchaseOrder PurchaseOrderNumber="99503" OrderDate="1999-10-20">
2      <Address Type="Shipping">
3        <Name>Ellen Adams</Name>
4        <Street>123 Maple Street</Street>
5        <City>Mill Valley</City>
6        <State>CA</State>
7        <Zip>10999</Zip>
8        <Country>USA</Country>
9      </Address>
10     <Address Type="Billing">
11       <Name>Tai Yee</Name>
12       <Street>8 Oak Avenue</Street>
13       <City>Old Town</City>
14       <State>PA</State>
15       <Zip>95819</Zip>
16       <Country>USA</Country>
17     </Address>
18     <DeliveryNotes>Please leave packages in shed by driveway.</DeliveryNotes>
19     <Items>
20       <Item PartNumber="872-AA">
21         <ProductName>Lawnmower</ProductName>
22         <Quantity>1</Quantity>
23         <USPrice>148.95</USPrice>
24         <Comment>Confirm this is electric</Comment>
25       </Item>
26       <Item PartNumber="926-AA">
27         <ProductName>Baby Monitor</ProductName>
28         <Quantity>2</Quantity>
29         <USPrice>39.98</USPrice>
30         <ShipDate>1999-05-21</ShipDate>
31       </Item>
32     </Items>
33   </PurchaseOrder>
34
```

haseO

/PurchaseOrder
s/Item/USPrice)

ILERNA Online

- ¿Qué mostraría este código?

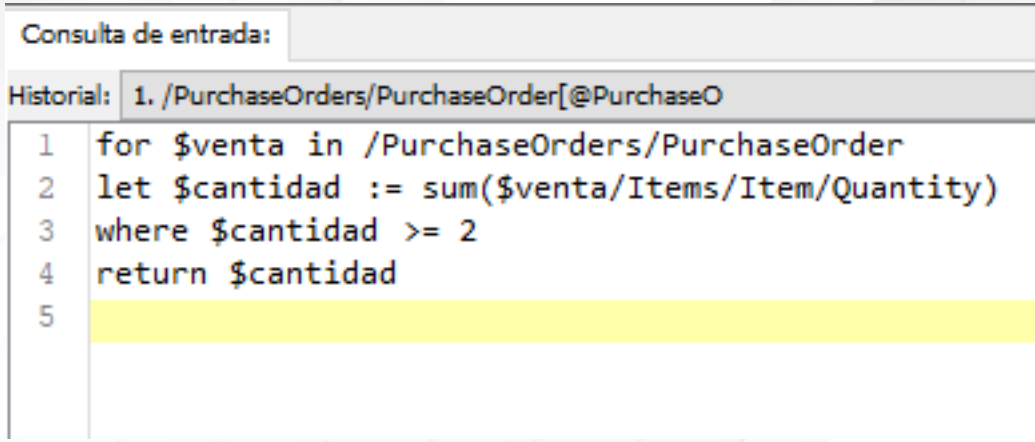
```
for $venta in /PurchaseOrders/PurchaseOrder
let $price := sum($venta/Items/Item/USPrice)
where $price > 50
return <ventas>  // Un nuevo document XML
<venta>{$venta}</venta>
<total>{$price}</total>
</ventas>
```

ILERNA Online

