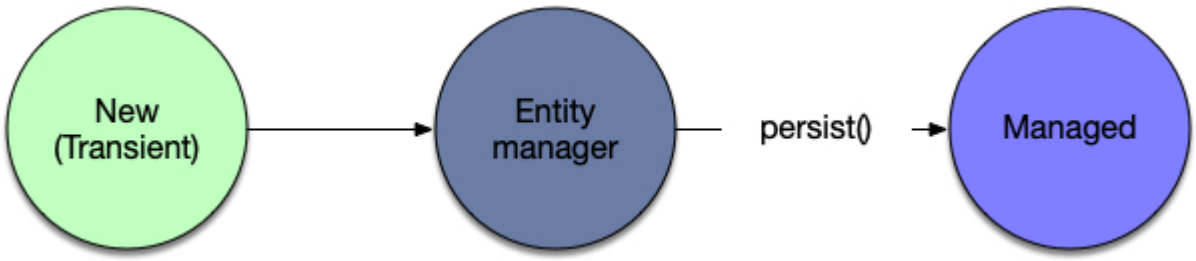
El concepto de JPA Object States es uno de esos conceptos que todos tenemos que conocer . Esto es debido a que continuamente gestionamos objetos con JPA y saber cuales son los estados por los que pueden pasar es muy importante para clarificar dudas y trabajar de una forma mejor. Vamos a echarle un vistazo a los diferentes estados

JPA Object States (New/Transient)

En este estado el objeto que hemos creado no ha sido todavía persistido en la base de datos y es un objeto que esta simplemente en un estado de New o Transient , la base de datos no sabe nada de él. Nosotros podemos usar un EntityManager y ejecutar la operación de persist . Esta operación nos permitirá guardar el objeto en la base de datos y que pase a estar gestionado.



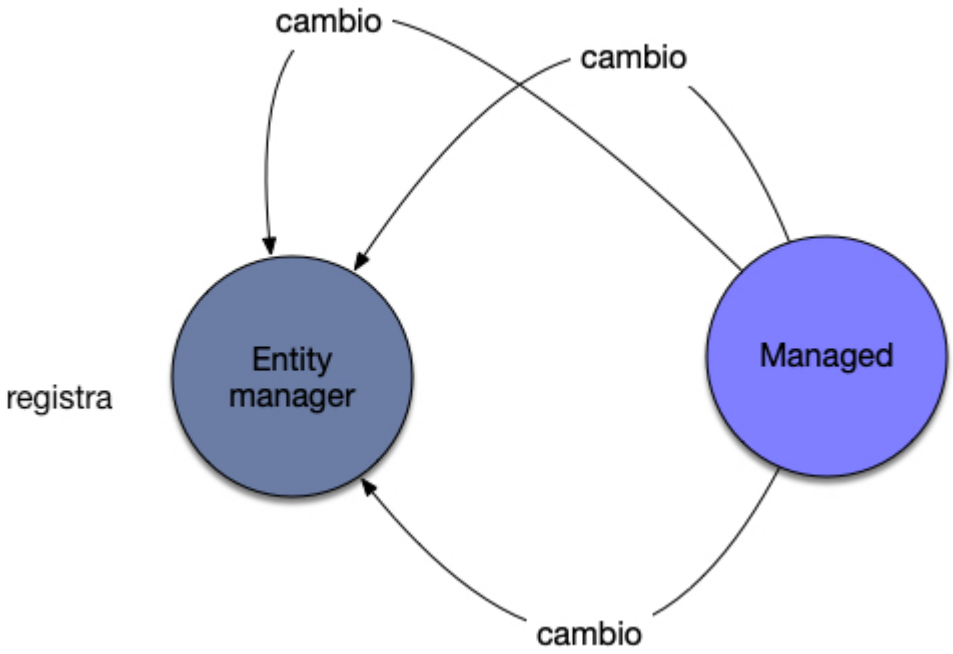
Veamos un ejemplo de código:

```
package main;

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

import model.Cliente;
```

```
public class Principal {

	public static void main(String[] args) {

		Cliente c = new Cliente("5", "pedro", "gomez", 30,
606778899);
		EntityManagerFactory emf =
Persistence.createEntityManagerFactory("UnidadPersonas");
		EntityManager em = emf.createEntityManager();
		try {
			em.getTransaction().begin();
			em.persist(c);
			em.getTransaction().commit();
		} catch (Exception e) {

			e.printStackTrace();
		} finally {
			em.close();

		}
	}

}
```

