# Java Core

## JVM

### JVM (Java Virtual Machine) là một máy ảo Java, nó hoạt động cung cấp môi trường run-time để có thể thực thi được các file java bytecode và thông dịch chúng sang mã máy.

#### Classloader: là một hệ thống con được sử dụng để load file. Bất cứ khi nào ta chạy chương trình Java thì classloader được load đầu tiên:

* Bootstrap ClassLoader: Đây là classloader đầu tiên, nó là super class của Extension classloader. Nó load the rt.jar flie chứa tất cả các file của Java Standard Edition như java.lang package clas,jave.net
* Extension ClassLoader: lớp con của Bootstrap class, nó load jar file xác định vị trí bên trong $JAVA?HOME/jre/lib…
* System/Application ClassLoader: lớp con của Extension, nó load class file từ class path.

#### JVM Memory:

* Method Area: lưu trữ tất cả các mức độ thông tin như class name, class name của cha trực tiếp, methods and variables…(meta data) Mỗi JVM chỉ có một method area và được chia sẻ với nguồn
* Heap Memory: Thông tin của tất cả các đối tượng được lưu trữ trong heap area. Mỗi JVM có một heap memory và cũng được chia sẻ nguồn
* Stack Area: Với mỗi luồng, JVM tạo một run-time stack được lưu trữ tại đây, nó lưu trữ, Khi một phương thức mới được gọi, một vùng nhớ mới sẽ được tạo trong Stack. Vùng mới này sẽ lưu trữ các giá trị tạm thời được gọi bởi phương thức và tham chiếu đến các đối tượng được lưu trữ trong Heap đang được phương thức sử dụng. Bất kỳ giá trị nào trong khối này chỉ có thể truy cập bằng phương thức hiện tại và sẽ không tồn tại khi nó kết thúc. Khi phương thức kết thúc, khối đó sẽ bị xóa. Phương thức tiếp theo được gọi sẽ sử dụng khối rỗng đó. Phương thức “LIFO” này giúp dễ dàng tìm thấy các giá trị cần thiết và cho phép truy cập nhanh vào các giá trị đó.
* PC Registers: Lưu trữ địa chỉ của
* Native method stack

#### Execution Engine:

#### Java Native Interface:

#### Native Method Libraries

## Memory, Garbage Collection In Java

### Heap Memory

### Stack Memory

### Garbage Collection

## Complie and Interpreter in Java

### Stack Memory: Là không gian vật lý (trong RAM) được phân vổ cho mỗi luồng chạy tại run time. Nó được tạo khi 1 luồng được tạo ra. Stack memory follow LIFO. Nó lưu trữ các biến

### Heap Space: Nó được tạo ra khi JVM khởi động và được sử dụng mỗi khi ứng dụng chạy. Nó lưu trữ các đối tượng

## Data Type:

### Data Type xác định size và kiểu của giá trị được truyền vào để hệ thống có thể tiến hành tạo vùng nhớ để lữu trữ

### Có 2 kiểu dữ liệu chính trong Java: Primitive Data Type và Non-Primitive Data Type

### Primitive: Là kiểu dữ liệu được xác định sẵn, có 8 kiểu dữ liệu Primitive: byte, short, int, long, float, double, boolean, char.

### Non-Primitive(Reference): Là kiểu dữ liệu được được người dùng tạo ra và nó không được xác định sẵn: String, Array, …

### Khác nhau của Primitive Data Type và Reference :

* Primitive luôn luôn có giá trị cụ thể, Reference thì có thể null
* Primitive được lưu trữ giá trị tại Stack Memory, Reference được lưu trữ tại Heap Memory
* Trong Java biến reference không là con trỏ giống như C/C++ nhưng khi xử lý một object thì nó cũng có thể làm thay đổi property value của object. Biến pritimitive thì không
* Primitive biến lưu trong stack memory, Reference biến được lưu trong heap memory
* Primitive biến sử dụng bộ nhớ ít hơn reference biến

