**AQUIREMENT J2EE/JEE**

**I. JEE: (Java Enterprise Edition – Java EE)**

- Là một Platform dùng để **phát triển các ứng dụng web** sử dụng **ngôn ngữ lập trình Java.**

- Là một **bộ sưu tập Java APIs** thuộc sở hữu của Oracle mà các lập trình viên phát triển phần mềm có thể sử dụng để lập trình ứng dụng phía máy chủ. (website)

***1. Tổng quan và kiến trúc:***

- J2EE cung cấp **Serverlet** và **Java Server Page (JSP) APIs**, chúng cung cấp các phương thức như tìm xem khách hàng gõ gì vào trường nhập liệu trong form hoặc thực hiện lưu trữ cookie trên trình duyệt của người dùng, xử lý việc nhận / xử lý / trả về kết quả theo yêu cầu đến từ khách hàng trên website.

* **Serverlet** giúp xử lý những yêu cầu và trả về cho máy khách qua giao thức HTTP, serverlet mở rộng chức năng của máy chủ qua mô hình request – response.
* **JSP** là công nghệ giúp nhúng đoạn mã Serverlet vào trong file văn bản tĩnh. Một tập tin JSP chứa 2 văn bản gồm dữ liệu tĩnh (HTML, XML, WML, …) và những đoạn mã Java dùng để xử lý động nội dung của trang web. Khi thực thi JSP được biên dịch thành Serverlet và thực hiện chức năng vốn có của một serverlet.

- Java EE cung cấp **Java Persistence API (JPA)** giúp dễ dàng ánh xạ dữ liệu được sử dụng trong một chương trình tới thông tin được lưu trữ trong các bảng và hàng của cơ sở dữ liệu.

- Việc tạo các web service hoặc các thành phần logic có khả năng mở rộng cao được đơn giản hóa thông qua việc sử dụng đặc tả **Enterprise JavaBeans (EJB).**

* **EJB** là công nghệ phía máy chủ dùng để đóng gói các lớp logic nghiệp vụ và hỗ trợ mạnh việc thực hiện các giao tác và bảo mật. EJB tích hợp 1 stack cho các …
* Hiện nay **EJB** được nhiều người sử dụng **Java Spring** thay thế.

*\* Ngoài 4 API đã nói ở trên,* ***có hơn 30 Java API*** *được bao gồm dưới dạng các công nghệ lõi của Java EE, với con số đó đã gần 50 ở phiên bản J2EE 8: JDBC, RMI, JMS, JNDI, JavaMail, …*

**- Các công nghệ lõi Java EE gồm:**

+ **HTTP client technology**: Công nghệ này sử dụng để xử lý các máy khách dựa trên **HTTP**, J2EE cũng bao gồm API Java cho lập trình **WebSocket**, API để xử lý **JSON**, API **JSF**, **Servlet** và **JSP Standard Tag Library (JSTL).**

* Web Socket: **WebSocket** là công nghệ hỗ trợ giao tiếp hai chiều giữa client và server bằng cách sử dụng một **TCP socket** thay cho **HTTP socket** để tạo một kết nối hiệu quả và ít tốn kém. **WebSocket** cho phép các kênh (máy tính) giao tiếp song song hai chiều và hiện đã được hỗ trợ trong nhiều trình duyệt.
  + Người ta thường dùng **WebSocket** thay cho **HTTP** cho những trường hợp yêu cầu realtime, ví dụ như hiển thị biểu đồ, web chat. Gói tin của WebSocket thì gọn nhẹ hơn so với HTTP.
  + Giao thức này thường là **ws://** cho chuẩn thông thường và ***wss://*** cho chuẩn secure.
* JSON: **JSON** là một kiểu định dạng dữ liệu trong đó sử dụng văn bản thuần tuý, định dạng **JSON** sử dụng các cặp key - value để dữ liệu sử dụng. [***https://www.json.org/json-en.html***](https://www.json.org/json-en.html)
* HTTP: (HyperText Transfer Protocol) là giao thức truyền tải siêu văn bản (HTML, XML, …) được sử dụng trong www dùng để tải dữ liệu giữa webserver đến các trình duyệt và ngược lại. Giao thức này chủ yếu sử dụng port 80.
  + Khi bạn gõ một từ tìm kiếm trên trình duyệt, lúc này trình duyệt sẽ gửi một yêu cầu tới server thông qua giao thức HTTP.
  + Các lỗi khi sử dụng trình duyệt web:
    - 404: Không tồn tại địa chỉ để có thể truy cập.
    - 401: Xảy ra khi truy cập vào một địa chỉ yêu cầu xác thực nhưng không vượt qua được sẽ xảy ra lỗi này.
    - 500: Lỗi thường do phía máy chủ bị lỗi nên không thể truy cập được.
* HTTPs: Là giao thức HTTP có sử dụng thêm SSL (Sercure Sockets Layer) để truyền mã hoá dữ liệu trong lúc truyền tải nhằm tăng tính an toàn trong việc truyền dữ liệu giữa webserver và trình duyệt web. HTTPs sử dụng port 433.

