Arrays.sort works for arrays which can be of primitive data type also. [Collections](https://www.geeksforgeeks.org/collections-in-java-2/).sort() works for objects Collections like [ArrayList](https://www.geeksforgeeks.org/arraylist-in-java/), [LinkedList](http://geeksquiz.com/linked-list-in-java/), etc.

Java 8 có Optional class giúp tránh  [NullPointerException](https://www.geeksforgeeks.org/null-pointer-exception-in-java/). mà không phải try catch nhiều ( nếu k trycatch sẽ văng lỗi)

**Singleton class**:

If you need to make sure only one instance of the class is created in your application, then use Singleton Pattern. For example, it can be used to store configuration items which the entire application can use. Also service classes can be singleton too so only a single service instance is used by all the calling classes in the application.

-static class chỉ được tạo ở inner class

DI là để nạp vào khi khởi tạo chương trình

Int là lớp nguyên thủy . Interger xem như là object

Khác biệt giữa đóng gói và trừu tượng.

Abstact 

Phải implement hết methods của abstract

<https://congdongjava.com/forum/threads/array-list-map-and-table.22192/#post-61537> // array,list

Array : fixed size>< Arraylist: dynamic

Array: both primitive and object>< Arraylist: object

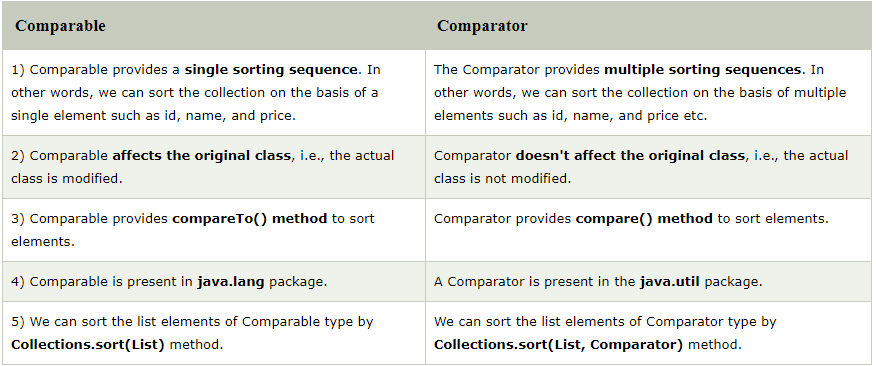
Array: Thêm là : a[0]=…>< ArrayList: add();

Array : length>< arraylist: size()

<https://www.geeksforgeeks.org/two-interfaces-methods-signature-different-return-types/>// interface problems : 2 cái interface cùng tên cùng kiểu trả về => lỗi

comparable implements compareTo( 1 parameter) : comparable chỉ order theo 1 cái duy nhất được define trong class

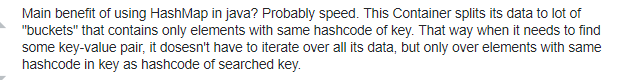
comparator có thể implements nhiều cái compareTo( 2 parameters)



hashset với treeset giá trị ko trùng nhau

Tree set khác vs hashset là tree set có thứ tự

the advantage of using HashMap



Array is **immutable><** **ArrayList** is kinda "**mutable**" array

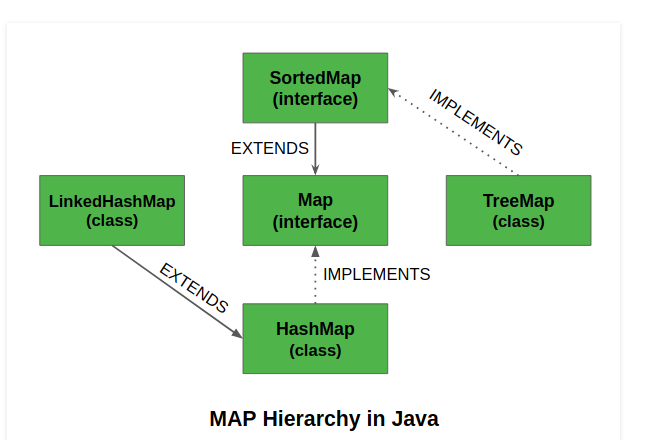
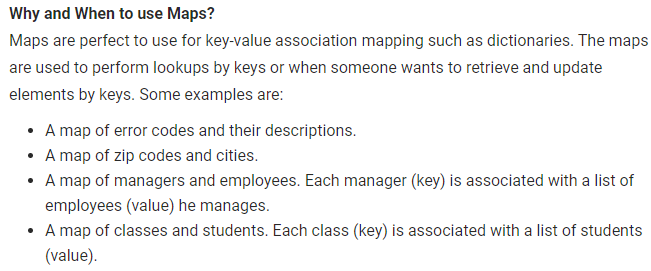
Lists and Maps are different data structures. Maps are used for when you want to associate a key with a value and Lists are an ordered collection.