## Wrapper Class, Nested Class, Anonymous Class, Mutable, Imutable Object

### wrapper class trong Java cung cấp cho chúng ta cơ chế chuyển đổi từ các data type primitive sang object (autoboxing) và ngược lại(unboxing)

### Nested Class

### Anonymous Class

### Mutable

### Imutable Object

## OOP

### Encapsulation (Tính đóng gói): Tính đóng gói là kỹ thuật ẩn giấu thông tin nhạy cảm và có thể giảm thiểu được mức độ phức tạp phát triển phần mềm. Trong một class các biến của class phải được ẩn đối với các class khác để có thể bảo mật các thông tin và dữ liệu của đối tượng và các thuộc tính đó chỉ được truy cập đến qua các phương thức (Get/Set)

### Inheritance (Tính kế thừa): Một class (class con) có thể kế thừa từ một class (class cha) khác bởi từ khóa extends class con sẽ được kế thừa tất cả các thuộc tính và phương thức (public và protected) của class cha

### Polymorphism (Tính đa hình): Một đối tượng có thể thực hiện một hành động theo nhiều cách khác nhau. Có 2 kiểu đa hình trong java : đa hình lúc biên dịch và đa hình lúc runtime

#### Đa hình lúc biên dich (Overloading): Trong một class có thể có nhiều phương thức có cùng tên khác nhau về data type truyền vào, số lượng biến truyền vào và chúng có thể khác data type trả về

#### Đa hình lúc runtime Over ridding : Khi một lớp con có cùng phương thức với lớp cha lớp con có thể triển khai lại phương thức đó

### Abtraction (Tính trừu tượng): Trừu tượng là ta phải ẩn đi những chi tiết thông tin không cần thiết. Ta chỉ quan tâm đến kết quả mà nó thực hiện và không cần biết đến các bước nó thực hiện.

#### VD: khi ta gọi điện thoại ta chỉ cần quan tâm đến cuộc gọi được kết nối với đúng đối tượng mà không cần quan tâm đến cách nó kết nối và truyền gửi thông tin

#### Trừu tượng trong Java chúng ta có thể dung Abstract Class và Interface

## String, String Builder, String Buffer

### String

#### String là một chuỗi các ký tự và String được coi như là Object

#### String là bất biến vì vậy nó không thể thay đổi

#### String được tạo bằng 2 cách : dùng dấu nháy kép “” hoặc khởi tạo bằng từ khóa new

#### Java String class cung cấp một số các phương thức để thực hiện các thao tác trên String như: compare(), concat(), equals(), split(), length(), …

#### String pool là một khu vực lưu trữ bên trong Java Heap, khi ta tạo String mà không dùng từ khóa “new” thì nó sẽ được lưu trong String Pool và khi ta tạo một String mới nếu String đó tồn tại trong String Pool thì String mới sẽ chứa địa chỉ của String trong String pool

### String Builder

#### **String Builder** trong Java đại diện cho một chuỗi ký tự có thể thay đổi. Vì String Class trong Java tạo ra một chuỗi ký tự bất biến, nên lớp StringBuilder cung cấp một sự thay thế cho Lớp Chuỗi, vì nó tạo ra một chuỗi ký tự có thể thay đổi

#### String Builder là một class nên nó có các Constructor với các giá trị truyền vào

#### String Builder được cung cấp các phương thức để xử lý các thao trên String

### String Buffer

#### String Buffer là một class cung cấp các chức năng cho String.

#### **StringBuffer** có thể có các ký tự và chuỗi con được chèn vào giữa hoặc nối vào cuối. Nó sẽ tự động phát triển để nhường chỗ cho những bổ sung

### String Builder với String Buffer :

#### String Builder với String Buffer giống nhau về các chức năng thao táo lên String

#### String Builder có khả năng xử lý chuỗi nhanh, mạnh mẽ

#### String Builder không được đồng bộ hóa, không an toàn cho luồng

#### String Buffer được đồng bộ hóa

## Collection:

### Cung cấp các cấu trúc để lưu trữ và thao tác với nhóm đối tượng. Java Collection cung cấp các Class (ArrayList, Vector, LinkList, HashSet, …) và Interface (Set, List, Queue, Deque)

### List :

#### Cơ chế mở rộng phần tử của List

#### ArrayList: Cung cấp cho chúng ta một mảng động trong Java và nó sẽ chậm hơn các mảng tĩnh nhưng nó sẽ rất hữu ích cho việc thao tác trên mảng.

