SSH 框架注解开发笔记

作者：liuyishi
原文链接：https://ld246.com/article/1517382969783
来源网站：链滴

实体类用注解

基本注解

常用的注解有: @Entity, @Table, @Id, @GeneratedValue, @Column, @Basic，@Transient, @Tmporald等.

● @Entity

@Entity 标注用于实体类声明语句之前，指出该Java 类为实体类，将映射到指定的数据库表。如声明个实体类 Customer，它将映射到数据库中的 customer 表上。

● @Table

当实体类与其映射的数据库表名不同名时需要使用 @Table 标注说明，该标注与 @Entity 标注并列用，置于实体类声明语句之前，可写于单独语句行，也可与声明语句同行。

@Table 标注的常用选项是 name，用于指明数据库的表名

@Table标注还有一个两个选项 catalog 和 schema 用于设置表所属的数据库目录或模式，通常为数库名。uniqueConstraints 选项用于设置约束条件，通常不须设置。

● @Id

@Id 标注用于声明一个实体类的属性映射为数据库的主键列。该属性通常置于属性声明语句之前，可声明语句同行，也可写在单独行上。

@Id标注也可置于属性的getter方法之前。

● @GeneratedValue

@GeneratedValue 用于标注主键的生成策略，通过 strategy 属性指定。默认情况下，JPA 自动选一个最适合底层数据库的主键生成策略：SqlServer 对应 identity，MySQL 对应 auto increment。

在 javax.persistence.GenerationType 中定义了以下几种可供选择的策略：

- IDENTITY：采用数据库 ID自增长的方式来自增主键字段，Oracle 不支持这种方式；

- AUTO： JPA自动选择合适的策略，是默认选项；

- SEQUENCE：通过序列产生主键，通过 @SequenceGenerator 注解指定序列名，MySql 不支持这方式

- TABLE：通过表产生主键，框架借由表模拟序列产生主键，使用该策略可以使应用更易于数据库移。

● @Basic

@Basic 表示一个简单的属性到数据库表的字段的映射,对于没有任何标注的 getXxxx() 方法,默认即为Basic

fetch: 表示该属性的读取策略,有 EAGER 和 LAZY 两种,分别表示主支抓取和延迟加载,默认为 EAGER.

optional:表示该属性是否允许为null, 默认为true

● @Column

当实体的属性与其映射的数据库表的列不同名时需要使用@Column 标注说明，该属性通常置于实体属性声明语句之前，还可与 @Id 标注一起使用。

@Column 标注的常用属性是 name，用于设置映射数据库表的列名。此外，该标注还包含其它多个性，如：unique 、nullable、length等。

@Column 标注的 columnDefinition 属性: 表示该字段在数据库中的实际类型.通常 ORM 框架可以据属性类型自动判断数据库中字段的类型,但是对于Date类型仍无法确定数据库中字段类型究竟是DATE

TIME还是TIMESTAMP.此外,String的默认映射类型为VARCHAR, 如果要将 String 类型映射到特定数库的 BLOB 或TEXT 字段类型.

@Column标注也可置于属性的getter方法之前.

多表常用注解

一对多: 客户与订单

新建订单表实体类，并与客户表建立实体关系。

```
@Entity
@Table(name="w_customer")

@NamedQuery(name="Customer.findAll",query="from Customer")
@NamedQueries(value=@NamedQuery(name="Customer.findAll2",query="from Customer"))
public class Customer {
    @Id
    @GeneratedValue
    private Integer id;
    private String name;
    private String city;
        @OneToMany(mappedBy="customer",//by 1 的一方 ： 在Order实体类中属性customer
                cascade=CascadeType.ALL,//级联  不包含 all-delete-orphan
                fetch=FetchType.LAZY,//懒加载
                orphanRemoval=true,//支持孤儿删除 delete-orphan
                targetEntity=cn.it.entity.Order.class//一般忽略 ， Order是接口的化，这里配置实现类
指定
                )
    private Set<Order> orders = new HashSet<Order>();
}

@Entity
@Table(name="w_order")//表名
public class Order {
    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    private Integer id;
    private String name;
    private Double price;
    @ManyToOne(cascade=CascadeType.ALL)
    @JoinColumn(name="c_id")//自定义关联外键属性  映射表中字段名为c_id 覆盖了默认的custom
r_id
    @Fetch(FetchMode.SELECT)//单表查询
    @LazyToOne(LazyToOneOption.PROXY)//代理懒加载
    private Customer customer;
}
```

多对多: 学生与课程

```
@Entity
@Table(name="t_course")
public class Course {
```

```
        //注解主键
        @Id
        @GenericGenerator(name="wc",strategy="assigned")
        @GeneratedValue(generator="wc")//生成随机数
        private Integer id;
        private String name;
        @ManyToMany
        @JoinTable(name="t_student_course",//配置中间表
        joinColumns=@JoinColumn(name="c_id"),//自定义关系表name
        inverseJoinColumns=@JoinColumn(name="s_id")//对方的外键名称
        )
        private Set<Course> course=new HashSet<Course>();
}



@Entity
@Table(name="t_student")
public class Student {
    //注解主键
    @Id
    @GenericGenerator(name="wc",strategy="assigned")
    @GeneratedValue(generator="wc")//生成随机数
    private Integer id;
    private String name;
    @ManyToMany
    @JoinTable(name="t_student_course",//配置中间表
    joinColumns=@JoinColumn(name="s_id"),//自定义关系表name
    inverseJoinColumns=@JoinColumn(name="c_id")//对方的外键名称
    )
    private Set<Course> course=new HashSet<Course>();
}
```

