Acceso a Datos

TEMA 2

Acceso a datos

- Recuperación o manipulación de datos extraídos de un origen de datos local o remoto
- Orígenes:
  - Ficheros de texto
  - Ficheros binarios
  - **Bases de datos relacionales remotas**
  - **Bases de datos relacionales locales**
  - Bases de datos orientadas a objetos
- Para las bases de datos relacionales remotas desde Java necesitamos hacer uso de conectores

Tipos de bases de datos

BBDD Orientadas a Objetos

BBDD Relacionales

BBDD Orientadas a Objetos

- Más aceptación hoy en día
- Solucionan necesidades en el desarrollo de aplicaciones complejas
- Este tipo de bases de datos entran dentro del paradigma de la Programación Orientada a Objetos cuyos elementos complejos son los propios Objetos

BBDD Relacionales

- No están diseñadas para almacenar objetos directamente
- Se incrementa la complejidad ya hay que programar la conversión entre los **Objetos** y las **Tablas de la BBDD**
- Se puede resolver la complejidad de este desfase entre Objetos y Tablas haciendo un **mapeo** del modelo de datos

Bases de datos relacionales embebidas

- Cuando no se necesita utilizar una base de datos compleja como MySQL u Oracle, como por ejemplo en aplicaciones móviles
- El **motor** de la BBDD se encuentra **embebido** en la aplicación y es exclusivo de ella
- La BBDD **arranca al iniciar** la aplicación y se **termina cuando se cierre**

Bases de datos embebidas: SQLite

- Software **libre** y escrito en C
- La BBDD se guarda **junto a la aplicación**
- Existe una utilidad para ejecutar en modo **consola**
- Implementa la mayor parte del **estándar SQL-92**
- Se hace uso mediante llamadas a **funciones** en distintos lenguajes de programación

Bases de datos embebidas: Apache Derby

- Código abierto y desarrollada en Java
- Soporta estándares de SQL
- Incluye un controlador JDBC para incrustar Derby en cualquier programa basado en Java
- Soporta el paradigma cliente-servidor
- Fácil de instalar, implementar y utilizar

Bases de datos embebidas: Apache Derby

- Para realizar la instalación descargamos la última versión de la página http://db.apache.org/derby/derby_downloads.html
- En windows descargamos db-derby-10.10.2.0-bin.zip.
- ApacheDerby_Tutorial_Instalación http://db.apache.org/derby/papers/DerbyTut/install_software.html
- Para utilizar Derby en los programas Java, será necesario tener accesible la librería **derby.jar** añadiéndolo a nuestro proyecto en Eclipse.

Bases de datos embebidas: Apache Derby

- Apache Derby también trae una serie de ficheros .BAT que nos permitirán ejecutar órdenes para crear nuestras bases de datos y ejecutar sentencias DDL y DML.

- El fichero es IJ.BAT y se encuentra en la carpeta bin, para crear y acceder a las bases de datos ejecutaremos el fichero IJ.BAT

- Para crear una base de datos utilizaremos el siguiente comando en la línea de comandos IJ:

```
ij> connect 'jdbc:derby:<PATH>\<NOMBD>;create=true';
```

  Para salir de la línea de comandos IJ teclearemos `exit;`

Bases de datos embebidas:

HSQLDB (Hyperthreated Structured Query Language Database)

- Es un SGBD relacional escrito en Java.
- Distribuida bajo licencia BSD
- La Suite ofimática OpenOffice lo incluye en su versión 2.0 para dar soporte a la aplicación Base.
- Es compatible con SQL ANSI-92 y SQL:2008
- Puede mantener los datos en memoria o en ficheros en disco.
- Permite integridad referencial, procedimientos almacenados en Java, triggers y tablas en disco de hasta 8GB.

