Java知识大全

# Spring 验证框架（JSR-303）

## 空检查

@Null 验证对象是否为空；

@NotNull 验证对象不为空；

@NotBlank 验证字符串不为空或者不是空字符串，比如””、” “都会验证失败；

@NotEmpty 验证对象不能为null，或者集合不为空。

## 长度检查

@Size(min=,max=) 验证对象长度，可支持字符串、集合；

@Length 字符串大小

## 数值校验

@Min 验证数字是否大于等于指定的值；

@Max 验证数字是否小于等于指定的值；

@Digits 验证数字是否符合指定格式，如@Digits(Integer=9,fraction=2);

@Range 验证数字是否在指定的范围内，如@Range（min=1，max=1000）

## 其他

@Email 验证是否为邮件格式，为null则不做校验；

@Pattern 验证Spring对象是否符合正则表达式的规则。

# Beetl模板引擎

## 设置定界符号和占位符

Beetl默认使用<% %>作为定界符号；使用${ }作为占位符号。也支持自定义定界符号和占位符号。常用的有：

@ 和回车作为定界符号 如:

@ var salary = user.salary \* 2;

<span>${user.name}</span>

<? ?> 类似PHP的定界符号 如:

<? var salary = user.salary \* 2; ?>

<span>${user.name}</span>

## 全局变量

request 标准的HTTPServletRequest，可以在模板中引用request属性(getter)，如${request. requestURL}

parameter 读取用户提交的参数，如${ parameter.userId}

# Spring Data JPA

## 相关配置

#----------------------

#持久层配置

#----------------------

#jpa规范使用的数据源

spring.jpa.database=MYSQL

#InnoDB是一种MySQL数据库引擎。MySQL5.5及之后使用它做为默认引擎。它提供了ACID兼容的事务（Transaction）功能，并提供外键支持

spring.jpa.database-platform=org.hibernate.dialect.MySQL5InnoDBDialect

#是否打印SQL

spring.jpa.show-sql=true

#是否自动建库

spring.jpa.hibernate.ddl-auto=update

### ddl-auto

 create：每次运行程序时，都会重新创建表，故而数据会丢失

 create-drop：每次运行程序时会先创建表结构，然后待程序结束时清空表

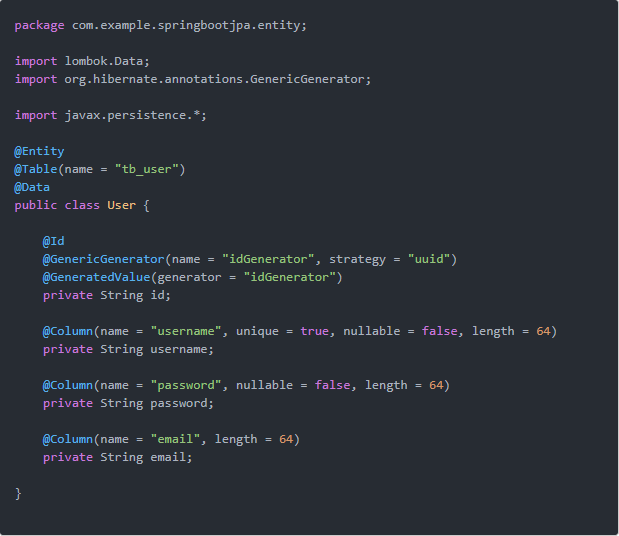
 upadte：每次运行程序，没有表时会创建表，如果对象发生改变会更新表结构，原有数据不会清空，只会更新（推荐使用）

 validate：运行程序会校验数据与数据库的字段类型是否相同，字段不同会报错

 none: 禁用DDL处理

## Demo实例

### 模型层（Model）



主键采用UUID策略  
@GenericGenerator是Hibernate提供的主键生成策略注解，注意下面的@GeneratedValue（JPA注解）使用generator = "idGenerator"引用了上面的name = "idGenerator"主键生成策略

#### JPA自带的几种主键生成策略

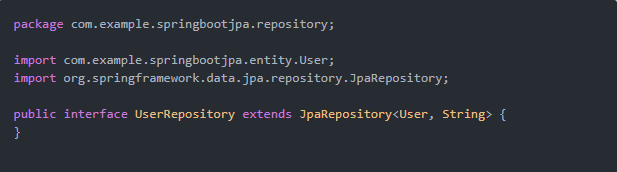
 TABLE： 使用一个特定的数据库表格来保存主键

 SEQUENCE： 根据底层数据库的序列来生成主键，条件是数据库支持序列。这个值要与generator一起使用，generator 指定生成主键使用的生成器（可能是orcale中自己编写的序列）

