**Hibernate аннотации**

**1. @Entity** – эта аннотация помечает те классы которые связанны с базой данных. Если мы хотим дать понять hibernate, что класс связан с базой данных — эти классы надо пометить аннотацией **@Entity**. Все классы с аннотацией **@Entity** должны иметь:

**1.** Пустой конструктор

**2.** Хотя бы одно поле с аннотацией **@Id**.

**2. @Table(name = <table name>)** - указывается название той таблицы котрая есть в базе данных и которая соответствует сущности(классу) над котрым мы пишет эту аннотацию. Если имя сущности и имя талицы совпадают, то использовать эту аннотацию не обязательно. Но лучше её всё равно использовать, чтобы было более наглядно.

**3. @Column(name = <column name>)** – используется для сопостовления каждого поля класса с каждым столбцом таблицы базы данных. В аргументе name указываем назване столбца из таблицы которое соответствует полю нашего класса.

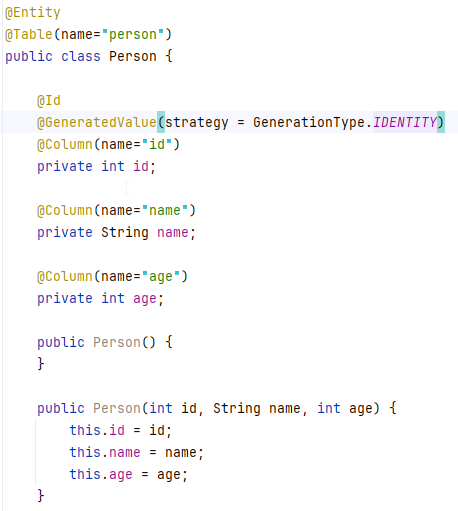
**4. @Id** – указывается для первичных ключей (**Primary key**). С помощью этой аннотации даём понять **hibernate**`у, что это поле особенное и оно соответствует столбцу **Id** из таблицы которое является первичным ключом (**Primary key**).  
  
**5. @GeneratedValue(<strategy>)** - указывает, что поле с этой аннотацией будет генерироваться автоматически с помощью postgres`a. Используется для поле **Id**.

В этой аннотации мы должны указать стратегию (**strategy)** с помощью которой генерируется для этого поля. Есть разные стратегии:

**1. IDENTITY** – при этой стратегии **hibernate** не будет трогать эту колонку, значение этой колонки будет генерироваться на стороне базы данных.

**2. SEQUENCE** – используется для ручной генерации **id** с помощью **sequence**`a.

**3. TABLE** – используется довольно редко.



**6. @Enumerated** - Этой аннотацией мы помечаем наше поле которое является перечислением чтобы **Hibernate** смог сохранить это поле в таблицу. В Параметрах этой аннотации нужно указать способ сохранения этого перчисления в таблице.

Возможные параметры:

**EnumType.Ordinal (стоит по умолчнатю)** – каждому значению перечисления выдает индекс начиная с 0. Именно этот индекс сохраняется в таблице с помощью Hibernate, тоесть Hibernate автоматически переводить yаш enum в целое число. Этот способ довольно эффективный так как целое число занимает довольно мало места, но у этого способа так же есть недостаток. Если мы в самом перечислении значения поменяем местами и при этом в базе данных будут содержаться какие-то записи, то у нас всё перемешается.

**EnumType.String** – сохраняет название перечисления в таблицу. Этот способ хорош тем, что если мы будем менять местами наши значения перечислений, то это никак не будет влиять на нашу работоспособность в приложении, так как индексы значений перечислений теперь не имеют значения. Но этот способ не очень эффективный потому-что строка заниает больше места в памяти.



**7. @Transient** – помечает те поля в сущности, которые мы не хотим связывать с таблицей в БД.

Поле помеченное аннотацией **@Transient** не будет сохраняться в БД и **Hibernate** вообще не будет его «замечать».

Такое поле полезно использовать для динамически вычисляемых полей.

