Spring Boot Security using JWT
Mr. RAGHU(NareshIT)

Email: javabyraghu@gmail.com

Preq:
REST
DATA JPA
JWT
SECURITY

Stage #1 User/Client Register Process
Stage #2 JwtUtil class code
Stage #3 JWT Token generation without Security (Testing)
Stage #4 JWT Token after User Authentication
Stage #5 JWT Token Verification from 2nd request onwards

-***-
Stage #1 User/Client Register Process
-***-
Model : User (id,name,username,password,roles)

*) Client has to send above details as JSON format
that will be converted into User class object
and finally saved in Database.

--code files--
1. application.properties
2. User Model class
3. User Repository
4. IUserService
5. UserServiceImpl
6. UserController

POSTMAN SCREEN
POST http://localhost:8080/user/save SEND
BODY
raw(*) JSON

```
{
 "name" : "SAM",
 "username" : "sam",
 "password" : "nit",
 "roles" : ["ADMIN","EMP"]
}
```

-***-
Stage #2 JwtUtil class code
-***-
FB: https://www.facebook.com/groups/thejavatemple/

Java JWT API : JJWT

```
<dependency>
    <groupId>io.jsonwebtoken</groupId>
    <artifactId>jjwt</artifactId>
    <version>0.9.1</version>
</dependency>
```

7. JwtUtil.java

a. Generate Token using Subject
b. Validate Token with username, tokenSubject and Exp Date.

```
*************************************************************************
       Stage #3 JWT Token generation without Security (Testing)
*************************************************************************
 Read username and password as JSON format , convert to UserRequest object
 [---validate with database rows--]
 Generate token and define one message, return this data as JSON Format
 using UserResponse object.


8. UserRequest (Http Request JSON Data)
9. UserResponse (http Response JSON Data)
10. UserRestController#loginUser


       --POSTMAN SCREEN--
POST  http://localhost:8080/user/login send
         Body
              raw(*)  JSON
  {
     "username" :  " ",
     "password" :  "  "
  }


*************************************************************************
            Stage #4 JWT Token after User Authentication
*************************************************************************
Add in pom.xml
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-security</artifactId>
</dependency>


11. AppConfig : Define one PasswordEncoder and use before saveUser
    => Modify code in UserServiceImpl for usersave(), encode Password and save.

12. SecurityConfig
   -> UserDetailsService (loadUserbyusername)
   -> authenticationEntryPoint (optional)
   -> SessionCreationPolicy.STATELESS

*** Note: AuthenticationManager (checks username and password) incase of Session
            here No Session, checking should be manual.

13. UserController#loginUser
     User AuthenticationManager and validate username and password  as a Token.
     [UsernamePasswordAuthenticationToken]

     If it is valid -> JwtToken is generated
     else : InvaildUserAuthEntryPoint is triggered.

------------TEST-----------
1. create new user

         POSTMAN SCREEN
  POST   http://localhost:8080/user/save   SEND
              BODY
                   raw(*)   JSON
   {
    "name" : "SAM",
    "username" : "sam",
    "password" : "nit",
    "roles" : ["ADMIN","EMP"]
  }
```

2. Try to login for Token
 --POSTMAN SCREEN--
POST http://localhost:8080/user/login send
         Body
              raw(*) JSON
  {
     "username" : "abc",
     "password" : "xyz"
  }

****
          Stage #5 JWT Token Verification from 2nd request onwards
****
*) Defin one filter which gets executed for every request.
  if Request is having Authorization Header , then validate using JwtUtil
  If valid then create Authentication inside SecurityContext.

=> user OncePerRequestFilter(AC) , override method   doFilterInternal()

http://localhost:8080/user/welcome

--Test--------

Try to login for Token
 --POSTMAN SCREEN--
POST http://localhost:8080/user/login send
         Body
              raw(*) JSON
  {
     "username" : "abc",
     "password" : "xyz"
  }

Access Resource

POST http://localhost:8080/user/welcome send
        Head

   Authorization = token