**PV WEB**

# 1. Hỏi về giao tiếp giữa FrontEnd và BackEnd thế nào?

- Giao tiếp giữa FrontEnd và BackEnd trong một ứng dụng web thường được thực hiện thông qua API (Application Programming Interface).

- API cung cấp một bộ các yêu cầu (request) và phản hồi (response) giữa FrontEnd và BackEnd để cho phép chúng giao tiếp với nhau.

- Về cơ bản, FrontEnd gửi các yêu cầu đến BackEnd thông qua API, ví dụ như yêu cầu lấy dữ liệu, cập nhật dữ liệu hoặc thực hiện một hành động nhất định. BackEnd sau đó xử lý các yêu cầu này và trả về phản hồi tương ứng, chẳng hạn như trả về dữ liệu, xác nhận hoặc thông báo lỗi.

- Các yêu cầu và phản hồi này được định dạng theo một chuẩn nhất định, chẳng hạn như JSON (JavaScript Object Notation) hoặc XML (eXtensible Markup Language), để FrontEnd và BackEnd có thể hiểu được lẫn nhau.

# 2. (Trending) hỏi nhiều về JWT ?

**1/JWT (JSON Web Token)** là một tiêu chuẩn mở cho việc tạo và truyền tải các token được mã hóa bằng JSON giữa các bên. JWT được sử dụng phổ biến trong việc xác thực người dùng và cung cấp quyền truy cập trong các ứng dụng web và di động.

Một JWT bao gồm ba phần: **Header, Payload và Signature.**

**a/Header:** chứa thông tin về loại token (JWT) và thuật toán mã hóa được sử dụng (ví dụ như HS256, RS256, ...).

**b/Payload:** chứa các thông tin về người dùng hoặc các dữ liệu liên quan đến ứng dụng web. Payload được mã hóa bằng Base64 để tránh bị đọc trộm.

**c/Signature:** sử dụng để xác minh tính toàn vẹn của token và đảm bảo rằng token không bị sửa đổi khi được truyền tải giữa các bên.

Signature là một chuỗi được mã hoá bao gồm các thông tin header + payload

Chữ ký Signature trong JWT là một chuỗi được mã hóa bởi header, payload cùng với một chuỗi bí mật theo nguyên tắc sau:

HMACSHA256(

base64UrlEncode(header) + "." +

base64UrlEncode(payload),

secret)

**2/Quá trình hoạt động của JWT như sau:**

a/Người dùng đăng nhập bằng tên đăng nhập và mật khẩu của mình vào ứng dụng web.

b/Ứng dụng web sẽ tạo một JWT và trả về cho người dùng.

c/Người dùng sử dụng JWT để truy cập các tài nguyên được bảo vệ của ứng dụng web.

d/Khi người dùng gửi yêu cầu đến ứng dụng web, JWT sẽ được đính kèm trong tiêu đề của yêu cầu (thông qua Authorization header).

e/Ứng dụng web sẽ giải mã JWT, xác minh chữ ký và kiểm tra quyền truy cập của người dùng.

f/Nếu JWT hợp lệ, ứng dụng web sẽ trả về tài nguyên yêu cầu cho người dùng. Việc sử dụng JWT giúp giảm thiểu tải cho server bởi vì server không cần phải lưu trữ phiên làm việc (session) cho từng người dùng. Thay vào đó, JWT chứa các thông tin cần thiết để xác thực và cho phép người dùng truy cập các tài nguyên của ứng dụng web.

# 3. Hỏi về kỹ thuật cache trong BackEnd để giúp Phần mềm tối ưu hơn?

**Caching** (lưu trữ tạm thời) là một kỹ thuật quan trọng trong phát triển phần mềm để tối ưu hóa hiệu suất của hệ thống. Cache giúp giảm thiểu số lần truy vấn cơ sở dữ liệu, tăng tốc độ phản hồi và giảm tải cho máy chủ.

Trong **BackEnd**, có nhiều cách để thực hiện caching, tùy thuộc vào mục đích và kiến trúc hệ thống. Dưới đây là một số kỹ thuật caching phổ biến:

**1/Cache tầng đầu:** Các hệ thống tăng tốc độ thông qua việc cache các dữ liệu phổ biến, nhanh thay đổi và dễ tái sử dụng trong bộ nhớ đệm tại tầng đầu của ứng dụng. Ví dụ: memcached hoặc Redis.

**2/Cache trên đĩa:** Các hệ thống cũng có thể cache dữ liệu trên đĩa (disk-based caching) để giảm tải cho cơ sở dữ liệu và đảm bảo khả năng truy cập dữ liệu nhanh chóng. Ví dụ: sử dụng Redis hoặc caching dựa trên tệp.

**3/Caching tầng trung gian:** Các hệ thống sử dụng tầng cache trung gian (middleware caching) giữa client và server để giảm tải cho máy chủ.

- Các tầng trung gian cache sẽ truy cập vào cơ sở dữ liệu hoặc tầng đầu để lấy dữ liệu, sau đó cache lại kết quả để trả về cho client.

**4/Caching dựa trên HTTP:** Các hệ thống cũng có thể sử dụng cache dựa trên giao thức HTTP để giảm tải cho máy chủ.

- Các trình duyệt web sẽ cache lại các tài nguyên như HTML, CSS và JavaScript để tránh phải tải lại từ máy chủ mỗi khi trang web được truy cập.

**\***Để tối ưu hiệu suất của hệ thống, các kỹ thuật caching này có thể được kết hợp với nhau để giảm thiểu số lần truy vấn cơ sở dữ liệu và tăng tốc độ phản hồi của ứng dụng. - Tuy nhiên, việc sử dụng cache cũng cần cân nhắc kỹ lưỡng để đảm bảo tính nhất quán và độ chính xác của dữ liệu được trả về.

# 4.(Trending) Các thành phần mô hình MVC? Luồng đi của request trong mô hình MVC?

**1/MVC –** là mô hình kiến trúc phần mềm, đc chia làm 3 tầng:

+ **View** – là nơi hiển thị dữ liệu + tương tác với người dùng.

+ **Model** – là nơi biểu diễn dữ liệu (model), xử lí logic nghiệp vụ (service), tương tác với DB (repository).

+ **Controller** – đóng vai trò điều hướng, là nơi nhận request, xử lý request, và là cầu nối giữa Model với View.

**2/Dòng di chuyển của MVC:** Khi có 1 request gửi tới

+ controller sẽ nhận request

+ gửi dữ liệu sang model xử lý

+ model sẽ thực hiện kiểm tra (validate), tính toán dữ liệu

+ nếu cần có thể truy cập vào DB (ví dụ như update, create, delete,…), hoặc không (ví dụ như chuyển trang, click vào thẻ <a>,…)

+ model sẽ trả dữ liệu về cho controller + controller sẽ render dữ liệu lên view

+ controller sẽ trả view về cho người dùng + hiển thị thông tin cho người dùng.

# 5. Validate ở tầng nào ở mô hình MVC, giải thích?

- Trong mô hình MVC (Model-View-Controller) web, việc validate (kiểm tra tính hợp lệ) thông thường được thực hiện ở tầng Controller.

- Controller là tầng trung gian giữa View (giao diện người dùng) và Model (dữ liệu và xử lý logic).

Nó nhận thông tin từ View, xử lý thông tin này bằng cách sử dụng các phương thức của Model và sau đó trả về kết quả đến View.

Trong quá trình xử lý thông tin, Controller thường sẽ thực hiện các bước kiểm tra tính hợp lệ của dữ liệu nhập vào từ người dùng

- Tuy nhiên, việc validate có thể cũng được thực hiện ở tầng Model, tùy thuộc vào cấu trúc và thiết kế của ứng dụng. Trong trường hợp này, các phương thức trong Model sẽ kiểm tra tính hợp lệ của dữ liệu và trả về kết quả đến Controller để xử lý tiếp theo.

