**CODE**

**src/main/java/com.cognizant.spring\_learn**

package com.cognizant.spring\_learn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class JWTExampleApplication {

public static void main(String[] args) {

SpringApplication.*run*(JWTExampleApplication.class, args);

}

}

**src/main/java/com.cognizant.spring\_learn/Model**

package com.cognizant.spring\_learn.Model;

import lombok.Data;

@Data

public class RequestAuth {

private String username;

private String password;

}

**src/main/java/com.cognizant.spring\_learn/Controller**

package com.cognizant.spring\_learn.Controller;

import com.cognizant.spring\_learn.Configuration.Security.JWTUtil;

import com.cognizant.spring\_learn.Model.RequestAuth;

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

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

import org.springframework.security.core.Authentication;

import org.springframework.security.core.AuthenticationException;

import org.springframework.security.core.userdetails.UserDetails;

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

@RestController

*//you forgot restcontroller*

*//lesson - unit test each code by chatgpt or by output verifying!*

*//do,because you are dumb!*

@RequestMapping("/signin/")

*//method name cannot be /login*

public class LoginController

{

private final AuthenticationManager authManager;

private final JWTUtil jwtUtil;

@Autowired

*//little need of Autowired tho*

public LoginController(AuthenticationManager authManager, JWTUtil jwtUtil) {

this.authManager = authManager;

this.jwtUtil = jwtUtil;

}

@ GetMapping("/test2")

public String hello2()

{

return "test\_success!";

}

@PostMapping("/test")

public String hello()

{

return "test\_success!";

}

@PostMapping("/yo")

public ResponseEntity<?> login(@RequestBody RequestAuth requestAuth)

*//could have accessed requestAuth through @RequestHeader too*

*//but for some reason that way is not standard*

*//in PostMan, do not forget to use Post command*

*//Add Body in body; and do not forget to use JSON as content-type!!!*

{

try {

Authentication auth = authManager.authenticate( new UsernamePasswordAuthenticationToken( requestAuth.getUsername(),requestAuth.getPassword() ));

UserDetails userDetails = (UserDetails) auth.getPrincipal();

String token = jwtUtil.generateToken(userDetails);

return ResponseEntity.*ok*(token);

} catch (AuthenticationException e) {

return ResponseEntity.*status*(HttpStatus.*UNAUTHORIZED*).build();

}

}

}

**src/main/java/com.cognizant.spring\_learn/Configuration.Security**

package com.cognizant.spring\_learn.Configuration.Security;

import io.jsonwebtoken.security.Keys;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import javax.crypto.SecretKey;

@Configuration

public class JWTConfig

{

@Bean

public SecretKey secretKey()

{

return Keys.*hmacShaKeyFor*("secretkey123456789 secretkey123456789".getBytes());

*//key must have sufficient byte length -> default = 256*

*//vl - Inject secretkey from application.properties later*

*//above function expects bytes*

*//string has a function for getBytes wohoo!*

}

}

**src/main/java/com.cognizant.spring\_learn/Configuration.Security**

package com.cognizant.spring\_learn.Configuration.Security;

import jakarta.servlet.FilterChain;

import jakarta.servlet.ServletException;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

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

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

import org.springframework.security.core.context.SecurityContextHolder;

import org.springframework.security.core.userdetails.UserDetails;

import org.springframework.security.core.userdetails.UserDetailsService;

import org.springframework.security.web.authentication.WebAuthenticationDetailsSource;

import org.springframework.web.filter.OncePerRequestFilter;

import java.io.IOException;

public class JWTRequestFilter extends OncePerRequestFilter {

final private JWTUtil jwtUtil;

final private UserDetailsService userDetailsService;

@Autowired

public JWTRequestFilter(JWTUtil jwtUtil, UserDetailsService userDetailsService) {

this.jwtUtil = jwtUtil;

this.userDetailsService = userDetailsService;

}

@Override

protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain filterChain) throws ServletException, IOException {

String authHeader = request.getHeader("Authorization");

if(authHeader != null && authHeader.startsWith("Bearer"))

{

String token = authHeader.substring(7);

String username = null;

username = jwtUtil.extractUsername(token);

if(username != null && SecurityContextHolder.*getContext*().getAuthentication() == null)

{

username = jwtUtil.extractUsername(token);

*//e can throw exception; check for it!!!*

UserDetails userDetails = userDetailsService.loadUserByUsername(username);

UsernamePasswordAuthenticationToken authToken = new UsernamePasswordAuthenticationToken(userDetails.getUsername(),userDetails.getPassword());

authToken.setDetails(new WebAuthenticationDetailsSource().buildDetails(request));

SecurityContextHolder.*getContext*().setAuthentication(authToken);

}

}

filterChain.doFilter(request, response);

}

}

**src/main/java/com.cognizant.spring\_learn/Configuration.Security**

package com.cognizant.spring\_learn.Configuration.Security;

import io.jsonwebtoken.Claims;

import io.jsonwebtoken.SignatureAlgorithm;

