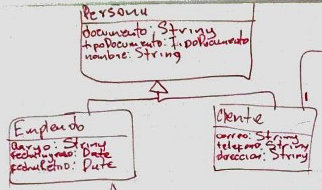
# Herencia

En JPA existen 4 alternativas para representar o resolver la herencia en el modelo, y normalmente responde a clases que se identifican con estructuras similares pero significado diferente en terminos de negocio.

Consideremos el caso de las personas en el proyecto de clase, tenemos dos categorias de personas; la primera son los empleados que realizan las labores de tomar los pedidos en la compañía y la segunda son los clientes quienes pagan por los servicios que la compañía ofrecen.



A continuación veremos como podemos aplicar cada una de las estrategias al ejemplo anterior.

## Herencia de una sola tabla

Es la estrategia mas simple y normalmente la de mejor rendimiento. En esta estrategia todas las entidades (en el ejemplo tanto Empleado como Cliente) pertenecientes a la jerarquia seran almacenadas en una unica tabla que tendra una columna indicando el tipo de entidad que representa cada registro; a esta columna se le conoce como discriminador y JPA provee la anotación @DiscriminatorColumn para defininir el nombre de dicha columna en la super clase y la anotación @DiscriminatorValue para indicar el valor que dicha columna va a tomar para cada una de las subclases.

Ejemplo:

@Entity

@Inheritance

@DiscriminatorColumn(name="OPTIPO\_PERSONA")

@ForceDiscriminator

@Table(name="TGF\_PERSONAS")

public abstract class Persona implements Serializable {

@EmbeddedId

private PersonaPK personaPK;

@Column(name = "DSNOMBRE")

private String nombre;

…

@Entity

@DiscriminatorValue("empleado")

public class Empleado extends Persona {

…

@Entity

@DiscriminatorValue("cliente")

public class Cliente extends Persona {

…

En el ejemplo anterior tanto clientes como empleados seran almacenados en la misma tabla (TGF\_PERSONAS); la cual tendra la columna “OPTIPO\_PERSONA” que en caso que sea un empleado tomará el valor “empleado” y en caso que el registro corresponda a un cliente tomará el valor “cliente”.

## Herencia de multiples tablas unidas

Es el tipo de herencia mas logica debido a que el modelo objetual es reflejado por el modelo de base de datos. En este tipo de herencia una tabla es creada para cada una de las entidades; dicha tabla almacenará los atributos únicos de cada clase. Cada tabla tambien almacenará la clave primaria, la cual será definida unicamente en la superclase.

**Ejemplo:**

@Entity

@Inheritance(strategy=InheritanceType.JOINED)

@DiscriminatorColumn(name="OPTIPO\_PERSONA")

@Table(name="TGF\_PERSONAS")

public abstract class Persona implements Serializable {

@EmbeddedId

private PersonaPK personaPK;

@Column(name = "DSNOMBRE")

private String nombre;

…

@Entity

@DiscriminatorValue("empleado")

@Table(name="TGF\_EMPLEADOS")

public class Empleado extends Persona {

…

@Entity

@DiscriminatorValue("cliente")

@Table(name="TGF\_CLIENTES")

public class Cliente extends Persona {

…

## Tabla por clase

Esta estrategia permite agregar herencia al modelo objetual, cuando esta no existe en el modelo de datos. En esta estrategia una tabla es definida por cada una de las subclases, y dicha tabla almacena todos los atributos de la superclase y la subclase. No todos los providers soportan esta estrategia.

**Ejemplo:**

@Entity

@Inheritance(strategy=InheritanceType.TABLE\_PER\_CLASS)

public abstract class Persona implements Serializable {

@EmbeddedId

private PersonaPK personaPK;

@Column(name = "DSNOMBRE")

private String nombre;

…

@Entity

@Table(name="TGF\_EMPLEADOS")

public class Empleado extends Persona {

…

@Entity

@Table(name="TGF\_CLIENTES")

public class Cliente extends Persona {

…

## MappedSuperClass

Esta estrategia permite definir una tabla diferente para almacenar todos los atributos de cada subclase y su superclase. Una clase anotada como MappedSuperClass no es una entidad en consecuencia no puede ser consultada, solo pueden consultarse las subclases.

**Ejemplo:**

@MappedSuperclass

public class Persona implements Serializable {

@EmbeddedId

private PersonaPK personaPK;

@Column(name = "DSNOMBRE")

private String nombre;

…

@Entity

@Table(name = "TGF\_CLIENTES")

public class Cliente extends Persona {

…

@Entity

@Table(name = "TGF\_EMPLEADOS")

public class Empleado extends Persona {

…