Map is an interface in the Java Collection Framework and a HashMap is one implementation of the Map interface. HashMap are efficient for locating a value based on a key and inserting and deleting values based on a key. The entries of a HashMap are not ordered.

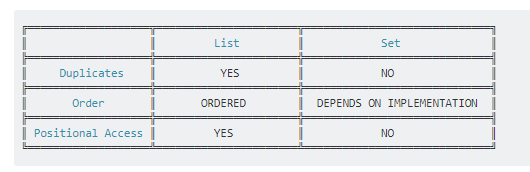
ArrayList and LinkedList are an implementation of the List interface. LinkedList provides sequential access and is generally more efficient at inserting and deleting elements in the list, however, it is it less efficient at accessing elements in a list. ArrayList provides random access and is more efficient at accessing elements but is generally slower at inserting and deleting elements.

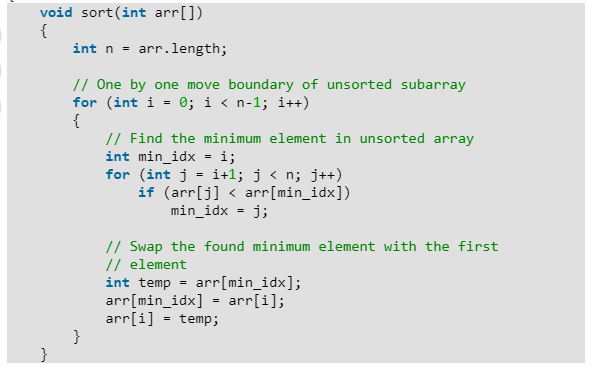
Linked list : thêm xóa dễ dàng nhưng không có truy cập index => duyệt mảng lâu ; Array list : truy cập từng phần tử dễ dàng theo index; thêm xóa sửa khó

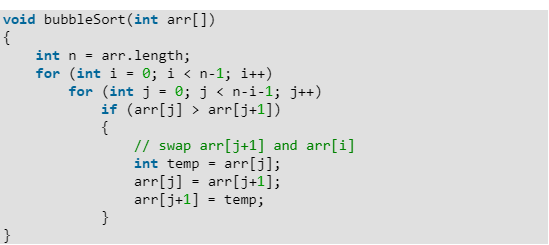
Order trong arraylist muốn phải dời các index kia ra, còn linked list k cần

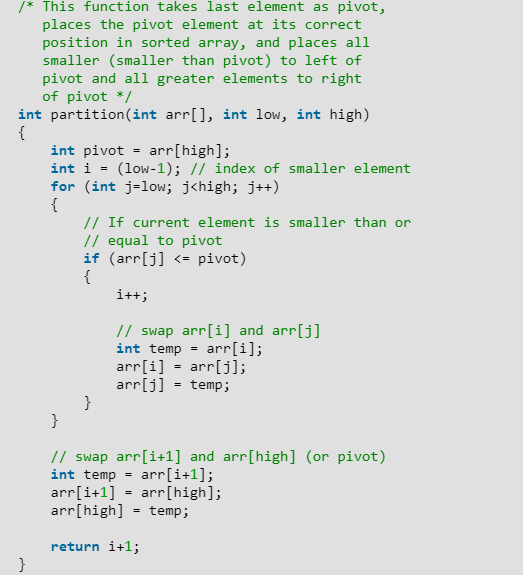
MAP : 