\* Sự khác nhau giữa HTTP và HTTPs:

- HTTPs: chủ yếu được sử dụng trong các trang web có giao dịch trực tuyến nhằm đảm bảo an toàn. Tránh những rủi ro bị lấy mất thông tin trong thẻ thanh toán. Ngoài ra, web sử dụng giao thức HTTPs là một tín hiệu để Google xếp hạng trực tuyến, làm nhiều trang web chuyển qua sử dụng giao thức này để được SEO tốt hơn. Tuy nhiên việc sử dụng giao thức này sẽ làm khiến cho việc truy cập website chậm hơn so với HTTP. Hiện nay Chrome đang có cảnh báo với các trang web có sử dụng HTTP và ưu tiên các trang web có sử dụng HTTPs. Để có thể sử dụng giao thức HTTPs, trang web của bạn cần có chứng chỉ SSL nếu không sẽ gặp tình trạng kết nối không bảo mật trên trình duyệt Chrome.

\* Chứng chỉ SSL: … ☺

* JSF: Java Server Face là một web framework MVC dùng để tập trung vào việc đơn giản hoá giao diện người dùng cho các ứng dụng J2EE và cho phép sử dụng lại các thành phần UI một cách dễ dàng.

 Một tập các hàm API để biểu diễn các thành phần UI (giao diện người dùng) và quản lý trạng thái của chúng, xử lý các sự kiện và kiểm tra dữ liệu đầu vào, quy định việc di chuyển trang (*page navigation*), hỗ trợ tính đa ngôn ngữ và hỗ trợ cho người sử dụng (*accessibility*)

 Một số thành phần (component) có sẵn.

 Hai thư viện chứa các thẻ tùy biến JSP (*JavaServer Pages custom tag*) để có thể biểu diễn một **JavaServer Faces interface** bên trong một trang JSP.

 Mô hình sự kiện ở phía máy chủ (server-side event model).

 Khả năng quản lý trạng thái

 Managed Beans (JavaBeans được tạo bằng [**dependency injection**](https://vi.wikipedia.org/w/index.php?title=Dependency_injection&action=edit&redlink=1) - xem thêm [Spring framework](https://vi.wikipedia.org/wiki/Spring_Framework)).

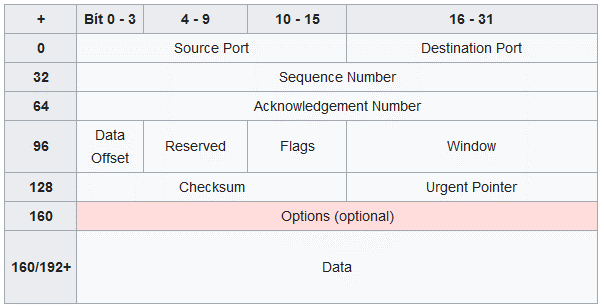
* JSP: **JSP** là công nghệ giúp nhúng đoạn mã Serverlet vào trong file văn bản tĩnh (các trang HTML, XML, …) thông qua các tag JSP. Một tập tin JSP chứa 2 văn bản gồm dữ liệu tĩnh (HTML, XML, WML, …) và những đoạn mã Java dùng để xử lý động nội dung của trang web. Khi thực thi JSP được biên dịch thành Serverlet và thực hiện chức năng vốn có của một serverlet.
* JSTL: JSP Standard Tag Library là thư viện thẻ chuẩn cung cấp các tag để kiểm soát hành vi trang, lặp đi lặp lại các lệnh điều khiển, các thẻ quốc tế hoá và các thẻ SQL.
  + Để sử dụng JSTL trong JSP của bạn thì bạn cần tải về thư viện JSTL cho Container Serverlet. Bạn cần đặt thư viện này thư mục WEB-INF/lib của project. Nếu project là Maven thì cần khai báo các thư viện này trong pom.xml
  + Các loại thẻ chính trong JSTL:
    - Các thẻ cơ bản (Core Tags): cung cấp hỗ trợ bộ lặp, các điều kiện logic, exception, url, forward, redirect. Để sử dụng được JSP Core Tags bạn cần đưa nó vào JSP như dưới đây:

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

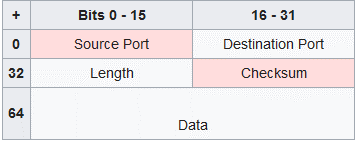
* + - Các thẻ định dạng và địa phương hoá (Formatting and Localization Tags): Những thẻ này cung cấp định dạng cho các con số và ngày tháng … bạn đưa nó vào trang JSP với cú pháp dưới đây:

<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>

* + - Các thẻ SQL: (SQL Tags) Cung cấp các hỗ trợ cho việc tương tác với CSDL như Oracle, MySQL, … Sử dụng các thẻ này để truy vấn dữ liệu, đưa nó vào trong JSP với cú pháp dưới đây: <%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql" %>
    - XML Tags: dùng để làm việc với các file XML, như phân tích cú pháp XML, … cú pháp: <%@ taglib uri="http://java.sun.com/jsp/jstl/xml" prefix="x" %>
    - Các thẻ hàm JSTL (Function Tags): Cung cấp các function mà ta có thể dễ dàng sử dụng để thực hiện với các toán tử thông dụng. Cú pháp: <%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>
* TCP/IP: Là một bộ các giao thức truyền thông, gồm hai giao thức chính là TCP và IP. Gồm 4 tầng: Ứng dụng (DNS, FTP, HTTP, POP3, …), Giao vận (TCP, UDP, … các giao thức định tuyến trên IP cũng có thể coi là một phần của tầng giao vận hoặc tầng mạng, …) , Tầng mạng (IP => IP v4 và IP v6), Giao thức ARP và RARP hoạt động bên dưới IP nhưng trên tầng liên kết có thể coi là khoảng trung gian giữa hai tầng), Tầng liên kết (Erthenet, Wifi, Token ring, ATM, …).
* TCP: (Tranmission Control Protocol) giao thức điều khiển truyền vận là một giao thức cốt lõi trong bộ giao thức TCP/IP, Các website được nối mạng có thể tạo ra các kết nối với nhau thông qua giao thức này. Từ đó chúng có thể trao đổi dữ liệu hoặc các gói tin. Giao thức này đảm bảo truyền dữ liệu tới nơi nhận một cách đánh tin cậy và đúng thứ tự.



* UDP: là một trong những giao thức cốt lõi của [giao thức TCP/IP](https://vi.wikipedia.org/wiki/TCP/IP). Dùng UDP, chương trình trên [mạng máy tính](https://vi.wikipedia.org/wiki/M%E1%BA%A1ng_m%C3%A1y_t%C3%ADnh) có thể gửi những dữ liệu ngắn được gọi là [*datagram*](https://vi.wikipedia.org/w/index.php?title=Datagram&action=edit&redlink=1) tới máy khác. UDP không cung cấp sự tin cậy và thứ tự truyền nhận mà [TCP](https://vi.wikipedia.org/wiki/TCP) làm; các gói dữ liệu có thể đến không đúng thứ tự hoặc bị mất mà không có thông báo. Tuy nhiên UDP nhanh và hiệu quả hơn đối với các mục tiêu như kích thước nhỏ và yêu cầu khắt khe về thời gian. Do bản chất không trạng thái của nó nên nó hữu dụng đối với việc trả lời các truy vấn nhỏ với số lượng lớn người yêu cầu.



