# Dưới đây là các tên + link học java sắp xếp có vẻ không đúng thứ tự.

# <https://www.tutorialspoint.com/java_dip/java_buffered_image.htm>

## Lập trình đa luồng với CompletableFuture trong Java 8

<https://viblo.asia/p/lap-trinh-da-luong-voi-completablefuture-trong-java-8-6J3ZgBMLKmB>

<https://gpcoder.com/3565-lap-trinh-da-luong-voi-callable-va-future-trong-java/>

\*\*

<https://stackjava.com/java8/completablefuture-la-gi-code-vi-du-java-completablefuture-java-8.html>

## Từ khóa Transient

<https://viettuts.vn/java-io/tu-khoa-transient-trong-java>

**Từ khóa transient trong java** được sử dụng trong serialization. Nếu bạn định nghĩa bất kỳ thành viên dữ liệu nào là transient, nó sẽ không được đánh dấu là tuần tự (serialize).

## Từ khóa Seriable trong java

<https://stackjava.com/java/java-serializable-la-gi-serialization-va-deserialization-trong-java.html>

* Serialization trong Java là cơ chế chuyển đổi trạng thái của một đối tượng (giá trị các thuộc tính trong object) thành một chuỗi byte sao cho chuỗi byte này có thể chuyển đổi ngược lại thành một đối tượng.
* Quá trình chuyển đổi chuỗi byte thành đối tượng gọi là deserialization.
* Một object có thể serializable (có thể thực hiện Serialization) nếu class của nó thực hiện implements interface java.io.Serializable
* [Home](https://stackjava.com/) » [Java](https://stackjava.com/category/java) » Java Serializable là gì? Serialization và Deserialization trong Java

Java Serializable là gì? Serialization và Deserialization trong Java

* Posted on [*Tháng Hai 8, 2018*](https://stackjava.com/java/java-serializable-la-gi-serialization-va-deserialization-trong-java.html)
* Java Serializable là gì? Serialization và Deserialization trong Java.
* Khi lập trình với Java chắc hẳn bạn đã bắt gặp khái niệm serialize nhất là khi đọc ghi object ra file, mapping với cơ sở dữ liệu…

Java Serializable là gì?

* Serialization trong Java là cơ chế chuyển đổi trạng thái của một đối tượng (giá trị các thuộc tính trong object) thành một chuỗi byte sao cho chuỗi byte này có thể chuyển đổi ngược lại thành một đối tượng.
* Quá trình chuyển đổi chuỗi byte thành đối tượng gọi là deserialization.
* Một object có thể serializable (có thể thực hiện Serialization) nếu class của nó thực hiện implements interface java.io.Serializable

Tại sao cần Serialization?

* Trong Java, khi trao đổi dữ liệu giữa các thành phần khác nhau (giữa các module cùng viết bằng Java) thì dữ liệu được thể hiện dưới dạng byte chứ không phải là đối tượng. Do đó ta cần có một cơ chế để hiểu các đối tượng được gửi và nhận.

## ****Khái niệm ThreadPool và Executor trong Java****

<https://kipalog.com/posts/Kha-i-nie--m-ThreadPool-va--Executor-trong-Java>

## ServletUriComponentsBuilder

## Vi-du-cac-loai-dependency-scope-trong-maven

<https://stackjava.com/maven/vi-du-cac-loai-dependency-scope-trong-maven.html>

## Cách convert mappstruct field này qua field khác

\*<https://stackoverflow.com/questions/48521903/map-custom-method-mapper-to-mapstruct>

<https://mapstruct.org/documentation/stable/reference/html/#mapping-method-resolution>

## Đọc ghi file ảnh trong java

<https://www.tutorialspoint.com/java_dip/java_buffered_image.htm>

## ResultMapping trong java( nó là jpa đấy nhé)

<https://stackoverflow.com/questions/25179180/jpa-joining-two-tables-in-non-entity-class/25184489#25184489>

Trong jpa thì bạn nên Select ntn nếu phải join thì lên join băng map hoặc new dto(b.x, b.a)

<https://stackoverflow.com/questions/36328063/how-to-return-a-custom-object-from-a-spring-data-jpa-group-by-query>

## ServletRequestAttribute và RequestContextHolder

RequestAttributes attrs = RequestContextHolder.*getRequestAttributes*();

Assert.*state*(attrs instanceof ServletRequestAttributes, "No current ServletRequestAttributes");  
return ((ServletRequestAttributes)attrs).getRequest();

**Có thể tóm tắt như sau:**

1. ServletRequestAttribute là 1 class chứa cả request(HttpServletRequest)

* Và reponse (HttpServletReponse)
* Hàm khởi tạo nó là 1 HttpServletRequest ( tức là đưa request đầu vào và nó phân tích cho bạn thôi)

1. RequestContextHolder: là 1 lớp context do spring tạo ra có thể lưu trữ các attr và convert nó thành 1 ServletRequestAttribute

* ((ServletRequestAttributes) RequestContextHolder.currentRequestAttributes()).getRequest()

**Các lớp hỗ trợ liên quan:**

* ServletUriComponentsBuilder (Cái này giúp lấy ra url của reponser cho request này
* UriComponentsBuilder

