

MỘT SỐ BÀI TOÁN TRÊN SPRING BOOT

Nguyễn Huy Cường - nh.cuong@hutech.edu.vn

06/2024

Nội dung

1- Tìm kiếm

2- Phân trang

3- Giỏ hàng - Đặt hàng – Thanh Toán

4- JWT

1- Tìm kiếm

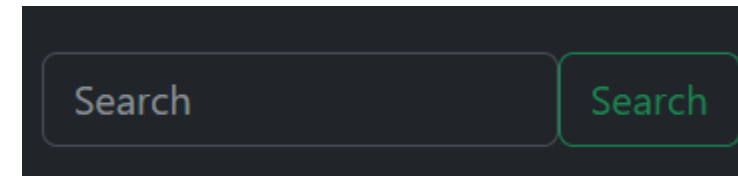
- Tìm kiếm giúp nâng cao trải nghiệm người dùng
- Ý tưởng thực hiện :

1 – Tạo giao diện **View** cho **Search**

```
<form class="d-flex" th:action="@{/products/search}" method="get">
  <input class="form-control me-2" type="search" placeholder="Search" aria-label="Search" name="key">
  <button class="btn btn-outline-success" type="submit">Search</button>
</form>
```

2 – Tạo **Action** thực hiện **Search** ở controller

```
@GetMapping("/search")
public String index(Model model, String key)
{
    List<Product> listProductSearchs = productService.GetSearchProducts(key);
    model.addAttribute("listproduct", listProductSearchs);
    return "product/products";
}
```



Phương thức tìm kiếm theo key

```

@Service
public class ProductService {
    5 usages

    @Autowired
    private ProductRepository productRepository;
    2 usages

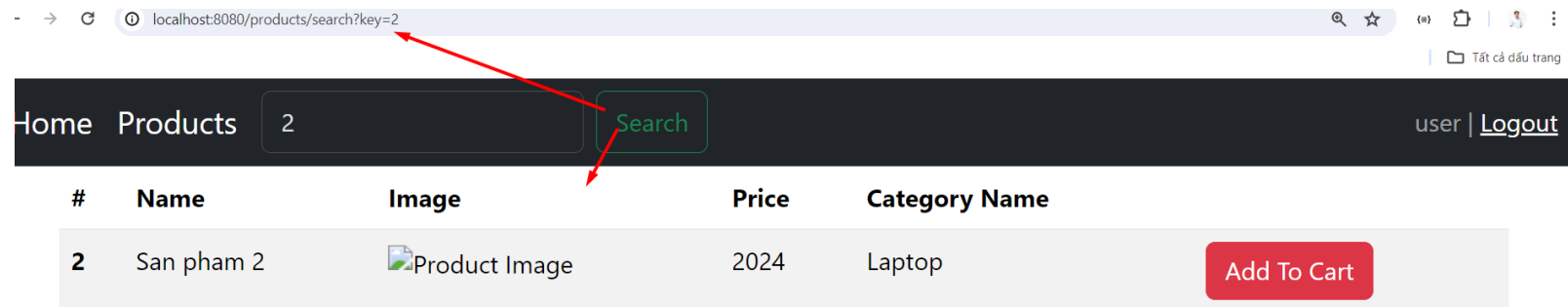
    public List<Product> getAll() {
        List<Product> listProduct = productRepository.findAll();
        return listProduct;
    }
    1 usage

    public List<Product> GetSearchProducts(String keySearch) {
        return productRepository.searchProductsByKeySearch(keySearch);
    }
}
    
```

←

```

@Repository
public interface ProductRepository extends JpaRepository<Product, Integer> {
    @Query("SELECT p FROM Product p WHERE p.name LIKE %:keySearch%")
    List<Product> searchProductsByKeySearch(@Param("keySearch") String keyword);
}
    
```



2- Phân trang

- Phân trang: tăng trải nghiệm người dùng trong việc tìm hiểu thông tin. Việc chia nhỏ thông tin bằng việc phân trang (Pagination) cũng giúp hạn chế quá tải thông tin trên cùng một site.
- Ý tưởng thực hiện:

Sử dụng thư viện: `import org.springframework.data.domain.`**Page**;

