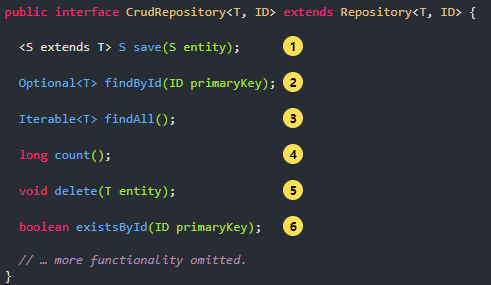
## 核心概念：

SpringDataRepository的核心是Respository接口。它以<T,ID>中的T（实体类型）和T里面的id类型作为参数来管理，例如：

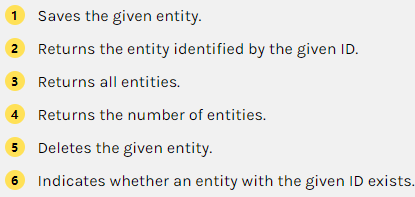
。此接口用作标记接口，主要是捕获要使用的类型和发现扩展此接口的接口。

### CrudRepository接口：

此接口继承了Repository接口，它为所管理的实体类提供了复杂的crud方法。



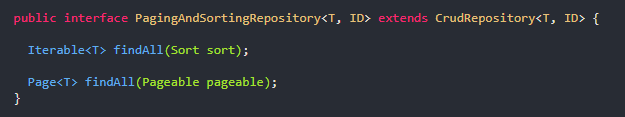
方法介绍：



#### 其他子接口

CrudRepository接口还有许多接口，比如JpaRepository和MongoRepository，PagingAndSortingRepository，它们不但具有CrudRepository接口的持久化功能，还扩展了其他功能。继承了CrudRepository的接口都会具有一些其他的派生功能，下面会讲到。

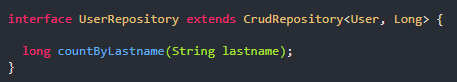
PagingAndSortingRepository：

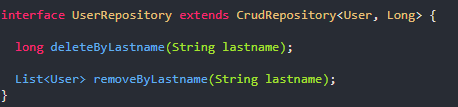


例子，查询第2页，然后每页显示20条数据：



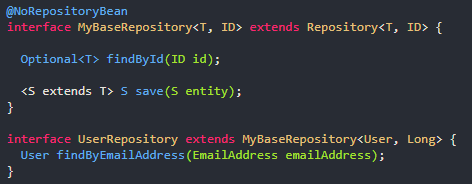
关于派生：





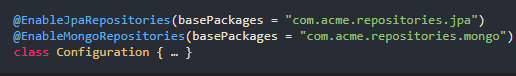
以上两个例子就是具备了count和delete的派生

#### 自定义接口



通过自定义接口来暴露需要的方法。@NoRepositoryBean注解告诉SpringData在运行时不要再创建对应的实例

#### 当配置了多个的SpringDataRepository





参考：<https://www.oschina.net/question/574036_2286640>

### 在接口中自定义查询方法的规则（根据方法名生成对应的sql）

Spring Data JPA框架在进行方法名解析时，会先把方法名多余的前缀截取掉，比如find，findBy，read，readBy，get，getBy，然后对剩下的部分进行解析。

假如创建如下的查询：findByUserDepUuid（），框架在解析该方法时，首先剔除findBy，然后对剩下的属性进行解析，假设查询实体为Doc，下面是查询解析步骤：

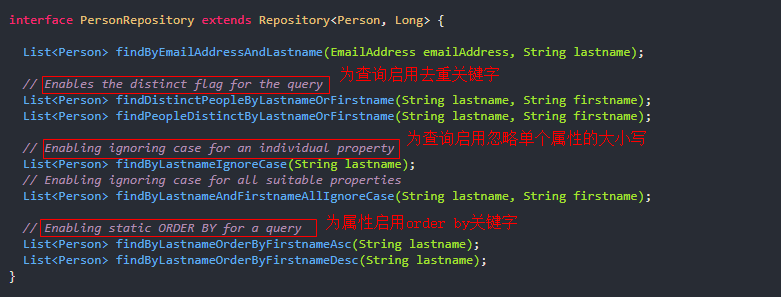
1、：先判断userDepUuid（根据POJO规范，首字母变为小写）是否为查询实体的一个属性，如果是，则表示根据该属性进行查询;如果没有该属性，继续第二步;