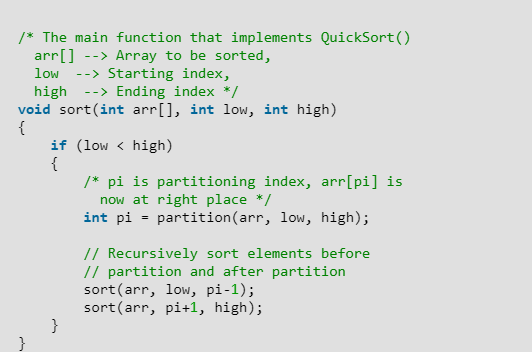
Map ko sort đc, muốn sort chuyển về list



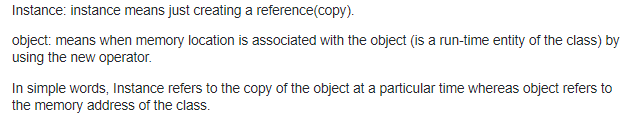
Selection sort: 

Bubble sort

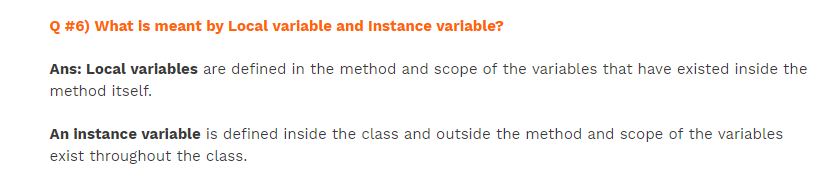
QuickSort 

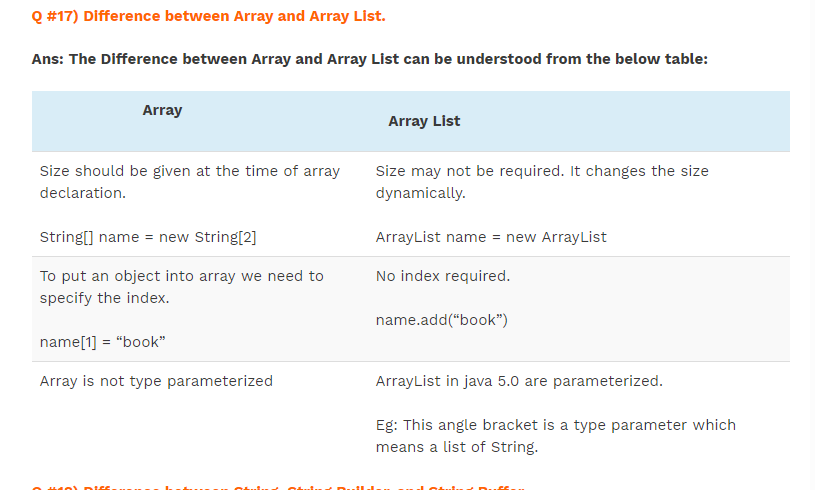


Instance and object









Static keyword<https://www.javatpoint.com/static-keyword-in-java>

Class Student{ static int a=1;}=> change a: Student.a=3; thay đổi 1 chỗ áp dụng nhiều nơi

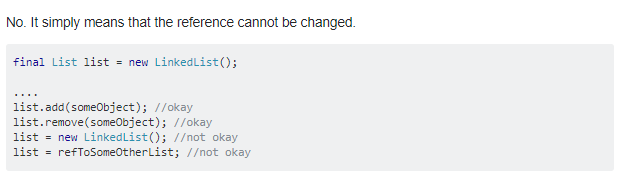
This keyword <https://www.javatpoint.com/this-keyword>

