**Why Micro services:** To increase modularity, scalability and loose coupling and more cohesion in Application.

**Wat Problem mainly with Micro services?:** Though now we do have multiple services to maintain modularity, scalability cohesion, so now if any common issues does come then we need to resolve that for all micro services. Since those are common problems so do we have so many technologies/solution at available.

**Why so many technologies? :** To solve a problem pattern we do have multiple solution provided as technology.

Like How micro services r going to talk to each other is via service discovery wch is a pattern, so solution of this pattern is Eureka technology.

**Is Microservices and SOA is same?** : NO, in SOA we don’t have idea for wch module or wr it is going to be used, like multiple utilities provided.

But in case of Microservices we sud be very clear idea about for wat pupose we r creating it, it may get use , but reusability is not main purpose of microservices.

**Spring RestTemplate and WEBClient:** Spring provide inbuild REST liabrary to handle REST services, it is asynchrounous in natue while WEBClient is latest Spring REST liabrary which supports asynchrounous and will not block executing thread.

UserRating ratingList = restTemplate.getForObject("http://localhost:8082/rating/user/"+userID, UserRating.**class**);

Below is dependency to be added to get spring webClient, wch supports aync rest api/reactive programming.

<dependency>

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

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

</dependency>

/\*

\* Movie movie = webBuilder.build() .get()

\* .uri("http://localhost:8081/movie/"+rating.getMovieId()) .retrieve()

\* .bodyToMono(Movie.class) .block();

\*/

**\*\*We can make webclient async calls to sync by blocking them**

**\*\* How to pass return List to RestTemplate?**

restTemplate.getForObject(“https://.......”, Parametrized(List<Rateing>))

or

* Passing List as a property of another class.

**public** **class** UserRating {

List<Rating> userRatingList;

**public** List<Rating> getUserRatingList() {

**return** userRatingList;

}

**public** **void** setUserRatingList(List<Rating> userRatingList) {

**this**.userRatingList = userRatingList;

}

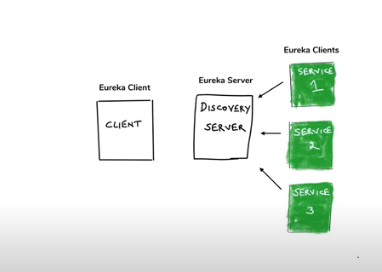
**How to make resttemplate async? :**

**Service Discovery:** All microservices needs to get registered with Service Discovery.

Client side Service discovery, and Server side Service Discovery

\*\*Spring Cloud used Client side service discovery.

Netflix provides open source service discovery called as Eureka Server



<dependency>

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

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

</dependency>

@SpringBootApplication

@EnableEurekaServer

**public** **class** ServiceDiscoveryApplication {

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

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

}

}

\*\* add @enableEurekaClient to all client microservices which are going to register themselves with server, and add eureka client dependency

<dependency>

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

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

</dependency>

@SpringBootApplication

@EnableEurekaClient

**public** **class** MoviecatalogserviceApplication {

@Bean

@LoadBalanced

**public** RestTemplate getRestTemplate()

{

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

}

@Bean

**public** WebClient.Builder getWEBClient()

{

**return** WebClient.*builder*();

}

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

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

}

}

\*\*@LoadBalanced: Does a whole a lot of things like loadbalancing + default fault tolerance

**Fault Tolerance and Resilience in Micro services**

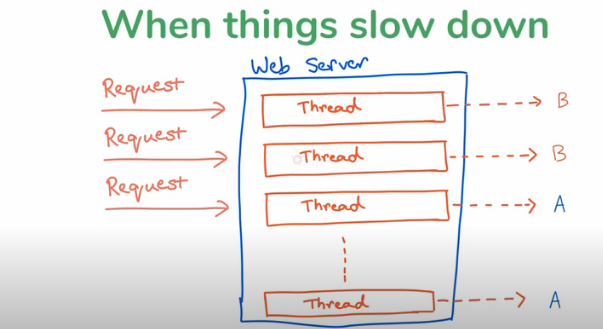
@Value(“${api.key}”)->get properties declared in application.properties file in spring boot

<https://www.themoviedb.org/settings/api>

<https://github.com/koushikkothagal/spring-boot-microservices-workshop>

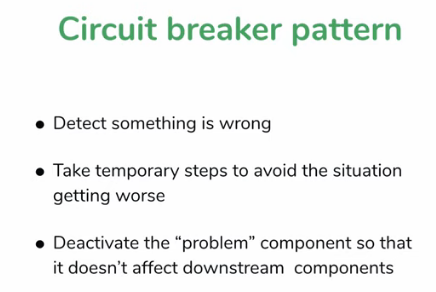
What if one or more microservices gets down??: we cn run multiple instances of same microservices.