- Muestra todos los purchaseOrder con con mas de dos artículos (aunque sean el mismo)

> Ejercicios XQuery

- Usamos la variable venta para que busque en la ruta /PurchaseOrders/PurchaseOrder y especificamos la condición que debe cumplirse
- Ahora nos vamos a centrar en contar los artículos que tiene cada pedido y vamos a quedarnos con los que tienen al menos dos artículos.
- Devolvemos la cantidad



> Renombrar nodos

ILERNA Online

- Para renombrar usaremos la sentencia rename:
- Update rename NODO as ‘NuevoNombre’
- No nos permite cambiar la raíz del documento (la primera etiqueta)

ILERNA Online

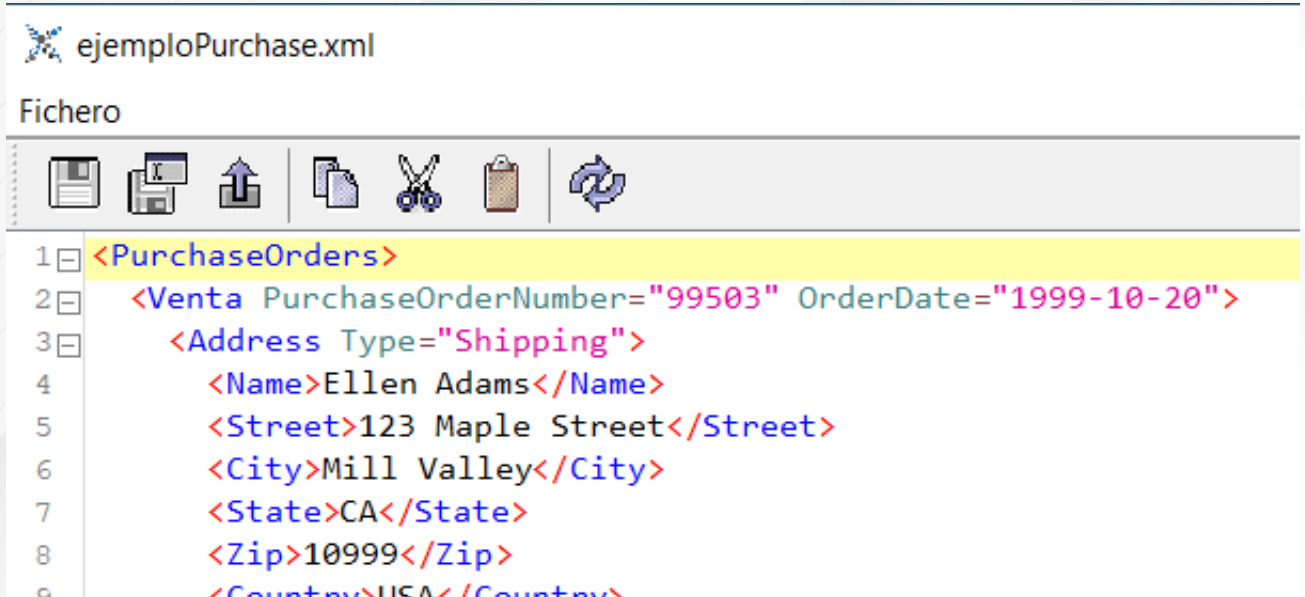
- Modifica el nombre del Nodo PurchaseOrder a Ventas

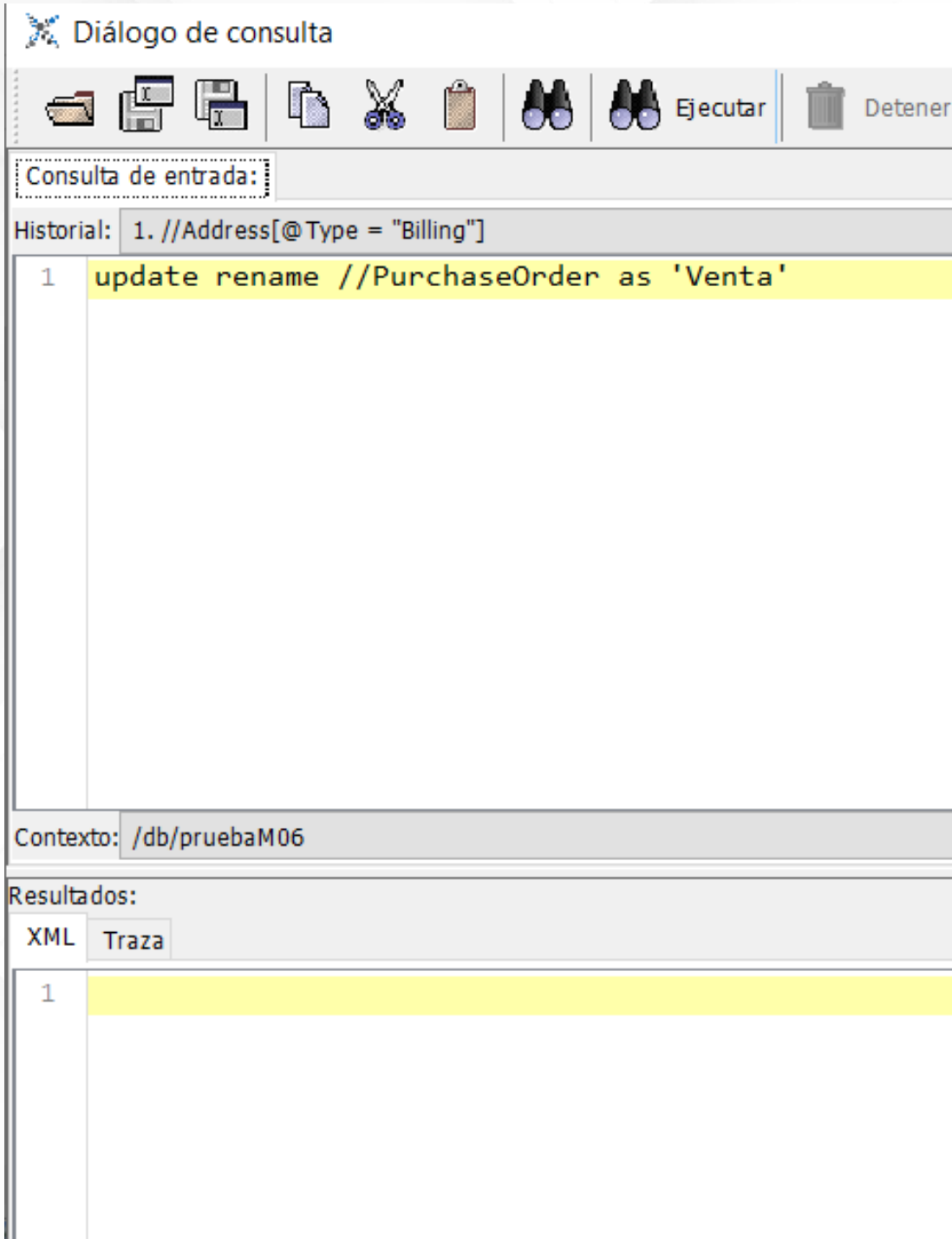
> Ejercicios XQuery

- Usamos la orden:

  update rename //PurchaseOrder as ‘Venta’

- Si abrimos el documento XML veremos que se actualizado con este cambio





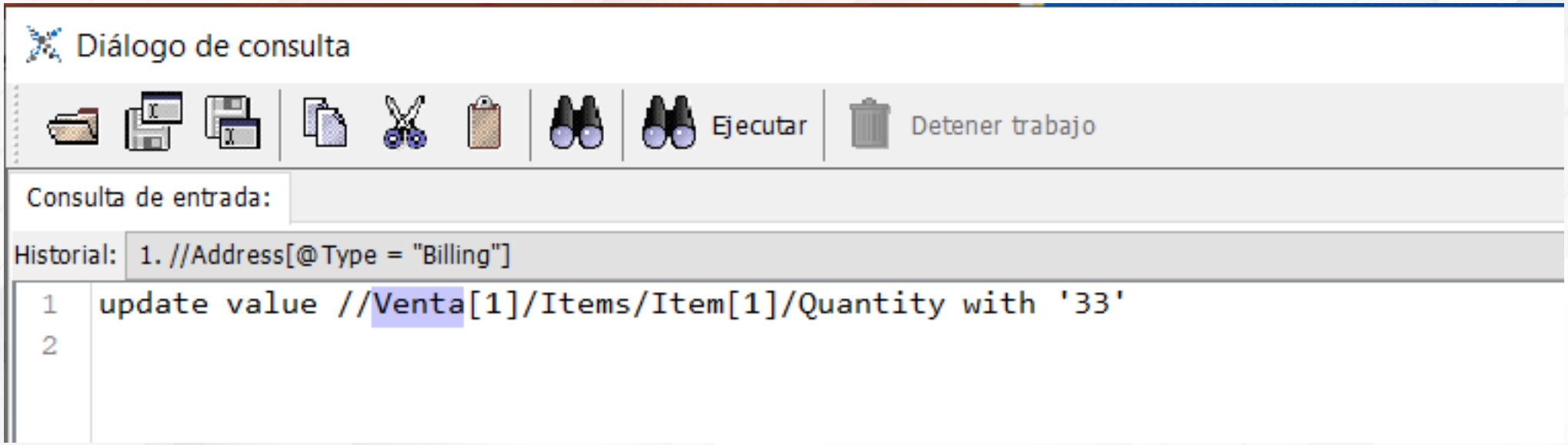
> Modificar nodos

- Para actualizar usaremos la sentencia value:
- update value Nodo with ‘value’

- Modifica la quantity del primer Venta, el 1r ítem a 33

- Usamos la orden:

  update value //Venta[1]/Items/Item[1]/Quantity with '33’



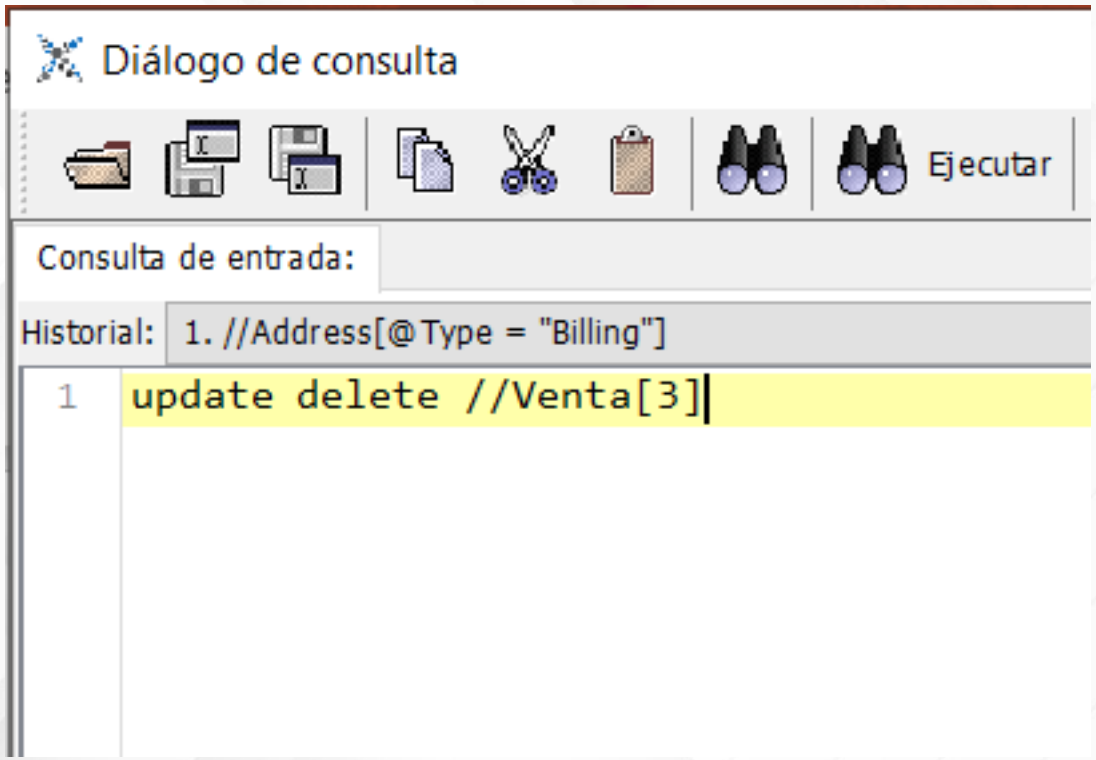
> Eliminar nodos

- Para eliminar usaremos la sentencia delete:
- Update delete xpath

> Eliminar nodos

- Elimina el 3r PurchaseOrder

> Eliminar nodos



> Ejercicios XQuery

ILERNA Online

- Cuenta todos los PurchaseOrder

> Ejercicios XQuery

Consulta de entrada:

Historial: 1. //Address[@Type = "Billing"]

