<http://blog.sina.com.cn/s/blog_7085382f0100uk4p.html>

基于代码复用和模型分离的思想，在项目开发中使用JPA的@MappedSuperclass注解将实体类的多个属性分别封装到不同的非实体类中。

****1.****@MappedSuperclass注解只能标准在类上：@Target({java.lang.annotation.ElementType.TYPE})

****2.****标注为@MappedSuperclass的类将不是一个完整的实体类，他将不会映射到数据库表，但是他的属性都将映射到其子类的数据库字段中。

****3.****标注为@MappedSuperclass的类不能再标注@Entity或@Table注解，也无需实现序列化接口。

但是如果一个标注为@MappedSuperclass的类继承了另外一个实体类或者另外一个同样标注了@MappedSuperclass的类的话，他将可以使用@AttributeOverride或@AttributeOverrides注解重定义其父类(无论是否是实体类)的属性映射到数据库表中的字段。

比如可以重定义字段名或长度等属性，使用@AttributeOverride中的子属性@Column进行具体的定义。

****注意：对于其父类中标注@Lob注解的属性将不能重载，****并且@AttributeOverride里的@Column设置都将不起作用。

JPA规范中对@Lob注解并没有说明不能同时标注@Column注解，但是在实际使用中Hibernate JPA不支持这中标注方式。

****4.****此外，这样的类还可以直接标注@EntityListeners实体监听器，他的作用范围仅在其所有继承类中，并且实体监听器同样可以保被其子类继承或重载。

****5.****标注为@MappedSuperclass的类其属性最好设置为protected或default类型的，以保证其同一个包下的子类可以直接调用它的属性。便于实体监听器或带参数构造函数的操作。

****6.****由于标注为@MappedSuperclass的类将不是一个完整的实体类，因此其不能标注@Table，并且无法使用@UniqueConstraint设置字段的Unique属性，这一点以及对属性类型重载(如重载标注为@Lob的属性)的支持ＪＰＡ规范还有待改进。

****7.****可以同时标注@DiscriminatorValue注解，以设定实体子类的实体标识字段的值。该属性一般是在实体继承的时候使用的较多，但是在实体映射的时候可以不用设置。

****8.****比较实体继承与实体映射的****区别****：

实体继承的三种策略分别是：SINGLE\_TABLE(所有继承的实体都保存在同一张数据库表中),JOINED(每个实体子类都将保存在一个单独的表中),TABLE\_PER\_CLASS(有继承关系的所有实体类都将保存在单独的表中)。

实体映射最类似于JOINED实体继承方式，他也是将实体子类单独保存为一张表，但是两者最大的区别就在于：查询的时候JOINED使用的是多态查询，在查询父类时其所有实体子类的数据也将同时被查询出，因此查询时间和性能都将有影响。但是实体映射方式的数据库查询等同于没有实体继承关系的查询，也就是说，他仅在实体层体现出一种继承的关系却并没有在数据库中体现这样一种关系，他的操作都是独立的并且将不会影响到实体子类。