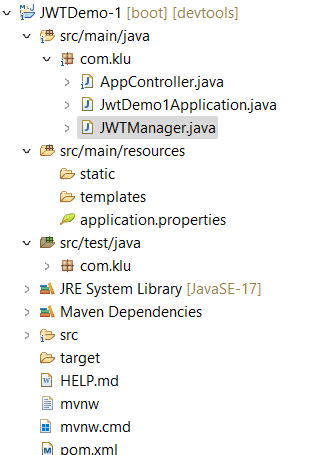
**Implementing JWT Token With Encryption And Decryption**

**Dependencies:**

---------------------

Spring Web

Spring Boot dev tools



**Additional Dependency to add in Dependencies tag in Pom.xml**

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt-api</artifactId>

<version>0.11.5</version>

</dependency>

<!-- https://mvnrepository.com/artifact/io.jsonwebtoken/jjwt-impl -->

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt-impl</artifactId>

<version>0.11.5</version>

<scope>runtime</scope>

</dependency>

<!-- https://mvnrepository.com/artifact/io.jsonwebtoken/jjwt-jackson -->

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt-jackson</artifactId>

<version>0.11.5</version>

<scope>runtime</scope>

</dependency>

----------------------------------------------------------------------------------------------------------

**JWTManager.java**

**------------------------------------**

**package** com.klu;

**import** java.security.MessageDigest;

**import** java.util.Base64;

**import** java.util.Date;

**import** java.util.HashMap;

**import** java.util.Map;

**import** javax.crypto.Cipher;

**import** javax.crypto.SecretKey;

**import** javax.crypto.spec.SecretKeySpec;

**import** org.springframework.stereotype.Service;

**import** io.jsonwebtoken.Claims;

**import** io.jsonwebtoken.Jwts;

**import** io.jsonwebtoken.security.Keys;

@Service

**public** **class** JWTManager {

**public** SecretKey key = Keys.*hmacShaKeyFor*("awdsiuchuidcidvijsuidjuiwehdcdhuichecefuerhfui".getBytes());

**public** String generateToken(String username) {

Map<String, String> claims = **new** HashMap<>();

claims.put("username", encrpytData (username));

**return** Jwts.*builder*()

.setClaims(claims)

.setIssuedAt(**new** Date())

.setExpiration(**new** Date(**new** Date().getTime() + 86400000))

.signWith(key)

.compact();

}

**public** Map<String, String> validateToken(String Token){

**try** {

Claims claims = Jwts.*parserBuilder*()

.setSigningKey(key)

.build()

.parseClaimsJws(Token)

.getBody();

Date expiration = claims.getExpiration();

**if**(expiration == **null** || expiration.before(**new** Date())) {

Map<String, String> res = **new** HashMap<>();

res.put("code", "404");

res.put("message", "Invalid Token");

**return** res;

}

Map<String, String> res = **new** HashMap<>();

res.put("code", "200");

res.put("message", decrpytData (claims.get("username", String.**class**)));

**return** res;

}

**catch** (Exception e) {

Map<String, String> res = **new** HashMap<>();

res.put("code", "404");

res.put("message", "Invalid Token");

**return** res;

}

}

**public** String encrpytData (String data) {

**try** {

MessageDigest MD5 = MessageDigest.*getInstance*("SHA-256");

**byte** [] keyBytes = MD5.digest("BALAJEE".getBytes());

SecretKey key1 = **new** SecretKeySpec(keyBytes, 0 ,16,"AES");

Cipher cipher = Cipher.*getInstance*("AES");

cipher.init(Cipher.***ENCRYPT\_MODE***, key1);

**byte**[] encryptData = cipher.doFinal(data.getBytes());

**return** Base64.*getEncoder*().encodeToString(encryptData);

}

**catch**(Exception e) {

**return** e.getMessage();

}

}

**public** String decrpytData (String data) {

**try** {

MessageDigest MD5 = MessageDigest.*getInstance*("SHA-256");

**byte** [] keyBytes = MD5.digest("BALAJEE".getBytes());

SecretKey key1 = **new** SecretKeySpec(keyBytes, 0 ,16,"AES");

Cipher cipher = Cipher.*getInstance*("AES");

cipher.init(Cipher.***DECRYPT\_MODE***, key1);

**byte**[] decryptData = cipher.doFinal(Base64.*getDecoder*().decode(data));

**return** **new** String(decryptData);

}

**catch**(Exception e) {

**return** e.getMessage();

}

}

}

----------------------------------------------------

**AppController.java**

**--------------------------------------------------**

**package** com.klu;

**import** java.util.Map;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.web.bind.annotation.GetMapping;

**import** org.springframework.web.bind.annotation.RequestParam;

**import** org.springframework.web.bind.annotation.RestController;

@CrossOrigin

@RestController

**public class** AppController {

@Autowired

JWTManager **jwt;**

@GetMapping(**"/login**") //

**public** String fun1(@RequestParam(**"username"**)Stringusername) {

**return jwt**.generateToken(username**);**

}

@GetMapping("/validate")

