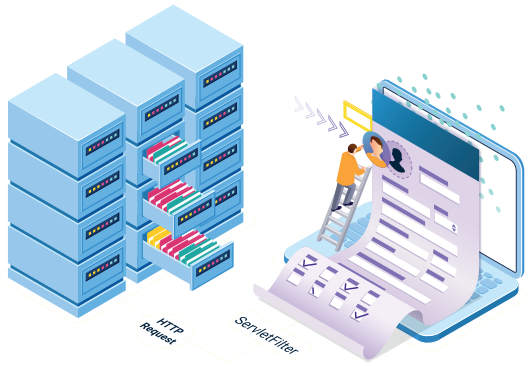
Clases e interfaces Java

Construcción de aplicaciones con JAVA



Además de tener a disposición el Driver JDBC correspondiente, para la base de datos que se desea conectar, se requiere de un conjunto adicional de clases e interfaces Java para hacer uso del driver, para lo cual es necesario importar los paquetes java.sql o javax.sql.

A continuación, se describen algunas de las clases más importantes dentro de estas librerías para gestionar correctamente una base de datos por medio de JDBC.

DriverManager:

Es una clase estática que permite la gestión del driver JDBC que esté disponible para la aplicación. Por medio del método getConnection crea una instancia de tipo Connection mediante la cual se realiza el flujo de envío y recibo de información hacia y desde la base de datos destino.

La clase DriverManager debe estar informada de los controladores JDBC disponibles, es decir que adicional a descargar el Driver JDBC y agregarlo al proyecto JAVA, es necesario registrar wdicho Driver para que sea cargado en memoria y pueda ser reconocido por el DriverManager. Para hacer este proceso se utiliza la siguiente sentencia:

```
Class.forName("string");
```

El parámetro esperado por la sentencia anterior que está marcado en rojo debe corresponder con el nombre de la clase que va a ser cargada e instanciada, teniendo en cuenta que previamente ya se ha descargado el Driver correspondiente y que fue anexado al proyecto. Para el caso de un Driver JDBC para MySQl deberá usar el comando:

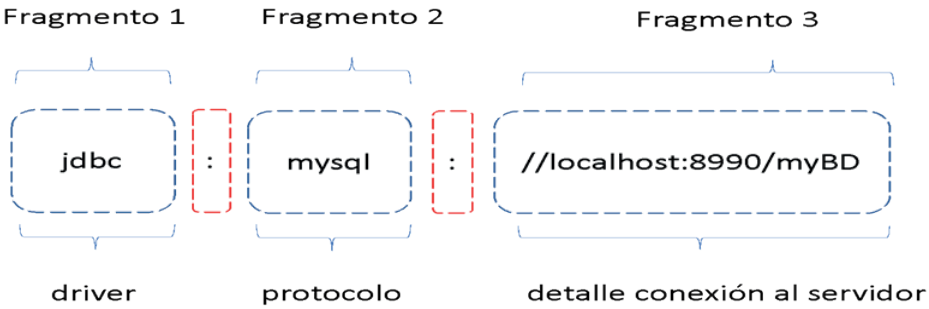
```
Class.forName("com.mysql.cj.jdbc.Driver");
```

Connection:

Es una interfaz Java que permite establecer una conexión con una base de datos específica por medio de la cual se puede extraer información de la descripción de tablas, procedimientos almacenados y ejecutar sentencias SQL. Se instancia un objeto de este tipo invocando el método getConnection del DriverManager al cual se le debe pasar como parámetros las credenciales (usuario y contraseña) del usuario de base de datos destino y la cadena de conexión correspondiente.

La cadena de conexión es un *String* que tiene un formato específico de tres fragmentos conformado por el nombre del driver, el protocolo del driver y los detalles de conexión al servidor de base de datos. Adicionalmente se requieren las credenciales de acceso al servidor.

