HO CHI MINH BRANCH
Hai Au Building - 10th Floor, 39B Truong Son Str., Ward 4, Tan Binh Dist., HCM City
T +84.8.629 738 59 F +84.8.629 738 86

DA NANG BRANCH
PVcomBank Building - 13th Floor, 30/4 Str., Hai Chau Dist., Da Nang City
T +84.511.362 61 26

SECURE WEB SERVICE WITH JSON WEB TOKEN (JWT)

Author: *ATOM Team*
Date: February 12, 2018

I. Introduction

JSON Web Token (JWT) is an open standard (RFC 7519) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object. This information can be verified and trusted because it is digitally signed. JWTs can be signed using a secret (with the HMAC algorithm) or a public/private key pair using RSA.

- **Compact**: Because of their smaller size, JWTs can be sent through a URL, POST parameter, or inside an HTTP header. Additionally, the smaller size means transmission is fast.
- **Self-contained**: The payload contains all the required information about the user, avoiding the need to query the database more than once.

II. The JSON Web Token structure

JSON Web Tokens consist of three parts separated by dots (.), which are:

- Header
- Payload
- Signature

Therefore, a JWT typically looks like the following: **xxxxx.yyyyy.zzzzz**
For details information, please go to: https://jwt.io/introduction/

III. Installation

1. Libraries

There are a lot of libraries for Token Signing/Verification. You can find in here: https://jwt.io/#libraries

For this document, we use Java JWT (https://github.com/auth0/java-jwt):

- Maven:

```
<dependency>
  <groupId>com.auth0</groupId>
  <artifactId>java-jwt</artifactId>
  <version>3.3.0</version>
</dependency>
```

2. Create JWT

In authentication, when the client needs a valid token of server, they send a request include their credentials to server. The server will verify the infomation in client's request, create a JWT and send it back to client.

- **Pick the Algorithm and select secret key:**
  The Algorithm defines how a token is signed and verified. It can be instantiated with the raw value of the secret in the case of HMAC algorithms

HO CHI MINH BRANCH
Hai Au Building - 10th Floor, 39B Truong Son Str., Ward 4, Tan Binh Dist., HCM City
T +84.8.629 738 59 F +84.8.629 738 86
DA NANG BRANCH
PVcomBank Building - 13th Floor, 30/4 Str., Hai Chau Dist., Da Nang City
T +84.511.362 61 26

```
String secretKey;
Algorithm algorithm = Algorithm.HMAC512(secretKey);
```

- **Add the claims to the payload part and create token:**
  You'll first need to create a *JWTCreator* instance by calling *JWT.create()*. Use the builder to define the custom Claims your token needs to have. Finally to get the String token call *sign()* and pass the *Algorithm* instance.
  The library supports many claims: **iss** (issuer), **exp** (expiration time), **sub** (subject). You can also create new claims.

```
try {
      String secretKey;
      Date expDate;
      //select algorithm and secret key
      Algorithm algorithm = Algorithm.HMAC512(secretKey);
      //create JWT
      String token = JWT.create()
                  //iss (issuer)
                  .withIssuer("issuer")
                  //exp (expiration time)
                  .withExpiresAt(expDate)
                  //create a new claim
                  .withClaim("claimName", "claimValue")
                  .sign(algorithm);
} catch (IllegalArgumentException e) {
      e.printStackTrace();
} catch (UnsupportedEncodingException e) {
      //UTF-8 encoding not supported
      e.printStackTrace();
} catch (JWTCreationException e) {
      //Invalid Signing configuration / Couldn't convert Claims.
      e.printStackTrace();
}
```

If a Claim couldn't be converted to JSON or the Key used in the signing process was invalid a *JWTCreationException* will raise.

3. Validate JWT

Whenever the user wants to access a protected route or resource, the user agent should send the JWT in their request. The server will check for a valid JWT.

