FPT Education

FPT POLYTECHNIC

cdio™

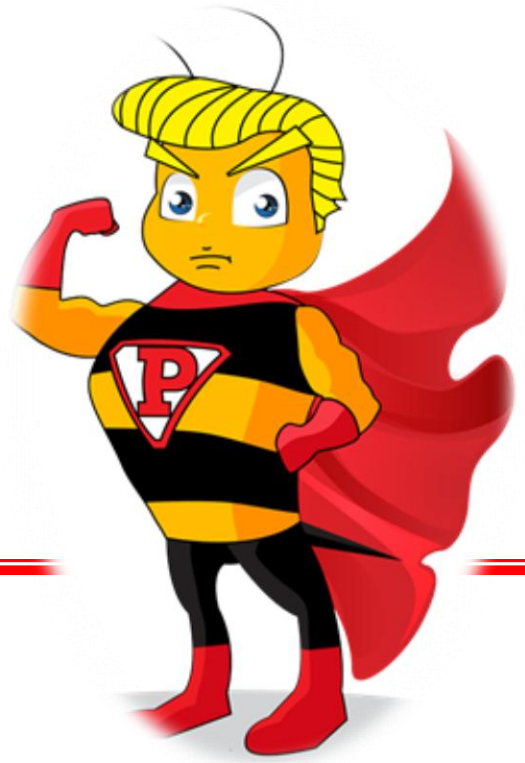
Conceive Design Implement Operate

LOGIN FROM GOOGLE & JWT

GIẢNG VIÊN: NGUYỄN NGHIỆM

AGENDA

- Giới thiệu JWT
- Triển khai JWT với Spring Boot
- Using JWT trong Postman
- Using JWT trong Axios



JWT

JSON Web Token (JWT)

- JSON Web Token (JWT) là một tiêu chuẩn mở định nghĩa cách thức thu gọn và độc lập để truyền thông tin an toàn giữa các bên dưới dạng đối tượng JSON. Thông tin này có thể được xác minh và tin cậy vì nó được ký kỹ thuật số.
- JWT là thông tin đã được mã hóa, bên trong nó có thể chứa dữ liệu bảo mật liên quan giữa các bên nhưng chúng ta chỉ quan tâm vào cấu trúc chính còn phần dữ liệu mở rộng tùy thuộc vào thỏa thuận của các bên liên quan.

KHI NÀO BẠN NÊN SỬ DỤNG JSON WEB TOKENS?

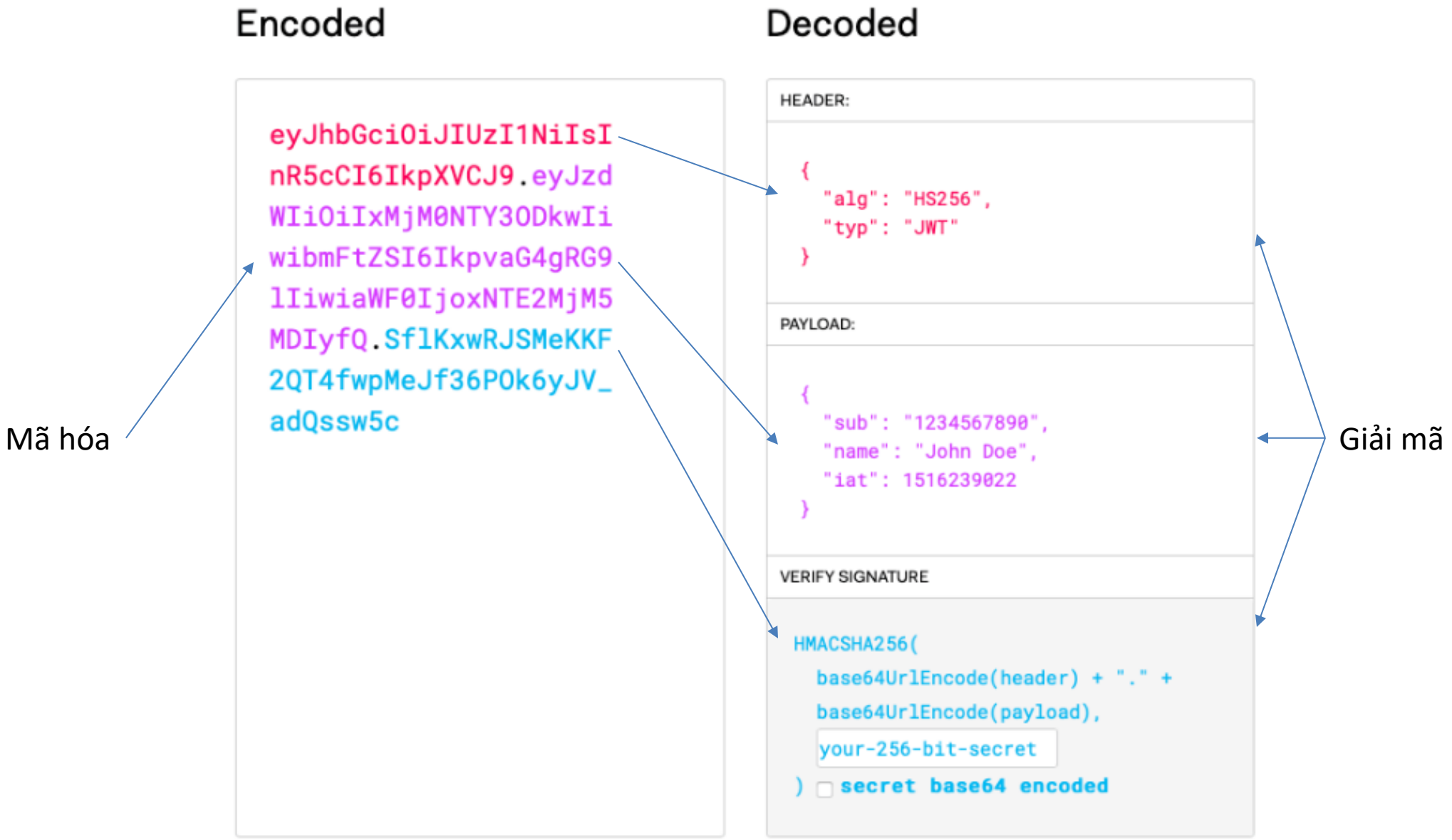
- Sau đây là một số trường hợp mà JSON Web Token hữu ích:
- **Authorization** : Đây là kịch bản phổ biến nhất khi sử dụng JWT. Sau khi người dùng đăng nhập, mỗi yêu cầu tiếp theo sẽ bao gồm JWT, cho phép người dùng truy cập các tài nguyên được phép bằng mã thông báo JWT. Single Sign On (SSO) là tính năng sử dụng rộng rãi JWT hiện nay, vì chi phí thấp và khả năng dễ dàng sử dụng trên nhiều domain khác nhau.
- **Information Exchange**: Dữ liệu được mã hóa và có sử dụng chữ ký số nên tạo sự tin cậy cao.

Cấu trúc JWT

- Ở dạng nhỏ gọn, JSON Web Tokens bao gồm ba phần được phân tách bằng dấu chấm (.), đó là:
  - Tiêu đề (Header)
  - Tải trọng (Payload)
  - Chữ ký (Signature)
- Do đó, JWT thường trông như sau.
  - <<header>>.<<payload>>.<<signature>>
- Sau đây là ví dụ về JWT có tiêu đề và nội dung trước được mã hóa và được ký bằng một mã bí mật.

