Relaciones paso a paso

# Primeros pasos

1. Crear la base de datos sin datos
2. En el fichero HibernateUtil:
   1. configurar password, user y base de datos
   2. añadir la clase anotada en la metadatasources. /!\ OJO. Aquí no hacemos uso de SET. Añadimos repitiendo el nombre del método.
3. Crear el modelo
   1. Nombre de clase
   2. Propiedades
   3. Constructores (sin parámetros y con parámetros)
   4. Getters Setters y (el To String 🡪 Opcional)
   5. Añadir las anotaciones al nivel del nombre de la clase: Entidad, nombre de la tabla…
   6. Añadir las anotaciones al nivel de cada propiedad para la columna y la validación.
   7. Añadir la llave primaria y la estrategia de generación de IDs
4. Probar el modelo. Ejecutar el programa MainPruebas(). No hace falta crear ningúna llamada a la clase. Se creará solas las clases que hemos configuradas al llevar la anotación @Entity

# OneToOne

1. Configurar el 1to1 desde la segunda clase
   1. Crear el modelo con el que vamos a conectar la 1era clase haciendo lo que hemos hecho en el punto 3
   2. Crear la propiedad para conectar la 1era clase con la segunda
      1. Si nuestra segunda clase se llama ProviderDetail, crearemos la propiedad providerDetail.
   3. Crear los getter setters
   4. Poner la anotación @OneToOne al nivel de la propiedad de la segunda clase.
   5. A este nivel, se puede añadir el parámetro fetch (sabiendo que tendríamos que poner LAZY ya que está configurado EAGER por defecto).
   6. Añadir la anotación @JoinColumna para indicar el nombre de la columna pivot entre las dos tablas. A priori será id.
2. Configurar el 1to1 desde la primera clase
   1. Poner la anotación @OneToOne al nivel del nombre de la propiedad de la primera clase.
   2. A este nivel, se puede añadir los parámetros cascade (A priori CascadeType.ALL), mappedBy (nombre de la primera clase con la primera letra en minúsculas) y fetch (sabiendo que tendríamos que poner LAZY ya que está configurado EAGER por defecto).
3. Crear la clase OneToOne
   1. En un método estático, crear una instancia de la primera clase y otra de la segunda.
   2. Reemplazando los elementos entre [] por los nombre adecuados, usar el método [objetoSegundaClase].set[PrimeraClase]([objetoPrimeraClase]); para conectar la primera clase a la segunda.
   3. Hacer lo mismo en el otro sentido.
   4. Hacer uso del método session.persist([objetoPrimeraClase])
   5. Probar.

# OneToMany

1. Crear los modelos de la tercera (product?) y de la cuarta (category?) clase como indicado en los puntos anteriores.
2. Configurar el ManyToOne desde la tercera cuarta clase.
   1. Añadir columna pivot ([nombre\_cuarta\_clase\_en\_minusculas\_id])
   2. Hacer los getters setters
3. Configurar el OneToMany desde la tercera clase
   1. Considerar no poner CascadeType.All pero las demás constantes ya que podríamos provocar catástrofe en caso de borrar una fila de esta cuarta clase.
   2. Indicar el mapeo ([nombreCuartaClaseCon1eraLetraEnMinusculas])
4. Crear la clase OneToMany
   1. En un método estático, crear una instancia de la tercera clase y otra de la cuarta.
   2. Reemplazando los elementos entre [] por los nombre adecuados, usar el método [objetoTerceraClase].set[CuartaClase]([objetoCuartaClase]); para conectar la tercera clase a la cuarta.
   3. Añadir cada objeto de la tercera clase a la lista Get[NombreTerceraClase].
   4. Hacer uso del método session.persist([objetoPrimeraClase])
   5. Probar.

# ManyToMany

1. Configurar las relaciones entre la 1era y la 3ra clase.
2. Ojo con el Eager y el CascadeType
3. Mapear desde la clase la menos importante de las 2
4. Crear la clase ManyToMany
5. Probar.