# **JWT**

## **Postman**

## Jwt.io



## Debugger

Warning: JWTs are credentials, which can grant access to resources. Be careful where you paste them! We do not record tokens, all validation and debugging is done on the client side.



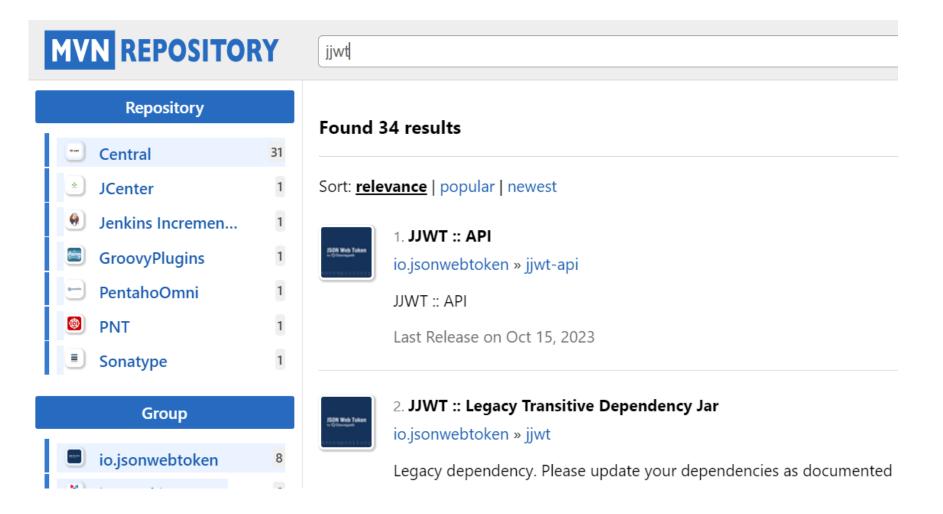
#### Encoded PASTE A TOKEN HERE

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ey
JzdWIiOiIxMjM0NTY30DkwIiwibmFtZSI6Ikpva
G4gRG91IiwiaWF0IjoxNTE2MjM5MDIyfQ.Sf1Kx
wRJSMeKKF2QT4fwpMeJf36P0k6yJV\_adQssw5c

#### Decoded EDIT THE PAYLOAD AND SECRET

## **Postman**

jjwt



# **JWT**

#### maven

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

# **JWT**

### maven

```
private final String baseKey =
"thisisdummykeythisisdummykeythisisdummykeythisisdummykeythisisdummykey";
private final SignatureAlgorithm signatureAlgorithm = SignatureAlgorithm.HS256;
private Key createKey() {
  // signiture에 대한 정보는 Byte array로 구성되어있습니다.
   byte[] apiKeySecretBytes = DatatypeConverter.parseBase64Binary(baseKey);
   Key signingKey = new SecretKeySpec(apiKeySecretBytes,
signatureAlgorithm.getJcaName());
   return signingKey;
public String createJwt(HttpServletRequest request) throws Exception {
   Map < String, Object > headerMap = new HashMap < String, Object > ();
   headerMap.put("typ", "JWT");
   headerMap.put("alg", "HS256");
   Map < String, Object > claims = new HashMap < String, Object > ();
   claims.put("name", request.getParameter("name"));
   claims nut("id" request getParameter("id")).
```