(1) Ở **Controller**

- Action: Thêm tham số page (trang), size (kích thước 1 trang)
- Trả về danh sách trong Page<> được thực hiện ở service
- Truyền dữ liệu tổng số trang (totalPages) tìm được tới giao diện

(2) **Service**

- Hàm lấy danh sách theo page và size và trả về Page<ObjectSearch>
- Trả về đối tượng PageRequest từ page, size
- Khi truy vấn truyền vào findAll theo page, size

(3) **Thymeleaf**

- Show Navigation cho footer cho search



2- Phân trang

(1) Ở Controller

```
@GetMapping("")
public String index(Model model,
    @RequestParam(defaultValue = "0") int page,
    @RequestParam(defaultValue = "2") int size)
{
    Page<Product> products = productService.GetAll(page, size);
    model.addAttribute("listproduct", products);
    model.addAttribute("totalPages", products.getTotalPages());
    return "products/index";
}
```

(2) Service

```
public Page<Product> GetAll(int pageNo, int pageSize) {
    PageRequest pageRequest = PageRequest.of(pageNo, pageSize);
    return productRepository.findAll(pageRequest);
}
```

(3) Thymeleaf



```
<nav aria-label="Page navigation example" th:if="${totalPages > 1}">
    <ul class="pagination">
        <li class="page-item" th:class="${listproduct.hasPrevious() ? '': 'disabled'}">
            <a class="page-link"
                th:href="@{/products?page=__${listproduct.number-1}__}" aria-label="Previous">
                <span aria-hidden="true">&laquo;</span>
            </a>
        </li>
        <li class="page-item" th:each="page : ${#numbers.sequence(0, totalPages - 1)}"
            th:class="${listproduct.number == page ? 'active' : ''}">
            <a class="page-link" th:href="@{/products?page=__${page}__}"
                th:text="${page + 1}"></a>
        </li>
        <li class="page-item" th:class="${listproduct.hasNext() ? '': 'disabled'}">
            <a class="page-link"
                th:href="@{/products?page=__${listproduct.number+1}__}" aria-label="Next">
                <span aria-hidden="true">&raquo;</span>
            </a>
        </li>
    </ul>
</nav>
```



Kết hợp tìm kiếm và phân trang

(1) Thymeleaf

```
<form class="form-inline d-flex"
th:action="${#authorization.expression('hasAnyAuthority('ADMIN'))' ?
'/admin/products/search' : '/products/search'}" method="get">
  <input class="form-control flex-grow-1" type="search" placeholder="Search"
aria-label="Search" name="key">
  <button class="btn btn-outline-success flex-grow-1"
type="submit">Search</button>
</form>
```

(3) Service

```
public Page<Product> GetAll(int pageNo, int pageSize) {
    PageRequest pageRequest = PageRequest.of(pageNo, pageSize);
    return productRepository.findAll(pageRequest);
}
public Page<Product> GetSearchProducts(String key, int pageNo, int
pageSize) {
    Pageable pageable = PageRequest.of(pageNo, pageSize);
    return productRepository.searchProducts(key, pageable);
}
```

(4) Repository

```
@Repository
public interface ProductRepository extends JpaRepository<Product, Integer> {
    @Query("SELECT p FROM Product p WHERE p.name like %:key%")
    Page<Product> searchProducts(@Param("key") String key, Pageable pageable);
}
```

(2) Controller

```
@GetMapping("/search")
public String searchProducts(Model model, @RequestParam String key,
    @RequestParam(defaultValue = "0") int pageNo,
    @RequestParam(defaultValue = "2") int pageSize)
{
    Page<Product> products = productService.GetSearchProducts(key,
pageNo, pageSize);
    int totalPages = products.getTotalPages();
    model.addAttribute("listproduct", products);
    model.addAttribute("totalPages", totalPages);
    return "products/index";
}
@GetMapping("")
public String index(Model model,
    @RequestParam(defaultValue = "0") int page,
    @RequestParam(defaultValue = "2") int size)
{
    Page<Product> products = productService.GetAll(page, size);
    model.addAttribute("listproduct", products);
    model.addAttribute("totalPages", products.getTotalPages());
    return "products/index";
}
```