Bases de datos embebidas:

Instalación HSQLDB

- Desde la web http://sourceforge.net/projects/hsqldb/files/ podemos descargarnos la última versión
- En windows la última versión es hsqldb-2.5.0.zip
- Lo descomprimimos y dentro de la carpeta hsqldb-2.5.0 movemos la carpeta hsqldb a la raíz de una unidad por ejemplo C:\
- A continuación creamos una carpeta llamada ejemplo en `C:\db\hsqldb\ejemplo\` para nuestra BD
- Abrimos el CMD y ejecutamos el fichero:

```
C:\hsqldb\bin\runUtil DatabaseManager
```

Bases de datos embebidas:

HSQLDB

- Se abrirá una ventana desde la que configuraremos la conexión.
- Escribirnos en el campo Setting Nameun nombre para la conexión.
- En la lista Type seleccionamos la opciónHSQLDatabase Engine Standalone, para que la BD la tome un fichero si existe y si no existe la cree.
- En la casilla de URL escribimos el nombre de la carpeta donde se almacenará la base de datos y el de la base de datos: ejemplo/ejemplo. Pulsamos OK.

Bases de datos embebidas:

HSQLDB

- Se abrirá una ventana desde la que configuraremos la conexión.
- A continuación se abre una nueva ventana desde la que podemos ejecutar comandosDDL y DML para crear manipular objetos de nuestra BD.

- Para ejecutar una sentencia SQL pulsamos el botón Execute.

- Desde la opción de menú View -> Refresh Tree podemos actualizar el árbol de objetos .

Bases de datos embebidas:

HSQLDB ACTIVIDAD

Instala HSQLDB and crea una BD denominada ejemplo.
Una vez creada, con SQL crea una tabla llamada paciente con los siguientes campos:
-patient_num integer y será primary key
-name string de 15 caracteres
-address string de 15 caracteres
-phone number

Inserta filas como esta:

435120, John Green, 75 Ninth Avenue 3th, Boston, (212) 630-0321

Bases de datos embebidas:

HSQLDB ACTIVIDAD

Crea una tabla llamada cita con los siguientes campos:
-num_cita numerical and will be primary key
-date
-time
-doctor_id numerical
-patien_no numerical and foreign key of patient

Inserta filas como esta:

1, 12/12/2012, 9:00, 45, 435120

Bases de datos embebidas: H2

- Es un SGBD relacional programado en Java.
- Está disponible bajo la licencia pública de mozilla o la eclipse public licenses.
- Desde la web: http://www.h2database.com/html/download.html podemos descargarnos la última versión.
- Descargamos la versión zip, para todas las plataformas h2-2019-10-14.zip
- Descomprimimos por ejemplo en C:\ y creará una carpeta con el nombre C:\h2
- Desde el cmd ejecutaremos el fichero C:\h2\bin\h2.bat, para arrancar la consola.

Bases de datos embebidas: H2

- Se abrirá en el navegador web la consola de adm. de H2
- Escribimos un nombre para la configuración de la BD: **jdbc:h2:C:/db/H2/ejemplo/ejemplo** (si las carpetas no existieran las crea).
- Pulsar el botón guardar y después el botón conectar.
- Para conectarnos a la BD utilizaremos el nombre que utilizamos en la configuración.
- Comprobamos la configuración pulsando el botón probar la conexión.
- Se creará la carpeta ejemplo en H2 y dentro los ficheros de nuestra BD.

Bases de datos embebidas: H2

- Una vez conectados se visualiza una nueva pantalla.
- Desde esta podremos realizar operaciones sobre la BD.
- Se muestran los comandos más importantes.
- Desde la zona de instrucciones SQL podremos escribir las sentencias para crear tablas, insertar filas, etc.
- Los botones **Ejecutar** (CTRL+ENTER) y |> nos permitirán ejecutar las sentencias SQL que escribamos en el área de instrucción SQL.
- El botón Desconectar nos lleva a la pantalla inicial donde elegimos la conexión.

Bases de datos embebidas: H2

En el enlace **Preferencias** de la ventana inicial se pueden configurar diversos aspectos como: los clientes permitidos (locales/remotos), conexión segura (SSL), puerto del servidor Web o notificar las sesiones activas.

La opción **Tools** presenta una serie de herramientas que se pueden utilizar sobre la base de datos: backup, restaurar base de datos, ejecutar scripts, convertir la base de datos en un script, encriptación, etc.