## Sự khác biệt giữa [RequestContextHolder , currentRequestAttributes() and getRequestAttributes()?](https://stackoverflow.com/questions/47586707/what-is-the-difference-between-these-methods-of-requestcontextholder-currentre)

<https://stackoverflow.com/questions/47586707/what-is-the-difference-between-these-methods-of-requestcontextholder-currentre>

## Cấu hình Swagger cho microService

<https://piotrminkowski.com/2017/04/14/microservices-api-documentation-with-swagger2/>

## Khái niệm Interceptor trong spring boot

<https://hocspringboot.net/2021/04/17/spring-boot-interceptor/>

<https://openplanning.net/11689/spring-boot-interceptor>

## LOGBACK

<https://howtodoinjava.com/logback/rollingfileappender/>

<https://viblo.asia/p/spring-spring-logback-config-GrLZDBgJ5k0>

Đây là lý do mà log méo thể ghi ra nhiều folder

<https://stackoverflow.com/questions/22188936/timebasedrollingpolicy-not-rolling-unless-there-are-new-logs>

## 15 Spring AUTO CONFIG

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

[**https://docs.spring.io/spring-boot/docs/2.0.0.M3/reference/html/boot-features-developing-auto-configuration.html**](https://docs.spring.io/spring-boot/docs/2.0.0.M3/reference/html/boot-features-developing-auto-configuration.html)

## 16 Cần hiểu sâu hơn về Servlet Nó có thể lấy cả ip host

<https://stackoverflow.com/questions/20969513/java-httprequest-getremoteaddr-allways-return-127-0-0-1>

<https://stackoverflow.com/questions/12491773/why-does-request-getremoteaddr-equals127-0-0-1-when-accessing-from-a-remot>

## 17 [Sự khác nhau giữa Rest-template và FeignClient trong Spring Boot](https://daynhauhoc.com/t/su-khac-nhau-giua-rest-template-va-feignclient-trong-spring-boot/117430)

<https://daynhauhoc.com/t/su-khac-nhau-giua-rest-template-va-feignclient-trong-spring-boot/117430>

## 18 DomParing\_Java Soap

<https://stackoverflow.com/questions/42279563/parsing-a-soap-message-with-dom-library-in-java>

## 19 Ghi file Java

<https://gpcoder.com/3046-huong-dan-su-dung-luong-vao-ra-nhi-phan-trong-java/>

## 20 Bộ nhớ trong java

<https://gochocit.com/ngon-ngu-lap-trinh/quan-ly-bo-nho-trong-java-bo-nho-stack-va-bo-nho-heap>

## 21 Hướng đối tượng là gì

<https://topdev.vn/blog/oop-la-gi/>

## Định nghĩa OOP là gì?

OOP (từ viết tắt của -Object Oriented Programming)\_lập trình hướng đối tượng là một phương pháp lập trình dựa trên các khái niệm về lớp và đối tượng. OOP thường tập trung vào các đối tượng thao tác hơn là logic để có thể thao tác chúng.

OOP là một nền tảng quen thuộc của các design pattern hiện nay. OOP đặt ra mục tiêu quản lý source code giúp gia tăng khả năng tái sử dụng và quan trọng hơn hết là có thể tóm gọn được các thủ tục đã biết trước tính chất thông qua quá trình sử dụng các đối tượng.

## 22. Collection và Collections khác gì nhau

* Collections là 1 class chứa các static method để thao tác vs tập dữ liệu
* colection thì là 1 interface trong collection framework

## 23. Abtract class và Interface

<https://viblo.asia/p/interface-vs-abstract-class-ke-tam-lang-nguoi-nua-can-07LKX9JeZV4>

Abstract class: nên có ít nhất 1 abstract method, có đẩy đủ tính năng như 1 class bình thường, có các access modifier cho abstract method, chỉ extend đc 1 abstract class, có constructor

[10:05](https://techasiansjsc.slack.com/archives/D03JJ4P6TNE/p1658070309068449)

Interface: chỉ chứa biến static final, chứa các method trìu tượng, mặc định là public, có thể extends nhiều interface khác, ko có constructor, default method đc implement trong interface và override ở class con, static method đc implement nhưng không đc override, ko thể khai báo interface là private và protected

1 class extends 1 class và implement nhiều interface

## 24. Quan hệ HAS-A, IS-A.

* Nếu class A **extends** class B: thì đó là quan hệ **is-a**. Vì sao? Is-a có nghĩa là cái này là cái kia. Khi A kế thừa B thì A sẽ có những tính chất của B, nói cách khác A là một bản sao có các tính chất của B.
* Nếu class A **implements** interface B: thì đó là quan hệ **has-a**. Vì sao? Has-a có nghĩa là cái này có khả năng gì đó. Khi A implements (triển khai - từ này dịch hơi ngựa) B thì là lớp A có các khả năng được mô tả trong các hàm của interface B.

## 25 Regex Nâng Cao

<https://viblo.asia/p/regular-expressions-regex-khong-he-kho-nhu-nhung-gi-ban-thay-ii-L4x5xg3YlBM>