```
1  for $ventas in /PurchaseOrders
2  let $total := count($ventas/PurchaseOrder)
3  return $total
```

Java Bean

ILERNA Online

Introducción

Gracias a los avances de la Telecomunicaciones y la Informática, está cambiando la forma en la que se desarrollan las aplicaciones software.

Los modelos de programación tradicionales se ven incapaces de manejar la complejidad de los requisitos de los sistemas abiertos y distribuidos.

El desarrollo de Software Basado en Componentes (DSBC), trata de sentar las bases para el diseño y desarrollo de aplicaciones distribuidas.

ILERNA Online

Según Szyperski, 1998:

Un componente es una unidad de composición de aplicaciones software, que posee un conjunto de interfaces y un conjunto de requisitos y que ha de poder ser desarrollado, adquirido e incorporado al sistema, y está compuesto por otro componentes de forma independientes en tiempo y espacio.

ILERNA Online

Características de un componente

- Debe ser independiente de la plataforma en la que se ejecute.
- Debe ser identificable, que permita acceder fácilmente a sus servicios y que permita su clasificación.
- Debe poseer su propio contenido sin necesitar fuentes externas para llevar a cabo su función.
- Puede ser remplazado por otro componente que lo mejore o una nueva versión.

ILERNA Online

Características de un componente

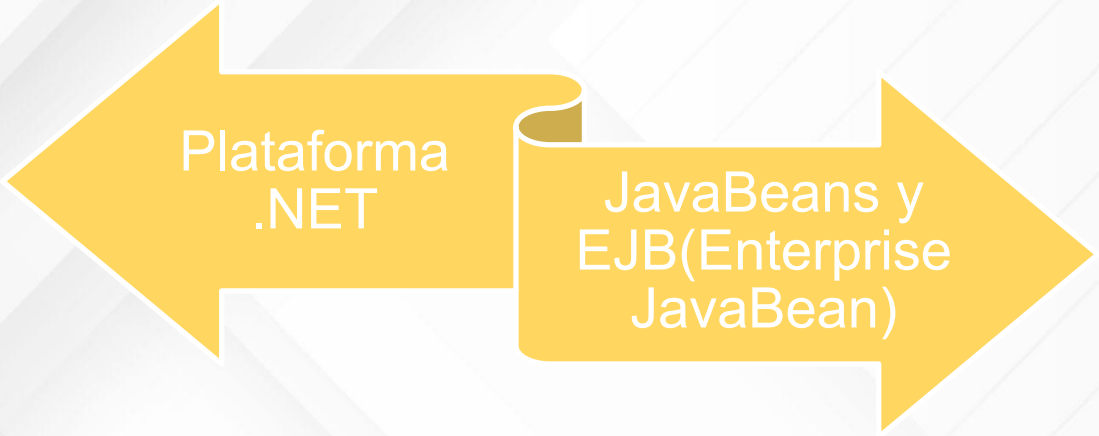
- Solo se puede tener acceso a través de su interfaz.
    - Esta define el conjunto de operaciones que un componente puede realizar, a estas operaciones también se les llama servicios o responsabilidades.
- Debe estar bien documentado y servir para varias aplicaciones.
- Se distribuye como un paquete, este contiene todos los elementos necesarios para su correcta utilización.

ILERNA Online

Tecnologías de componentes

Las tecnologías basadas en componentes incorporan dos elementos:

- Modelo de componentes: Se especifican las reglas para el diseño de componentes, interfaces y la interacción entre ellos.
- Plataforma de componentes: Es la infraestructura de software requerida para la ejecución de aplicaciones basadas en componentes.



Ventajas:

- Reutilización de software
- Disminuye la complejidad del software, ya que es posible ir probando pequeños fragmentos del componente antes de probar el conjunto de todos los componentes unidos.
- Mejora el mantenimiento
- Los errores son fáciles de detectar.
- Aumenta la calidad del software ya que se pueden realizar versiones mejoradas de un componente

Desventajas:

- No siempre se localizan los componentes para un proyecto.
- Faltan estándares y procesos de certificación que garanticen la calidad de los mismos

ILERNA Online

- En esta unidad se explican los JavaBeans, tecnología de componentes basada en Java. Frecuentemente se usa el término Bean para hacer referencia a un JavaBean.

> Java Bean

ILERNA Online

Un “bean” es un componente creado en Java que puede ser reutilizado y manipulado desde el IDE

Unos ejemplos de JavaBeans son las librerías gráficas AWT, API de Java que permite hacer aplicaciones con componentes GUI (ventanas, botones, campos de texto, etc) …

> Java Bean


