Monolithic – Single endpoint and Single project and written in single language.

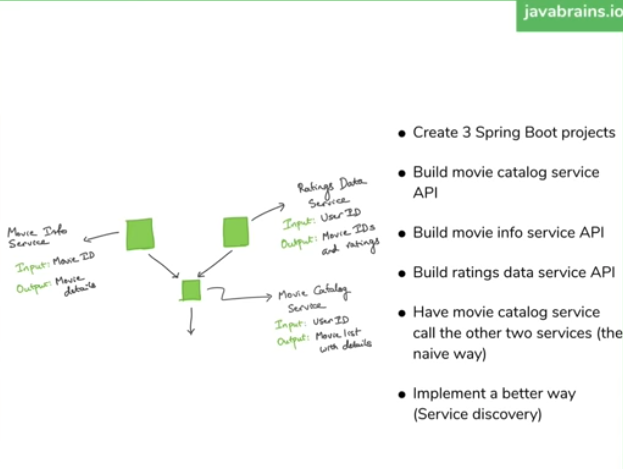
Service Oriented Architecture – Can have different endpoint and single project and written in single language

Micro Services Architecture - can have different endpoint and different project and in different language.

Spring cloud to micro services implementation.

Complexity between micro service.

Workshop:



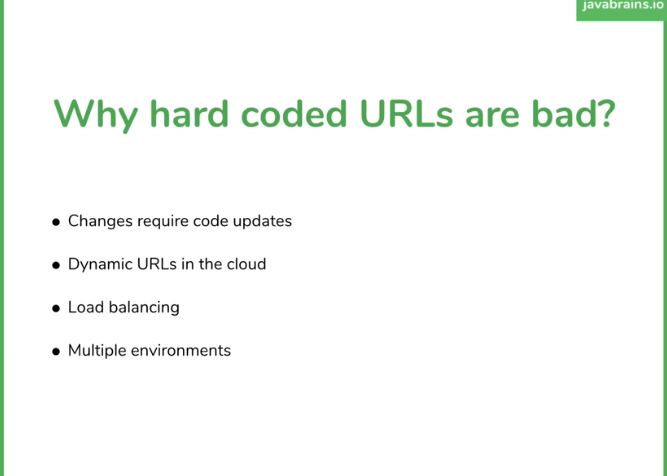
Calling the external API:

1. RestTempleta
2. WebClient

Mono: bodyToMono

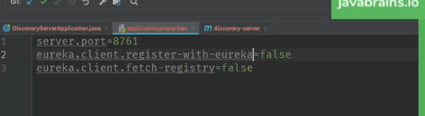
<https://www.baeldung.com/spring-5-webclient>

List<Rating> result = restTemplate.getForObject("http://localhost:8980/rating/byuser/123",ParameterizedTypeReference<ResponseWrapper<List<Rating>>>(){});



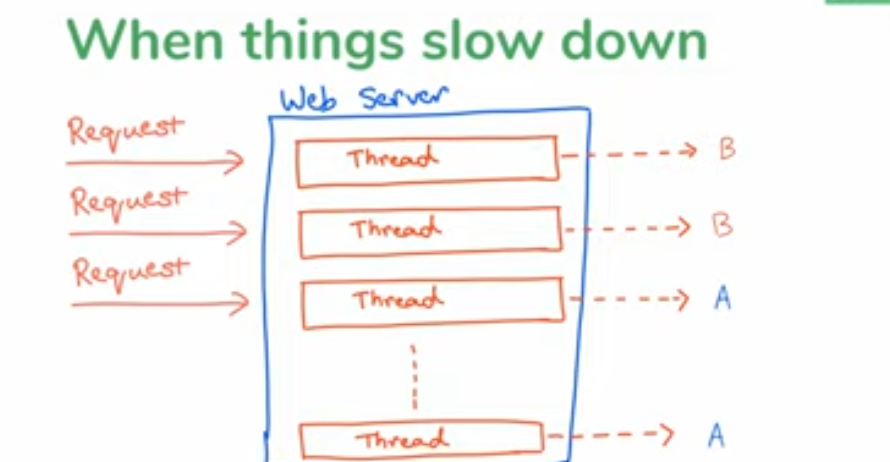
Service Discovery:

Technology : Eureka , Ribbon, Hysterix, Zuul

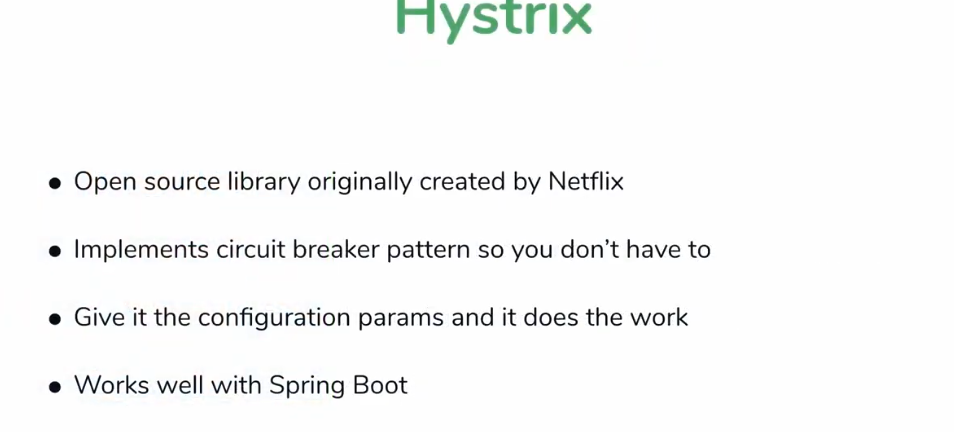




Fault tolerance:

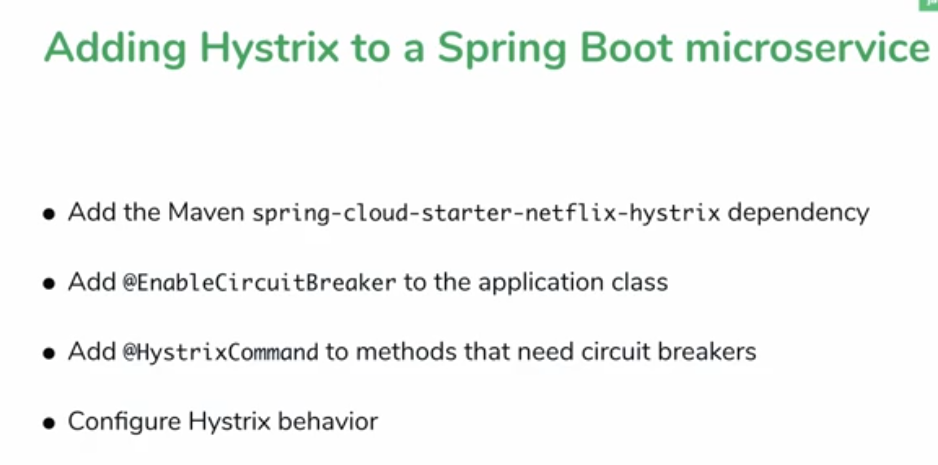


Hystrix:



Disadvantage :

* Not under active development on maintenances .
* We need to set circuit break parameter.



<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<parent>

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

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

<version>2.1.9.RELEASE</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.sachin.microservice</groupId>

<artifactId>move-catalog</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>move-catalog</name>

<description>Demo project for Spring Boot</description>

<properties>

<java.version>1.8</java.version>

<spring-cloud.version>Greenwich.SR3</spring-cloud.version>

</properties>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

<artifactId>spring-cloud-starter-netflix-hystrix</artifactId>

</dependency>

<!-- <dependency>

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

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

</dependency>

<dependency>

<groupId>org.projectreactor</groupId>

<artifactId>reactor-spring</artifactId>

<version>1.0.1.RELEASE</version>

</dependency> -->

<dependency>

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

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

<scope>test</scope>

<exclusions>

<exclusion>

<groupId>org.junit.vintage</groupId>

<artifactId>junit-vintage-engine</artifactId>

</exclusion>

</exclusions>

</dependency>

</dependencies>

<dependencyManagement>

<dependencies>

<dependency>

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

<artifactId>spring-cloud-dependencies</artifactId>

<version>${spring-cloud.version}</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

<build>

<plugins>

<plugin>

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

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

</plugin>

</plugins>

</build>

<repositories>

<repository>

<id>spring-milestones</id>

<name>Spring Milestones</name>

<url>https://repo.spring.io/milestone</url>

</repository>

</repositories>

</project>



How the hystrix work?



@SpringBootApplication

@EnableCircuitBreaker

**public** **class** MoveCatalogApplication {

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(MoveCatalogApplication.**class**, args);

}

@Bean

**public** RestTemplate getRestTemplate() {

**return** **new** RestTemplate();

}

}

Controller:

@GetMapping("/{userId}")

@HystrixCommand(fallbackMethod="getMoveCatalogFallBack")

**public** List<CatalogItem> getMoveCatalog(@PathVariable String userId){

/\*return Collections.singletonList(

new CatalogItem("Titanic", "test", "5"));\*/

UserRatting result = restTemplate.getForObject("http://localhost:8980/rating/byuser/" + userId ,UserRatting.**class**);

List<CatalogItem> catalog = **new** ArrayList<>();

**for**(Rating rating:result.getUserRatting()) {

restTemplate.getForObject("http://localhost:8090/moveinfo/" + rating.getMoveId(), Move.**class**);

catalog.add(**new** CatalogItem(rating.getMoveId(), "test", rating.getRating()));

}

**return** catalog;

/\*return result.stream()

