**Micro Services**

Contents

[URLs 2](#_Toc114506074)

[Annotations 2](#_Toc114506075)

[Application.properties 3](#_Toc114506076)

[Maven 3](#_Toc114506077)

[Add folder to buildpath -maven project right click build path configure 3](#_Toc114506078)

[Springboot – distributed app 4](#_Toc114506079)

[SpringBootAtuoconfiguration 5](#_Toc114506080)

[Path Parameter 6](#_Toc114506081)

[Response of a rest 6](#_Toc114506082)

[Exception 6](#_Toc114506083)

[Database 7](#_Toc114506084)

[Dependency 7](#_Toc114506085)

[--documentation and testing 7](#_Toc114506086)

[CURL 8](#_Toc114506087)

[Maven 8](#_Toc114506088)

[Filtering json response 8](#_Toc114506089)

[Actuator 9](#_Toc114506090)

[Hal Explorer 9](#_Toc114506091)

[docker 10](#_Toc114506092)

[zipkin 12](#_Toc114506093)

[Kubectl -Kubernetes Controller 12](#_Toc114506094)

[SpringCloud 14](#_Toc114506095)

[Zipkin distributed tracing 14](#_Toc114506096)

[Others 15](#_Toc114506097)

[Response code: 16](#_Toc114506098)

[Eclipse commands 16](#_Toc114506099)

[Internationalization 16](#_Toc114506100)

[Version 17](#_Toc114506101)

[Hateoas 18](#_Toc114506102)

[Filter 18](#_Toc114506103)

[static 18](#_Toc114506104)

[Dynamic filter 18](#_Toc114506105)

[Lombook 18](#_Toc114506106)

[Limitation 19](#_Toc114506107)

# URLs

<https://start.spring.io/>

<https://github.com/in28minutes/spring-microservices/blob/master/02.restful-web-services/2.3.1.RELEASE-upgrade.md>

<https://github.com/misbaharchitect/suneratech>

# Annotations

@SpringBootApplication = below 3

@SpringBootConfiguration

@EnableAutoConfiguration

@ComponentScan

@component

@Autowire

@Entity

@Table(name = "TABLE\_X\_CREDIT\_CARD", schema = "SA")

@Data

@NoArgsConstructor

@AllArgsConstructor

@JsonIgnoreProperties({ "hibernateLazyInitializer", "handler" })

@RestController

@GetMapping(“/hello”)

@ControllerAdvice -- to tell all the controller while doing exception hand

# Application.properties

server.port = 8081

[spring.application.name](https://docs.spring.io/spring-boot/docs/current/reference/html/application-properties.html#application-properties.core.spring.application.name)=euroserver

spring.jpa.show-sql=true

spring.h2.console.enabled=true

eureka.client.service-url.default-zone=http://localhost:8761/eureka

# Maven

./mvnw dependency:tree

# Add folder to buildpath -maven project right click build path configure

Graphical user interface, text, application

Description automatically generated

# Springboot – distributed app

Diagram

Description automatically generated

Diagram

Description automatically generated

Graphical user interface, text, application

Description automatically generated

# SpringBootAtuoconfiguration

starter web

DispatcherServlet

DispatcherServletAutoConfiguration

Front controller

/

It know the all the mappings in app

# Path Parameter

@GetMapping("/users/{id}")

@GetMapping("/hello/{user}")

**public** String helloUser(@PathVariable String user) {

to send a bean of values in post request

**public** String helloUser(@RequestBody User user) {

# Response of a rest

ResponseEntity

200 ResponseEntity.ok(userrepo.findAll())

404 return ResponseEntity.status(HttpStatus.NOT\_FOUND).body("User "+id +" not found");

201 ResponseEntity.created(new URI(user2.getId().toString())).body(user2)

# Exception

ResponseEntityExceptionHandler

# Database

Jpa

H2

# Calling MS from another MS

ResponseEntity<CurrencyConversionBean> responseEntity = new RestTemplate().getForEntity(

# "http://localhost:8000/currency-exchange/from/{from}/to/{to}", CurrencyConversionBean.class, uriVariables);

# CurrencyConversionBean response = responseEntity.getBody();

@FeignClient --proxy

@SpringBootApplication

@EnableFeignClients("com.in28minutes.microservices.currencyconversionservice")

@EnableDiscoveryClient

//@FeignClient(name="currency-exchange-service", url="localhost:8000")

//@FeignClient(name="currency-exchange-service")

@FeignClient(name="netflix-zuul-api-gateway-server")

@RibbonClient(name="currency-exchange-service")

**public** **interface** CurrencyExchangeServiceProxy {

//@GetMapping("/currency-exchange/from/{from}/to/{to}")

@GetMapping("/currency-exchange-service/currency-exchange/from/{from}/to/{to}")

**public** CurrencyConversionBean retrieveExchangeValue

(@PathVariable("from") String from, @PathVariable("to") String to);

}

## Ribbon load balancing replace Feign

Starter-ribbon

Application.pro

#currency-exchange-service.ribbon.listOfServers=http://localhost:8000,http://localhost:8001

## 3.Eureka naming server

Service discovery

While starting any application they register with Eureka

@EnableEurekaServer

@SpringBootApplication

In eureka server

<dependency>

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

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

</dependency>

In clients

@SpringBootApplication

@EnableDiscoveryClient

eureka.client.service-url.default-zone=http://localhost:8761/eureka

<dependency>

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

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

<exclusions>

<exclusion>

<groupId>com.fasterxml.jackson.dataformat</groupId>

<artifactId>jackson-dataformat-xml</artifactId>

</exclusion>

</exclusions>

</dependency>

# Dependency

# --documentation and testing

<dependency>

<groupId>org.springdoc</groupId>

<artifactId>springdoc-openapi-ui</artifactId>

<version>1.6.9</version>

</dependency>

<http://localhost:8082/swagger-ui/index.html>

list of apis we have developed

Graphical user interface, text, application

Description automatically generated

**Actuator**

<dependency>

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

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

</dependency>

<http://localhost:8082/actuator>

application.properties

management.endpoints.web.exposure.include=env

=\*

# CURL

CURL -X GET <http://localhost:8082/hello/>

CURL -X GET http://localhost:8082/hello2

# Maven

Right click – maven –update project – force update of Snapshot and release

# Filtering json response

Static filtering done at the java bean

@JsonIgnore

Dynamic filtering

In controller user MappingJacksonValue(bean)

And user FilterProvider SimpleBeanPropertyFilter

# Actuator

Gives health info

Bean metrics mappings

/src/main/resources/application.properties

management.endpoints.web.exposure.include=\*

http://Localhost:8082/actuator

http://Localhost:8080/actuator/metrix

http://Localhost:8080/actuator/metrix/onemetrixval

# Hal Explorer

<dependency>

<groupId>org.springframework.data</groupId>

<artifactId>spring-data-rest-hal-explorer</artifactId>

<version>3.2.0.RC1</version>

</dependency>

<http://localhost:8082/explorer/index.html#url=/>

<http://localhost:8082/explorer/index.html#url=http://localhost:8082/actuator/beans>

<http://localhost:8082/explorer/index.html#url=http://localhost:8082/actuator/env>

<http://localhost:8082/explorer/index.html#url=http://localhost:8082/users/201>

# docker

docker --version

docker run -p 5000:5000 -d in28min/todo-rest-api-h2:1.0.0.RELEASE

docker run -p 5001:5000 -d in28min/todo-rest-api-h2:1.0.0.RELEASE

docker run -p 5000:5000 -d --restart=always in28min/todo-rest-api-h2:1.0.0.RELEASE

docker container pause contid

docker container unpause contid

docker container inspect contid

docker container prune – to delete all stopped containers

-p HOSTPORT:CONTAINERPORT

-d detached

docker logs -f containerid

docker container ls

docker container ls -a

docker container stop containerid – SIGKILL grace full shut down close all others

docker container kill containerid

hub.docker.com – public repository

for company we need private repository

docker pull mysql

docker image remove imageid

docker search mysql

docker image history imageid

docker image inspect imageid

docker images

http://hub.docker.com/r/in28min/todo-rest-api-h2

docker events

docker top contid

<build>

<plugins>

<plugin>

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

<artifactId>spring-boot-maven-plugin</artifactId>

<configuration>

<image>

<name>gayazahamed/mmv2-${project.artifactId}:${project.version}</name>

</image>

<pullPolicy>IF\_NOT\_PRESENT</pullPolicy>

</configuration>

</plugin>

</plugins>

</build>

project run as maven build

goals

spring-boot:build-image -DskipTests

docker.io/gayazahamed/mmv2-demo:0.0.1-SNAPSHOT

docker run -p 8082:8082 -d gayazahamed/mmv2-demo:0.0.1-SNAPSHOT

docker push gayazahamed/mmv2-demo:0.0.1-SNAPSHOT

docker run --detach --env MYSQL\_ROOT\_PASSWORD=dummypassword --env MYSQL\_USER=social-media-user --env MYSQL\_PASSWORD=dummypassword --env MYSQL\_DATABASE=social-media-database --name mysql --publish 3306:3306 mysql:8-oracle

# zipkin

<http://127.0.0.1:9411/>

docker run -p 9411:9411 openzipkin/zipkin:2.23

# Kubectl -Kubernetes Controller

[Master Microservices with Spring Boot and Spring Cloud | Udemy](https://www.udemy.com/course/microservices-with-spring-boot-and-spring-cloud/learn/lecture/24356386#overview)

[spring-microservices-v2/05.kubernetes at main · in28minutes/spring-microservices-v2 · GitHub](https://github.com/in28minutes/spring-microservices-v2/tree/main/05.kubernetes#commands)

docker run -p 8080:8080 in28min/hello-world-rest-api:0.0.1.RELEASE

kubectl create deployment hello-world-rest-api --image=in28min/hello-world-rest-api:0.0.1.RELEASE

kubectl expose deployment hello-world-rest-api --type=LoadBalancer --port=8080

kubectl scale deployment hello-world-rest-api --replicas=3

kubectl delete pod hello-world-rest-api-58ff5dd898-62l9d

kubectl autoscale deployment hello-world-rest-api --max=10 --cpu-percent=70

kubectl edit deployment hello-world-rest-api #minReadySeconds: 15

kubectl set image deployment hello-world-rest-api hello-world-rest-api=in28min/hello-world-rest-api:0.0.2.RELEASE

gcloud container clusters get-credentials in28minutes-cluster --zone us-central1-a --project solid-course-258105

kubectl create deployment hello-world-rest-api --image=in28min/hello-world-rest-api:0.0.1.RELEASE

kubectl expose deployment hello-world-rest-api --type=LoadBalancer --port=8080

kubectl set image deployment hello-world-rest-api hello-world-rest-api=DUMMY\_IMAGE:TEST

kubectl get events --sort-by=.metadata.creationTimestamp

kubectl set image deployment hello-world-rest-api hello-world-rest-api=in28min/hello-world-rest-api:0.0.2.RELEASE

kubectl get events --sort-by=.metadata.creationTimestamp

kubectl get componentstatuses

kubectl get pods --all-namespaces

kubectl get events

kubectl get pods

kubectl get replicaset

kubectl get deployment

kubectl get service

kubectl get pods -o wide

kubectl explain pods

kubectl get pods -o wide

kubectl describe pod hello-world-rest-api-58ff5dd898-9trh2

kubectl get replicasets

kubectl get replicaset

kubectl scale deployment hello-world-rest-api --replicas=3

kubectl get pods

kubectl get replicaset

kubectl get events

kubectl get events --sort.by=.metadata.creationTimestamp

kubectl get rs

kubectl get rs -o wide

kubectl set image deployment hello-world-rest-api hello-world-rest-api=DUMMY\_IMAGE:TEST

kubectl get rs -o wide

kubectl get pods

kubectl describe pod hello-world-rest-api-85995ddd5c-msjsm

kubectl get events --sort-by=.metadata.creationTimestamp

kubectl set image deployment hello-world-rest-api hello-world-rest-api=in28min/hello-world-rest-api:0.0.2.RELEASE

kubectl get events --sort-by=.metadata.creationTimestamp

kubectl get pods -o wide

kubectl delete pod hello-world-rest-api-67c79fd44f-n6c7l

kubectl get pods -o wide

kubectl delete pod hello-world-rest-api-67c79fd44f-8bhdt

gcloud container clusters get-credentials in28minutes-cluster --zone us-central1-c --project solid-course-258105

docker login

docker push in28min/mmv2-currency-exchange-service:0.0.11-SNAPSHOT

docker push in28min/mmv2-currency-conversion-service:0.0.11-SNAPSHOT

kubectl create deployment currency-exchange --image=in28min/mmv2-currency-exchange-service:0.0.11-SNAPSHOT