* + Những ứng dụng phổ biến sử dụng UDP như DNS (Domain Name System) ứng dụng [streaming media](https://vi.wikipedia.org/wiki/Streaming_media), [Voice over IP](https://vi.wikipedia.org/wiki/VoIP), [Trivial File Transfer Protocol](https://vi.wikipedia.org/wiki/Trivial_File_Transfer_Protocol) (TFTP), và [game trực tuyến](https://vi.wikipedia.org/wiki/Game_tr%E1%BB%B1c_tuy%E1%BA%BFn).
* CGI: (Commond Gateway Interface): Lỗi thời rồi, không quan tâm ☺

+ **Công nghệ truy cập CSDL và tài nguyên**: Các công nghệ này sử dụng để tương tác với các hệ thống bên ngoài và hệ thống Backend. Java EE bao gồm JavaMail, Standard conenctor Architecture, Java Message Service (JMS) API và Java Transaction API (JTA).

**+ Công nghệ REST và Web Service:** Để giúp lập trình và triển khai các dịch vụ web dựa trên REST-, SOAP-, XML và JSON, các API Java cho Dịch vụ web RESTful (JAX-RS) và các dịch vụ web dựa trên XML (JAX-WS), cùng với các API cho thông báo XML và đăng ký XML (JAXR).

+ **Java EE security and container management:** Để triển khai bảo mật Java EE tùy chỉnh và quản lý các thùng chứa Java EE, các lập trình viên phần mềm có quyền truy cập vào Java Authorization Contract cho các Container và Java Authentication Service Provider Interface cho các container.

**- Kiến trúc: Gồm 3 tầng:**

- Client – Side Presentation: Tầng hiển thị cho người dùng, các thành phần chính ở tầng này gồm: JSP, Applet, HTML, CSS, Java Script, … đa số đều dùng để hiển thị tương tác với người dùng.

- Server – Side Presentation: Có chức năng tương tự như Client nhưng hiển thị ở mức server. Client và Server giao tiếp qua Web Service.

- Bussiness – Side Presentation: Là tầng xử lý vấn đề thuộc về logic, sử dụng thông qua các hàm thuộc EJB, EJB phụ trách xử lý, kết nối cơ sở dữ liệu JDBC, trả thông tin về cho người dùng.

- **JEE** hỗ trợ tối đa để có thể tạo ra một ứng dụng theo mô hình MVC.

- Web Server cho JEE: Glassfish, Tomcat.

**II. Java Serverlet:**

***1) Achitecture:***

-

***2) Life Circle:***

- Vòng đời của Serverlet gồm 5 giai đoạn như sau:

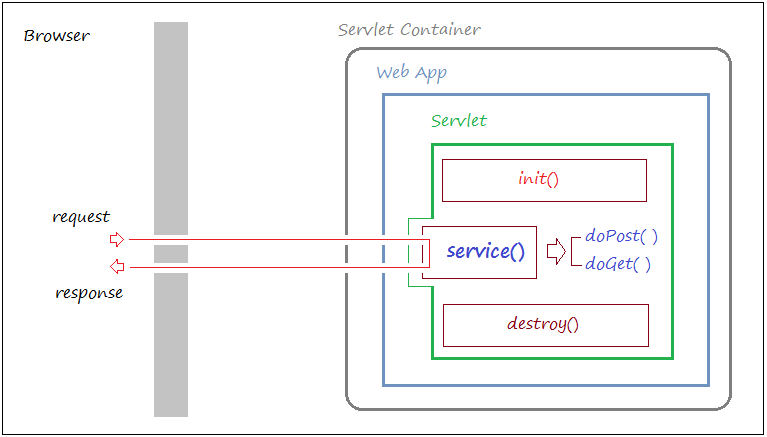
+ Tải lớp Serverlet vào bộ nhớ.

+ Tạo đối tượng Serverlet.

+ Gọi phương thức Init của Serverlet

+ Gọi phương thức Service của Serverlet.

+ Gọi phương thức Destroy của Serverlet.



- Bước 1 tới Bước 3 được thực thi một lần duy nhất, khi mà Serverlet được nạp lần đầu, mặc định các Serverlet không được tải lên (load) cho tới khi nhận được một đòi hỏi đầu tiên từ phía người dùng. Bạn có thể bắt buộc Serverlet Container tải các Serverlet khi nó khởi động.

- Bước 4 được thực thi nhiều lần, mỗi khi có đòi hỏi từ phía Client.

- Bước 5 được thực thi khi Serverlet Container gỡ bỏ (unload) một Serverlet ra.