# 6. Xử lý nghiệp vụ ở tầng nào trong mô hình MVC? Model=> xử lí logic nghiệp vụ (service)

# 7.(Trending) Session & Cookie?

- **Session**: là phiên làm việc giữa client và server.

+ Thông thường, sử dụng session để lưu trữ thông tin đăng nhập, giỏ hàng, hoặc những dữ liệu mang tính chất tạm thời.

+ 1 session đc bắt đầu khi client thực hiện request lần đầu tiên đến server và kết thúc khi client dừng làm việc với server (đóng trình duyệt, hết time session, server bị tắt,…).

- **Cookie**: là những trang web do người dùng truy cập tạo ra, giúp trãi nghiệm trực tuyến dễ dàng hơn (ví dụ như ghi nhớ nội dung tùy chọn các dữ liệu đã nhập của trang web, thông tin lịch sử truy cập web => giúp cho website đề xuất những nội dung phù hợp với người dùng).

- **So sánh:**

1. **Giống:** đều lưu trữ thông tin tam thời.
2. **Khác:**
3. **Cookie**  được lưu trữ trên trình duyệt người dùng – **Session** dữ liệu được lưu trữ ở phía máy chủ.
4. **Cookei:**  tồn tại cho đến hết thời hạn – **Session:** tồn tại đến hết thời hạn, đóng kết nối với trình duyệt, ngừng làm việc vs sever.
5. **Cookie** truyền tải tất cả cookie đang có của trình duyện – **Session**  chỉ truyền tài coockie file đang có session id
6. Dữ liệu lưu ở cookie dễ dàng bị thay đổi vì lưu ở client - Dữ liệu ở Session lưu ở máy chủ nên khó bị thay đổi, an toàn bảo mật hơn
7. 1 **session** bị phụ thuộc vào **cookie** thông qua **session id**, ngược lại thì không.

(Nghĩa là, mỗi **session** sẽ có 1 **SessionID** riêng biệt. Khi kết thúc một **session** và bắt đầu một **session** mới thì ta sẽ có 1 **SessionID** khác. Giá trị của **SessionID** sẽ được lưu vào trong **cookie**, giúp cho **server** biết **session** này của **client** nào).

+ **JsessionId** khởi tạo khi vào **session** => phân biệt các **session** ở các **client** khác nhau.

# 8. Hỏi về các framework FrontEnd nào mà mình biết?

- React: Được phát triển bởi Facebook, React là một framework JavaScript phổ biến được sử dụng để xây dựng các ứng dụng web động.

- Angular: Angular là một framework JavaScript mã nguồn mở được phát triển bởi Google, được sử dụng để xây dựng các ứng dụng web động đơn trang (Single Page Application - SPA).

- Vue: Vue là một framework JavaScript mã nguồn mở, tương tự như Angular và React, được sử dụng để xây dựng các ứng dụng web động đơn trang (SPA) và các ứng dụng web đa trang.

- Bootstrap: Bootstrap là một framework CSS phổ biến được sử dụng để thiết kế các giao diện người dùng (UI) cho các ứng dụng web.

- Materialize: Materialize là một framework CSS được phát triển bởi Google, được sử dụng để thiết kế các giao diện người dùng (UI) cho các ứng dụng web. Foundation: Foundation là một framework CSS phổ biến được sử dụng để thiết kế các giao diện người dùng (UI) cho các ứng dụng web.

# 9. Hỏi về servlet, luồng thực thi của 1 request.

**Servlet**: + Servlet là một công nghệ được sử dụng để tạo ra ứng dụng web.

+ Servlet là một API cung cấp các interface và lớp bao gồm các tài liệu.

+ Servlet là một thành phần web được triển khai trên máy chủ để tạo ra trang web động.

Khi có 1 request gửi tới

1. Máy chủ Web nhận HTTP request

2. Máy chủ Web chuyển tiếp yêu cầu tới servlet container

3. Servlet được lấy ra một cách tự động và tải lên địa chỉ không gian của container, nếu nó nằm trong container.

4. Container gọi init () method của servlet để khởi tạo (được gọi một lần khi servlet được load lần đầu tiên)

5. Container gọi service () method của servlet để xử lý HTTP request, tức là đọc dữ liệu trong yêu cầu và hình thành một response

6. Web server trả lại kết quả động đúng với vị trí yêu cầu

# 10. UI/UX là gì

- **UI** là viết tắt của từ **User Interface** có nghĩa là giao diện người dùng.

1. Nó bao gồm các thành phần như đồ họa, màu sắc, font chữ, hình ảnh, các thành phần tương tác và bố cục tổng thể của sản phẩm.
2. UI thường được tập trung vào mặt ngoài của sản phẩm, đảm bảo rằng giao diện sẽ được thiết kế sao cho trực quan, thân thiện và dễ sử dụng.

- **UX** là viết tắt của từ **User Experience** có nghĩa là trải nghiệm người dùng.

1. Nó bao gồm việc tạo ra các trải nghiệm người dùng dễ sử dụng, thuận tiện, nhanh chóng và đáp ứng các mục tiêu cụ thể của người dùng.

UX thường được tập trung vào các yếu tố ẩn bên trong của sản phẩm, đảm bảo rằng sản phẩm sẽ đem lại

# 11. Phân biệt JS và Java

* + - JS là ngôn ngữ lập trình javascript chuyên lập trình front end
    - Java là ngôn ngữ lập trình java do sun microsystem phát triển chuyên lập trình back end
    - Java là ngôn ngữ lập trình tĩnh, hướng đối tượng, hoạt động trên nhiều nền tảng.
    - JavaScript là ngôn ngữ lập trình động (hay ngôn ngữ kịch bản - scripted language) được sử dụng để làm cho các trang web và ứng dụng trở nên sinh động. Java dựa trên lớp (class), còn JavaScript thì động

# 12. Authentication vs Authorization?

* + - **Authentication** (xác thực): là 1 cơ chế / hành động nhằm xác thực user có phải là người dùng của ứng dụng.
    - **Authorization** (phân quyền): là quá trình xác định xem user có quyền truy cập vào những trang nào.

# 13. Model trong MVC làm cái gì? Khi form submit, model làm gì, xuất hiện khi nào, đi đâu, về đâu?

* + - **Model** – là nơi biểu diễn dữ liệu (model), xử lí logic nghiệp vụ (service), tương tác với DB (repository).
    - Khi một form được submit, dữ liệu từ form sẽ được gửi đến server để xử lý.
    - Model sẽ xử lý các dữ liệu được gửi từ form, sau khi xử lý xong, model trả về kết quả cho controller, controller render dữ liệu, trả view về cho người dùng.

# 14. Session lưu ở đâu? Khi nào dùng session? Thế nào là phiên làm việc? Session có phải set đi set lại ko?

* **Session** được lưu ở server và phía client sẽ lưu session id trong cookie như chìa khoá để truy cập session lưu trên máy chủ
* Phiên làm việc là từ khi client gửi request đầu tiên lên server và đến khi client hết kết nối với sever
* Tắt trình duyệt -Log out
* Không tương tác với server sau 1 khoảng thời gian Session thường được sử dụng trong các ứng dụng web để lưu trữ thông tin liên quan đến người dùng như giỏ hàng thông tin đăng nhập lịch sử tìm kiếm...
* Session sẽ được tạo ra khi bắt đầu 1 phiên làm việc và trong 1 phiên đó session sẽ được cập nhập nếu cần thiết, session id lưu trên client sẽ không thay đổi

# 15. Authenticated nó lưu ở đâu? Được lưu trong database, session storage và local storage.

# 16. Cookie là gì? Cookie có lưu trên server ko? Ko lưu sao nó biết được cookie? Cookie lưu theo domain à? Server có thể xóa cookie của client ko?