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4
gRG9lIiwiaXNTb2NpYWwiOnRydWV9.
4pcPyMD09olPSyXnrXCjTwXyr4BsezdI1AVTmud2fU4
```

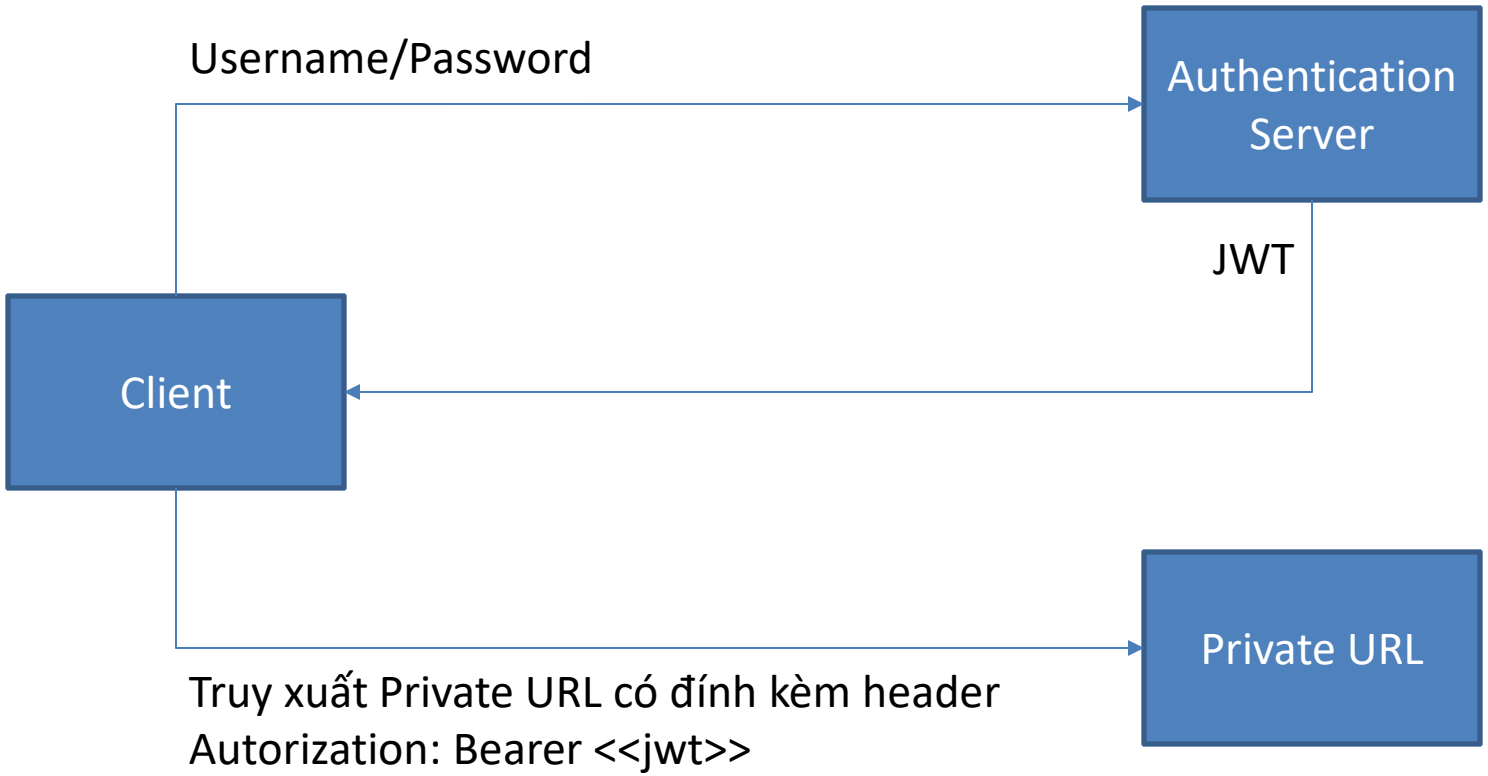
Cấu trúc JWT


Encoded
Decoded
eyJhbGciOiJIUzI1NiIsI
nR5cCI6IkpXVCJ9.eyJzd
WIiOiIxMjM0NTY3ODkwIi
wibmFtZSI6IkpvaG4gRG9
lIiwiaWF0IjoxNTE2MjM5
MDIyfQ.SflKxwRJSMeKKF
2QT4fwpMeJf36POk6yJV_
adQssw5c
HEADER:
{
"alg": "HS256",
"typ": "JWT"
}
PAYLOAD:
{
"sub": "1234567890",
"name": "John Doe",
"iat": 1516239022
}
VERIFY SIGNATURE
HMACSHA256(
base64UrlEncode(header) + "." +
base64UrlEncode(payload),
your-256-bit-secret
) secret base64 encoded
Mã hóa
Giải mã

JSON Web Token hoạt động như thế nào?

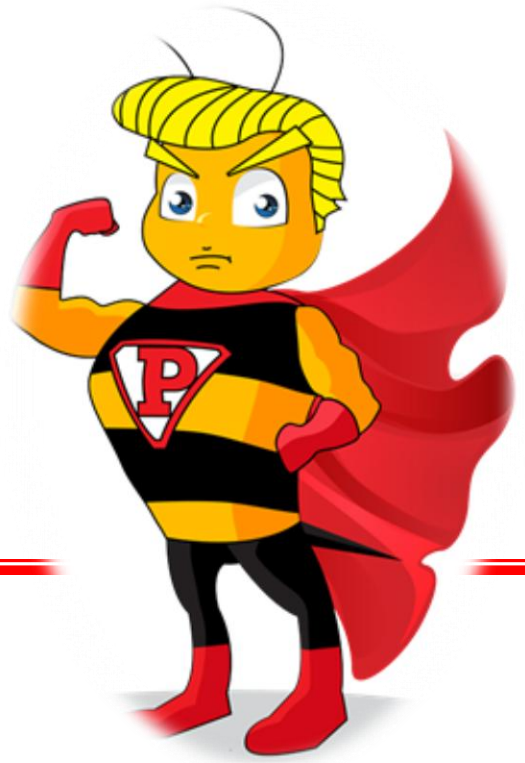
- Trong xác thực, khi người dùng đăng nhập thành công bằng thông tin xác thực của họ, một JSON Web Token sẽ được trả về. Vì các token là thông tin xác thực, nên phải hết sức cẩn thận để ngăn ngừa các vấn đề về bảo mật.
  - Không nên giữ các token lâu hơn mức cần thiết (thời hạn hiệu lực).
  - Không nên lưu trữ dữ liệu phiên nhạy cảm trong localStorage, sessionStorage của trình duyệt vì thiếu tính bảo mật.
- Bất cứ khi nào người dùng muốn truy cập một tài nguyên được bảo vệ phải gửi kèm JWT, thường đặt trong header **Authorization** sử dụng lược đồ **Bearer**. Nội dung của header trông giống như:
  - Authorization: Bearer <<token>>

JSON Web Token hoạt động như thế nào?



Thực hiện xác thực và tạo (create) JWT gửi về client

Kiểm tra (validate) JWT nếu hợp lệ thì cho truy xuất tài nguyên được bảo vệ



Triển khai JWT

CASE STUDY