Bases de datos embebidas: H2

H2 ACTIVIDAD

Instala H2. Crea una BD llamada ***ejemplo***.
Crea una tabla denominada ***borrowed*** con los siguientes campos:

- Borrower_ID, numérico, primary key
- Name, String de 20 caracteres
- Department, String de 20 caracteres
- Status, String de 20 caracteres
- Telephone, String de 14 caracteres

Inserta filas como estas:

1, John Green, Computer and Automatic, Professor (212) 630-0321
2, Martin McFly , Computer and Automatic, Professor, (212) 435

Bases de datos embebidas: H2

H2 ACTIVIDAD

Crea una tabla llamada ***book*** con los siguientes campos:

- Book_Id numerical, primary key
- Title, String de 20 caracteres
- Authors, String de 20 caracteres
- Year, numerical
- Publisher, String de 20 caracteres
- Borrower_ID, foreign key of borrower

Inserta filas como estas:

1, "The Art of Computer Programming", Donald Knuth, 2011, McGrawHill, 1
2, "Compilers: Principles, Techniques, and Tools", Aho, et al., 2012, Prentice Hall, 1
3, "TCP/IP Illustrated", W. Richard Stevens, 2012, Penguin, 2

Bases de datos embebidas: DB4O

DB4O (DataBase for Objects)

- Es un motor de BD orientado a objetos.
- Se puede utilizar de forma embebida o en aplicaciones cliente-servidor. Está disponible para entornos Java y .Net-
- Dispone de licencia dual GPL/comercial
- Algunas características interesantes son:
  - Evita el problema del desfase objeto-relacional.
  - No existe un lenguaje SQL para la manipulación de datos, en lugar existen métodos delegados.
  - Se instala añadiendo un único fichero de librería (JAR para Java o DLLpara .NET). Se crea un único fichero .YAP con la BD.

Bases de datos embebidas: DB4O

Instalación DB4O (DataBase for Objects)

Si la página web (http://www.db4o.com) está caída, sus desarrolladores han habilitado un enlace para su descarga (db4o 8.0-java.zip) https://sourceforge.net/projects/db4o/

Dentro creará una carpeta lib donde se encuentra los JAR (db4o-.jar) que tenemos que agregar a nuestro proyecto para utilizar el motor de la BD.

El archivo JAR más importante es db40-8.0.224.15975.core-java5.jar

Bases de datos embebidas: DB4O

Agregar librería con todos los db4o-,jar en Eclipse

Seleccionar nuestro proyecto y pulsar con el botón derecho del ratón
Seleccionar Build Path->Add Library y después seleccionamos User Library pulsamos el botón Next.
A continuación se visualizará una nueva ventana desde la que pulsamos el botón User Libraries.
Pulsamos el botón New para dar un nombre a la librería, por ejemplo Db4oLib.
Después pulsamos el botón Add External JARs..., localizamos todos los db4o-.jar de la carpeta lib y pulsamos el botón Abrir
A continuación OK y para finalizar pulsamos Finish.

Bases de datos embebidas: DB4O

Clase Person

Vamos a crear una clase **Person** para crear una BD de **objetos Person** en ella:

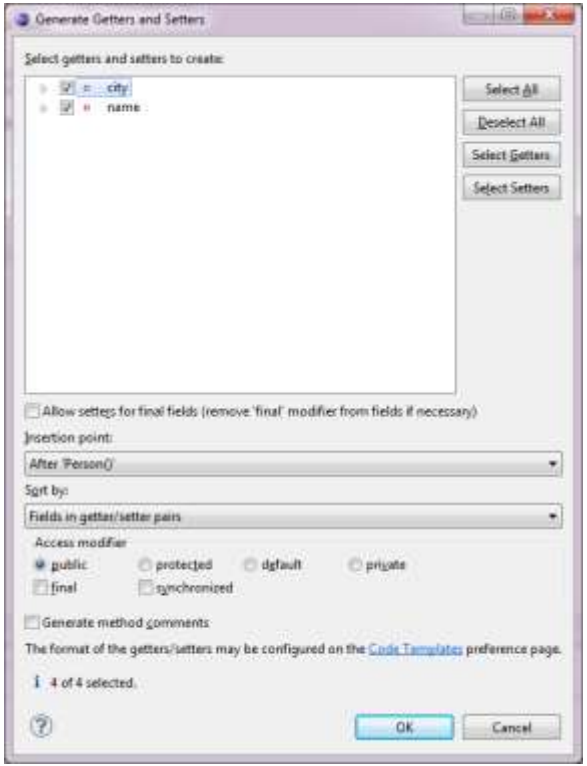
```
public class Person {
    private String name;
    private String city;

    public Person(String name, String city) {
        this.name = name;
        this.city = city;
    }

    public Person() {
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getCity() {
        return city;
    }

    public void setCity(String city) {
        this.city = city;
    }

}
```