```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.herprogramacion.botones.ActividadBotones">

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:text="AGREGAR" />
</RelativeLayout>
```

Atributo	Descripción
`android:text`	Permite cambiar el texto de un botón
`android:background`	Se usa para cambiar el fondo del botón. Puedes usar un recurso del archivo `colors.xml` o un *drawable*.
`android:enabled`	Determinar si el botón está habilitado ante los eventos del usuario. Usa `true` (valor por defecto) para habilitarlo y `false` en caso contrario.
`android:gravity`	Asigna una posición al texto con respecto a los ejes x o y dependiendo de la orientación deseada. *Por ejemplo*: Si usas `top`, el texto se alineará hacia el borde superior.
`android:id`	Representa al identificador del botón para diferenciar su existencia de otros views.
`android:onClick`	Almacena la referencia de un método que se ejecutará al momento de presionar el botón.
`android:textColor`	Determina el color del texto en el botón
`android:drawable*`	Determina un drawable que será dibujado en la orientación establecida. *Por ejemplo*: Si usas el atributo `android:drawableBottom`, el drawable será dibujado debajo del texto.

> Definir un Java Bean

ILERNA Online

- Para **definir un Bean** se requieren ciertas **reglas**:
  - Tener uno o más **constructores**, al menos uno sin argumentos.
  - Debe implementar la interfaz **Serializable** (para poder implementar persistencia).
  - Debe ofrecer **acceso (público)** a sus propiedades mediante métodos **get y set.**
  - Los nombres de los **métodos** (getter y setter) deben cumplir ciertas normas.

> Definir un Java Bean

ILERNA Online

- Para **definir un Bean** se requieren ciertas **reglas**:
  - Tener uno o más **constructores**, al menos uno sin argumentos.
  - Debe implementar la interfaz **Serializable** (para poder implementar persistencia).
  - Debe ofrecer **acceso (público)** a sus propiedades mediante métodos **get y set.**
  - Los nombres de los **métodos** (getter y setter) deben cumplir ciertas normas.

ILERNA Online

- Normas para los nombres de los métodos:
  - El nombre debe estar precedido por get para leer una propiedad no booleana.
    - → getNombre() seria válido para la propiedad nombre.
  - Si la propiedad es booleana, el prefijo del método debe ser get o is.
    - → getAlive() o isAlive() serían validos para una propiedad booleana.
  - Para almacenar un valor debe tener el prefijo set.
  - Para completar el nombre de algún método getter o setter, solo hay que poner la primera letra que los une en mayúsculas. (Camel Case)
  - Los métodos setter deben ser públicos, devolver un tipo void y recibir un argumento del tipo de propiedad al que van a dar valor.
  - Los métodos getter deben ser públicos y no aceptan argumentos. Devuelven un valor del mismo tipo que recibe el método setter.

> Definir un Java Bean

```java
public class Usuario implements Serializable {
    private int id;
    private String nombre;
    private String apellido;
    private String mail;
    private String telf;

    //Constructor por defecto
    public Usuario(){

    }

    public int getId() {
        return id;
    }

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

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getApellido() {
        return apellido;
    }

    public void setApellido(String apellido) {
        this.apellido = apellido;
    }