* This constructor must be the first statement in constructor

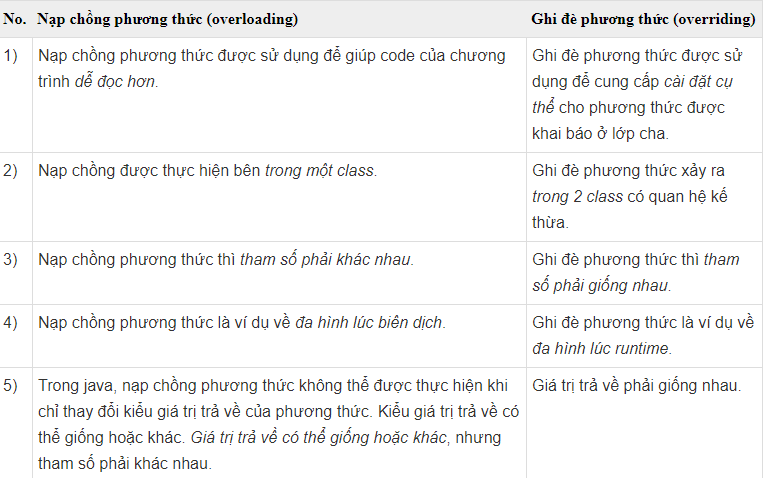
Super <https://www.javatpoint.com/super-keyword>

Final static variables <https://www.geeksforgeeks.org/final-static-variable-java/>

Biến này chỉ được initialize in static block, array có tạo final nhưng cũng change được array, nhưng reference của array ko changed=>

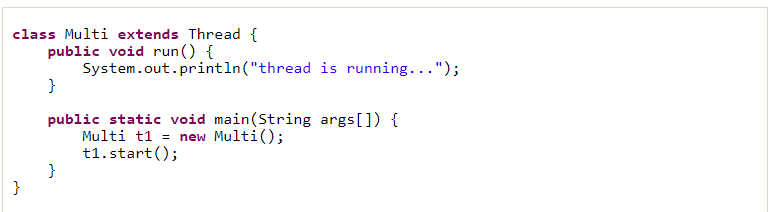
 final overload đc nhưng không override, biến final mà chưa init thì chỉ init trong constructor

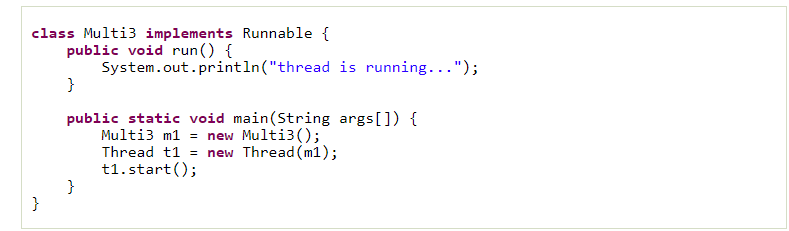
Overide and OverLoad



<https://github.com/MaximAbramchuck/awesome-interview-questions> // question4

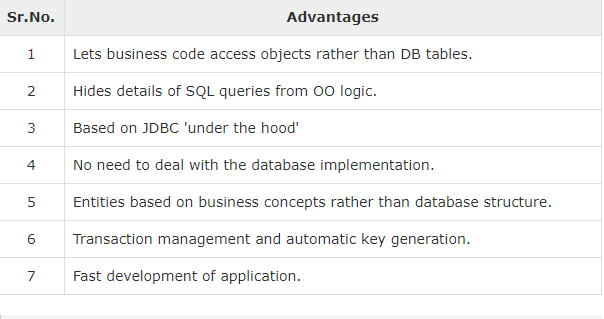
Có 2 cách tạo thread :

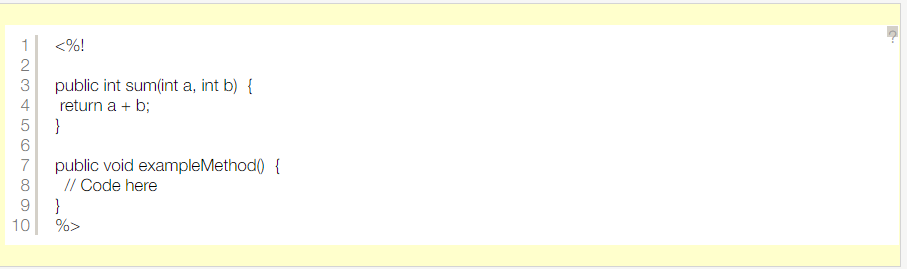
C1: 