**public** Map<String, String>fun2(@RequestParam**("token**")Stringtoken){

**return jwt**.validateToken(**token**);

}

}

-----------------------------------------------------------

**Frontend Code:**

**---------------------------------------**

**JWTLogin.jsx**

import React, { useState } from 'react';

import axios from 'axios';

const Login = () => {

const [username, setUsername] = useState('');

const [token, setToken] = useState('');

const [message, setMessage] = useState('');

const handleLogin = async () => {

try {

const response = await axios.get(`http://localhost:8080/login?username=${username}`);

setToken(response.data);

localStorage.setItem('jwt', response.data);

setMessage('Login successful! Token saved.');

} catch (error) {

console.error('Login failed', error);

setMessage('Login failed.');

}

};

const validateToken = async () => {

const storedToken = localStorage.getItem('jwt');

if (!storedToken) {

setMessage('No token found. Please log in.');

return;

}

try {

const response = await axios.get(`http://localhost:8080/validate?token=${storedToken}`);

if (response.data.code === '200') {

setMessage(`Token valid! Welcome, ${response.data.message}`);

} else {

setMessage('Invalid token. Please log in again.');

}

} catch (error) {

console.error('Token validation failed', error);

setMessage('Token validation failed.');

}

};

return (

<div style={{ padding: '20px' }}>

<h2>JWT Login</h2>

<input

type="text"

placeholder="Enter username"

value={username}

onChange={(e) => setUsername(e.target.value)}

/>

<button onClick={handleLogin}>Login</button>

<button onClick={validateToken} style={{ marginLeft: '10px' }}>

Validate Token

</button>

<p>{message}</p>

</div>

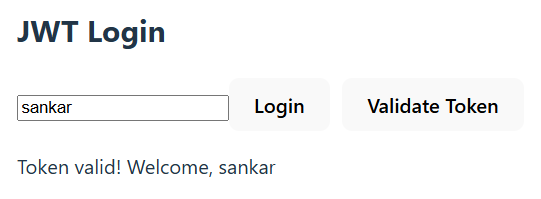
);

};

export default Login;

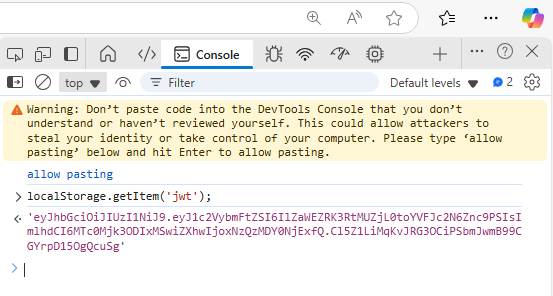
**---------------------------------**

**OUTPUT**

****

**To verify the token in the browser:**

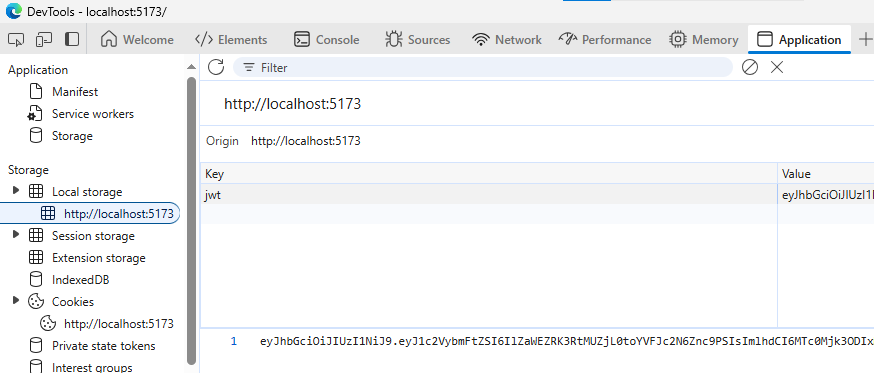
**Method1:**

****

**Method2:**

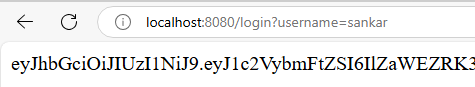
**Go to the "Application" Tab in the Browser Console**

* In the DevTools menu, select the **"Application"** tab.
* On the left sidebar, find **Storage** → **Local Storage**
* Click on your website’s URL under **Local Storage**.

****

**Method3: (For JWT login with getmapping :))**

http://localhost:8080/login?username=sankar

****

**For JWT Validate Token**

[http://localhost:8080/validate?token= **copy**](http://localhost:8080/validate?token=%20copy) **and paste the generated token**

