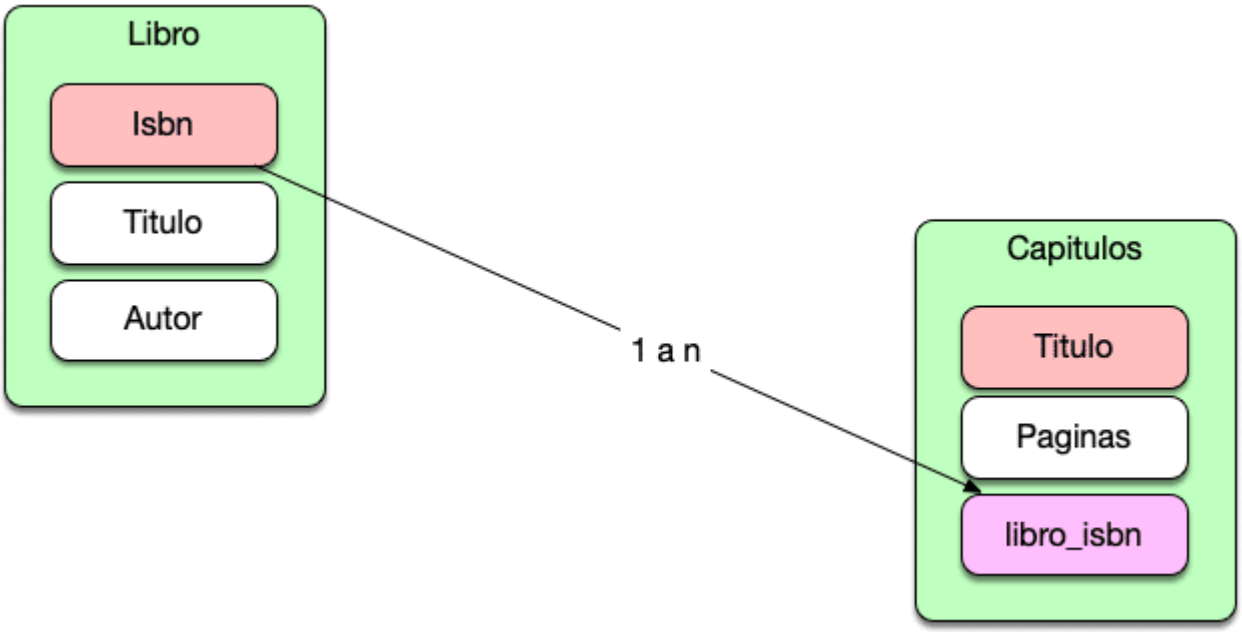
Tabla de Contenidos

- JPA Orphan Removal
  - ¿Cómo podemos solventar este problema?
- Otros artículos relacionados

JPA Orphan Removal ,es una de las características con las que todos los desarrolladores cuando empiezan tienen problemas. Cuando nosotros trabajamos con bases de datos las relaciones más comunes son las de 1 a n . Por ejemplo un Libro dispone de varios Capítulos esto es una relación de 1 a n . Vamos a ver el diagrama de tablas simplificado con los campos fundamentales



JPA Orphan Removal

Si nosotros queremos crear esta estructura en JPA deberemos crear algo de este estilo :

```
package com.arquitecturajava.data1;
```

```
import java.util.ArrayList;
import java.util.Date;
import java.util.List;

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

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

  @Id
  private String isbn;
  private String titulo;
  private String autor;
  private double precio;
  private Date fecha;
  @OneToMany(mappedBy = "libro")
  private List<Capitulo> capitulos= new ArrayList<Capitulo>();
  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 double getPrecio() {
  return precio;
}
public void setPrecio(double precio) {
  this.precio = precio;
}
public Date getFecha() {
  return fecha;
}
public void setFecha(Date fecha) {
  this.fecha = fecha;
}
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;
  }
  public List<Capitulo> getCapitulos() {
    return capitulos;
  }
  public void setCapitulos(List<Capitulo> capitulos) {
    this.capitulos = capitulos;
  }
  public Libro(String isbn, String titulo, String autor, double
precio, Date fecha) {
    super();
    this.isbn = isbn;
```

```
    this.titulo = titulo;
    this.autor = autor;
    this.precio = precio;
    this.fecha = fecha;
  }
}
```