- Khi có yêu cầu (request) tới Serverlet, Serverlet sẽ gọi phương thức service() để phục vụ yêu cầu của người dùng. Phương thức service() sẽ gọi một trong hai phương thức doGet() hoặc doPost(). Trong Serverlet của bạn, bạn cần override lại các phương thức này.

- Phương thức destroy() dùng để huỷ Serverlet, nó sẽ được gọi một lần duy nhất, khi bạn undeploy Serverlet hoặc tắt máy chủ Web.

- Khi một Serverlet được gọi tới, tuỳ tình huống mà hàm doGet() hoặc doPost() sẽ được gọi.

***3) Request and Responses:***

-

***4) Session Management:***

-

***5) Serverlet Filter:***

-

**III. Java Spring:**

***1.***

**III. JSP và JavaBean:**

**IV. JDBC:**

**V. MVC trong JEE:**

**VI. Web Service và EJB:**

***1. Web Service:***

***a) HTTP:***

***b) SOAP:***

***c) XML:***

**VII. Các kiến thức liên quan:**

***1. DAO (Data Access Object):***

-

***2. Java Spring:***

-

***3.CSS:***

-

***4. JavaScript:***

-

***5. AJAX request:***

***6. SOA:***

- Service Oriented Achitecture: Kiến trúc hướng dịch vụ.

***7. Best Practice:***

***8. JSON Web Token và REST API:***

**- REST**: REST là viết tắt của cụm từ **Representational State Transfer (đôi khi còn được viết là ReST)** là một kiểu kiến trúc được sử dụng trong việc giao tiếp giữa các máy tính (máy tính cá nhân và máy chủ của trang web) trong việc quản lý các tài nguyên trên internet. REST được sử dụng rất nhiều trong việc phát triển các ứng dụng Web Services sử dụng giao thức HTTP trong giao tiếp thông qua mạng internet. Các ứng dụng sử dụng kiến trúc REST này thì sẽ được gọi là ứng dụng phát triển theo kiểu RESTful. Đây là một tiêu chuẩn dùng trong thiết kế các API, quy đinh cách sử dụng các HTTP method. Được cụ thể như sau:

* + Khi **Create một *resource* thì sử dụng phương thức POST**
  + Khi **Update** **một *resource* thì sử dụng phương thức PUT**.
  + Khi **xoá một *resource* thì sử dụng phương thức DELETE**.
  + Khi **hiển thị một bài viết** cho độc giả họ đọc thì sử dụng phương thức **GET**, phương thức này trả về một resource hoặc một list các resource

- Đa số lập trình viên chọn sử dụng **JSON** là format chính thức cho **RESTful API.**

**- RESTful API** không sử dụng session và Cookie mà sử dụng một access\_token (JWT) với mỗi request, dữ liệu trả về thường có cấu trúc như sau:

{

    "data" : {

        "id": "1",

        "name": "TopDev"

    }

}

- Do là các HTTP method nên các Status code cũng giống nhau:

+ 00 OK – Trả về thành công cho những phương thức GET, PUT, PATCH hoặc DELETE.

+ 201 Created – Trả về khi một Resouce vừa được tạo thành công.

+ 204 No Content – Trả về khi Resource xoá thành công.

+ 304 Not Modified – Client có thể sử dụng dữ liệu cache.

+ 400 Bad Request – Request không hợp lệ

+ 401 Unauthorized – Request cần có auth.

+ 403 Forbidden – bị từ chối không cho phép.

+ 404 Not Found – Không tìm thấy resource từ URI

+ 405 Method Not Allowed – Phương thức không cho phép với user hiện tại.

+ 410 Gone – Resource không còn tồn tại, Version cũ đã không còn hỗ trợ.

+ 415 Unsupported Media Type – Không hỗ trợ kiểu Resource này.

+ 422 Unprocessable Entity – Dữ liệu không được xác thực

+ 429 Too Many Requests – Request bị từ chối do bị giới hạn

- JSON Web Token: là một tiêu chuẩn mở định nghĩa cách thức truyền tin an toàn giữa các thành viên và được xác thực dựa vào chữ ký của nó, phần chữ ký của JWT được mã hoá bằng HMAC hoặc RSA.