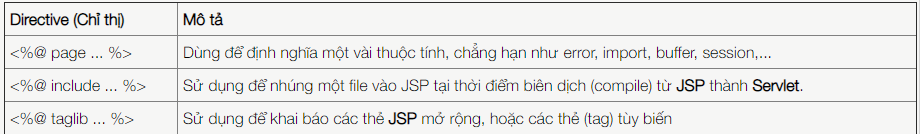
C2: 

Sleep : làm thread ngưng hđ;

Join: chờ khi thread này làm xong cái thread khác mới làm

Compare hibernate với JDBC

JSP 

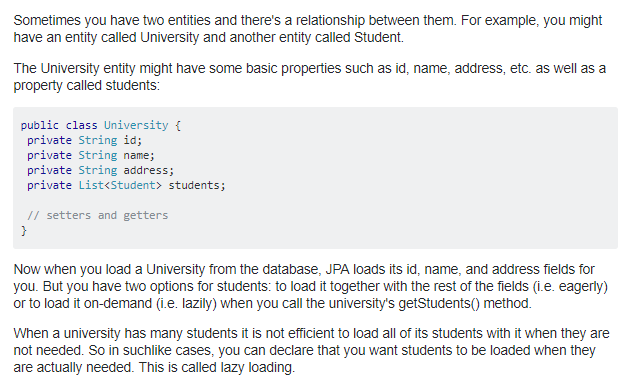


Inclue: nhúng file

**Hibernate:**

@OneTomany (mappedBy =’cart’)



=>LAZY = fetch when needed

EAGER = fetch immediately

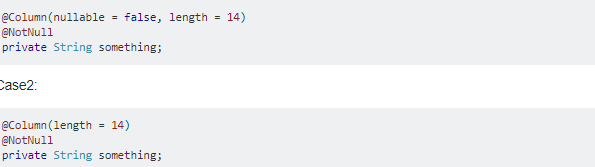
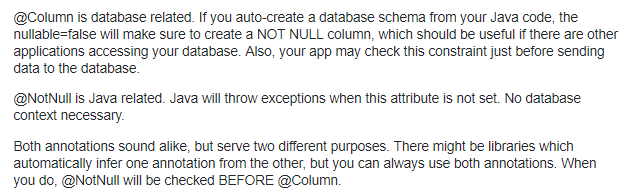
 @ManyToOne

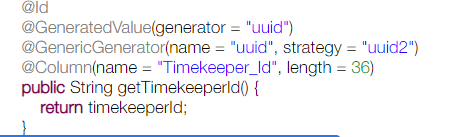
    @JoinColumn(name="cart\_id", nullable=false)

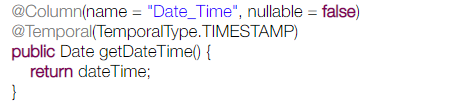
-Single field in hibernate : 



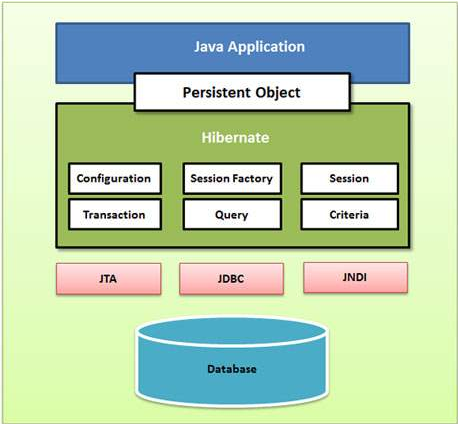
-Not Null in hibernate

-Generated ID in hibernate : 

Date Time in hibernate 

* HQL là truy vấn vào Entity. SQL truy vấn vào tables



-Configuration tạo đầu tiên , tạo 1 lần khi app initialization.chứa 2 keys : database connection; class mapping setup

-SessionFactory được tạo bởi configuration để sử dụng configuration file và tạo Session object, cái này cũng tạo 1 lần dùng suốt.Nhiều database thì nên tạo dùng SessionFactory.