##### Không thể lưu trữ Primitive Data Type mà ta cần phải autoboxing sang wrapperclass, có thể lưu trữ các giá trị trùng lặp

##### cho phép truy cập ngẫu nhiên vì mảng hoạt động trên cơ sở chỉ mục, duy trì thứ tự chèn

##### Thao tác chậm hơn so với LinkList nếu thay đổi kích thước mảng

##### ArrayList không được đồng bộ hóa

#### LinkList: Là một cấu trúc dữ liệu tuyến tính, các phần tử tách biệt phần dữ liệu và phần địa chỉ, mỗi phần tử được gọi là một nút.

##### Không thể lưu trữ Primitive Data Type mà ta cần phải autoboxing sang wrapperclass , có thể chứa các phần tử trùng lặp, duy trì thứ tự chèn

##### Thao tác nhanh chóng không cần phải dịch chuyển

##### Có thể được sử dụng như một list, stack, queue,

### Set, Hash Set

#### Set

#### Hash Set

#### Cơ chế của Set khi so sánh các phần tử giống nhau

### Map

#### Hash Map: Cho ta lưu dữ liệu dưới dạng cặp (Key, Value)

##### Chỉ chứa các giá trị key duy nhất

##### Có thể có một key null và nhiều value null

##### Không được đồng bộ hóa , duy trì không có thứ tự

#### Hash Table: Là một mảng danh sách. Mỗi danh sách được gọi là một nhóm

##### Chỉ chứa các phần tử duy nhất

##### Không được phép null Key hoặc Value

##### Được đồng bộ hóa

## Thread :

### Thread là một lồng thực thi xử lý trong một chương trình,

### JVM cho phép một ứng dụng có nhiều luồng thực thi xử lý được chạy đồng thời và hoạt động độc lập nên bạn có thể nhiều tác vụ cùng một lúc để tiết kiệm thời gian (Multithreading)

### Vòng đời của thread có 5 trạng thái: New, Runnable, Running, Non-Runable(Blocked), Terminated

#### New: Thread ở trạng thái new nếu ta tạo một đối tượng của lớp Thread nhưng trước khi gọi phương thức start()

#### Runnable: Thread ở trạng thái runnable sau khi gọi phương thức start(), nhưng trình lên lịch (scheduler) của thread đã không chọn nó là thread đang chạy

#### Running: Thread ở trạng thái running nếu trình lên lịch của thread đã chọn nó

#### Non-Runnable: Trạng thái khi thread vẫn còn sống, nhưng hiện tại ko được chọn để chạy

#### Terminated: Thread ở trong trạng thái terminated/dead khi phương thức run() của nó bị thoát

### Các luồng đều có độ ưu tiên, luồng nào có độ ưu tiên cao hơn sẽ được ưu tiên thực hiện so với các luồng có độ ưu tiên thấp hơn

### MultiThread

### ThreadPool

### Deadlock, xử lý Deadlock

### Cách quản lý thread trong bộ nhớ

## Comparable và Comparator

### Comparable : Comparable là một interface được sử dụng để sắp xếp các đối tượng của người dùng định nghĩa dựa trên một thuộc tính nhất định của đối tượng. Trong interface Comparable chỉ có duy nhất phương thức compareTo() và chỉ có thể sắp xếp duy nhất một thuộc tính của đối tượng

### Comparator: Comparator là một intrtface được sử dụng để sắp xếp các đối tượng có cùng data type. Ta có thể tạo nhiều phép so sánh theo các thuộc tính mà chúng ta muốn so sánh

## Exception : Exception là một ngoại lệ không tuân theo luồng chạy mong muốn và xảy ra trong quá trình thực hiện chương trình.

