

Tabla de Contenidos



- Paso a Paso para Consultar Datos con JDBC ResultSet
 - 1. Configuración Previa
 - 2. Conexión a la Base de Datos
- Explicación del Código
- Java ResultSet Simplificación con Try with Resources
- Java ResultSet Conclusión

JDBC ResultSet un ejemplo sencillo . Al trabajar con bases de datos en Java utilizando JDBC (Java Database Connectivity), el uso de ResultSet es fundamental para obtener y manipular los datos que provienen de una consulta SQL. En este artículo, veremos un ejemplo sencillo de cómo consultar una tabla llamada persona de la base de datos mibasedatos y obtener los campos dni, nombre y apellidos.

Paso a Paso para Consultar Datos con JDBC ResultSet

1. Configuración Previa

Antes de comenzar, necesitamos:

- Una base de datos llamada mibasedatos.
- Una tabla llamada persona con las columnas dni, nombre y apellidos.
- Un controlador JDBC compatible con nuestra base de datos (como MySQL o PostgreSQL).
Supongamos que estamos trabajando con MySQL, y ya tenemos el conector mysql-connector-java incluido en nuestro proyecto.

2. Conexión a la Base de Datos

Primero, necesitamos establecer una conexión con la base de datos. Aquí está el ejemplo de cómo hacerlo:

```
import java.sql.Connection;
```

```
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class ConsultaPersona {

    public static void main(String[] args) {
        // URL de la base de datos, nombre de usuario y contraseña
        String url = "jdbc:mysql://localhost:3306/mibasedatos";
        String usuario = "root";
        String contraseña = "password";

        // Variables para la conexión
        Connection conexion = null;
        Statement sentencia = null;
        ResultSet resultado = null;

        try {
            // Establecemos la conexión a la base de datos
            conexion = DriverManager.getConnection(url, usuario,
contraseña);

            // Creamos una sentencia SQL
            sentencia = conexion.createStatement();

            // Definimos la consulta SQL
            String consultaSQL = "SELECT dni, nombre, apellidos FROM
persona";

            // Ejecutamos la consulta y obtenemos los resultados
            resultado = sentencia.executeQuery(consultaSQL);
```

```

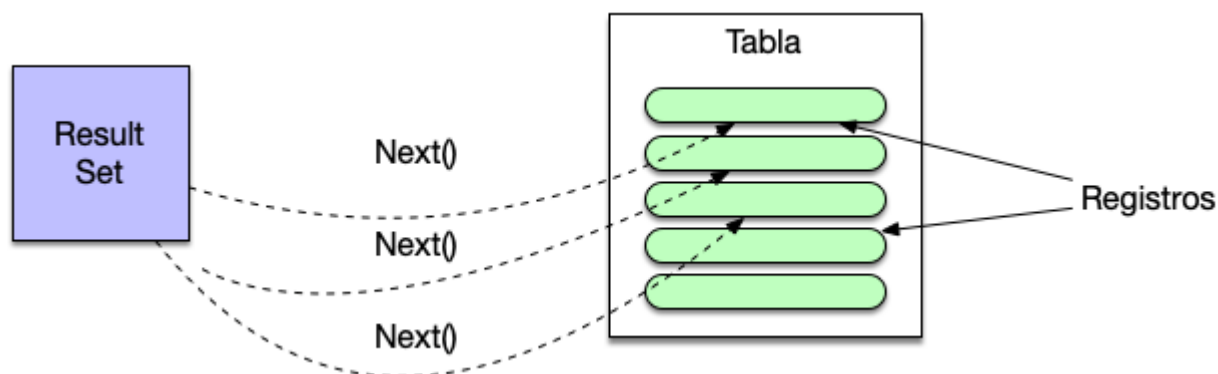
        // Recorremos el ResultSet para obtener los resultados
        while (resultado.next()) {
            String dni = resultado.getString("dni");
            String nombre = resultado.getString("nombre");
            String apellidos = resultado.getString("apellidos");

            // Imprimimos los resultados
            System.out.println("DNI: " + dni + ", Nombre: " +
nombre + ", Apellidos: " + apellidos);
        }

    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        // Cerramos los recursos en el orden adecuado
        try {
            if (resultado != null) resultado.close();
            if (sentencia != null) sentencia.close();
            if (conexion != null) conexion.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
}
}

```

Este código se conecta a la base de datos y utiliza un ResultSet para traerse la información .
Este ResultSet recorre la tabla fila a fila.