Figura 1 Cadena de conexión



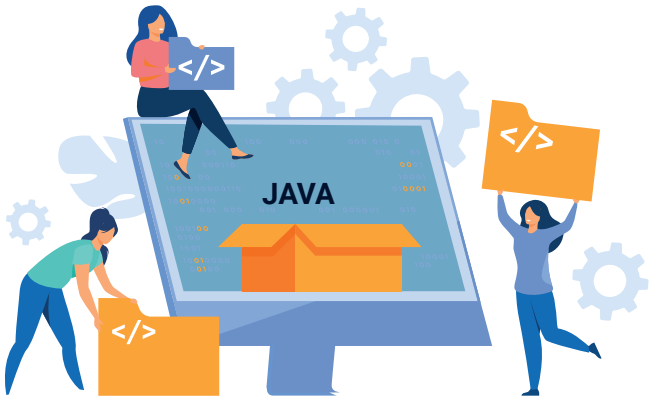
Un ejemplo para crear un objeto conexión a una base de datos Mysql es el siguiente:

```
String usuario = "root";
String password = "123@S";
String url = "jdbc:mysql://localhost:3306/prueba";
...
Connection conexion =
DriverManager.getConnection(url,usuario,password);
```

Statement:

Es una interfaz java usada para la ejecución de sentencias SQL estáticas y retornar los resultados que son producidos por estas sentencias. Una instancia de este tipo se obtiene de la ejecución del método createStatement() desde una instancia de un objeto de tipo Connection.

```
Statement st = conexion.createStatement();
```



Una instancia de tipo Statement puede invocar un gran conjunto de métodos disponibles para la ejecución de sentencias SQL, la tabla 2 muestra algunos de los métodos más utilizados:

Tabla 1 Métodos comúnmente usados de un Statement.

Método y parámetros	Descripción	Retorno
execute(String sql)	Ejecuta la sentencia SQL entregada como parámetro la cual puede retornar múltiples resultados	Boolean
executeQuery(String sql)	Ejecuta la sentencia SQL entregada como parámetro y retorna un objeto de tipo ResultSet con los resultados.	ResultSet
executeUpdate(String sql)	Ejecuta la sentencia SQL entregada como parámetro la cual puede ser de tipo INSERT, UPDATE o DELETE o una sentencia SQL que no tiene retorno como por ejemplo una sentencia DDL	int

ResultSet:

Es un tipo especial de dato que representa un conjunto de datos de tipo tabular el cual es ideal para referenciar los resultados obtenidos por medio de una consulta que retorna un conjunto de datos estructurados como por ejemplo los de una consulta SELECT.

Un ResultSet es un apuntador de tipo Cursor que está direccionado a la primera posición de toda la estructura tabular que está referenciando.

Figura 2 ResultSet

ResultSet:

123001	Jose Miguel	Perez Torres	Admin
123002	Maria Jose	Muñoz Trujillo	User
123003	Eduardo	Ante Garcia	User
123004	Michael Steven	Cabezas Paz	Salesman

Conjunto de datos obtenidos desde una sentencia SELECT

El método next()::