```
@RestController
public class MyRestApi {
	@GetMapping({"/poly/url0", "/"})
	public Object method0() {
		return Map.of("/poly/url0", "method0()");
	}
	@GetMapping("/poly/url1")
	public Object method1() {
		return Map.of("/poly/url1", "method1()");
	}
	// @/poly/url2 => {"/poly/url2":"method2()"}
	// @/poly/url2 => {"/poly/url3":"method3()"}
	// @/poly/url2 => {"/poly/url4":"method4()"}
	// @/poly/url2 => {"/poly/url2":"method2()"}
}
```

Tài nguyên

Url	Role
@/poly/url1	Authenticated
@/poly/url2	USER
@/poly/url3	ADMIN
@/poly/url4	ADMIN và USER
Any Request	Permit All

Phân quyền

DEPENDENCY

```
<dependency>
        <groupId>io.jsonwebtoken</groupId>
        <artifactId>jjwt-api</artifactId>
        <version>0.11.5</version>
</dependency>
<dependency>
        <groupId>io.jsonwebtoken</groupId>
        <artifactId>jjwt-impl</artifactId>
        <version>0.11.5</version>
</dependency>
<dependency>
        <groupId>io.jsonwebtoken</groupId>
        <artifactId>jjwt-jackson</artifactId>
        <version>0.11.5</version>
</dependency>
```

JWT API

- Jwt, JwtBuilder
  - Jwt chứa dữ liệu token (header, body, signature)
  - JwtBuilder cung cấp các phương thức tạo Jwt
- JwtParser, JwtParserBuilder
  - JwtParser cung cấp các phương thức bóc tách các thành phần header, body và signature
  - JwtParserBuilder cấp cấp các phương thức chuyển đổi token sang JwtParser
- Claims (extends Map)
  - Chứa dữ liệu của body

SPRING BEAN JWTSERVICE

```
@Service
public class JwtService {
    public String create(UserDetails user, int expirySeconds) {...}
    public Claims getBody(String token) {...}
    public boolean validate(Claims claims) {...}
    private Key getSigningKey() {...}
}
```

- create(): Tạo JWT (JSON Web Token)
  - user là UserDetails chứa thông tin để tạo token
  - expirySeconds là thời hạn có hiệu lực (tính bằng giây)
- getBody(): Bóc tách phần body từ JWT
- validate(): Xác minh thời gian hiệu lực
  - claims là body của JWT
- getSigningKey(): Tạo chữ ký số để ký và xác minh JWT

SPRING BEAN JWTSERVICE

```
public String create(UserDetails user, int expirySeconds) {
        long now = System.currentTimeMillis();
        return Jwts.builder()
                .setClaims(Map.of("name", "Poly")) // bổ sung thêm nội dung khác vào body
                .setSubject(user.getUsername()) // thêm thuộc tính sub vào body
                .setIssuedAt(new Date(now)) // thêm thuộc tính iat vào body
                .setExpiration(new Date(now + 1000 * expirySeconds)) // thêm thuộc tính exp vào body
                .signWith(this.getSigningKey(), SignatureAlgorithm.HS256) // ký số
                .compact();
}
```

- Body (Claims) được tạo ra: {***name***="Poly", ***sub***=?, ***iat***=?, ***exp***=?}
- Chú ý: setClaims(Map<String, String>) cho phép bổ sung dữ liệu mở rộng tùy ý

Spring Bean JwtService

```
public Claims getBody(String token) {
        return Jwts.parserBuilder()
            .setSigningKey(this.getSigningKey()) // cung cấp chữ ký số
            .build()
            .parseClaimsJws(token) // chuyển đổi chuỗi token sang đối tượng jwt
            .getBody(); // bóc tách body {name=?, iat=?, exp=?, sub=?}
}
public boolean validate(Claims claims) {
    return claims.getExpiration().after(new Date());
}
private Key getSigningKey() {
        String secret = "0123456789.0123456789.0123456789"; // HS256 >= 32 byte
        return Keys.hmacShaKeyFor(secret.getBytes());
}
```

JWT Authentication Filter

```
@Service
public class JwtAuthFilter extends OncePerRequestFilter {
    @Autowired
    JwtService jwtService;

    @Autowired
    UserDetailsService userService;

    @Override
    protected void doFilterInternal(HttpServletRequest request,
                                    HttpServletResponse response, FilterChain filterChain)
            throws ServletException, IOException {
        // ...tạo và thay đổi đối tượng Authentication nếu JWT hợp lệ...
        filterChain.doFilter(request, response);
    }
}
```

JWT Authentication Filter

```
if (SecurityContextHolder.getContext().getAuthentication() == null) { // Chưa đăng nhập
        var authorization = request.getHeader("Authorization"); // Bearer <<token>>
        if (authorization != null && authorization.startsWith("Bearer ")) {
                var token = authorization.substring(7).trim(); // <<token>>
                var claims = jwtService.getBody(token);
                if (jwtService.validate(claims)) {
                        var username = claims.getSubject();
                        UserDetails user = userService.loadUserByUsername(username);
                        var authentication =
                                new UsernamePasswordAuthenticationToken(user, null, user.getAuthorities());
                        SecurityContextHolder.getContext().setAuthentication(authentication);
                }
        }
}
```

CẤU HÌNH SECURITY

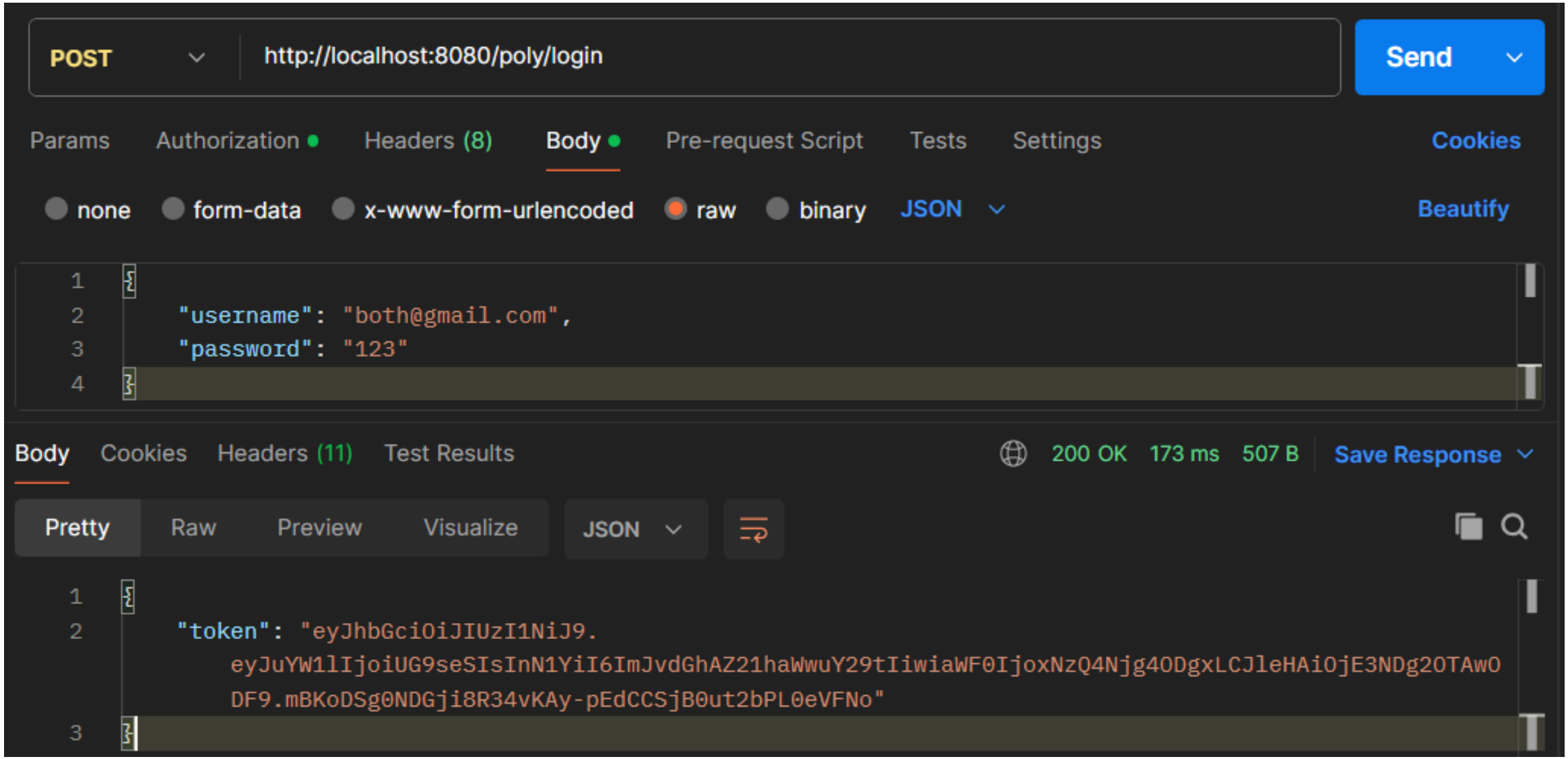
```
@Configuration
@EnableWebSecurity
public class SecurityConfig {
    @Bean public PasswordEncoder passwordEncoder() {...}
    @Bean public UserDetailsService userDetailsService(PasswordEncoder pe) {...}
    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http, JwtAuthFilter jwtAuthFilter) throws Exception {
        ...
        // Bỏ duy trì user trong session
        http.sessionManagement(cfg -> cfg.sessionCreationPolicy(SessionCreationPolicy.STATELESS));