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

import org.springframework.security.core.GrantedAuthority;

import org.springframework.security.core.userdetails.UserDetails;

import org.springframework.stereotype.Component;

import io.jsonwebtoken.Jwts;

import org.springframework.stereotype.Service;

import javax.crypto.SecretKey;

import java.util.Date;

import java.util.stream.Collectors;

@Service

public class JWTUtil

{

private final SecretKey secretKey;

@Autowired

public JWTUtil(SecretKey secretKey) {

this.secretKey = secretKey;

}

public String generateToken(UserDetails userDetails)

{

System.*out*.println("Inside generateToken");

*//try debugging*

return Jwts.*builder*()

.setSubject(userDetails.getUsername())

.claim("roles", userDetails.getAuthorities()

.stream().map(GrantedAuthority::getAuthority)

.collect(Collectors.*toList*()))

.setIssuedAt(new Date())

.setExpiration( new Date(System.*currentTimeMillis*() + 1000\*60\*60) ) *//valid for 1 hour*

.signWith(secretKey,SignatureAlgorithm.*HS256*)

.compact();

}

public Claims extractClaims(String token)

{

return Jwts.*parserBuilder*()

.setSigningKey(secretKey)

.build()

.parseClaimsJws(token)

.getBody();

}

public String extractUsername(String token)

{

String username = extractClaims(token).getSubject();

return username;

}

public boolean validateExpiration(String token)

{

Date d = extractClaims(token).getExpiration();

long exp\_time = d.getTime();

long curr\_time = System.*currentTimeMillis*();

return exp\_time > curr\_time;

}

}

**src/main/java/com.cognizant.spring\_learn/Configuration.Security**

package com.cognizant.spring\_learn.Configuration.Security;

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

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.config.Customizer;

import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

import org.springframework.security.config.http.SessionCreationPolicy;

import org.springframework.security.core.userdetails.UserDetailsService;

import org.springframework.security.crypto.password.PasswordEncoder;

import org.springframework.security.web.SecurityFilterChain;

import static org.springframework.security.config.Customizer.*withDefaults*;

@Configuration

@EnableWebSecurity

public class SecurityConfig

{

private final UserDetailsService userDetailsService;

private final PasswordEncoder passwordEncoder;

@Autowired

public SecurityConfig(UserDetailsService userDetailsService, PasswordEncoder passwordEncoder) {

this.userDetailsService = userDetailsService;

this.passwordEncoder = passwordEncoder;

}

@Bean

public AuthenticationManager authenticationManager(HttpSecurity http) throws Exception {

AuthenticationManagerBuilder authBuilder = http.getSharedObject(AuthenticationManagerBuilder.class);

authBuilder.userDetailsService(userDetailsService)

.passwordEncoder(passwordEncoder);

return authBuilder.build();

}

@Bean

public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {

http

.csrf(csrf -> csrf

*//.ignoringRequestMatchers("/h2-console/\*\*", "/signin/\*\*")*

.disable()

)

*//ignore for understanding*

.authorizeHttpRequests(auth -> auth

*//.requestMatchers("/signin","/signin/","/signin/\*\*").permitAll()*

.requestMatchers("/signin","/signin/\*\*").permitAll()

*// /signin and /signin/ are totally different*

*// /signin means signin is an end path and a resource, not a directory*

*// /signin/ can be used in the Controller's RequestMapping because then it will be a directory*

*// /signin/\* and /signin/\*\* are a bit different*

*//.anyRequest().authenticated()*

.anyRequest().permitAll()

)

.headers(headers -> headers

.frameOptions(frame -> frame.sameOrigin())

)

*//ignore*

.sessionManagement(session ->

session.sessionCreationPolicy(SessionCreationPolicy.*STATELESS*))

.formLogin(form -> form.disable());

*//spring still respects your UserDetailsService over application.properties*

return http.build();

*// in requestMatchers - \*\* means every subpath inside, \* means only one path inside*

*// /signin/"\*\*"/verify, shows "\*\*" -> is a kind of regular expression*

}

*/\**

*@Bean*

*public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {*

*http*

*.csrf(csrf -> csrf.disable())*

*.authorizeHttpRequests(auth -> auth*

*//.requestMatchers("/signin","/signin/","/signin/\*\*").permitAll()*

*.requestMatchers("/signin","/signin/\*").permitAll()*

*// /signin and /signin/ are totally different*

*// /signin means signin is an end path and a resource, not a directory*

*// /signin/ can be used in the Controller's RequestMapping because then it will be a directory*

