# 1. @NamedQuery 查询语句 可以视情况变为List<Person>

## 1.实体类

@Entity *//1*@NamedQuery(name = **"Person.withNameAndAddressNamedQuery"**,  
query = **"select p from Person p where p.name=?1 and address=?2"**)  
**public class** Person {  
 @Id *//2* @GeneratedValue *//3* **private** Long **id**;  
   
 **private** String **name**;  
   
 **private** Integer **age**;  
   
 **private** String **address**;

## 2.jpa接口层

Person withNameAndAddressNamedQuery(String name,String address);

# 2.query利用名称来匹配参数

@Query(**"select p from Person p where p.name= :name and p.address= :address"**)  
Person withNameAndAddressQuery(@Param(**"name"**)String name,@Param(**"address"**)String address);

3.

# 3.排序

*/\*\*  
 \* 测试排序  
 \*/*@RequestMapping(**"/sort"**)  
**public** List<Person> sort(){  
 直接可以使用  
 List<Person> people = **personRepository**.findAll(**new** Sort(Direction.***ASC***,**"age"**));  
 **return** people;  
}

## 2.通过名字查找，并再利用age排序，需要写入JPA接口层，才能引用

List<Person> people = **personRepository**.findByName(String name,**Sort sort**));

# 4,、分页

@RequestMapping(**"/auto"**)  
**public** Page<Person> auto(Person person){  
 Page<Person> pagePeople = **personRepository**.findByAuto(person, **new** PageRequest(0, 10));  
   
 **return** pagePeople;  
   
}

## 2.通过名字查找，

Page<Person> pagePeople = **personRepository**.findByName(String name,**new PageRequest(0,10)**));

## 3.真实项目

*/\*\*  
 \* 分页查询  
 \** ***@return*** *\*/*@Query(**"from StudentInformation as s where ("** +  
 **"s.name like %?1% or ?1 =null) and "** +  
 **"(s.stuNo =?2 or ?2 = null) and "** +  
  
 **"(s.endDate >= ?9 or cast(?9 as date)=null) and "** +  
 **"(s.endDate <= ?10 or cast(?10 as date)=null) and "** +  
 **"(s.tutorName =?11 or ?11 = null) and "** +  
 **"(s.status =?12 or ?12 = null) and "** +  
 **"(s.eduDegree =?13 or ?13 = null)"**)  
Page<StudentInformation> queryGetDepartStusPage(  
 String name,  
 String stuNo,  
 LocalDate endDateMin ,  
 LocalDate endDateMax ,  
 String tutorName,  
 String status,  
 String eduDegree,  
 Pageable pageable);

## 3.2实现层

Pageable pageable = **new** PageRequest(page, 1);  
**return studentInformationRepository**.queryGetDepartStusPage(name, stuNo, gender, ethnic, department, majorCode, eduYear, eduMode, endDateMin, endDateMax, tutorName, status, eduDegree,pageable);

## 3.3 controller层

*/\*\*  
 \* 分页查询 学生学籍信息  
 \*/*@GetMapping(**"/teacher/listDepartStuPage"**)  
**public** ModelAndView listDepartStuPage(String name, String stuNo, String gender, String ethnic, String department, String majorCode, String eduYear, String eduMode, String endDate, String tutorName, String status, String eduDegree ,**int** page) {  
 ModelAndView mav = **new** ModelAndView(**"teacherHTML/tea\_enrollment"**);  
 Page<StudentInformation> studentInformations = **studentInformationService**.queryGetDepartStusPage(name, stuNo, gender, ethnic, department, majorCode, eduYear, eduMode, endDate, tutorName, status, eduDegree,page);  
 **int** pageCount = **studentInformationService**.countStuInfo();  
 mav.addObject(**"studentInformations"**,studentInformations);  
 mav.addObject(**"pageCount"**,pageCount);  
  
 **return** mav;

## 4.补充

//排序对象

Sort sort = new Sort(Sort.Direction.DESC, "timeCreated");

Pageable pageable = new PageRequest(page, pageSize, sort);

## 5.自己之前随便联系的

*使用Pageable作为控制器方法参数*

@Controller

@RequestMapping("/users")

public class UserController {

@Autowired UserRepository repository;

@RequestMapping

public String showUsers(Model model, Pageable pageable) {

model.addAttribute("users", repository.findAll(pageable));

return "users";

}

}

此方法签名将导致Spring MVC尝试使用以下默认配置从请求参数派生一个Pageable实例：

|  |  |
| --- | --- |
| *表1.为Pageable实例评估的请求参数* | |
| page | 您要检索的页面，0已编入索引并默认为0。 |
| size | 您要检索的页面的大小，默认为20。 |
| sort | 应以格式排序的属性property,property(,ASC|DESC)。默认排序方向是上升。sort如果要切换路线，请使用多个参数，例如?sort=firstname&sort=lastname,asc。 |