kubectl expose deployment currency-exchange --type=LoadBalancer --port=8000

kubectl get svc

kubectl get services

kubectl get pods

kubectl get po

kubectl get replicaset

kubectl get rs

kubectl get all

kubectl create deployment currency-conversion --image=in28min/mmv2-currency-conversion-service:0.0.11-SNAPSHOT

kubectl expose deployment currency-conversion --type=LoadBalancer --port=8100

kubectl get svc --watch

kubectl get deployments

kubectl get deployment currency-exchange -o yaml >> deployment.yaml

kubectl get service currency-exchange -o yaml >> service.yaml

kubectl diff -f deployment.yaml

kubectl apply -f deployment.yaml

kubectl delete all -l app=currency-exchange

kubectl delete all -l app=currency-conversion

kubectl rollout history deployment currency-conversion

kubectl rollout history deployment currency-exchange

kubectl rollout undo deployment currency-exchange --to-revision=1

kubectl logs currency-exchange-9fc6f979b-2gmn8

kubectl logs -f currency-exchange-9fc6f979b-2gmn8

kubectl autoscale deployment currency-exchange --min=1 --max=3 --cpu-percent=5

kubectl get hpa

kubectl top pod

kubectl top nodes

kubectl get hpa

kubectl delete hpa currency-exchange

kubectl create configmap currency-conversion --from-literal=CURRENCY\_EXCHANGE\_URI=http://currency-exchange

kubectl get configmap

kubectl get configmap currency-conversion -o yaml >> configmap.yaml

watch -n 0.1 curl http://34.66.241.150:8100/currency-conversion-feign/from/USD/to/INR/quantity/10

docker push in28min/mmv2-currency-conversion-service:0.0.12-SNAPSHOT

docker push in28min/mmv2-currency-exchange-service:0.0.12-SNAPSHOT

# SpringCloud

Projects.spring.io/spring-cloud

# Zipkin distributed tracing

used for distributed tracing

assigns a unique id for series of micro service call

A call B

a unique id number given to A and same to B

<http://127.0.0.1:9411/>

docker run -p 9411:9411 openzipkin/zipkin:2.23

maven plung in

spring-cloud-starter-sleuth

spring-cloud -sleuth-zipkin

if distributed server is down then use rabbit MQ

use sampler to trace only few %of request so avoid perfomace

in application.prop

spring.sleuth.sampler.probability=0.1

above is 10%

# Others

pom.xml right click build 🡪 clean install

to run a jar

spring boot jar has tomcat

Java -jar myjar.jar

Post to create new

Put to update

Patch to update a part of resource

# Response code:

404 resource not found

200 successes

201 created

204 no content

401 unauthorized

400 bad request

500 internal server error

Talend Api tester

# Eclipse commands

Cnt Shift R to open a java file

Cnt Shift T to open a java file jar also

Cnt O organize imports

Cnt 1 for possible hints

# Internationalization

**private** MessageSource ms;

messages.properties

message\_es.properties

**public** HelloWorldRest(MessageSource ms1) {

**this**.ms = ms1;

}

@GetMapping("/helloIn")

**public** String helloUserInterNational() {

Locale lx = LocaleContextHolder.*getLocale*();

**return** ms.getMessage("good.morning.message", **null**, "Default Message", lx);

}

In request header

Accept-Language = es

# Version

<http://localhost:8082/v1/person>

<http://localhost:8082/v2/person>

params

<http://localhost:8082/person>?version=1

<http://localhost:8082/person>?version=1

header

@GetMapping(path="/person" ,headers="X-API-VERSION=2")

**public** PersonV2 getPersonHeader2() {

**return** **new** PersonV2("First2","Last2");

}

Accept

Application/vnd.company.app-v1+json

# Hateoas

Provides the links to other resources

# Filter

## static

@JSONProperty(“newname”)

@JsonIgnore

Class level

@JsonIgnoreProperties(“name,id”)

## Dynamic filter

MappingJacksonValue

SimpleBeanPropertyFilter

FilterProvider

# Classes

Environment.getProperty(“local.server.port”)

# Change port / run multi instance of server with diff port

Run as – run configuration

Give app name and java class name to run

And VM Arguments -- -Dserver.port=8001

# Lombook

@NoArgsConstructor, @RequiredArgsConstructor, @AllArgsConstructor

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<optional>true</optional>

</dependency>

# Limitation

Defining boundaries

Configuration management

Dynamic scale up and down

Visibility – where is the bug which MS is down

Fault tolerance otherwise pack of cards