Bases de datos embebidas: DB4O

Truco

Desde eclipse para generar automáticamente los getters y los setters de los atributos pulsamos el botón derecho del ratón en el código de la clase seleccionamos la opción Source y luego Generate Getters and Setters.

A continuación seleccionamos los atributos y pulsamos Ok



Bases de datos embebidas: DB4O

Ejemplo DB4O

El siguiente código muestra la clase **ExampleDb4o.java** que crea la BD (si no existe) y añade objetos Person en ella.

```
import com.db4o.Db4oEmbedded;
import com.db4o.ObjectContainer;

public class ExampleDb4o {

    final static String BDPer = "./../db4o-8.0/DBPeople.yap";

    public static void main(String[] args) {

        ObjectContainer db = Db4oEmbedded.openFile(
                Db4oEmbedded.newConfiguration(), BDPer);
        // Create people
        Person p1 = new Person("Juan", "Guadalajara");
        Person p2 = new Person("Ana", "Madrid");
        Person p3 = new Person("Luis", "Granada");
        Person p4 = new Person("Pedro", "Asturias");
        // Store objects Person in the data
        db.store(p1);
        db.store(p2);
        db.store(p3);
        db.store(p4);

        db.close();

    } //end main
} // end ExampleDb4o
```

Bases de datos embebidas: DB4O

Leer registros de la base de datos DB4O

Para realizar cualquier acción en la BD debemos manipular una instancia de **ObjectContainer** donde se define el fichero de BD

```
Person person = new Person(null, null);

ObjectContainer db = Db4oEmbedded.openFile(
        Db4oEmbedded.newConfiguration(), BDPer);

// run through the database

ObjectSet<Person> result = db.queryByExample(person);
if (result.size() == 0)
    System.out.println("there aren't any register of person...");
else {
    System.out.println("Number of registers: " + result.size());
    while (result.hasNext()) {
        Person p = result.next();
        System.out.println("Name: " + p.getName() + " city: "
                + p.getCity());
    }

}
```

Bases de datos embebidas: DB4O

Buscar un registro de la base de datos DB4O

```
// Obtain objects person with name Juan
Person p1 = new Person("Juan", null);
ObjectSet<Person> result1 = db.queryByExample(p1);

if (result1.size() != 0) {
    person = (Person) result1.next();
    System.out.println("Name: " + person.getName());
} else
    System.out.println("No existe Juan...");

// Obtain objects Person with citiy Guadalajara
Person p2 = new Person(null, "Guadalajara");
ObjectSet<Person> result2 = db.queryByExample(p2);

if (result2.size() != 0) {
    person = (Person) result2.next();
    System.out.println(" city: " + person.getCity());
} else
    System.out.println("No existe Guadalajara...");
```

Bases de datos embebidas: DB4O

Modificar un registro de la base de datos DB4O

```
// Modify an object
// Look for the object

ObjectSet<Person> result3 = db.queryByExample(new Person("Juan", null));
if (result3.size() == 0)
    System.out.println("No existe Juan...");
else { // modify the city
    person = (Person) result3.next();
    person.setCity("Toledo");
    db.store(person); // modified city
    // show data
    result3 = db.queryByExample(new Person("Juan", null));
    person = (Person) result3.next();
    System.out.println("Name: " + person.getName() + "New city: "
            + person.getCity());
}
```