    public String getMail() {
        return mail;
    }

    public void setMail(String mail) {
        this.mail = mail;
    }

    public String getTelf() {
        return telf;
    }

    public void setTelf(String telf) {
        this.telf = telf;
    }
}
```

ILERNA Online

Las **propiedades de un *Bean*** son los atributos capaces de determinar su apariencia y comportamiento. Por ejemplo, "*Película*" puede tener las siguientes propiedades: *id, nombre, género, duración*, etc.

Para obtener información de alguna de ellas se usarán los métodos anteriormente nombrados *getter* y, para cambiar el valor, se utilizarán los métodos *setter*, de la siguiente forma:

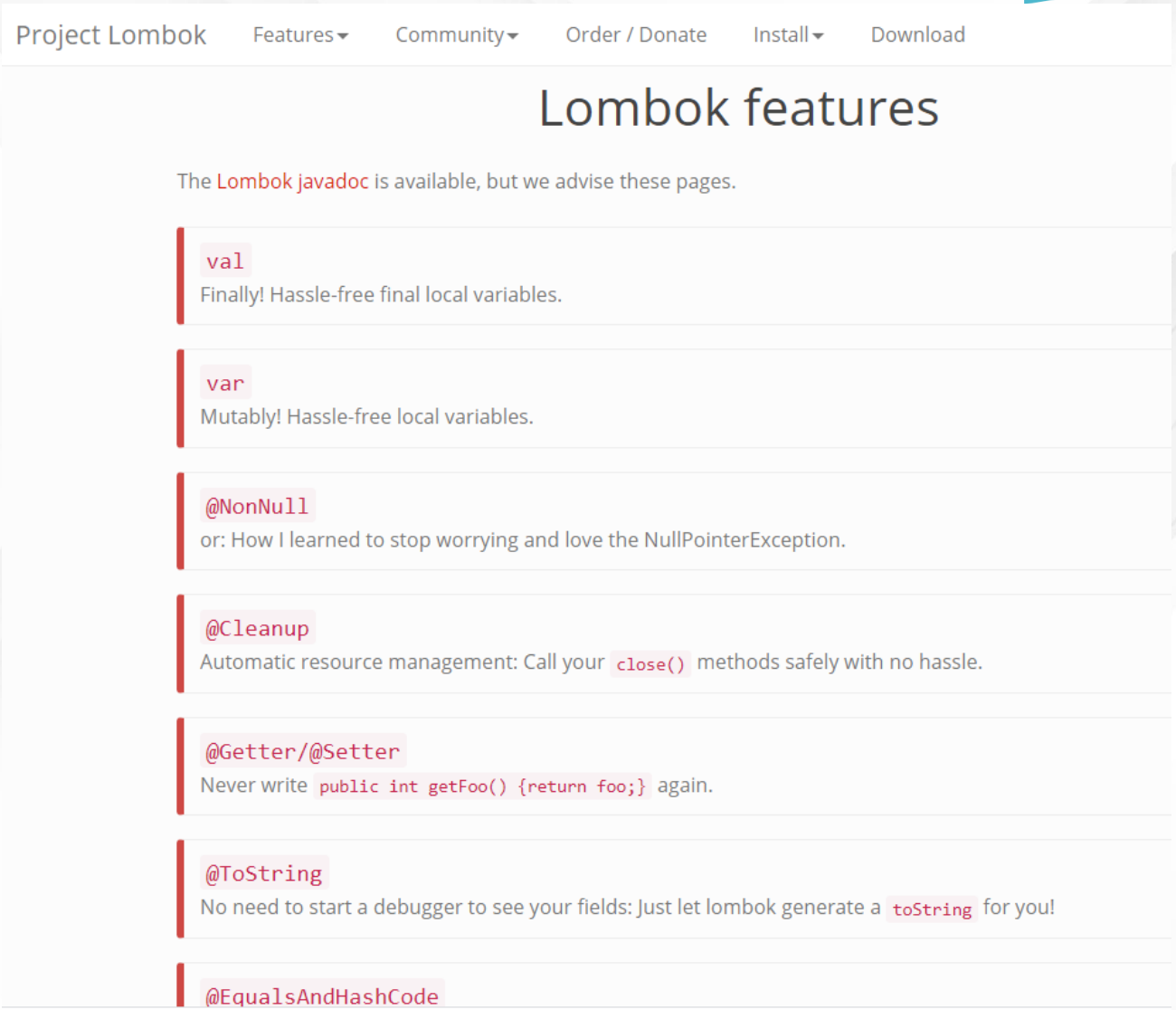
```
public TipoPropiedad getNombrePropiedad() { ... }

