@Autowire: nhúng từ Repository sang Service

@Service: đánh dấu dây là Service

@Entity:

@Repository: như DAO

@ManyToMany(fetch = FetchType.LAZY)

* LAZY đc ưu tiên dùng hơn, gọi ra thì mới load
* Còn cái EARGE gì đó thì nó load hết all roles khi mk gọi user ra, đấy là khi roles này còn ít dữ liệu á, nó mà nhiều thì thôi 😊 nghỉ
* Để mà dung đc thì cần add them vào JpaConfig đoạn properties.setProperty("hibernate.enable\_lazy\_load\_no\_trans", "true");

1. **Tạo Entity (tạo ra khi chạy sẽ tự động tạo bảng có tên là user trong mysql)**

B1: Tạo 1 package com.laptrinhjavaweb.entity

B2: Tạo 1 Class mới, UserEntity

* Trênđầu class ghi @Entity, @Table(name = "user")
* Tạo Id:

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

* Tạo các trường khác:

@Column(name = "username")

private String username;

(name = “username” có nghĩa sẽ tạo trường trong mysql có tên là username)

Còn có cả @ManyToMany các thứ vui lắm

Ngoài ra, JPA còn hỗ trợ @CreatedDate, @CretedBy, … nhưng cần phải tạo them file JpaAuditingConfig.java trong pagekage com.laptrinhjavaweb.config, nội dung file thì vào mà cop 😊

B3: VÀo file persistence.xml (trong resource) thêm: <class>com.laptrinhjavaweb.entity.UserEnitty</class>

1. **Tạo Reponsitory (có vai trò như là DAO)**

public interface UserRepository extends JpaRepository<UserEnitty, Long>{

// viết các hàm cần dung vào đây

UserEnitty findOneByUserNameAndStatus(String name, int status);

}

1. **JPASecurity**

* Thêm giao diện login vào folder decorators và template
* Thêm folder database vào project, them file insert.sql (tùy lúc thôi)
* Vào decorator.xml them 1 sitemet vào

<decorator name="login" page="login.jsp">

<pattern>/dang-nhap</pattern>

</decorator>

* Vào context, tạo 1 file security.xml, nội dung thì lại cop
* Lưu ý:

<form-login login-page="/dang-nhap" **-> tên controller**

username-parameter="j\_username" -> **thay vào chỗ name của username trong jsp**

password-parameter="j\_password" " **-> thay vào chỗ name của pasword trong jsp**

login-processing-url="/j\_spring\_security\_check" **-> thay vào chỗ action trong jsp**

authentication-success-handler-ref="customSuccessHandler" **-> tạo ở hàng cuối đấy**

authentication-failure-url="/dang-nhap?incorrectAccount"/> **-> thất bại**

<session-management invalid-session-url="/dang-nhap?sessionTimeout"/> **-> timeout**

<logout delete-cookies="JSESSIONID"/>

* Vào web.xml, them đoạn Declare Spring Security Filter (đoạn này phải ở trên đọan sitemet, vì mk phải login trc r mới apply giao diện),

trong context-param, them đoạn config về security.xml

* Tạo file CustomUserDetailsService.java trong com.laptrinhjavaweb.service.iml (trong đó có đoạn MyUser, thì cái ý là mk viết thêm class MyUser trong dto, dung để override cái hàm của nó, vì mk muốn nó lưu them cả fullname cho mk nữa)
* Tạo package: com.laptrinhjavaweb.securiry
* Tạo file CustomSuccessHandler trong nó
* Tajo 1 package com.laptrinhjavaweb.util, tạo file SecurityUtils.java
* Sửa file tablib.jsp, them cái dòng securiry gì đó vào
* Thêm <intercept-url pattern="/quan-tri/trang-chu" access="hasAnyRole('ADMIN')" /> vào security.xml. Nghĩa là để vào đc dường link kia thì cần có role là ADMIN, ko thì cút

