<https://stackoverflow.com/questions/60281520/persistence-createentitymanagerfactorysomething-returns-null>

ProductSpecChar pscChild = new ProductSpecChar();  
BeanUtils.*copyProperties*(psc,pscChild);

# Các method sql hay

<https://quantrimang.com/nhung-diem-moi-trong-sql-server-2017-148401>

# Difference between @Autowired and @Resource in Spring

The main difference is that @Autowired wires per type and @Resource wires per bean name. But @Autowired in combination with @Qualifier also autowires by name.

\*\* Intellij

Trong intellij có thể tạo file setting.xml của m2 giúp nó tải maven từ https ko phải http như trước

\*\* Sql:

Pagination: Thông thường sử dụng offset để bỏ qua số phần tử đầu vd:

Bạn kế họp với limit để Select số page mông muốn

|  |
| --- |
| SELECT \*  FROM Student  LIMIT 5 OFFSET 2  ORDER BY ROLLNO; |

Học Sql ở đây để trở thành pro

<https://www.geeksforgeeks.org/sql-concatenation-operator/>

\*\* Sercurity phân quyền cơ bản

@PreAuthorize("hasAuthority('ADMIN')")

<https://helpex.vn/article/bao-mat-ung-dung-spring-boot-voi-spring-security-va-preauthorize-608eb8bd2744c49a00465ccc>

\*\* Audit Sercurity

<https://stackoverflow.com/questions/53658463/createdby-and-lastmodifiedby-set-actual-entity-instead-of-id>

\*\* Docker

<https://xuanthulab.net/gioi-thieu-ve-docker-lam-quen-voi-docker-tao-container.html>

Các image có sẵn của docker

# Các link học java wed hay

## Spring core

<https://techmaster.vn/posts/36087/spring-core-phan-5-spring-aop-la-gi-code-vi-du-voi-spring-aop>

## <https://stackjava.com/>

<https://automationstepbystep.com/>

**https://www.tutorialspoint.com/questions/category/Java**

tip key Intellij

https://www.jetbrains.com/help/idea/mastering-keyboard-shortcuts.html

## 1.2 Sự khác nhau giữa hibernate-core và jpa

https://docs.spring.io/spring-data/commons/docs/current/api/org/springframework/data/

### Sự khác nhau giữa hibernate-core và jpa ( entityManager ) (hibernate là implement của interface jpa)

<https://stackjava.com/hibernate/so-sanh-su-khac-nhau-hibernate-session-voi-jpa-entitymanager.html>

### Spring Sercurity

<https://kipalog.com/posts/Huong-dan-lap-trinh-Spring-Security>

Chỉ cần nhớ SecurityContextHolder (Authen), SimpleGrantedAuthority (Author)

\*\* File UserDetailImpl không thể @Autowired nhớ cái này(dùng final cnos sẽ hiện ra lỗi luôn)

Authentication authentication = SecurityContextHolder.*getContext*().getAuthentication();

User user = (User) authentication.getPrincipal();

Cái authentication sẽ lấy dc UserDetail nhờ vào SecurityContextHolder

**public** Collection<? **extends** GrantedAuthority> getAuthorities() {// trả về danh sách role của user

Set<Role> roles = user.getRoles();

List<SimpleGrantedAuthority> authorities = **new** ArrayList<SimpleGrantedAuthority>();

**for** (Role role : roles) {

authorities.add(**new** SimpleGrantedAuthority(role.getName()));

}

**return** authorities;

}

----------------------

@Bean

**public** DaoAuthenticationProvider authenticationProvider() {

DaoAuthenticationProvider authProvider = **new** DaoAuthenticationProvider();

System.***out***.println("create authProvider ----~~");

authProvider.setUserDetailsService(userDetailsService);

authProvider.setPasswordEncoder(passwordEncoder());

**return** authProvider;

}

Lấy ra các vai trò của user cài này override rồi cứ code vào đây là đc

|  |
| --- |
| Không cần UserDetail như cái trên AutenticationProvider  CustomAuthenticationProvider implements AuthenticationProvider  Sau đó  @Override protected void configure(AuthenticationManagerBuilder auth) throws Exception {  auth.authenticationProvider(authProvider); } |

### Spring Aop để quản lý Logging

<https://viblo.asia/p/su-dung-aop-trong-spring-boot-va-aspectj-vyDZOkbaZwj>

= tương tự bit or &= tương tự bit and

String… arg = String []arg

## 1.3 System.getProperty()

<https://www.tutorialspoint.com/system-getproperty-in-java#:~:text=The%20getProperty(String%20key)%20method,System%20Class.&text=where%20key%20is%20the%20name%20of%20the%20System%20property>.