2、：从右往左截取第一个大写字母开头的字符串此处为UUID），然后检查剩下的字符串是否为查询实体的一个属性，如果是，则表示根据该属性进行查询;如果没有该属性，则重复第二步，继续从右往左截取;最后假设用户为查询实体的一个属性;

3、：接着处理剩下部分（DepUuid），先判断用户所对应的类型是否有depUuid属性，如果有，则表示该方法最终是根据“Doc.user.depUuid”的取值进行查询;否则继续按照步骤2的规则从右往左截取，最终表示根据“Doc.user.dep.uuid”的值进行查询。

4：可能会存在一种特殊情况，比如Doc包含一个用户的属性，也有一个userDep属性，此时会存在混合。可以明确在属性之间加上“\_”以显式表达意思，比如“findByUser\_DepUuid ）“或者”findByUserDep\_uuid（）“。

还可以和**distinct、and、or、asc、desc、Between, LessThan, GreaterThan、Like**等关键字联合使用，然后会生成具有特定意义的sql。下面是一个例子：



定义的方法最后生成的sql取决于数据库。

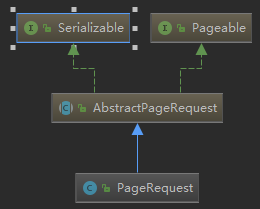
自定义方法各种命名规则参考：

1. <https://blog.csdn.net/LHS19940203/article/details/80576867>
2. <https://blog.csdn.net/youngsend/article/details/51832581>

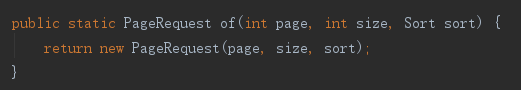
### 关于分页和排序

#### Pageable

Pageable是一个接口，主要用于分页，它可以通过PageRequest来创建。它是Pageable接口的孙子类，结构图如下：



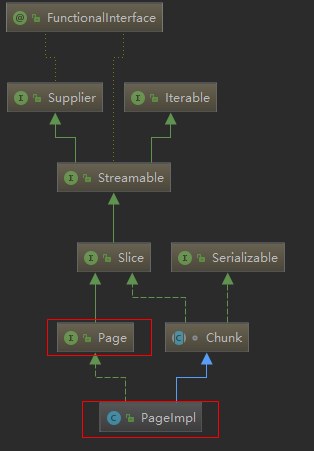
PageableRequest可通过of方法来创建pageable，方法如下：



#### Page

通过page可知道数据总条数，和多少页，它是通过触发一个count查询来计算总量的，所以当结果集的数据量很大时，它的开销是很大的，此时就可以考虑返回一个Slice。因为Slice是当前只知道下一个slice。