You'll first need to create a *JWTVerifier* instance by calling *JWT.require()* and passing the *Algorithm* instance. If you require the token to have specific Claim values, use the builder to define them. The instance returned by the method *build()* is reusable, so you can define it once and use it to verify different tokens. Finally call *verifier.verify()* passing the token.

```
String token = "xxxxx.yyyyy.zzzzz";
try {
      String secretKey = appConfigService.getSecretKey();
      Algorithm algorithm = Algorithm.HMAC512(secretKey);
      JWTVerifier verifier = JWT.require(algorithm)
                              .withIssuer("issuer")
                              .build();
      DecodedJWT jwt = verifier.verify(token);
} catch (JWTVerificationException e) {
      //Invalid signature/claims
      e.printStackTrace();
} catch (IllegalArgumentException e) {
      e.printStackTrace();
} catch (UnsupportedEncodingException e) {
```

HO CHI MINH BRANCH
Hai Au Building - 10th Floor, 39B Truong Son Str., Ward 4, Tan Binh Dist., HCM City
T +84.8.629 738 59 F +84.8.629 738 86

DA NANG BRANCH
PVcomBank Building - 13th Floor, 30/4 Str., Hai Chau Dist., Da Nang City
T +84.511.362 61 26

```
            //UTF-8 encoding not supported
            e.printStackTrace();
        }
```

If the token has an invalid signature or the Claim requirement is not met, a *JWTVerificationException* will raise.

IV. References

For more information and features, please go to:

- Home page: https://jwt.io/
- RFC standard for JSON Web Token: https://tools.ietf.org/html/rfc7519

V. Example

This is a RestAPI before we apply JWT. This API will save an employee into system:

```
@POST
@Consumes({MediaType.APPLICATION_JSON, MediaType.APPLICATION_XML})
public Response employee(@Valid Employee emp){
    empService.save(empService.toEntity(emp));
    return Response.created(appendCurrentUriWith(emp.getEmployeeid()))
            .entity(ResultStatus.buildSuccessResultStatus())
            .type(MediaType.APPLICATION_JSON).build();
}
```

- **Step 1:** Create a RestAPI for user to get a JWT

```
@Stateless
@Path("auth")
public class JWTAuthenticationResources {

    @EJB
    JWTAuthenticationServices jwtAuthServices;

                                                    User will send usernam and password to server
    @POST
    @Consumes({MediaType.APPLICATION_JSON, MediaType.APPLICATION_XML})
    public JWTAuthenticationResponse getAuthenticationToken(@Valid User user) {
        Token token = jwtAuthServices.createToken(user);  <-- Validate user info and create token
        return new JWTAuthenticationResponse(token, ResultStatus.buildSuccessResultStatus());
    }
```

HO CHI MINH BRANCH
Hai Au Building - 10th Floor, 39B Truong Son Str., Ward 4, Tan Binh Dist., HCM City
T +84.8.629 738 59 F +84.8.629 738 86

DA NANG BRANCH
PVcomBank Building - 13th Floor, 30/4 Str., Hai Chau Dist., Da Nang City
T +84.511.362 61 26