* **Cookie**: là những trang web do người dùng truy cập tạo ra, giúp trãi nghiệm trực tuyến dễ dàng hơn.
* **Cookie** được lưu trữ ở client-side.
  + - Khi một trình duyệt yêu cầu một trang web, nó sẽ gửi cookie tương ứng với domain của trang web đó cho server. Server sử dụng thông tin trong cookie để cung cấp trải nghiệm tốt hơn cho người dùng.
    - Cookie được lưu trữ theo domain của trang web. Nếu một trang web sử dụng cookie, thì cookie đó sẽ chỉ được gửi đến server của domain đó.
    - Server không thể xóa cookie trực tiếp trên máy tính của người dùng.

# 17. Phân biệt MVC và Servlet?

**MVC (Model-View-Controller)** và **Servlet** là hai khái niệm khác nhau trong lập trình web.

**1.MVC** là một mô hình thiết kế phần mềm, đưa ra cách thức tổ chức các thành phần trong một ứng dụng web. Nó được chia thành 3 thành phần chính:

* + - **Model**: biểu diễn dữ liệu và logic xử lý dữ liệu.
    - **View**: hiển thị dữ liệu cho người dùng.
    - **Controller**: điều khiển luồng điều hướng, xử lý yêu cầu và tương tác giữa Model và View.

**2.Servlet** là một thành phần của Java Servlet API, cung cấp một cách để xử lý các yêu cầu web và tạo ra các phản hồi tương ứng.

- Servlet chịu trách nhiệm cho xử lý các yêu cầu HTTP, phân tích cú pháp và xử lý các thông tin yêu cầu và phản hồi.

- Servlet được triển khai bằng cách sử dụng Java và được sử dụng rộng rãi để tạo ra các ứng dụng web trên nền tảng Java.

Tóm lại, MVC và Servlet là hai khái niệm khác nhau trong lập trình web. MVC là một mô hình thiết kế phần mềm, trong khi Servlet là một thành phần của Java Servlet API được sử dụng để xử lý các yêu cầu web.

# 18. Khi viết code xong , muốn đưa code lên để người dùng chạy bằng cách nào ? Muốn dùng máy tính của mình làm máy chủ được không ? bằng cách nào?

**1.**Có nhiều lựa chọn cho việc triển khai ứng dụng web nhưng một trong những phổ biến nhất là sử dụng các dịch vụ máy chủ web như Amazon Web Services (AWS), Google Cloud Platform (GCP), hoặc Microsoft Azure.

- Ngoài ra, bạn cũng có thể triển khai ứng dụng trên một máy chủ web của riêng mình bằng cách sử dụng phần mềm máy chủ web như Apache, Nginx, hoặc IIS.

**2.**Để sử dụng máy tính của mình làm máy chủ, có thể cài đặt và cấu hình một phần mềm máy chủ web như Apache hoặc Nginx trên máy tính của mình.

* + - Sau đó, cần cấu hình router để cho phép các yêu cầu được gửi đến máy tính của mình thông qua địa chỉ IP công cộng của router.
    - Sau đó, bạn có thể truy cập vào ứng dụng của mình thông qua địa chỉ IP công cộng của router và cổng được cấu hình trên máy tính của bạn.

# 19. Tomcat là gì. JSP là gì. Cách sử dụng.

**1.Apache Tomcat** là một máy chủ ứng dụng web miễn phí và mã nguồn mở được phát triển bởi Apache Software Foundation.

- Tomcat là một trong những máy chủ ứng dụng web phổ biến nhất được sử dụng để triển khai các ứng dụng web Java, bao gồm cả các ứng dụng sử dụng JSP và Servlet.

**2.JSP (JavaServer Pages)** là một công nghệ cho phép bạn tạo ra các trang web động bằng cách sử dụng mã Java trong các tập tin HTML hoặc XML.

- JSP cho phép bạn tạo ra các trang web động được tạo ra dựa trên dữ liệu được lấy từ các nguồn khác nhau, chẳng hạn như cơ sở dữ liệu hoặc các tệp tin khác.

**3.**Để sử dụng Tomcat để triển khai các ứng dụng JSP, bạn cần phải cài đặt Tomcat trên máy tính của mình và cấu hình nó cho phù hợp với nhu cầu của bạn. Sau đó, bạn có thể tạo các tập tin JSP và triển khai chúng trên Tomcat để tạo ra các trang web động.

# 20. Message queue?

**- Message queue (MQ)** là một hệ thống phần mềm cho phép các ứng dụng gửi và nhận các tin nhắn một cách bất đồng bộ.

Điều này có nghĩa là các ứng dụng không cần phải kết nối trực tiếp với nhau để giao tiếp, mà thay vào đó chúng gửi và nhận các tin nhắn thông qua một hệ thống trung gian được gọi là message queue.

- **Message queue** cho phép các ứng dụng gửi và nhận các tin nhắn một cách độc lập với nhau, điều này mang lại nhiều lợi ích cho các ứng dụng như:

+ Giảm độ trễ trong quá trình giao tiếp giữa các ứng dụng.

+ Đảm bảo tính toàn vẹn và độ tin cậy của dữ liệu được gửi qua lại giữa các ứng dụng.

+ Tăng khả năng mở rộng của hệ thống bằng cách cho phép các ứng dụng được triển khai trên các máy tính khác nhau, tăng khả năng xử lý cho hệ thống.

# 21. Phân biệt sự khác nhau của Statement và PrepareStatement?

Statement và PreparedStatement là hai loại đối tượng trong Java được sử dụng để thực hiện các truy vấn SQL đến cơ sở dữ liệu. Tuy nhiên, chúng có những điểm khác biệt như sau:

* Sự tối ưu hóa và hiệu năng:

-PreparedStatement được sử dụng để thực hiện các truy vấn đã được thực thi trước đó, cho phép cơ sở dữ liệu tối ưu hóa để sử dụng lại các kết quả được lưu trữ trong bộ nhớ. Vì vậy, sử dụng PreparedStatement thường sẽ nhanh hơn và hiệu quả hơn so với Statement.

* Sự bảo mật:

-PreparedStatement giúp tránh được các cuộc tấn công SQL Injection bằng cách sử dụng tham số đã được truyền vào truy vấn. Bằng cách này, giá trị của các tham số sẽ được tự động làm sạch và ép kiểu đúng cách để đảm bảo an toàn.

* Sự linh hoạt:

-Statement cho phép thực hiện bất kỳ truy vấn SQL nào mà bạn muốn. Tuy nhiên, với PreparedStatement, bạn chỉ có thể sử dụng các truy vấn đã được chuẩn bị trước đó với các tham số có thể thay đổi.

* Độ phức tạp:

PreparedStatement phức tạp hơn Statement trong việc chuẩn bị truy vấn. Vì vậy, khi sử dụng PreparedStatement cho các truy vấn đơn giản, có thể không cần thiết và dẫn đến việc làm chậm hệ thống.

Tóm lại, PreparedStatement được ưu tiên hơn so với Statement trong hầu hết các trường hợp do nó cung cấp sự tối ưu hóa và bảo mật tốt hơn.

# 22.(Trending) Có hiểu SQL Injection là gì ko?

- SQL Injection là một kỹ thuật tấn công phổ biến trong lập trình web, trong đó kẻ tấn công sử dụng các chuỗi đầu vào bất hợp pháp để thực thi các truy vấn SQL không được mong đợi đến cơ sở dữ liệu của ứng dụng web.