public void setNombrePropiedad (TipoPropiedad valor) { ... }
```

```
public String getNombre() {
        return nombre;
}
public void setNombre(String nombre) {
        this.nombre = nombre;
}
```

ILERNA Online

LIBRERÍA LOMBOK

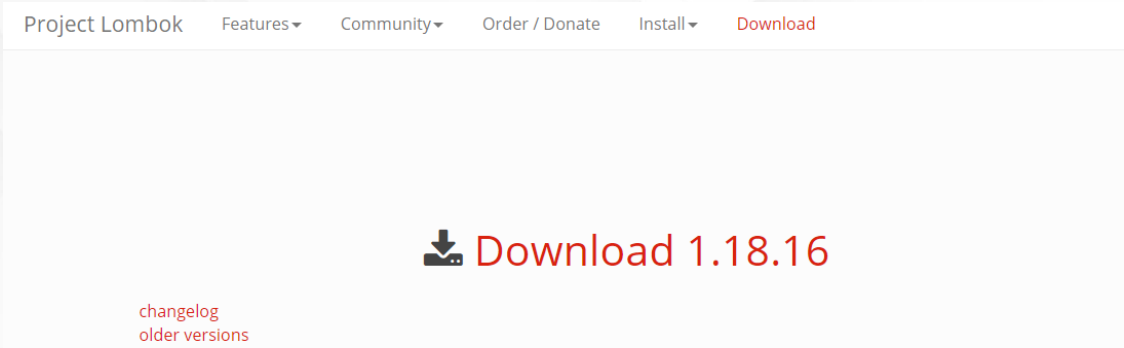
- Conjunto de anotaciones que permiten simplificar las clases
- @toString: nos permite crearlo sin ponerlo
- @Data: nos permite usar los getters y setter, junto con toString y EqualsAndHasCode sin declararlos
- Solo poniendo la anotación, tendríamos la funcionalidad

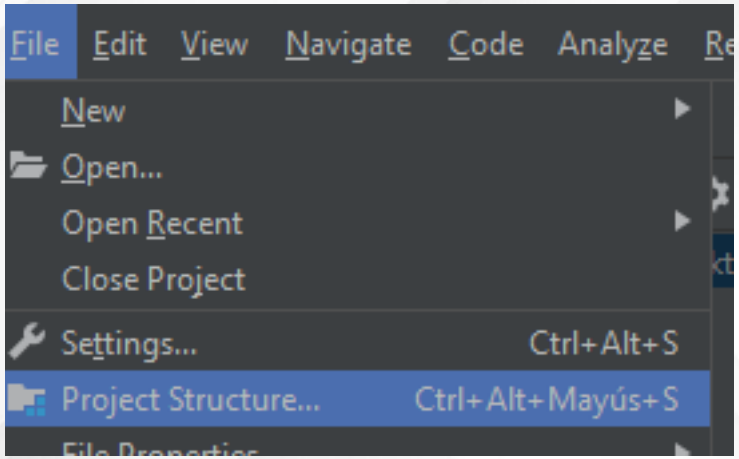


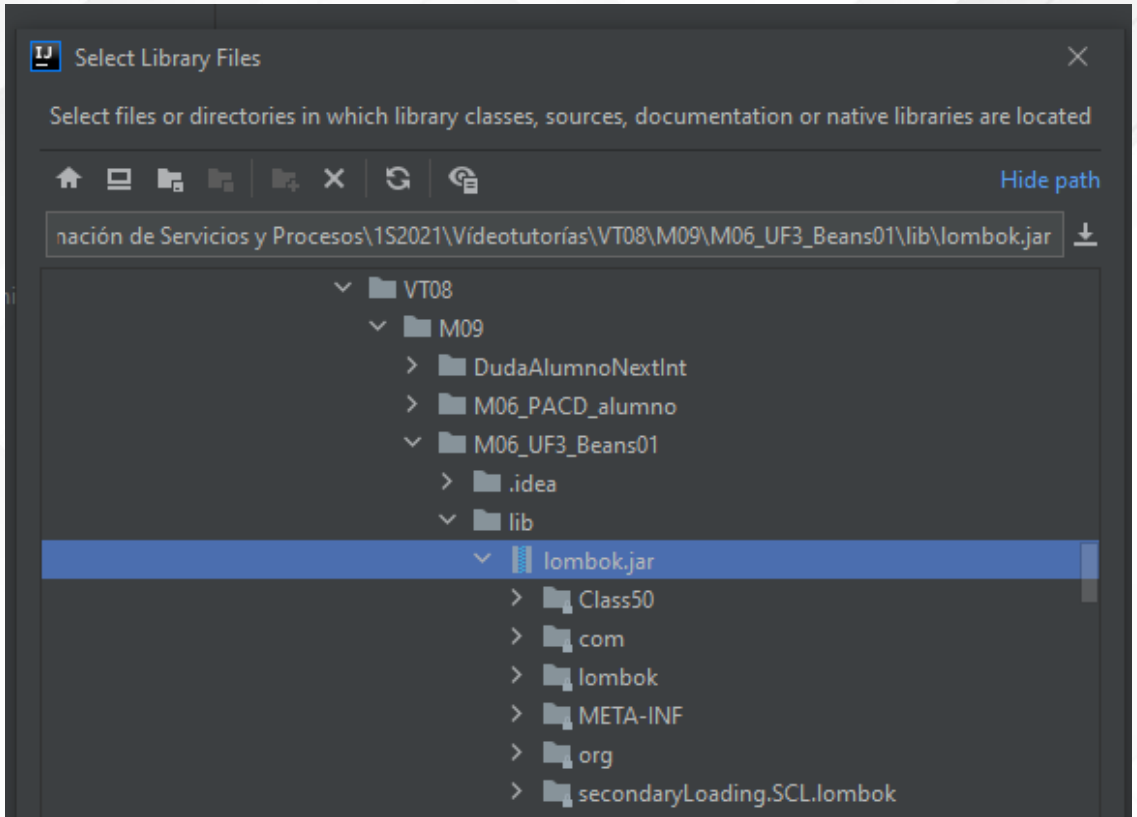
> Java BEAN

Descargar LOMBOK

- URL: https://projectlombok.org/download
- Librería que podemos descargar e importar en el IDE
- Guardar el fichero en el directorio lib del proyecto (si no existe, lo creamos)
- Importar la librería externa en el proyecto: File/ Project Structure / Libraries/ símbolo + y buscamos el fichero que hemos pegado en lib



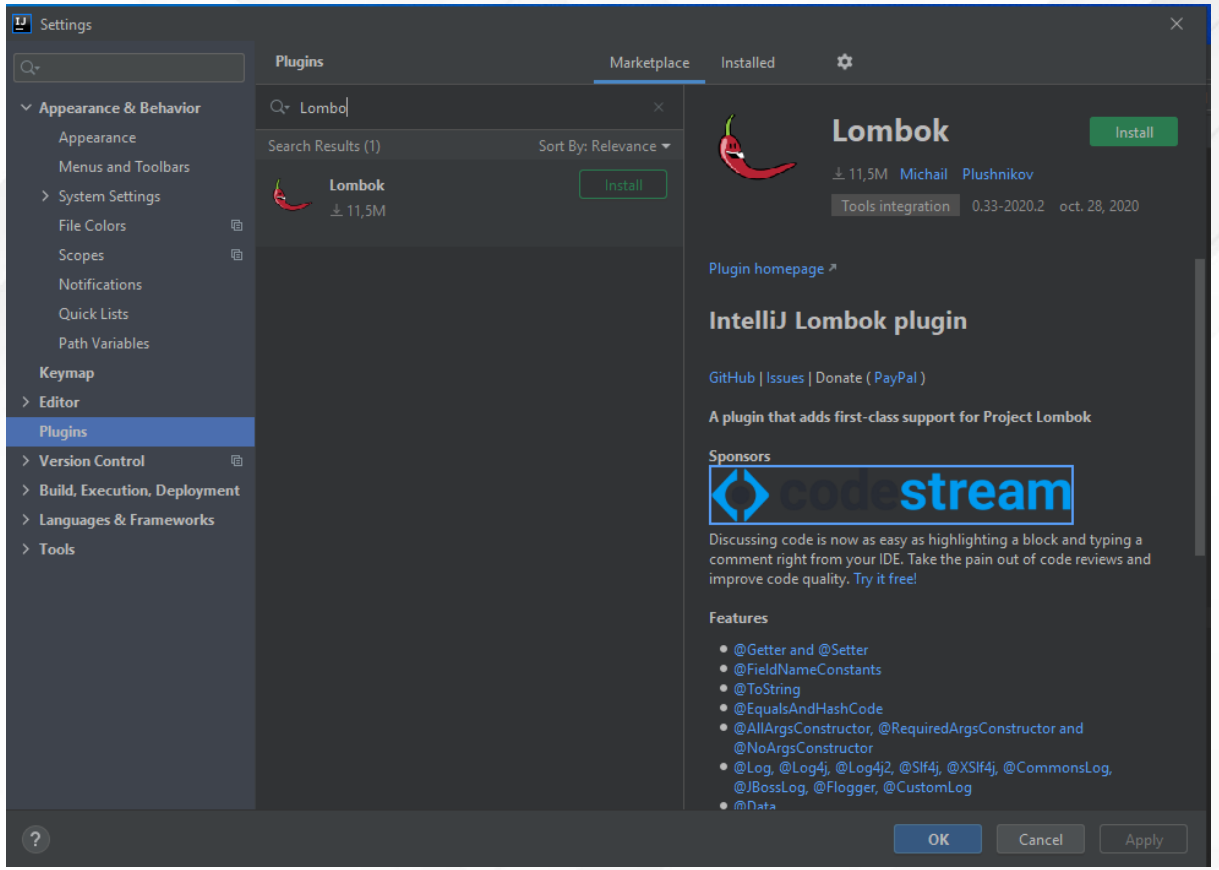




ILERNA Online

Descargar LOMBOK

- Si estamos usando IntelliJ IDEA tendremos que instalar un plugin, pero es muy sencillo
- File / Settings / Plugins / Buscamos Lombok



> Java BEAN

Usando LOMBOK

- Encima de la clase podemos poner @Data (imprtar librería Lombok)
- Desde el método Main podemos usar getters y setters sin haberlo definido manualmente

Usuario.java Main.java

```java
import lombok.Data;

import java.io.Serializable;
import java.sql.ResultSet;
import java.sql.SQLException;

@Data
public class Usuario implements Serializable {
    private int id;
    private String nombre;
    private String apellido;
    private String mail;
    private String telf;

