Consulta JPA

Mauricio David Molineros Molineros

Instituto Tecnológico Quito

Febrero 24 de 2021

Nota del autor

Mauricio David Molineros Landines, Análisis de Sistemas, Instituto Tecnólogico Quito

La correspondencia relacionada con esta investigación debe ser dirigida a nombre de

Mauricio David Molineros Landines, Instituto Tecnológico Quito, Av. 10 de Agosto Y Colon

Quito – Ecuador

Contacto: mauro8668@gmail.com

Contenido

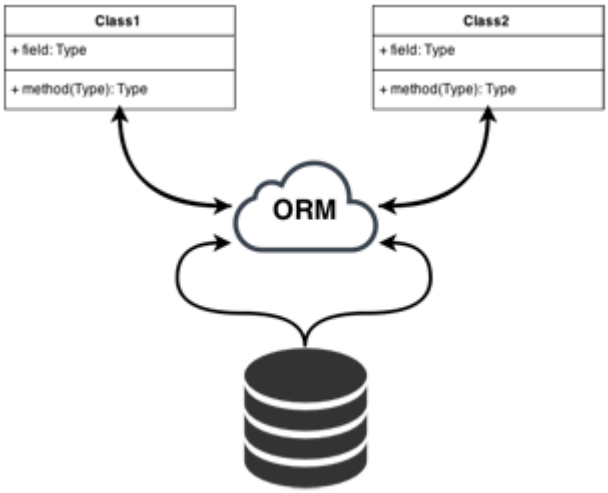
REPOSITORIO GIT

El presente trabajo se encuentra el siguiente repositorio GitHub.

Link: https://github.com/mmolineros/DEBRES.git

Que es ORM

En la actualidad se habla del Mapeo Objeto-Relacional (ORM). es un modelo de programación que consiste en la transformación de las tablas de una base de datos, en una serie de entidades que simplifiquen las tareas básicas de acceso a los datos para el programador.



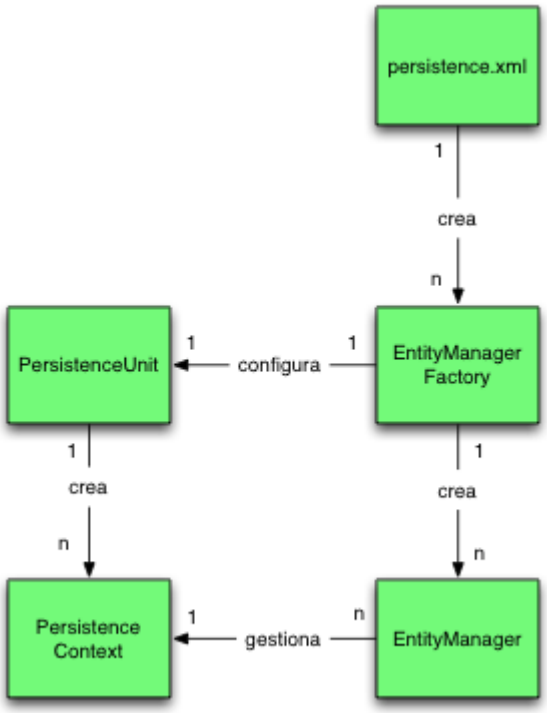
Aunque el lenguaje SQL es usado para el acceso a muchas de las bases de datos existentes, existen múltiples variaciones en las funciones que los distintos SGBD usan. Como ejemplo sería delimitar el número de registros de una consulta.

```
SELECT TOP 10 * FROM usuarios //SqlServer
SELECT * FROM usuarios LIMIT 10 //MySQL
SELECT * FROM usuarios WHERE rownum<=20; //Oracle
```

Como podemos observar de las tres bases de datos utilizadas para este ejemplo vemos diferencias en la sintaxis de su código.

Como funciona JPA

Partiendo de que JPA es el standard de Java el cual está encargada de automatizar dentro de lo posible la persistencia de nuestros objetos en la base de datos. Para explicarlo de mejor manera se tomará el siguiente ejemplo:



Que es JDBC

JDBC (Java Database Connectivity) es la especificación JavaSoft de una interfaz de programación de aplicaciones (API) estándar que permite que los programas Java accedan a sistemas de gestión de bases de datos. La API JDBC consiste en un conjunto de interfaces y clases escritas en el lenguaje de programación Java.

Que es un POJO

Según el concepto planteado por Martin Fowler, Rebecca Parsons y Josh MacKenzie en septiembre de 2000. Las clases a partir de las cuales se instancian este tipo de objetos, son clases simples e independientes del framework utilizado. Estas clases no poseen restricciones especiales (más allá de las proporcionadas por el lenguaje), y se usan para simplificar la estructuración de los desarrollos, reduciendo su complejidad, aumentando la legibilidad y facilitando la reutilización de código.

