Plan for the day:

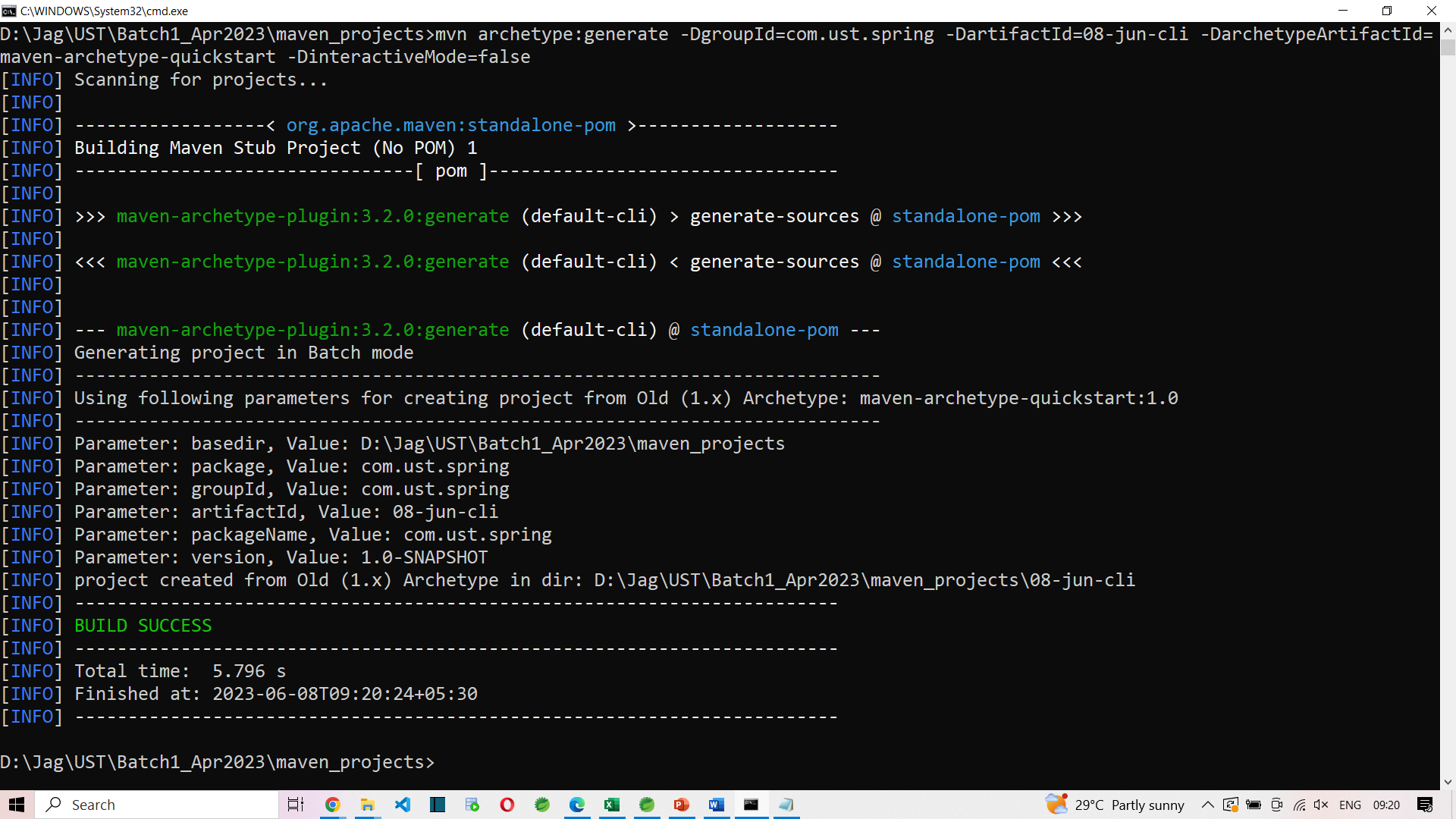
Pendings:

1. Angular + Spring REST api with Security JWT
2. Spring security OAuth2
3. Spring data JPA with Mongodb
4. Introduction to Microservices
5. Spring CLI
6. Spring Cloud - Eureka Server
7. Angular components life cycle

Spring CLI

Spring Command Line Interface

Lets create a project of archetype “quickstart”:



To run a spring boot application from command line (maven project)

mvn spring-boot:run

this command must be run in a location where pom.xml is present.

Spring Cloud

Eureka server and client

Gateway for microservices

Suppose we create 4 rest apis

Customer-api

Product-api

Sales-api

Payment-api

Demo:

1. Lets create one eureka server application 8761
2. Lets create few microservices
   1. Customer api 8383
   2. Product api 8384
   3. Sales api 8385
3. All these apis must define their eureka server at 8761

Steps for eureka:

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>

</dependency>

Steps for creating Eureka Server application:

1) create a spring boot starter application and choose "Eureka Server" in dependencies

2) In application class where main method is there

below @SpringBootApplication, add the following annotation

@EnableEurekaServer

3) in application.properties file, add the following:

eureka.client.registerWithEureka = false

eureka.client.fetchRegistry = false

server.port = 8761

4) update project and run as spring boot application

5) check localhost:8761

------------------

Steps for creating Eureka Client application:

1) create a spring boot starter application and choose "Eureka Discovery Clinet" and "Spring Web" in dependencies

the following dependencies will be added to pom.xml automatically:

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-tomcat</artifactId>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

2) In application (main) class add the below dependencies:

@SpringBootApplication

@EnableEurekaClient

3) define @RestController with request mappings

@SpringBootApplication

@EnableEurekaClient

@RestController

public class EurekaClient1Application {

public static void main(String[] args) {

SpringApplication.run(EurekaClient1Application.class, args);

}

@GetMapping("/")

public String home()

{

return "Jag welcomes you to eureka client application";

}

}

4) application.properties

server.port=8383

spring.application.name=account-service

eureka.instance.hostname=localhost

eureka.client.serviceUrl.defaultZone=http://${eureka.instance.hostname}:8761/eureka/

server.context-path=/account

5) check localhost:8383

6) In Rest controller class,

@Autowired

EurekaClient eurekaClient;

and create a method

@GetMapping("/applications")

public Applications getApplications()

{

return eurekaClient.getApplications();

}

7) restart the app and check

localhost:8383/applications

we get the meta data of the information about our instances

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-actuator</artifactId>

</dependency>

Modern Javascript tutorial

<https://javascript.info/>

OAuth2

Open Authentication

This allows us to use the authentication feature provided by service providers like

Google

Facebook

GitHub

Etc.

First we need to login to Google Cloud Platform

And note down the client id and client secret:

# OAuth2 configuration

security.oauth2.client.clientId=

security.oauth2.client.clientSecret=

security.oauth2.client.preEstablishedRedirectUri=http://localhost:5001/callback

security.oauth2.client.accessTokenUri=https://www.googleapis.com/oauth2/v3/token

security.oauth2.client.userAuthorizationUri=https://accounts.google.com/o/oauth2/auth

security.oauth2.client.tokenName=oauth\_token

security.oauth2.client.authenticationScheme=query

security.oauth2.client.clientAuthenticationScheme=form

security.oauth2.client.scope=profile

security.oauth2.resource.user-info-uri=https://www.googleapis.com/userinfo/v2/me

security.oauth2.client.useCurrentUri=false

spring.data.mongodb.uri=mongodb+srv://abced:efgh@cluster0.ijklx.mongodb.net/db1

spring.data.mongodb.database=db1

Typescript:

Observables

Publish

Subscribe