CascadeType en relaciones con JPA

Cuando tenemos relaciones entre objetos, al momento de persistirlos debemos tener en consideración el tipo de dependencia que existe entre ellos. Si tenemos una entidad Persona donde uno de sus atributos es DatosContacto con información relacionada a esa persona, y eliminamos a la Persona, seguramente también vamos a querer eliminar los datos de contacto ya que no tienen ningún sentido dentro de nuestra base de datos si la Persona del cual dependen ya no existe. En este posteo, vamos a ver como definir estas cuestiones mediante el atributo cascade que se aplica a las relaciones de JPA.

Las operaciones en cascada con JPA nos permiten propagar las operaciones que se aplican en una relación entre dos entidades, definiendo el atributo cascade en las relaciones que vimos en el posteo anterior. Los valores que puede tomar cascade, son representados en el enum javax.persistence.CascadeType y son los siguientes:

CascadeType.ALL
Propaga todas las operaciones de una entidad, a la entidad con la que se relaciona. Es decir, que si insertamos, actualizamos o eliminamos una entidad, también se aplican estas operaciones a la entidad que se relaciona.

En una relación @ManyToOne entre Club y Asociacion, si aplicamos alguna de las operaciones mencionadas anteriormente sobre Club, también se aplicará sobre Asociacion.

```
@Entity
@Table(name = "clubes")
public class Club {
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private Long id;
  @ManyToOne(cascade = CascadeType.ALL)
  private Asociacion asociacion;
}
```

En este caso puntual, si eliminamos un club, también se eliminara la asociación. Por lo que al momento de usar CascadeType.ALL deberíamos tener en cuenta nuestra funcionalidad de la aplicación, ya que si hacemos esto, también se perderá la relación entre la asociación eliminada y los demás clubes que se relacionan con ella. Por tal motivo este tipo no es recomendado utilizar si no tenemos bien en claro nuestro dominio. Sin embargo, en el caso que sí amerite utilizar este tipo de cascada, tenemos como ventaja evitarnos asignar todos los tipos de cascada a la relación y esto nos da una mejor legibilidad a nuestro código.

CascadeType.PERSIST
Propaga la persistencia de una entidad a sus entidades relacionadas. Es decir, si persistimos un Club, también vamos a persistir a su Entrenador

```
@Entity
@Table(name = "clubes")
public class Club {
 @Id
 @GeneratedValue(strategy = GenerationType.IDENTITY)
 private Long id;
 @OneToOne(cascade = CascadeType.PERSIST)
 private Entrenador entrenador;
}
```

Este tipo es útil cuando ambas entidades, la principal y su relación, se crean en el mismo momento y deben ser persistidas. En este caso si al momento de crear un Club, también necesitamos crear al Entrenador, ambas entidades se van a persistir en la base de datos como nuevos registros. Podemos ver como fácilmente realizamos dos operaciones si necesidad de tener que persistir primero al Entrenador, y luego asignarlo al Club.

CascadeType.MERGE
Propaga la actualización en la base de datos de una entidad a las entidades relacionadas.

Cuando modificamos una entidad y la entidad relacionada, ambas se modifican al mismo tiempo y no se genera un nuevo registro de la entidad relacionada en la base de datos.

Si en la entidad Asociacion tenemos una relación con su Presidente, y modificamos la Asociacion, también se modificara el el Presidente en el caso de que le hayamos realización algún cambio.

```
@Entity
@Table(name = "asociaciones")
public class Asociacion {
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private Long id;
  private String nombre;
  @ManyToOne(cascade = CascadeType.MERGE)
  private Presidente presidente;
}
```

CascadeType.REMOVE
Propaga la eliminación de la entidad relacionada a la entidad principal cuando ésta es eliminada.

Este tipo de cascada nos sirve cuando una entidad solo existe a causa de la entidad que la contiene. En otras palabras, se recomienda usar solo en relaciones @OneToOne o @OneToMany, ya que las entidades relacionadas existen por la existencia de otra entidad.

Por ejemplo, en el caso del Club y el Entrenador, es una relación @OneToOne, y el entrenador va a existir si y solo si existe un Club al que pertenece.

```
@Entity
@Table(name = "clubes")
public class Club {
 @Id
 @GeneratedValue(strategy = GenerationType.IDENTITY)
 private Long id;
 private String nombre;
 @OneToOne(cascade = CascadeType.REMOVE)
 private Entrenador entrenador;
}
```
Pero en el caso de querer usarlo en una relación @ManyToOne o @ManyToMany, la entidad relacionada puede pertenecer a mas de una instancia de la entidad que la contiene, por ejemplo si aplicamos REMOVE a Asociación, estaremos eliminando una Asociación que puede estar relacionada a otro Club.

```
@ManyToOne(cascade = CascadeType.REMOVE)
 private Asociacion asociacion;
```

CascadeType.REFRESH
Cuando utilizamos el método refresh del EntityManager, los objetos que se encuentran en el entorno de persistencia del EntityManager se vuelven a cargar desde la base de datos, inclusive si alguno de estos sufrió alguna modificaciones antes de ser persistido. Este tipo no suele ser de uso común, pero nos puede ser útil para que el contexto de persistencia mantenga las versiones mas actualizadas de los objetos.

CascadeType.DETACH
Al igual que REFRESH, suele ser poco utilizado, pero es util para sacar a un objeto del contexto de persistencia de forma manual, y de esta manera podemos cancelar cualquier cambio del objeto antes de que sea persistido en la base de datos.

Como podemos ver, el uso de cascade nos evita tener que pensar en operaciones previas o posteriores de entidades relacionadas al momento de realizar una operación sobre una entidad. Sin embargo es importante tener el conocimiento de nuestro dominio y tomarnos un tiempo para pensar antes de aplicar cualquier tipo de cascade a cualquiera de las relaciones, ya que una mala implementación nos puede generar inconsistencia y perdida de integridad en nuestra base de datos.