1. Restful API

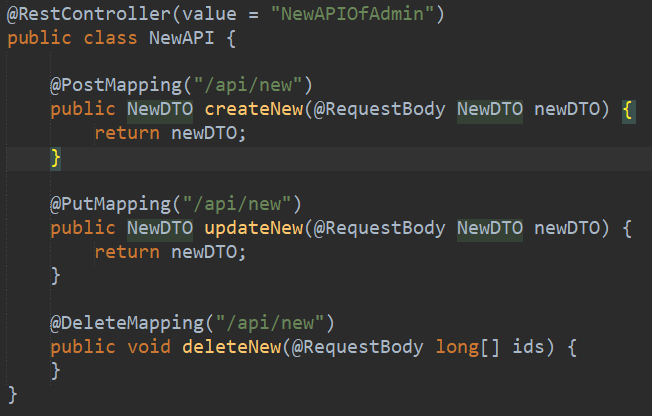
* Tạo package com.laptrinhjavaweb.api.admin
* Tạo file NewAPI.java
* @RestController(value = "NewAPIOfAdmin")
* Thêm định nghĩa com.laptrinhjavaweb.api vào trong file dispartcher-servlet.xml
* Thêm depencedency của json vào pom.xml
* 4 method chính:

+ POST: thêm mới dl

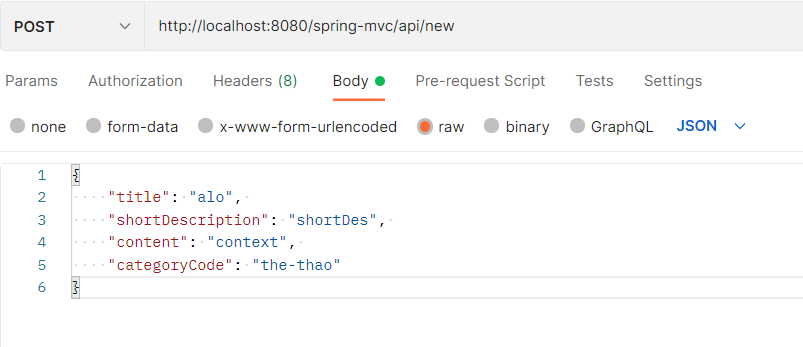
+ PUT: update

+ DELETE: delete

+ GET: lấy dl



* Chạy Postman :



1. **Thêm, sửa, xóa**

* **Converter:**
* Thêm @Component vào đầu class, để tý nữa mk còn nhúng vào Service, vì muốn nhúng thì cần có @repository hoặc @Component
* Convert thủ công thôi, Có cái MapStruct, từ từ r xem
* Lúc convert từ entity qua DTO thì cần setID vì đó là lúc mk update, còn covert ngược lại thì ko cần vì khi đó mk đang add, mà add thì id tự tăng database
* **<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>**
* Thẻ <form:form>: dùng cho trường hợp mk có 1 thẻ select vs nhiều option. Khi mk update bài viết thì khi update xong thẻ select sẽ chọn option mk vừa update. Dĩ nhiên, khi đó trong thẻ <form:form> mk phải khai báo thêm modelAttribute="model" vs model là tên mk đẩy từ controller qua.
* Mk cũng ko dung thẻ select của html nữa, mk dung <form:select> và <form:option>, trong <form:select> cần có path và id, cứ cho categoryCode vào
* Như bthg là mk phải dung toán tử 3 ngôi

@RequestMapping(value = "/quan-tri/bai-viet/chinh-sua", method = RequestMethod.GET)

public ModelAndView editNew(@RequestParam(value = "id", required = false) Long id)

* Cái @RequestParam nó lấy dữ liệu từ param id truyền về, nhưng nó có thể có hoặc ko, nếu ko thì vẫn ok, ko bị null exception

