

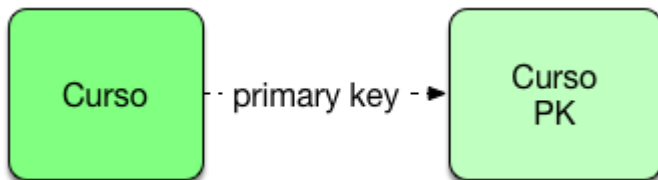
El uso de JPA composite key es muy común cuando trabajamos con JPA , pronto aparecerán tablas en el modelo de datos que necesitan crear este tipo de claves. Vamos a construir un ejemplo de como crear una clave compuesta utilizando JPA. Para ello vamos a partir del concepto de Curso que contiene los siguientes campos:

1. Título
2. Nivel
3. Categoría
4. Horas

En un primer lugar podemos pensar que con el título del Curso es suficiente . Sin embargo pueden existir en el modelo cursos con el mismo nombre pero diferente nivel. Así pues la clave primaria del curso estará compuesta por título y nivel.

JPA Composite key y Cursos

Vamos a declarar dos clases a nivel de Java (Curso y CursoPK) .



La clase CursoPK se va a encargar de almacenar la clave primaria compuesta (título y el nivel).

```
package com.arquitecturajava;
```

```
import java.io.Serializable;
```

```

import javax.persistence.Embeddable;

@Embeddable
public class CursoPK implements Serializable {

    private String titulo;
    private int nivel;
    public String getTitulo() {
        return titulo;
    }
    public void setTitulo(String titulo) {
        this.titulo = titulo;
    }
    public int getNivel() {
        return nivel;
    }
    public void setNivel(int nivel) {
        this.nivel = nivel;
    }
    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime * result + nivel;
        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;
    CursoPK other = (CursoPK) obj;
    if (nivel != other.nivel)
        return false;
    if (titulo == null) {
        if (other.titulo != null)
            return false;
    } else if (!titulo.equals(other.titulo))
        return false;
    return true;
}
public CursoPK(String titulo, int nivel) {
    super();
    this.titulo = titulo;
    this.nivel = nivel;
}
}

```

Esta clase lleva la anotación `@Embeddable` que nos permite usarla dentro de otra. Esa otra clase va a ser la clase `Curso` que será el objeto de negocio con el que trabajamos.

```
package com.arquitecturajava;
```

```
import java.io.Serializable;
```

```
import javax.persistence.EmbeddedId;
import javax.persistence.Entity;

@Entity
public class Curso implements Serializable {
    private String categoria;
    private int horas;
    @EmbeddedId
    private CursoPK cursoPK;
    public Curso(String titulo,String categoria, int horas , int
nivel) {
        super();
        this.categoria = categoria;
        this.horas = horas;
        cursoPK= new CursoPK(titulo,nivel);
    }
    public String getTitulo() {
        return cursoPK.getTitulo();
    }
    public void setTitulo(String titulo) {
        cursoPK.setTitulo(titulo);
    }
    public int getNivel() {
        return cursoPK.getNivel();
    }
    public void setNivel(int nivel) {
        cursoPK.setNivel(nivel);
    }

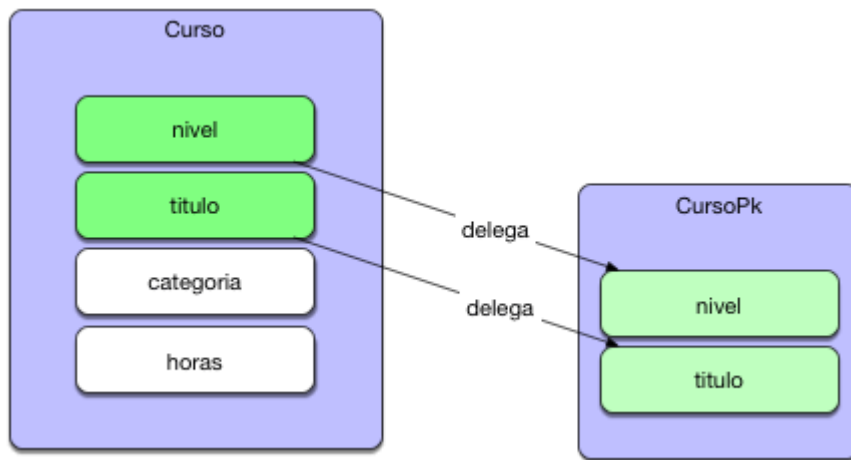
    public String getCategoria() {
        return categoria;
    }
}
```

```

    }
    public void setCategoria(String categoria) {
        this.categoria = categoria;
    }
    public int getHoras() {
        return horas;
    }
    public void setHoras(int horas) {
        this.horas = horas;
    }
    public int hashCode() {
        return cursoPK.hashCode();
    }
    public boolean equals(Object obj) {
        return cursoPK.equals(obj);
    }
}

