@Transient打在get方法上表示该属性不被持久化

@ManyToOne的fetch默认为EAGER

@OneToMany的fetch默认是LAZY,它的mappedBy,以User(有一属性userGroup[UserGroup类型的]表示这个用户属于哪个组)和UserGroup为例子

在UserGroup端对User有 OneToMany(mappedBy="userGroup") Set<User> getUsers(){…};

表示获得User集合的方法是 select \* from User as u where u.g\_id = userGroup.id

即双方的关系保存在User表中而不是第三章表中

@OneToOne //比如一个丈夫对应一个妻子 在丈夫这方有妻子的id

@JoinColumn(name="wife\_no\_id") //可以显示指定列名 否则列名自动是 wife\_id

@Immutable某个属性不可更改(可以优化程序)

@Proxy(lazy=false) 这样后这个类就不是lazy的

所以即使你用load方法获取一个对象 效果等同于get方法

@NotNull

@Enumerated(EnumType.STRING)

@Lob

@Lob indicates that the property should be persisted in a Blob or a Clob depending on the property type: java.sql.Clob, Character[], char[] and java.lang.String will be persisted in a Clob. java.sql.Blob, Byte[], byte[] and Serializable type will be persisted in a Blob.

@Lob

**public** String getFullText() {

**return** fullText;

}

@Lob

**public** byte[] getFullCode() {

**return** fullCode;

}

@Formula("obj\_length \* obj\_height \* obj\_width")

**public** long getObjectVolume()

用于计算字段,并没有真正存到数据库里

@MappedSuperclass

**public** **class** BaseEntity {

@Basic

@Temporal(TemporalType.TIMESTAMP)

**public** Date getLastUpdate() { ... }

**public** String getLastUpdater() { ... }

...

}

@Entity **class** Order **extends** BaseEntity {

@Id **public** Integer getId() { ... }

...

}

在Person里有

@ManyToOne

@JoinTable(name = "pg", joinColumns = @JoinColumn(name = "p\_id"), inverseJoinColumns = @JoinColumn(name = "g\_id"))

**public** Group getGroup() {

**return** group;

}

意思是Person和Group的关系(many to one)保存在第三张表(pg)里,其中person用p\_id识别,group用g\_id识别

因此在Person表里就不会有group\_id的字段了

@OneToMany(mappedBy="customer")

@OrderBy("number desc")

public List<Order> getOrders() { return orders; }

@OneToMany(mappedBy="customer")

@OrderColumn(name="orders\_index")

public List<Order> getOrders() { return orders; }

@OneToMany(mappedBy="customer")

@MapKey(name="number")

public Map<String,Order> getOrders() { return orders; }

@AttributeOverrides({

@AttributeOverride(name="street1", column=@Column(name="fld\_street"))

})

它的作用应该是将一些字段拼凑成一个对象

@Embedded

@AttributeOverrides({

@AttributeOverride(name="startDate", column=@Column("

EMP\_START")),

@AttributeOverride(name="endDate", column=@Column("EMP\_END"))

})

public EmploymentPeriod getEmploymentPeriod() { ... }

排序

@OneToMany(cascade=CascadeType.ALL, fetch=FetchType.EAGER)

@JoinColumn(name="CUST\_ID")

@Sort(type = SortType.COMPARATOR, comparator = TicketComparator.**class**)

**public** SortedSet<Ticket> getTickets() {

**return** tickets;

}

sortedUsers = s.createFilter( group.getUsers(), "order by this.name" ).list();

一对多

一般是在"多"端保存"一"端的id

一般hibernate的命名规则是 一端的名字\_id

如果要显示更改这个字段的名字

需要在many一方指定@JoinColumn(name="新名字")

在one的一方要mappedBy="one在many一方的类的属性名"