<http://blog.csdn.net/u010786902/article/details/50902921>

版权声明：本文为博主原创文章，未经博主允许不得转载。

（例）： @Transient表示该属性并非一个到数据库表的字段的映射,ORM框架将忽略该属性.   
如果一个属性并非数据库表的字段映射,就务必将其标示为@Transient,否则,ORM框架默认其注解为@Basic  
    示例:  
    //根据birth计算出age属性

**[java]** [view plain](http://blog.csdn.net/u010786902/article/details/50902921) [copy](http://blog.csdn.net/u010786902/article/details/50902921)

1. @Table(name="person")
2. **public** **class** Person{
4. **private** Date birth;
6. @Column(name = "birth")
7. **public** Date getBirth() {
8. **return** birth;
9. }
11. **public** **void** setBirth(Date birth) {
12. **this**.birth = birth;
13. }
15. @Transient
16. **public** **int** getAge(){
17. **return** getYear(**new** Date())-getYear(getBirth());    //getYear为类中的自定义方法，返回传入日期的年份
18. }
19. }

上面的这个转载对@Transient的作用说的不是很清晰。

说一点个人的心得，比如上面例子在一个持久化类person中定义了birth的属性并且用了@Column注解在数据表中进行了映射，而getAge()方法注解为@Transient，表明在数据库表中并没有age这个映射字段。

         当你通过字段映射从数据库获取数据时，因为getAge()加了@Transient，并且在数据库中没有字段。所以getAge()没有从数据库中获取数据，而是通过getBirth()方法获取数据库中返回的birth，并通过计算返回getAge()的值。

return getYear(new Date())-getYear(getBirth());

当你在程序中获取到持久化类person的对象时，person.getAge()和person.getBirth()都有数据返回提供给你使用。

         这样做的话，能减少数据库的字段,比如这个例子中就不需要新建一个关于age的数据库字段了，通过birth字段和计算就能获取到person的age属性。@Transient还可以在持久化类中直接获取关联表中的字段值，许多业务逻辑中都有很大的作用。

个人理解，如有不足之处，望指正。