Centro Asociado Palma de Mallorca

Introducción Práctica de Programación Java

Antonio Rivero Cuesta

Sesión XI

Java JDBC

JDBC

Java DataBase Connectivity

JDBC

Es un driver que conecta una aplicación con un Sistema Gestor de Bases de Datos.

Funciones:

- Establece conexión con la BBDD.
- Permite manipular la BBDD.

Conexión

Para conectar con la BBDD, necesitamos importar las siguientes librerías:

```
import Java.sql.*;
import Javax.sql.*;
```

Fases del acceso a BBDD

1. Establecer conexión con la BBDD.
2. Crear un objeto de tipo Statement.
3. Ejecutar una sentencia SQL.
4. Leer el Resultset.

Establecer conexión con BBDD

Creamos un objeto de tipo **`Connection`**.

Creamos un **`String`** para almacenar la cadena de conexión.

`jdbc:`	**`mysql:`**	**`//localhost/base.sql`**
`driver`	**`Protocolo driver`**	**`Detalles conexión`**

Necesitaremos un:

- Un Usuario.
- Una Contraseña.

```
private String usuario = "root";
private String pass = "";
```

Proteger la conexión con un **`try - catch`**

Crear un objeto de Statement

Una vez que tenemos el objeto **`Connection`**.

Usamos el método **`createStatement`**.

`createStatement` nos devuelve un objeto de tipo **`Statement`**.

Ejecutar una Sentencia SQL

Con el objeto **`Statement`** creado, usamos el método **`executeQuery(sql)`** que nos devolverá un **`Resultset`**.

Un **`Resultset`** es un objeto en donde se almacena la información de la consulta SQL.

Utilización de los tres métodos de ejecución de sentencias SQL, ofrecidos por el interfaz Statement.

executeQuery()	SELECT	Resultset
executeUpdate()	INSERT - DELETE UPDATE - CREATE	String Int
execute()	Sentencias desconocidas en tiempo de compilación o sentencias que devuelven resultados complejos	true - false