Tips

mappedBy 属性跟 xml 配置文件里的inverse一样。在一对多或一对一的关系映射中，如果不表明 mppedBy 属性，默认是由本方维护外键。但如果两方都由本方来维护的话，会多出一些 update 语句性能有一定的损耗。

解决的办法就是在一的一方配置上 mappedBy 属性，将维护权交给多的一方来维护，就不会有 upde 语句了。

至于为何要将维护权交给多的一方，可以这样考虑：要想一个国家的领导人记住所有人民的名字是不能的，但可以让所有人民记住领导人的名字！

注意，配了 mappedBy 属性后，不要再有 @JoinColumn，会冲突！

applicationContext.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:context="http://www.springframework.org/schema/context"
    xmlns:aop="http://www.springframework.org/schema/aop" xmlns:tx="http://www.springframework.org/schema/tx"
```

```
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context.xsd
        http://www.springframework.org/schema/tx
        http://www.springframework.org/schema/tx/spring-tx.xsd
        http://www.springframework.org/schema/aop
        http://www.springframework.org/schema/aop/spring-aop.xsd
        ">

    <!-- 数据源 -->
    <bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource">
        <property name="driverClassName" value="com.mysql.jdbc.Driver"/>
        <property name="url" value="jdbc:mysql://localhost:3306/erp?characterEncoding=utf-8
/>
        <property name="username" value="root"/>
        <property name="password" value="root"/>
    </bean>

    <bean id="sessionFactory" class="org.springframework.orm.hibernate5.LocalSessionFactor
Bean">
        <property name="dataSource" ref="dataSource"/>
        <property name="hibernateProperties">
            <props>
                <prop key="hibernate.dialect">org.hibernate.dialect.MySQL5Dialect</prop>
                <prop key="hibernate.show_sql">true</prop>
                <prop key="hibernate.format_sql">false</prop>
                <prop key="hibernate.hbm2ddl.auto">update</prop>
            </props>
        </property>
        <!-- 自动扫描 entity 包下的实体类上的注解 -->
        <property name="packagesToScan" value="xin.liuyishi.erp.entity">
        </property>
    </bean>

    <bean id="transactionManager" class="org.springframework.orm.hibernate5.HibernateTra
sactionManager">
        <property name="sessionFactory" ref="sessionFactory"></property>
    </bean>

    <tx:annotation-driven transaction-manager="transactionManager"/>

    <!-- 自动扫描 action 层/ biz 层/ dao 层 impl 包下的类上的注解 -->
    <context:component-scan base-package="xin.liuyishi.erp.action, xin.liuyishi.erp.dao.impl, x
n.liuyishi.erp.biz.impl"></context:component-scan>

</beans>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://java.sun
```

```
com/xml/ns/javaee" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.c
m/xml/ns/javaee/web-app_2_5.xsd" version="2.5">
  <display-name>erp_web</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>

  <listener>
      <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class

    </listener>
    <context-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>classpath:applicationContext.xml</param-value>
    </context-param>

    <filter>
        <filter-name>openSessionInView</filter-name>
        <filter-class>org.springframework.orm.hibernate5.support.OpenSessionInViewFilter</fi
ter-class>
    </filter>

    <filter-mapping>
      <filter-name>openSessionInView</filter-name>
      <url-pattern>/*</url-pattern>
    </filter-mapping>

    <filter>
      <filter-name>struts2</filter-name>
      <filter-class>org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter</filter
class>
      <init-param>
        <!-- 采用动态方法调用，例：URL为 http://127.0.0.1:8080/erp/user!login.action 就可以
用名称为 user的Action 中的 login 方法 -->
        <param-name>struts.enable.DynamicMethodInvocation</param-name>
        <param-value>true</param-value>
      </init-param>
    </filter>

    <filter-mapping>
      <filter-name>struts2</filter-name>
      <url-pattern>/*</url-pattern>
    </filter-mapping>
</web-app>
```

Action

struts2 使用注解方式必须依赖于struts2-convention-plugin

在 pom.xml 文件中添加如下依赖:

```
<dependency>
 <groupId>org.apache.struts</groupId>
 <artifactId>struts2-convention-plugin</artifactId>
 <version>2.3.24</version>
</dependency>
```

```
@Controller("depAction") // 控制层的Spring注解
@Scope("prototype") //支持多例
@ParentPackage("struts-default") //表示继承的父包
@Namespace(value = "/") // 表示当前Action所在命名空间
public class DepAction extends BaseAction<Dep> {

  private DepBiz depBiz;

  @Resource(name = "depBiz")
  public void setMyDepBiz(DepBiz depBiz) {
  this.depBiz = depBiz;
  setBaseBiz(depBiz);
  }

  @Override
 @Action(value = "dep_list", //表示action的请求名称
  results = { //表示结果跳转
  @Result(name = "success", location = "/success.jsp", type = "redirect"),
  @Result(name = "error", location = "/error.jsp", type = "redirect")
 })  public void list() {
  super.list();
  }
}
```