-Session Object: dùng để lấy connection với database. Lúc nào cần giao tiếp vs database mới tạo cái này. Vì cái này k phải thread safe

Session nó có get, update, delete. Nếu muốn tạo query dùng TyperdQuery<class> query= session.createQuery(‘câu sql’,Product.class)=> tạo HQL query

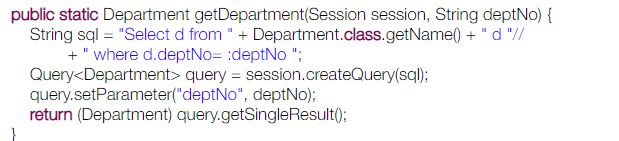
Session.createSqlQuery là tạo sqlquery

**Dialect là gì?** **Dialect** là một lớp, nó nói với **Hibernate** cách chuyển đổi các kiểu dữ liệu của **Database** thành các kiểu dữ liệu của **Java**, và ngược lại. Đồng thời nó cũng được sử dụng để định nghĩa cách chuyển đổi giữa các hàm của **HSQL** ( **Hibernate SQL**) thành các hàm của**Database** tương ứng.

Sql là truy vấn trên các bảng >< HQL truy vấn trên entity

Ví dụ HQL



Single Result 

# Phân biệt openSession() và getCurrentSession() trong Hibernate => openSession là khi thực hiện xong session vẫn mở, getCurrentSessiong là thực hiện xong nó tự đóng session đó

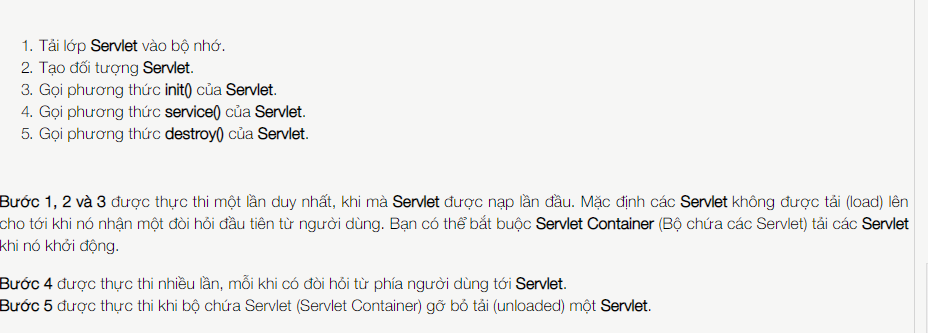
Hibernate lifecycle :

Khi tạo object = java => Transient

Khi ta save() or saveOrUpdate()hoặc mergeObject hoặc get data từ database => chuyển sang Persistent

**SERVLET**:

* Vòng đời Servelt

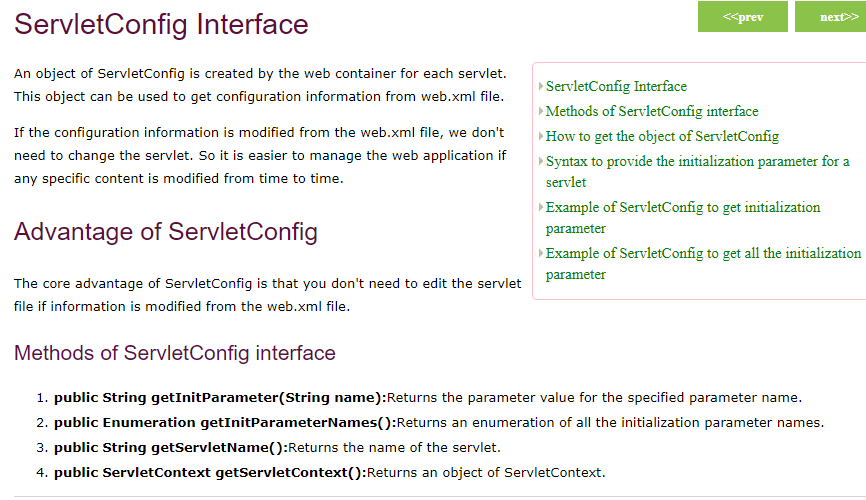


Cấu hình C1 trong xml : 