.map(rating ->{

restTemplate.getForObject("http://localhost:8090/moveinfo/" + rating.getMoveId(), Move.class);

new CatalogItem("Titanic", "test", "5");}

)

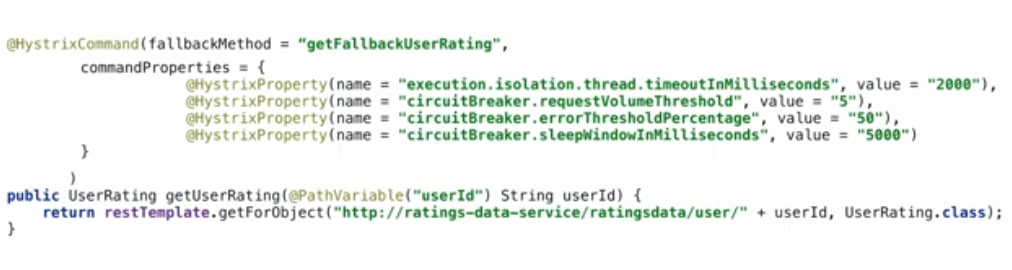
.collect(Collectors.toList());\*/

}

**public** List<CatalogItem> getMoveCatalogFallBack(@PathVariable String userId){

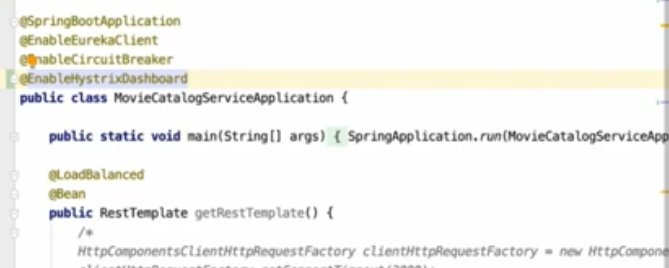
**return** Arrays.*asList*(**new** CatalogItem("No Move", "Hystrix fallback method", "0"));

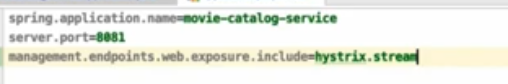
}



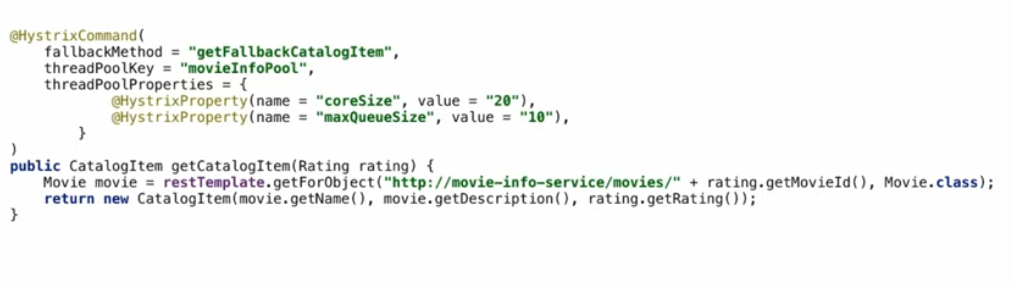
**Hystrix Dashboard**

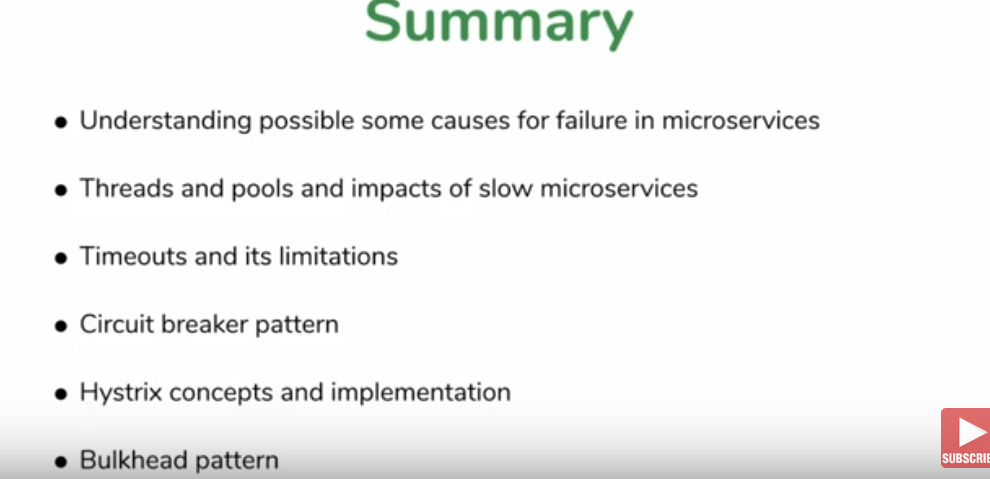






Bulkhead Pattern:





Architecture:

