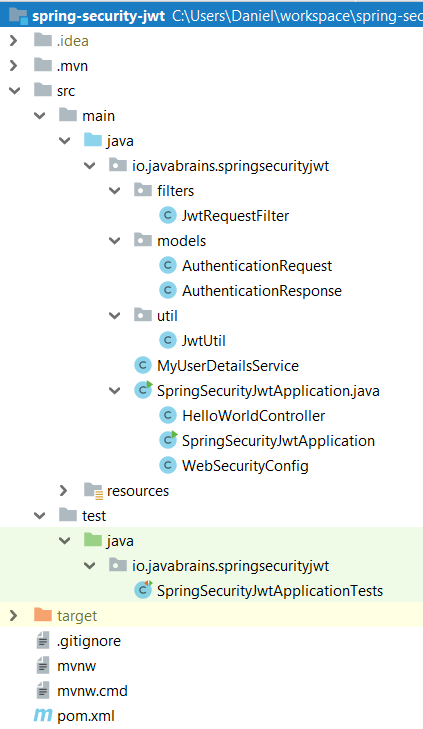
JTW WORKSHOP



**package** io.javabrains.springsecurityjwt;  
  
**import** io.javabrains.springsecurityjwt.filters.JwtRequestFilter;  
**import** io.javabrains.springsecurityjwt.models.AuthenticationRequest;  
**import** io.javabrains.springsecurityjwt.models.AuthenticationResponse;  
**import** io.javabrains.springsecurityjwt.util.JwtUtil;  
**import** org.springframework.beans.factory.annotation.Autowired;  
**import** org.springframework.boot.SpringApplication;  
**import** org.springframework.boot.autoconfigure.SpringBootApplication;  
**import** org.springframework.context.annotation.Bean;  
**import** org.springframework.http.ResponseEntity;  
**import** org.springframework.security.authentication.AuthenticationManager;  
**import** org.springframework.security.authentication.BadCredentialsException;  
**import** org.springframework.security.authentication.UsernamePasswordAuthenticationToken;  
**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.annotation.web.configuration.WebSecurityConfigurerAdapter;  
**import** org.springframework.security.config.http.SessionCreationPolicy;  
**import** org.springframework.security.core.userdetails.UserDetails;  
**import** org.springframework.security.core.userdetails.UserDetailsService;  
**import** org.springframework.security.crypto.password.NoOpPasswordEncoder;  
**import** org.springframework.security.crypto.password.PasswordEncoder;  
**import** org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;  
**import** org.springframework.web.bind.annotation.RequestBody;  
**import** org.springframework.web.bind.annotation.RequestMapping;  
**import** org.springframework.web.bind.annotation.RequestMethod;  
**import** org.springframework.web.bind.annotation.RestController;  
  
@SpringBootApplication  
**public class** SpringSecurityJwtApplication {  
  
 **public static void** main(String[] args) {  
 SpringApplication.*run*(SpringSecurityJwtApplication.**class**, args);  
 }  
  
}  
  
@RestController  
**class** HelloWorldController {  
  
 @Autowired  
 **private** AuthenticationManager **authenticationManager**;  
  
 @Autowired  
 **private** JwtUtil **jwtTokenUtil**;  
  
 @Autowired  
 **private** MyUserDetailsService **userDetailsService**;  
  
 @RequestMapping({ **"/hello"** })  
 **public** String firstPage() {  
 **return "Hello World"**;  
 }  
  
 @RequestMapping(value = **"/authenticate"**, method = RequestMethod.***POST***)  
 **public** ResponseEntity<?> createAuthenticationToken(@RequestBody AuthenticationRequest authenticationRequest) **throws** Exception {  
  
 **try** {  
 **authenticationManager**.authenticate(  
 **new** UsernamePasswordAuthenticationToken(authenticationRequest.getUsername(), authenticationRequest.getPassword())  
 );  
 }  
 **catch** (BadCredentialsException e) {  
 **throw new** Exception(**"Incorrect username or password"**, e);  
 }  
  