```

La anotación de `@EmbeddedId` es la encargada de definir que la clase embebida es la que se usa como primary key de la tabla . En este caso al tener `CursoPK` varios campos la clave es compuesta . Nosotros además hemos usado el concepto de delegación a la hora de relacionar la clase `Curso` y `CursoPK` diseñando campos específicos que delegan de una clase a la otra. Acabamos de terminar de configurar un JPA composite key.



Nos queda definir el fichero de persistence.xml para poder salvar nuestros 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="arquitecturajava">
```

```
<properties>
```

```
<property name = "hibernate.show_sql" value = "true" />
```

```
<property name="hibernate.dialect"
```

```
value="org.hibernate.dialect.MySQLDialect" />
```

```
<property name="javax.persistence.jdbc.driver"
```

```
value="com.mysql.jdbc.Driver" />
```

```
<property name="javax.persistence.jdbc.user" value="root" />
```

```
<property name="javax.persistence.jdbc.password" value="miclave" />
```

```
<property name="javax.persistence.jdbc.url"
value="jdbc:mysql://localhost:8889/java" />
</properties>
</persistence-unit>

</persistence>
```

Por último creamos el programa principal e insertamos :

```
package com.arquitecturajava;

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

public class Principal {

    public static void main(String[] args) {
        EntityManagerFactory emf =
Persistence.createEntityManagerFactory("arquitecturajava");
        EntityManager em = emf.createEntityManager();
        EntityTransaction transaccion= em.getTransaction();
        transaccion.begin();

        Curso c= new Curso("Java","Programacion",20,1);
        em.persist(c);
        transaccion.commit();
    }
}
```

```
em.close();
```

```
}
```

```
}
```

El programa nos insertará el Curso en la base de datos. Podemos modificar el objeto y seleccionar un curso de Java con diferente nivel:

```
Curso c= new Curso("Java","Programacion",20,2);
```

El código se ejecutará sin ningún tipo de problema ya tenemos dos cursos en la base de datos:

#	titulo	categoria	nivel	horas
1	Java	Programacion	1	20
2	Java	Programacion	2	20

Ahora bien si volvemos a intentar insertar el curso de nivel 1 nos saltará una excepción de clave duplicada.

```
com.mysql.jdbc.exceptions.jdbc4.MySQLIntegrityConstraintViolationException: Duplicate entry 'Java-1'  
sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)  
sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:62)  
sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.java:45)  
java.lang.reflect.Constructor.newInstance(Constructor.java:423)  
com.mysql.jdbc.Util.handleNewInstance(Util.java:406)
```

Acabamos de crear un ejemplo con JPA composite key utilizando como clave compuesta el título y el nivel.

Otros artículos relacionados:

1. [Un ejemplo de JPA Entity Graph](#)
2. [Ejemplo de JPA , Introducción \(I\)](#)
3. [JPA Single Table Inheritance](#)