- **Step 2:** Implement a service (Eg: JWTAuthenticationServices) with a function (Eg: createToken) to create JWT

```
@EJB
private AuthenticationServices authServices;

public Token createToken(User user) {
    // Validate user info before create JWT.
    // If info is invalid, this function will throw an exception
    authServices.checkValidUser(user);

    // Initiate info of JWT.
    // You can get this info form anywhere you save it (db, properties file)
    String token = null;
    String secretKey = appConfigService.getSecretKey();// Create secret key
    String issuer = appConfigService.getJWTIssuer();   // Create issuer
    int timeToLive = appConfigService.getTimeToLive(); // Create the valid time of JWT

    try {
        // Select a algorithm and set secret key to it
        Algorithm algorithm = Algorithm.HMAC512(secretKey);
        // Create JWT. This is the place to add public, private claim
        token = JWT.create()
                // Set issuer to JWT
                .withIssuer(issuer)
                // Set an id to JWT to make sure it's unique
                .withJWTId(UUID.randomUUID().toString())
                // Set your private claim to JWT
                .withClaim("username", user.getUsername())
                // Set the valid time of JWT
                .withExpiresAt(this.setTokenTimeToLive(timeToLive))
                // Set algorithm to JWT and create JWT
                .sign(algorithm);
    } catch (IllegalArgumentException e) {
        logger.info(e.getMessage());
    } catch (UnsupportedEncodingException e) {
        logger.info(e.getMessage());
    }
    if (token == null) {
        throw new UnauthorizedAccessException(ErrorCode.INVALID_INPUT_REQUEST,
                messageService.get(ErrorCode.INVALID_INPUT_REQUEST));
    }
    return new Token(token, timeToLive);
}
```

HO CHI MINH BRANCH
Hai Au Building - 10th Floor, 39B Truong Son Str., Ward 4, Tan Binh Dist., HCM City
T +84.8.629 738 59 F +84.8.629 738 86

DA NANG BRANCH
PVcomBank Building - 13th Floor, 30/4 Str., Hai Chau Dist., Da Nang City
T +84.511.362 61 26

- **Step 3:** Add JWT into the header of the RestAPI that you need to apply JWT.
  The format of header is: **Authorization – Bearer *<your JWT>***.
  (Eg: Authorication – Bearer xxxxx.yyyyy.zzzzz)
  This is the RestAPI after we apply JWT:

```
@POST
@Consumes({MediaType.APPLICATION_JSON, MediaType.APPLICATION_XML})
public Response employee(@HeaderParam("Authorization") String authorization,
                         @Valid Employee emp){
    // Validate Authorization header.
    // It will throw an exception if the header (include JWT) is invalid
    jwtAuthServices.checkAuthorizedToken(authorization);
    // Save employee info
    empService.save(empService.toEntity(emp));
    return Response.created(appendCurrentUriWith(emp.getEmployeeid()))
            .entity(ResultStatus.buildSuccessResultStatus())
            .type(MediaType.APPLICATION_JSON).build();
}
```

- **Step 4:** Implement a function (Eg: checkAuthorizedToken) to check JWT in JWTAuthenticationServices

