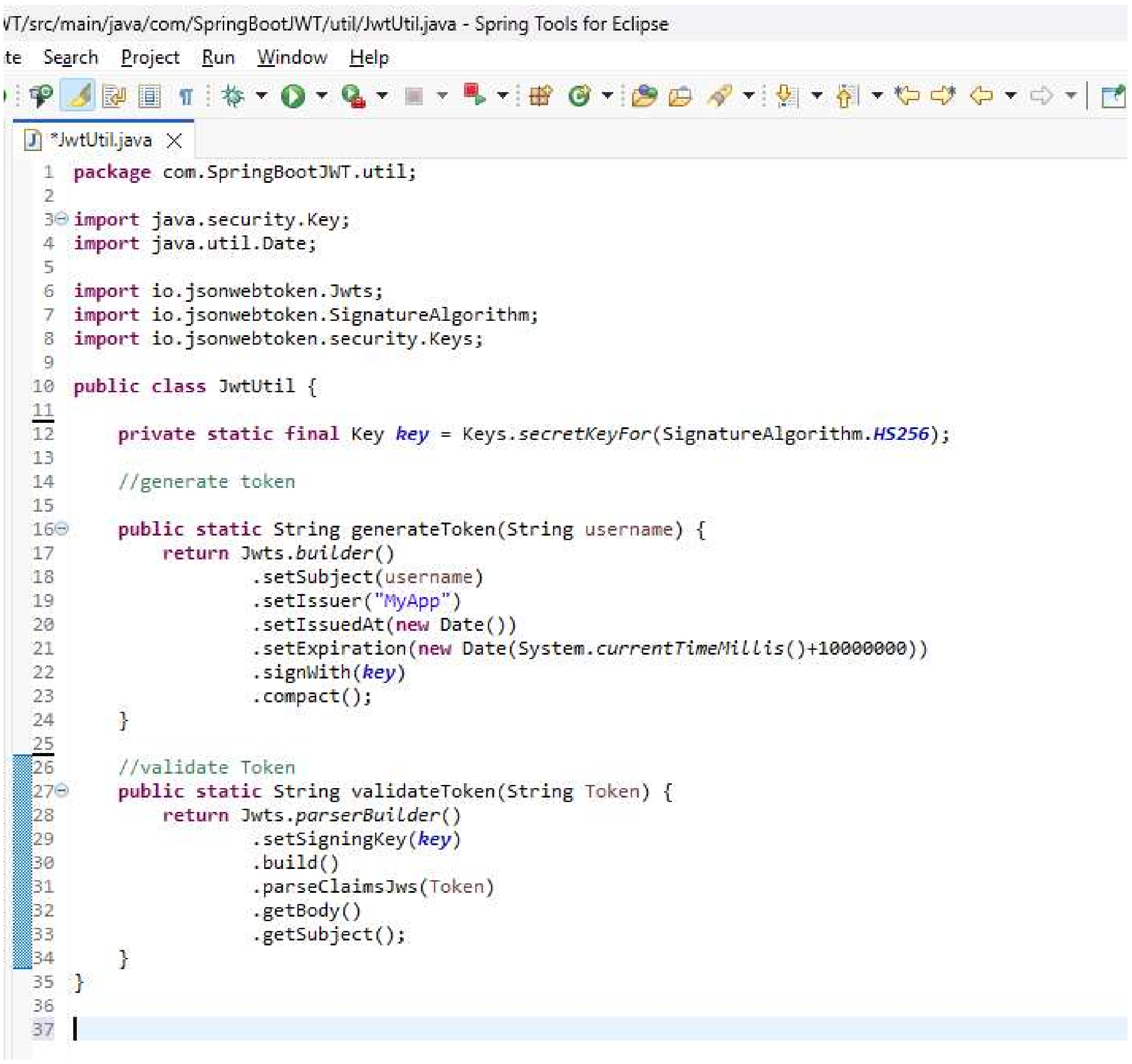
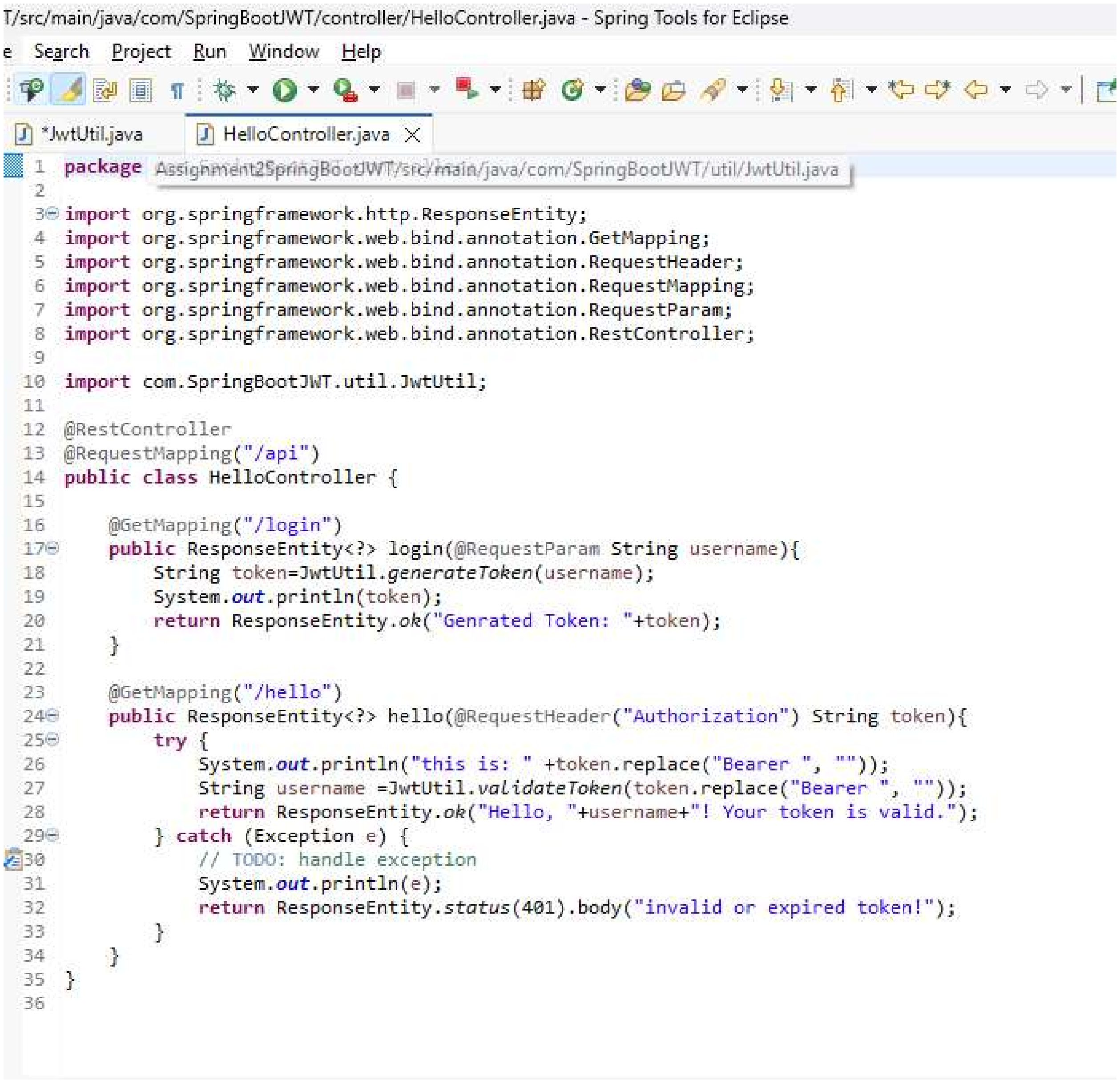
Rudraksh Chourey 10/8/2025

JWT Util code:


VT/src/main/java/com/SpringBootJWT/util/JwtUtil.java - Spring Tools for Eclipse
te Search Project Run Window Help
*JwtUtil.java

```java
package com.SpringBootJWT.util;

import java.security.Key;
import java.util.Date;

import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.SignatureAlgorithm;
import io.jsonwebtoken.security.Keys;

public class JwtUtil {

    private static final Key key = Keys.secretKeyFor(SignatureAlgorithm.HS256);

    //generate token

    public static String generateToken(String username) {
        return Jwts.builder()
                .setSubject(username)
                .setIssuer("MyApp")
                .setIssuedAt(new Date())
                .setExpiration(new Date(System.currentTimeMillis()+10000000))
                .signWith(key)
                .compact();
    }

    //validate Token
    public static String validateToken(String Token) {
        return Jwts.parserBuilder()
                .setSigningKey(key)
                .build()
                .parseClaimsJws(Token)
                .getBody()
                .getSubject();
    }
}

```

Controller Code:

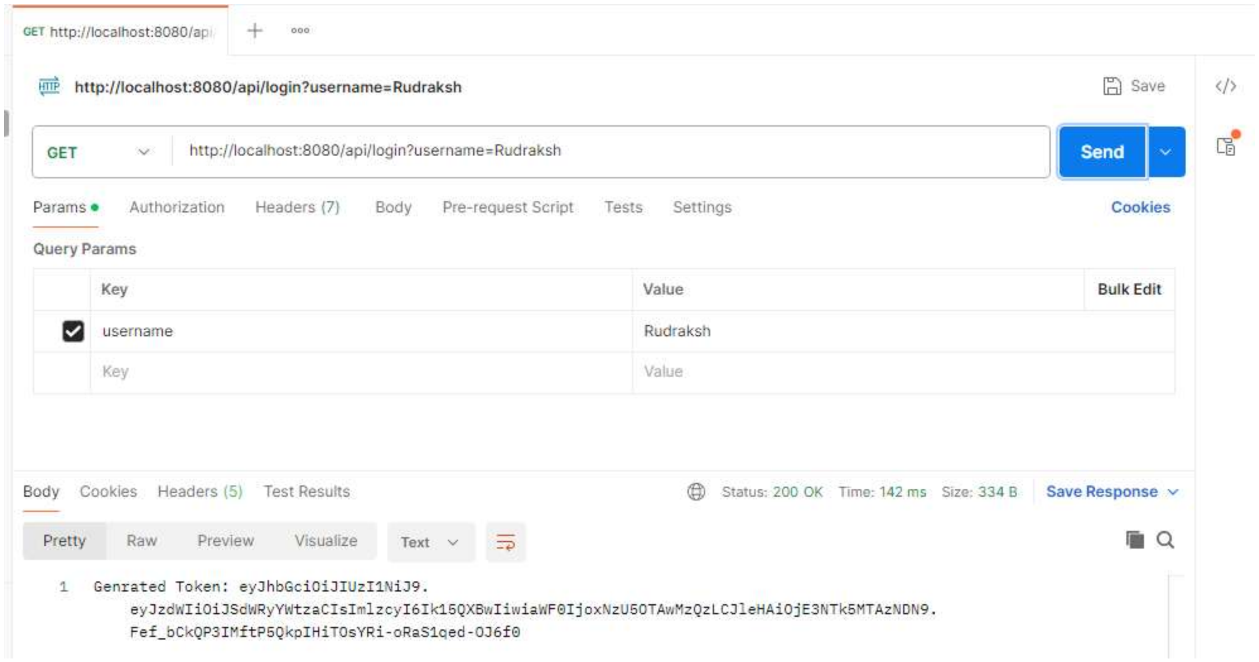


T/src/main/java/com/SpringBootJWT/controller/HelloController.java - Spring Tools for Eclipse

```java
package Assignment2SpringBootJWT/src/main/java/com/SpringBootJWT/util/JwtUtil.java

import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestHeader;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import com.SpringBootJWT.util.JwtUtil;

@RestController
@RequestMapping("/api")
public class HelloController {

    @GetMapping("/login")
    public ResponseEntity<?> login(@RequestParam String username){
        String token=JwtUtil.generateToken(username);
        System.out.println(token);
        return ResponseEntity.ok("Genrated Token: "+token);
    }

    @GetMapping("/hello")
    public ResponseEntity<?> hello(@RequestHeader("Authorization") String token){
        try {
            System.out.println("this is: " +token.replace("Bearer ", ""));
            String username =JwtUtil.validateToken(token.replace("Bearer ", ""));
            return ResponseEntity.ok("Hello, "+username+"! Your token is valid.");
        } catch (Exception e) {
            // TODO: handle exception
            System.out.println(e);
            return ResponseEntity.status(401).body("invalid or expired token!");
        }
    }
}
```

Output:

Token generation:



GET http://localhost:8080/api/login?username=Rudraksh

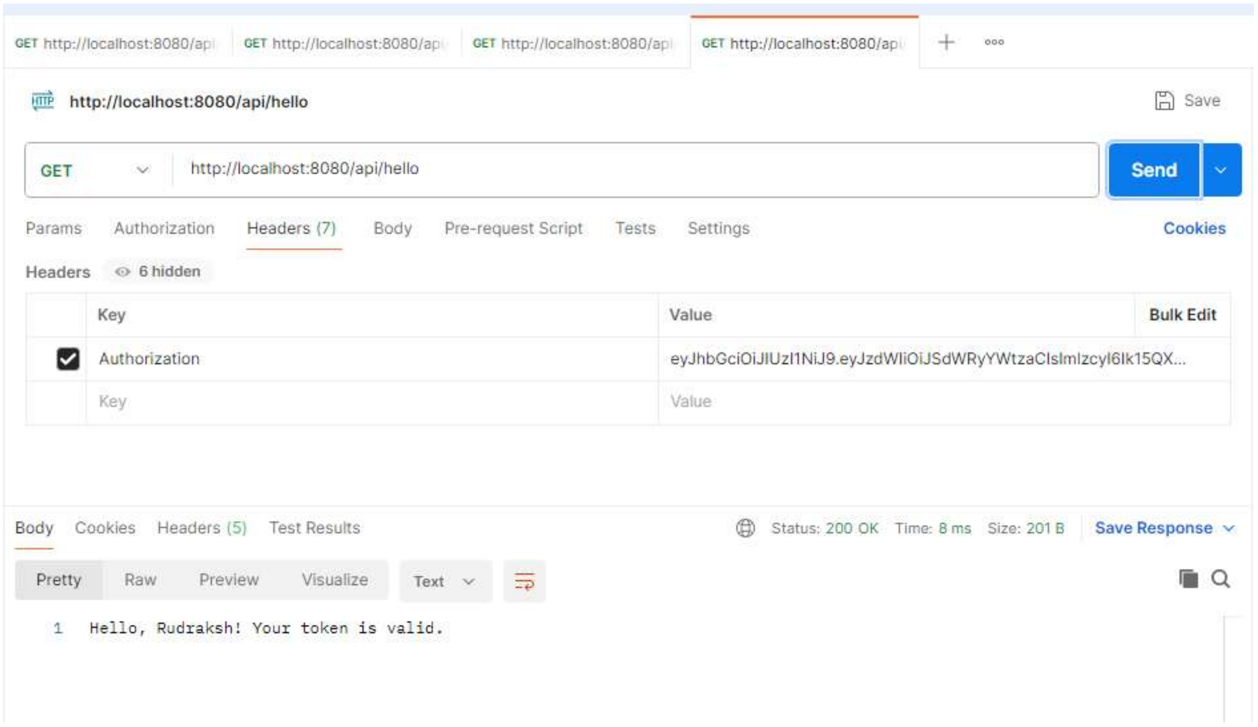
Query Params

Key	Value
username	Rudraksh

Status: 200 OK Time: 142 ms Size: 334 B

```
Genrated Token: eyJhbGciOiJIUzI1NiJ9.
    eyJzdWIiOiJSdWRyYWtzaCIsImlzcyI6Ik15QXBwIiwiaWF0IjoxNzU5OTAwMzQzLCJleHAiOjE3NTk5MTAzNDN9.
    Fef_bCkQP3IMftP5QkpIHiTOsYRi-oRaS1qed-OJ6f0
```

Token Verification:



GET http://localhost:8080/api/hello

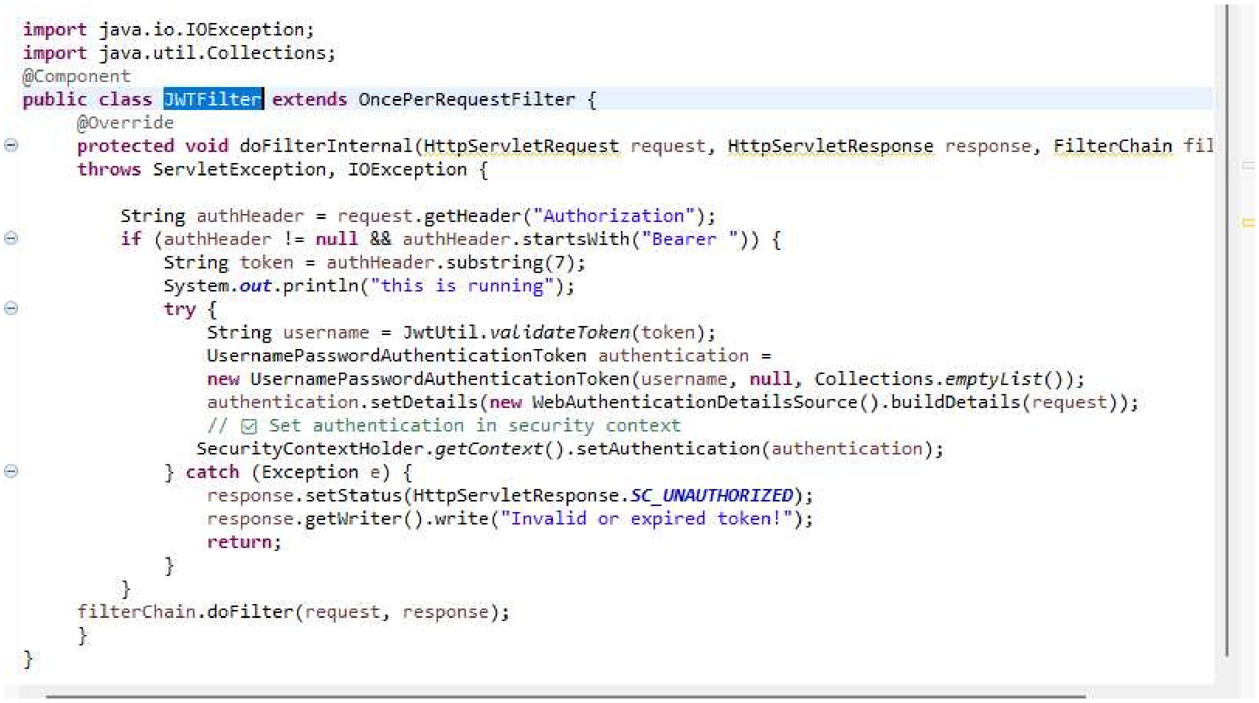
Headers

Key	Value
Authorization	eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJSdWRyYWtzaCIsImlzcyI6Ik15QX...

Status: 200 OK Time: 8 ms Size: 201 B

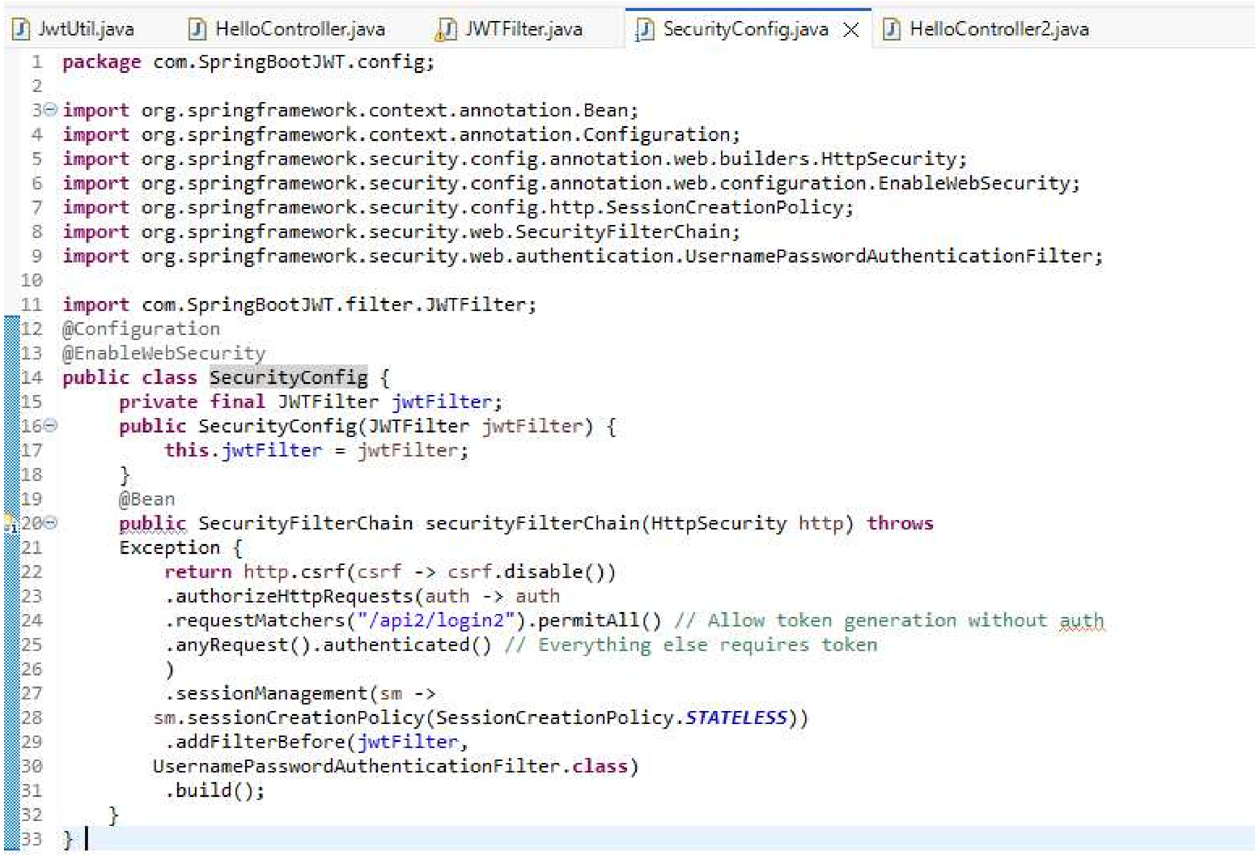
```
Hello, Rudraksh! Your token is valid.
```

JWT FILTER:



```
import java.io.IOException;
import java.util.Collections;
@Component
public class JWTFilter extends OncePerRequestFilter {
     @Override
     protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain fil
     throws ServletException, IOException {

         String authHeader = request.getHeader("Authorization");
         if (authHeader != null && authHeader.startsWith("Bearer ")) {
             String token = authHeader.substring(7);
             System.out.println("this is running");
             try {
                 String username = JwtUtil.validateToken(token);
                 UsernamePasswordAuthenticationToken authentication =
                 new UsernamePasswordAuthenticationToken(username, null, Collections.emptyList());
                 authentication.setDetails(new WebAuthenticationDetailsSource().buildDetails(request));
                 // ☑ Set authentication in security context
                SecurityContextHolder.getContext().setAuthentication(authentication);
             } catch (Exception e) {
                 response.setStatus(HttpServletResponse.SC_UNAUTHORIZED);
                 response.getWriter().write("Invalid or expired token!");
                 return;
             }
         }
     filterChain.doFilter(request, response);
     }
}
```

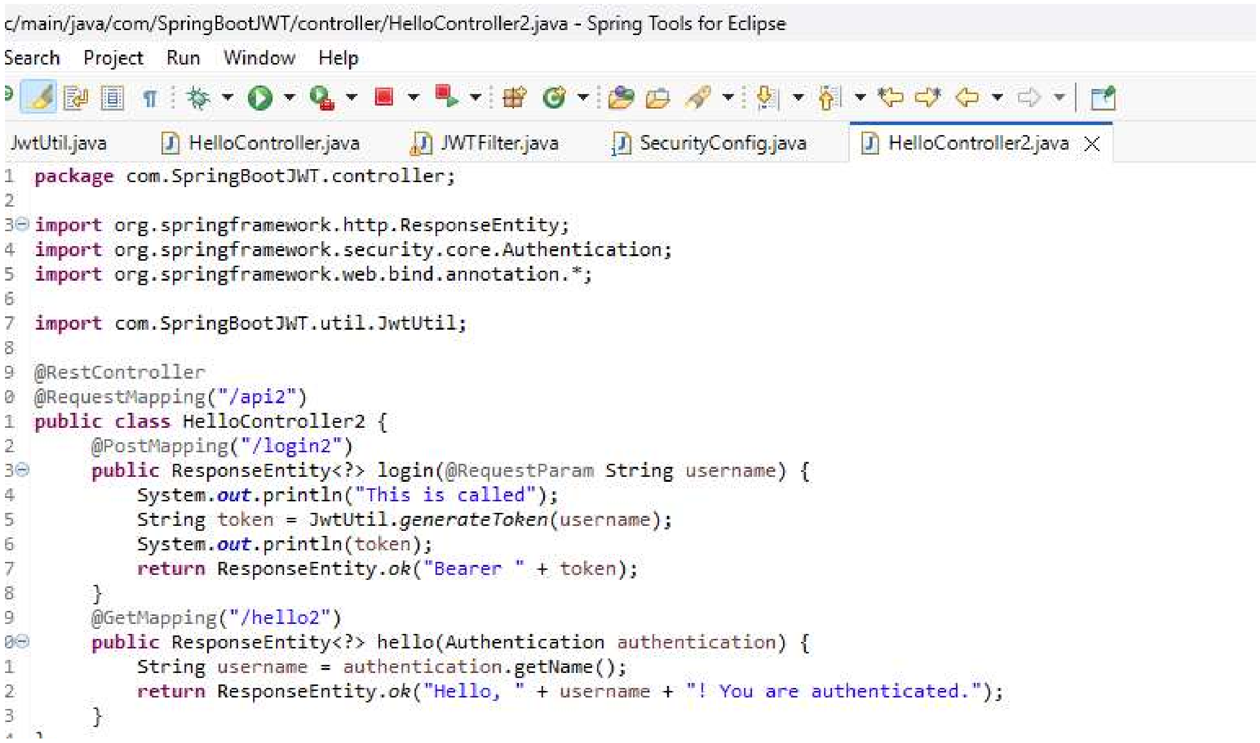
JWT security config :



JwtUtil.java | HelloController.java | JWTFilter.java | SecurityConfig.java | HelloController2.java

```
package com.SpringBootJWT.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
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.web.SecurityFilterChain;
import org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;

import com.SpringBootJWT.filter.JWTFilter;
@Configuration
@EnableWebSecurity
public class SecurityConfig {
     private final JWTFilter jwtFilter;
     public SecurityConfig(JWTFilter jwtFilter) {
         this.jwtFilter = jwtFilter;
     }
     @Bean
     public SecurityFilterChain securityFilterChain(HttpSecurity http) throws
     Exception {
         return http.csrf(csrf -> csrf.disable())
         .authorizeHttpRequests(auth -> auth
         .requestMatchers("/api2/login2").permitAll() // Allow token generation without auth
         .anyRequest().authenticated() // Everything else requires token
         )
         .sessionManagement(sm ->
        sm.sessionCreationPolicy(SessionCreationPolicy.STATELESS))
         .addFilterBefore(jwtFilter,
        UsernamePasswordAuthenticationFilter.class)
         .build();
    }
}
```

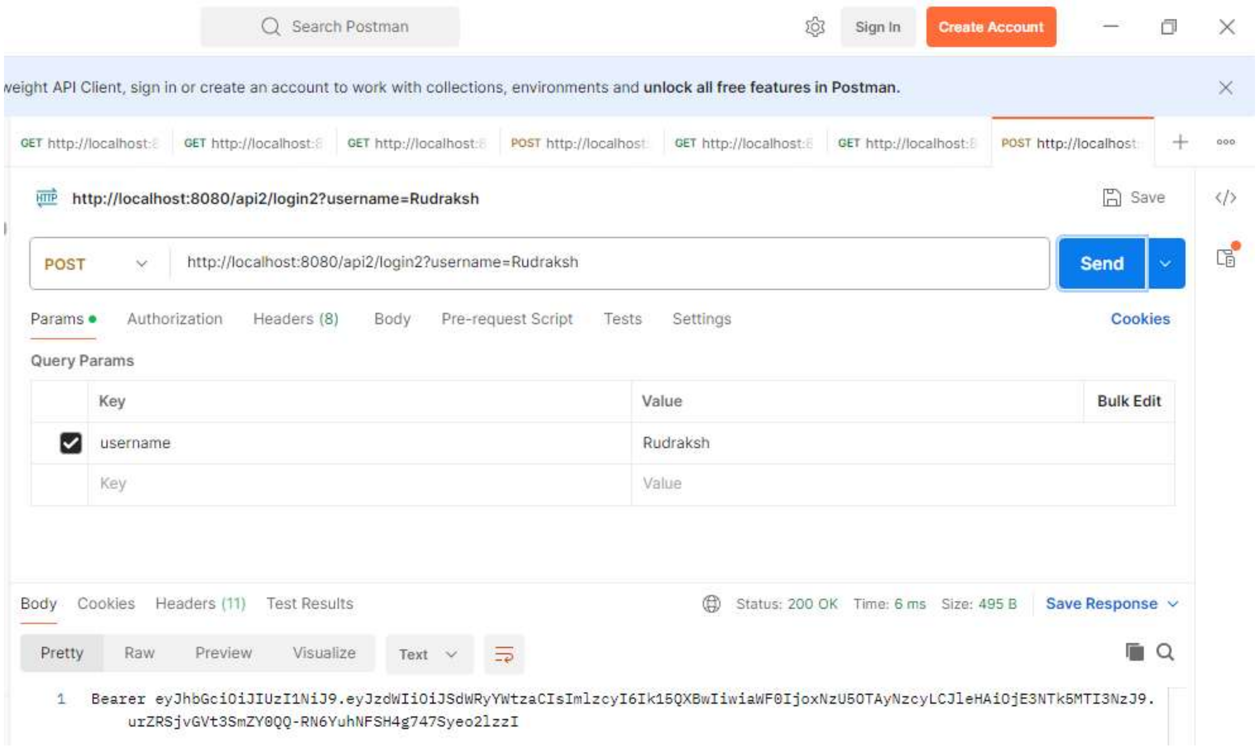
Controller for filter:



c/main/java/com/SpringBootJWT/controller/HelloController2.java - Spring Tools for Eclipse

```java
package com.SpringBootJWT.controller;

import org.springframework.http.ResponseEntity;
import org.springframework.security.core.Authentication;
import org.springframework.web.bind.annotation.*;

import com.SpringBootJWT.util.JwtUtil;

@RestController
@RequestMapping("/api2")
public class HelloController2 {
    @PostMapping("/login2")
    public ResponseEntity<?> login(@RequestParam String username) {
        System.out.println("This is called");
        String token = JwtUtil.generateToken(username);
        System.out.println(token);
        return ResponseEntity.ok("Bearer " + token);
    }
    @GetMapping("/hello2")
    public ResponseEntity<?> hello(Authentication authentication) {
        String username = authentication.getName();
        return ResponseEntity.ok("Hello, " + username + "! You are authenticated.");
    }
```

Output:



POST http://localhost:8080/api2/login2?username=Rudraksh

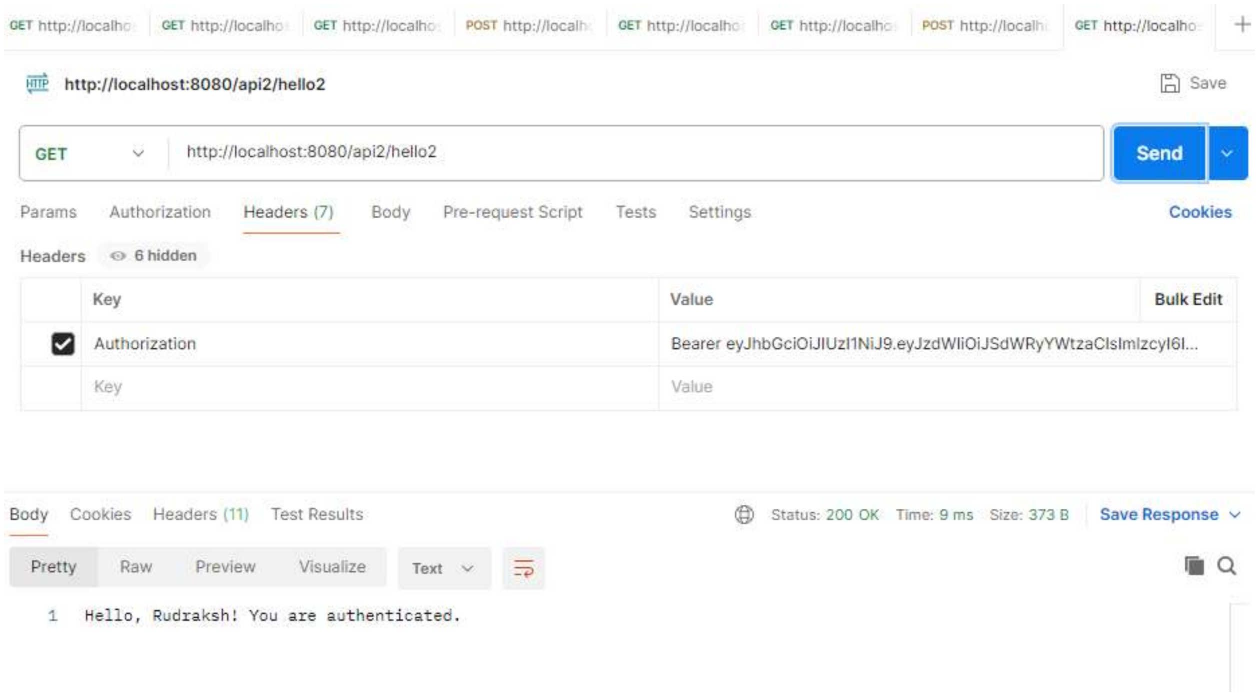
Query Params

Key	Value
username	Rudraksh

Status: 200 OK Time: 6 ms Size: 495 B

```
Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJSdWRyYWtzaCIsImlzcyI6Ik15QXBwIiwiaWF0IjoxNzU5OTAyNzcyLCJleHAiOjE3NTk5MTI3NzJ9.
   urZRSjvGVt3SmZY0QQ-RN6YuhNFSH4g747Syeo2lzzI
```

Authentication:



GET http://localhost:8080/api2/hello2

Headers

Key	Value
Authorization	Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJSdWRyYWtzaCIsImlzcyI6I...

Status: 200 OK Time: 9 ms Size: 373 B

```
Hello, Rudraksh! You are authenticated.
```