Tabla de Contenidos

El concepto de JPA Persist es uno de los conceptos más elementales de JPA pero a veces cuesta entender a detalle como funciona a la hora de persistir datos contra la base de datos . Vamos a ver un ejemplo sencillo que se encargue de persistir la información partiendo de la entidad Libro.

```
package com.arquitecturajava.dominio;

import java.util.Date;

import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name = "libros")
public class Libro {

        @Id
        private String isbn;
        private String titulo;
        private String autor;
        private Date fecha;
        private int precio;
        public String getIsbn() {
                return isbn;
        }
```

```
	public void setIsbn(String isbn) {
		this.isbn = isbn;
	}
	public String getTitulo() {
		return titulo;
	}
	public void setTitulo(String titulo) {
		this.titulo = titulo;
	}
	public String getAutor() {
		return autor;
	}
	public void setAutor(String autor) {
		this.autor = autor;
	}
	public Date getFecha() {
		return fecha;
	}
	public void setFecha(Date fecha) {
		this.fecha = fecha;
	}
	public int getPrecio() {
		return precio;
	}
	public void setPrecio(int precio) {
		this.precio = precio;
	}
	public Libro(String isbn, String titulo, String autor, Date
fecha, int precio) {
		super();
		this.isbn = isbn;
```

```
		this.titulo = titulo;
		this.autor = autor;
		this.fecha = fecha;
		this.precio = precio;
	}
	public Libro() {
		super();
	}
	public Libro(String isbn) {
		super();
		this.isbn = isbn;
	}
	@Override
	public int hashCode() {
		final int prime = 31;
		int result = 1;
		result = prime * result + ((isbn == null) ? 0 :
isbn.hashCode());
		return result;
	}
	@Override
	public boolean equals(Object obj) {
		if (this == obj)
			return true;
		if (obj == null)
			return false;
		if (getClass() != obj.getClass())
			return false;
		Libro other = (Libro) obj;
		if (isbn == null) {
			if (other.isbn != null)
```

```
                                return false;
                        } else if (!isbn.equals(other.isbn))
                                return false;
                        return true;
                }
                @Override
                public String toString() {
                        return "Libro [isbn=" + isbn + ", titulo=" + titulo +
", autor=" + autor + ", fecha=" + fecha + ", precio="
                                        + precio + "]";
                }
}
```

Acabamos de construir la clase Libro es momento de persistirla con un EntityManager . Para ello vamos a construir un programa Main y encargarnos de usar el EntityManager y EntityManagerFactory para salvar la información en la base de datos:

```
package com.arquitecturajava.main;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

import com.arquitecturajava.dominio.Libro;

public class Principal2Insertar {

        public static void main(String[] args) {
```

```
                // unidad de persistencia
                EntityManagerFactory emf=
Persistence.createEntityManagerFactory("biblioteca");
                EntityManager em= emf.createEntityManager();
                String textoFecha= "1/01/2020";
                SimpleDateFormat ffecha= new
SimpleDateFormat("d/M/yyyy");
                Date fecha=null;
                try {
                        fecha= ffecha.parse(textoFecha);
                } catch (ParseException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
                Libro l1= new Libro ("2A","JPA","Gema",fecha,3);
                em.getTransaction().begin();
                em.persist(l1);
                em.getTransaction().commit();

        }

}
```

Todo correcto . Mucha gente piensa que cuando construimos el Libro , si modificamos alguna propiedad a posteriori se lanzarán varias consultas SQL con cada uno de los cambios que se produce, esto no es así ya que el Libro mantiene su estado en memoria y el EntityManager sabe que propiedades han cambiado y por lo tanto que hay que persistir . Imaginemonos que modificamos un poco el código y cambiamos una propiedad de valor (el titulo del libro).

```
                em.getTransaction().begin();
                em.persist(l1);
```

```
l1.setTitulo("JPA2");
em.merge(l1);
em.getTransaction().commit();
```

En este caso al cambiar el valor del titulo del Libro lactualizamos el estado del objeto y lanzamos una SQL adicional que se encargue de una vez insertado el objeto en la base de datos de actualizarlo.

```
Hibernate: insert into libros (autor, fecha, precio, titulo, isbn)
values (?, ?, ?, ?, ?)
Hibernate: update libros set autor=?, fecha=?, precio=?, titulo=?
where isbn=?
```

¿Que pasaría en el caso de que volvamos a cambiar el estado del objeto a su valor inicial? .

```
em.getTransaction().begin();
em.persist(l1);
l1.setTitulo("JPA2");
em.merge(l1);
l1.setTitulo("JPA");
em.merge(l1);
em.getTransaction().commit();
```

En este caso el framework de Persistencia que mantiene el estado del objeto en memoria es capaz de darse cuenta de que realmente no hay cambios a nivel del estado inicial del objeto y por lo tanto únicamente lanza una consulta de inserción.

```
Hibernate: insert into libros (autor, fecha, precio, titulo, isbn)
values (?, ?, ?, ?, ?)
```

¿Se puede optimizar más? . Si pero para ello vamos a hablar de flushing

JPA Persist Flushing y Persistencia (Premium)

Contenido Premium

Este contenido es solo para subsciptores (usuarios premium) . Conviertete en subscriptor por solo 2.99$ al mes . Apoya el blog y ayuda a que la plataforma crezca .

Nombre de usuario:

Contraseña:

Registro

Has perdido tu contraseña

Acceder

Otros artículos relacionados

1. Ejemplo de JPA (Java Persistence API)
2. ¿JPA vs Hibernate?
3. JPA Orphan Removal y como usarlo