要自定义此行为可扩展SpringDataWebConfiguration或启用HATEOAS启用的等效项，并覆盖pageableResolver()或sortResolver()方法并导入自定义配置文件，而不是使用@Enable-annotation。

如果您需要从请求中解析多个Pageable或Sort实例（例如，对于多个表），则可以使用Spring的@Qualifier注释来区分出来。然后请求参数必须加上前缀${qualifier}\_。所以对于像这样的方法签名：

public String showUsers(Model model,

@Qualifier("foo") Pageable first,

@Qualifier("bar") Pageable second) { … }

你有填充foo\_page和bar\_page等。

该Pageable方法的默认值相当于一个，new PageRequest(0, 20)但可以使用@PageableDefaults参数上的Pageable注释进行自定义。

超媒体支持页面

Spring HATEOAS提供了一个表示模型类PagedResources，它允许Page使用必要的Page元数据丰富实例的内容，以及链接，让客户端轻松浏览页面。PagedResources通过Spring HATEOAS ResourceAssembler接口的实现来完成页面的转换PagedResourcesAssembler。

### 5.2代码

@RequestMapping(**"/pageable"**)  
**public** String showUsers(Model model) {  
  
Pageable pageable = **new** Pageable() {  
 @Override  
 **public int** getPageNumber() {  
 **return** 0;  
 }  
 @Override  
 **public int** getPageSize() {  
 **return** 4;  
 }  
 @Override  
 **public int** getOffset() {  
 **return** 0;  
 }  
 @Override  
 **public** Sort getSort() {  
 **return null**;  
 }  
 @Override  
 **public** Pageable next() {  
 **return null**;  
 }@Override  
 **public** Pageable previousOrFirst() {  
 **return null**;  
 }@Override  
 **public** Pageable first() {  
 **return null**;  
 }@Override  
 **public boolean** hasPrevious() {  
 **return false**;  
 }  
};  
 model.addAttribute(**"largeOrders"**, **testPageableRepository**.findAll(pageable));  
 **return "find"**;  
}

# 5.模糊查询 ，日期的格式问题

## 1.localdate

**"(s.endDate >= ?9 or cast(?9 as date)=null) and "** +  
**"(s.endDate <= ?10 or cast(?10 as date)=null) and "** +

## 2.LocalDateTime

timestamp

## 3.模糊查询

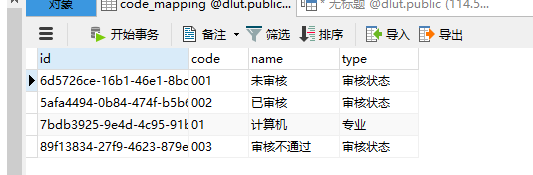
@Query(value = **"from JobRequistion j where j.recruiter=?1 and ( audit.status=?2 or ?2=null) and (j.title like (%?3%) or ?3=null) and ( jobCategory=?4 or ?4=null) and ( jobArea in ?5 or ?5=null) and j.status=null and (j.department.name in ?6 or ?6=null)"**)  
**public** Page<JobRequistion> findListByConditionForAuditStatusNoDate(RecruiterInfo recruiter,

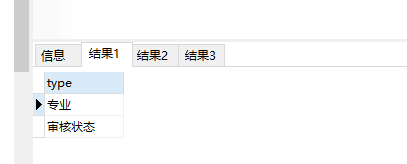
# 6.group by

SELECT DISTINCT type from code\_mapping ;

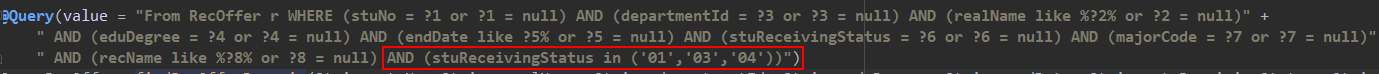
SELECT DISTINCT type from code\_mapping ORDER BY type DESC;

SELECT type from code\_mapping GROUP BY type ORDER BY type DESC;





# 7.in



# 5.判断某个字段是NULL is NULL，下面是不空

**WHERE r.end\_date LIKE ?1% AND graduate\_destination.ID = r.ID AND graduate\_destination.rec\_name IS NOT NULL "**,nativeQuery = **true**)  
**int** updateUderemployed(String graduateDate);

# 6. 3.直接根据字段导入另一张表

INSERT INTO "public"."tb\_student"(id,created\_on,email,mobile,pwd,real\_name,source,updated\_on) SELECT tn\_account.uuid\_accid,create\_date,email

,mobile,pwd,real\_name,source,modify\_date from tn\_account;