3 – Giỏ hàng

- Giỏ hàng chính là cầu nối trung gian giữa trang sản phẩm và quy trình thanh toán. Khách hàng có đi đến quyết định thanh toán và hoàn thành đơn hàng hay không phụ thuộc vào sự tối ưu của tính năng giỏ hàng.
- Các bước thực hiện:
 - (1) Tạo data model tương ứng cho giỏ hàng (**CartItem**)
 - (2) Tạo **Controller** cho giỏ hàng (**CartController**).
 - Sử dụng **Session** (**@SessionScope**) để lưu **List<CartItem>**
 - Thực hiện các **Action**:
 - ☐ **/cart**: Lấy ds CartItem trong giỏ hàng, tính tổng số lượng và tổng tiền
 - ☐ **/cart/add**: Thêm vào giỏ hàng
 - ☐ **/cart/update** : Cập nhật số lượng => tổng số lượng, số tiền
 - ☐ **/cart/remove/{productid}** : Xóa 1 Item trong giỏ hàng
 - ☐ **/cart/removeAll** : Xóa giỏ hàng (remove session)
 - ☐ Giữ số lượng giỏ hàng ở layout: sử dụng session



Tạo CartItem cho giỏ hàng

(1) Tạo model tương ứng cho giỏ hàng

- Thông tin của sản phẩm giỏ hàng
- Thông tin số lượng (quantity)

```
@Data
public class CartItem {

    //Product info
    private Integer id;
    private String name;
    private String image;
    private long price;

    //Quantity
    private int quantity;
}
```



Lưu giỏ hàng (cartitem) ở session

(2) Tạo **CartController**

2.1 Để lưu giỏ hàng ở session => Sử dụng **@Service**: CartService có scope = **@SessionScope**

- Danh sách các mục trong giỏ hàng
- Lấy danh sách
- Xóa danh sách
- ..

```
@Service
@SessionScope
public class CartService {

    private List<CartItem> cartItems = new ArrayList<>();
    public List<CartItem> getCartItems() {
        return cartItems;
    }
    public void clearCart() {
        cartItems.clear();
    }
}
```



Lấy giỏ hàng từ session đã lưu

(2.2) Viết Action **Index** để Lấy giỏ hàng

Ý tưởng: Lấy List<CartItem>, tính tổng số lượng, đơn giá tương ứng và tới view giỏ hàng (**cart/index**)

```
@GetMapping("")
```

```
public String getCartItems(Model model) {
    List<CartItem> cartItems = cartService.getCartItems();
    model.addAttribute("cartItems", cartItems);
```


```
// Calculate the total price
```

```
long totalPrice = cartItems.stream()
    .mapToLong(cartItem -> cartItem.getPrice() *
    cartItem.getQuantity())
    .sum();
    model.addAttribute("totalPrice", totalPrice);
```


```
// Calculate cart count
```

```
model.addAttribute("cartCount", cartItems.size());
return "cart/index";
```

```
}
```

#	Name	Image	Price	Quantity	Total
1	LAPTOP 1		1111	<input type="text" value="1"/>	1111
					Update
					Remove
					Total Price: 1111
					Order

View giỏ hàng

#	Name	Image	Price	Quantity	Total
1	LAPTOP 1		1111	<input type="text" value="1"/> <input type="button" value="Update"/>	1111 <input type="button" value="Remove"/>
Total Price: 1111					<input type="button" value="Order"/>

```
<div layout:fragment="content" class="container body-content">
  <table class="table table-striped">
    <thead>
      <tr>
        <th scope="col">#</th>
        <th scope="col">Name</th>
        <th scope="col">Image</th>
        <th scope="col">Price</th>
        <th scope="col">Quantity</th>
        <th scope="col">Total</th>
        <th scope="col"></th>
      </tr>
    </thead>
    <tbody>
      <tr th:each="cartItem, itemIndex : ${cartItems}">
        <td th:text="${itemIndex.index + 1}"></td>
        <td th:text="${cartItem.name}"></td>
        <td>
          
        </td>
        <td th:text="${cartItem.price}"></td>
        <td>
```

```
<form th:action="@{/cart/update}" method="post">
  <input type="hidden" name="productId" th:value="${cartItem.id}" />
  <input type="number" name="quantity" min="1"
th:value="${cartItem.quantity}" />
  <button type="submit">Update</button>
</form>
</td>
<td th:text="${cartItem.price * cartItem.quantity}"></td>
<td>
  <form
th:action="@{/cart/remove/{productId}}(productId=${cartItem.id})" method="post">
    <button type="submit" class="btn btn-danger">Remove</button>
  </form>
</td>
</tr>
<tr>
  <td colspan="5"></td>
  <td class="total-price" colspan="2">
    <h3>Total Price: <span th:text="${totalPrice}"></span></h3>
    <form th:action="@{/cart/order}" method="post">
      <button type="submit" class="btn btn-success">Order</button>
    </form>
  </td>
</tr>
</tbody>
</table>
</div>
```



