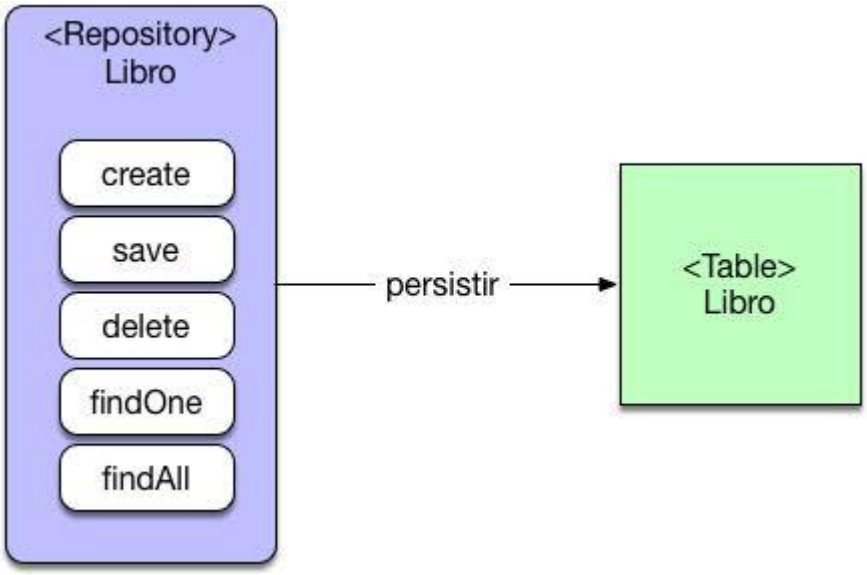
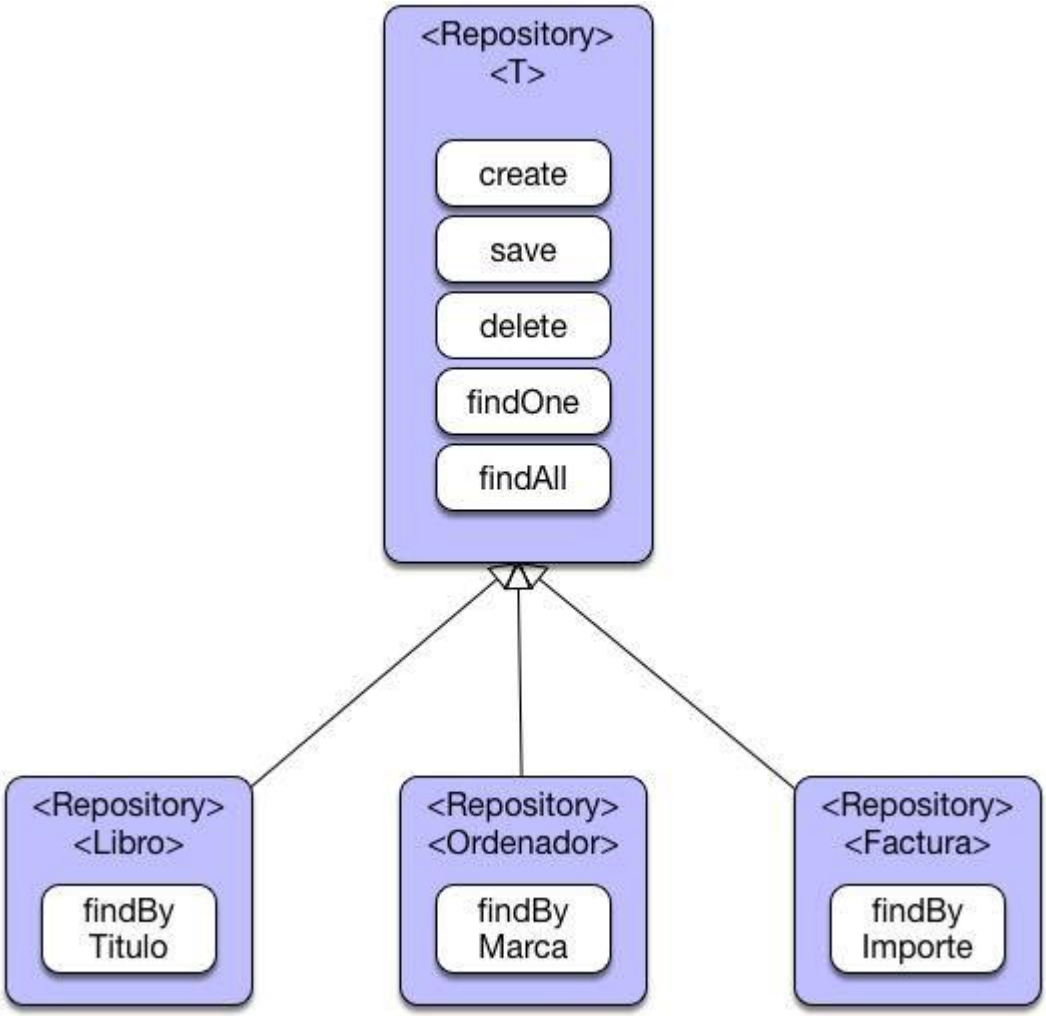
El concepto de Java Generic Repository es muy habitual cuanto trabajamos con tecnologías de persistencia. El concepto de Repository como clase que se encarga de gestionar todas las operaciones de persistencia contra una tabla de la base de datos es hace ya mucho tiempo un clásico.