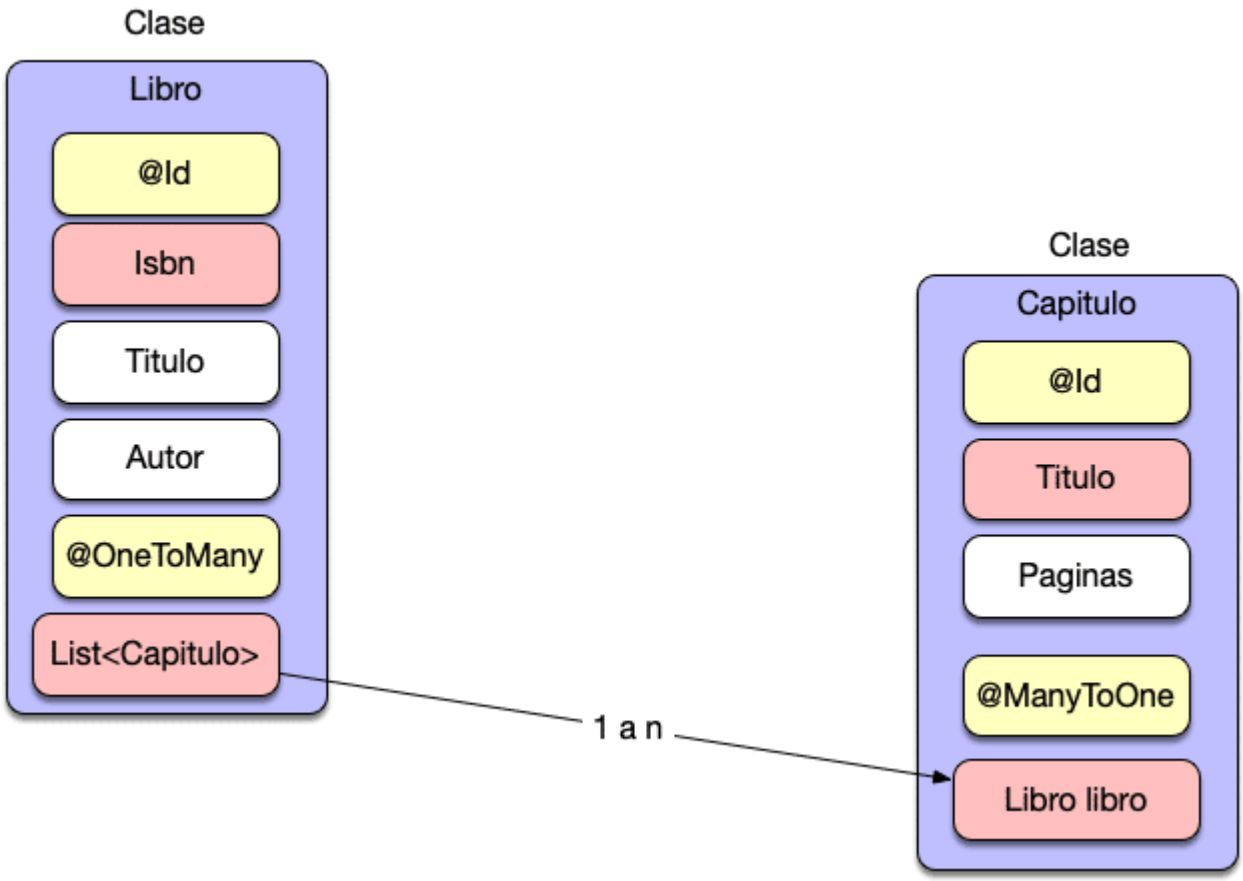
```
package com.arquitecturajava.data1;

import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;
@Entity
@Table(name="capitulos")
public class Capitulo {
  @Id
  private String titulo;
  private int paginas;
  @ManyToOne
  @JoinColumn(name="libros_isbn")
  private Libro libro;
  public Libro getLibro() {
    return libro;
  }
  public void setLibro(Libro libro) {
    this.libro = libro;
  }
  public String getTitulo() {
    return titulo;
  }
```

```
  public void setTitulo(String titulo) {
    this.titulo = titulo;
  }
  public int getPaginas() {
    return paginas;
  }
  public void setPaginas(int paginas) {
    this.paginas = paginas;
  }
  public Capitulo(String titulo, int paginas) {
    super();
    this.titulo = titulo;
    this.paginas = paginas;
  }
  public Capitulo() {
    super();
  }
  @Override
  public int hashCode() {
    final int prime = 31;
    int result = 1;
    result = prime * result + ((titulo == null) ? 0 :
titulo.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;
    Capitulo other = (Capitulo) obj;
    if (titulo == null) {
      if (other.titulo != null)
        return false;
    } else if (!titulo.equals(other.titulo))
      return false;
    return true;
  }
}
```

