**Micro Services**

Contents

[URLs 2](#_Toc113887863)

[Annotations 2](#_Toc113887864)

[Application.properties 2](#_Toc113887865)

[Maven 3](#_Toc113887866)

[Springboot – distributed app 3](#_Toc113887867)

[Anotations 4](#_Toc113887868)

[SpringBootAtuoconfiguration 4](#_Toc113887869)

[Path Parameter 5](#_Toc113887870)

[Database 5](#_Toc113887871)

[Dependency 5](#_Toc113887872)

[CURL 6](#_Toc113887873)

[Maven 6](#_Toc113887874)

[Filtering json response 6](#_Toc113887875)

[Actuator 6](#_Toc113887876)

[/src/main/resources/application.properties 7](#_Toc113887877)

[docker 7](#_Toc113887878)

[zipkin 9](#_Toc113887879)

[Kubectl -Kubernetes Controller 9](#_Toc113887880)

[zipkin distributed tracing 12](#_Toc113887881)

# 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

@component

@Autowire

@Entity

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

@Data

@NoArgsConstructor

@AllArgsConstructor

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

@RestController

@GetMapping(“/hello”)

# 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

# Maven

./mvnw dependency:tree

# Springboot – distributed app

Diagram

Description automatically generated

Diagram

Description automatically generated

Graphical user interface, text, application

Description automatically generated

# Anotations

@RestController

# 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) {

# Database

Jpa

H2

# 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

**Actuator**

<dependency>

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

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

</dependency>

# 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

http://Localhost:8080/actuator

### /src/main/resources/application.properties

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

http://Localhost:8080/actuator/metrix

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

# 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

# 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

# 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%