## 1.4 Đừng validate bằng tay

<https://viblo.asia/p/dung-validation-bang-tay-trong-spring-boot-phan-1-3P0lPGyoZox>

## 1.5 Lưu ý khi ghi file excel

* chỉ có thể ghi header footer sau khi ghi data
* chỉ có thể autosize khi đã ghi xong mọi thứ ghi excel rất lỗi cần chú ý
* Kết luận ghi file (row -> cell) cần đưa vào vòng for (Nếu bỏ cell ++ thì nó lỗi cell đầu = 1 )
* cell đầu = 1, row đầu = 0)

## 1.6 Phương thức equals() và hashCode() trong java

<https://viettuts.vn/java/phuong-thuc-equals-va-hashcode-trong-java>

## 1.7 huong-dan-su-dung-jackson-json-annotations

<https://gpcoder.com/3436-huong-dan-su-dung-jackson-json-annotations/#:~:text=%40JsonIgnoreProperties%2F%20%40JsonIgnore,-%40JsonIgnoreProperties%3A%20Ch%C3%BA%20th%C3%ADch&text=Ngh%C4%A9a%20l%C3%A0%20ch%C3%BAng%20s%E1%BA%BD%20kh%C3%B4ng,and%20Deserialization%20d%E1%BB%AF%20li%E1%BB%87u%20JSON>.

## 1.8 Jaxb

<https://openplanning.net/10143/java-jaxb>

## 1.9 SpringApplicationType

<https://zetcode.com/springboot/webapplicationtype/>

## 2.10 Spring.factories (auto config)

<https://medium.com/codeshake/spring-boot-auto-configuration-mystery-revealed-d734516dddfb>

https://docs.spring.io/spring-boot/docs/2.0.0.M3/reference/html/boot-features-developing-auto-configuration.html

## 2.11 Load file Xml,Yaml,Properties to Properties in java

https://tvd12.com/properties-vs-yaml/

<https://howtodoinjava.com/java/xml/convert-xml-to-properties/>

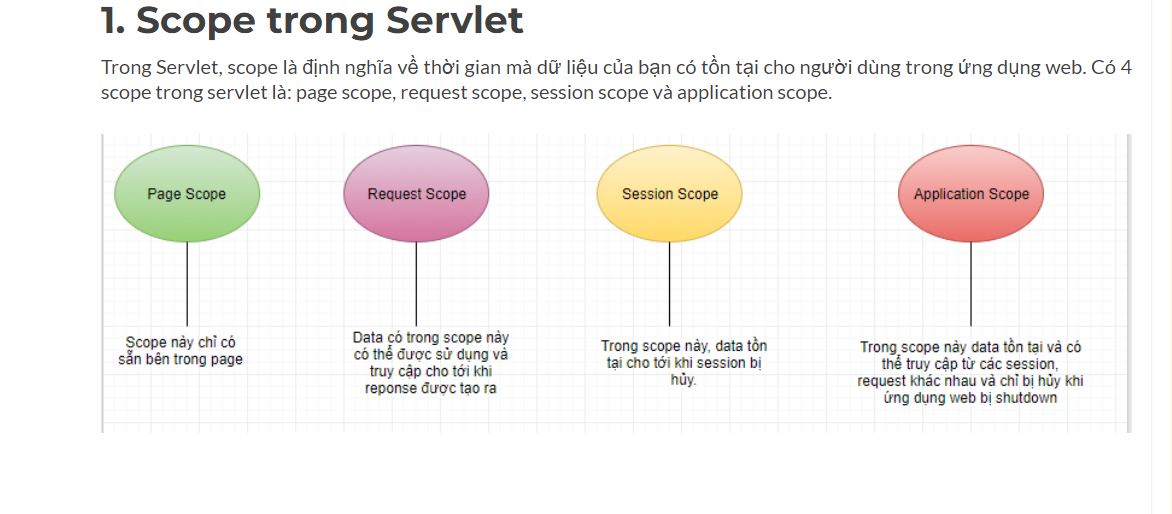
## 2.12 Autoconfig Spring boot ( spring.factories thay file application.properties)

<https://docs.spring.io/spring-boot/docs/2.0.0.M3/reference/html/boot-features-developing-auto-configuration.html>

https://www.baeldung.com/spring-boot-custom-auto-configuration

## 2.13. Bean Scope

Các web Scope



<https://stackjava.com/jsp-servlet/cac-scope-trong-servlet-application-request-session-va-page-scope.html>