  
 **final** UserDetails userDetails = **userDetailsService** .loadUserByUsername(authenticationRequest.getUsername());  
  
 **final** String jwt = **jwtTokenUtil**.generateToken(userDetails);  
  
 **return** ResponseEntity.*ok*(**new** AuthenticationResponse(jwt));  
 }  
  
}  
  
@EnableWebSecurity  
**class** WebSecurityConfig **extends** WebSecurityConfigurerAdapter {  
 @Autowired  
 **private** UserDetailsService **myUserDetailsService**;  
 @Autowired  
 **private** JwtRequestFilter **jwtRequestFilter**;  
  
 @Autowired  
 **public void** configureGlobal(AuthenticationManagerBuilder auth) **throws** Exception {  
 auth.userDetailsService(**myUserDetailsService**);  
 }  
  
 @Bean  
 **public** PasswordEncoder passwordEncoder() {  
 **return** NoOpPasswordEncoder.*getInstance*();  
 }  
  
 @Override  
 @Bean  
 **public** AuthenticationManager authenticationManagerBean() **throws** Exception {  
 **return super**.authenticationManagerBean();  
 }  
  
 @Override  
 **protected void** configure(HttpSecurity httpSecurity) **throws** Exception {  
 httpSecurity.csrf().disable()  
 .authorizeRequests().antMatchers(**"/authenticate"**).permitAll().  
 anyRequest().authenticated().and().  
 exceptionHandling().and().sessionManagement()  
 .sessionCreationPolicy(SessionCreationPolicy.***STATELESS***);  
 httpSecurity.addFilterBefore(**jwtRequestFilter**, UsernamePasswordAuthenticationFilter.**class**);  
  
 }  
}

…

**package** io.javabrains.springsecurityjwt;  
  
**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.core.userdetails.UsernameNotFoundException;  
**import** org.springframework.stereotype.Service;  
  
**import** java.util.ArrayList;  
  
@Service  
**public class** MyUserDetailsService **implements** UserDetailsService {  
  
 @Override  
 **public** UserDetails loadUserByUsername(String s) **throws** UsernameNotFoundException {  
 **return new** User(**"foo"**, **"foo"**,  
 **new** ArrayList<>());  
 }  
}

…

**package** io.javabrains.springsecurityjwt.util;  
  
**import** io.jsonwebtoken.Claims;  
**import** io.jsonwebtoken.Jwts;  
**import** io.jsonwebtoken.SignatureAlgorithm;  
**import** org.springframework.security.core.userdetails.UserDetails;  
**import** org.springframework.stereotype.Service;  
  
**import** java.util.Date;  
**import** java.util.HashMap;  
**import** java.util.Map;  
**import** java.util.function.Function;  
  
@Service  
**public class** JwtUtil {  
  
 **private** String **SECRET\_KEY** = **"secret"**;  
  
 **public** String extractUsername(String token) {  
 **return** extractClaim(token, Claims::getSubject);  
 }  
  
 **public** Date extractExpiration(String token) {  
 **return** extractClaim(token, Claims::getExpiration);  
 }  
  
 **public** <T> T extractClaim(String token, Function<Claims, T> claimsResolver) {  
 **final** Claims claims = extractAllClaims(token);  
 **return** claimsResolver.apply(claims);  
 }  
 **private** Claims extractAllClaims(String token) {  
 **return** Jwts.*parser*().setSigningKey(**SECRET\_KEY**).parseClaimsJws(token).getBody();  
 }  
  
 **private** Boolean isTokenExpired(String token) {  
 **return** extractExpiration(token).before(**new** Date());  
 }  
  
 **public** String generateToken(UserDetails userDetails) {  
 Map<String, Object> claims = **new** HashMap<>();  
 **return** createToken(claims, userDetails.getUsername());  
 }  
  
 **private** String createToken(Map<String, Object> claims, String subject) {  
  
 **return** Jwts.*builder*().setClaims(claims).setSubject(subject).setIssuedAt(**new** Date(System.*currentTimeMillis*()))  
 .setExpiration(**new** Date(System.*currentTimeMillis*() + 1000 \* 60 \* 60 \* 10))  
 .signWith(SignatureAlgorithm.***HS256***, **SECRET\_KEY**).compact();  
 }  
  