        // Bổ sung JwtAuthFilter chạy trước khi thực hiện security
        http.addFilterBefore(jwtAuthFilter, UsernamePasswordAuthenticationFilter.class);
        return http.build();
    }
    @Bean
    public AuthenticationManager authenticationManager(AuthenticationConfiguration config) throws Exception {
        return config.getAuthenticationManager();
    }
}
```

LOGIN REST API

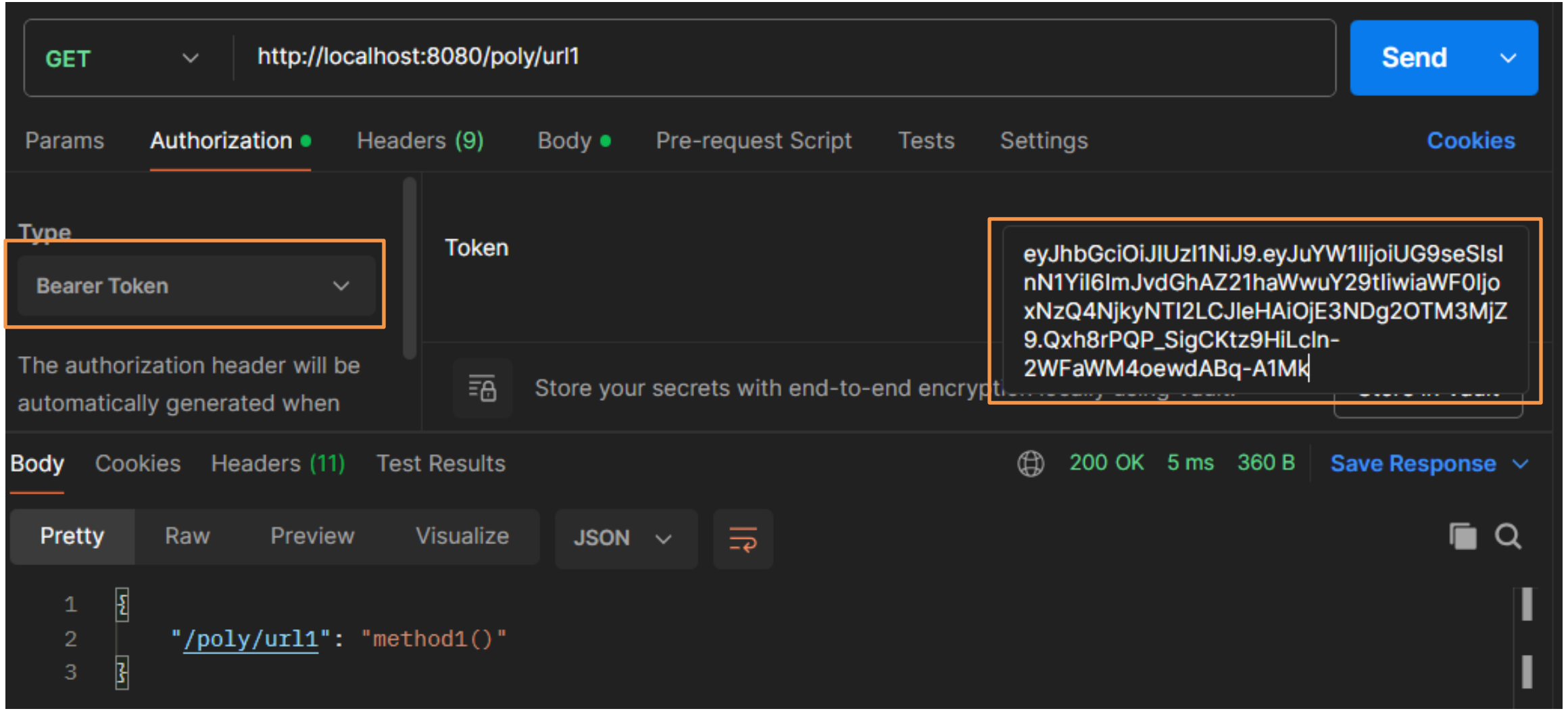
```java
@RestController
public class LoginController {
    @Autowired AuthenticationManager authenticationManager;
    @Autowired JwtService jwtService;
    @PostMapping("/poly/login")
    public Object login(@RequestBody Map<String, String> userInfo) {
        String username = userInfo.get("username");
        String password = userInfo.get("password");
        var authInfo = new UsernamePasswordAuthenticationToken(username, password);
        var authentication = authenticationManager.authenticate(authInfo); // Xác thực
        if (authentication.isAuthenticated()) {
            UserDetails user = (UserDetails) authentication.getPrincipal();
            String token = jwtService.create(user, 20 * 60); // JWT có hiệu lực 20 phút
            return Map.of("token", token); // response {token: <<token>>}
        }
        throw new UsernameNotFoundException("Username not found!");
    }
}
```

POSTMAN WITH JWT

ĐĂNG NHẬP VỚI POSTMAN



TRUY XUẤT TÀI NGUYÊN BẢO MẬT



Axios with JWT

AXIOS WITH JWT