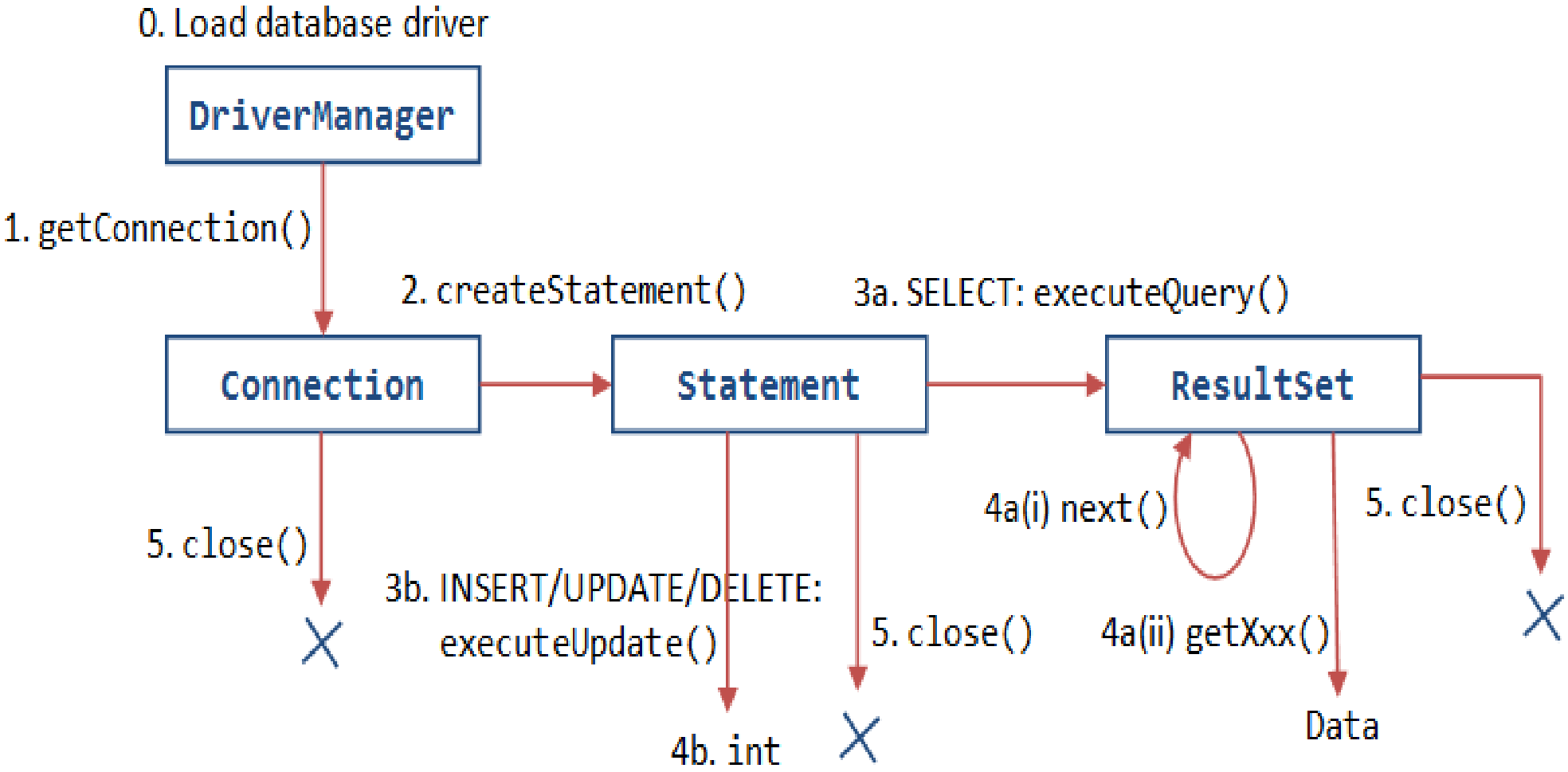
Leer el Resultset

Tenemos los métodos:

- **`getString()`**
- **`getInt`**
- **`next()`**

Recorreremos el **Resultset** línea a línea con un bucle **While**.

```
while (rs.next()) {
   System.out.println("DNI:"+rs.getInt(1));
   System.out.println("Nombre:"+rs.getString(2));
   System.out.println("Apellidos:"+rs.getString(3));
}
```


0. Load database driver
DriverManager
1. getConnection()
2. createStatement()
3a. SELECT: executeQuery()
Connection
Statement
ResultSet
5. close()
3b. INSERT/UPDATE/DELETE:
executeUpdate()
5. close()
4a(i) next()
4a(ii) getXxx()
5. close()
4b. int
Data

Interface PreparedStatement

Nos permite ejecutar sentencias SQL sobre una conexión establecida con una base de datos.

Ejecutaremos sentencias SQL más especializadas.

Se van a denominar sentencias SQL precompiladas y van a recibir parámetros de entrada.

La interfaz PreparedStatement hereda de la interfaz Statement.

Tiene dos diferencias:

Las instancias de PreparedStatement contienen sentencias SQL que ya han sido compiladas.

Esto es lo que hace a una sentencia "***prepared***" *(preparada).*

La sentencia SQL que contiene un objeto PreparedStatement puede contener uno o más parámetros de entrada.

Un parámetro de entrada es aquél cuyo valor no se especifica cuando la sentencia es creada.

En su lugar la sentencia va a tener un signo de interrogación (?) por cada parámetro de entrada.

Antes de ejecutarse la sentencia se debe especificar un valor para cada uno de los parámetros a través de los métodos setXXX apropiados.

Estos métodos setXXX los añade el interfaz PreparedStatement.

Debido a que las sentencias de los objetos PrepareStatement están precompiladas su ejecución será más rápida que la de los objetos Statement.