La mayor parte de los lenguajes de programación soportan el uso de clases Genéricas . Estas permiten avanzar en el uso del patrón Repository y generar repositorios genéricos que reducen de forma significativa el código que tenemos que construir. Todos nuestros repositorios extenderán del repositorio genérico y añadirán más operaciones.



Un Ejemplo de Java Generic Repository

Vamos a construir un ejemplo de Java Generic Repository utilizando JPA . Para ello construiremos un proyecto de Maven con Eclipse y añadiremos las dependencias necesarias.

```
<dependency>
    <groupId>org.hibernate.javax.persistence</groupId>
    <artifactId>hibernate-jpa-2.1-api</artifactId>
    <version>1.0.0.Final</version>
</dependency>
<dependency>
```

```
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>5.2.12.Final</version>
</dependency>
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>6.0.6</version>
</dependency>
```

Una vez añadidas las dependencias vamos a construir el fichero persistence.xml que nos configura los parametros de acceso a la base de datos.

```
<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="mislibros">

        <properties>
            <property name="hibernate.dialect"
value="org.hibernate.dialect.MySQL5Dialect"/>
            <property name="hibernate.hbm2ddl.auto" value="create-
drop"/>

            <property name="javax.persistence.jdbc.driver"
value="com.mysql.jdbc.Driver"/>
            <property name="javax.persistence.jdbc.url"
value="jdbc:mysql://localhost:8889/mibasedatos"/>
            <property name="javax.persistence.jdbc.user"
```

```
value="root"/>
            <property name="javax.persistence.jdbc.password"
value="root "/>
        </properties>
    </persistence-unit>
</persistence>
```

El siguiente paso es construir nuestra Entidad

```
package bo;

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

@Entity
public class Libro {
        @Id
        private String titulo;
        private String autor;
        private int paginas;
        private String categorias;

        public Libro() {
                super();
        }
        public Libro(String titulo, String autor, int paginas, String
categorias) {
                super();
                this.titulo = titulo;
```

```
                this.autor = autor;
                this.paginas = paginas;
                this.categorias = categorias;
        }
        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 int getPaginas() {
                return paginas;
        }
        public void setPaginas(int paginas) {
                this.paginas = paginas;
        }
        public String getCategorias() {
                return categorias;
        }
        public void setCategorias(String categorias) {
                this.categorias = categorias;
        }
}
```