 **public** Boolean validateToken(String token, UserDetails userDetails) {  
 **final** String username = extractUsername(token);  
 **return** (username.equals(userDetails.getUsername()) && !isTokenExpired(token));  
 }  
}

…

**package** io.javabrains.springsecurityjwt.filters;  
  
**import** io.javabrains.springsecurityjwt.MyUserDetailsService;  
**import** io.javabrains.springsecurityjwt.util.JwtUtil;  
**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.web.authentication.WebAuthenticationDetailsSource;  
**import** org.springframework.stereotype.Component;  
**import** org.springframework.web.filter.OncePerRequestFilter;  
  
**import** javax.servlet.FilterChain;  
**import** javax.servlet.ServletException;  
**import** javax.servlet.http.HttpServletRequest;  
**import** javax.servlet.http.HttpServletResponse;  
**import** java.io.IOException;  
  
@Component  
**public class** JwtRequestFilter **extends** OncePerRequestFilter {  
  
 @Autowired  
 **private** MyUserDetailsService **userDetailsService**;  
  
 @Autowired  
 **private** JwtUtil **jwtUtil**;  
  
 @Override  
 **protected void** doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain chain)  
 **throws** ServletException, IOException {  
  
 **final** String authorizationHeader = request.getHeader(**"Authorization"**);  
  
 String username = **null**;  
 String jwt = **null**;  
  
 **if** (authorizationHeader != **null** && authorizationHeader.startsWith(**"Bearer "**)) {  
 jwt = authorizationHeader.substring(7);  
 username = **jwtUtil**.extractUsername(jwt);  
 }  
  
  
 **if** (username != **null** && SecurityContextHolder.*getContext*().getAuthentication() == **null**) {  
  
 UserDetails userDetails = **this**.**userDetailsService**.loadUserByUsername(username);  
  
 **if** (**jwtUtil**.validateToken(jwt, userDetails)) {  
  
 UsernamePasswordAuthenticationToken usernamePasswordAuthenticationToken = **new** UsernamePasswordAuthenticationToken(  
 userDetails, **null**, userDetails.getAuthorities());  
 usernamePasswordAuthenticationToken  
 .setDetails(**new** WebAuthenticationDetailsSource().buildDetails(request));  
 SecurityContextHolder.*getContext*().setAuthentication(usernamePasswordAuthenticationToken);  
 }  
 }  
 chain.doFilter(request, response);  
 }  
  
}

…

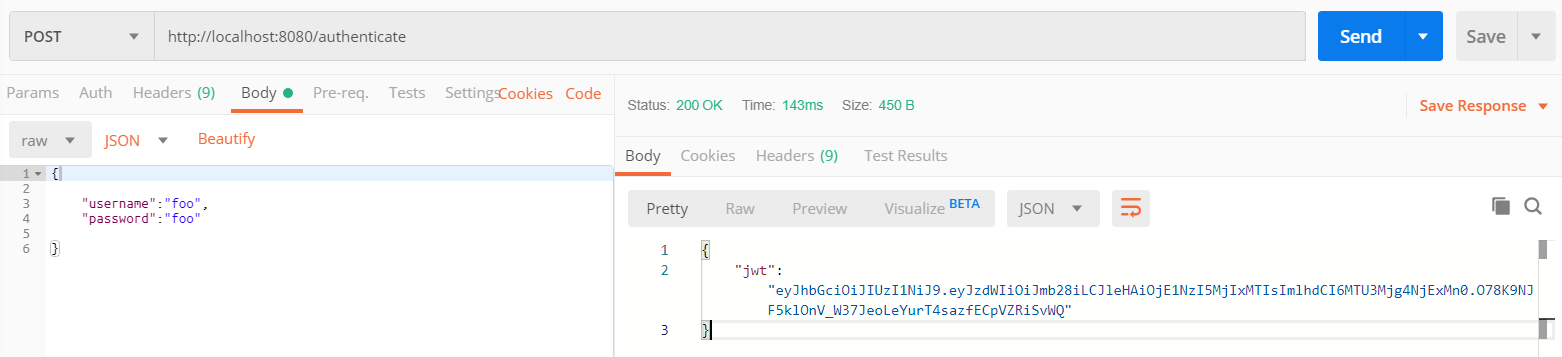
**package** io.javabrains.springsecurityjwt.models;  
  