    //Constructor por defecto
    public Usuario(){

    }

    public Usuario(int id, String nombre, String apellido, String mail, String telf) {
        this.id = id;
        this.nombre = nombre;
        this.apellido = apellido;
        this.mail = mail;
        this.telf = telf;
    }
}
```

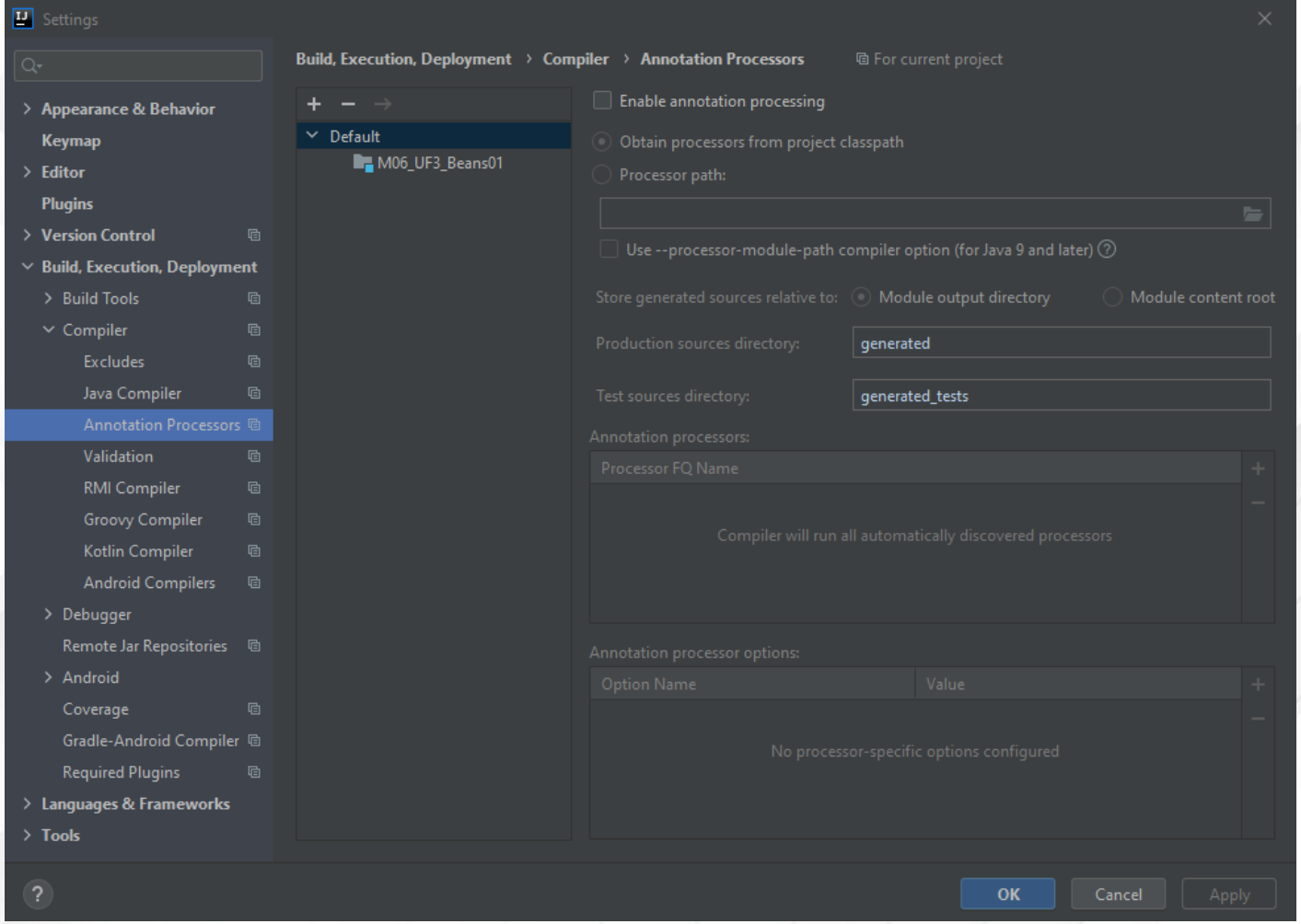
Usuario.java Main.java

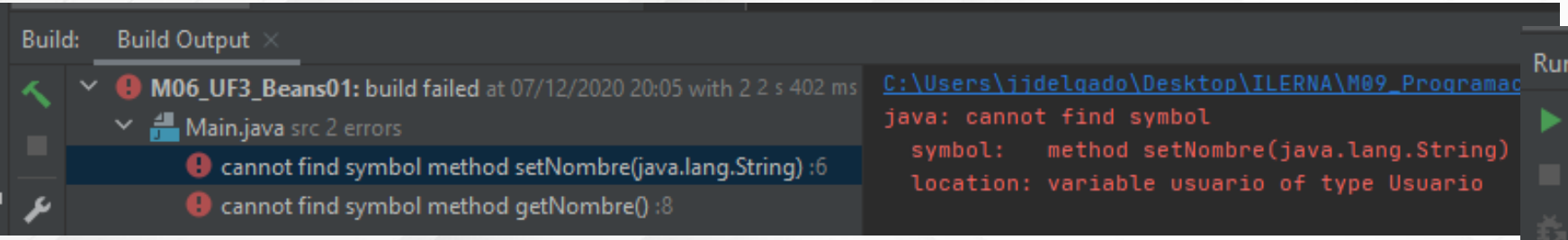
```java
public class Main {
    public static void main(String[] args) {
        Usuario usuario = new Usuario();
        usuario.setNombre("Jorge");

    }


}
```

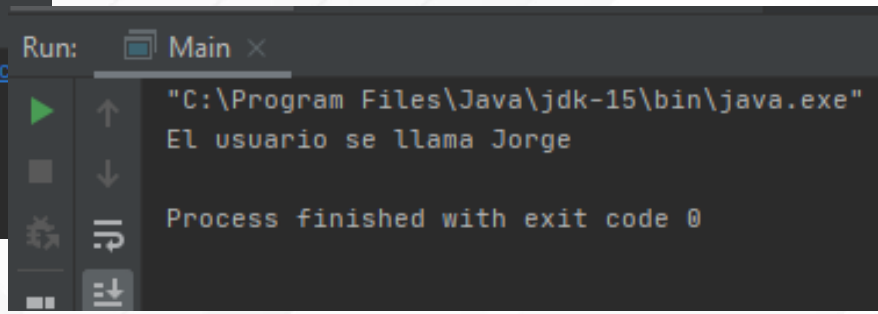
> Java BEAN

Usando LOMBOK

- Si os da este error de no reconocer el método setNombre, es porque no está habilitado el uso Annotation Processors
- File / Settings / Build, Execution, Deployment / Annotation Processors/ Enable annotation processing







> Java BEAN