Es momento de construir el Java Generic Repository del cual el resto de nuestras clases se

va a apoyar junto con su interface.

```
package com.arquitecturajava.repository;

public interface GenericRepository<T>{

    T create(T t);

    void delete(T t);

    T find(T t);

    T update(T t);

    Iterable<T> findAll();
}
```

```
package com.arquitecturajava.repository;

import java.lang.reflect.ParameterizedType;
import java.lang.reflect.Type;

import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import javax.persistence.TypedQuery;
import javax.persistence.criteria.CriteriaBuilder;
import javax.persistence.criteria.CriteriaQuery;
import javax.persistence.criteria.Root;

public class GenericRepositoryJPA<T> implements GenericRepository<T>{

        protected EntityManager entityManager;
```

```
private Class<T> type;

public EntityManager getEntityManager() {
        return entityManager;
}

@PersistenceContext
public void setEntityManager(EntityManager entityManager) {
        this.entityManager = entityManager;
}

public GenericRepositoryJPA() {
        Type t = getClass().getGenericSuperclass();
        ParameterizedType pt = (ParameterizedType) t;
        type = (Class) pt.getActualTypeArguments()[0];
}

public T create(final T t) {

        entityManager.persist(t);
        return t;
}

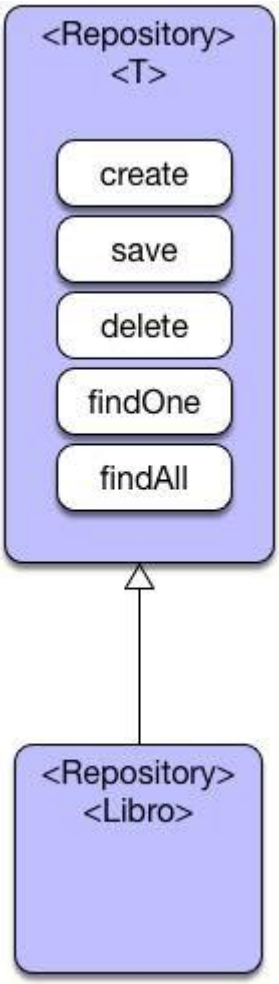
public void delete(final Object objeto) {
         entityManager.remove(entityManager.merge(objeto));
}

public T find(final Object id) {
        return (T) entityManager.find(type, id);
}
```

```
        public T update(final T t) {
                return entityManager.merge(t);
        }

        public Iterable<T> findAll() {
                 CriteriaBuilder cb =
this.entityManager.getCriteriaBuilder();
                 CriteriaQuery<T> criteriaQuery =
cb.createQuery(type);
                 Root<T> root = criteriaQuery.from(type);
                 criteriaQuery.select(root);
                 TypedQuery<T> query =
entityManager.createQuery(criteriaQuery);
                return query.getResultList();
        }
}
```

Creada la clase Genérica nos queda extender de ella y construir nuestro repositorio para la clase Libro. En este caso es muy sencillo ya que no vamos a añadir métodos adicionales




```
package com.arquitecturajava.repository;

import bo.Libro;

public class LibroRepository extends
GenericRepositoryJPA&amp;amp;amp;amp;amp;lt;Libro&amp;amp;amp;amp;amp;
gt; {

}
```

Es momento de usar nuestro Repositorio de Libro e insertar datos en la base de datos usando sus capacidades de herencia

```
package com.arquitecturajava;

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

import com.arquitecturajava.repository.LibroRepository;

import bo.Libro;

public class Principal {

	public static void main(String[] args) {
		// TODO Auto-generated method stub

		EntityManagerFactory
emf=Persistence.createEntityManagerFactory("mislibros");
		EntityManager em=emf.createEntityManager();

		LibroRepository lr= new LibroRepository();
		lr.setEntityManager(em);
		lr.create(new Libro ("el juego de ender","Orson Scott
Card",500,"Cienciaficcion"));

	}

}
```

Acabamos de insertar en la base de datos . Hemos construido un Java Generic Repository.

1. JPA Lazy fetching proxies y rendimiento
2. JPA DTO (Data Transfer Object) y JPQL
3. JPA Criteria API , un enfoque diferente
4. Un ejemplo de JPA Entity Graph
5. JPA Wiki