Bases de datos embebidas: DB4O

Eliminar un registro de la base de datos DB4O

```
// Delete an object
ObjectSet<Person> result4 = db.queryByExample(new Person("Juan", null));
if (result4.size() == 0)
    System.out.println("No existe Juan...");
else { // modify the city
    person = (Person) result4.next();
    System.out.println("Registerts to delete: " + person.getName()
            + " " + person.getCity());
    if (result4.size() > 1) {// many Juan registers
        while (result4.hasNext()) {
            person = result.next();
            db.delete(person);
            System.out.println("Borrado");

        }
    } else db.delete(person);
}
```

Bases de datos embebidas: DB4O

Actividad

Crea una BBDD Db4o de nombre *EMPLEDEP.YAP* e inserta objetos EMPLEADOS y DEPARTAMENTOS en ella.

Después obtén todos los objetos empleados de un departamento concreto. Visualiza también el nombre completo de dicho departamento.

Bases de datos embebidas: FireBird

- Se deriva de código de InterBase 6.0 de Borland.
- Es un sistema gestor de bases de datos relacional y de código totalmente abierto.
- Se puede usar tanto en aplicaciones comerciales como de código abierto.
- Se presenta en 3 versiones de servidor SuperServer, Classic y Embedded.
- Es fácil distribuir las aplicaciones ya que no requieren instalación.
- Se suele usar para crear catálogos en CD-ROM.

Bases de datos embebidas: FireBird

Características:

- Completo soporte para los procedimientos almacenados y disparadores.
- Integridad referencial.
- Tiene un bajo consumo de recursos
- Lenguaje interno para los procedimientos almacenados y disparadores.
- Soporte para funciones externas.
- Escasa necesidad de administradores especializados.
- Múltiples formas de acceder a la base de datos.

Bases de datos embebidas: SQL Server Compact

Es una base de datos compacta y gratuita para que los desarrolladores compilen sitios web ASP.NET y aplicaciones de escritorio de Windows.

Ocupa poco espacio, es compatible con la implementación privada de binarios dentro de la carpeta de la aplicación y permite una fácil implementación en Visual Studio y la migración de conexión directa de esquemas y datos a SQL Server.

Tiene gran variedad de funciones y está diseñada para un gran número de tabletas y de dispositivos móviles inteligentes.

Incluye varias características de las bases de datos relacionales.

http://msdn.microsoft.com/es es/sqlserver/default.aspx
https://www.microsoft.com/es-es/download/details.aspx?id=17876

Bases de datos embebidas: SQL Server Compact

Características:

- Tiene un motor de bases de datos compacto y un sólido optimizador de consultas.
- Permite el acceso a datos remoto.
- Integrado con Microsoft SQL Server.
- Integrado con Microsoft Visual estudio.
- Es compatible con Microsoft ADO Net.
- Es un conjunto de sintaxis SQL.
- Es compatible con la tecnología de ClickOnce.

Bases de datos embebidas: Oracle Berkeley DB

Oracle Berkeley DB, nos proporciona una base de datos embebida que permite varias opciones de almacenamiento. Sus características son:

- Ejecución en memoria o disco con alto rendimiento.
- Rendimiento extraordinario, eliminando los gastos de comunicación interprocesos y SQL.
- La API se encarga de la administración, integrándose por completo en la aplicación y siendo invisible para los usuarios
- Para dispositivos móviles ocupa muy poco (<1MB) y el tiempo de ejecución en memoria dinámica es muy pequeño.
- Bajo coste total. Al ser embebida no gasta en licencias, administración, hardware o servidor.

https://www.oracle.com/database/technologies/related/berkeleydb.html

Bases de datos embebidas: Oracle Berkeley DB

Berkeley DB es una familia de bibliotecas de bases de datos de valores clave integradas que proporcionan servicios escalables de gestión de datos de alto rendimiento para aplicaciones.

Utilizan API de llamadas a funciones simples para el acceso y la gestión de datos.

Berkeley DB proporciona una colección de tecnologías de bloques de construcción bien probadas que se pueden configurar para abordar cualquier necesidad de aplicación desde el dispositivo portátil hasta el centro de datos, desde una solución de almacenamiento local hasta una distribuida en todo el mundo, desde kilobytes hasta petabytes.