```java
import java.io.Serializable;

public class EjemBean implements Serializable{

    private String atributo;

    public EjemBean() {
        this.atributo = "";
    }

    public String getAtributo() {
        return this.atributo;
    }

    public void setAtributo(String atributo) {
        this.atributo = atributo;
    }
}
```

Los JavaBeans se han diseñado para utilizarse y ser manipulados por herramientas visuales, pero en esta UF los trataremos como componentes de acceso a base de datos, así el mismo componente se podrá utilizar para acceder a diferentes bases de datos.

El ejemplo de la izquierda muestra un JavaBean no visual con un atributo, un constructor sin parámetros y los métodos get y set.

> Propiedades de un JavaBean

ILERNA Online

```java
private String nombre;

public String getNombre() {
    return nombre;
}

public void setNombre(String nombre) {
    this.nombre = nombre;
}
```

```java
private boolean open;

public boolean isOpen() {
    return open;
}

public void setOpen(boolean open) {
    this.open = open;
}
```

> Propiedades de un JavaBean

Interfaz de componente

Usuario.java | Main.java | UsuarioBean.java | UsuarioBeanImpl.java

```java
import java.sql.SQLException;
import java.util.List;

public interface UsuarioBean {
    //Dos funciones:

    //Función para listar un conjunto de usuarios
    public List<Usuario> getUsuarios() throws SQLException;

    //Función para escribir el usuario
    public void insertUsuario(Usuario usuario) throws SQLException;
}
```

Clase que implementa la interfaz

Usuario.java | Main.java | UsuarioBean.java | UsuarioBeanImpl.java

```java
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class UsuarioBeanImpl implements UsuarioBean {

    //Sobreescribimos los métodos de la Interfaz

    //Definimos el método para conectarnos a la BD
    public static Connection getConnection() throws SQLException {
        String BBDD = "jdbc:mysql://localhost:3360/m06";
        String USER = "root"; //Aquí van mis datos de conexión en XAMPP
        String PASSWORD = "";
        return DriverManager.getConnection(BBDD, USER, PASSWORD);
    }

    @Override
    public List<Usuario> getUsuarios() throws SQLException {
        //Vamos a consultar todos los usuarios que existen en la BD
        List<Usuario> usuarios = new ArrayList<>();
```

Añadimos Constructor a Usuario que reciba un resultSet

```java
    public Usuario(ResultSet resultSet) throws SQLException {
        this.id = resultSet.getInt( columnIndex: 1);
        this.nombre = resultSet.getString( columnIndex: 2);
        this.apellido = resultSet.getString( columnIndex: 3);
        this.mail = resultSet.getString( columnIndex: 4);
        this.telf = resultSet.getString( columnIndex: 5);

    }
}
```

```
"C:\Program Files\Java\jdk-15\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\Int
El usuario se llama Jorge
Usuario(id=1, nombre=Pedro, apellido=Lara, mail=pedro@google.es, telf=111111111)
Usuario(id=2, nombre=Carmen, apellido=Sánchez, mail=carmen@google.es, telf=222222222)
Usuario(id=3, nombre=Alba, apellido=Cuevas, mail=pedro@google.es, telf=333333333)
Usuario(id=4, nombre=Jorge, apellido=Ríos, mail=pedro@google.es, telf=444444444)
Usuario(id=5, nombre=Marcos, apellido=López, mail=pedro@google.es, telf=555555555)

Process finished with exit code 0
```

> Propiedades de un JavaBiean

En JavaBeans podemos tener cuatro TIPOS DE PROPIEDADES

1, **Propiedades simples**

Es aquella propiedad que representa un valor único, esto es, aquella que tiene un getter y un setter, y el valor no cambiará

```
class Empleado { private String nombre;

        public getNombre(String nombre){

              this.nombre;
       }

       public setNombre(String nombre){
              this.nombre = nombre;
        }
}
```

ILERNA Online

2, **Propiedades indexadas**

Representan un array de valores a los que se accede mediante un índice.

```
private int[] categorias = {1, 2, 3};

public int[] getCategorias() {
    return categorias;
}

public void setCategorias(int[] categorias) {
    this.categorias = categorias;
}

public int getCategorias(int idx) {
    return categorias[idx];
}

public void setCategorias(int idx, int valor) {
    this.categorias[idx] = valor;
}
```

3, **Propiedades ligadas**: representan las propiedades asociadas a eventos.

Cuando la propiedad cambia se notifica a todos los objetos que están atentos a los mismos.

Para que el Bean soporte propiedades ligadas debe mantener una lista de los objetos que están atentos a su cambio y alertarlos.

Para ello proporciona una serie de métodos de la clase PropertyChangeSupport

> Propiedades Ligadas

ILERNA Online

```
public class BeanFuente implements Serializable {
    private PropertyChangeSupport propertySupport;
    private String nombre;

    public String getNombre() {
        return nombre;
    }
    public void setNombre(String nombre) {
        String oldNombre = this.nombre;
        String newNombre = nombre;
        this.nombre = nombre;
        propertySupport.firePropertyChange("nombre",oldNombre,newNombre);
    }
    public BeanFuente() {
        propertySupport = new PropertyChangeSupport(this);
    }
    public void addPropertyChangeListener(PropertyChangeListener listener) {
        propertySupport.addPropertyChangeListener(listener);
    }
    public void removePropertyChangeListener(PropertyChangeListener listener) {
        propertySupport.removePropertyChangeListener(listener);
    }
}
```

> Propiedades ligadas

```
public class BeanReceptor implements PropertyChangeListener {
    @Override
    public void propertyChange(PropertyChangeEvent evt) {
        System.out.println("Property: "+evt.getPropertyName());
        System.out.println("Valor anterior: "+evt.getOldValue());
        System.out.println("Valor nuevo: "+evt.getNewValue());
    }
}
```

> Propiedades ligadas

```
public static void main(String[] args){
    BeanFuente beanFuente = new BeanFuente();
    BeanReceptor beanReceptor = new BeanReceptor();
    beanFuente.addPropertyChangeListener(beanReceptor);
    beanFuente.setNombre(“Pedro");
}
```

4, **Propiedades restringidas**: Muy similares a las propiedades ligadas pero con la diferencia de que los objetos pueden ser vetados si no poseen unas características.

Persistencia de un JavaBean

Todos los Bean deben implementar la interfaz Serializable para poder recuperar su valor en cualquier momento determinado.
El bean se convierte en una cadena de bytes y se almacenan en un fichero desde donde se podrá reconstruir con posterioridad.

Todos los atributos exceptos los que son static o transient son serializados.