Khi gõ url xin chào=> servlet có tên hello=> lấy class HelloServlet

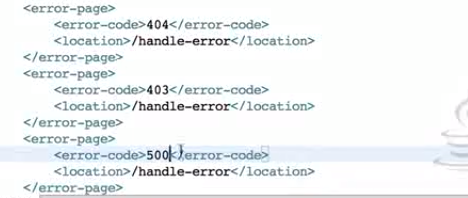
Cấu hình C2 trong javaServletRequest; ServletResponse

Response.addCookie; Response.setHeader, response.sendError

Servletconfig

Get parameter from url : ”ten” là tên biến

Trả về trang báo lỗi khi có lỗi 

=>Trong xml 

Redirect 

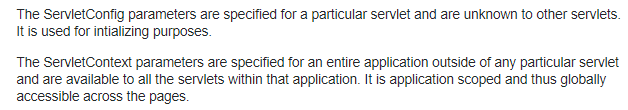
RequestDispatcher:

Phân phối các request :+ **forward :**Forwards a request from a servlet to another resource (servlet, JSP file, or HTML file) on the server.

+**include:** Includes the content of a resource (servlet, JSP page, or HTML file) in the response.

Ví dụ trong form login: nếu đúng thì chuyển sang trang home:forward ; sai chuyển về lại trang login +response=> include

ServletConfig compare ServletContext

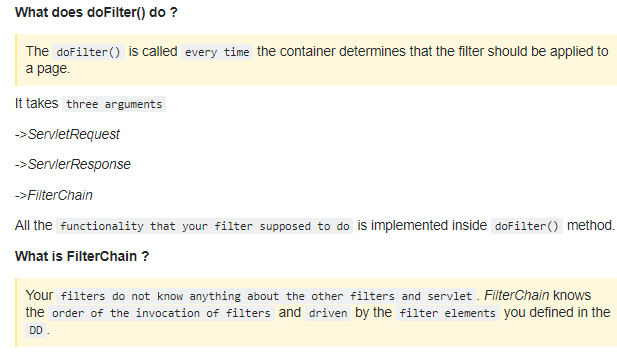
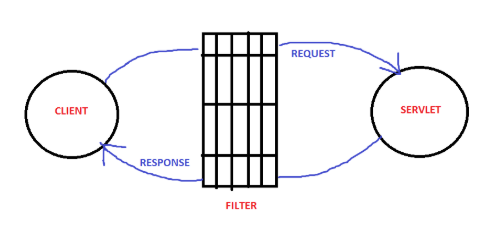


Session

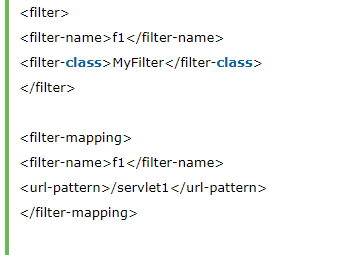
Set : 

Get: 

Filter: dùng để check login, ip, validate



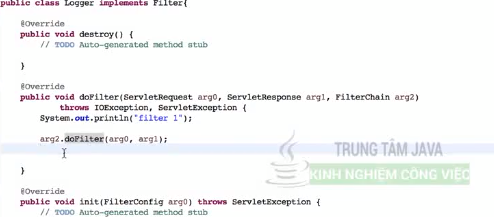
Khai báo XML



Mapping tất cả url 

Khai báo = annotation 

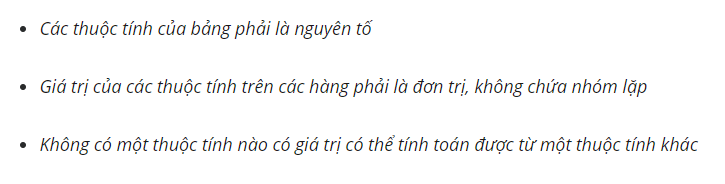
Filter 3 trạng thái init, doFilter, destroy=> thực thi ở Filter