Thêm vào giỏ hàng

(2.3) Viết AddToCart: thêm 1 sản phẩm vào giỏ hàng và redirect tới giỏ hàng

Ý tưởng: Nếu sản phẩm chưa có trong giỏ hàng -> Add vào danh sách: cartItems.

Ngược lại sẽ chỉ tăng số lượng

CartController

```
@PostMapping("/add/{id}")
public String addToCart(@PathVariable("id") Integer productId) {
    Product product = productService.get(productId);
    if (product != null) {
        cartService.addToCart(product);
    }
    return "redirect:/cart";
}
```

CartService

```
public void addToCart(Product product) {

    CartItem findCart = cartItems.stream()
        .filter(item -> item.getId().equals(product.getId()))
        .findFirst().orElse(null);
    if (findCart != null)
    {
        findCart.setQuantity(findCart.getQuantity()+1);
    }
    else
    {
        System.out.print("case item = null");
        findCart = new CartItem();
        findCart.setQuantity(1);

        findCart.setId(product.getId());
        findCart.setName(product.getName());
        findCart.setImage(product.getImage());
        findCart.setPrice(product.getPrice());

        cartItems.add(findCart);
    }
}
```



Cập nhật số lượng giỏ hàng

(2.4) Viết Action **UpdateCart** để cập nhật thay đổi giỏ hàng (số lượng)


Ý tưởng: - Được gọi từ Update ở View/update để thay đổi số lượng

CartController

```
@PostMapping("/update")
public String updateCartItem(@RequestParam("productId") Integer productId,
                             @RequestParam("quantity") int quantity) {
    cartService.updateCartItem(productId, quantity);
    return "redirect:/cart";
}
```

CartService

```
public void updateCartItem(Integer productId, int quantity) {
    CartItem findCart = cartItems.stream()
        .filter(item -> item.getId().equals(productId))
        .findFirst().orElse(null);
    if(findCart != null)
    {
        findCart.setQuantity(quantity);
    }
}
```

#	Name	Image	Price	Quantity		Total	
1	LAPTOP 1		1111	<input type="text" value="1"/>	<div>Update</div>	1111	<div>Remove</div>
					Total Price: 1111		
					<div>Order</div>		



Lưu giữ số item trong giỏ hàng

(2.5) Điều chỉnh lại Action /cart để lưu số item trong giỏ hàng

- Ý tưởng:
- Sử dụng session để lưu trữ số mục giỏ hàng
 - Thay đổi ở _layout tương ứng

```
no usages
@GetMapping("/{id}")
public String getCartItems(Model model, HttpSession session) {
    List<CartItem> cartItems = cartService.getCartItems();
    model.addAttribute( attributeName: "cartItems", cartItems);
    // Calculate the total price
    long totalPrice = cartItems.stream() Stream<CartItem>
        .mapToLong(cartItem -> cartItem.getPrice() * cartItem.getQuantity())
        .sum();
    model.addAttribute( attributeName: "totalPrice", totalPrice);
    // Calculate cart count
    session.setAttribute( name: "cartCount", cartItems.size());
    // model.addAttribute( "cartCount", cartItems.size());
    return "cart/index";
}
```

<!-- Cart Summary Section -->

```
<div sec:authorize="hasAnyAuthority('USER')" class="cart-summary">
    <a th:href="@{/cart}" class="navbar-brand"> Cart(<span
        {session.cartCount}></span>)</a>
```

Đặt hàng, thanh toán

❑ **Đặt hàng:** Đưa dữ liệu từ giỏ hàng vào CSDL.