Por lo cual tenemos que los POJOs no pueden ser:

- Extender ostras clases
- Implementar interfaces
- Contener decoradores

A continuación, se mostrará un ejemplo de un POJO.

```
public class Autor {

    public Long id;
    private String nombre;
    private String apellido;

    public Autor(String nombre, String apellido) {
        this.nombre = nombre;
        this.apellido = apellido;
    }

    public Long getId() {
        return id;
    }

    public String getNombre() {
        return nombre;
    }

    public String getApellido() {
        return apellido;
    }

}
```

Que es el EntityManager

Es un gestor de recursos que mantiene la colección activa de objetos de entidad que está utilizando la aplicación. Esta maneja la interacción y metadatos de bases de datos para las correlaciones relacionales de objetos. Una instancia de un EntityManager representa un contexto de persistencia.

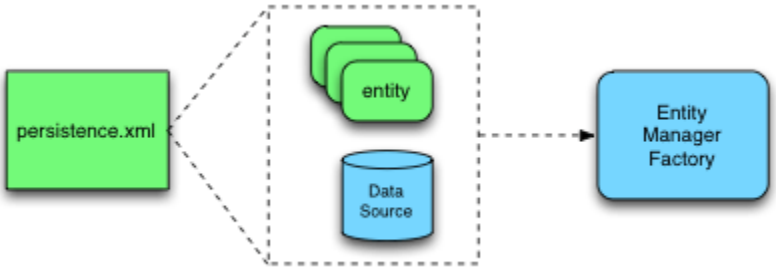
Una vez disponemos de un EntityManagerFactory este será capaz de construir un objeto de tipo EntityManager que como su nombre indica gestiona un conjunto de entidades u objetos.

Para que sirve el Fichero Persistence.xml

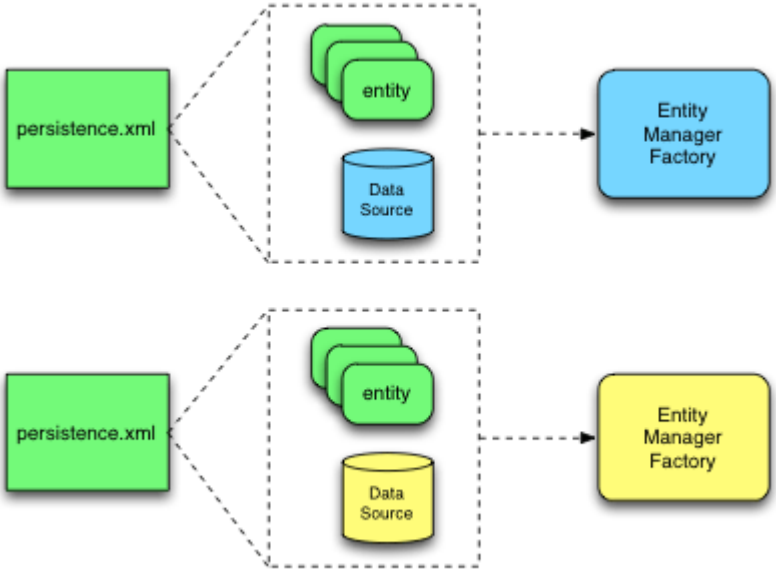
Este fichero se encarga de conectarnos a la base de datos y define el conjunto de entidades que vamos a gestionar.

```
<persistence xmlns="http://java.sun.com/xml/ns/persistence"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://java.sun.com/xml/ns/persistence http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd"
 version="2.0">
 <persistence-unit name="UnidadPersonas">
 <class>es.curso.bo.Persona</class>
 <properties>
 <property name= "hibernate.show_sql" value="true" />
 <property name="hibernate.dialect" value="org.hibernate.dialect.MySQLDialect" />
 <property name="javax.persistence.jdbc.driver" value="com.mysql.jdbc.Driver" />
 <property name="javax.persistence.jdbc.user" value="root" />
 <property name="javax.persistence.jdbc.password" value="jboss" />
 <property name="javax.persistence.jdbc.url" value="jdbc:mysql://localhost/jpa" />
</properties>
</persistence-unit>
```

Partiendo del ejemplo que se propuso para la explicación de un JPA tenemos que en nuestro caso únicamente tenemos una entidad “Persona” y luego la parte que se encarga de definir el acceso a la base de datos generando un pool de conexiones etc.



De esta forma tendremos a nuestra disposición un EntityManagerFactory con el que empezar a gestionar las entidades que se encuentran definidas a nivel del fichero persistence.xml.



Notas al pie

Bibliografía

https://programarfacil.com/blog/que-es-un-orm/

https://www.oscarblancarteblog.com/tutoriales/java-persistence-api-jpa/

http://expertojava.ua.es/experto/restringido/2015-16/jpa/jpa.html