**import** java.io.Serializable;  
  
**public class** AuthenticationRequest **implements** Serializable {  
  
  
 **private** String **username**;  
 **private** String **password**;  
  
 **public** String getUsername() {  
 **return username**;  
 }  
  
 **public void** setUsername(String username) {  
 **this**.**username** = username;  
 }  
  
 **public** String getPassword() {  
 **return password**;  
 }  
  
 **public void** setPassword(String password) {  
 **this**.**password** = password;  
 }  
  
 *//need default constructor for JSON Parsing* **public** AuthenticationRequest()  
 {  
  
 }  
  
 **public** AuthenticationRequest(String username, String password) {  
 **this**.setUsername(username);  
 **this**.setPassword(password);  
 }  
}

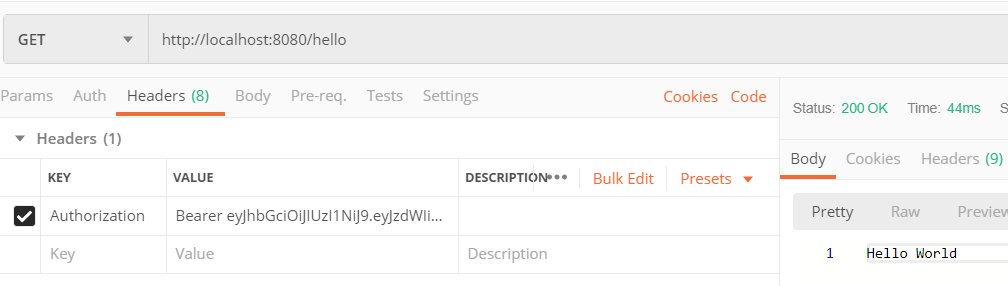
…

**package** io.javabrains.springsecurityjwt.models;  
  
**import** java.io.Serializable;  
  
**public class** AuthenticationResponse **implements** Serializable {  
  
 **private final** String **jwt**;  
  
 **public** AuthenticationResponse(String jwt) {  
 **this**.**jwt** = jwt;  
 }  
  
 **public** String getJwt() {  
 **return jwt**;  
 }  
}

…

**package** io.javabrains.springsecurityjwt;  
  
**import** org.junit.Test;  
**import** org.junit.runner.RunWith;  
**import** org.springframework.boot.test.context.SpringBootTest;  
**import** org.springframework.test.context.junit4.SpringRunner;  
  
@RunWith(SpringRunner.**class**)  
@SpringBootTest  
**public class** SpringSecurityJwtApplicationTests {  
  