*// /signin/\* and /signin/\*\* are a bit different*

*.anyRequest().authenticated()*

*)*

*.sessionManagement(session ->*

*session.sessionCreationPolicy(SessionCreationPolicy.STATELESS)*

*)*

*.httpBasic(Customizer.withDefaults())*

*.formLogin(form -> form.disable());*

*return http.build();*

*}\*/*

}

**src/main/java/com.cognizant.spring\_learn/Configuration.Security**

package com.cognizant.spring\_learn.Configuration.Security;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.core.userdetails.User;

import org.springframework.security.core.userdetails.UserDetails;

import org.springframework.security.core.userdetails.UserDetailsService;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

import org.springframework.security.crypto.password.PasswordEncoder;

import org.springframework.security.provisioning.InMemoryUserDetailsManager;

@Configuration

public class UserConfig {

@Bean

public UserDetailsService userDetailsService()

{

UserDetails user = User.*withUsername*("Aryan")

.password(passwordEncoder().encode("pass123"))

.roles("user","dummy")

.build();

UserDetails admin = User.*withUsername*("DBA")

.password(passwordEncoder().encode("321ssap"))

.roles("admin")

.build();

return new InMemoryUserDetailsManager(user, admin);

}

@Bean

public PasswordEncoder passwordEncoder()

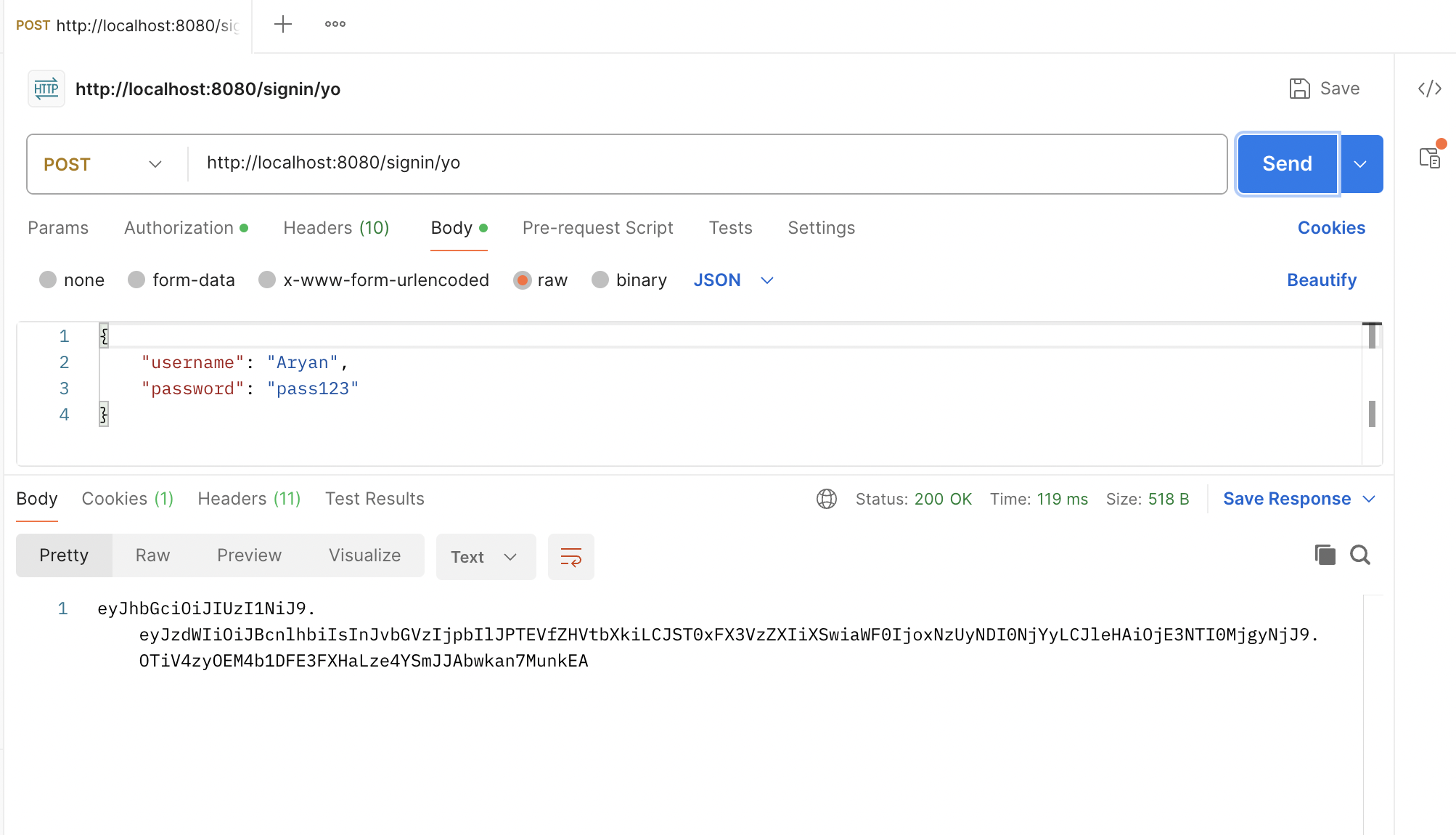
{

return new BCryptPasswordEncoder();

}

}

**OUTPUT**

****