ArgentinaPrograma
YoPrograma

JDBC

por Leonardo Blautzik, Federico Gasior y Lucas Videla

Julio / Diciembre 2021

Ministerio de
Desarrollo Productivo
Argentina

Bonus track: Maven

Es una herramienta para **manejar y construir** cualquier proyecto Java.

Provee una forma estándar para la realización de tareas rutinarias, ya sea la adquisición de dependencias (bibliotecas), ejecución de pruebas, construcción de artefactos, entre otras.

Nosotros lo vamos a utilizar —por el momento— para facilitar el trabajo de búsqueda de dependencias.

Ejemplo 0

¡Vamos a Eclipse!

Una API para controlarlos a todos...

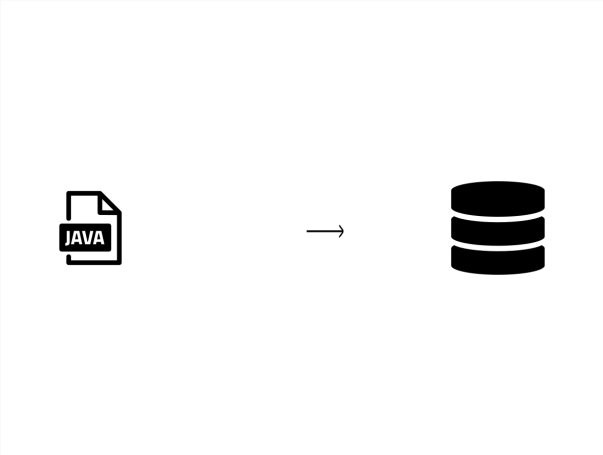

Figure 1: Necesidad

Una API para controlarlos a todos...

Figure 2: Solución

Una API para controlarlos a todos...

Figure 3: Solución específica

Una API para controlarlos a todos...

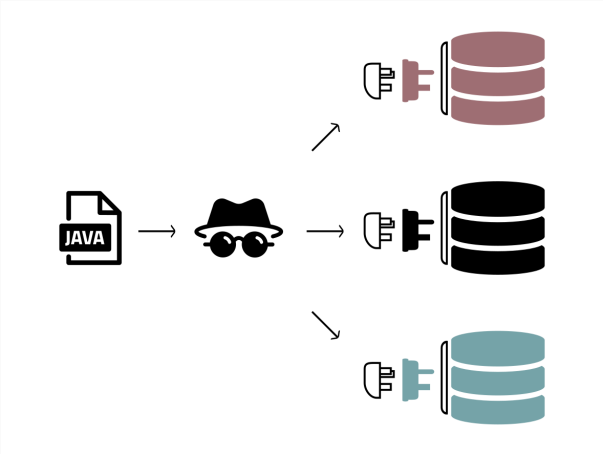

Figure 4: Ecosistema de soluciones

JDBC

Java Database Connectivity es un framework provisto por Java que permite acceder a datos de manera uniforme.

Puede conectarse a diversas fuentes, pero lo utilizaremos principalmente con una base de datos relacional.

JDBC provee una API estándar, que nos permite **abstraernos de los detalles específicos** del motor de base de datos utilizado.

Drivers

Proporcionados por proveedores específicos, complementan la interacción entre la API JDBC y el motor de base de datos.

Pasos para establecer una conexión

1. Cargar el Driver específico
2. Obtener un objeto Connection
3. Obtener un objeto Statement
4. Ejecutar la consulta o actualización
5. Leer y utilizar los resultados, si corresponde
6. Cerrar la conexión

Ejemplo en Java

```
public static void main(String[] args) throws SQLException {
  String url = "jdbc:sqlite:users.db";
  Connection connection = DriverManager.getConnection(url);

  String sql = "SELECT COUNT(1) AS TOTAL FROM USERS";
  PreparedStatement statement = connection.prepareStatement(sql);

  ResultSet resultados = statement.executeQuery();
  resultados.next();
  System.out.println(resultados.getInt("TOTAL"));

  connection.close();
}
```

Consultas

De los objetos tipo `Statement`, hay dos comandos que son los mayormente utilizados:

- `statement.executeQuery(query)`, para realizar consultas
- `statement.executeUpdate(query)`, para realizar inserciones, actualizaciones o borrados

ResultSet

El `ResultSet` es la abstracción de la tabla resultante para cualquier consulta ejecutada mediante JDBC.

Se inicializa en la posición **anterior al primer resultado**, por lo que debe desplazarse para comenzar a leer.

Permite obtener los resultados convertidos a tipos primitivos, mediante el nombre de la columna o su posición:

```
// ...
System.out.println(resultados.getInt("TOTAL"));
System.out.println(resultados.getInt(1));
```

Ejemplo 1

¡Vamos a Eclipse!

Capas

Figure 5: Arquitectura de capas

Ordenándonos: Patrón DAO

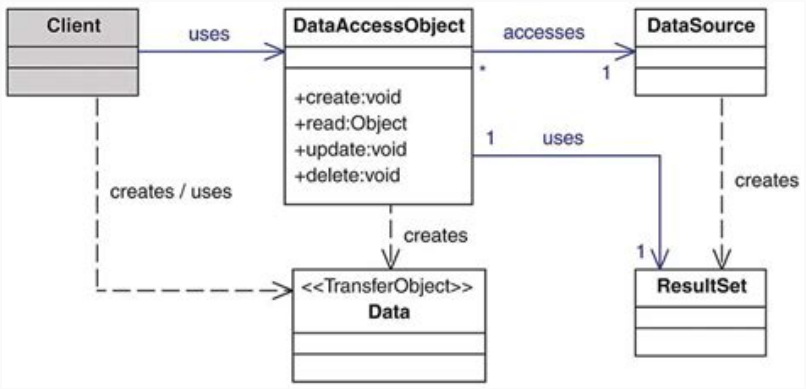

Figure 6: Data Access Object

Ejemplo 2

¡Vamos a Eclipse!

Desacoplando el DAO

- Utilizaremos una interfaz para los DAO.
  Ej: `PersonDAO`
- La implementación se realizará en una clase.
  Ej: `PersonaDAOImpl`
- Accederemos a las implementaciones utilzando un Patrón Factory.
  Ej: `DAOFactory`

De este modo, si hubiera que cambiar la implementación de la capa de persistencia sólamente necesitaremos cambiar dicha capa y no todos sus clientes.

Ejemplo 3

¡Vamos a Eclipse!

Evitando filtraciones

- Utilizaremos excepciones de negocio en lugar de las excepciones de SQL.
  Ej: `MissingDataException`

De este modo, nuestras excepciones específicas no llegarán a otras capas de la aplicación.

Puntos extra si son `RuntimeException`.

Ejemplo 4

¡Vamos a Eclipse!

Recapitulando

- Maven
- JDBC, Drivers
- Arquitectura de capas
- DAO
- Desacoplamiento
- Evitar filtraciones entre capas

Para profundizar

- Reese, G. (2000). Database programming with JDBC and Java (2nd ed). O'Reilly.
- Reese, G. (2003). Java database best practices (1st ed). O'Reilly.

¡Muchas Gracias!

continuará...

Ministerio de
Desarrollo Productivo
Argentina