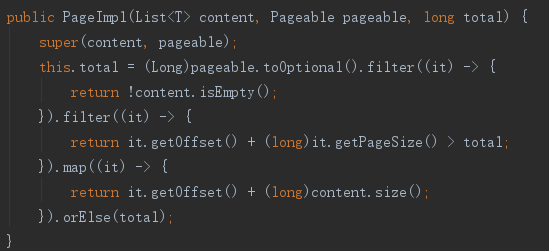
Page接口有PageImpl实现类，结构图如下：



PageImpl的分页实现，例子：



关于构造方法：

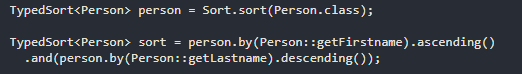


#### Slice

Page接口继承了Slice接口，在遍历较大的结果集时可以使用slice接口

#### Sort

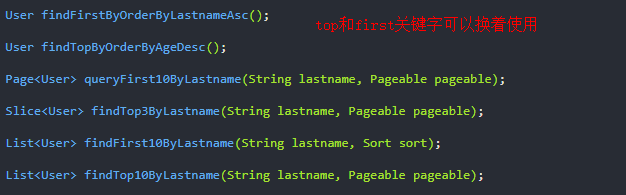




Querydsl的写法（需要数据库支持）



#### 限制查询结果条数

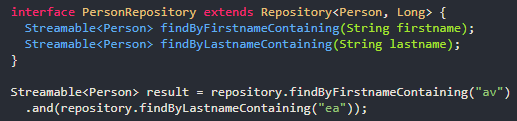


方法定义还支持distinct关键字，结果可以用Optional进行包装

### 采用SpringData中的Streamable来包装查询结果

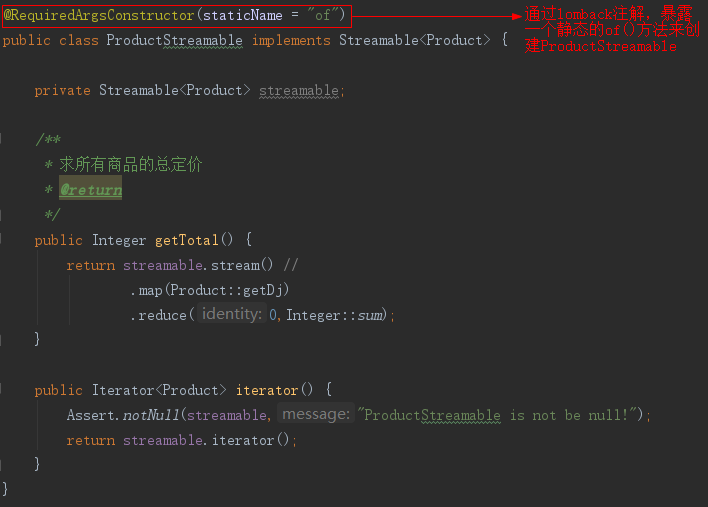
Streamable不需要手动调用close()方法来关闭

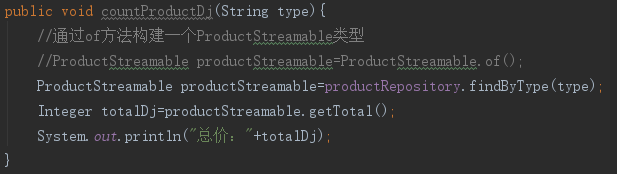
#### 直接用Streamable：



异步查询---@Async

#### 扩展Streamable接口，实现具体操作：





### 对查询方法返回的结果进行null验证控制

1. [@NonNullApi](https://docs.spring.io/spring/docs/5.2.2.RELEASE/javadoc-api/org/springframework/lang/NonNullApi.html)--》对某个包下的类中所有方法返回的结果都进行null验证，此时就不需要再在方法或者属性上加@NonNull注解了



1. @NonNull---》规定返回结果不能为null，否则抛出异常



1. @Nullable--》允许结果可为空

### 异步查询--@Async

[https://docs.spring.io/spring-data/jpa/docs/2.2.3.RELEASE/reference/html/#repositories.query-streaming](https://docs.spring.io/spring-data/jpa/docs/2.2.3.RELEASE/reference/html/" \l "repositories.query-streaming)

### RepositoryFactory

[https://docs.spring.io/spring-data/jpa/docs/2.2.3.RELEASE/reference/html/#repositories.create-instances.java-config](https://docs.spring.io/spring-data/jpa/docs/2.2.3.RELEASE/reference/html/" \l "repositories.create-instances.java-config)

### Custom SpringDataRepository

[https://docs.spring.io/spring-data/jpa/docs/2.2.3.RELEASE/reference/html/#repositories.customize-base-repository](https://docs.spring.io/spring-data/jpa/docs/2.2.3.RELEASE/reference/html/" \l "repositories.customize-base-repository)