### Có 3 loại Exception: Checked Exception, Unchecked Exception, Eror:

### Exception Handling: Đây là một cơ chế mạnh mẽ để xác định và đưa ra các lỗi ngoại lệ và tiếp tục duy trì hoạt động của chương trình

### Try-Catch:

#### Try dung để bao bọc khối lệnh có thể có Exception, nếu Exception xảy ra thì phần còn lại của khối lệnh sẽ không được thực thi

#### Catch dung để xử lý Exception bằng cách khai báo biến Exception

### Throw vs Throws:

#### Throw: được sử dụng ném một Exception rõ rang bên trong phương thức.

#### Throws: Được sử dụng để khai báo Exception sử dụng khi khai báo method và theo sau là tên các Exception được khai báo

## “This”, “Static”, “Final” key word

### Static key word

#### Từ khóa Static trong Java chủ yếu được sử dụng cho quản lý bộ nhớ. Từ khóa static được dùng để chia sẻ cùng một biến hoặc phương thức của một class. Ta có thể dùng từ khóa static với biến, phương thức, khối, class lồng nhau.

#### Từ khóa static thuộc về class hơn instance của class

#### Static blocks: được sử dụng để khởi tạo Static data member, nó sẽ được thực thi trước hàm main tại thời điểm mà class được gọi đến lần đầu tiên

#### Static variable: Khi một biến được khai báo là static thì một bản sao duy nhất của biến được tạo và chia sẻ ccho tất cả các đối tượng của class, static variable are essentially, global variables. Ta chỉ có thể tạo biến static trong mức độ class

#### Static method: Là các phương thức được khai báo với từ khóa static. Chỉ có thể gọi trực tiếp các phương thức static khác, chỉ truy cập trực tiếp các static data, không thể sử dụng từ khóa “this” or “super”

### Final key word :

#### Từ khóa final được sử dụng để hạn chế người dùng và trong nhiều ngữ cảnh khác nhau

#### Final variable: Khi một biến được khai báo với từ khóa final thì giá trị của nó sẽ sẽ không được sửa đổi

#### Final Method: Nếu một method được khai báo là final thì bạn không thể override nó

## Equal(), hashCode() Method in Java

# TECH WEB

## HTML:

### HTML là viết tắt của Hype Text Markup Language (Ngôn ngữ đánh dấu siêu văn bản)

### HTML là ngôn ngữ đánh dấu tiêu chuẩn để tạo ra một trang Web, là cấu trúc của một trang Web

### HTML gồm các element, các elements nói với trình duyệt rằng các thức để hiển thị nội dung

### <!DOCTYPE html> Khai báo rằng tài liệu này chính là một file HTML5, <element> thẻ gốc của trang HTML, <body> Xác định phần thân chứa các nội dung chính ra muốn hiển thị cho người dùng, <h1> to <h6> Thẻ xác định tiêu đề của nhóm nội dung, <p> Hiển thị một đoạn văn

<img> Hiển thị ảnh

<a href=”url”> Hiển thị một đường dẫn

<form> Hiển thị một form cho người dùng nhập

<input> Người dùng có thể nhập giá trị vào thẻ này

<option> Đưa ra các lựa chọn cho người dùng

<table> Hiển thị theo dạng bảng

## CSS:

### CSS viết tắt của Cascading Style Sheets là một ngôn ngữ được sử dụng để tạo kiểu cho một trang Web. CSS sẽ quyết định các elements của HTML sẽ được hiển thị lên màn hình như thế nào. CSS tiết kiệm rất nhiều cho lập trình viên, nó có thể kiểm soát bố cục nhiều màn hình cùng lúc

### CSS giúp chúng ta điều chỉnh bố cục một cách linh hoạt

### CSS có thể thay đổi các thuộc tính của các element HTML:Color, Background, Border, Padding, Margin, Text, Icon, Position….