Por lo tanto, una sentencia SQL que va a ser ejecutada varias veces se suele crear como un objeto PreparedStatement para ganar en eficiencia.

También se utilizará este tipo de sentencias para pasarle parámetros de entrada a las sentencias SQL.

Al heredar de la interfaz Statement, la interfaz PreparedStatement hereda todas funcionalidades de Statement.

Los métodos:

- execute()
- executeQuery()
- executeUpdate()

Están sobrecargados y para los objetos PreparedStatement no toman ningún tipo de argumentos.

De esta forma, a estos métodos nunca se les deberá pasar por parámetro el objeto String que representaba la sentencia SQL a ejecutar.

En este caso, un objeto PreparedStatement ya es una sentencia SQL por sí misma.

A diferencia de lo que ocurría con las sentencias Statement que poseían un significado sólo en el momento en el que se ejecutaban.

Métodos PreparedStatement

- boolean execute(): ejecuta cualquier tipo de sentencia SQL.
- ResultSet executeQuery(): ejecuta la sentencia SQL representada por el objeto PreparedStatement, devolviendo un objeto ResultSet con el resultado de la ejecución de la consulta.
- int executeUpdate(): ejecuta una sentencia SQL del tipo INSERT, UPDATE o DELETE.

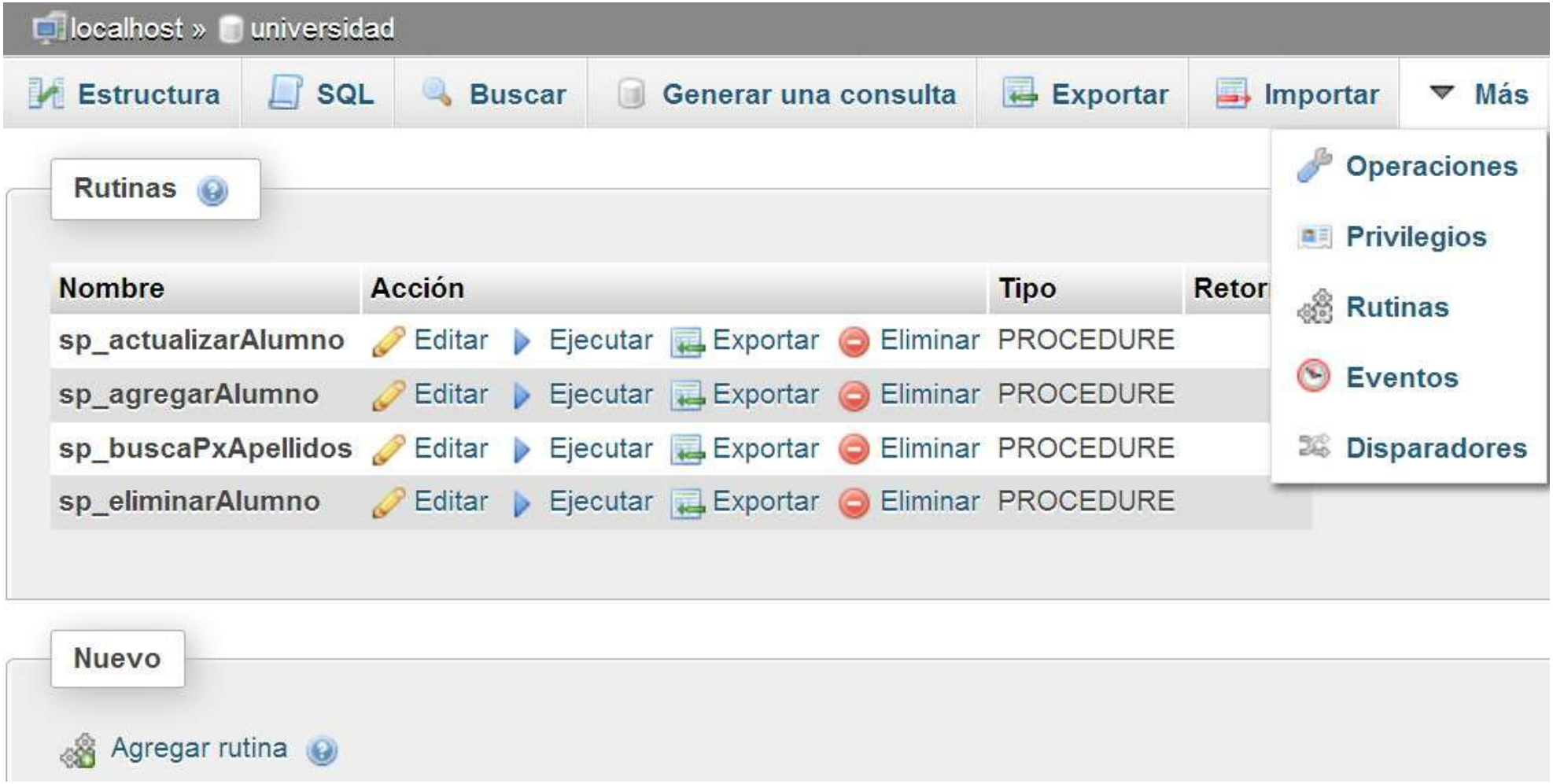
Interface CallableStatement

El último tipo de sentencias que podemos utilizar en JDBC son las sentencias CallableStatement.

Esta interfaz hereda del interfaz PreparedStatement.

Ofrece la posibilidad de:

- Manejar parámetros de salida.
- Realizar llamadas a procedimientos almacenados de la base de datos.


localhost » universidad
Estructura
SQL
Buscar
Generar una consulta