**hiểu đơn giản là bạn chỉ có thể dùng RequestScope trong 1 request (1 rescontroller thế thôi còn khai báo requestScope ở đâu thì tùy)**

Các Scope chung chung

<https://shareprogramming.net/hieu-ro-spring-bean-scope/>

## 2.14. Criterial Hibernate

<https://www.baeldung.com/spring-data-criteria-queries>

https://viblo.asia/p/tai-sao-nen-su-dung-jpa-criteria-V3m5WB8xlO7

## 2.15 Gitlab CI/CD

<https://www.youtube.com/watch?v=yXJ7vUCPRTA>



## 2.16 Docker Beginer

<https://www.dataset.com/blog/create-docker-image/>

<https://docs.docker.com/develop/develop-images/baseimages/>

<https://www.stereolabs.com/docs/docker/creating-your-image/>

## 2.17 ModelMapper

<https://viblo.asia/p/su-dung-modelmapper-trong-spring-boot-63vKj1Vd52R>

## 2.18 Kiểu Dữ liệu

Int,float, 4 byte

Char,Short 2 byte

Boolean 1bit

## 2.19 link request pageable

<http://localhost:8081/api/v1/equip_type/search?page=0&size=20&sort=equipGroupId,asc&sort=groupTypeId,desc>

nó sẽ tìm các row thỏa mãn các đk trên bỏ các row ko thỏa mãn đi kể cả không đủ size

## 2.20 LastModify ,@ParamObject

@LastModifiedBy

<https://stackoverflow.com/questions/53658463/createdby-and-lastmodifiedby-set-actual-entity-instead-of-id>



**- @ParamObject covert @RequestParam to Object**

**Thư viện đây: cái thư viện này supper cả swagger, sercurity và rất nhiều thứ khác**

<dependency>  
 <groupId>org.springdoc</groupId>  
 <artifactId>springdoc-openapi-ui</artifactId>  
 <version>1.6.8</version>  
</dependency>

## 2.21 Collection và Collections trong java

<https://viettuts.vn/java-collection>

## 2.22 Tính đa hình chỉ áp dụng với method còn variable thì ko

<https://stackoverflow.com/questions/12086298/why-is-an-instance-variable-of-the-superclass-not-overridden-by-a-subclass#:~:text=Because%20instance%20variables%20CANNOT%20be,can%20even%20be%20used%20>...

## 2.23 Hikari Connection Pool trong Spring Boot như thế nào?

<https://viblo.asia/p/cau-hinh-hikari-connection-pool-trong-spring-boot-nhu-the-nao-eW65Gey6ZDO>

* Hiện tại spring mới thì nó có cái này rồi nên đọc để lấy các thông tin cấu hình autoconfig trong file application.properties thôi

## 2.24 Spring Oauth2

<https://console.cloud.google.com/apis/credentials/oauthclient/731079018552-0qjvepqgu19nvanqhud535ns251d1duh.apps.googleusercontent.com?project=zippy-sublime-295502>

## 2.25 Config Swagger

Có 2 cách :

* springfox-swagger-ui (Cái này chỉ dùng cho swagger-ui mà thôi)
* springdoc-openapi-ui (cái này có nhiều thư viện khác nữa)
* tìm hiểu xem

## 2.26 bản chất Pageble

- Bản chất nó sẽ Select count trước khi vào tìm page ko dùng offset

- Có thể custome cái bến countQuery cho cái pageable như thế này

|  |
| --- |
| @Query(value = "SELECT e FROM Employee e LEFT JOIN FETCH e.addresses a " +  "WHERE e.company.id = :companyId " +  "AND e.deleted = false " +  "AND e.primaryAddress.deleted = false " +  "ORDER BY e.id, a.id",  countQuery="SELECT count(e) FROM Employee e WHERE e.companyId = :companyId AND e.deleted = false AND e.primaryAddress.deleted = false"  )  Page<Employee> findAllEmployeesWithAddressesForCompany(@Param("companyId") long companyId, Pageable pageable); |

## \*2.27 Cách Select tìm kiếm join nhiều bảng sử dụng bảng tạm @Entity trong jpa

- Bản chất jpa set mapper sang các pojo được đáng dấu là 1 bảng của jpa ( @Entity) hãy @Entity vào Pojo bảng bạn muốn trả ra