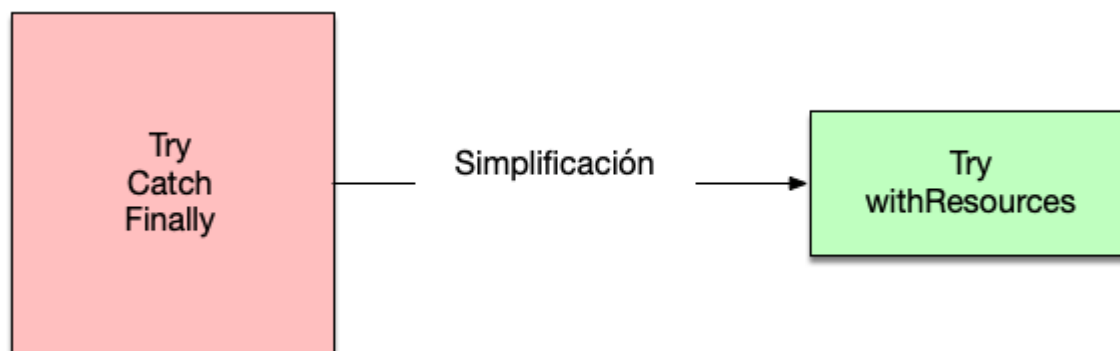
Explicación del Código

1. Conexión a la Base de Datos: Utilizamos `DriverManager.getConnection()` para conectarnos a la base de datos `mibasedatos` en MySQL. Debes reemplazar la URL, el usuario y la contraseña por los datos correctos de tu entorno.
2. Crear una Sentencia SQL: Creamos una instancia de `Statement` para ejecutar la consulta. Esta es una de las formas más sencillas de ejecutar consultas SQL, aunque en aplicaciones más avanzadas podríamos usar `PreparedStatement` para consultas parametrizadas.
3. Ejecutar la Consulta: Con `executeQuery()`, ejecutamos la consulta SQL `SELECT dni, nombre, apellidos FROM persona`, la cual devuelve un conjunto de resultados que almacenamos en un objeto `ResultSet`.
4. Recorrer los Resultados: Utilizamos un bucle `while` para recorrer cada fila del `ResultSet`. El método `next()` mueve el cursor a la siguiente fila y devuelve `false` cuando no hay más filas disponibles.
5. Obtener los Valores: Usamos los métodos `getString("nombreColumna")` para obtener los valores de las columnas `dni`, `nombre` y `apellidos` de cada fila.
6. Cerrar Recursos: Es importante cerrar los recursos (el `ResultSet`, la `Statement` y la `Connection`) en el bloque `finally`, para asegurarnos de que se liberen independientemente de si ocurrió una excepción o no.

Java ResultSet Simplificación con Try with Resources

Podemos reducir el código usando el bloque `Try with Resources` que mejora en la gestión de

recursos ya que cierra automáticamente los objetos que implementan la interfaz `AutoCloseable` (como `Connection`, `Statement` y `ResultSet`) sin necesidad de hacerlo manualmente en un bloque `finally`.



```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class ConsultaPersona {

    public static void main(String[] args) {
        // URL de la base de datos, nombre de usuario y contraseña
        String url = "jdbc:mysql://localhost:3306/mibasedatos";
        String usuario = "root";
        String contraseña = "password";

        // Usamos try-with-resources para asegurar que los recursos se
        // cierren automáticamente
        try (Connection conexion = DriverManager.getConnection(url,
            usuario, contraseña);
            Statement sentencia = conexion.createStatement();
            ResultSet resultado = sentencia.executeQuery("SELECT dni,
            nombre, apellidos FROM persona")) {
```

```

        // Recorremos el ResultSet para obtener los resultados
        while (resultado.next()) {
            String dni = resultado.getString("dni");
            String nombre = resultado.getString("nombre");
            String apellidos = resultado.getString("apellidos");

            // Imprimimos los resultados
            System.out.println("DNI: " + dni + ", Nombre: " +
nombre + ", Apellidos: " + apellidos);
        }

    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

Java ResultSet Conclusión

El objeto `ResultSet` es una herramienta clave en JDBC para trabajar con los resultados de las consultas SQL. Este ejemplo muestra cómo se puede realizar una consulta básica a la tabla `persona` y recorrer los resultados obtenidos. Para aplicaciones más avanzadas, puedes utilizar técnicas adicionales como el uso de `PreparedStatement` para consultas más seguras y eficientes.

Este es solo el comienzo de lo que puedes hacer con JDBC. A medida que tu aplicación crezca, aprender a optimizar las consultas y manejar los resultados de manera eficiente será crucial para el rendimiento de tu sistema.

- [JDBC ResultSet Types y su funcionamiento](#)
- [Creando un Java 8 Custom Stream con JDBC ResultSets](#)
- [Java Ordered vs Sorted y sus diferencias](#)

- ¿Qué es un Java DataSource?
- Java Try Catch y su manejo