=> Sử dụng 2 bảng để lưu giữ giỏ hàng: **Orders** và **OrdersDetail**

Orders: lưu đơn hàng: mã, trạng thái thanh toán, ngày đặt, tổng tiền ...

OrdersDetail: chi tiết từng sản phẩm trong đơn hàng...

Columns: + Add × Remove ▲ Up ▼ Down							
#	Name	Datatype	Length/Set	Unsign...	Allow N...	Zerofill	Default
1	id	BIGINT		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	AUTO_INCREMENT
2	is_paid	BIT	1	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL
3	order_date	DATETIME	6	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL
4	user_id	INT		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL
5	total_amount	BIGINT		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL

Columns: + Add × Remove ▲ Up ▼ Down							
#	Name	Datatype	Length/Set	Unsign...	Allow N...	Zerofill	Default
1	id	BIGINT		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	AUTO_INCREMENT
2	price	BIGINT		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL
3	quantity	INT		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL
4	order_id	BIGINT		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL



Tạo Entity: Orders và OrdersDetail

```

@Entity
@Data
public class Orders {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column
    private Date order_date;

    @Column
    private Boolean isPaid;

    @Column(name = "total_amount") // Add the total amount column
    private long totalAmount;

    // Other order properties (e.g., customer information, shipping address, etc.)
    @ManyToOne
    @JoinColumn(name = "user_id")
    private User user;

    @OneToMany(mappedBy = "order", cascade = CascadeType.ALL)
    private List<OrdersDetail> orderDetails;
}

```

```

@Data
@Entity
public class OrdersDetail {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @ManyToOne
    @JoinColumn(name = "order_id")
    private Orders order;

    @Column
    private long price;

    @Column
    private int quantity;
}

```



Đặt hàng

```

@PostMapping("/order")
public String Order() {
    // Get the currently logged-in user
    Authentication authentication =
    SecurityContextHolder.getContext().getAuthentication();
    User user = (User) authentication.getPrincipal();
    laptrinhungdungjava.springsecurity.entity.User findUser =
    userRepository.getUserByUsername(user.getUsername());

    cartService.orderCart(findUser);
    // Redirect to a success page or return a success message
    return "/cart/order.html";
}

```

@Autowired

```
private OrderRepository orderRepository;
```

@Transactional

```

public void orderCart(User user) {
    // Create a new Order
    Orders order = new Orders();
    order.setOrder_date(new Date());
    order.setIsPaid(false);
    order.setUser(user);
    // Iterate over cart items and create OrderDetails
    List<OrdersDetail> orderDetails = new ArrayList<>();
    for (CartItem cartItem : cartItems) {
        OrdersDetail orderDetail = new OrdersDetail();

        orderDetail.setOrder(order);
        orderDetail.setPrice(cartItem.getPrice());
        orderDetail.setQuantity(cartItem.getQuantity());
        orderDetails.add(orderDetail);
    }
    // Set order details in the order
    order.setOrderDetails(orderDetails);
    // Save the order to the database
    orderRepository.save(order);
    // Clear the cart
    clearCart();
}

```



Sau khi đặt hàng thành công: order.html

Đặt hàng thành công!

Đơn đặt hàng đã được ghi nhận.!

