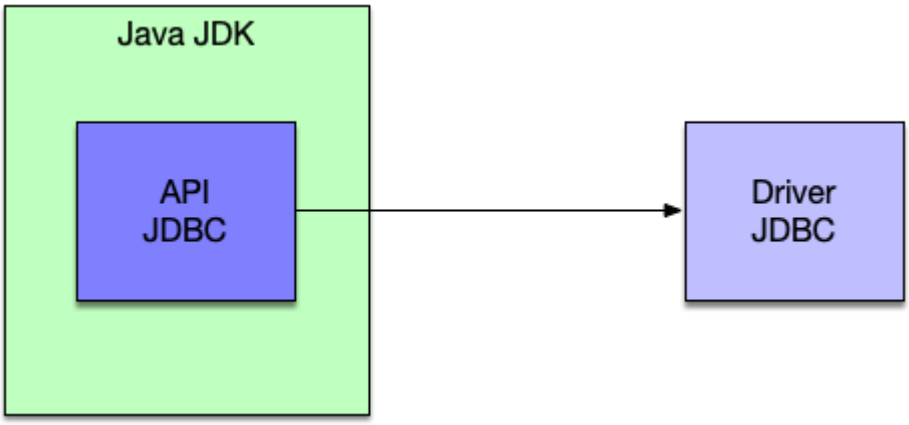
Tabla de Contenidos

El concepto de JDBC Driver es uno de los conceptos clave cuando empezamos a trabajar con Java y una base de datos. ¿Para qué sirve exactamente un Driver JDBC? .Muchas veces los desarrolladores que estan empezando no entienden de forma correcta como utilizarlo. Vamos a explicarlo de forma sencilla.



Vamos mostrar el código de la operación más sencilla qué solemos hacer con JDBC .

```
package es.avalon.jdbc;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;

public class Principal {

        public static void main(String[] args) {
```

```
                Connection conexion;
                String url="jdbc:mysql://localhost:3306/biblioteca";
                String usuario="root";
                String clave="";
                String consulta="insert into Libros
(isbn,titulo,autor,precio,categoria) values
('5','net','juan',20,'web')";
                try {
        Class.forName("com.mysql.jdbc.Driver");
        conexion=DriverManager.getConnection(url,usuario,clave);
        Statement sentencia=conexion.createStatement();
                        sentencia.execute(consulta);
                } catch (Excepcion e) {
                        e.printStackTrace();
                }
        }

}
```

Se trata de un insert en la base de datos. Para ello es necesario contar con dos objetos Connection y ResultSet.

La pregunta más habitual es ¿Qué tenemos que importar ?

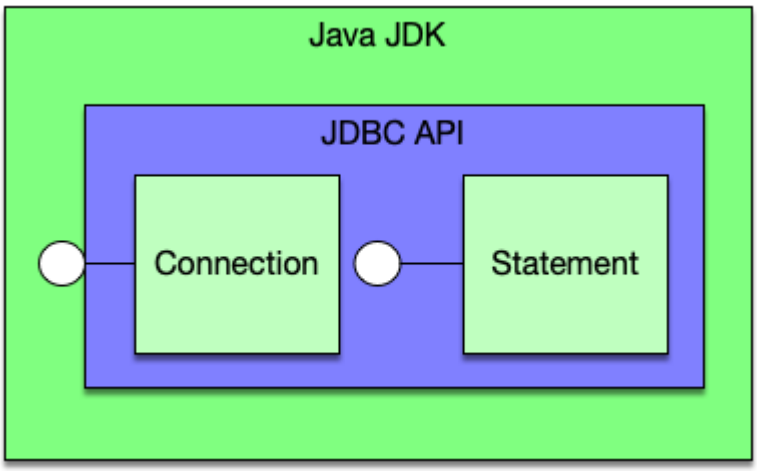
java.sql.Connetion y java.sql.Statement

o por el contrario:

com.mysql.Connection y com.mysql.Statement

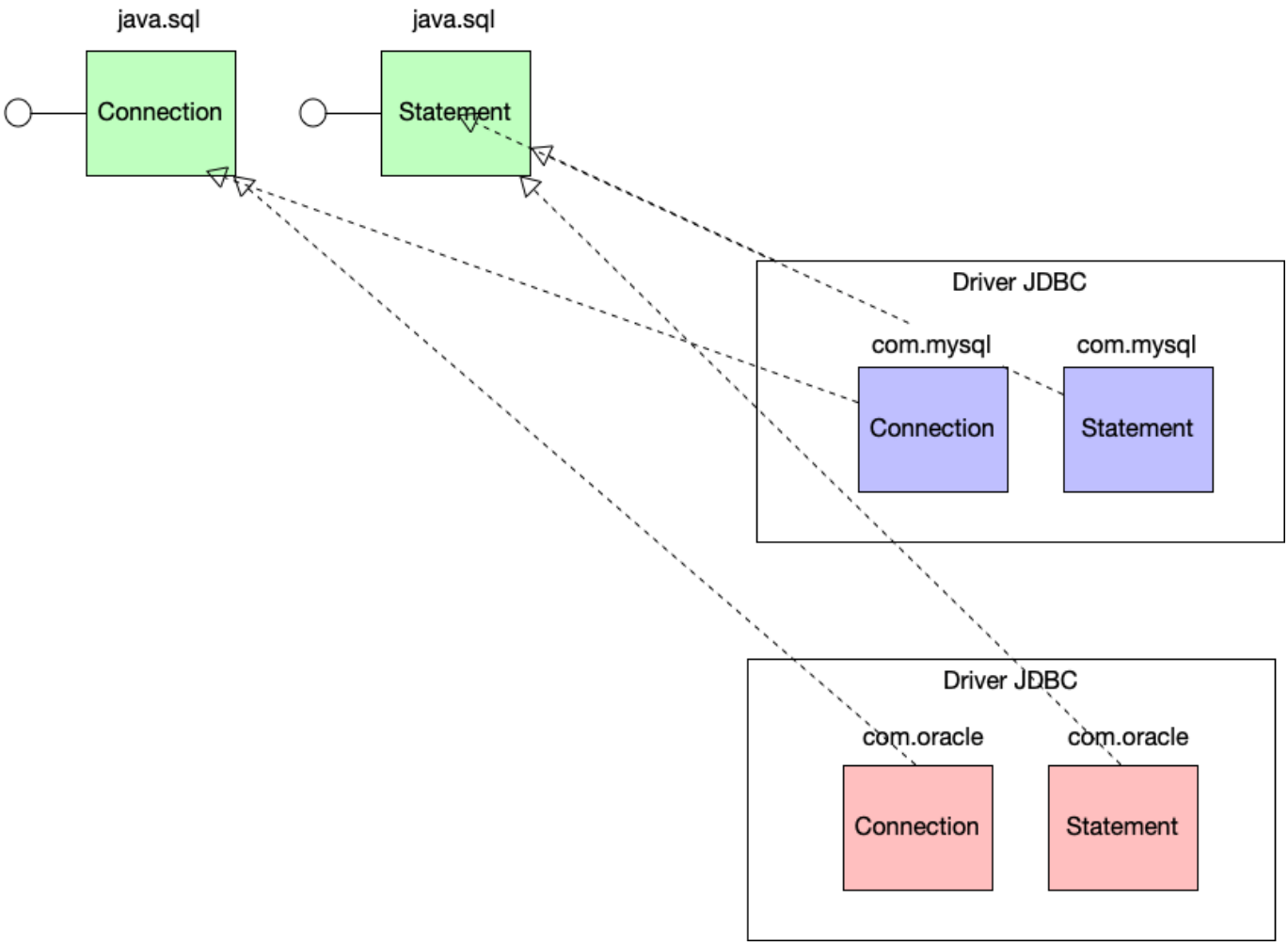
JDBC Driver y MySQL

Muchos de los principiantes suelen apostar por esta segunda opción ya que se conectan a una base de datos y al ser esta MySQL pues no dudan . Lamentablemente es la incorrecta y eso es algo que cuesta entender ¿Porque tenemos que elegir java.sql.Connection y java.sql.Statement.? Tanto java.sql.Connection como java.sql.Statement son interfaces del API de Java.



Es decir el lenguaje los incorpora como punto de extensibilidad en el sistema a la hora de conectarnos a una base datos

Cada base de datos debe aportar sus propias implementaciones y es ahí donde el Driver JDBC realiza sus aportes . El concepto de Driver hace referencia al conjunto de clases necesarias que implementa de forma nativa el protocolo de comunicación con la base de datos en un caso será Oracle y en otro caso será MySQL.



Por lo tanto para cada base de datos deberemos elegir su Driver .¿Cómo se encarga Java de saber cual tenemos que usar en cada caso?. Muy sencillo ,Java realiza esta operación en dos pasos . En el primero registra el driver con la instrucción:

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

Una vez registrado el Driver , este es seleccionado a través de la propia cadena de conexión que incluye la información sobre cual queremos usar, en la siguiente linea podemos ver que una vez especificado el tipo de conexión define el Driver “mysql”

```
String url="jdbc:mysql://localhost:3306/biblioteca";
```

Conclusiones

De esta manera Java consigue construir un código neutro que le permite conectarse a cualquier tipo de base de datos de forma transparente seleccionando el Driver JDBC.

Otros artículos relacionados

- JDBC Prepared Statement
- Ejemplo JPA
- Try with Resources
- Curso JDBC