JPA Object States (Managed)

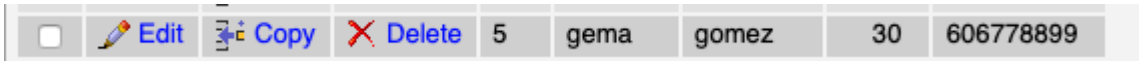
Este probablemente es el estado más importante a nivel de JPA y es cuando nuestro objeto esta controlado por el EntityManager y cualquier cambio que hagamos sobre él queda registrado.



Es decir si nosotros después de persistir el objeto hacemos un cambio en él mientras seguimos bajo la protección del EntityManager estaremos registrando esos cambios y JPA se dará cuenta de ello cuando realicemos la operación de commit. Es decir si cambiamos el bloque try/catch del ejemplo anterior por el siguiente:

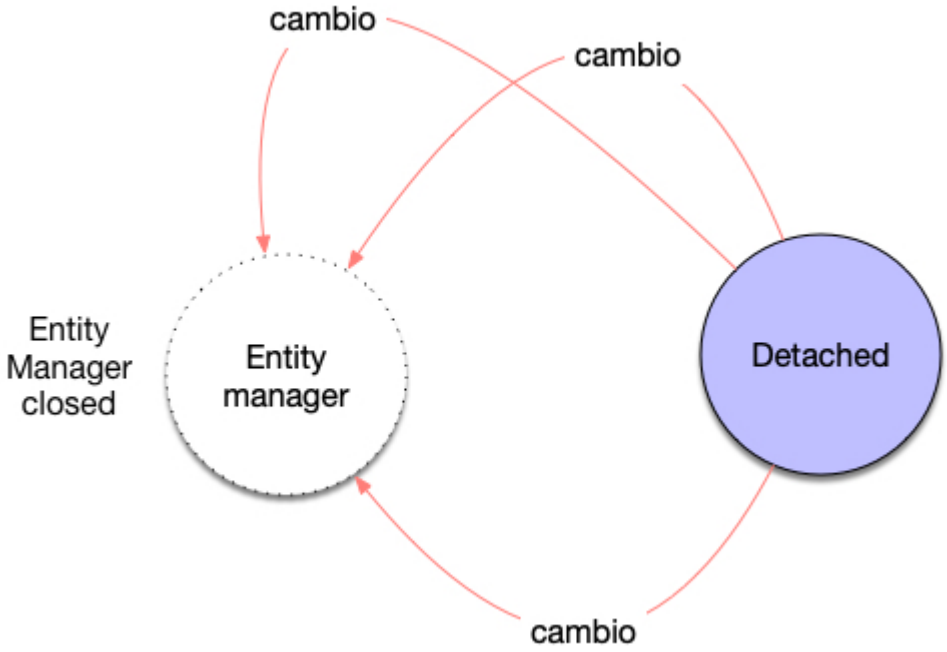
```
try {
        em.getTransaction().begin();
        em.persist(c);
        c.setNombre("gema");
        em.getTransaction().commit();
} catch (Exception e) {

        e.printStackTrace();
}
```

Al estar el objeto c ya gestionado por el EntityManager cualquier cambio que realicemos sobre el será registrado en este caso si cambiamos el nombre por gema cuando lo salvemos en la base de datos. Ese será el nombre que finalmente se almacenará esto es debido a que hemos usado el EntityManager y el método persist para adquirir el control sobre el objeto y sus cambios.