## JAVASCRIPT:

### JavaScript  là một ngôn ngữ lập trình hướng đối tượng và là ngôn ngữ thông dịch, nó giúp bạn phát triển tốt về font-end cũng như là back-end

### JavaScript là ngôn được sử dụng rất rộng rãi nhất, nó có thể thực hiện rất nhiều vai trò trong Application: Client side, Back-end Data, Sever Applications,..

### document.getElementById(id): Tìm và lấy về một phần tử bằng id

### document.getElementByTagName(name) : TÌm các phần tử bởi tên thẻ(a,p,div,img,….), giá trị trả về sẽ là một mảng các phần tử.

### document.getElementByClassName(name): TÌm các phần tử bởi tên class(a,p,div,img,….), giá trị trả về sẽ là một mảng các phần tử.

### element.innerHTML = new value Được sử dụng để lấy về hoặc thay đổi giá trị của thẻ element

### element.attribute = new value Thay đổi giá trị thuộc tính của element

## HTTP Protocol: (Giao thức truyền siêu văn bản): Là cơ sở của World Wide Web và được sử dụng để tải các trang Web bằng cách liên kết các siêu văn bản. HTTP là một “application layer protocol” được thiết kế để truyền thông tin giữa các thiết bị được kết nối Internet sử dụng dịch vụ (Web client) tới máy chủ(Web server) để người dùng có thể dễ dàng truy cập các dữ liệu.

### GET: phương thức GET sử dụng URL để truy xuất lấy về dữ liệu từ server. Request sử dụng phương thức GET chỉ nên lấy dữ liệu không nên có các tác động khác đến dữ liệu

#### GET requests có thể cached

#### GET requests duy trì trên lịch sử trình duyệt

#### GET requests có thể được đánh dấu

#### GET requests không nên được sử dugj khi xử lý thông tin nhạy cảm cần được bảo mật

#### GET requests có hạn chế độ dài

#### GET requests chỉ được sử dụng để yêu cầu dữ liệu

### PUT: phương thức PUT: Nếu URL đề cập đến một tài nguyên đã có, nó sẽ bị sửa đổi, nếu URL không trỏ đến một tài nguyên hiện có, thì máy chủ có thể tạo ra tài nguyên với URL đó.

### POST: phương thức POST được sử dụng để gửi dữ liệu tới server để tạo hoặc sửa resource

#### Sự khác biệt giữa POST và PUT là các yêu cầu PUT là “idempotent”. Đó là, gọi cùng một yêu cầu PUT nhiều lần sẽ luôn tạo ra cùng một kết quả. Ngược lại, việc gọi một yêu cầu POST liên tục có tác dụng phụ là tạo cùng một resource nhiều lần.

#### Dữ liệu gửi tới server khi dùng POST sẽ được lưu trữ tại request body của HTTP request

#### POST request sẽ không bao giờ cached

#### POST request sẽ không bao giờ duy trì trên lịch sử browser

#### POST request không thể được đánh dấu

#### POST request không có hạn chế độ dài dữ liệu

### DELETE: phương thức DELETE được yêu cầu tới server để xóa tất cả các đại diện của tài nguyên chỉ định bởi URL

## FORM

### Form được dùng bắt buộc khi bạn muốn thu nhập một số dữ liệu từ người dùng truy cập sử dụng trang web. VD: khi người dùng Register, bạn muốn thu thập thông tin như tên, địa chỉ, mail,… thì một form sẽ nhận input từ người dùng và sẽ gửi nó đến một back-end application và thực hiện xử lý yêu cầu đối với dữ liệu được gửi.