Tiếp tục mua sắm

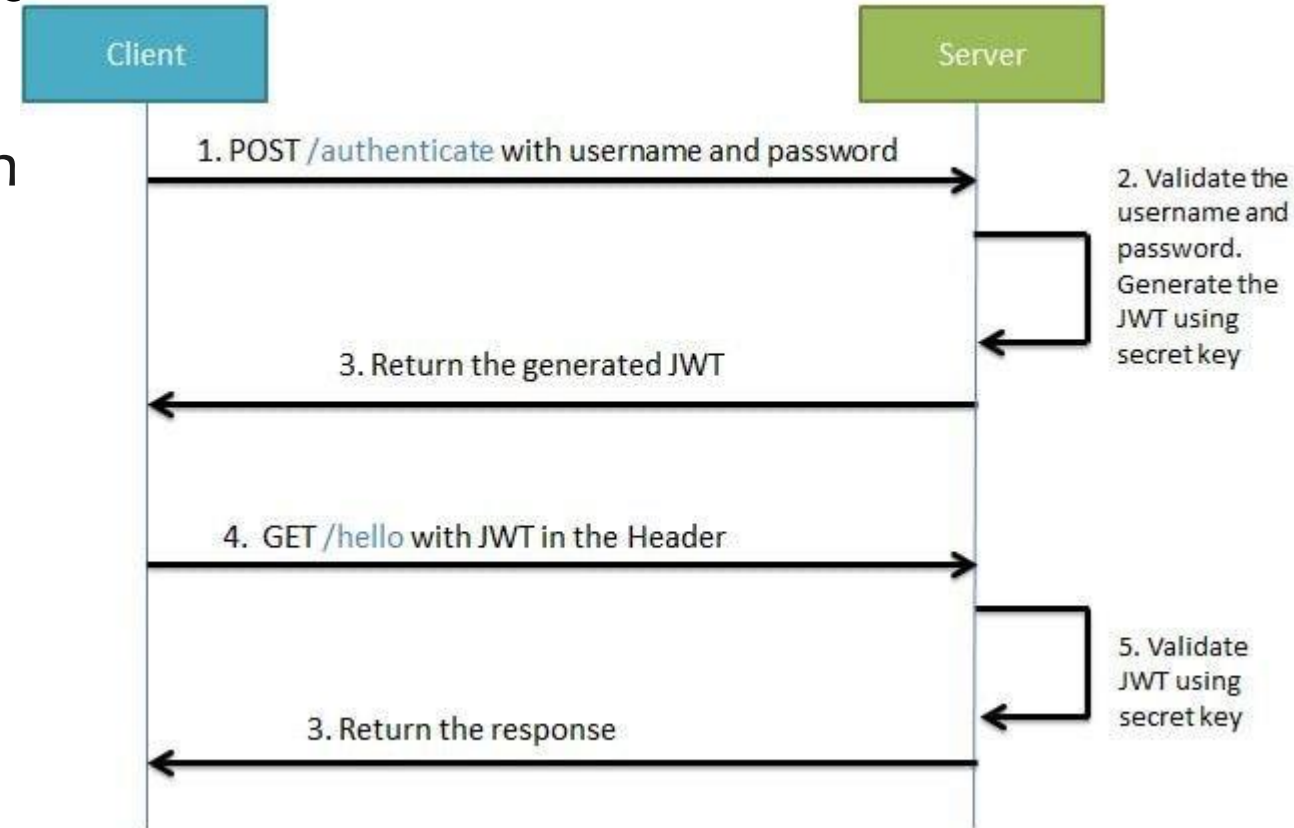
Thanh toán

```
<!DOCTYPE html>
<html lang="en"
  xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
  layout:decorate="_layout"
  xmlns:custom="http://www.w3.org/1999/xhtml">
<head>
  <meta charset="UTF-8">
  <title>List product</title>
</head>
<body>
<div layout:fragment="content" class="container body-content">
  <h1>Đặt hàng thành công!</h1>
  <p>Đơn đặt hàng đã được ghi nhận.!!</p>
  <a href="/products" class="btn btn-primary">Tiếp tục mua sắm</a>
  <a href="/cart/payment" class="btn btn-primary">Thanh toán</a>
</div>
</body>
</html>
```

4- JWT

- Bảo mật Web với Json Web Token (JWT)
- khi đăng nhập, mỗi yêu cầu kèm theo chuỗi token JWT
- Cấu trúc của Json Web Token: 3 phần ngăn cách nhau bởi dấu chấm (.)
 - ❑ Header
 - ❑ Payload
 - ❑ Signature

<https://jwt.io/>



4- JWT

- Tham khảo

<https://reflectoring.io/spring-security-jwt/>

<https://www.javaguides.net/2024/01/spring-boot-security-jwt-tutorial.html>