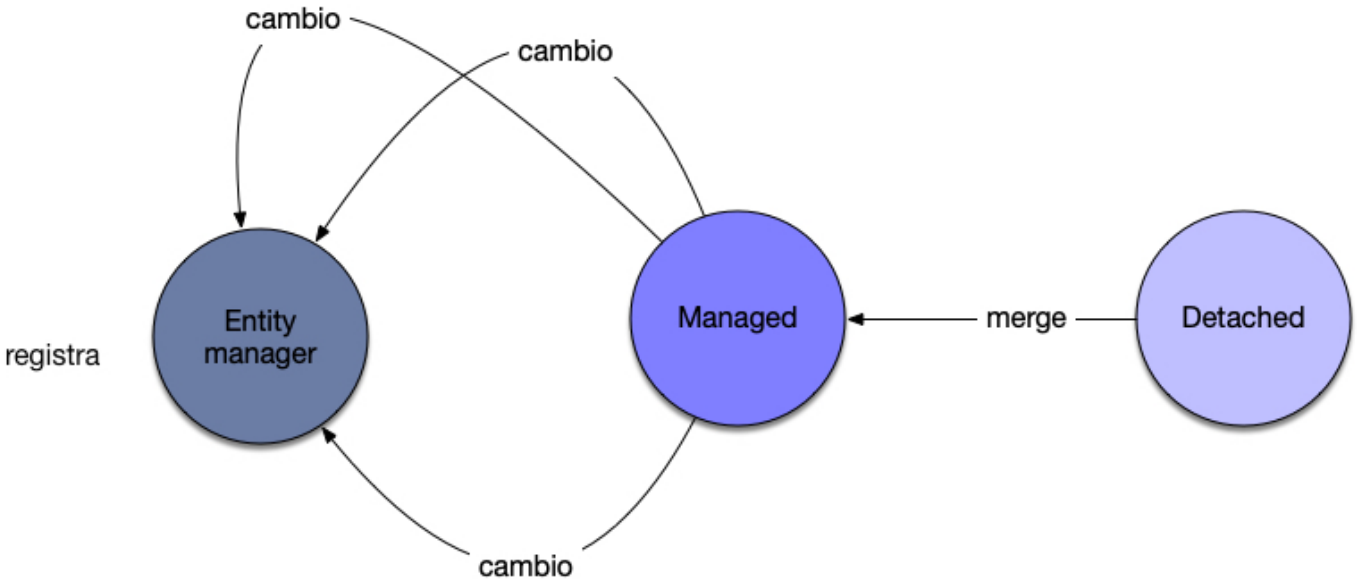
JPA Object States (Detached)

Este estado se produce cuando nosotros en algún momento cerramos el EntityManager. En cuanto esta operación se produzca los cambios que realizamos en el objeto no son ya gestionados por el Entity Manager y digamos que no quedan registrados de ninguna forma .



Por lo tanto cuando queramos salvar o guardar los cambios de este objeto en la base de datos deberemos primero ponerlo bajo el control del EntityManager. Para ello

