Java Spring Boot Restful – OAuth2 - Getting Started with JWT

Add this piece of code in SecurityConfig.java

```
.oauth2ResourceServer(oauth2 -> oauth2
     .jwt(Customizer.withDefaults()))
     .sessionManagement(session -> session
     .sessionCreationPolicy(SessionCreationPolicy.STATELESS));
```

```
  @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws
Exception {
        http
                .authorizeHttpRequests(customizer -> customizer
                        .requestMatchers("/token").permitAll()
                        .requestMatchers("/").permitAll()
                        .requestMatchers("/swagger-ui/**").permitAll()
                        .requestMatchers("/v3/api-docs/**").permitAll()
                        .requestMatchers("/test").authenticated())
                .oauth2ResourceServer(oauth2 -> oauth2
                        .jwt(Customizer.withDefaults()))
                .sessionManagement(session -> session
                        .sessionCreationPolicy(SessionCreationPolicy.STATELESS));

        return http.build();
    }
```

This means enabling the OAuth resource server.

This means our backend will act as a OAuth server and it will be give us a token whenever there is requirement.

After executing, you get below error

Consider defining a bean of type 'org.springframework.security.oauth2.jwt.JwtDecoder' in your configuration.

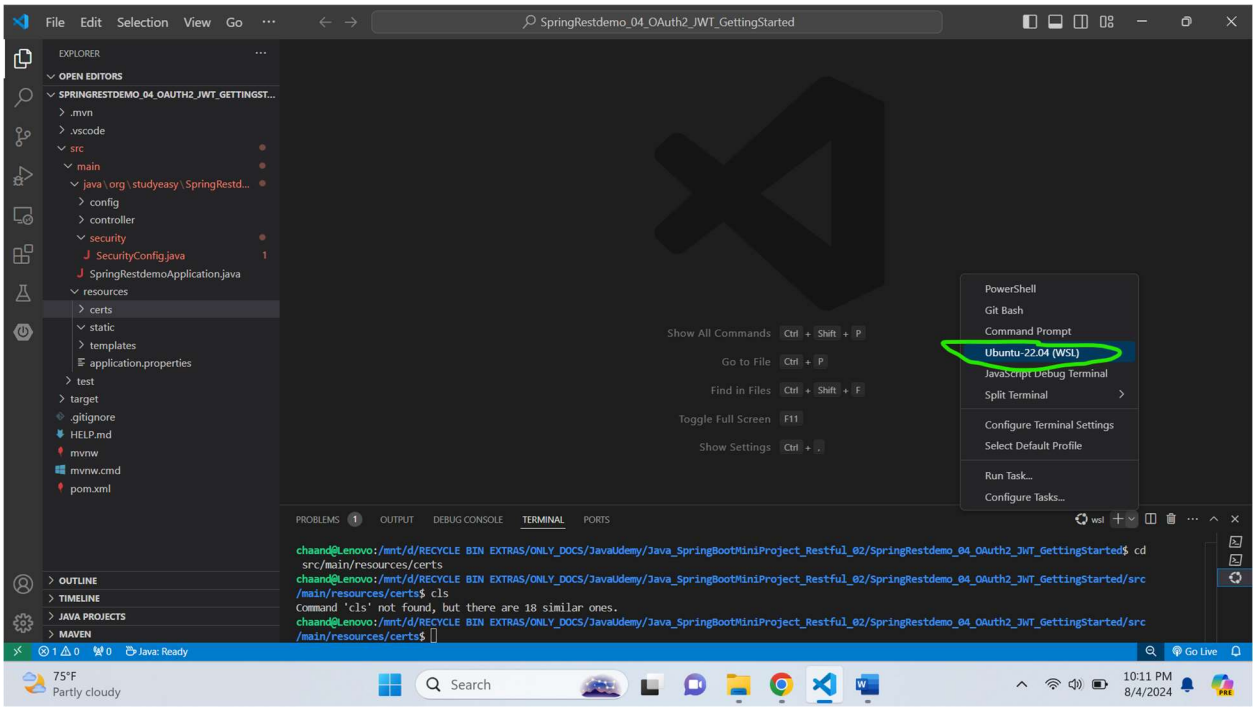
Java decoder

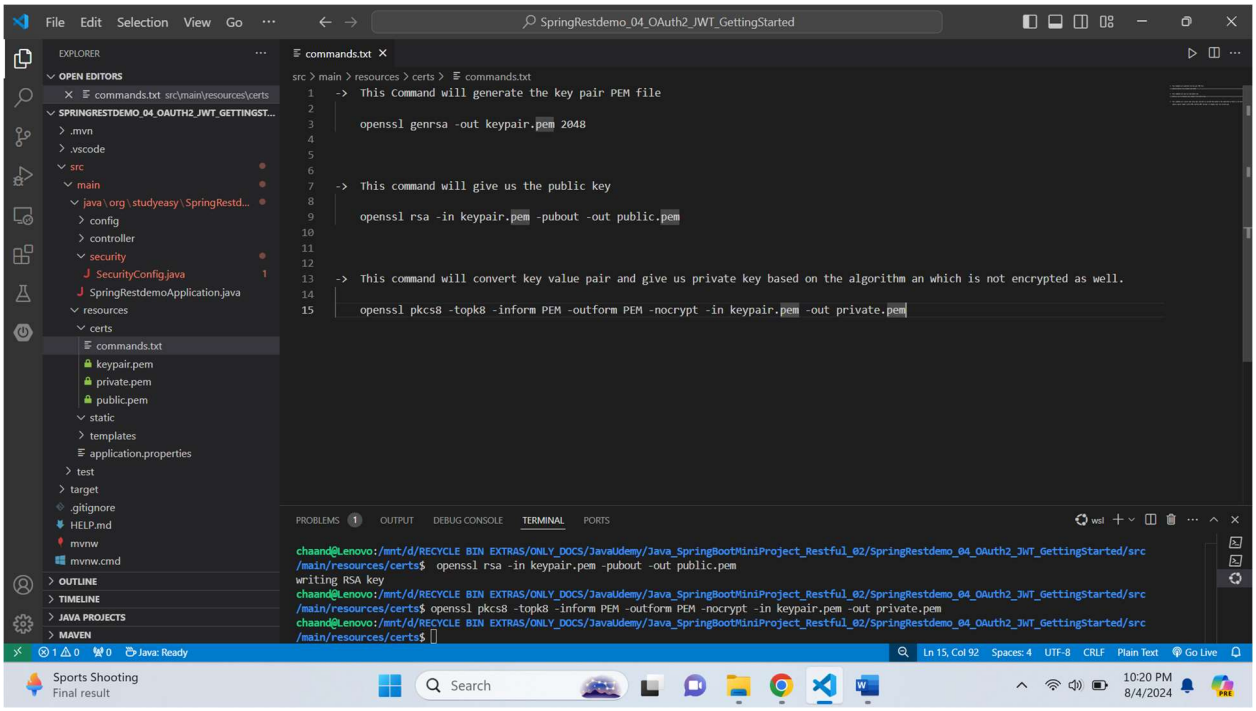
```
@Bean
 JwtDecoder jwtDecoder() {
    return NimbusJwtDecoder.withPublicKey(rsaKeys.publicKey()).build();
 }
```

Rsa keys

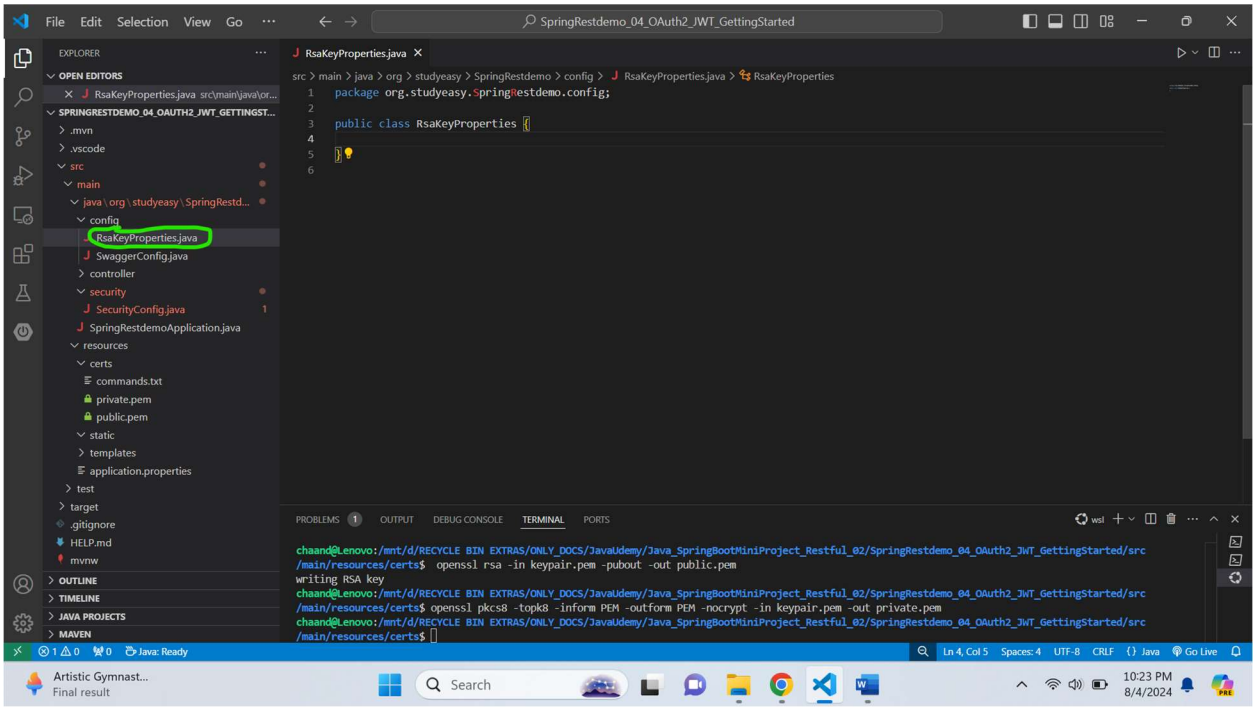
Create a file in the resources folder -> certs

chaand@Lenovo:/mnt/d/RECYCLE BIN EXTRAS/ONLY_DOCS/JavaUdemy/Java_SpringBootMiniProject_Restful_02/SpringRestdemo_04_OAuth2_JWT_GettingStarted$ cd src/main/resources/certs





RsaKeyProperties.java



RsaKeyProperties.java

```
package org.studyeasy.SpringRestdemo.config;

import org.springframework.boot.context.properties.ConfigurationProperties;

import java.security.interfaces.RSAPrivateKey;
import java.security.interfaces.RSAPublicKey;

@ConfigurationProperties(prefix = "rsa")
public record RsaKeyProperties(RSAPublicKey publicKey, RSAPrivateKey privateKey)
{
}
```

Initialize the RSA Key -> Add this code in the SecurityConfig.java

```
 private final RsaKeyProperties rsaKeys;

 public SecurityConfig(RsaKeyProperties rsaKeys) {
       this.rsaKeys = rsaKeys;
 }
```

Output

Consider defining a bean of type 'org.studyeasy.SpringRestdemo.config.RsaKeyProperties' in your configuration.

Add an encoder

```
  @Bean
  JwtEncoder jwtEncoder() {
      JWK jwk = new
RSAKey.Builder(rsaKeys.publicKey()).privateKey(rsaKeys.privateKey()).build();
      JWKSource<SecurityContext> jwks = new ImmutableJWKSet<>(new JWKSet(jwk));
      return new NimbusJwtEncoder(jwks);
  }
```

Authentication Manager is handled by spring.

Final Source Code of SecurityConfig.java

```java
package org.studyeasy.SpringRestdemo.security;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.Customizer;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableWebSecurit
y;
import org.springframework.security.config.http.SessionCreationPolicy;
import org.springframework.security.core.userdetails.User;
import org.springframework.security.oauth2.jwt.JwtDecoder;
import org.springframework.security.oauth2.jwt.JwtEncoder;
import org.springframework.security.oauth2.jwt.NimbusJwtDecoder;
import org.springframework.security.oauth2.jwt.NimbusJwtEncoder;
import org.springframework.security.provisioning.InMemoryUserDetailsManager;
import org.springframework.security.web.SecurityFilterChain;
import org.studyeasy.SpringRestdemo.config.RsaKeyProperties;

import com.nimbusds.jose.jwk.JWK;
import com.nimbusds.jose.jwk.JWKSet;
import com.nimbusds.jose.jwk.RSAKey;
import com.nimbusds.jose.jwk.source.ImmutableJWKSet;
import com.nimbusds.jose.jwk.source.JWKSource;
import com.nimbusds.jose.proc.SecurityContext;

@Configuration
@EnableWebSecurity
public class SecurityConfig {

        private final RsaKeyProperties rsaKeys;

        public SecurityConfig(RsaKeyProperties rsaKeys) {
                this.rsaKeys = rsaKeys;
        }

        @Bean
        public InMemoryUserDetailsManager users() {
                return new InMemoryUserDetailsManager(
                     User.withUsername("chaand")
                           .password("{noop}password")
                           .authorities("read")
                           .build());
        }
```

```
        @Bean
        JwtDecoder jwtDecoder() {
                Return
NimbusJwtDecoder.withPublicKey(rsaKeys.publicKey()).build();
        }

        @Bean
        JwtEncoder jwtEncoder() {
              JWK jwk = new
RSAKey.Builder(rsaKeys.publicKey()).privateKey(rsaKeys.privateKey()).build();
           JWKSource<SecurityContext> jwks = new ImmutableJWKSet<>(new
JWKSet(jwk));
            return new NimbusJwtEncoder(jwks);
        }

        @Bean
        public SecurityFilterChain securityFilterChain(HttpSecurity http) throws
Exception {
                http
                .authorizeHttpRequests(customizer -> customizer
                .requestMatchers("/token").permitAll()
                .requestMatchers("/").permitAll()
                .requestMatchers("/swagger-ui/**").permitAll()
                .requestMatchers("/v3/api-docs/**").permitAll()
                .requestMatchers("/test").authenticated())
                .oauth2ResourceServer(oauth2 -> oauth2
                    .jwt(Customizer.withDefaults()))
                .sessionManagement(session -> session
.sessionCreationPolicy(SessionCreationPolicy.STATELESS));

                return http.build();
        }

}
```