B1 Đánh dấu @Entity vào pojo @Column và các cột tưng ứng muốn lấy ra

* Cái này cứ @Column các tên cột mà mình muốn lấy ra nó chỉ ảnh xạ tên kq của câu Lệnh Select

Vd:

|  |
| --- |
| @Entity @Getter @Setter public class UserAllInfoWeb implements Serializable {   private static final long *serialVersionUID* = 1L;   @Id  @Column(name = "RN")  private int rowNum;   @Column(name = "USER\_ID")  private int userId;   @Column(name = "USERNAME")  private String username;   @Column(name = "PHONE")  private String phone;  } |

**Ở đây cột PHONE kia có thể là cột mà bảng join trả ra**

B2 Chúng ta thực hiện Tạo @Repository hoặc dùng @EntityManage

Vd

* Với @Repository chỉ cần extends JpaRepository<**Entity**, Long>

Chú ý **Entity** phải là cái pojo mà ta đánh dấu bảng tạm @Entity

|  |
| --- |
| * @Repository public interface UserAllInfoRepository extends JpaRepository<UserAllInfoWeb, Long> {   @Query(value = "select row\_number() over() rn,u.user\_id, u.username, u.phone, u.fullname, u.gender, u.role\_id, u.email, u.groups\_code, u.created\_date,\n" +  " ta.register\_no as register\_no, ta.device\_code as device\_code, ta.sim as sim,\n" +  " substr(u.groups\_code, 1, 6) as parent\_groups\_code, \n" +  " g1.name as direct\_group, g2.name as parent\_group\n" +  " from users u\n" +  " left join groups g1 on u.groups\_code = g1.code and g1.is\_active = 1\n" +  " left join groups g2 on substr(u.groups\_code, 1, 6) = g2.code and g2.is\_active = 1\n"  countQuery = *sqlCount* ,nativeQuery = true) Page<UserAllInfoWeb> getAllInfo(String groupCode, String userNameOrFullName, String groupNameOrParentGroupName,  String deviceCode, String registerNo, Pageable pageable); |

* Với @EntityManage

Chú ý đặt ánh xạ class Entity là pojo chúng ta đặt bảng tạm @**Entity**

|  |
| --- |
| * q = em.createNativeQuery(stringBuilder.toString(),**Entity**.class); |

1 Chú ý quan trong với cái này : Khi truyền pageable vào nó sẽ lấy các bảng join nên nhiều khi sẽ bị lôi

Đơn giản chúng ta sử dụng **as**

Vd

|  |
| --- |
| ta.register\_no as register\_no, ta.device\_code as device\_code |

Tóm lại sử dụng as nó khác với dấu cách

**ta.register\_no as register\_no != ta.register\_no register\_no**

2 cái này khác hoàn toàn nhau ấy

# Mấy cái liên quan đến wed

## Các Mã lỗi Http:

### 1xx: Informational – yêu cầu (request) đã được nhận, tiếp tục tiến trình xử lí

* 100: Continue – Tiếp tục. Yêu cầu máy khách (client) gửi tiếp phần còn lại. Phần đầu của yêu cầu đã được máy chủ nhận và chờ gửi lại mã này để máy khách (client) gửi tiếp phần còn lại.
* 101: Switching Protocols – Thay đổi giao thức. Thay đổi giao thức (protocol) cho phù hợp

### 2xx: Success – thành công

* 200: OK - Truy vấn đến máy chủ thành công. Thông tin trả lời của máy chủ sẽ phụ thuộc vào phương thức yêu cầu (GET – HEAD – POST – TRACE)
* 201: Created – Đã khởi tạo. Yêu cầu đã được chấp nhận và kết quả sẽ dẫn tới tài nguyên mới được tạo ra.
* 202: Accepted – Chấp nhận. Yêu cầu đã được máy chủ chấp nhận để thi hành (nhưng chưa xong)
* 203 : Non-Authoritative Information – Thông tin chưa được xác định. Thông thường thông tin được gởi thẳng từ máy chủ đến máy con. Nếu không phải , mã này sẽ được gửi để báo đây không phải là data gửi từ máy chủ xác định.
* 204 : No content – không có nội dung. Yêu cầu đã được máy chủ nhận nhưng không có dữ liệu nào được gửi ngược lại, khi trình duyệt nhận được thông báo này sẽ không thay đổi nội dung trang web đang hiển thị
* 205: Reset Content– xoá nội dung.Trình duyệt được thông báo phải xóa nội dung của form đã được dùng để người sử dụng có thể nhập vào 206: Partial Content Nội dung bộ phận

