Brenda Margarita Becerra Ruiz
312232140

JPA

Java Persistence Application, o más conocida como JPA es la API estándar que nos ofrece Java para poder crear frameworks que permitan a nuestros códigos y páginas comunicarse con una base de datos, convirtiendo objetos Java a instrucciones para el manejador de base de datos. Estos frameworks son llamados Framework Object Relational Mapping (ORM). Esto lo logra gracias a que podemos identificar modelar nuestras entidades como si fueran clases de objetos Java, las cuales serán convertidas a las instrucciones Insert, Update o Select según sea el caso. Gracias a este framework nos permite desarrollar de forma más rápida y con menos errores en tiempo de ejecución, claro si estamos acostumbrados al paradigma orientado a objetos

JPA y JDBC son herramientas para Java y completamente diferentes, aunque podemos lograr resultados muy parecidos. Las principales diferencias entre ambos son:

- JPA nos permite desarrollar mucho más rápido que JDBC.
- JPA nos permite trabajar con la base de datos por medio de entidades en vez de Querys,como JDBC.
- JPA nos ofrece un paradigma 100% orientado a objetos, en JDBC tenemos que entender a mayor nivel el funcionamiento de las consultas a través de Query y álgebra relacional.
- JPA elimina errores en tiempo de ejecución (JDBC no los elimina, de hecho, puede producir demasiados) y mejora el mantenimiento del software a diferencia de JDBC que tiene un mantenimiento más caro.
- JPA no ofrece todo el funcionamiento que ofrecería trabajar con consultas nativas, JDBC nos permite explotar al máximo las funcionalidades de la BD.
- JDBC ofrece un performance superior al que podemos lograr con JPA.

JPA funciona con anotaciones/etiquetas/decoradores, entre los más comunes tenemos:

- ➔ @Entity: indica que la clase en una entidad.
- ➔ @Table: especifica el nombre de la tabla relacionada con la entidad. Admite el parámetro name.
- ➔ @Column: indica el nombre de la columna en la BD. Admite el parámetro name.
- ➔ @Enumerated: indica que los valores de la propiedad van a estar dentro del rango de un objeto enumerador. Admite los parámetros value y EnumType
- ➔ @Id: aplicado sobre una propiedad, indica que es la clave primaria de una entidad. Admite los parámetros de tipo primitivo, string, java.util.Date, java.sql.Date, java.math.BigDecimal, java.math.BigInteger y @GeneratedValue (especifica la estrategia de generación de la llave primaria con strategy).
- ➔ @NotEmpty: indica que el valor no puede ser nulo.
- ➔ @OneToOne: especifica un valor único que contiene una relación uno a uno con otro objeto. Admite los parámetros cascade y fetch.
- ➔ @OneToMany: especifica una relación uno a muchos. Acepta los parámetros cascade y mappedBy.
- ➔ @ManyToOne: especifica una relación muchos a uno.
- ➔ @ManyToMany: especifica una relación muchos a muchos.

Bibliografía

Ramón Carrasco. (2020). *Resumen de anotaciones de Java Persistence Api (JPA).* Recuperado el 25 de Octubre de 2020 de

https://www.ramoncarrasco.es/es/content/es/kb/117/resumen-de-anotaciones-de-java-persistence-api-jpa

Oscar Blancarte. (?). *Java Persistence API (JPA).* Recuperado el 20 de Octubre de 20202 de https://www.oscarblancarteblog.com/tutoriales/java-persistence-api-jpa/