### Action: file backend bạn muốn truyền dữ liệu đến

### Method: Phương thức được sử dụng để gửi dữ liệu. Chúng ta thường sử dụng 2 method là GET và POST

### Target: Xác định cửa sổ hoặc khung nởi kết quả sẽ được hiển thị (\_blank, \_self, \_parent, \_top)

### Enctype: chỉ định cách trình duyệt mã hóa dữ liệu trước khi nó gửi đến máy chủ (application/x-www-form-urlencoded, multipart/form-data, text/plain)

## DESIGN PATTERN

### Factory Pattern: Chỉ tạo một Object thông qua Interface hoặc Abstract class, class con sẽ quyết định xem Object nào sẽ được tạo ra

### Abstact Factory Pattern: Chỉ xác định một Interface hoặc Abstract class để tạo các đối tượng liên quan (hoặc phụ thuộc) nhưng không chỉ định các lớp con cụ thể của chúng. Abstract Factory Pattern là level cao hơn cảu Factory Pattern

### Singleton Pattern là một trong những mẫu thiết kế đơn giản nhất trong Java. Nó cung cấp một trong những cách tốt nhất để tạo một đối tượng, chỉ một đối tượng duy nhất được tạo. Class này cung cấp một cách duy nhất để truy cập đối tượng duy nhất của nó có thể được truy cập trực tiếp mà không cần khởi tạo đối tượng của lớp

### Bulder

### Adapter

### CQRS

### Dependency Ịnection

### SOLID

# DATABASE

## ResultSet

**Đối tượng của ResultSet** duy trì một con trỏ trỏ đến một hàng của một bảng. Ban đầu, con trỏ trỏ đến hàng đầu tiên.

## Statement

Các câu lệnh SQL đọc dữ liệu từ một truy vấn cơ sở dữ liệu, trả về dữ liệu trong một tập kết quả. Câu lệnh SELECT là cách tiêu chuẩn để chọn các hàng từ cơ sở dữ liệu và xem chúng trong một tập kết quả

## Connection

Connection là một phiên giữa ứng dụng Java và cơ sở dữ liệu. Nó giúp thiết lập kết nối với cơ sở dữ liệu.

Giao diện Connection là một factory của Statement, PreparedStatement và DatabaseMetaData, tức là, một đối tượng của Connection có thể được sử dụng để lấy đối tượng Statement và DatabaseMetaData. Connection Interface cung cấp nhiều phương thức để quản lý giao dịch như commit (), rollback (), setAutoCommit (), setTransactionIsolation (),…

## API Add/Edit/Delete

### Statement

**Statement interface trong java** cung cấp các phương thức để thực thi các câu lệnh truy vấn với cơ sở dữ liệu SQL. Statement interface là một nhà máy của ResultSet, tức là nó cung cấp phương thức để tạo ra đối tượng ResultSet.

ExecuteQuery: được sử dụng để thực hiện truy vấn SELECT. Nó trả về đối tượng của ResultSet.

ExecuteUpdate: được sử dụng để thực thi câu truy vấn được chỉ định, nó có thể là create, drop, insert, update, delete, ...

### Preparestatement

Giao diện **PreparedStatement trong java** là một sub-interface của Statement. Nó được sử dụng để thực hiện truy vấn tham số.  Hiệu suất của ứng dụng sẽ nhanh hơn nếu bạn sử dụng giao diện PreparedStatement vì truy vấn được biên dịch chỉ một lần.

## SQL Query

### Insert: INSERT INTO Product VALUES(1,”Apple”,10,20.5)

### Update: UPDATE Product SET price=21 WHERE ID in (1,100)

### Delete: DELETE Product WHERE ID =1

### Join: Select \* from Product join Category on CategoryID=ID

### Group By: Select \* from Product group by Name

# SERVLET/JSP

## SERVLET:

## 

### Servlet là một công nghệ được sử dụng để tạo ra ứng dụng web, là một API cung cấp các interface và lớp bao gồm các tài liệu, là một thành phần web được triển khai trên máy chủ để tạo ra trang web động

### Servlet được khởi tạo bằng cách gọi phương thức init ().

#### Phương thức init() được gọi chỉ một lần để khởi tạo servlet.

#### Servlet thường được tạo khi người dùng lần đầu tiên gọi một URL tương ứng với servlet, nhưng bạn cũng có thể chỉ định rằng servlet được tải khi máy chủ được khởi động lần đầu tiên.