 @Test  
 **public void** contextLoads() { }  
}

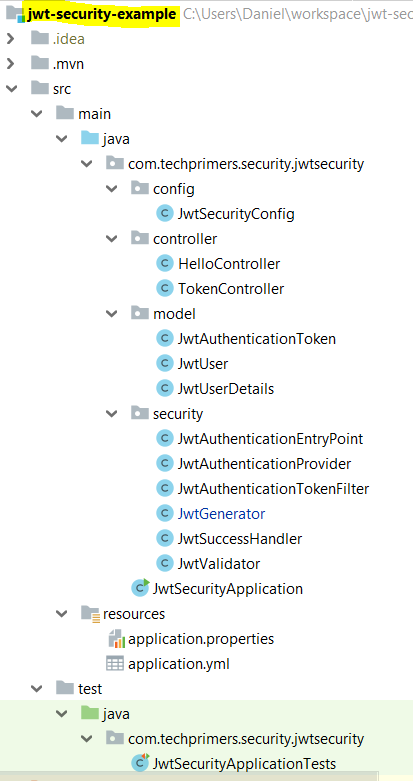




**pom.xml**

*<?***xml version="1.0" encoding="UTF-8"***?>*<**project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd"**>  
 <**modelVersion**>4.0.0</**modelVersion**>  
 <**parent**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-parent</**artifactId**>  
 <**version**>2.1.9.RELEASE</**version**>  
 <**relativePath**/> *<!-- lookup parent from repository -->* </**parent**>  
 <**groupId**>io.javabrains</**groupId**>  
 <**artifactId**>spring-security-jwt</**artifactId**>  
 <**version**>0.0.1-SNAPSHOT</**version**>  
 <**name**>spring-security-jwt</**name**>  
 <**description**>Demo project for Spring Boot</**description**>  
  
 <**properties**>  
 <**java.version**>1.8</**java.version**>  
 </**properties**>  
  
 <**dependencies**>  
 <**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-security</**artifactId**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-web</**artifactId**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>io.jsonwebtoken</**groupId**>  
 <**artifactId**>jjwt</**artifactId**>  
 <**version**>0.9.1</**version**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>javax.xml.bind</**groupId**>  
 <**artifactId**>jaxb-api</**artifactId**>  
 <**version**>2.3.0</**version**>  
 </**dependency**>  
  
 <**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-test</**artifactId**>  
 <**scope**>test</**scope**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>org.springframework.security</**groupId**>  
 <**artifactId**>spring-security-test</**artifactId**>  
 <**scope**>test</**scope**>  
 </**dependency**>  
 </**dependencies**>  
  
 <**build**>  
 <**plugins**>  
 <**plugin**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-maven-plugin</**artifactId**>  
 </**plugin**>  
 </**plugins**>  
 </**build**>  
  
</**project**>

PROJECT 2 --- Tech Primers



**package** com.techprimers.security.jwtsecurity.controller;  
  
  
**import** org.springframework.web.bind.annotation.GetMapping;  
**import** org.springframework.web.bind.annotation.RequestMapping;  
**import** org.springframework.web.bind.annotation.RestController;  
  
@RestController  
@RequestMapping(**"/rest/hello"**)  
**public class** HelloController {  
  
 @GetMapping  
 **public** String hello() {  
 **return "Hello World"**;  
 }  
}

…

**package** com.techprimers.security.jwtsecurity.controller;  
  
**import** com.techprimers.security.jwtsecurity.model.JwtUser;  
**import** com.techprimers.security.jwtsecurity.security.JwtGenerator;  
**import** org.springframework.web.bind.annotation.PostMapping;  
**import** org.springframework.web.bind.annotation.RequestBody;  
**import** org.springframework.web.bind.annotation.RequestMapping;  
**import** org.springframework.web.bind.annotation.RestController;  
  
@RestController  
@RequestMapping(**"/token"**)  
**public class** TokenController {  
  
  
 **private** JwtGenerator **jwtGenerator**;  
  
 **public** TokenController(JwtGenerator jwtGenerator) {  
 **this**.**jwtGenerator** = jwtGenerator;  
 }  
  
 @PostMapping  
 **public** String generate(@RequestBody **final** JwtUser jwtUser) {  
  
 **return jwtGenerator**.generate(jwtUser);  
  
 }  
}

…

**package** com.techprimers.security.jwtsecurity.security;  
  
**import** org.springframework.security.core.AuthenticationException;  
**import** org.springframework.security.web.AuthenticationEntryPoint;  
**import** org.springframework.stereotype.Component;  
  
**import** javax.servlet.ServletException;  
**import** javax.servlet.http.HttpServletRequest;  
**import** javax.servlet.http.HttpServletResponse;  
**import** java.io.IOException;  
  
@Component  
**public class** JwtAuthenticationEntryPoint **implements** AuthenticationEntryPoint {  
 @Override  
 **public void** commence(HttpServletRequest httpServletRequest, HttpServletResponse httpServletResponse, AuthenticationException e) **throws** IOException, ServletException {  
  
 httpServletResponse.sendError(HttpServletResponse.***SC\_UNAUTHORIZED***, **"UNAUTHORIZED"**);  
 }  
}

…

**package** com.techprimers.security.jwtsecurity.security;  
  
**import** com.techprimers.security.jwtsecurity.model.JwtAuthenticationToken;  
**import** com.techprimers.security.jwtsecurity.model.JwtUser;  
**import** com.techprimers.security.jwtsecurity.model.JwtUserDetails;  
**import** org.springframework.beans.factory.annotation.Autowired;  
**import** org.springframework.security.authentication.UsernamePasswordAuthenticationToken;  
**import** org.springframework.security.authentication.dao.AbstractUserDetailsAuthenticationProvider;  
**import** org.springframework.security.core.AuthenticationException;  
**import** org.springframework.security.core.GrantedAuthority;  
**import** org.springframework.security.core.authority.AuthorityUtils;  
**import** org.springframework.security.core.userdetails.UserDetails;  
**import** org.springframework.stereotype.Component;  
  
**import** java.util.List;  
  
@Component  
**public class** JwtAuthenticationProvider **extends** AbstractUserDetailsAuthenticationProvider {  
  
 @Autowired  
 **private** JwtValidator **validator**;  
  
 @Override  
 **protected void** additionalAuthenticationChecks(UserDetails userDetails, UsernamePasswordAuthenticationToken usernamePasswordAuthenticationToken) **throws** AuthenticationException {  
  
 }  
  
 @Override  
 **protected** UserDetails retrieveUser(String username, UsernamePasswordAuthenticationToken usernamePasswordAuthenticationToken) **throws** AuthenticationException {  
  
 JwtAuthenticationToken jwtAuthenticationToken = (JwtAuthenticationToken) usernamePasswordAuthenticationToken;  
 String token = jwtAuthenticationToken.getToken();  
  
 JwtUser jwtUser = **validator**.validate(token);  
  
 **if** (jwtUser == **null**) {  
 **throw new** RuntimeException(**"JWT Token is incorrect"**);  
 }  
  
 List<GrantedAuthority> grantedAuthorities = AuthorityUtils  
 .*commaSeparatedStringToAuthorityList*(jwtUser.getRole());  
 **return new** JwtUserDetails(jwtUser.getUserName(), jwtUser.getId(),  
 token,  
 grantedAuthorities);  
 }  
  
 @Override  
 **public boolean** supports(Class<?> aClass) {  
 **return** (JwtAuthenticationToken.**class**.isAssignableFrom(aClass));  
 }  
}

…

**package** com.techprimers.security.jwtsecurity.security;  
  
**import** com.techprimers.security.jwtsecurity.model.JwtAuthenticationToken;  
**import** org.springframework.security.core.Authentication;  
**import** org.springframework.security.core.AuthenticationException;  
**import** org.springframework.security.web.authentication.AbstractAuthenticationProcessingFilter;  
  
**import** javax.servlet.FilterChain;  
**import** javax.servlet.ServletException;  
**import** javax.servlet.http.HttpServletRequest;  
**import** javax.servlet.http.HttpServletResponse;  
**import** java.io.IOException;  
  
**public class** JwtAuthenticationTokenFilter **extends** AbstractAuthenticationProcessingFilter {  
  
 **public** JwtAuthenticationTokenFilter() {  
 **super**(**"/rest/\*\*"**);  
 }  
  
 @Override  
 **public** Authentication attemptAuthentication(HttpServletRequest httpServletRequest, HttpServletResponse httpServletResponse) **throws** AuthenticationException, IOException, ServletException {  
  
 String header = httpServletRequest.getHeader(**"Authorisation"**);  
  
  
 **if** (header == **null** || !header.startsWith(**"Token "**)) {  
 **throw new** RuntimeException(**"JWT Token is missing"**);  
 }  
  
 String authenticationToken = header.substring(6);  
  
 JwtAuthenticationToken token = **new** JwtAuthenticationToken(authenticationToken);  
 **return** getAuthenticationManager().authenticate(token);  
 }  
  
  
 @Override  
 **protected void** successfulAuthentication(HttpServletRequest request, HttpServletResponse response, FilterChain chain, Authentication authResult) **throws** IOException, ServletException {  
 **super**.successfulAuthentication(request, response, chain, authResult);  
 chain.doFilter(request, response);  
 }  
}

…

**package** com.techprimers.security.jwtsecurity.security;  
  
**import** com.techprimers.security.jwtsecurity.model.JwtUser;  
**import** io.jsonwebtoken.Claims;  
**import** io.jsonwebtoken.Jwts;  
**import** io.jsonwebtoken.SignatureAlgorithm;  
**import** org.springframework.stereotype.Component;  
  
@Component  
**public class** JwtGenerator {  
  
  
 **public** String generate(JwtUser jwtUser) {  
  
  
 Claims claims = Jwts.*claims*()  
 .setSubject(jwtUser.getUserName());  
 claims.put(**"userId"**, String.*valueOf*(jwtUser.getId()));  
 claims.put(**"role"**, jwtUser.getRole());  
  
  
 **return** Jwts.*builder*()  
 .setClaims(claims)  
 .signWith(SignatureAlgorithm.***HS256***, **"youtube"**)  
 .compact();  
 }  
}

…

**package** com.techprimers.security.jwtsecurity.security;  
  
  
**import** org.springframework.security.core.Authentication;  
**import** org.springframework.security.web.authentication.AuthenticationSuccessHandler;  
  
**import** javax.servlet.ServletException;  
**import** javax.servlet.http.HttpServletRequest;  
**import** javax.servlet.http.HttpServletResponse;  
**import** java.io.IOException;  
  
**public class** JwtSuccessHandler **implements** AuthenticationSuccessHandler{  
 @Override  
 **public void** onAuthenticationSuccess(HttpServletRequest httpServletRequest, HttpServletResponse httpServletResponse, Authentication authentication) **throws** IOException, ServletException {  
 System.***out***.println(**"Successfully Authentication"**);  
 }  
}

…

**package** com.techprimers.security.jwtsecurity.security;  
  
**import** com.techprimers.security.jwtsecurity.model.JwtUser;  
**import** io.jsonwebtoken.Claims;  
**import** io.jsonwebtoken.Jwts;  
**import** org.springframework.stereotype.Component;  
  
@Component  
**public class** JwtValidator {  
  
  
 **private** String **secret** = **"youtube"**;  
  
 **public** JwtUser validate(String token) {  
  
 JwtUser jwtUser = **null**;  
 **try** {  
 Claims body = Jwts.*parser*()  
 .setSigningKey(**secret**)  
 .parseClaimsJws(token)  
 .getBody();  
  
 jwtUser = **new** JwtUser();  
  
 jwtUser.setUserName(body.getSubject());  
 jwtUser.setId(Long.*parseLong*((String) body.get(**"userId"**)));  
 jwtUser.setRole((String) body.get(**"role"**));  
 }  
 **catch** (Exception e) {  
 System.***out***.println(e);  
 }  
  
 **return** jwtUser;  
 }  
}

…

**package** com.techprimers.security.jwtsecurity;  
  
**import** org.springframework.boot.SpringApplication;  
**import** org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
**public class** JwtSecurityApplication {  
  
 **public static void** main(String[] args) {  
 SpringApplication.*run*(JwtSecurityApplication.**class**, args);  
 }  
}

…

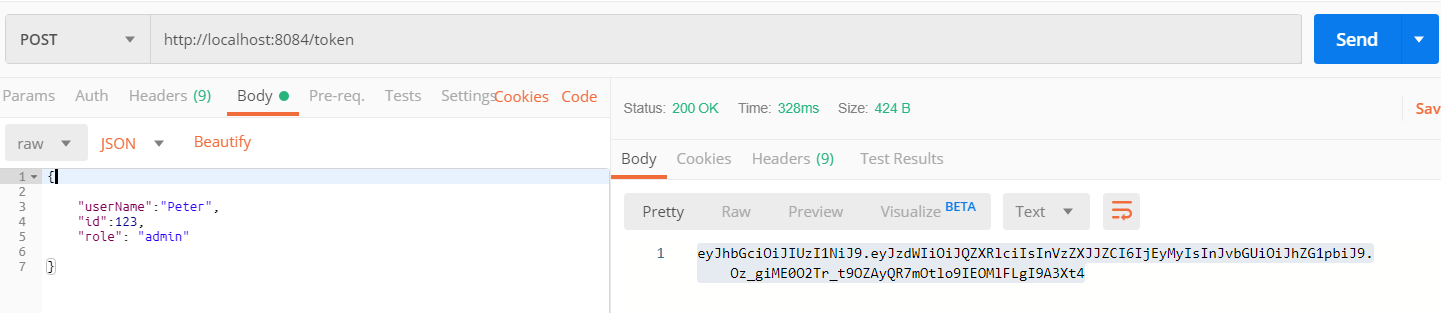
application.properties

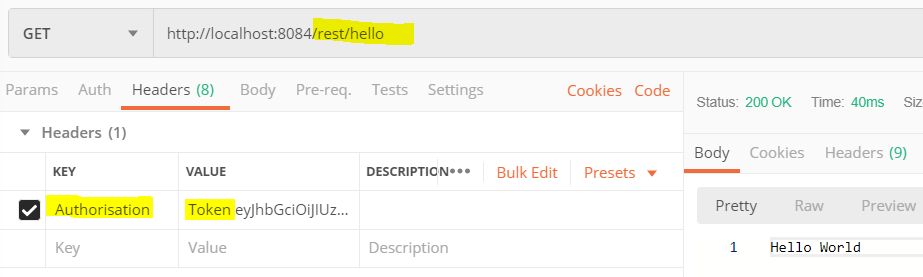
**server.port** = **8084**

…

application.yml

**management**:  
 **security**:  
 **enabled**: true  
  
  
**spring**:  
 **resources**:  
 **chain**:  
 **enabled**: true





**pom.xml**

*<?***xml version="1.0" encoding="UTF-8"***?>*<**project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"**>  
 <**modelVersion**>4.0.0</**modelVersion**>  
  
 <**groupId**>com.techprimers.security</**groupId**>  
 <**artifactId**>jwt-security</**artifactId**>  
 <**version**>0.0.1-SNAPSHOT</**version**>  
 <**packaging**>jar</**packaging**>  
  
 <**name**>jwt-security</**name**>  
 <**description**>Demo project for Spring Boot</**description**>  
  
 <**parent**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-parent</**artifactId**>  
 <**version**>1.5.6.RELEASE</**version**>  
 <**relativePath**/> *<!-- lookup parent from repository -->* </**parent**>  
  
 <**properties**>  
 <**project.build.sourceEncoding**>UTF-8</**project.build.sourceEncoding**>  
 <**project.reporting.outputEncoding**>UTF-8</**project.reporting.outputEncoding**>  
 <**java.version**>1.8</**java.version**>  
 </**properties**>  
  
 <**dependencies**>  
 <**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-security</**artifactId**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-web</**artifactId**>  
 </**dependency**>  
  
 <**dependency**>  
 <**groupId**>io.jsonwebtoken</**groupId**>  
 <**artifactId**>jjwt</**artifactId**>  
 <**version**>0.6.0</**version**>  
 </**dependency**>  
  
 <**dependency**>  
 <**groupId**>javax.xml.bind</**groupId**>  
 <**artifactId**>jaxb-api</**artifactId**>  
 <**version**>2.3.0</**version**>  
 </**dependency**>  
  
  
 <**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-test</**artifactId**>  
 <**scope**>test</**scope**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>org.springframework.security</**groupId**>  
 <**artifactId**>spring-security-test</**artifactId**>  
 <**scope**>test</**scope**>  
 </**dependency**>  
 </**dependencies**>  
  
 <**build**>  
 <**plugins**>  
 <**plugin**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-maven-plugin</**artifactId**>  
 </**plugin**>  
 </**plugins**>  
 </**build**>  
  
  
</**project**>