### 3xx: Redirection – chuyển hướng

* 300: Multiple choice – nhiều tùy chọn     Thông tin được gửi đi có nhiều tùy chọn cho phép bên trung gian có thể tùy ứng để chuyển tiếp nó đến bên yêu cầu.
* 301: Move Permanently – Chuyển cố định     Yêu cầu đã được chuyển đến 1 URL mới, và điều này là cố định do đó nó cần được ghi nhận

### 4xx: Client Error – yêu cầu sai cú pháp hoặc không thỏa đáng

* 401: Unauthorized - Không có quyền
* 402: Payment Require – Yêu cầu trả tiền
* 403: Forbidden – Bị cấm truy nhập
* 404: Not found– không tìm thấy.

### 5xx: Server Error – máy chủ gặp lỗi

## EntityManager

Đầu tiên thì entityManager là của Jpa, Hibernate kế thừa từ jpa nên Hibernate nó sẽ có nhiều thừ hơn nhưng EntityManager lại đc ưa dùng hơn.

Trong 1 Session persistence context, các entity có 4 trạng thái:

transient: đối tượng chưa bao giờ bị quản lí bởi session và nó không tương ứng với bản ghi nào trong database; thông thường đây là 1 đối tượng mới được tạo để save vào database

persistent: đối tượng bị quản lý bởi session và là unique (trong 1 session không thể tồn tại 2 object có cùng id), sau khi flush bởi session sẽ tồn tại 1 bản ghi tương ứng đối tượng này trong database.

detached: đối tượng này đã từng bị quản lý bởi session nhưng hiện tại thì không. (bị evict(), clear(), close())

removed:cũng giống như detached nhưng bản ghi tương ứng với đối tượng này trước đó đã bị xóa khỏi database. (bị remove())

## RequestContextHolder

Cái này nó dữ request của hệ thống nếu có ( tóm lại nó là 1 cái cờ nhớ request)

### ServletRequestAttributes attributes = (ServletRequestAttributes) RequestContextHolder.*getRequestAttributes*();

### HttpServletRequest request = attributes.getRequest();

Lấy đc HttpServletRequest từ cái này luôn

## File Trong java

Một số phương thức lớp File

* **exists()** kiểm tra xem file có tồn tại hay không
* **getName()** lấy tên file (input-file.txt)
* **getParent()** lấy đường dẫn thư mục của file
* **getPath()** đường dẫn đầy đủ
* **isDirectory()** kiểm tra xem là thư mục hay không
* **isFile()** kiểm tra xem là file hay không
* **length()** cỡ file (byte)
* **createNewFile()** tạo ra file mới
* **delete()** xóa file
* **list()** lấy tên file, thư mục chứa trong đường dẫn
* **mkdir()** tạo thư mục
* **renameTo(File dest)** đổi tên file

Có nhiều cách đọc ghi file nhưng t dùng 1 đã sửa dụng trong trường cho dễ

Ghi file

|  |
| --- |
| public void KH\_Add(KhachHang data) throws IOException {  File file = new File(path + "KH.TXT");  if (!file.exists()) {  file.createNewFile();  }  FileOutputStream outputStream = new FileOutputStream(file, true);  OutputStreamWriter outputStreamWriter = new OutputStreamWriter(outputStream);  if (data.getMaKH() == -1) {  System.err.println("Chua co maKH 50--> KH\_File\_TXT");  return;  }  outputStreamWriter.write(data.getMaKH() + "");  outputStreamWriter.write("\n");  outputStreamWriter.write(data.getHoTen());  outputStreamWriter.write("\n");  outputStreamWriter.write(data.getDiaChi());  outputStreamWriter.write("\n");  outputStreamWriter.write(data.getLoaiKH());  outputStreamWriter.write("\n");  outputStreamWriter.flush();  }  **flush**() **trong khi đọc ghi file dữ liệu sẽ đc lưu vào bộ đệm để nó ghi luôn thì dùng flush()** |

* Note những thứ cần để trao dồi kỹ năng tìm kiếm

+ nói chung tìm kiếm mọi thứ có thể tìm kiếm , thửu mọi thứu có thể thử nhưng phải thông minh

+ tìm kiếm class ko đc thì tìm từ package

+ tìm kiếm tất cả các file trong sql : chỉ trong sechema m đang đứng (oracle có package nhé nó gộp function)

+ complie báo not found thì là 404 đấy ( tức là api ko tồn tại)

+ nhìn lỗi ở complie trước khi debug(đề bấc)