# JWT Authentication

## Tài liệu tham khảo

1. Website [gpcoder](https://gpcoder.com/5827-gioi-thieu-json-web-token-jwt/) (lý thuyết) – Tiếng Việt
2. Website [gpcoder](https://gpcoder.com/5883-rest-web-service-jwt-token-based-authentication-trong-jersey-2-x/) (mã nguồn ví dụ) – Tiếng Việt

## Tổng quan

1. Sử dụng thư viện **io.jsonwebtoken** để hỗ trợ tạo và verify JWT. Ngoài ra cũng có thể sử dụng thư viện **com.auth0**.
2. Luồng thực thi giống với Basic Authentication. Lần đầu client xác thực với **username** và **password**, nếu so khớp thành công server sẽ trả về 1 chuỗi **JWT**. Client đặt JWT trong **header AUTHORIZATION** và thực hiện các request sau đó.
3. Tại server sẽ có **1 end-point** thực thi **authentication** với username và password, so khớp và trả về JWT.
4. Tại server sẽ có **1 filter** xử lý chuỗi JWT (nếu có) được client gửi lên. Tiến hành phân tích chuỗi JWT và **lấy ra thông tin User** (JWT tự mô tả). Thực hiện gói User vào 1 lớp thực thi SecurityContext để dùng trong quá trình authorization.
5. Chúng ta sẽ tạo ra 1 class đặc biệt dùng để **tạo** và **verify JWT**. Xem lại lý thuyết để biết các thành phần của 1 JWT tại mục [1.1].

Các thông tin cơ bản để tạo ra JWT trong mã nguồn ví dụ bao gồm (Còn những thông tin khác nữa):

* + **Issued**: Thời điểm bắt đầu
  + **Expiration**: Thời điểm hết hạn
  + **signWith**: Key Secret
  + **Id**: ID JWT
  + **Các thông tin tự định nghĩa**

Quá trình **decode** **sẽ** **trả về Claims**, sau đó lấy ra các thông tin bằng **method get()** và **ép lại kiểu** mong muốn. Lưu ý: Các exception sau sẽ được ném ra trong quá trình decode vì vậy không cần phải tạo lại và xử lý nữa

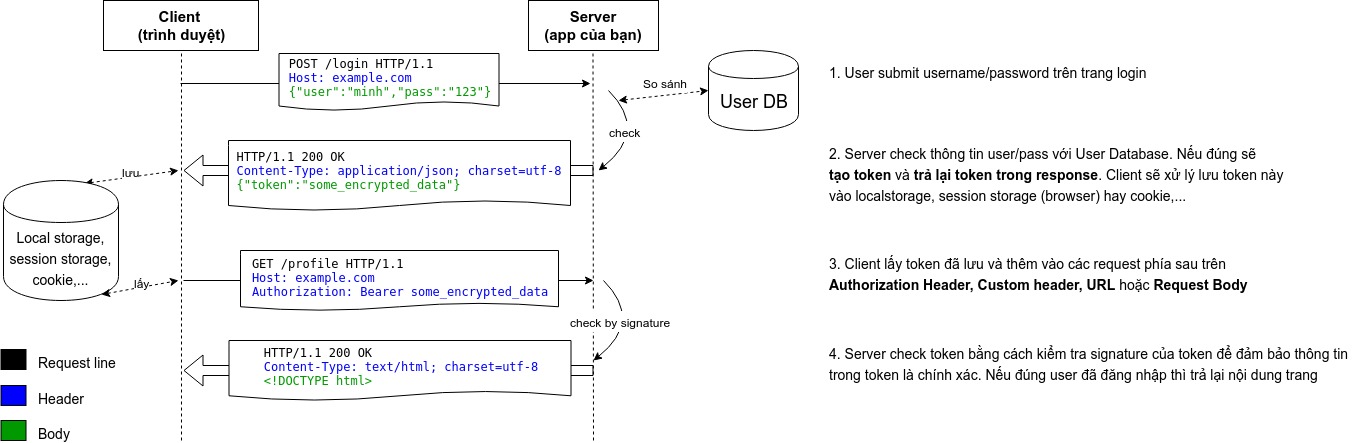
* **ExpiredJwtException**
* **UnsupportedJwtException**
* **MalformedJwtException**
* **SignatureException**
* **IllegalArgumentException**

1. Trong ví dụ, client sẽ sử dụng **form kèm username và password** gửi theo phương thức **post** để nhận chuỗi JWT. Sau đó kèm JWT này trong **header Authorization** của mỗi request, (nên sử dụng Filter để tự động thêm JWT).

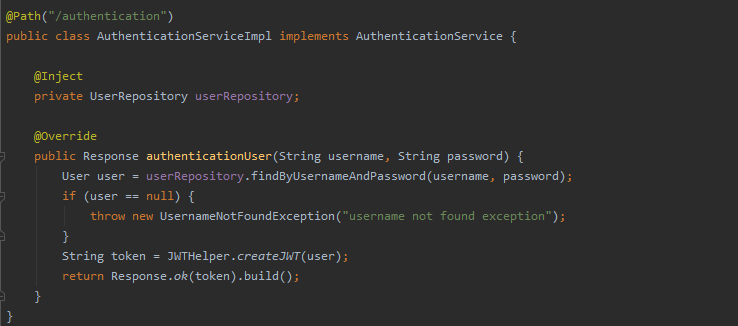
## Giải thích step-by-step

[2.1] – Thêm 3 thư viện này vào file pom.xml (**jjwt-api**, **jjwt-impl**, **jjwt-jackson**) hoặc copy từ file JWT.xml trong thư mục mvn.

[2.2] – Luồng thực hiện:



[2.3] – **end-point authentication** nhận **username** và **password**, **create JWT.**



[2.4] – Luồng xử lý bộ lọc cũng giống như Basic Authentication mà không có nhiều điểm khác biệt, sự khác biệt lớn nhất là quá trình lấy User từ JWT thay vì lấy username và password và tiến hành so khớp như trong Basic Authentication. Mã nguồn filter này tham khảo [tại đây](https://github.com/ytbhoanghai/Jersey-JAXRS/blob/master/REST/Jersey%20-%202.x/TokenAuthentication/src/main/java/com/nguyen/filter/AuthenticationFilter.java).

[2.5] – Xem lại lý thuyết về cấu trúc của 1 JWT (nếu cần) tại mục [1.1]. Để tạo và verify 1 JWT, tham khảo lớp JWTHelper [tại đây](https://github.com/ytbhoanghai/Jersey-JAXRS/blob/master/REST/Jersey%20-%202.x/TokenAuthentication/src/main/java/com/nguyen/helper/JWTHelper.java).

[2.6] – Mã nguồn ví dụ không kèm theo phần chương trình client. Tham khảo mã nguồn client [tại đây](https://gpcoder.com/5883-rest-web-service-jwt-token-based-authentication-trong-jersey-2-x/).

## Code ví dụ

[TokenAuthentication](https://github.com/ytbhoanghai/Jersey-JAXRS/tree/master/REST/Jersey%20-%202.x/TokenAuthentication) giành cho phía Server.