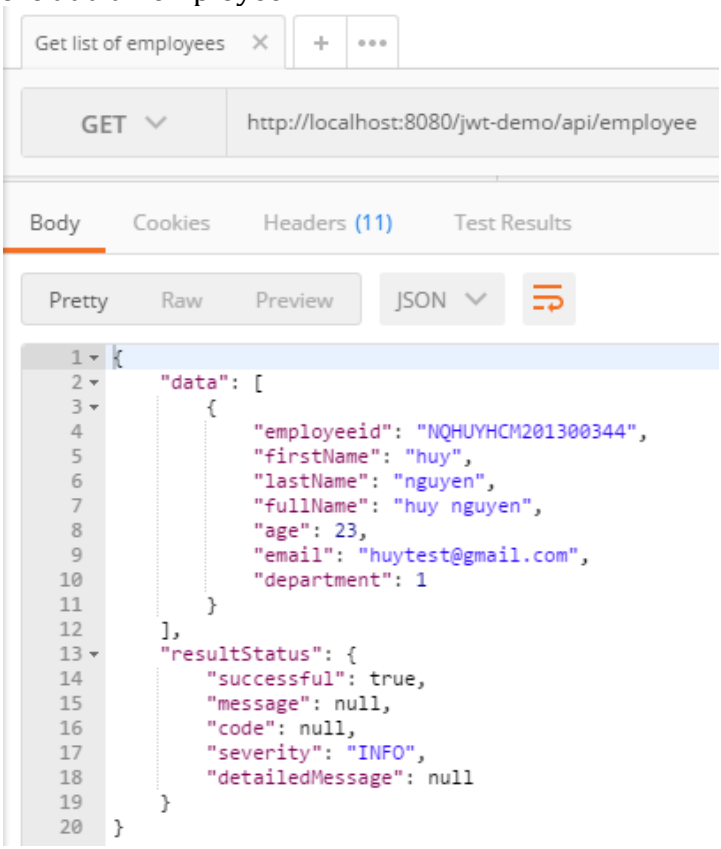
```
public void checkAuthorizedToken(String authorization) {
    if (authorization == null)
        throw new UnauthorizedAccessException(ErrorCode.INVALID_INPUT_REQUEST,
                messageService.get(ErrorCode.INVALID_INPUT_REQUEST));
    // Validate the format of header and get JWT from it
    String[] authParts = authorization.split("\\s+");
    if ( (authParts.length < 2) || !"Bearer".equals(authParts[0]) )
        throw new UnauthorizedAccessException(ErrorCode.INVALID_INPUT_REQUEST,
                messageService.get(ErrorCode.INVALID_INPUT_REQUEST));
    String jwtToken = authParts[1];

    try {
        // Get the secret key that we used to created the JWT
        String secretKey = appConfigService.getSecretKey();
        // Select the algorithm and set secret key that we used to created the JWT
        Algorithm algorithm = Algorithm.HMAC512(secretKey);
        // Create a JWTVerifier instance by calling JWT.require()
        // and passing the Algorithm instance
        JWTVerifier verifier = JWT.require(algorithm)
                // Define the claims that the JWTs have to include
                // Eg: The valid JWTs have to include the iss claim
                .withIssuer(appConfigService.getJWTIssuer())
                .build();
        // Call verifier.verify() to check the JWT
        DecodedJWT jwt = verifier.verify(jwtToken);
    } catch (JWTVerificationException e) {
        // Throw exception if the JWT is in valid
        logger.info(e.getMessage());
        throw new UnauthorizedAccessException(ErrorCode.INVALID_INPUT_REQUEST,
                messageService.get(ErrorCode.INVALID_INPUT_REQUEST));
    } catch (IllegalArgumentException | UnsupportedEncodingException e) {
        logger.info(e.getMessage());
    }
}
```