- Khi các truy vấn SQL không được kiểm soát và không được sử dụng cách thức đúng đắn để tạo ra các truy vấn từ các tham số đầu vào, các hacker có thể chèn thêm mã độc hoặc lấy được thông tin bảo mật từ cơ sở dữ liệu của ứng dụng web.

- Để bảo vệ ứng dụng web khỏi SQL Injection, các lập trình viên cần sử dụng các kỹ thuật chống tấn công như sử dụng các câu lệnh Prepared Statement, kiểm tra và xác thực đầu vào của người dùng, giới hạn quyền truy cập vào cơ sở dữ liệu, và thực hiện các giải pháp bảo mật khác để đảm bảo an toàn cho ứng dụng của họ.

# 23. Nếu token lưu ở local storage bị đánh cắp thì ta xử lý như thế nào để người khác không lấy được dữ liệu

Để ngăn chặn việc này, có một số biện pháp mà bạn có thể áp dụng:

* **Sử dụng** **HttpOnly Cookies**: Sử dụng cookies với thuộc tính **HttpOnly** để ngăn chặn các tấn công Cross-Site Scripting (XSS) thông qua các kịch bản bên phía máy khách. Điều này giúp ngăn chặn việc truy cập đến cookie thông qua các kịch bản bên phía máy khách.
* **Sử dụng cơ chế xác thực kép (Two-factor authentication):** Sử dụng một phương thức xác thực thứ hai (như OTP, SMS, Authenticator,..) để xác nhận danh tính của người dùng khi đăng nhập. Điều này giúp ngăn chặn việc truy cập đến tài khoản người dùng một cách trái phép, ngay cả khi token của họ đã bị đánh cắp.
* **Sử dụng Refresh Tokens**: Sử dụng Refresh Tokens để tạo ra một token mới khi token cũ hết hạn. Khi token cũ hết hạn, người dùng sẽ cần phải sử dụng Refresh Token để lấy một token mới. Điều này giúp giảm thiểu rủi ro khi token bị đánh cắp.
* **Sử dụng mã hóa (Encryption):** Sử dụng mã hóa để mã hóa token và lưu trữ nó trong local storage. Việc này sẽ làm cho việc đánh cắp token trở nên khó khăn hơn, vì người tấn công cần phải có khả năng giải mã token.
* **Sử dụng HTTPS:** Sử dụng HTTPS để mã hóa thông tin được truyền tải giữa trình duyệt và máy chủ. Việc này giúp ngăn chặn tấn công Man-in-the-Middle (MITM) và giúp bảo vệ thông tin truyền tải.

# 24. Hibernate là gì?

- **Hibernate** - là một thư viện **ORM** mã nguồn mở giúp **LTV** viết ứng dụng **Java** có thể **mapping** các **Object** với hệ **CSDL**, và hỗ trợ thực hiện các khái niệm **OOP** với **CSDL**.

**Hibernate** có thể hỗ trợ tạo bảng và các dữ liệu thông qua ngôn ngữ **Java**.

# 25. JSP: Chuyển dữ liệu 1 từ một trang này tới trang kia?

Sử dụng phương thức GET: dữ liệu được gửi đi thông qua URL.

Sử dụng phương thức POST: dữ liệu được gửi đi thông qua form và phương thức POST.

# 26. Phương thức POST và GET khác nhau như thế nào.

**GET**

- Truyền thông tin thông qua URL.

- Data hiển thị trên URL.

- Kém bảo mật.

- Data chỉ gửi được dạng text, dung lượng gửi bị giới hạn (tối đa 4MB).

- Có thể được cache bởi trình duyệt.

- Dữ liệu gửi đi được lưu lại trong lịch sử web và có thể xem lại.

- Thực thi nhanh hơn vì những dữ liệu gửi đi luôn được webbrowser cached lại.

- Không gửi được nhị phân.

- Đối với dữ liệu ít thay đổi thường dùng phương thức GET để truy xuất và xử lý nhanh hơn.

**POST**

- Truyền thông tin thông qua HTTP header.

- Data không hiển thị trên URL.

- Bảo mật hơn.

- Data gửi được dạng text, hình ảnh, âm thanh,.. không giới hạn dung lượng gửi.

- Không cache bởi trình duyệt.

- Không được lưu lại trong lịch sử.

- Thực thi chậm hơn (vì phải mã hóa thông tin).

- Có thể gửi đc ASCII cũng như dữ liệu nhị phân.

- Đối với những dữ liệu luôn được thay đổi thì thường sử dụng phương thức POST

# 27. Session được lưu cụ thể ở đâu trên server?

Session được lưu trên server trong các vùng nhớ đặc biệt được gọi là session store hoặc session storage.

# 28. Cách triển khai login bằng JSP/Servlet và Spring Framework. Khác nhau giữa chúng. Mã hóa password.

**JSP/Servlet:**

* Sử dụng form để lấy thông tin đăng nhập từ người dùng, dữ liệu được gửi đến Servlet xử lý.
* Servlet kiểm tra thông tin đăng nhập của người dùng, nếu thông tin hợp lệ, Servlet tạo session để lưu thông tin người dùng đăng nhập và chuyển hướng đến trang chính.
* Nếu thông tin không hợp lệ, Servlet trả về trang đăng nhập với thông báo lỗi tương ứng.

**Spring Framework:**

* Sử dụng Spring Security để xác thực và phân quyền người dùng.
* Cấu hình Spring Security để định nghĩa các URL được bảo vệ, phương thức xác thực và phân quyền.
* Sử dụng form để lấy thông tin đăng nhập từ người dùng, Spring Security xử lý thông tin đăng nhập, nếu thông tin hợp lệ, tạo session để lưu thông tin người dùng đăng nhập và chuyển hướng đến trang chính.
* Nếu thông tin không hợp lệ, Spring Security trả về trang đăng nhập với thông báo lỗi tương ứng.

**Khác nhau giữa JSP/Servlet và Spring Framework:**

* Spring Framework cung cấp nhiều tính năng hỗ trợ xác thực và phân quyền người dùng, giúp phát triển ứng dụng an toàn và dễ bảo trì hơn.
* Với JSP/Servlet, phải viết code để kiểm tra thông tin đăng nhập của người dùng, trong khi Spring Security đã xây dựng sẵn tính năng này.
* Với Spring Framework, có thể cấu hình Spring Security để mã hóa mật khẩu người dùng để đảm bảo an toàn, trong khi với JSP/Servlet, phải tự mã hóa mật khẩu.

**Mã hóa password:**

* Trong JSP/Servlet: có thể sử dụng thuật toán mã hóa như MD5 hoặc SHA để mã hóa mật khẩu người dùng.
* Trong Spring Framework: có thể sử dụng các tính năng của Spring Security để mã hóa mật khẩu. Spring Security cung cấp các thuật toán mã hóa mật khẩu như BcryptPasswordEncoder

# 29. Phân quyền ở tầng nào trong MVC?

Phân quyền trong mô hình MVC (Model-View-Controller) được thực hiện ở tầng **Controller**.

Controller đóng vai trò trung gian giữa Model và View, xử lý các yêu cầu từ người dùng và cập nhật dữ liệu trong Model hoặc View.

**Khi người dùng truy cập vào một chức năng nào đó,**

* Controller sẽ xác định xem người dùng có quyền truy cập vào chức năng đó hay không.
* Nếu người dùng có quyền truy cập, Controller sẽ cho phép thực hiện yêu cầu và trả về kết quả tương ứng cho View.
* Nếu người dùng không có quyền truy cập, Controller sẽ trả về lỗi hoặc chuyển hướng người dùng đến trang khác.

**Phân quyền ở tầng Controller có nhiều lợi ích, bao gồm:**

* Dễ dàng kiểm soát quyền truy cập vào các chức năng của ứng dụng.
* Tách biệt logic xử lý yêu cầu từ người dùng và phân quyền, giúp mã nguồn dễ bảo trì và tái sử dụng.
* Tăng tính bảo mật cho ứng dụng, đảm bảo chỉ có người dùng có quyền mới được truy cập vào các chức năng quan trọng.

# 30. Nêu quá trình xử lý xử lý một ô input khi người dùng nhập dữ liệu

* Dữ liệu nhập vào được gửi đến máy chủ hoặc trang web.
* Máy chủ hoặc trang web sẽ tiếp nhận và xử lý dữ liệu nhập vào (kiểm tra tính hợp lệ của dữ liệu, chuyển đổi dữ liệu nhập vào thành định dạng phù hợp).
* Nếu dữ liệu không hợp lệ hoặc không đầy đủ, máy chủ hoặc trang web sẽ trả về thông báo lỗi hoặc yêu cầu người dùng nhập lại dữ liệu.
* Nếu dữ liệu hợp lệ và đầy đủ, máy chủ hoặc trang web sẽ tiếp tục xử lý dữ liệu đó để hiển thị kết quả cho người dùng.

# 31. Em dùng Maven hay Gradle? File pom.xml có chức năng gì? Đuôi .jar là gì?

Em dùng Gradle

**File pom.xml là file cấu hình dự án Maven**, được sử dụng để quản lý các thư viện và tài nguyên được sử dụng trong dự án. File này chứa thông tin về tên dự án, phiên bản, các phụ thuộc, các plugin được sử dụng, các tài nguyên, các tập tin cấu hình, và các mô-đun phụ thuộc khác của dự án.

Một số chức năng chính của file pom.xml bao gồm:

* Quản lý các thư viện phụ thuộc và phiên bản của chúng.
* Định nghĩa các bước xây dựng, đóng gói, triển khai và kiểm thử dự án.
* Định nghĩa các tài nguyên được sử dụng trong dự án như các file cấu hình, tài liệu hướng dẫn, và tài nguyên khác.

**Đuôi .jar là định dạng tập tin nén (archive)** dùng để đóng gói các file lớn thành một tập tin nhỏ hơn để dễ dàng phân phối và sử dụng.

Đây là định dạng tập tin phổ biến trong Java và được sử dụng để đóng gói các thư viện và ứng dụng Java.

Khi một ứng dụng được đóng gói thành file .jar, người dùng có thể dễ dàng triển khai và sử dụng ứng dụng mà không cần phải cài đặt các phụ thuộc khác.

# 32. Facebook, Google làm thế nào để tối ưu dung lượng khi có hàng tỉ lượt truy cập

* Sử dụng các máy chủ trung tâm dữ liệu (data center) để lưu trữ thông tin.
* Sử dụng các thuật toán tối ưu hóa và định tuyến (routing) để đưa thông tin tới người dùng một cách nhanh chóng và hiệu quả.
* Sử dụng các kỹ thuật nén dữ liệu (compression) để giảm dung lượng dữ liệu cần truyền tải trên mạng.
* Sử dụng các kỹ thuật cache để lưu trữ các dữ liệu phổ biến nhất.

# 33. Các cách bảo mật website?

1. Sử dụng HTTPS: Sử dụng giao thức HTTPS giúp mã hóa thông tin giữa trình duyệt và máy chủ, ngăn chặn tấn công giữa đường.
2. Cập nhật thường xuyên: Cập nhật các phần mềm và công nghệ liên quan đến website để đảm bảo sự an toàn và bảo mật.
3. Kiểm tra lỗ hổng bảo mật: Kiểm tra và tìm kiếm các lỗ hổng bảo mật của website để có thể giải quyết kịp thời tránh việc bị tấn công.
4. Sử dụng mật khẩu mạnh: Sử dụng mật khẩu mạnh và đổi mật khẩu thường xuyên để giữ an toàn thông tin tài khoản người dùng.
5. Tạo chính sách bảo mật: Tạo ra chính sách bảo mật cho website để người dùng có thể hiểu và tuân thủ các quy định bảo mật được đặt ra.
6. Xác thực người dùng: Sử dụng các hình thức xác thực người dùng như xác thực hai yếu tố hoặc mã xác thực để giảm thiểu nguy cơ bị tấn công từ các kẻ tấn công trực tuyến.
7. Sử dụng mã hóa dữ liệu: Sử dụng các phương pháp mã hóa dữ liệu như RSA, AES để giữ cho thông

# 34. Các thuật toán sort và search mà bạn biết ? Giải thích flow?

**- Thuật toán sort**:

+ Bubble Sort: So sánh 2 phần tử liền kề, hoán vị 2 phần tự đó nếu thỏa mãn điều kiện đang xét.

+ Selection Sort: Tìm phần tử bé nhất trong mảng, sau đó hoán vị phần tử bé nhất cho số đầu tiên của mảng, xét tiếp mảng từ vị trí 1 -> length-1, tiếp tục cho đến khi mảng đã được sắp xếp.

+ Insertion Sort: Chèn phần tử vào mảng con đã được sắp xếp, đảm bảo thứ tự sắp xếp sau khi chèn.

**- Thuật toán search:**

+ Linear Search: Duyệt qua tất cả các phần tử ở trong mảng, tìm thấy thì trả về index, không tìm thấy thì trả về -1.

+ Binary Search: Mảng đầu vào phải được sắp xếp tăng giảm.

1. Bước 1: kiểm tra phần cần tìm với phần tử giữa mảng.
2. Bước 2: nếu phần tử cần tìm = = phần tử giữa mảng => trả về index.
3. Bước 3: nếu phần tử cần tìm < phần tử giữa mảng => xét mảng con bên trái.
4. Bước 4: nếu phần tử cần tìm > phần tử giữa mảng => xét mảng con bên phải.
5. Bước 5: lặp lại Bước 1 -> Bước 4.

# 35. Login bằng post?

**1/Khái niệm:**

* Phương thức POST là một trong những phương thức HTTP được sử dụng để gửi dữ liệu từ client lên server. Thông thường, phương thức này được sử dụng để gửi thông tin được nhập bởi người dùng như tên đăng nhập và mật khẩu, địa chỉ email, hoặc bất kỳ thông tin nào khác liên quan đến việc tương tác với server.
* Việc sử dụng phương thức POST để gửi thông tin đăng nhập đảm bảo rằng thông tin này được mã hóa và gửi an toàn qua kết nối HTTPS đến server. Việc này giúp ngăn chặn kẻ tấn công có thể giả mạo thông tin đăng nhập của người dùng và truy cập vào tài khoản của họ.

**2/Quy trình hoạt động:**

* Người dùng nhập thông tin đăng nhập vào trang đăng nhập.
* Client gửi yêu cầu POST chứa thông tin đăng nhập đến server.
* Server xác minh thông tin đăng nhập.
* Server gửi phản hồi cho client để cấp quyền truy cập hoặc thông báo lỗi nếu thông tin đăng nhập không chính xác.

# 36. Nếu ko dùng JPA thì dùng JDBC như thế nào?

1. Tải và cài đặt driver JDBC cho cơ sở dữ liệu mà bạn sử dụng. Đối với MySQL, bạn có thể tải driver từ trang chủ của MySQL.
2. Sau khi cài đặt driver, import các package của JDBC vào project của bạn. Các package này thường bao gồm java.sql và javax.sql.
3. Tạo kết nối đến cơ sở dữ liệu bằng cách sử dụng class DriverManager và method getConnection.
4. Sau khi đã kết nối đến cơ sở dữ liệu, bạn có thể sử dụng các đối tượng của JDBC như Statement, PreparedStatement và ResultSet để thực hiện các thao tác với cơ sở dữ liệu.
5. Bạn cũng có thể sử dụng PreparedStatement để thực hiện các truy vấn động, hoặc CallableStatement để thực hiện các stored procedure trong cơ sở dữ liệu.

# 37. Hoạt động khi mình muốn lưu mật khẩu trên 1 trang web để lần sau ko phải nhập lại?

1. Sau khi đăng nhập thành công, trình duyệt của bạn sẽ hiển thị một cửa sổ yêu cầu bạn lưu mật khẩu. Nếu không có cửa sổ này, bạn có thể nhấp vào biểu tượng khóa hoặc "https" ở đầu thanh địa chỉ trang web để truy cập vào cài đặt bảo mật của trang web.
2. Chọn lưu mật khẩu và trình duyệt sẽ tự động lưu mật khẩu vào máy tính của bạn.
3. Lần sau khi bạn truy cập vào trang web đó, trình duyệt sẽ tự động điền mật khẩu cho bạn.

# 38. Các giao thức Http? GET, POST, HEAD, PUT, PATCH, DELETE, OPTIONS, CONNECT, và TRACE.

# 39. Copy cookie từ chrome thả bên firefox thì có vào facebook mà ko đăng nhập lại được không?

* Không.=> Cookie là các tệp văn bản nhỏ được lưu trữ trên máy tính của bạn, chứa thông tin về tài khoản đăng nhập của bạn và các hoạt động trên trang web. Mỗi trình duyệt có cách lưu trữ và mã hóa cookie khác nhau, do đó, không thể sử dụng cookie của Chrome trên Firefox và ngược lại. Ngoài ra, Facebook có các biện pháp bảo mật bổ sung như kiểm tra IP, thiết bị và địa điểm đăng nhập để bảo vệ tài khoản của người dùng.

# 40. (Trending) Thực hiện chức năng remember me?

* Chức năng "remember me" cho phép người dùng lưu thông tin đăng nhập của mình trên trang web, để họ không cần phải đăng nhập lại mỗi khi truy cập vào trang đó. Để thực hiện chức năng này, khi người dùng đăng nhập lần đầu tiên trên trang web, trang web sẽ lưu thông tin đăng nhập của người dùng vào cookie hoặc session trên máy tính của họ.
* Khi người dùng truy cập lại trang web, trang web sẽ tự động điền thông tin đăng nhập của người dùng vào các trường tương ứng để họ có thể đăng nhập nhanh chóng.
* Để thực hiện chức năng "remember me", trang web phải có một cơ chế lưu trữ thông tin đăng nhập của người dùng, thông thường là thông qua cookie hoặc session.

Trong Angular 11, chức năng "remember me" có thể được thực hiện bằng cách sử dụng Local Storage để lưu trữ thông tin đăng nhập của người dùng.VD:

onSubmit() {

// Kiểm tra xem người dùng đã chọn "remember me" hay không

if (this.rememberMe) {

// Lưu thông tin đăng nhập của người dùng vào Local Storage

localStorage.setItem('username', this.username);

localStorage.setItem('password', this.password);

} else {

// Nếu người dùng không chọn "remember me", xóa thông tin đăng nhập khỏi Local Storage

localStorage.removeItem('username');

localStorage.removeItem('password');

}

// Chuyển hướng đến trang chính sau khi đăng nhập thành công

this.router.navigate(['/home']); }

# 41. Cho biết về MVC - Mô tả hoạt động. Truy vấn thuộc về model hay controller theo mô hình MVC?

1. Mô hình MVC (Model-View-Controller) là một mô hình kiến trúc phần mềm được sử dụng trong lập trình máy tính để phân tách ứng dụng thành ba phần chính: Model, View và Controller.

* Model đại diện cho dữ liệu và các logic xử lý dữ liệu.
* View đại diện cho giao diện người dùng/
* Controller đại diện cho các logic điều khiển và xử lý sự kiện của người dùng.
* [Mô hình này giúp cho việc phát triển ứng dụng dễ dàng hơn bằng cách phân tách các thành phần chính của ứng dụng và giúp cho việc bảo trì và mở rộng ứng dụng dễ dàng hơn](https://bkhost.vn/blog/mo-hinh-mvc/)

1. Trong mô hình MVC (Truy vấn thuộc về Model hay Controller phụ thuộc vào loại truy vấn.

* Nếu truy vấn liên quan đến dữ liệu thì thuộc về Model, còn nếu truy vấn liên quan đến điều khiển và xử lý sự kiện của người dùng thì thuộc về Controller .

# 42. Làm sao để biết 1 người đã đăng nhập hệ thống hay chưa?

* Xem danh sách người dùng đang hoạt động: Trên các hệ điều hành như Windows hoặc Linux, bạn có thể sử dụng lệnh "who" để xem danh sách người dùng đang hoạt động trên hệ thống.
* Kiểm tra file log: Một số hệ thống lưu trữ thông tin đăng nhập của người dùng vào các file log. Bạn có thể kiểm tra các file log này để xác định người dùng nào đã đăng nhập vào hệ thống.
* Sử dụng công cụ quản lý hệ thống: Nếu bạn đang quản lý một hệ thống mạng hoặc máy chủ, bạn có thể sử dụng các công cụ quản lý hệ thống để kiểm tra người dùng đang hoạt động trên hệ thống.

# 43. Làm sao để request gửi đến đúng controller mình cần?

- Để conttroller bắt đúng request gửi từ front controller thì trong Spring Boot sử dụng anotation @RequestMapping (Get, Post, Delete, Put, Patch..) trong này chưa các thuộc tính để ánh xạ chính xác tới controller tương ứng.

+ value: URL pattern chỉ định địa chỉ sẽ đến đâu.

+ methob: Get, Post... thể hiện hành động gì đấy với URL đấy.

# 44. Muốn cập nhật chỉ mỗi trường tên với những nhân viên có id = 1 thì địa chỉ trên thanh url sẽ có dạng như thế nào? <https://example.com/employees/1?fields=name>

# 45. So sánh cookie và local storage

Cả hai đều là công nghệ lưu trữ dữ liệu trên trình duyệt web, tuy nhiên chúng có một số điểm khác nhau:

1. **Cookies**:

* Lưu trữ dưới dạng chuỗi văn bản, gồm cả tên và giá trị của cookie.
* Có thể được sử dụng để lưu trữ thông tin đăng nhập, trang web đã truy cập trước đó, các thiết lập ngôn ngữ và thông tin khác.
* Cookie được gửi đến máy chủ web mỗi khi người dùng yêu cầu trang web.
* Có hạn chế về kích thước (chỉ khoảng 4KB).
* Có thể được xóa hoặc hết hạn sau một khoảng thời gian nhất định.
* Có thể bị giới hạn hoặc chặn bởi các trình duyệt hoặc chương trình chặn quảng cáo.

1. **Local Storage:**

* Lưu trữ dưới dạng key-value pairs, có thể lưu trữ các đối tượng JavaScript dưới dạng chuỗi JSON.
* Không có hạn chế về kích thước (lưu trữ tối đa từ 5MB đến 10MB).
* Dữ liệu lưu trữ trong local storage sẽ không bị xóa khi trình duyệt web đóng lại hoặc máy tính bị khởi động lại.
* Local storage có thể được truy cập và cập nhật thông qua JavaScript.

# 46. Các thành phần của HTTP request?

Một khi kết nối được thiết lập bằng giao thức HTTP giữa máy khách và máy chủ, máy khách sẽ gửi một Request (yêu cầu) dưới dạng dữ liệu nhị phân đến máy chủ để yêu cầu truy cập các file hoặc thông tin cụ thể từ máy chủ.