### Phương thức servlet service() được gọi để xử lý yêu cầu của khách hàng.

#### Phương thức service() là phương thức chính để thực hiện nhiệm vụ thực tế. Bộ chứa servlet (tức là web server) gọi phương thức service() để xử lý các yêu cầu đến từ khách hàng (trình duyệt) và trả về kết quả.

#### Mỗi lần máy chủ nhận được một yêu cầu cho một servlet, máy chủ sẽ tạo ra một luồng mới và gọi phương thức service(). Phương thức service() kiểm tra kiểu yêu cầu HTTP (GET, POST, PUT, DELETE, v.v.) và gọi các phương thức doGet, doPost, doPut, doDelete, vv

### Servlet được hủy bằng cách gọi phương thức destroy().

Phương thức destroy() chỉ được gọi một lần vào cuối chu kỳ sống của một servlet. Phương thức này cho phép servlet đóng kết nối cơ sở dữ liệu, chặn các luồng chạy gầm, viết các cookie hoặc đếm số lượt truy cập và thực hiện các hoạt động dọn dẹp khác.

### Cuối cùng, servlet được thu thập bởi bộ sưu tập rác của JVM.

## HTPP Servlet, HTTPServletRequest, HTTPServletResponse

HttpServelt là một abstract class, nó nằm trong package ' **javax.servlet.http.HttpServlet** '. Để tạo một servlet, lớp phải extend lớp HttpServlet và ghi đè ít nhất một trong các phương thức của nó (doGet, doPost, doDelete, doPut). Lớp HttpServlet mở rộng lớp GenericServlet và thực hiện một giao diện có thể nối tiếp.

## Session

Session sẽ tạo một id phiên cho mỗi người dùng. Vùng chứa sử dụng id này để xác định người dùng cụ thể. Một đối tượng của HttpSession có thể được sử dụng để thực hiện hai tác vụ:

ràng buộc các đối tượng

xem và thao tác thông tin về một phiên, chẳng hạn như số nhận dạng phiên, thời gian tạo và thời gian truy cập lần cuối.

Session là các tệp phía máy chủ lưu trữ thông tin người dùng.

Session thúc khi người dùng đóng trình duyệt hoặc đăng xuất khỏi chương trình.

Session có thể lưu trữ một lượng thông tin không giới hạn.

Có một giới hạn bộ nhớ tối đa là 128 megabyte mà một tập lệnh có thể sử dụng cùng một lúc. Tuy nhiên, chúng tôi có thể tự do duy trì bao nhiêu dữ liệu tùy thích trong một phiên.

Dữ liệu được lưu ở định dạng mã hóa trong các phiên, một phiên có thể lưu trữ lượng dữ liệu không giới hạn.

## Cookie

Cookie là các tệp phía máy khách được lưu trữ trên máy tính cục bộ và chứa thông tin người dùng.

Cookie hết hạn sau thời gian tồn tại của người dùng đã chỉ định.

Nó chỉ có thể lưu trữ một lượng dữ liệu giới hạn.

Cookie chỉ có thể lưu trữ tối đa 4 KB dữ liệu trong trình duyệt.

Chúng tôi không cần thiết phải thực thi một chức năng để bắt các cookie bởi vì chúng được lưu trữ trên máy tính cục bộ.

Cookie được sử dụng để lưu trữ thông tin trong một tệp văn bản.

Cookie được lưu trữ trên một lượng dữ liệu giới hạn.

## JSP

JSP là viết tắt của JavaServer Pages là một công nghệ để phát triển các trang web động. JSP giúp các nhà phát triển chèn java code vào các trang HTML bằng cách sử dụng các thẻ JSP đặc biệt.

JSP là một kiểu Java servlet được thiết kế để tạo ra giao diện người dùng cho một ứng dụng Java web. Các nhà phát triển web viết các JSP như các tệp văn bản kết hợp mã HTML hoặc XHTML, các phần tử XML, các action và lệnh JSP.