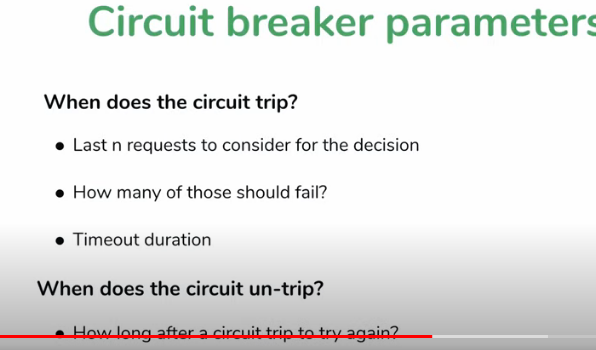
What if one microservice is slow, and making un related api call flow also slow?

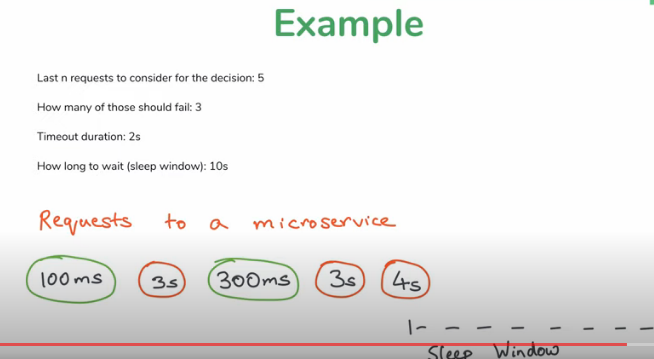


Resource utilization exhausted, and application server thread pool size will cross its limit

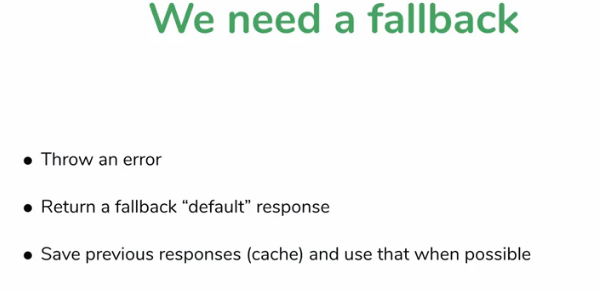
Solution: adding timeout

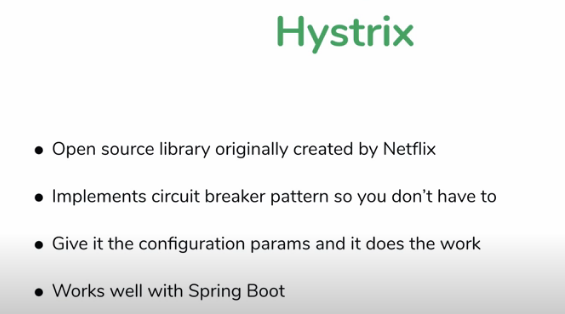


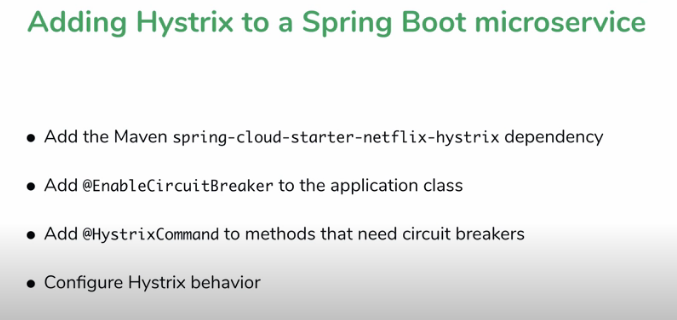




What if circuit breaks, But still requests are coming in?

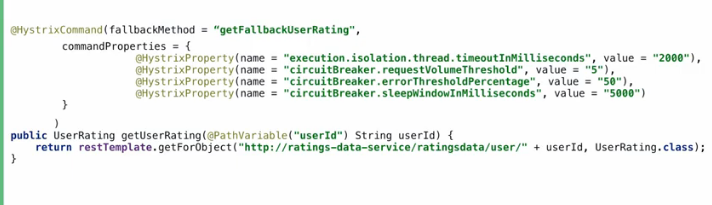






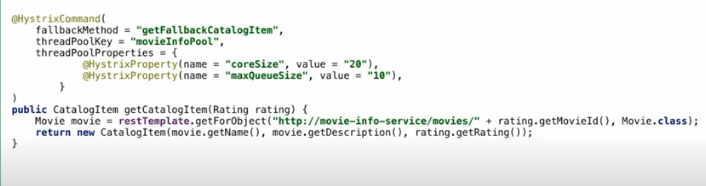
Hystrix proxy issue: it will not able to implement circuit break we within same instance we do hv multiple fallbacks. So better to move out those fallbacks into separate services.

\*\*Hystrix parameters:



\*\*Hystrix Dashboard

Bulkhead Pattern: maintain separate thread pool for each services



**Spring Microservices configuration with Spring Boot**