- JWT có các ưu điểm là kích thước nhỏ, có thể được truyền thông qua URL hoặc giao thức POST, hay nhét vào bên trong phần HTTP header. Một ưu điểm nữa đó là tính khép kín, khối hàng chứa toàn bộ những thông tin mà người dùng cần không cần phải truy vấn DB nhiều lần.

*\*Các trường hợp sử dụng JSON Web Token:*

- Authentication: Khi người dùng đã đăng nhập vào hệ thống thì những request tiếp theo sẽ chứa thêm mà JWT, cho phép người dùng truy cập vào các đường dẫn, dịc vụ, tài nguyên mà cần phải có sự cho phép của token đó.

Client application API

-------- -----------

| |

| GET /api/employees |

|----------------------------------------------------->|

| 403 Forbidden |

|<-----------------------------------------------------|

| |

| |

| POST /api/authenticate |

| { login: "paduvi", password: "chotoxautinh" } |

|----------------------------------------------------->|

| 200 Success |

| { token: "my.personal.token" } |

|<-----------------------------------------------------|

| |

| |

| GET /api/employees |

| Header { "Authorization: Bearer "my.personal.token" }|

|----------------------------------------------------->|

| 200 Success |

|<-----------------------------------------------------|

| |

- Trao đổi thông tin: JWT là phương thức không lời để truyền thông tin giữa các thành viên với nhau nhờ vào chữ ký của nó. Chữ ký được tạo bằng cách kết hợp cả phần header và page load nên có thể xác thực được là chữ ký có giả mạo hay không?

*\*Cấu trúc của JSON Web Token:*

- **Header**: Phần này sẽ khai báo kiểu chữ ký và thuật toán dùng để mã hoá token.

+ Ví dụ:

{

“alg”: “HS256”,

“typ”: “JWT”

}

=> Đoạn header này được khai báo rằng đối tượng là 1 JWT (phân biệt với JWS hay JWE), chữ ký của nó sử dụng thuật toán mã hoá HMAC SHA-256.

+ Đoạn header này sẽ được mã hoá base64url và thu được phần đầu tiên của JWT:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9

+ Có thể mô phỏng một hàm mã hoá base64url từ code JavaScript như sau:

function base64url(source) {

// Encode in classical base64

encodedSource = CryptoJS.enc.Base64.stringify(source);

// Remove padding equal characters

encodedSource = encodedSource.replace(/=+$/, '');

// Replace characters according to base64url specifications

encodedSource = encodedSource.replace(/\+/g, '-');

encodedSource = encodedSource.replace(/\//g, '\_');

return encodedSource;

}

=> Ở đây là dùng thư viện CryptoJS để có thể mã hoá basr 64 và thay thế các ký tự cần thiết để tạo thành mã hoá base64url.

Để có thể sử dụng hàm mã hoá này đầu vào cần là 1 mảng byte định dạng UTTF-8, có thể chuyển từ dạng chuỗi sang mảng byte bằng một hàm khác được cung cấp bởi CryptoJS:

var source = "Hello!";

// 48 65 6c 6c 6f 21

console.log(CryptoJS.enc.Utf8.parse(source).toString());

Cuối cùng ta thu được phần đầu tiên của header:

var header = {

"alg": "HS256",

"typ": "JWT"

};

var stringifiedHeader = CryptoJS.enc.Utf8.parse(JSON.stringify(header));

var encodedHeader = base64url(stringifiedHeader);

**- Payload (Claim):** Phần thứ 2 của token đó là payload, nơi chứa các nội dung của thông tin, thông tin truyền đi có thể là một mô tả thực tế hoặc bổ xung thêm thông tin cho phần header. Có 3 loại Payload là:

+ Reserved: Bao gồm những thông tin:

iss (issuer): tổ chức phát hành token.

sub (subject): chủ đề của token.

aud (audience): đối tượng sử dụng token.

exp: thời điểm token hết hạn.

nbf: token sec không hợp lệ trước thời điểm này.

iat: thời điểm token được phát hành/

jti: JWT ID

+ Public: Khoá nên được quy định trong IANA JSON Web Token Registry hoặc là 1 URI chứa không gian tên không bị trùng lặp.

Ví dụ: “https://www.techmaster.vn/jwt\_claims/is\_admin”: true

+ Private: Phần thông tin thêm dùng để truyền qua giữa các máy thành viên.

…