|  |
| --- |
| Viewnext |
| ¿Qué es JPA, Hibernate & JDBC? |
| Breve introducción |

|  |
| --- |
| CAMILO PEREZ GUERRA  14-5-2024 |

**Introducción**

Empezaré comentando que **JPA** y **JDBC** son dos maneras que existen en el lenguaje **Java** para conectarse y trabajar con **BBDD**.

**JDBC** es más simple y secuencial a la hora de ejecutar sus pasos.

En cambio, por otro lado, **JPA** tiene un nivel más de abstracción en el cual ya tienes que comprender mejor las bases de la programación. Lo bueno de **JPA** es que es mejor a la hora de automatizar consultas.

**JDBC (Java Database Connectivity)**

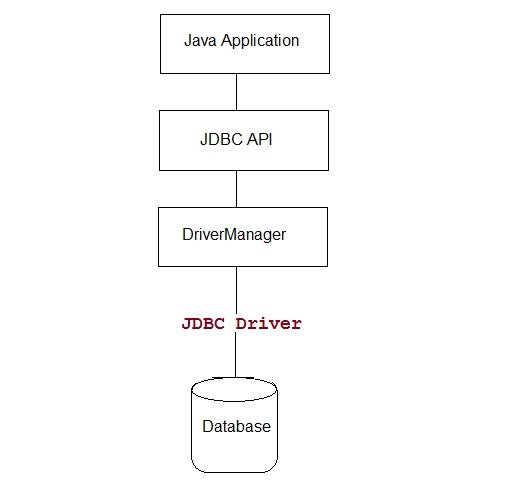
Para trabajar con **JDCB** como requisito necesitamos tener el controlador **JDBC** adecuado para el tipo de datos que utilizaremos.

Después necesitamos establecer la conexión. Normalmente se suele hacer creando una clase aparte y propia para poder instanciarla y llamarla desde otras, cuando sea necesario.

Luego necesitamos generar nuestras propias consultas, se suele hacer concatenando en tipo **String** la sentencia en **SQL** que necesitemos (*SELECT, UPDATE, INSERT etc..*).

Por último, recibir esos datos para mostrarlos y trabajar con ellos.

El tema es, que es muy secuencial y tienes que hacer todo este proceso constantemente siempre que necesites algo de la **BD**.



Esquema sencillo mostrando su funcionamiento.

**JPA(Java Persistence API)**

Es una **API** (**Application Programming Interface**) de **Java**. Esto quiere decir que son un conjunto de funciones y procedimientos que permite comunicar sistemas.

Simplifica la relación de **Java** con las **BBDD**, porque su objetivo principal es poder mapear objetos Java con las tablas de la **BD**.

Sus principales requisitos son:

Primero que nada, definir entidades como clases. (*Clase = Tabla*).

Mediante el uso de anotaciones (**tags**) puedes especificar muchas cosas,

incluso mapear atributos de la clase a columnas de la tabla.

Luego vamos a necesitar un archivo **.xml** en nuestro proyecto el cual se llama **Persistence.xml.**

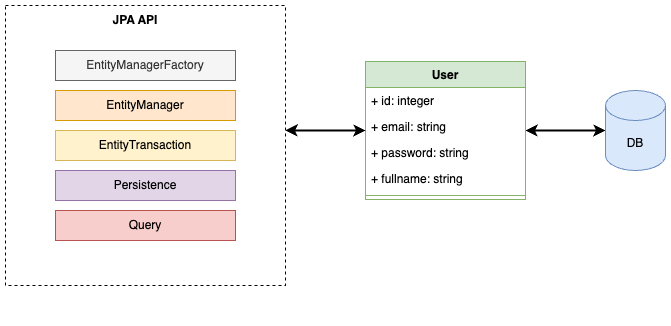
Este archivo nos va permitir configurar la conexión a la **BD** y otras opciones.

Esta es una ventaja grande respecto a **JDBC** ya que se establece la conexión automáticamente siempre que se requiera y se *auto-mapea* todo mediante el uso de las etiquetas, solo lo indicas una vez y ya queda hecho.

Por último, ya solo quedaría, crear e interactuar con el **EntityManagerFactory** y el **EntityManager** para realizar todas las consultas que quieras.

Para estas consultas se utiliza un “lenguaje” llamado **JPQL (Java Persistence Query Language)** que no es **SQL** pero está pensado para que sea muy parecido a **SQL** y simular sus consultas per en **Java**.

En resumen, está más orientado a trabajar con objetos y con otra capa de abstracción.



Esquema sencillo mostrando su funcionamiento.

**Hibernate**

**Hibernate** va de la mano con **JPA**, resumiéndolo, es un *framework* de *mapeo objeto-relacional* para **JPA**.

Simplifica la interacción entre las **BBDD** y Java, por eso es muy preferido y usado **JPA** en la actualidad.

Sus 3 mayores cualidades son:

-Optimiza consultas y transacciones.

-Mapeo objeto-relacional automático.

-Abstracción de la **BD**.