En esta relación disponemos de dos clases Libro y Capitulo , ambas están relacionas vía JPA con dos relaciones @OnetoMany y otra @ManyToOne.



Si nosotros utilizamos JPA para eliminar un Libro tendremos un problema importante y es que no podemos eliminar un Libro sino tenemos eliminados sus Capítulos ya que existe una foreign key entre las dos tablas. Veamoslo en un test sencillo :

```
package com.arquitecturajava.data1;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.PersistenceUnit;

import org.junit.jupiter.api.Test;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.test.context.jdbc.Sql;
```

```
@SpringBootTest
@Sql({ "/schema.sql", "/data.sql" })
class Data2ApplicationTests {

  @PersistenceUnit
  EntityManagerFactory emf;
  @Test
  void borrar() {
    EntityManager em= emf.createEntityManager();
    em.getTransaction().begin();
    Libro libro=em.find(Libro.class, "1");
    em.remove(libro);
    em.flush();
    em.getTransaction().commit();
  }


}
```

Este código al ejecutarse falla ya que no podemos eliminar un libro y dejar en la base de datos sus capítulos:



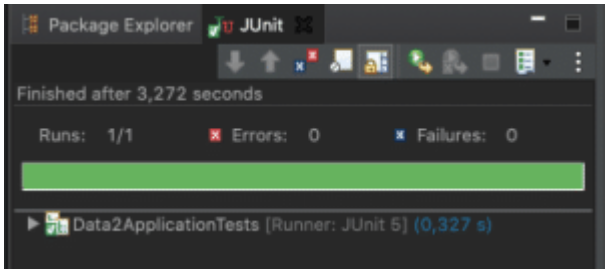
¿Cómo podemos solventar este problema?

La solución es relativamente sencilla , necesitamos usar JPA Orphan removal y modificar nuestra clase Libro de tal forma que cada vez que eliminemos un Libro automáticamente se eliminen todos los capítulos que están asociados a él:

```
@OneToMany(mappedBy = "libro",orphanRemoval = true)
```

```
private List<Capitulo> capitulos= new ArrayList<Capitulo>();
```

De esta forma JPA se encargará de eliminar esos hijos de forma transparente para nosotros sin que tengamos que preocuparnos . El test entonces se ejecutará sin problema



Otros artículos relacionados

1. JPA Transaction, un concepto importante
2. JPA @OneToOne y relaciones 1 a 1
3. JPA Merge y el manejo del EntityManager
4. JPA @Basic , optimizando los fechings
5. JPA Wikipedia