Mỗi HTTP Request chứa ba thành phần: Request Line, Request Header và Request Body (tuỳ chọn).

● **Request Line:**

- Chỉ định phương thức (GET, POST, PUT, DELETE, …), cho biết máy chủ phải làm gì với thông tin hoặc tài nguyên;

- Chứa URL của Request được sử dụng để tìm tài nguyên trên máy chủ;

● **Request Header:** Bao gồm 0 hoặc nhiều Header.

Các Header được sử dụng để truyền đi nhiều thông tin hơn về Request, và từ việc sử dụng các Request Header, máy chủ biết cách xử lý thông tin mà máy khách yêu cầu

● **Request Body:** Đây là một thành phần tùy chọn của HTTP Request được sử dụng để gửi dữ liệu bổ sung đến máy chủ. Ví dụ: một loại file JSON hoặc XML.

Request Body sẽ gửi thông tin bổ sung theo yêu cầu của máy chủ để xử lý Request hiện tại đúng cách.

# 47. Nếu một ứng dụng FE chạy bất đồng bộ xong, muốn chuyển sang chế độ xử lý đồng bộ thì làm ntn?

-Có thể sử dụng các hàm như async/await hoặc Promise.all() để đợi cho các tác vụ bất đồng bộ hoàn thành.

-VD:async someFunction(): Promise<void> {

const result = await someObservable.toPromise(); // tiếp tục thực hiện các tác vụ đồng bộ với result }

# 48. So sánh sự khác nhau của Web application và Web Service?

**Web application:**

- Có giao diện.

- Tương tác giữa người dùng và ứng dụng, thông qua **form**, **link**, **image**, **button**,…

- Kết quả trả về cho người dùng là **view** hình ảnh, âm thanh, video,…

**Web service:**

- Không có giao diện.

- Tương tác giữa các ứng dụng với nhau, thông qua các phương thức **HTTP** (GET, POST, PUT, DELETE,…) và **HTTP status code**.

- Kết quả trả về cho người dùng chỉ là data **XML** hoặc **JSON** (khó đọc đối với người dùng).

# 49. Có bao nhiêu loại session, trong project dùng loại nào hoạt động như thế nào?

1. **Server-side sessions:** Session này được lưu trữ trên máy chủ và được quản lý bởi ứng dụng web. Mỗi lần người dùng tương tác với ứng dụng, các thông tin session sẽ được lưu trữ và xử lý bởi máy chủ. Server-side session có thể được lưu trữ bằng nhiều cách, bao gồm trong bộ nhớ, trên đĩa cứng hoặc trong cơ sở dữ liệu.
2. **Client-side sessions:** Session này được lưu trữ trên trình duyệt của người dùng, thường là bằng cách sử dụng cookies hoặc local storage. Khi người dùng tương tác với ứng dụng, các thông tin session sẽ được truyền qua lại giữa trình duyệt và máy chủ. Client-side session thường được sử dụng trong các ứng dụng web tĩnh hoặc trong các trường hợp không cần bảo mật cao.

# 50. Có bao nhiêu loại token?

Có nhiều loại token được sử dụng trong lập trình, một số phổ biến bao gồm:

1. **Access token:** Access token là một loại token được sử dụng để xác thực quyền truy cập của người dùng hoặc ứng dụng trong một hệ thống. Access token thường được cấp phát sau khi người dùng hoặc ứng dụng đăng nhập thành công và có thể được sử dụng để truy cập các tài nguyên được bảo vệ trong hệ thống.
2. **Refresh token:** Refresh token là một loại token được sử dụng để đổi mới access token khi nó hết hạn. Khi access token hết hạn, người dùng hoặc ứng dụng có thể sử dụng refresh token để lấy một access token mới mà không cần phải đăng nhập lại.
3. **JSON Web Token (JWT):** JWT là một loại token được sử dụng để truyền thông tin giữa các bên trong một hệ thống phân tán. JWT bao gồm ba phần: header, payload và signature. Header chứa thông tin về loại token và thuật toán mã hóa, payload chứa thông tin cần truyền và signature được sử dụng để xác thực tính toàn vẹn của token.
4. **CSRF token: CSRF (Cross-site Request Forgery)** token là một loại token được sử dụng để bảo vệ khỏi các cuộc tấn công CSRF. CSRF token được tạo ra bằng cách tạo ra một chuỗi ngẫu nhiên và gắn nó vào một yêu cầu của người dùng. Khi yêu cầu được gửi đi, server sẽ kiểm tra xem CSRF token có khớp với token được gắn vào yêu cầu hay không.
5. **ID token:** ID token là một loại token được sử dụng trong các hệ thống xác thực người dùng. ID token chứa thông tin về người dùng đã được xác thực, bao gồm tên, email và thông tin khác.

# 51. Patch dùng làm gì (Spring)?

Trong Spring Framework, phương thức PATCH được sử dụng để cập nhật một phần của một tài nguyên HTTP. Nó cho phép bạn chỉ định các trường cụ thể của tài nguyên mà bạn muốn cập nhật thay vì cập nhật toàn bộ tài nguyên. [Điều này giúp giảm thiểu lưu lượng mạng và giảm thiểu khả năng xung đột khi nhiều người dùng cùng cập nhật cùng một tài nguyên**1**](https://levunguyen.com/laptrinhspring/2020/04/23/cac-phuong-thuc-http-trong-lap-trinh-web/).

# 52. Phân biệt PUT với PATCH?

PUT và PATCH là hai phương thức HTTP được sử dụng để cập nhật tài nguyên HTTP. PUT được sử dụng để cập nhật toàn bộ tài nguyên, trong khi PATCH được sử dụng để cập nhật một phần của tài nguyên.

# 53. Phân biệt PUT với POST?

POST được sử dụng để tạo mới một tài nguyên và PUT được sử dụng để cập nhật một tài nguyên.

# 54. Trường hợp FE chạy vòng for gọi 1 triệu lần 1 api thì làm sao để server không bị quá tải?

Để tránh tình trạng server bị quá tải, bạn có thể sử dụng một số giải pháp như sau:

[Sử dụng **load balancer** để điều tiết tải lên các server**1**](https://toidicodedao.com/2017/08/01/thiet-ke-he-thong-trieu-nguoi-dung-high-scalability/).

[Tăng số lượng server nếu lượng người dùng tăng**1**](https://toidicodedao.com/2017/08/01/thiet-ke-he-thong-trieu-nguoi-dung-high-scalability/).

[Sử dụng **caching** để giảm số lượng request gửi đến server**2**](https://aws.amazon.com/vi/what-is/api/).

[Sử dụng **throttling** để giới hạn số lượng request gửi đến server**2**](https://aws.amazon.com/vi/what-is/api/).

[Sử dụng **queue** để xử lý các request một cách tuần tự**2**](https://aws.amazon.com/vi/what-is/api/).

# 55.(Trending)API là gì?

* API là viết tắt của "Application Programming Interface" - một giao diện lập trình ứng dụng cho phép các ứng dụng liên kết với nhau.Các ứng dụng này truy cập vào chức năng hoặc dịch vụ của một ứng dụng, hệ thống, thư viện hoặc công nghệ khác một cách chuẩn mực.
* Chúng cho phép các nhà phát triển tận dụng các tính năng của ứng dụng hoặc hệ thống khác, giảm thiểu sự phức tạp và tăng tính linh hoạt của việc phát triển.
* Có nhiều loại API, bao gồm:

1. Web API: Cung cấp các dịch vụ hoặc tài nguyên thông qua mạng Internet, cho phép các ứng dụng khác truy cập và sử dụng chúng.
2. Operating System API: Cung cấp các chức năng hoặc tài nguyên của hệ thống điều hành, cho phép các ứng dụng truy cập và sử dụng chúng.
3. Database API: Cung cấp các phương thức để truy cập và thao tác với cơ sở dữ liệu, cho phép các ứng dụng khác truy cập và thao tác dữ liệu một cách đơn giản.