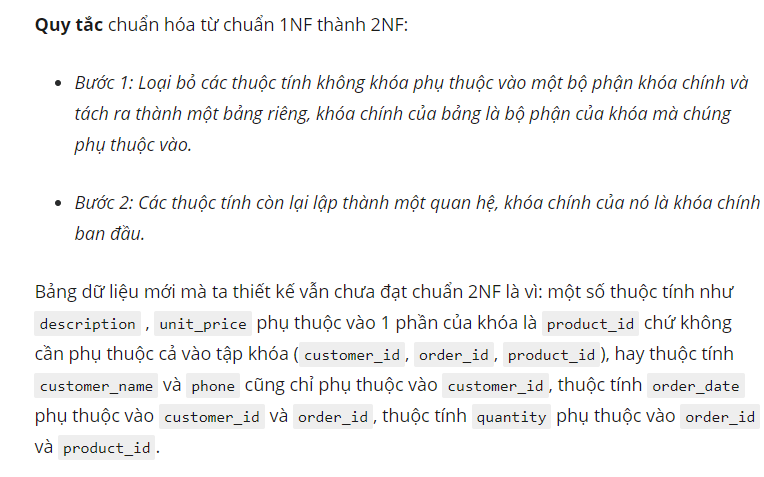


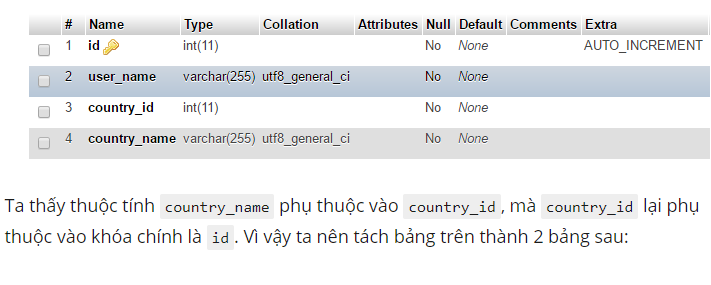
doFIlter là cho đi qua các filter tiếp theo , nếu ko có cái này người dùng sẽ bị chặn lại

FilterConfig : khởi tạo giá trị ban đầu cho filter => get 

**SQL**

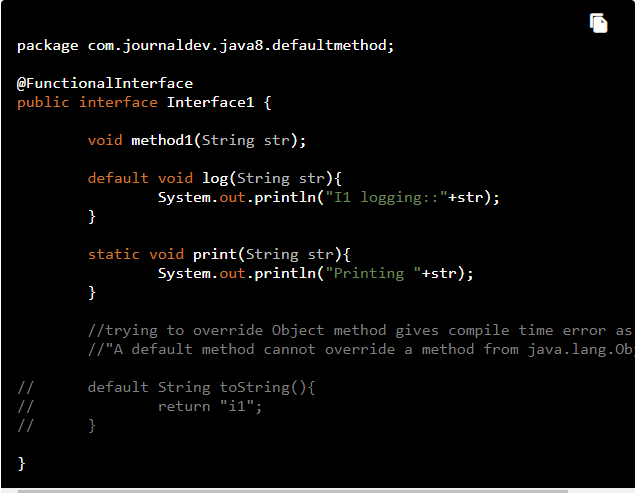
Chuẩn 1: Ví dụ địa chỉ nhà tách đường , quận ,… total tính = uniquePrice\*quantity

Dạng chuẩn 2: 

Dạng chuẩn 3 là: 1 thuộc tính phụ thuộc vào thuộc tính khác mà k phải khóa 

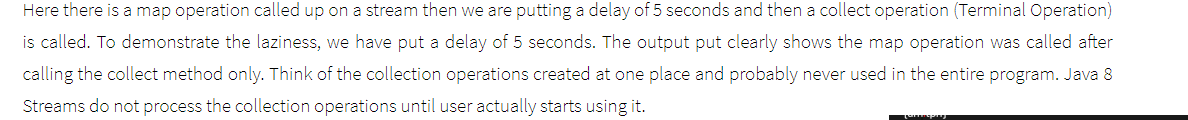
**JAVA 8**

-Trong Java 8 interface có thể có phương thức có nội dung bằng khai báo default hoặc static



-Stream giúp lazy loads collections: chỉ chạy khi gọi





-Optional để kiểm tra Null hay ko