JPA映射持久化对象（Entity）

标签： jpa 映射 持久化对象 entity criteria 查询

2017-05-06 12:38 91人阅读 评论(0) 收藏 编辑 删除

分类： JPA

推荐阅读：JPA criteria 查询:类型安全与面向对象

来源： http://blog.sina.com.cn/s/blog_49fd52cf0100rzjn.html

一个普通的POJO类通过@Entity可以映射成为可持久化的类；
类JavaBean风格：
• 类属性必须为private；
• 有Getter和Setter方法；

映射实体：@Entity
@Entity实体必须有一个无参的构造方法；
实现Serializable接口，建议每一个Entity都实现该接口；
其中，Entity中，name属性表示实体的名称，比如：

```
@Entity(name=Contacts)
public class ContactsEO{
...
}
```

在JPA执行jpql的时候，需要使用Contacts作为实体名称，而不是ContactsEO。

```
String jpql = "select * from  Contacts  c";
```

如果name没有配置，则实体名默认为类名。

可继承性：
实体可继承，非实体类可以继承自实体类，实体类也可以继承自非实体类；
抽象类也可以标注为实体类；

标注主键：
一个实体类至少要有一个主键（Primary Key）；
使用@Id标注为实体主键；

默认实体映射：
一个类标注了@Entity的可持久化类，如果不标注其他任何注释，该类的属性和方法自动映射为数据库中默认的表和字段。如：

```
@Entity
public class ContactEO implement Serializable{
        public ContactEO(){}
        private Integer id;
        private String name;
        //getter和setter方法略
}
```

则该实体默认对应的数据库表名为：contacteo，字段为：int id; varchar name;

映射表和字段：
@Table注释可定义映射的表；
@Column注释可定义映射到字段；

映射表@Table：

```
@Target({TYPE}) @Retention(RUNTIME)
public @interface Table{
```

```
        String name() default "";
        String catalog() default "";
        String schema() defalut "";
        UniqueConstraint[] uniqueConstraints() default {};
}
# @Table必须标注在类名前；
# name属性表示实体所对应表的名称；
# catalog和schema属性表示实体指定点目录名称或数据库名称；
# uniqueConstraints属性表示该实体所关联的唯一约束条件，一个实体可以有多个唯一约束条件，默认没有约束；
# 若使用uniqueConstraints标记时，徐奥配合标记UniqueConstraint标记来使用；
示例：
@Entity
@Table(name="contact", schama="jpadb", uniqueConstraints={
        @UniqueConstraint(
                columnNames = {"name", "email"}
        )},
        @UniqueConstraint(
                  columnNames = {"other_col_1", "other_col_2"}
        )}
)
# 说明：在注释中，属性值是不区分大小写的。
```

映射方法和属性（@Column）

```
@Column标记表示所持久化属性所映射表中的字段。
@Target({METHOD, FIELD}) @Retention(RUNTIME)
public @interface Column{
        String name() default "";
        boolean unique() default false;
        boolean nullable() default false;
        boolean insertable() default false;
        boolean updateable() default false;
        String columnDefinition() default "";
        String table() defalut "";
        int lenght() default 255;
        int precision() default 0;
        int scale() default 0;
}
# 标记可以标注在Getter方法前或属性前；
# name为字段名；
# unique为字段是否唯一标识，默认为false；
# nullable为该字段是否可以为null值，默认为true；
# insertable为使用“INSERT”脚本插入数据时，是否需要插入该字段的值；
# updateable为使用“UPDATE”脚本更新数据时，是否需要更新该字段的值；
以上两个多用于只读属性，比如主键或外键等，这些字段通常为自动生成的。
# columnDefinition表示创建表时，该字段创建的SQL语句，一般用于通过Entity生成表定义时使用；
# table属性表示当映射多个表时，指定表的表中的字段，默认值为主表的表名；
# lenght表示字段的长度；
# precision和scale均表示精度；
示例：
@Column(name="contact_name", nullable=false, length=200)
@Column(name="contact_name", nullable=false, columnDefinition="clob not null")
```

映射优化：

```
• 基本类型 VS 封装类型
# 当为null时，若此时Entity的对应属性的类型为int，则将一个null转换为int型必定产生转换异常；但是如果此时Entity属性类型为Integer，它是一个对象，对象的值可以为null，此时不会有问题；
# 建议标注实体的属性使用Java基本类型的包装类（这可能会牺牲一些转换的效率）；

• @Basic设置加载方式
@Target({METHOD, FIELD}) @Retention(RUNTIME)
public @interface Basic {
        FetchType fetch() default EAGER;
        boolean optional() default true;
}
# 默认Entity的属性加载方式都是即时加载（EAGER）；
# 两种加载方式：
        LAZY惰性加载
```

EAGER即时加载（默认）
示例：@Basic(fetch=FetchType.EAGER)

主键映射：
主键标识@Id
@GeneratedValue主键生成策略：TABLE, SEQUENCE, IDENTITY, AUTO.
@GeneratedValue(strategy = GenerationType.SEQUENCE - Oracle
@GeneratedValue(strategy = GenerationType.IDENTITY - SQL server
生成策略：自增主键、表生成器（@TableGenerator）、Sequence生成器（@Sequence）、Identity生成器、复合主键（@IdClass）和嵌入式主键（@Embeddedld）。
JPA可定义的生成主键策略比较：
• SEQUENCE，IDENTITY主要针对一些特殊的数据库，未确定系统要支持的数据库类型时，最好不要使用。
• AUTO用于比较简单的主键，对主键生成策略要求少
• TABLE生成策略是将主键的值持久化在数据库的表中，因为只要是关系型数据库，都可以创建一个表，专门来保存生成的值，这样就消除了数据库之间的不兼容性，既能保证支持多种数据库，又有一定的灵活性，建议使用。
• 如果以上方法不能满足需求时，可以通过一定的规则来设置主键的值，可以使用UUID，使其在程序中自动生成，然后映射到实体的主键上，不能通过JPA的主键生成策略来实现。

示例：
@Id
@GeneratedValue(strategy=GenerationType.AUTO)

映射特殊类型：
• 映射Blob和Clob类型（@Lob）
Blob和Clob类型都可以通过@Lob属性来标注（text类型字段也可以用该属性标注）；
Clob（Character Large Object）类型是长字符串类型，映射到为实体中的类型可以为char[]、Charater[]或者String类型；
Blob（Binary Large Object）类型是字节类型，映射为实体中的类型可为byte[]、Byte[]或者实现了Serializable接口的类；
因为上述两种类型的数据一般占用内存比较大，因此通常使用惰性加载；

• 映射时间（Temporal）类型（@Temporal）
时间日期类型，需要用@Temporal来标注；
TemporalType枚举类型定义为：DATE, TIME, TIMESTAMP；默认为TemporalType.TIMESTAMP；

• 映射枚举（Enumerated）型
通过@Enumerated类标注枚举类型；
使用@Enumerated时需要注意：
枚举类型有两个属性：名称和值。通过EnumType来定义：ORDINAL, STRING
ORDINAL表示持久化的为枚举类型的值；STRING表示持久化的为枚举类型的名称；
建议使用ORDINAL类型来持久化枚举类型；
枚举类型的定义位置（实体内部或外部）根据具体情况而定。

• 映射非持久化类型（@Transient）
如果实体中有属性或者Getter方法并不需要持久化，则需要使用@Transient来标注。

- 上一篇 Excel 报表导入导出
- 下一篇 Eclipse 反编译插件

相关文章推荐

- JavaBean持久化
- Java持久化bean原理（一）
- JavaBean 持久化
- 【EJB基础】Persistence Bean（持久化Bean）
- Ejb-开发持久化Bean
- javabean规范中要求实现Serializable接口，有什...
- 实体 Bean@与数据库映射
- 什么是Java bean
- EJB---->实体Bean(Entity Bean) 持久化 和 persis...
- 理解JPA，第一部分：面向对象的数据持久化方案