Lợi ích của việc sử dụng JSP

### Là sự mở rộng, cải tiến của Servlet

 Chúng ta có thể sử dụng tất cả các tính năng của Servlet trong JSP. Ngoài ra, chúng ta có thể sử dụng các đối tượng tiềm ẩn, các thẻ được xác định trước, ngôn ngữ biểu đạt và các thẻ Tùy chỉnh trong JSP, điều này giúp cho việc phát triển JSP trở nên dễ dàng.

### JSP có thể được quản lý dễ dàng vì chúng ta có thể dễ dàng tách logic nghiệp vụ của mình với logic trình bày. Trong công nghệ Servlet, chúng tôi kết hợp logic nghiệp vụ của mình với logic trình bày.

### Nếu trang JSP được sửa đổi, chúng tôi không cần phải biên dịch lại và triển khai lại dự án. Mã Servlet cần được cập nhật và biên dịch lại nếu chúng ta phải thay đổi giao diện của ứng dụng.

### Vòng đời của JSP :

1. Translation of JSP Page
2. Compilation of JSP Page
3. Classloading (the classloader loads class file)
4. Instantiation (Object of the Generated Servlet is created).
5. Initialization ( the container invokes jspInit() method).
6. Request processing ( the container invokes \_jspService() method).
7. Destroy ( the container invokes jspDestroy() method).

# Spring

### Spring Boot

### Beans

### Context

### Spring Security vs OAuth2, JWT

### Spring Data

### Filter

### AOP

### Interceptor

### Scheduling

### Transactions

# Web Service

# Question

- Trình độ chuyên môn:

+ Nắm vững java core hay chưa:

- Các kiểu biến, các kiểu dữ liệu trong java

- Giá trị mặc định của biến instance đối với từng kiểu là bao nhiêu.

- Các class trong java được ngầm hiểu mặc định kế thừa từ lớp nào

- Các phần tử trong array có thể dùng kiểu dữ liệu nào, giá trị mặc định là bao nhiêu

- Khi 1 class tạo 1 contructor có tham số thì class đó còn contructor mặc định hay không

- OOP, các tính chất liên quan

- Đối với biến, method có những loại access modifier nào được áp dụng, đối với class là những loại nào

- Lớp con kế thừa được những gì từ class cha

- Phân biệt Overloading và Overriding

- Phân biệt Array, ArrayList

- Phân biệt ArrayList và LinkedList.

- Cơ chế HashMap, HashTable

- So sánh Comparable và Comparator, khi nào dùng cái nào?

- Nguyên lý của Stack, Queue là gì?

- Các vấn đề liên đến collection

+ JDBC & Servlet

- Để thực thi 1 câu truy vấn có những cách triển khai nào?

- Làm thế nào để tránh được SQL injection trong JDBC

- Transactions trong JDBC là gì, tại sao lại phải sử dụng Transactions, cách áp dụng

- Khi nào thì sử dụng JDBC Batch

- Những Annotations nào hay được sử dụng trong servlet

- Để định nghĩa 1 lớp là 1 controller trong servlet, có những cách nào

- Khác nhau giữa forward và redirect

- Trình bày 1 luồng xử lý từ khi gửi request đến khi forward hoặc redirect về JSP

+ Nếu ứng viên biết spring boot thì hỏi thêm 1 số liên quan về spring, hibernate(không nhiều)

- Spring bao gồm những module nào

- Bean là gì? bean được quản lý do đâu

- Các anotation thường dùng, mục đích của chúng

- Các cách triển khai spring boot security

- Cách xử lý Exceptions & Errors trong spring boot

- Các mối quan hệ giữa các đối tượng trong hibernate

- ...

+ Kiểm tra tư duy         -

Hỏi về cách xử lý 1 luồng nào đấy, ví dụ như luồng login logout,...

- 1 bài thuật toán liên quan đến Stack, Queue hoặc áp dùng Conllection

+ 1 vài câu hỏi khác

- Các vấn đề về git

- Quy trình làm việc

- Giải quyết các mâu thuẫn xảy ra trong dự án