Exportar
Importar
Más
Operaciones
Privilegios
Rutinas
Eventos
Disparadores
Rutinas
Nombre
Acción
Tipo
Retor
sp_actualizarAlumno Editar Ejecutar Exportar Eliminar PROCEDURE
sp_agregarAlumno Editar Ejecutar Exportar Eliminar PROCEDURE
sp_buscaPxApellidos Editar Ejecutar Exportar Eliminar PROCEDURE
sp_eliminarAlumno Editar Ejecutar Exportar Eliminar PROCEDURE
Nuevo
Agregar rutina

Detalles

Nombre de rutina sp_agregarAlumno

Tipo PROCEDURE

Parámetros

Dirección	Nombre	Tipo	Longitud/Valores	Opciones	
IN	pdni	C	8	Juego	Eliminar
IN	pnombr	V	45	Juego	Eliminar
IN	papellid	V	45	Juego	Eliminar
IN	pfechaN	V	45	Juego	Eliminar
IN	pdirecci	V	45	Juego	Eliminar

Agregar parámetro

Definición

```
1 BEGIN
2 insert into alumnos(dni,nombre,apellidos,fechaN,d
3 values (pdni,pnombre,papellidos,pfechaN,pdireccio
4
5 END
```

Es determinístico

Definidor root@localhost

Tipo de seguridad DEFINER

Acceso de datos SQL CONTAINS SQL

Comentario

Un procedimiento almacenado se encuentra dentro de una base de datos.

La llamada a un procedimiento es lo que contiene un objeto CallableStatement.

Esta llamada está escrita con una sintaxis de escape.

Sintaxis CallableStatement

Para realizar la llamada a un procedimiento almacenado:

```
{call nombre_del_procedimiento[(?,?,...)]}
```

```
{call sp_agregarAlumno(?,?,?,?,?)}
{call sp_eliminarAlumno(?)}
{call sp_actualizarAlumno(?,?,?,?,?)}
{call sp_buscaPxApellidos(?)}
```

Si devuelve un parámetro de resultado:

```
{?=call nombre_del_procedimiento[(?.?...)]}
```

La sintaxis de una llamada a un procedimiento sin ningún tipo de parámetros sería:

```
{call nombre_del_procedimiento}
```