```
let token;
function login() {
        var url = "http://localhost:8080/poly/login";
        var data = {"username": "user@gmail.com","password": "123"}
        axios.post(url, data).then(resp => {
                token = resp.data.token;
        })
}
function gotoUrl1() {
        var url = `http://localhost:8080/poly/url1`;
        var config = {headers: {"Authorization": `Bearer ${token}`}};
        axios.get(url, config).then(resp => {
                console.log("get() success", resp.data);
        }).catch(error => {
                console.log("get() error", error);
        })
}
```

- Gọi phương thức login() để đăng nhập và lấy token
- Gọi phương thức gotoUrl1() để truy xuất /poly/url1 với token đã đăng nhập

SUMMARY

- ☑ Giới thiệu JWT
- ☑ Triển khai JWT với Spring Boot
- ☑ Using JWT trong Postman
- ☑ Using JWT trong Axios


TIN HỌC VĂN PHÒNG QUỐC TẾ THEO CHUẨN ICDL
Syllabus 5 - Office 2007
LUẬT KINH TẾ
LUẬT KINH TẾ
KỸ THUẬT SOẠN THẢO VÀ TRÌNH BÀY VĂN BẢN
TIN HỌC VĂN PHÒNG QUỐC TẾ THEO CHUẨN ICDL
Syllabus 5 - Office 2007
Cảm ơn