necesitaremos una operación de merge



Vamos a verlo en ejecución:

```
package main;

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

import model.Cliente;

public class Principal {

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

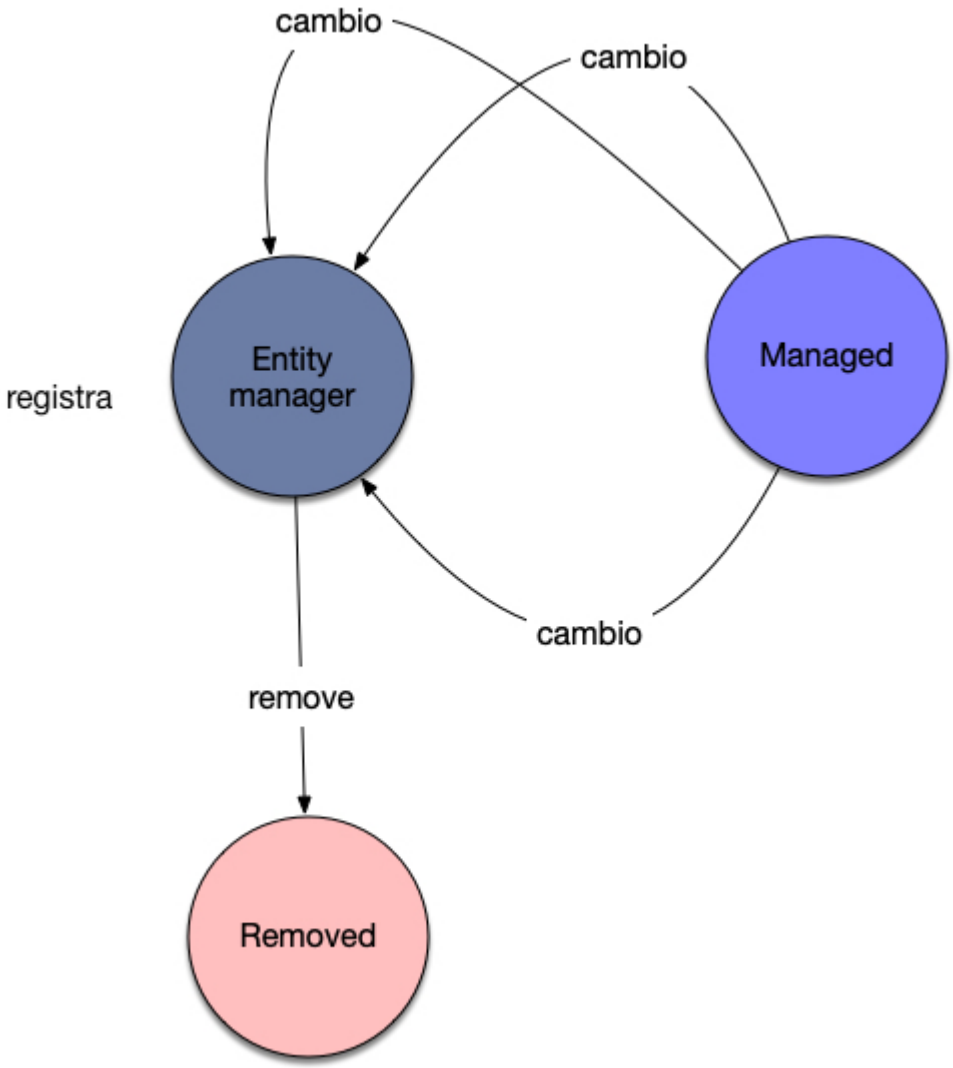
```
		Cliente c = new Cliente("5", "pedro", "gomez", 30,
606778899);
		EntityManagerFactory emf =
Persistence.createEntityManagerFactory("jpa001");
		EntityManager em = emf.createEntityManager();
		try {
			em.getTransaction().begin();
			em.persist(c);
			c.setNombre("gema");
			em.getTransaction().commit();
		} catch (Exception e) {

			e.printStackTrace();
		} finally {
			em.close();

		}
		EntityManager em2 = emf.createEntityManager();
		try {
			em2.getTransaction().begin();
			em2.merge(c);
			c.setNombre("angel");
			em.getTransaction().commit();
		} catch (Exception e) {

			e.printStackTrace();
		} finally {
			em.close();

		}
	}
```

```
}
```

En este caso lo que hemos hecho es persistir un objeto en la base de datos y cerrar el EntityManager . Realizada esta operación el objeto queda en una situación detached y por lo tanto las modificaciones que hagamos sobre él no son salvadas en la base de datos hasta que volvamos a realizar una operación de merge. Esta operación deja a nuestro objeto otra vez bajo el control de un EntityManager y los cambios que realizamos serán trackeados.

JPA Object States (Removed)

Cuando un objeto esta gestionado por el EntityManager tenemos la opción de eliminarlo y borrar el registro de la base de datos de tal manera que cuando el EntityManager realice una operación de Flush o se cierre el registro se elimina de la tabla.



Estos son los diferentes estados por los que una entidad de JPA puede pasar conocernos siempre nos resultará útil en la programación del día a día.

1. EntityManager, EntityManagerFactory y Singletons
2. Eclipse JPA y clases de dominio
3. Utilizando un JPA Stream con JPA 2.2
4. Un ejemplo de JPA Entity Graph
5. EntityManager