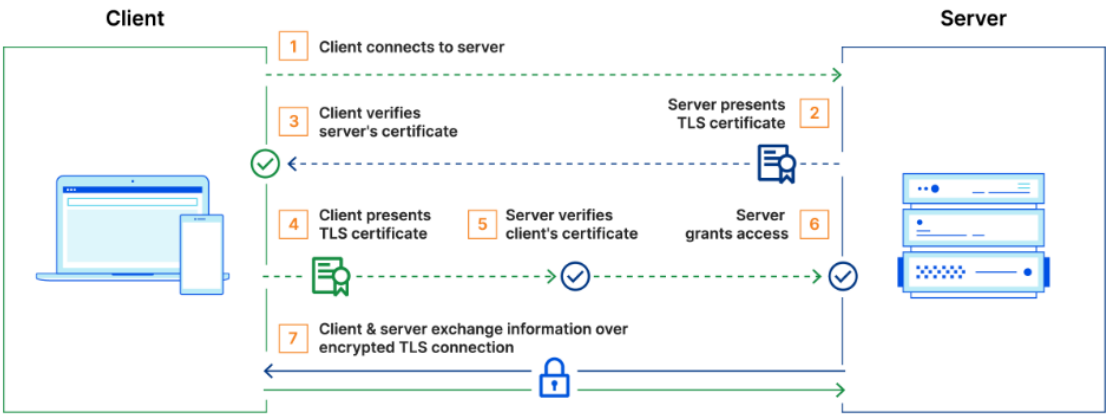
WNP-5741 POC - Mutual Authentication

Mutual authentication is when two sides of a communications channel verify each other's identity, instead of only one side verifying the other



In mTLS, the client and server both have a certificate, and both sides authenticate using their public/private key pair:

1. Client connects to server
2. Server presents its TLS certificate
3. Client verifies the server's certificate
4. Client presents its TLS certificate
5. Server verifies the client's certificate
6. Server grants access
7. Client and server exchange information over encrypted TLS connection

Terminology

Certificate Request :

A Client or Server in order to acquire a valid certificate first need to create a ***CSR*** (Certificate Request) file which they need to submit to the ***Certificate Authority*** which can sign and issue a valid certificate

Certificate Authority:

A Certificate Authority is a *trusted organization* who confirm the authenticity of a website (Server). Their primarily responsibilities are,

- Issue the Certificates to the Entities (Servers, Clients, Websites)
- Confirm whether it's the same website which it claims it is

Certificate :

A (public) certificate is what is issued to the Entity (Server, or Client). As CA issues the certificates they can confirm also whether it's a valid certificate signed by them or not.

SSL: Server Socket Layer

TLS: Transport Layer Security

Public Key: Public Key is used to *Encrypt* the data.

Private Key: Private Key is used to *Decrypt* the data, encrypted by Public key

Steps to create self-signed certificate execute below command in terminal

Step 1: Generate the CA Certificate

1. **Create the CA's Private Key:**

   openssl genrsa -out ca.key 2048

2. **Self-Sign and Create the CA Certificate:**
   openssl req -x509 -new -nodes -key ca.key -sha256 -days 365 -out ca.pem

   This creates a self-signed CA certificate valid for 365 days**.**

Step 2: Generate Server and Client Certificates:

1. **Generate Private Keys**

   For server: openssl genrsa -out server.key 2048

   For client: openssl genrsa -out client.key 2048

2. **Generating CSRs:**

   For server: openssl req -new -key server.key -out server.csr

   For client: openssl req -new -key client.key -out client.csr

3.Sign CSRs with the CA Key:

For server: openssl x509 -req -in server.csr -CA ca.pem -CAkey ca.key - CAcreateserial -out server.crt -days 365

For client: openssl x509 -req -in client.csr -CA ca.pem -CAkey ca.key -CAcreateserial -out client.crt -days 365

Step 3: Create PKCS#12 Keystores

1.Convert Certificates and Keys to PKCS#12 Format:

For server: openssl pkcs12 -export -out server.p12 -name "server" -inkey server.key -in server.crt -certfile ca.pem

For Client: openssl pkcs12 -export -out client.p12 -name "client" -inkey client.key -in client.crt -certfile ca.pem

If above command are not working in machine try below commands to create PKCS#12 keystores

For server: openssl pkcs12 -export -out server.p12 -name "server" -inkey server.key -in server.crt -passin pass:1234 -password pass:1234

For client: openssl pkcs12 -export -out client.p12 -name "client" -inkey client.key -in client.crt -passin pass:1234 -password pass:1234

Step 4: Create the Truststore

Import the CA Certificate into a PKCS#12 Truststore:

keytool -import -file ca.pem -alias "ca" -keystore truststore.p12 -storetype PKCS12

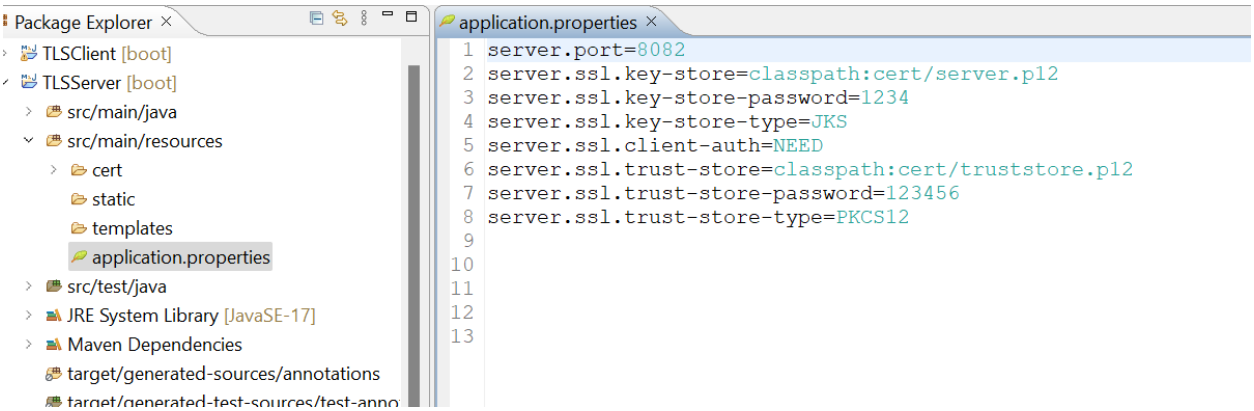
Verfication : After execution of all command we will be getting the below files

ca.key	8/27/2024 3...	KEY File	2 KB
ca.pem	8/27/2024 3...	PEM File	2 KB
ca.srl	8/27/2024 3...	SRL File	1 KB
client	8/27/2024 3...	Security Certificate	2 KB
client.csr	8/27/2024 3...	CSR File	2 KB
client.key	8/27/2024 3...	KEY File	2 KB
client	8/28/2024 5...	Personal Information ...	3 KB
rootCA	8/27/2024 2...	Security Certificate	2 KB
rootCA.key	8/27/2024 2...	KEY File	4 KB
server	8/27/2024 3...	Security Certificate	2 KB
server.csr	8/27/2024 3...	CSR File	2 KB
server.key	8/27/2024 3...	KEY File	2 KB
server	8/28/2024 4...	Personal Information ...	3 KB
server2	8/28/2024 4...	Personal Information ...	3 KB
truststore	8/27/2024 3...	Personal Information ...	2 KB

After verification ,create springboot project for TLSServer and TLSClient.

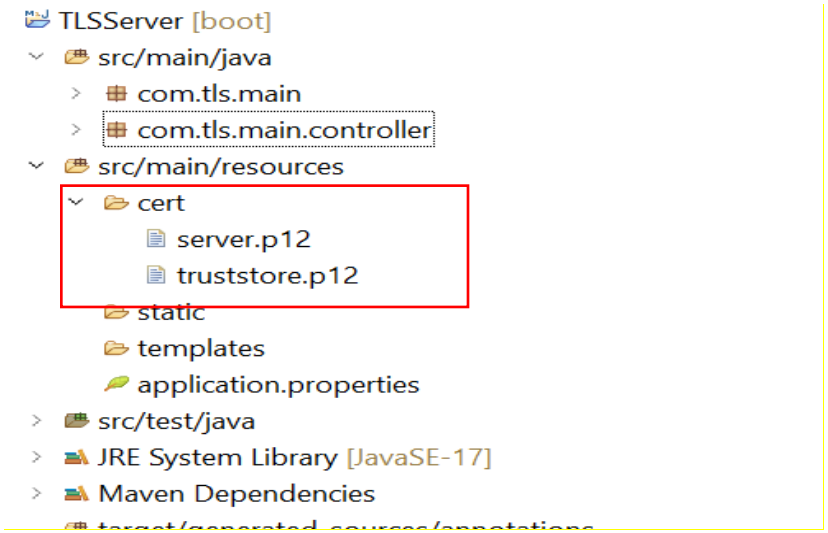
For TLSServer

1.add below properties in application.properties



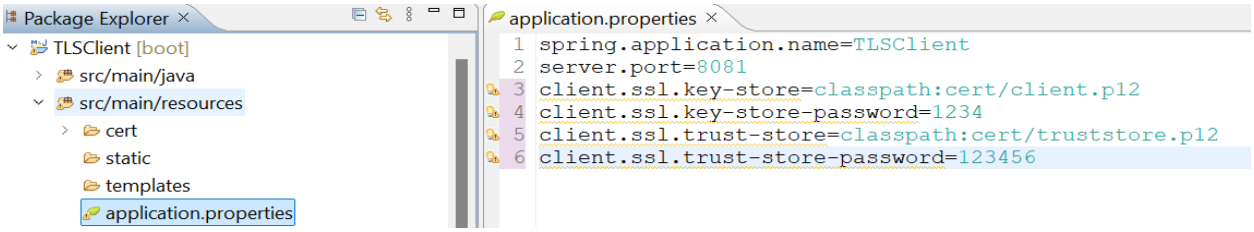
2.Create a get api in ServerController.java :

3. Create folder named ‘cert’ in src/main/resource and copy the server.p12 and truststore.p12

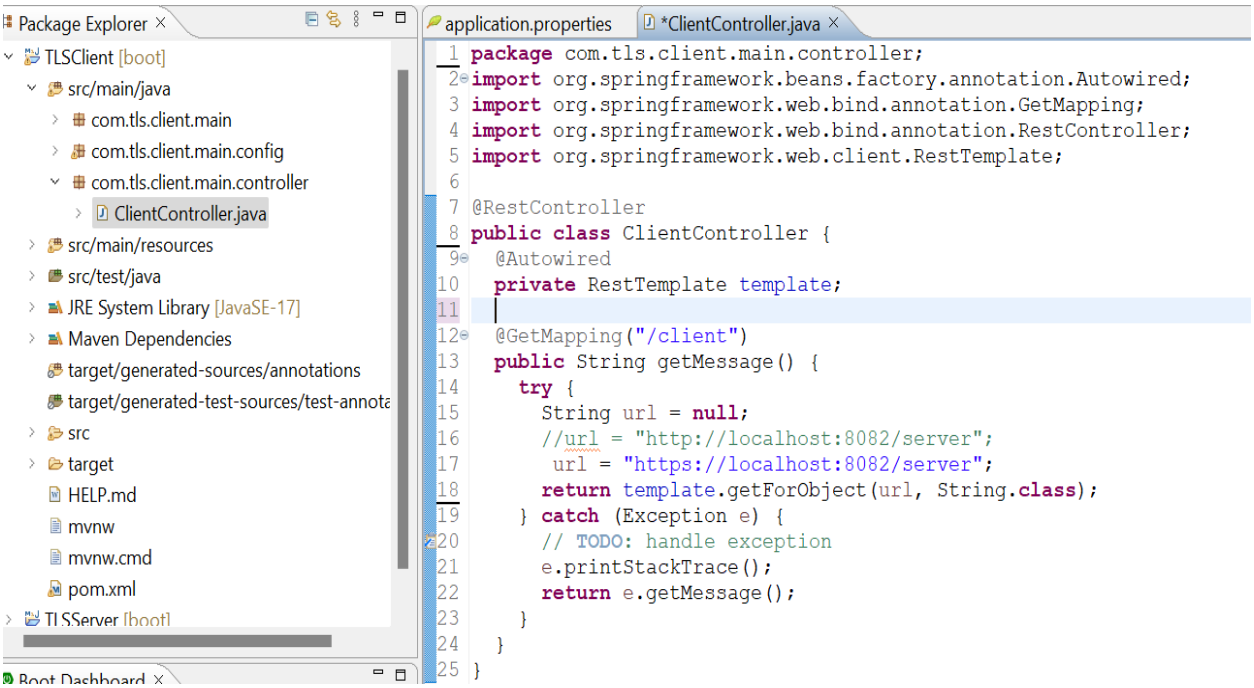


For TLSClient

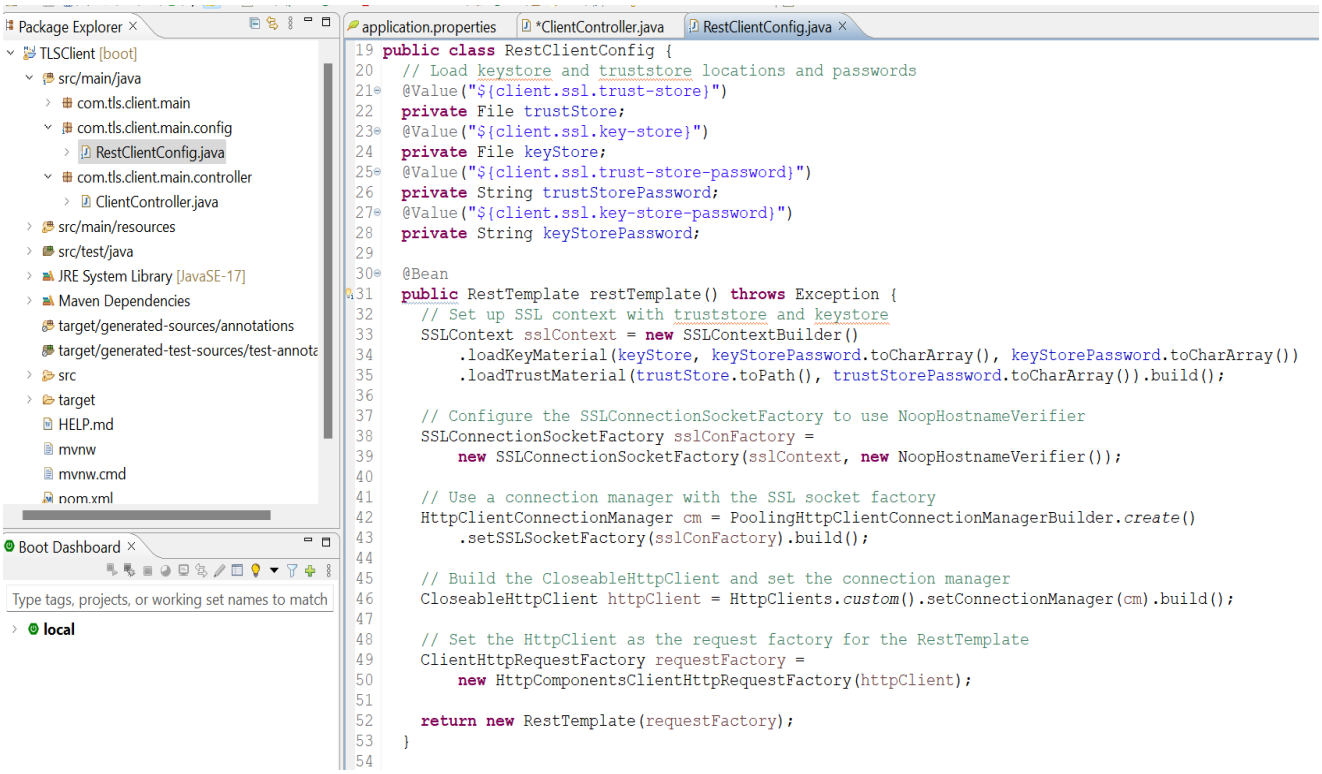
1.add below properties in application.properties



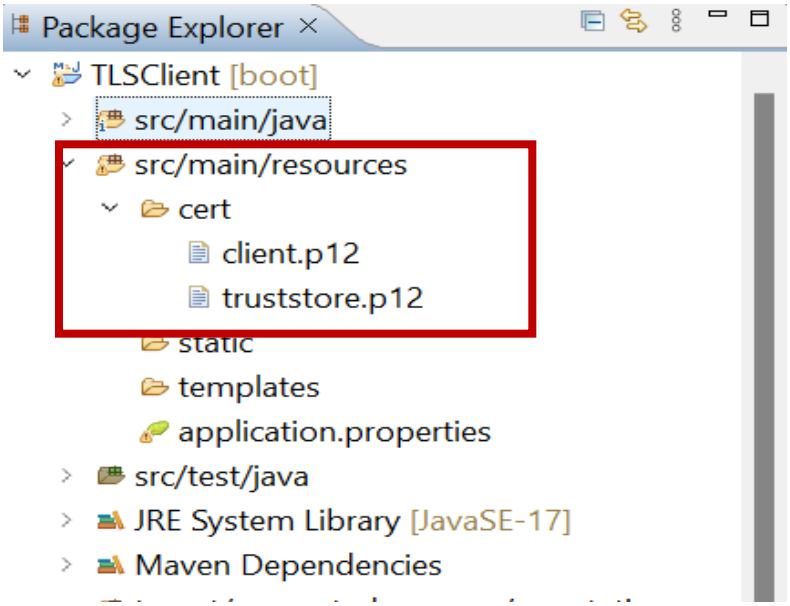
2.Create a get api in ClientController.java:



3.Create RestClientConfig.java



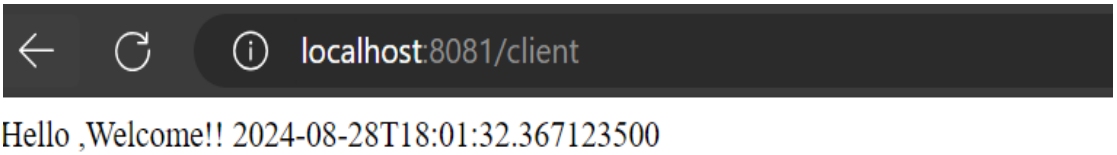
4. Create folder named ‘cert’ in src/main/resource and copy the client.p12 and truststore.p12



Now run the both application , and hit the below url in brower, If mTls connection is sucessful we will be getting the below response

url : localhost:8081/client

output:



Hello ,Welcome!! 2024-08-28T18:01:32.367123500

References:

- What is mTLS? | Mutual TLS | Cloudflare
- MTLS-Everything You need to know (Part-I) | by Sachin Kumar Shukla | Medium
- MTLS-AWS API Gateway (Part-II). This document is the Part—II of... | by Sachin Kumar Shukla | Medium
- GitHub - skshukla/tlsdemo
- Security: mTLS in Spring Boot. Welcome to this technical walkthrough... | by Serhii Bohutskyi | Medium
- openssl - Creating a .p12 file - Stack Overflow
- GitHub - Hakky54/mutual-tls-ssl: 🔐 Tutorial of setting up Security for your API with one way authentication with TLS/SSL and mutual authentication for a java based web server and a client with both Spring Boot. Different clients are provided such as Apache HttpClient, OkHttp, Spring RestTemplate, Spring WebFlux WebClient Jetty and Netty, the old and the new JDK HttpClient, the old and the new Jersey Client, Google HttpClient, Unirest, Retrofit, Feign, Methanol, vertx, Scala client Finagle, Featherbed, Dispatch Reboot, AsyncHttpClient, Sttp, Akka, Requests Scala, Http4s Blaze, Kotlin client Fuel, http4k, Kohttp and ktor. Also other server examples are available such as jersey with grizzly. Also gRPC, WebSocket and ElasticSearch examples are included