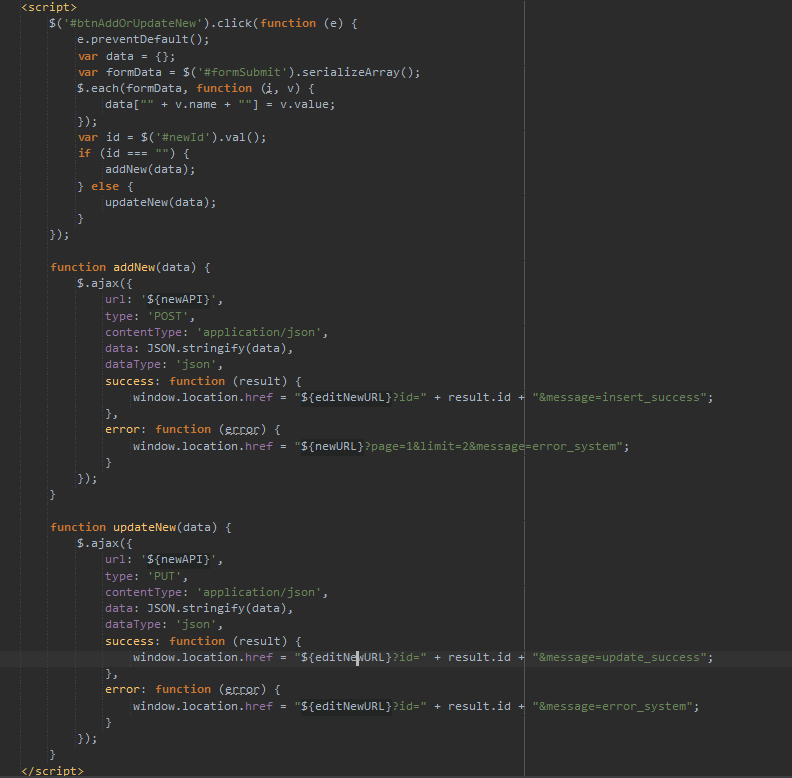
1. Thêm, sửa:

<c:url var="newURL" value="/quan-tri/bai-viet/danh-sach"/>

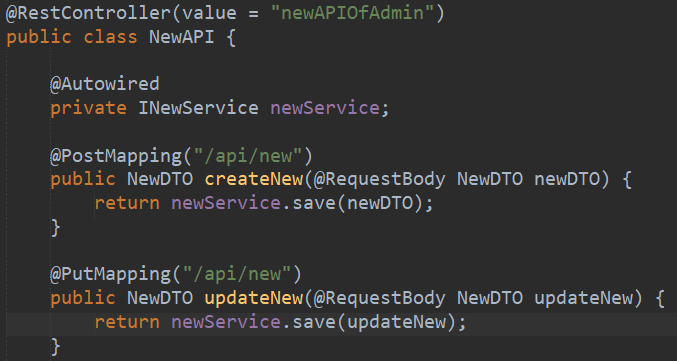
<c:url var="editNewURL" value="/quan-tri/bai-viet/chinh-sua"/>

<c:url var="newAPI" value="/api/new"/>

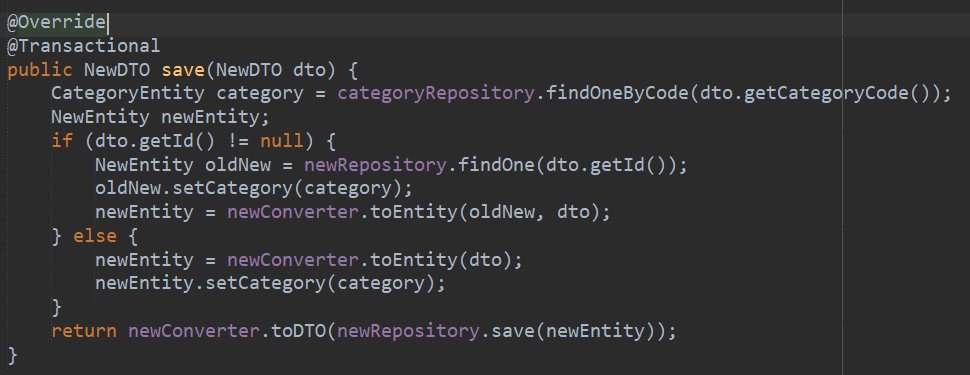
* TRước các hàm thêm sửa xóa (trong Service ) cần them @Transactional để mở transaction
* Sử dụng ajax bên jsp, gọi url là con API.



* Bên package util tạo class MessageUtil
* Bên API thì đơn giản



* Bên NewService (implement từ INewService)



1. Xóa:

* Thêm vào INewService
* Bên NewSErvice thì imp thêm vào
* Gọi hàm newRepository.delete là xong
* Bên API thì gọi lại Service -> ok
* Sửa bên giao diện nữa:
* Thêm css, js của sweetAleart
* Thêm 