 IDENTITY： 主键由数据库自动生成（主要是支持自动增长的数据库，如mysql）

 AUTO： 主键由程序控制，也是GenerationType的默认值

### 持久层（Dao）



### 业务层（Service）

这里简单起见省略Service层

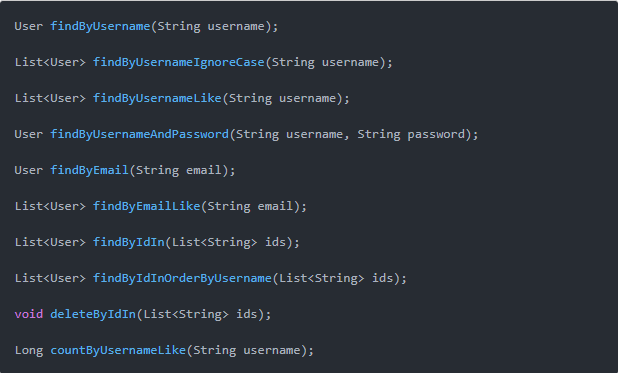
### Web控制器（Controller）



### 查询方法

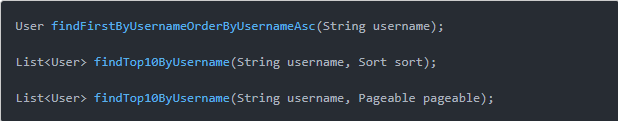
#### 查询创建Query Creation

Spring Data Jpa通过解析方法名创建查询，框架在进行方法名解析时，会先把方法名多余的前缀find…By, read…By, query…By, count…By以及get…By截取掉，然后对剩下部分进行解析，第一个By会被用作分隔符来指示实际查询条件的开始。 我们可以在实体属性上定义条件，并将它们与And和Or连接起来，从而创建大量查询：



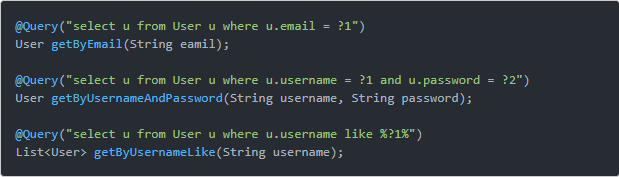
#### 限制查询结果

Spring Data Jpa支持使用first、top以及Distinct 关键字来限制查询结果，如：



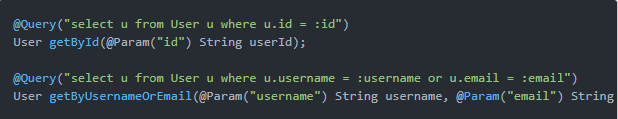
#### 自定义查询Using @Query

@Query 注解的使用非常简单，只需在声明的方法上面标注该注解，同时提供一个 JPQL 查询语句即可



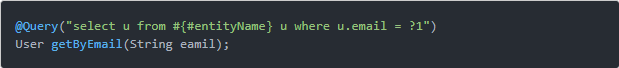
#### 使用命名参数Using Named Parameters

默认情况下，Spring Data JPA使用基于位置的参数绑定，如前面所有示例中所述。 这使得查询方法在重构参数位置时容易出错。 要解决此问题，可以使用@Param注解为方法参数指定具体名称并在查询中绑定名称，如以下示例所示：



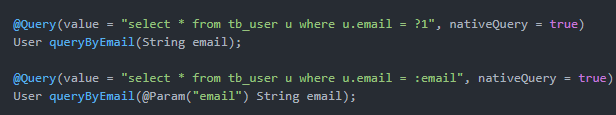
#### Using SpEL Expressions

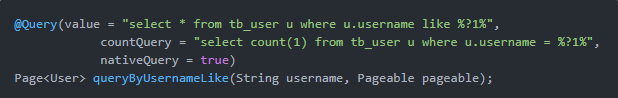
从Spring Data JPA release 1.4开始，Spring Data JPA支持名为entityName的变量。 它的用法是select x from #{#entityName} x。 entityName的解析方式如下：如果实体类在@Entity注解上设置了name属性，则使用它。 否则，使用实体类的简单类名。为避免在@Query注解使用实际的实体类名，就可以使用#{#entityName}进行代替。如以上示例中，@Query注解的查询字符串里的User都可替换为#{#entityName}



#### 原生查询Native Queries

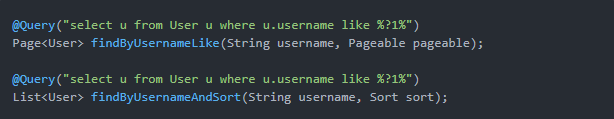
@Query注解还支持通过将nativeQuery标志设置为true来执行原生查询，同样支持基于位置的参数绑定及命名参数，如：



注意：Spring Data Jpa目前不支持对原生查询进行动态排序，但可以通过自己指定计数查询countQuery来使用原生查询进行分页、排序，如：

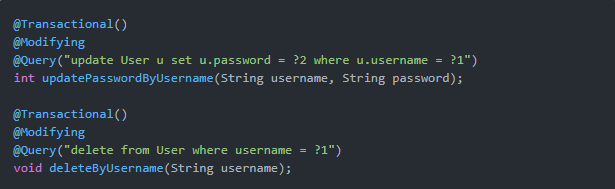
#### 分页查询及排序

Spring Data Jpa可以在方法参数中直接传入Pageable或Sort来完成动态分页或排序，通常Pageable或Sort会是方法的最后一个参数，如：



#### 自定义修改、删除 Modifying Queries

单独使用@Query注解只是查询，如涉及到修改、删除则需要再加上@Modifying注解，如：



注意：编辑查询只能使用void或int / Integer作为返回类型！

# 缓存机制

## 大企业缓存机制流程：

**Redis → ORM缓存 → DB**

# 其他

## @Deprecated

划线注解：该注解作用在方法上来声明——该方法后期不在进行更新，且随着jar包的升级，该方法将不在兼容，直至废除。

## 研究方向

缓存、并发、事务、创建实例的方式、设计模式

# 相关链接

## 玩转spring boot——properties配置

<https://www.cnblogs.com/GoodHelper/p/6260708.html>

## SpringBoot学习笔记九：Spring Data Jpa的使用

<https://www.jianshu.com/p/c23c82a8fcfc>