# 56. Cách lấy dữ liệu API?

Để lấy dữ liệu từ một API, cần có một số bước cơ bản sau:

1. Xác định URL API: URL API là đường dẫn để truy cập tài nguyên hoặc dịch vụ từ API. Thông thường, API cung cấp tài liệu hướng dẫn để hỗ trợ các nhà phát triển xác định URL API.
2. Tạo yêu cầu HTTP: Tạo yêu cầu HTTP bằng cách sử dụng các phương thức HTTP như GET, POST, PUT, DELETE... Yêu cầu này sẽ gửi đến URL API và cung cấp thông tin cần thiết để truy cập tài nguyên hoặc dịch vụ từ API.
3. Nhận phản hồi từ API: Sau khi gửi yêu cầu HTTP, API sẽ trả về phản hồi cho yêu cầu đó. Phản hồi này có thể là một chuỗi JSON, XML hoặc các định dạng khác, phụ thuộc vào API mà bạn đang sử dụng.
4. Xử lý phản hồi: Cuối cùng, bạn cần xử lý phản hồi từ API để sử dụng các dữ liệu trả về hoặc hiển thị các thông tin cho người dùng

# 57. Để gọi API thì làm thế nào, cần truyền những gì?

Để gọi API, bạn cần truyền các thông tin cơ bản sau:

* Đường dẫn API (endpoint): đây là địa chỉ của API mà bạn muốn gọi.
* Phương thức (method): phương thức HTTP để gọi API, ví dụ GET, POST, PUT, DELETE...
* Tham số (parameter): các tham số cần thiết để gọi API. Tham số này thường được truyền thông qua URL hoặc body của yêu cầu HTTP.
* Headers: thông tin bổ sung gửi cùng yêu cầu HTTP như authorization token, user agent, content type..

# 58. Bảo mật cho API?

Bảo mật cho API là một vấn đề quan trọng trong quá trình phát triển ứng dụng web và đòi hỏi sự chú ý đặc biệt. Dưới đây là một số cách để bảo mật cho API:

* Xác thực (authentication): Để đảm bảo chỉ người dùng được phép truy cập và sử dụng API, bạn nên thực hiện xác thực đối với mỗi yêu cầu API. Các phương thức xác thực như OAuth, JSON Web Token (JWT), Basic Auth... được sử dụng rộng rãi để xác thực người dùng.
* Phân quyền (authorization): Bạn cần xác định và kiểm soát quyền truy cập của từng người dùng khi sử dụng API. Ví dụ như xác định quyền truy cập cho các nhóm người dùng khác nhau, hoặc xác định quyền truy cập cho các tài nguyên cụ thể.
* Sử dụng HTTPS: Đảm bảo API của bạn sử dụng giao thức HTTPS để mã hóa thông tin giao dịch giữa máy khách và máy chủ. Điều này sẽ giúp bảo mật thông tin giao dịch tránh khỏi các cuộc tấn công giả mạo dữ liệu hoặc theo dõi thông tin.
* Giới hạn số lượng yêu cầu: Bạn có thể giới hạn số lượng yêu cầu được thực hiện bởi một người dùng trong một khoảng thời gian nhất định để ngăn chặn tấn công từ hacker hoặc người dùng không đúng mục đích.
* Sử dụng mã hóa dữ liệu: Để đảm bảo tính toàn vẹn và bảo mật dữ liệu của bạn, bạn nên sử dụng mã hóa dữ liệu bằng các thuật toán mã hóa mạnh như AES, RSA...
* Cập nhật và bảo trì: Cập nhật thường xuyên cho API để đảm bảo nó hoạt động tốt và tránh lỗi bảo mật mới được tìm thấy. Đồng thời, bạn nên kiểm tra định kỳ bảo mật cho API để đảm bảo bảo mật tối đa.

# 59.Các bước gọi API cơ bản?

1. Xác định URL API: Trước khi gọi API, bạn cần xác định URL API, chứa thông tin về địa chỉ IP, tên miền, đường dẫn cần gọi.
2. Xác định phương thức HTTP: API có thể hỗ trợ nhiều phương thức HTTP như GET, POST, PUT, DELETE,...Bạn cần xác định phương thức HTTP phù hợp với API mà bạn muốn gọi.
3. Tạo yêu cầu HTTP: Sử dụng thư viện HTTP để tạo yêu cầu HTTP và truyền đường dẫn API, phương thức HTTP và các tham số (nếu có).
4. Gửi yêu cầu HTTP: Sau khi tạo yêu cầu HTTP, bạn gửi yêu cầu bằng cách sử dụng phương thức gửi yêu cầu HTTP tương ứng của thư viện HTTP.
5. Nhận phản hồi: Sau khi gửi yêu cầu, bạn nhận phản hồi từ API. Phản hồi có thể được trả về dưới dạng JSON, XML hoặc một định dạng khác.
6. Xử lý phản hồi: Bạn xử lý phản hồi từ API bằng cách sử dụng các phương thức xử lý của thư viện HTTP, bao gồm đọc dữ liệu phản hồi, kiểm tra mã trạng thái HTTP, xử lý lỗi (nếu có) và cập nhật giao diện người dùng (nếu cần).

# 60.Tính chất của reful?

RESTful là một kiến trúc dịch vụ web (web service) được thiết kế để sử dụng các giao thức và tiêu chuẩn web thông thường như HTTP, URI và XML/JSON để truyền tải dữ liệu giữa các hệ thống phân tán khác nhau. Các tính chất chính của RESTful trong web service bao gồm:

1. Statelessness (Không trạng thái): RESTful không giữ lại bất kỳ thông tin trạng thái nào về phiên làm việc (session) của client trên server. Mỗi request đều được xử lý độc lập và được truyền đầy đủ thông tin cần thiết để xử lý.
2. Client-server architecture (Kiến trúc máy khách-máy chủ): RESTful cho phép phân tách rõ ràng giữa các thành phần của client và server để giảm sự phụ thuộc và tăng khả năng mở rộng.
3. Các tài nguyên được xác định bằng URI (Uniform Resource Identifiers): RESTful sử dụng các URI để xác định các tài nguyên cần truy cập.
4. CRUD Operations (Create, Read, Update, Delete): RESTful hỗ trợ các hoạt động cơ bản trên tài nguyên như tạo mới, đọc, cập nhật, xoá.
5. Sử dụng các phương thức HTTP: RESTful sử dụng các phương thức HTTP như GET, POST, PUT, DELETE để thực hiện các thao tác CRUD trên tài nguyên.
6. Sử dụng các định dạng dữ liệu chuẩn: RESTful sử dụng các định dạng dữ liệu chuẩn như XML, JSON để truyền tải và đối tượng hóa dữ liệu giữa client và server.
7. Hệ thống được cache: RESTful cho phép client lưu trữ thông tin trên cache để giảm thiểu số lượng request tới server và tăng tốc độ xử lý.
8. Hỗ trợ các ngôn ngữ lập trình khác nhau: RESTful không phụ thuộc vào ngôn ngữ lập trình nào cụ thể, giúp cho các ứng dụng có thể giao tiếp với nhau dù được viết bằng các ngôn ngữ khác nhau.
9. Tính nhất quán (consistency): RESTful đảm bảo tính nhất quán trong các tài nguyên và hoạt động trên tài nguyên.