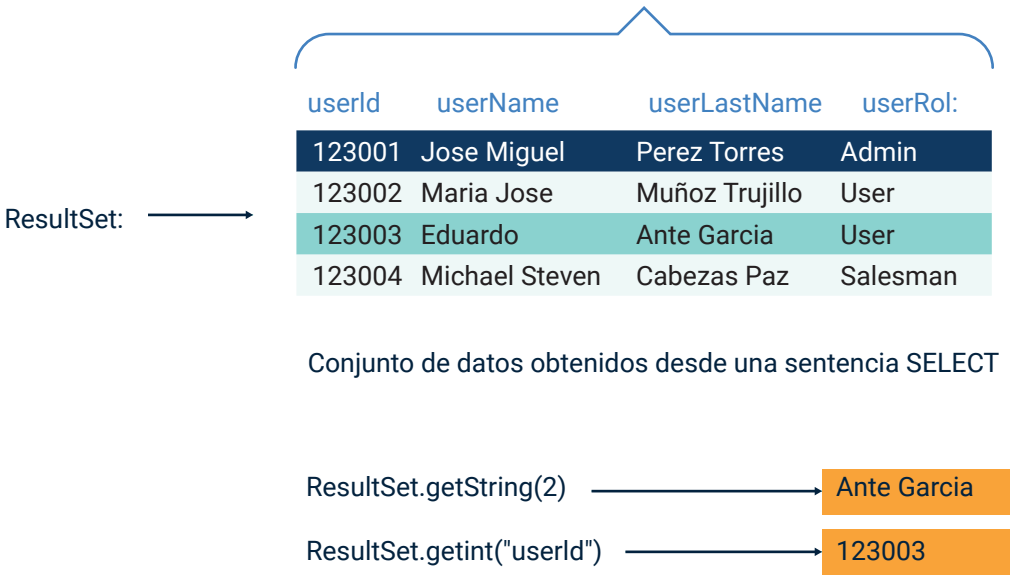
Además de desplazarse una posición de forma lineal dentro de la estructura también devuelve como resultado un valor Booleano para informar si el desplazamiento realizado encontró o no un conjunto de datos válidos, así es posible entonces determinar cuándo se alcanza el final de los elementos. Algunos métodos útiles del ResultSet se muestran en la tabla 3.

Tabla 2 Métodos comúnmente usados de un ResultSet

Método	Descripción	Retorno
isFirst()	Indica si el cursor se encuentra en la primera fila del objeto ResultSet	Boolean
isLast()	Indica si el cursor se encuentra en la última fila del objeto ResultSet	Boolean
first()	Mueve el cursor a la primera fila del objeto ResultSet	Boolean
last()	Mueve el cursor a la última fila del objeto ResultSet	Boolean
next()	Mueve el curso a la fila siguiente del objeto ResultSet	Boolean
previous()	Mueve el cursor a la fila previa del objeto ResultSet	Boolean
getInt(String columnLabel)	Obtiene el valor de la columna especificada como parámetro para la fila en la que actualmente está posicionado el cursor, obtiene el valor en formato Int por lo que se requiere que el valor que se encuentre en la columna corresponda con un valor Int.	Int
getInt(int columnIndex)	Obtiene el valor de la columna especificada como parámetro tomando como referencia el número de su posición dentro de la estructura para la fila en la que actualmente está posicionado el cursor, obtiene el valor en formato Int por lo que se requiere que el valor que se encuentre en la columna corresponda con un valor Int.	Int
getDouble(String columnLabel)	Obtiene el valor de la columna especificada como parámetro para la fila en la que actualmente está posicionado el cursor, obtiene el valor en formato Double por lo que se requiere que el valor que se encuentre en la columna corresponda con un valor Double.	Double
getDouble(int columnIndex)	Obtiene el valor de la columna especificada como parámetro tomando como referencia el número de su posición dentro de la estructura para la fila en la que actualmente está posicionado el cursor, obtiene el valor en formato Double por lo que se requiere que el valor que se encuentre en la columna corresponda con un valor Double.	Double
getString(String columnLabel)	Obtiene el valor de la columna especificada como parámetro para la fila en la que actualmente está posicionado el cursor, obtiene el valor en formato String por lo que se requiere que el valor que se encuentre en la columna corresponda con un valor String.	String
getString(int columnIndex)	Obtiene el valor de la columna especificada como parámetro tomando como referencia el número de su posición dentro de la estructura para la fila en la que actualmente está posicionado el cursor, obtiene el valor en formato String por lo que se requiere que el valor que se encuentre en la columna corresponda con un valor String.	String

Statement:

Figura 2 Conjunto de datos obtenidos desde una sentencia SELECT



Es común encontrar estructuras cíclicas apoyadas en el método next para realizar el recorrido completo y manipulación de los elementos obtenidos con un ResultSet. A continuación, se muestra un ejemplo:

```
resultSet.next();
do{
    ...
    //Acciones para procesar cada una de las filas

    System.out.println(resultSet.getInt("userid")+" : "+
resultSet.getString("username"));

}while(resultSet.next());
```