JWT - JSON Web Token

JSON - Java Script object Notation

Represent the data in small format, also can encode the data.

* JSON Web Token are open, industry standard RFC 7519 method for representing claims securely between two parties.

* It has diff algorithm.

* When we exchange info between client & Server we can use encoded & have diff Signature.

Plain text → Cipher Text → Plain key

Cryptography

* Use HTTPS to secure token.

* It is mostly used for authorizat"

Using JSON web token, websites will remember & manage Sessions.

Session ID + Cookies →

popular mechanism for authorizat"

Three parts of JWT

- **Header** → Algo (says its a JWT & what type of sig)
- **Payload** → Value inside (name, id etc)
- **Signature** → Like a stamp to prove token is real & not changed.

* JWT is a Value token
* Main point of creating JWT is to send value from server to the client so that the client can sent it back.

Payload → Data that we want to send encoded into Base 64.

Signature has a secret key.

JWT is for future interact^n. it comes in picture after authentication.

Q. How secure is a JWT if it is readable by anyone?
- No Confidential / Sensitive info in a JWT.
- Enough info for server to know the user.

~~How to disable JWT?~~

→ Dependencies to be added

* jjwt-api
* jjwt-impl
* jjwt-jackson.

@Bean → when we want to handle it for authenticat".

How JWT works

1 Send email & password to the login API.
2 if it is correct, server gives a JWT Token.
3. Send token in authorizat" header
   Eg - authorizat": Bearer <your-Token>.
4. Server will check token & allow or block access.

How to Create JWT Token

Syntax

```
String token = Jwts.builder()
 .setSubject("user@gmail.com") // username / email
 .setIssueAt(new Date())       // Current time
 .setExpiration(new Date(System.currentTimeMillis()
        +86400000)) // 1 day.
 .signWith(SignatureAlgorithm.HS256, "mysecretkey")
  // Secret key
 .compact();
```

Why use JWT

* No need to check DB everytime
* Token has signature, can't be faked
* Good for APIs & Mobile apps.

Dependency

Spring Web

Spring Data JPA

H2 Database

Spring Security

#PTR.

@ElementCollect" → allows storing a list of roles for each user.

*Header

Consist of two parts: the type of token (JWT) & signing algo being used such as HMAC SHA256 or RSA

```
{
    "alg": "HS256",
    "typ": "JWT"
}.
```

Payload

Contains the claims. Claims are statements about an entity (typically, the user) & additional metadata.

```
{"Email": "email@gmail.com",
    "Name": "John Doe";
}
```

Signature

Used to verify that sender of JWT is who it says it is & to ensure that the message wasn't changed along the way.

To create signature part, you have to take encoded header, the encoded payload, a secret, the algo. specified in header & sign that.

```
HMACSHA256(
  Secret,
  base64UrlEncode(header) + "." + base64UrlEncode(payload)
)
```