# Microservice practice

### Java

Installing jdk 15 (jre embedded)

Set environment variable, + JAVA\_HOME path

### Micro service

Deploying component as a service

* Service can talk each other and get the job done

### Spring cloud

Released in 2015 as open source framework

Run anywhere

Build industry standard patterns and best practices

### Spring boot

Released in 2014

Less configuration but Convention (this kind files should place as this location)

Dependency management made simple (maven)

## Create Demo-application

start.spring.io

* Create
* Insert project in to intellij idea

### Dependency

* Spring **web**
* spring-boot-starter-web
  + by “starter” spring knows about transitive dependencies
* spring boot **actuator**

### Structure of spring

Repository

Service

Controller

Controller

\* handle the traffic

\* Controller is the last layer

We have to handle the exception here

(Throw early cutch late)

#### Service

\* hold business logics

#### Repository

\* Deal with databases

(Spring data JPA)

### Create controller

@RestController // this class is a controller  
public class MainController {  
  
 @RequestMapping(value = "/hello") // handle request  
 public String greeting(){  
 return "Hello am i working";  
 }  
  
}

### Run application

1. Run the application
   1. Server is starting in local machine port 8080
2. insomnia rest client
   1. create service hello => <http://localhost:8080/hello>

### Change server port

To change resources/application.properties or .yml

server:  
 port: 8081

/resources/config directory and

/resources directory are considered as “classDirectory” in Runtime

### RequestMapping

@RequestMapping(value = "/hello")

Get

Put

Post /hello will hit this method

Patch

Delete

@RequestMapping(value = "/hello", method = RequestMethod.*GET*)

Get /hello only will his this method

## Spring boot actuator

It gives us some endpoints

[http://localhost:8081**/actuator/beans**](http://localhost:8081/actuator/beans)

These endpoints are not available by default

application.yml

management:  
 endpoints:  
 web:  
 exposure:  
 include: '\*' // expose all endpoint

### By endpoints we can get tons of information

81/actuator/beans

81/actuator/threaddump

:at this movement all the threads running on your application

81/actuator/metrics

81/actuator/heapdump

### Create own endpoints

Examples

We need to change the log level dynamically

Enable/ disable features dynamically

@Component

= create bean of this class

@Service, @Repository extends @Component so creates bean

#### Concurrenthashmap

T1

Tn

Hash

table

hashmap

Multiple threads are allow to only one thread is allow

operate simultaneously at one time

so Not thread safe (total mapObj logged) thread safe

|  |
| --- |
| Bucket 15 |
| Bucket 14 |
| … |
| … |
| Bucket 1 |
| Bucket 0 |

* Log is available at bucket level (segment log)
* Thread safe
* Any number of threads can do read operations simultaneously
* Concurrent level # of threads can do the update operation simultaneously (in this case 16)
* If concurrent level 32 there is 2 logs with a segment