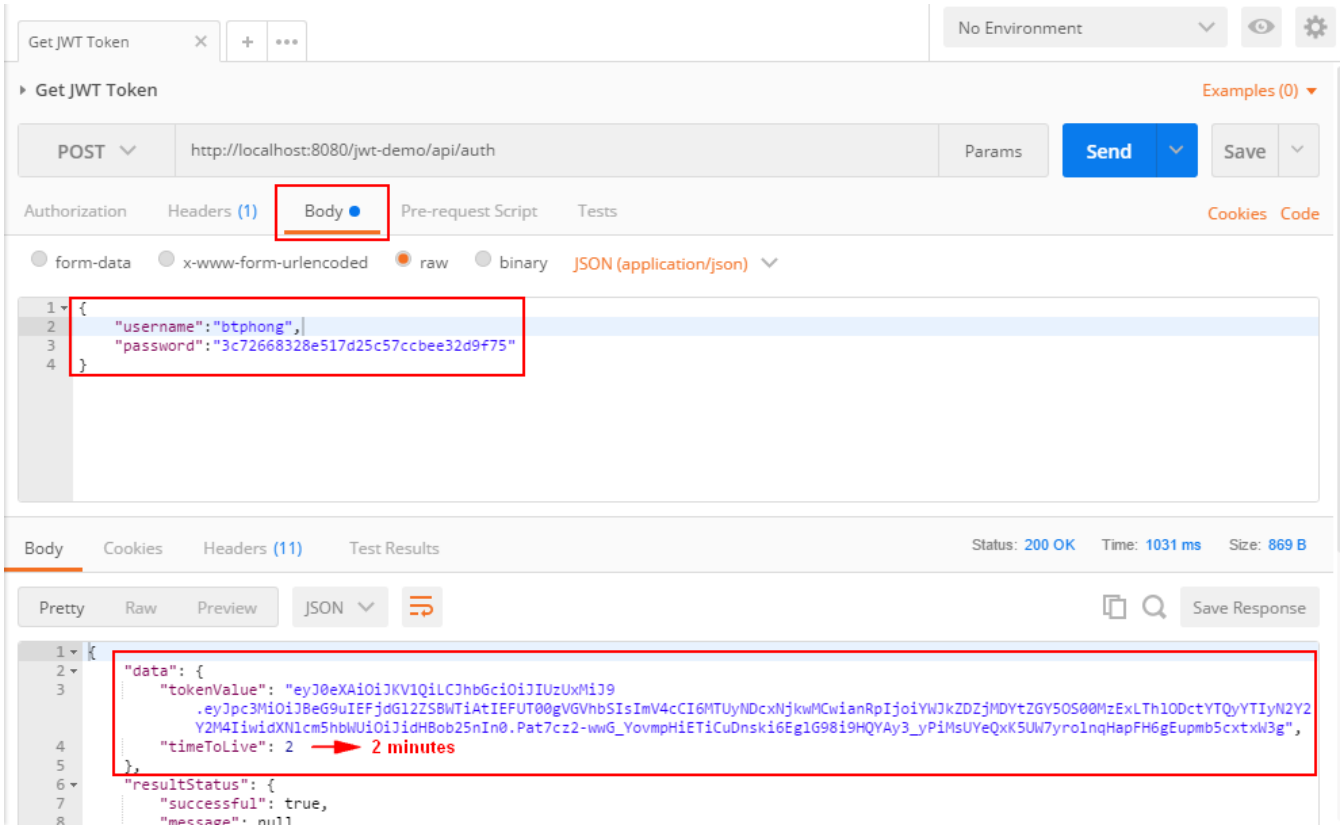
HO CHI MINH BRANCH
Hai Au Building - 10th Floor, 39B Truong Son Str., Ward 4, Tan Binh Dist., HCM City
T +84.8.629 738 59 F +84.8.629 738 86

DA NANG BRANCH
PVcomBank Building - 13th Floor, 30/4 Str., Hai Chau Dist., Da Nang City
T +84.511.362 61 26

- **Step 5:** Demo
  List of employees before add an employee:



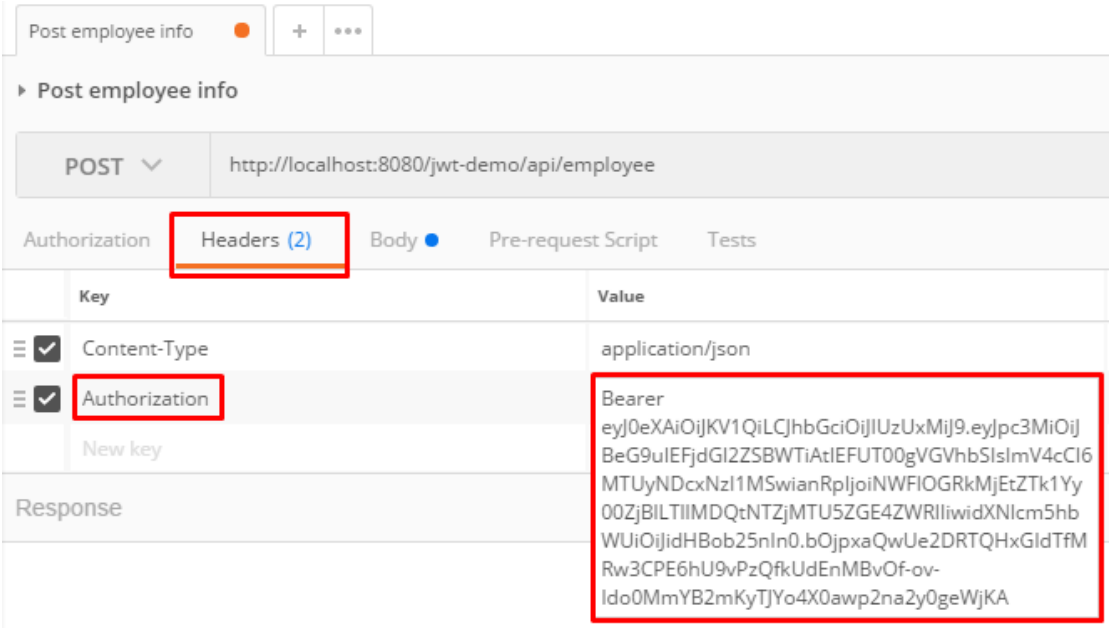
Get JWT:

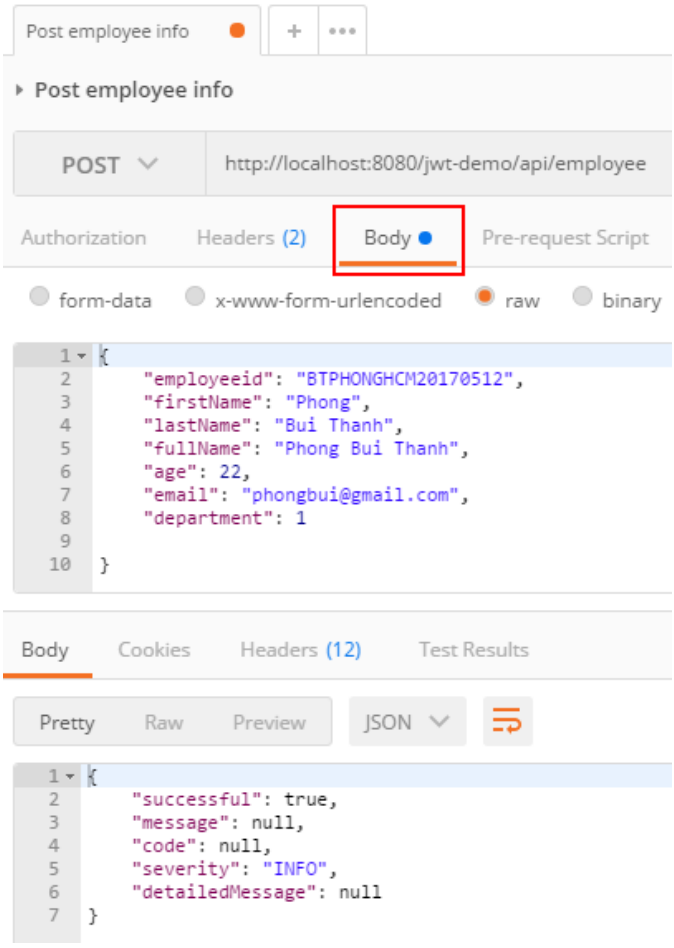


HO CHI MINH BRANCH
Hai Au Building - 10th Floor, 39B Truong Son Str., Ward 4, Tan Binh Dist., HCM City
T +84.8.629 738 59 F +84.8.629 738 86

DA NANG BRANCH
PVcomBank Building - 13th Floor, 30/4 Str., Hai Chau Dist., Da Nang City
T +84.511.362 61 26

Add JWT into header of API and save employee info:





HO CHI MINH BRANCH
Hai Au Building - 10th Floor, 39B Truong Son Str., Ward 4, Tan Binh Dist., HCM City
T +84.8.629 738 59 F +84.8.629 738 86

DA NANG BRANCH
PVcomBank Building - 13th Floor, 30/4 Str., Hai Chau Dist., Da Nang City
T +84.511.362 61 26

Result: