

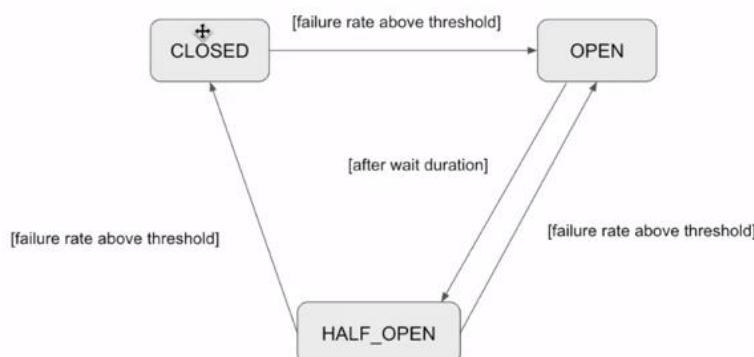


## Resilience4j

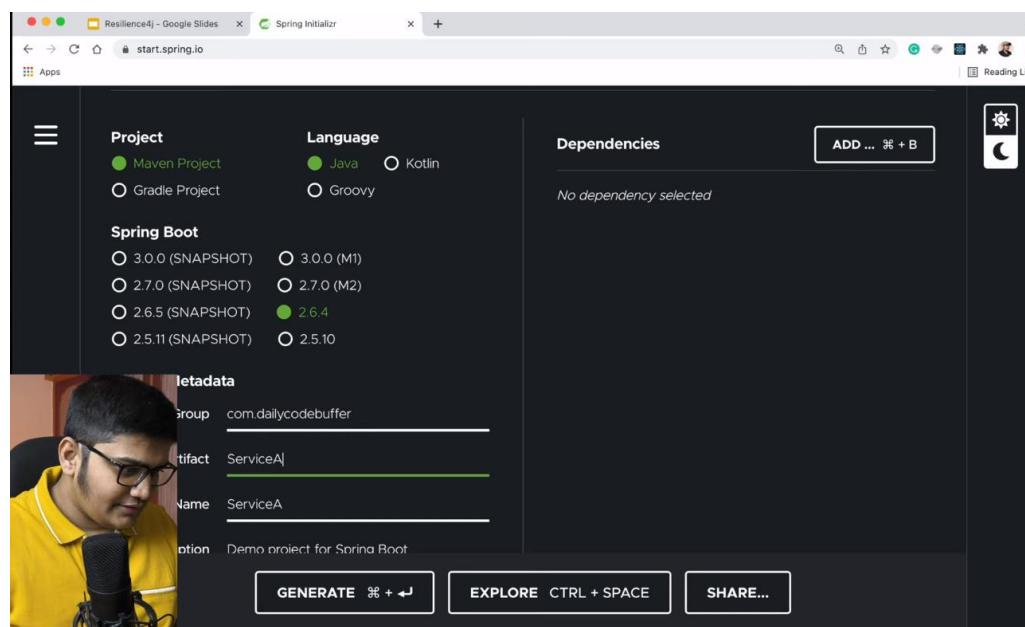
- Circuit Breaker
- Retry
- Rate Limiter
- Bulkhead
- Time Limiter
- Cache

There are different modules that we can use within our architecture

### Circuit Breaker



You can define the request X% **threshold** per N requests for the circuit to **open** for the T **wait period** in seconds. We need to add the configuration for this in our YAML file.

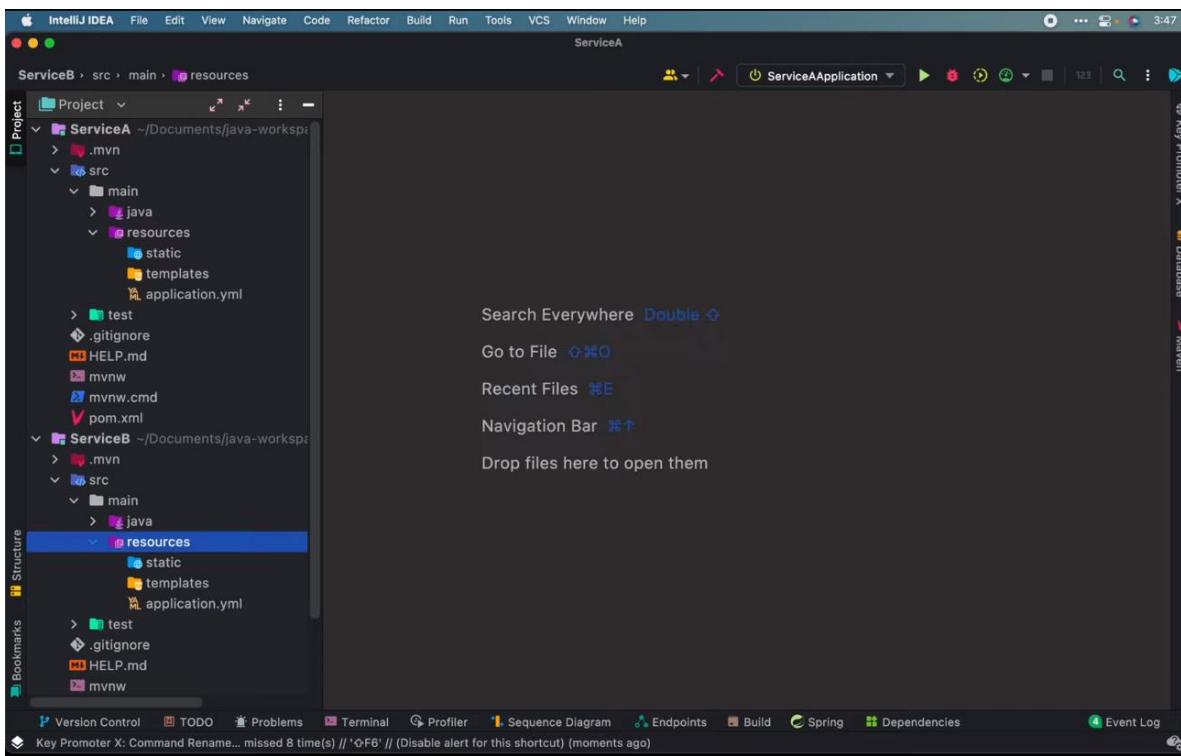
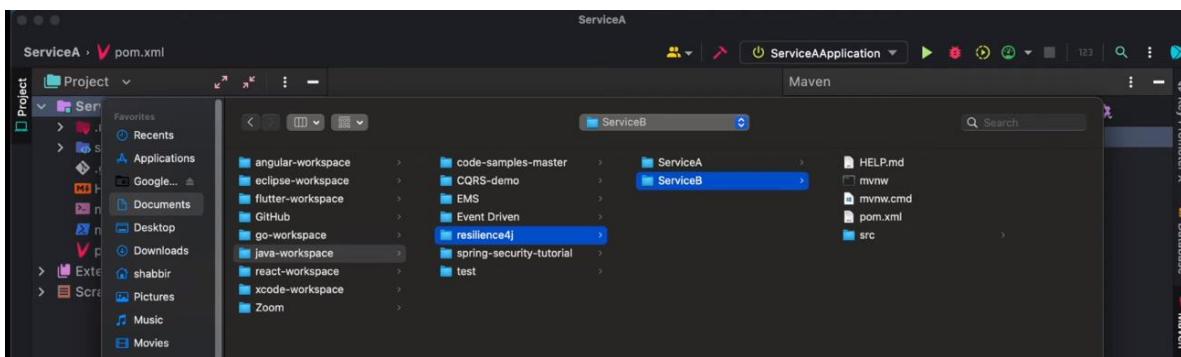
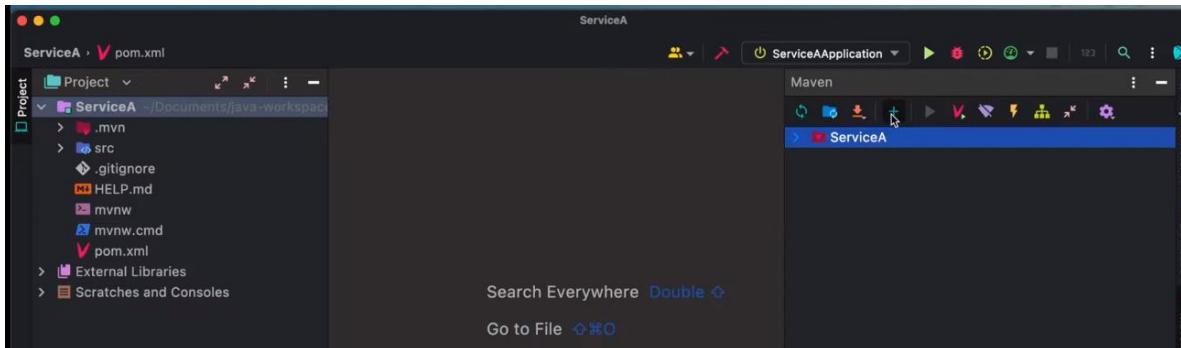
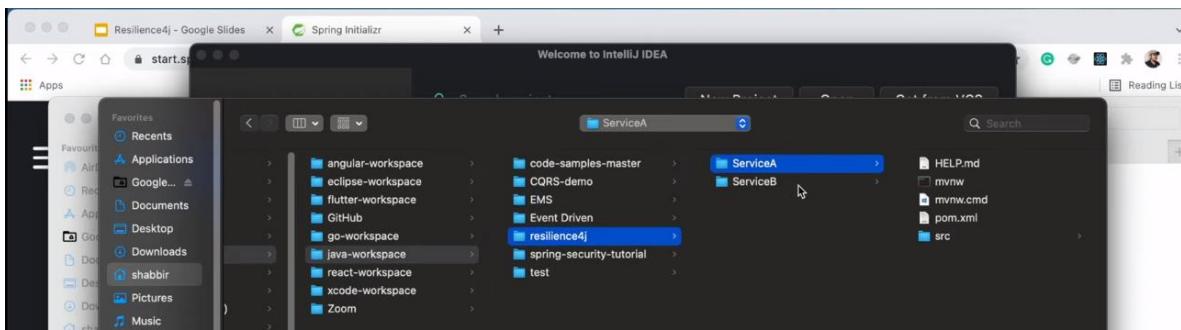


The screenshot shows the Spring Initializr web application. On the left, there's a sidebar with a video thumbnail of a person speaking. The main area has sections for 'Project' (Maven Project selected), 'Language' (Java selected), 'Spring Boot' (2.6.4 selected), and 'Dependencies'. Under 'Dependencies', 'Spring Web' (WEB) is selected, and 'Resilience4J' (SPRING CLOUD CIRCUIT BREAKER) is also selected. A large 'ADD ... ⌘ + B' button is at the top right.

This screenshot shows the 'Project Metadata' section of the Spring Initializr interface. It includes fields for 'Group' (com.dailycodebuffer), 'Artifact' (ServiceB), 'Name' (ServiceB), 'Description' (Demo project for Spring Boot), and 'Name' (com.dailycodebuffer.ServiceB). The 'Packaging' option is set to 'Jar'. The right side of the screen displays the 'Resilience4J' dependency details.

The screenshot shows the final configuration of the project. The 'Project' section shows Maven Project selected. The 'Dependencies' section lists 'Spring Web' (WEB) and 'Resilience4J' (SPRING CLOUD CIRCUIT BREAKER). At the bottom, there are three buttons: 'GENERATE ⌘ + ↵', 'EXPLORE ⌄ + SPACE', and 'SHARE...'. The video thumbnail on the left shows the person concluding the presentation.

This screenshot shows a Mac OS X desktop with a Finder window open. The path 'resilience4j' is shown in the sidebar. Inside, there are two folders: 'ServiceA' and 'ServiceB'. The 'ServiceA' folder was created by the generated code. The desktop also has a 'Resilience4 - Google Slides' window and a 'Spring Initializr' browser tab.



IntelliJ IDEA 2023.2.3

ServiceA - ServiceAController.java [ServiceA]

```
1 package com.dailycodebuffer.ServiceA.controller;
2
3 import org.springframework.web.bind.annotation.RestController;
4
5 @RestController
6 public class ServiceAController {
7 }
8
```

Project

- ServiceA
- ServiceB

Structure

Bookmarks

Version Control TODO Problems Terminal Profiler Sequence Diagram Endpoints Build Spring Dependencies Event Log

Key Promoter X: Command Rename... missed 8 time(s) // ⌘F6 // (Disable alert for this shortcut) (a minute ago)

IntelliJ IDEA 2023.2.3

ServiceB - ServiceBController.java [ServiceB]

```
1 package com.dailycodebuffer.ServiceB.controller;
2
3 import org.springframework.web.bind.annotation.RestController;
4
5 @RestController
6 public class ServiceBController {
7 }
8
```

Project

- ServiceA
- ServiceB

Structure

Bookmarks

Version Control TODO Problems Terminal Profiler Sequence Diagram Endpoints Build Spring Dependencies Event Log

Key Promoter X: Command Rename... missed 8 time(s) // ⌘F6 // (Disable alert for this shortcut) (a minute ago)

IntelliJ IDEA 2023.2.3

ServiceA - application.yml [ServiceA]

```
1 server:
2   port: 8080
```

Project

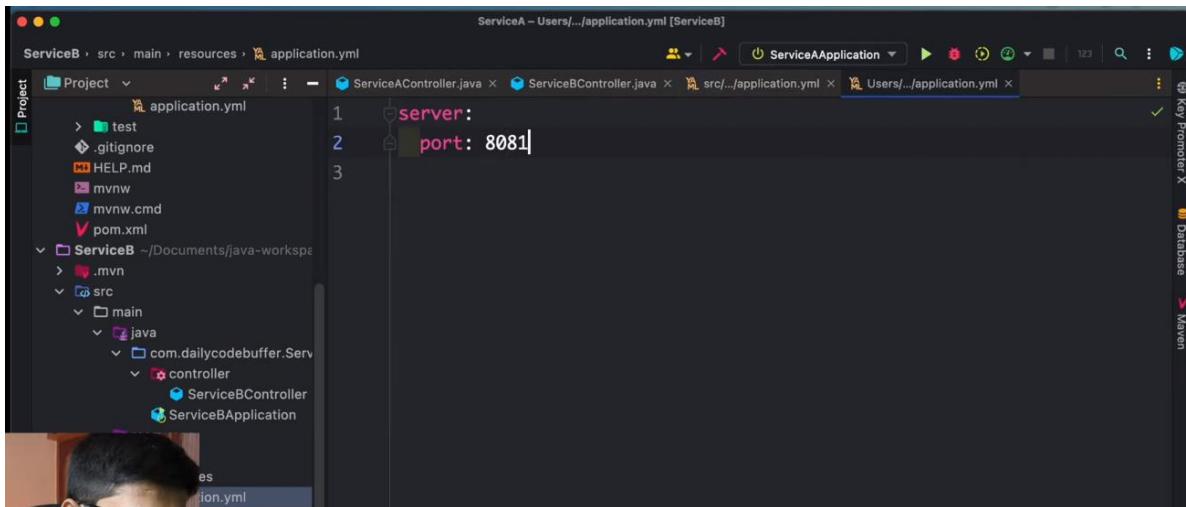
- ServiceA
- ServiceB

Structure

Bookmarks

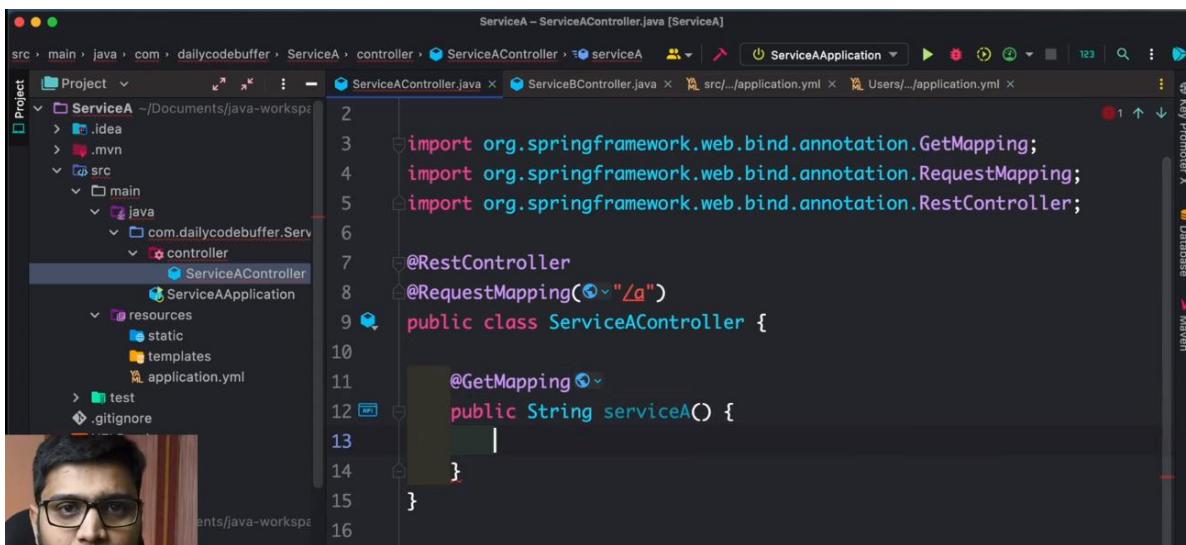
Version Control TODO Problems Terminal Profiler Sequence Diagram Endpoints Build Spring Dependencies Event Log

Key Promoter X: Command Rename... missed 8 time(s) // ⌘F6 // (Disable alert for this shortcut) (a minute ago)



The screenshot shows the IntelliJ IDEA interface with the project 'ServiceB' open. The 'application.yml' file is being edited. The configuration includes:

```
server:
  port: 8081
```



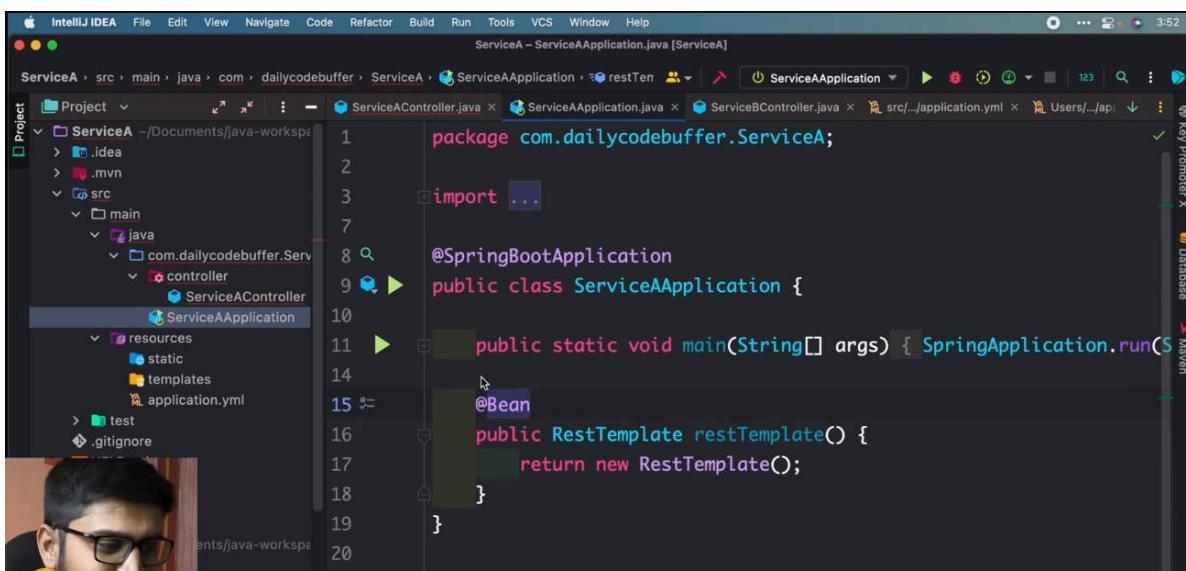
The screenshot shows the IntelliJ IDEA interface with the project 'ServiceA' open. The 'ServiceAController.java' file is being edited. The code defines a controller for service A:

```
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/a")
public class ServiceAController {

    @GetMapping
    public String serviceA() {
        return "Service A";
    }
}
```

Let us now implement the circuit breaker, retry and rate limiter in the services, we want to call service B from service A using the RestTemplate or WebClient by creating the bean and autowiring it.



The screenshot shows the IntelliJ IDEA interface with the project 'ServiceA' open. The 'ServiceAApplication.java' file is being edited. The code sets up a Spring Boot application and provides a RestTemplate bean:

```
package com.dailycodebuffer.ServiceA;

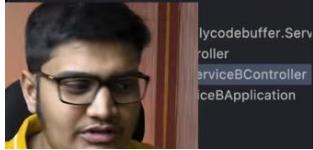
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;
import org.springframework.web.client.RestTemplate;

@SpringBootApplication
public class ServiceAApplication {

    public static void main(String[] args) {
        SpringApplication.run(ServiceAApplication.class, args);
    }

    @Bean
    public RestTemplate restTemplate() {
        return new RestTemplate();
    }
}
```

We can now go use this bean in our controller below.



ServiceA – ServiceBController.java [ServiceB]

```
src main java com dailycodebuffer ServiceB controller ServiceBController serviceB ServiceApplication ServiceBController.java ServiceAApplication.java ServiceBController.java src.../application.yml Users/.../ap...
```

Project

```
2
3 import org.springframework.web.bind.annotation.GetMapping;
4 import org.springframework.web.bind.annotation.RequestMapping;
5 import org.springframework.web.bind.annotation.RestController;
6
7 @RestController
8 @RequestMapping("/b")
9 public class ServiceBController {
10
11     @GetMapping
12     public String serviceB() {
13         return "Service B is called from Service A";
14     }
15
16 }
17
```

Key Promoter X Database Maven



ServiceA – ServiceAController.java [ServiceA]

```
src main java com dailycodebuffer ServiceA controller ServiceAController serviceA ServiceApplication ServiceAController.java ServiceAApplication.java ServiceBController.java src.../application.yml Users/.../ap...
```

Project

```
7 import org.springframework.web.client.RestTemplate;
8
9 @RestController
10 @RequestMapping("/a")
11 public class ServiceAController {
12
13     @Autowired
14     private RestTemplate restTemplate;
15
16     @GetMapping
17     public String serviceA() {
18
19     }
20 }
21
```

Key Promoter X Database Maven



ServiceA – ServiceAController.java [ServiceA]

```
src main java com dailycodebuffer ServiceA controller ServiceAController serviceA ServiceApplication ServiceAController.java ServiceAApplication.java ServiceBController.java src.../application.yml Users/.../ap...
```

Project

```
10 @RequestMapping("/a")
11 public class ServiceAController {
12
13     @Autowired
14     private RestTemplate restTemplate;
15
16     private static final String BASE_URL
17         = "http://localhost:8081";
18
19     @GetMapping
20     public String serviceA() {
21         return restTemplate.getForObject(
22             url:BASE_URL + "/b",
23             String.class
24         );
25     }
26 }
27
```

Key Promoter X Database Maven



ServiceA – ServiceAController.java [ServiceA]

```
src > main > java > com > dailycodebuffer > ServiceA > controller > ServiceAController > serviceA > ServiceApplication
```

Project

```
ServiceA ~/Documents/java-workspace
> .idea
> .mvn
> src
  > main
    > java
      > com.dailycodebuffer.ServiceA
        > controller
          > ServiceAController
        > ServiceAApplication
      > resources
        > static
        > templates
        > application.yml
    > test
    > .gitignore
```

```
13
14     @Autowired
15     private RestTemplate restTemplate;
16
17     private static final String BASE_URL
18         = "http://localhost:8081/";
19
20     @GetMapping
21     public String serviceA() {
22         String url = BASE_URL + "b";
23         return restTemplate.getForObject(
24             url,
25             String.class
26         );
27     }
28 }
```

Key Promoter X Database Maven

We are now calling service B from service A. then we run the services

Services

```
Spring Boot
  > Running
    > ServiceAApplication :8080/
      > ServiceBApplication :8081/
```

Console Actuator

```
2022-02-27 15:57:29.610 INFO 4765 --- [main] o.a.c.c.C.[Tomcat].[localhost].w.s.c.Ser... :main] o.s.b.w.e.c.d.Servi...
```

Version Control TODO Problems Terminal Profiler Sequence Diagram Services Endpoints Build Spring Dependencies Event Log

Chrome File Edit View History Bookmarks Profiles Tab Window Help

localhost:8080/a

Service B is called from Service A

ServiceA – ServiceAController.java [ServiceA]

```
src > main > java > com > dailycodebuffer > ServiceA > controller > ServiceAController > serviceA > ServiceApplication
```

Project

```
ServiceA ~/Documents/java-workspace
> .idea
> .mvn
> src
  > main
    > java
      > com.dailycodebuffer.ServiceA
        > controller
          > ServiceAController
        > ServiceAApplication
      > resources
        > static
        > templates
        > application.yml
    > test
    > target
```

```
15
16     private static final String BASE_URL
17         = "http://localhost:8081/";
18
19     @GetMapping
20     public String serviceA() {
21         String url = BASE_URL + "b";
22         return restTemplate.getForObject(
23             url,
24             String.class
25         );
26     }
27 }
```

Services

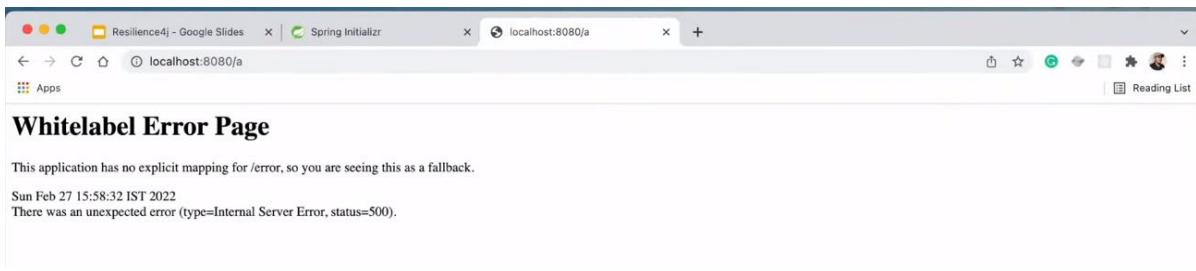
```
Spring Boot
  > Running
    > ServiceAApplication :8080/
  > Finished
    > ServiceBApplication
```

Console Actuator

```
2022-02-27 15:57:50.116 INFO 4765 --- [nio-8081-exec-1] o.s.web.s... :main] o.a.c.c.C.[Tomcat].[localhost].w.s.c.Ser... :main] o.s.b.w.e.c.d.Servi...
```

Process finished with exit code 130 Key Promoter X Command Stop missed 4 time(s)

Next, let us kill service B above, we will now get the response below

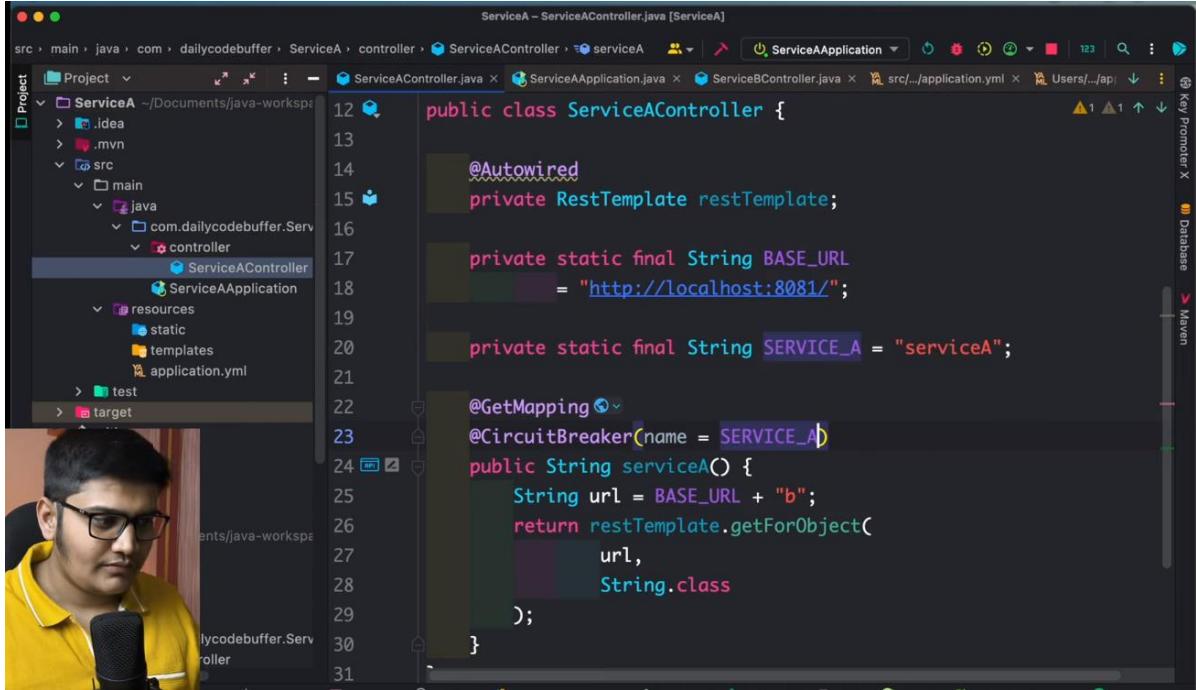


This application has no explicit mapping for /error, so you are seeing this as a fallback.

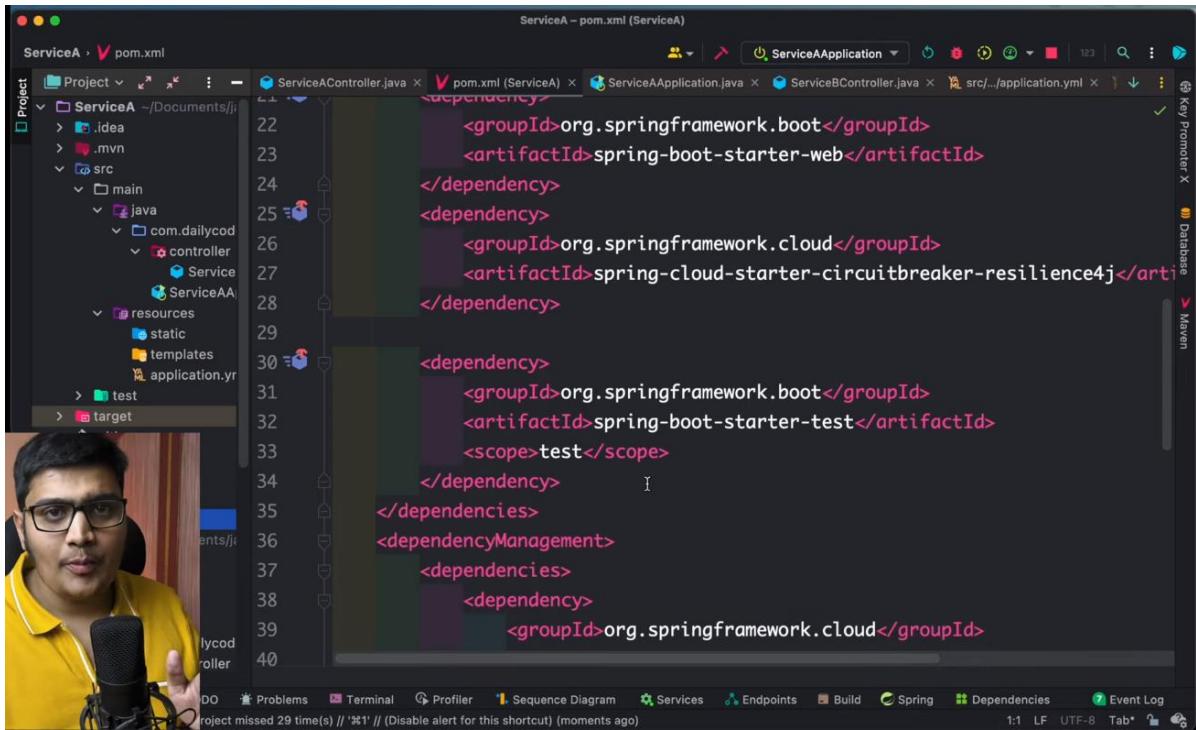
Sun Feb 27 15:58:32 IST 2022

There was an unexpected error (type=Internal Server Error, status=500).

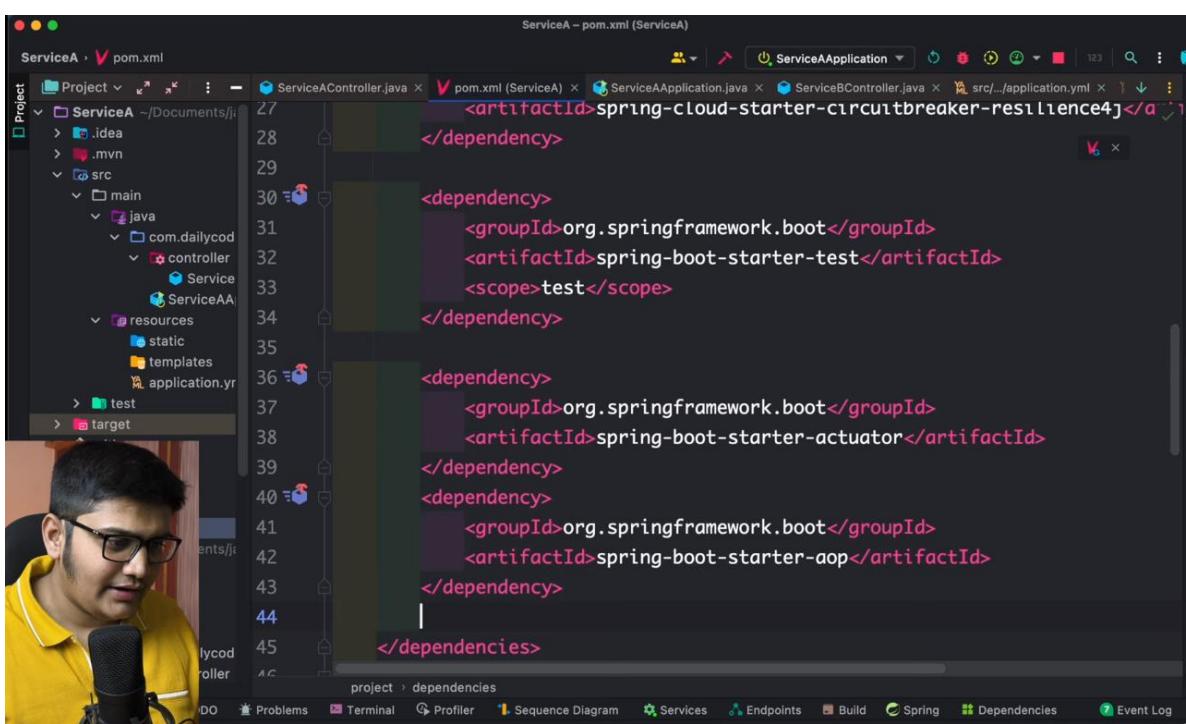
Let us now implement the circuit breaker pattern for the scenario where service B is down to use a fallback method



We add the `service_name` and the `@CircuitBreaker` annotation from Resilience4j to our service A endpoints as above.



To service A, we need to add the AOP dependency and the actuator dependency as below



ServiceA - pom.xml (ServiceA)

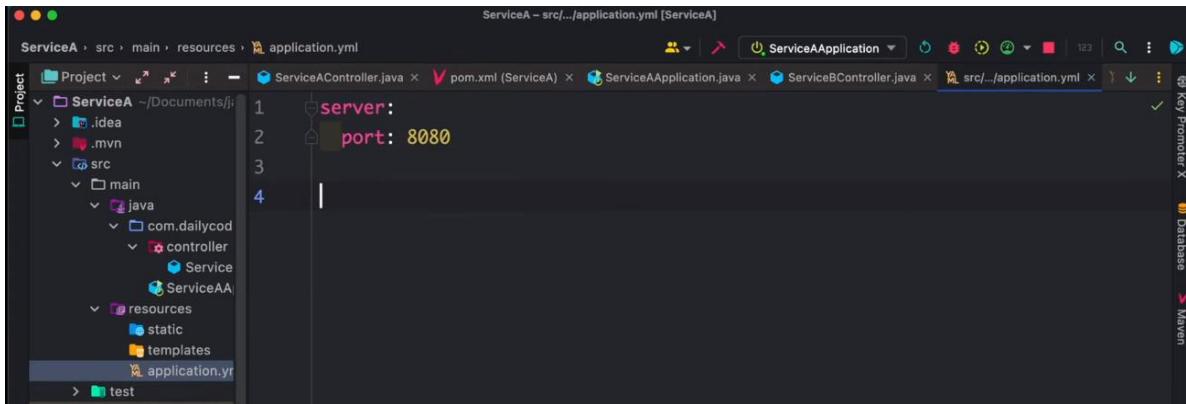
```
<dependency>
    <artifactId>spring-cloud-starter-circuitbreaker-resilience4j</artifactId>
</dependency>

<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
</dependency>

<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-aop</artifactId>
</dependency>

```

project dependencies Problems Terminal Profiler Sequence Diagram Services Endpoints Build Spring Dependencies Event Log

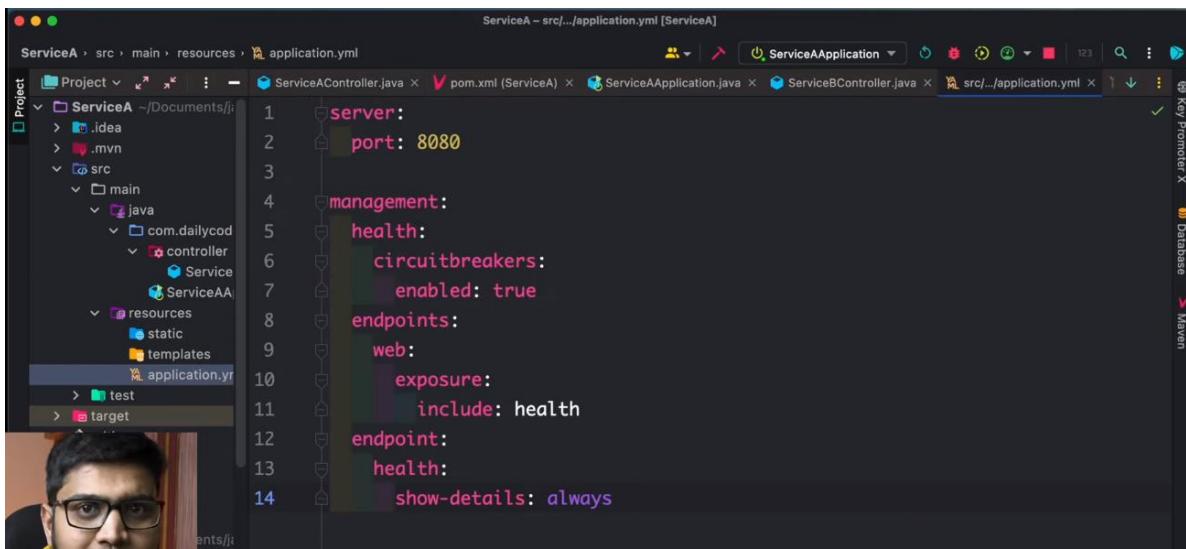


ServiceA - src/main/resources/application.yml

```
server:
  port: 8080
```

Project .idea .mvn src main java com.dailycod controller Service ServiceAA resources static templates application.yr test target

Next, we then add the circuit breaker and fallback configurations to our YAML file as below.



ServiceA - src/main/resources/application.yml

```
server:
  port: 8080

management:
  health:
    circuitbreakers:
      enabled: true
  endpoints:
    web:
      exposure:
        include: health
  endpoint:
    health:
      show-details: always
```

Project .idea .mvn src main java com.dailycod controller Service ServiceAA resources static templates application.yr test target

We enable the management endpoints for our actuators as above



ServiceA - src/.../application.yml [ServiceA]

```
12 endpoint:
13     health:
14         show-details: always
15
16 resilience4j:
17     circuitbreaker:
18         instances:
19             serviceA:
20                 registerHealthIndicator: true
21                 eventConsumerBufferSize: 10
22                 failureRateThreshold: 50
23                 minimumNumberOfCalls: 5
24                 automaticTransitionFromOpenToHalfOpenEnabled: true
25                 waitDurationInOpenState: 5s
26                 permittedNumberOfCallsInHalfOpenState: 3
27                 slidingWindowSize: 10
28                 slidingWindowType: COUNT_BASED
```

We then add the configuration for our Resilience4j circuit breaker for all our instances above



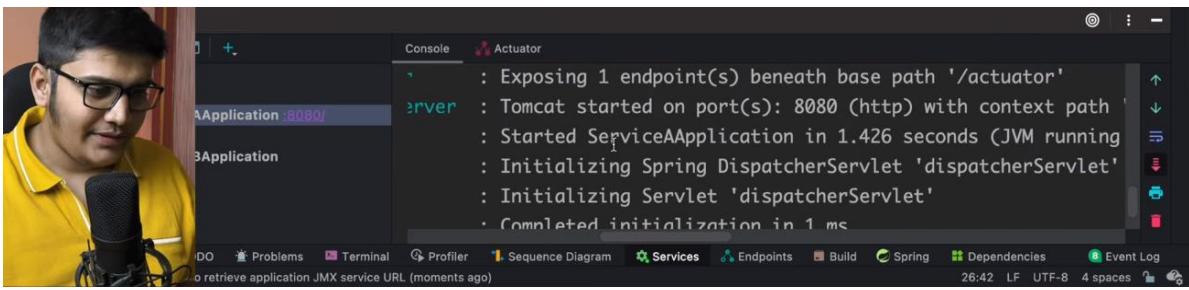
ServiceA - ServiceAController.java [ServiceA]

```
11 @RequestMapping("/*")
12 public class ServiceAController {
13     = "http://localhost:8081/";
14
15     private static final String SERVICE_A = "serviceA";
16
17     @GetMapping
18     @CircuitBreaker(name = SERVICE_A)
19     public String serviceA() {
20         String url = BASE_URL + "b";
21         return restTemplate.getForObject(
22             url,
23             String.class
24         );
25     }
26 }
```



```
src / main / java / com / dailycodebuffer / ServiceA / controller / ServiceAController.java
Project ServiceA ~/Documents/ji/.idea .mvn src main Java com.dailycod.controller ServiceA Application.java pom.xml ServiceBController.java src/.../application.yml
10
11     private static final String BASE_URL
12     = "http://localhost:8081/";
13
14     private static final String SERVICE_A = "serviceA";
15
16     @GetMapping
17     @CircuitBreaker(name = SERVICE_A, fallbackMethod = "serviceAFallback")
18     public String serviceA() {
19         String url = BASE_URL + "b";
20         return restTemplate.getForObject(
21             url,
22             String.class
23         );
24     }
25
26     public String serviceAFallback(Exception e) {
27         return "This is a fallback method for Service A";
28     }
29 }
```

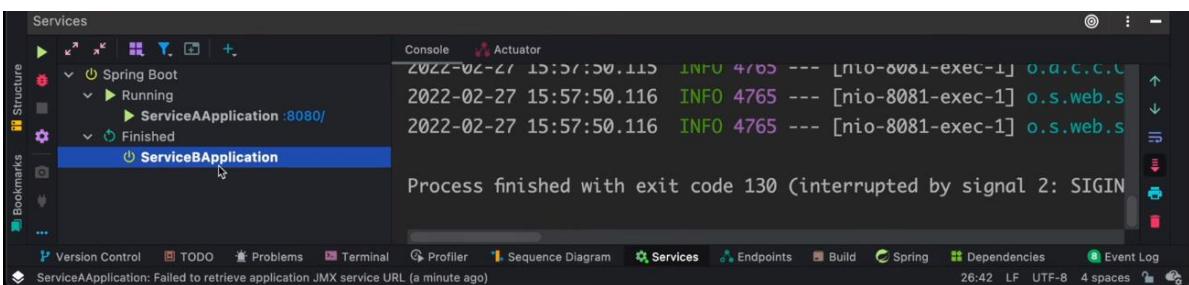
We create our fallback method that injects the exception and use it in the circuitbreaker method as above



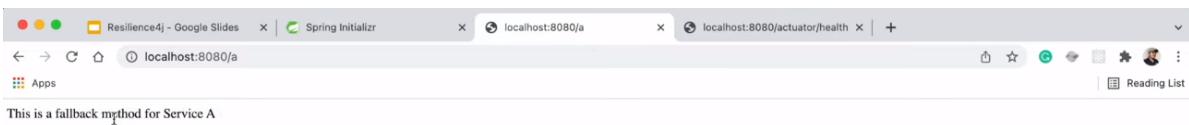
Restart app

```

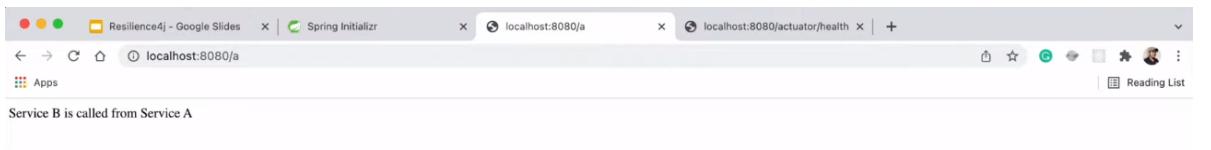
1 // 20220227160731
2 // http://localhost:8080/actuator/health
3
4 {
5   "status": "UP",
6   "components": {
7     "circuitBreakers": {
8       "status": "UP",
9       "details": {
10         "serviceA": {
11           "status": "UP",
12           "details": {
13             "failureRate": "-1.0%",
14             "failureRateThreshold": "50.0%",
15             "slowCallRate": "-1.0%",
16             "slowCallRateThreshold": "100.0%",
17             "bufferedCalls": 0,
18             "slowCalls": 0,
19             "slowFailedCalls": 0,
20             "failedCalls": 0,
21             "notPermittedCalls": 0,
22             "state": "CLOSED"
23           }
24         }
25       }
26     },
27     "discoveryComposite": {
28       "description": "Discovery Client not initialized",
29       "status": "UNKNOWN",
30     }
31   }
32 }
33 
```



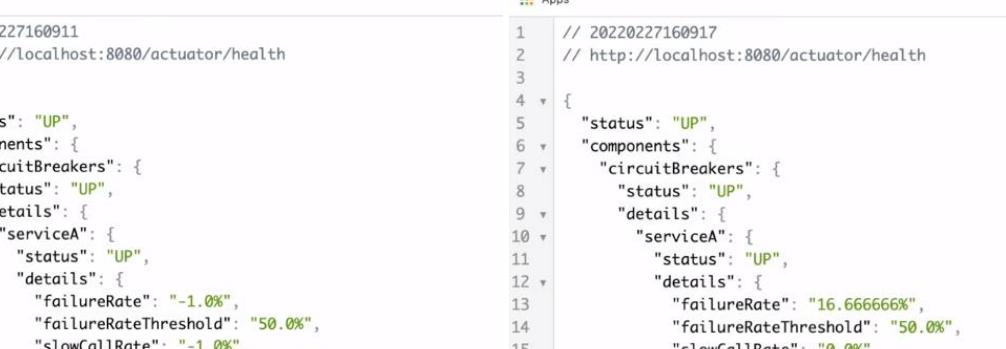
Note that service B is currently down



We can now see the fallback method response instead of a 404 page



Restart service B again



The screenshot displays two browser tabs side-by-side, both showing the `/actuator/health` endpoint for a Spring application running on port 8080.

**Left Tab (Healthy State):**

```
// 20220227160911
// http://localhost:8080/actuator/health

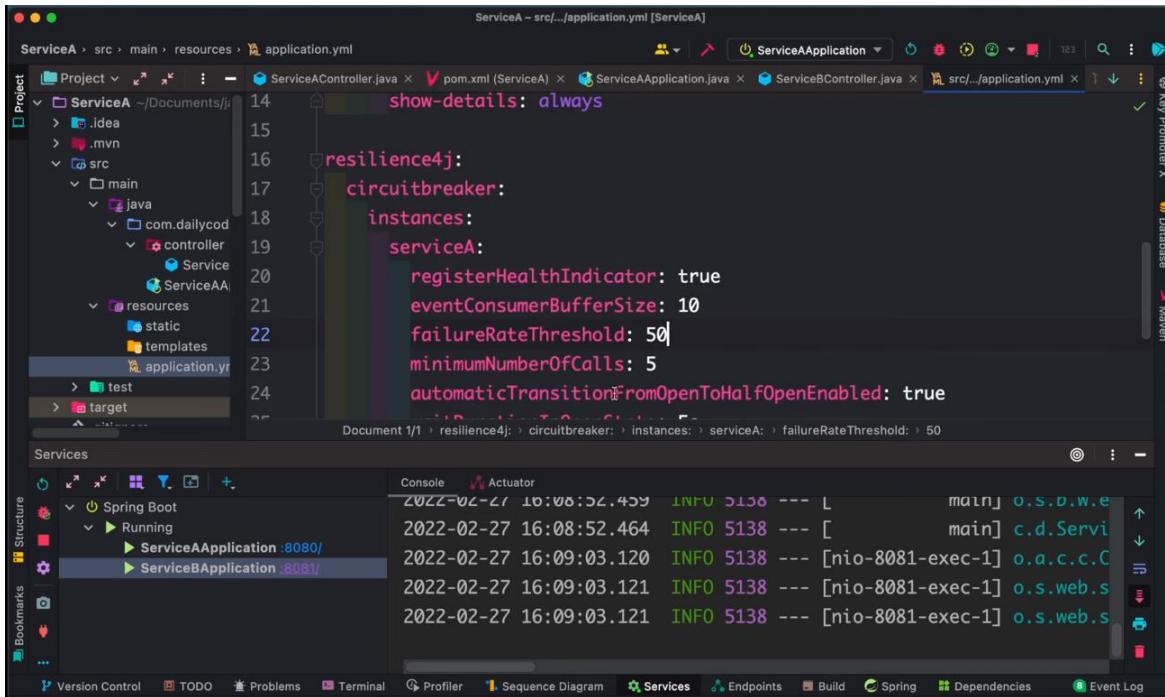
{
  "status": "UP",
  "components": {
    "circuitBreakers": {
      "status": "UP",
      "details": {
        "serviceA": {
          "status": "UP",
          "details": {
            "failureRate": "-1.0%",
            "failureRateThreshold": "50.0%",
            "slowCallRate": "-1.0%",
            "slowCallRateThreshold": "100.0%",
            "bufferedCalls": 2,
            "slowCalls": 0,
            "slowFailedCalls": 0,
            "failedCalls": 1,
            "notPermittedCalls": 0,
            "state": "CLOSED"
          }
        }
      }
    },
    "discoveryComposite": {
      "description": "Discovery Client not initialized",
      "status": "UNKNOWN",
      "details": {}
    }
  }
}
```

**Right Tab (Unhealthy State):**

```
// 20220227160917
// http://localhost:8080/actuator/health

{
  "status": "UP",
  "components": {
    "circuitBreakers": {
      "status": "UP",
      "details": {
        "serviceA": {
          "status": "UP",
          "details": {
            "failureRate": "16.666666%",
            "failureRateThreshold": "50.0%",
            "slowCallRate": "0.0%",
            "slowCallRateThreshold": "100.0%",
            "bufferedCalls": 6,
            "slowCalls": 0,
            "slowFailedCalls": 0,
            "failedCalls": 1,
            "notPermittedCalls": 0,
            "state": "CLOSED"
          }
        }
      }
    },
    "discoveryComposite": {
      "description": "Discovery Client not initialized",
      "status": "UNKNOWN",
      "details": {}
    }
}
```

Note that



```

Resilience4j - Google Slides | Spring Initializr | localhost:8080/a | localhost:8080/actuator/health | + | Apps | Reading List
This is a fallback method for Service A

```

```

// 20220227161004
// http://localhost:8080/actuator/health
{
  "status": "UP",
  "components": {
    "circuitBreakers": {
      "status": "UNKNOWN",
      "details": {
        "serviceA": {
          "status": "CIRCUIT_OPEN",
          "details": {
            "failureRate": "50.0%",
            "failureRateThreshold": "50.0%",
            "slowCallRate": "0.0%",
            "slowCallRateThreshold": "100.0%",
            "bufferedCalls": 10,
            "slowCalls": 0,
            "slowFailedCalls": 0,
            "failedCalls": 5,
            "notPermittedCalls": 1,
            "state": "OPEN"
          }
        }
      }
    },
    "discoveryComposite": {
      "description": "Discovery Client not initialized",
      "status": "UNKNOWN",
    }
  }
}

// 20220227161013
// http://localhost:8080/actuator/health
{
  "status": "UP",
  "components": {
    "circuitBreakers": {
      "status": "UNKNOWN",
      "details": {
        "serviceA": {
          "status": "CIRCUIT_HALF_OPEN",
          "details": {
            "failureRate": "-1.0%",
            "failureRateThreshold": "50.0%",
            "slowCallRate": "-1.0%",
            "slowCallRateThreshold": "100.0%",
            "bufferedCalls": 1,
            "slowCalls": 0,
            "slowFailedCalls": 0,
            "failedCalls": 1,
            "notPermittedCalls": 0,
            "state": "HALF_OPEN"
          }
        }
      }
    },
    "discoveryComposite": {
      "description": "Discovery Client not initialized",
      "status": "UNKNOWN",
    }
}

```

```

Resilience4j - Google Slides | Spring Initializr | localhost:8080/a | localhost:8080/actuator/health | + | Apps | Reading List
This is a fallback method for Service A

```

```

// 20220227161044
// http://localhost:8080/actuator/health
{
  "status": "UP",
  "components": {
    "circuitBreakers": {
      "status": "UNKNOWN",
      "details": {
        "serviceA": {
          "status": "CIRCUIT_OPEN",
          "details": {
            "failureRate": "100.0%",
            "failureRateThreshold": "50.0%",
            "slowCallRate": "0.0%",
            "slowCallRateThreshold": "100.0%",
            "bufferedCalls": 3,
            "slowCalls": 0,
            "slowFailedCalls": 0,
            "failedCalls": 3,
            "notPermittedCalls": 1,
            "state": "OPEN"
          }
        }
      }
    },
    "discoveryComposite": {
      "description": "Discovery Client not initialized",
      "status": "UNKNOWN",
    }
  }
}

// 20220227161048
// http://localhost:8080/actuator/health
{
  "status": "UP",
  "components": {
    "circuitBreakers": {
      "status": "UNKNOWN",
      "details": {
        "serviceA": {
          "status": "CIRCUIT_HALF_OPEN",
          "details": {
            "failureRate": "-1.0%",
            "failureRateThreshold": "50.0%",
            "slowCallRate": "-1.0%",
            "slowCallRateThreshold": "100.0%",
            "bufferedCalls": 0,
            "slowCalls": 0,
            "slowFailedCalls": 0,
            "failedCalls": 0,
            "notPermittedCalls": 0,
            "state": "HALF_OPEN"
          }
        }
      }
    },
    "discoveryComposite": {
      "description": "Discovery Client not initialized",
      "status": "UNKNOWN",
    }
}

```

ServiceA – src.../application.yml [ServiceA]

```

ServiceA > src > main > resources > application.yml
Project ServiceA ~/Documents/ji
> .idea
> .mvn
> src
  > main
    > java
      > com.dailycod
        > controller
          > Service
          > ServiceAA
    > resources
      > static
      > templates
        > application.y
  > test
  > target

14 show-details: always
15
16 resilience4j:
17   circuitbreaker:
18     instances:
19       serviceA:
20         registerHealthIndicator: true
21         eventConsumerBufferSize: 10
22         failureRateThreshold: 50
23         minimumNumberOfCalls: 5
24         automaticTransitionFromOpenToHalfOpenEnabled: true

```

Document 1/1 resilience4j: > circuitbreaker: > instances: > serviceA: > failureRateThreshold: > 50

Services

Structure Bookmarks

Spring Boot Running ServiceAApplication:8080/ ServiceBApplication:8081/

Console Actuator

```

2022-02-27 16:11:00.204 INFO 5199 --- [           main] org.apache.catalina.core.StandardEngine  : [main] o.a.c.c.C.[localhost].w.s.c.Ser...
2022-02-27 16:11:00.226 INFO 5199 --- [           main] org.apache.catalina.core.StandardEngine  : [main] o.s.b.w.e...
2022-02-27 16:11:00.351 INFO 5199 --- [           main] org.apache.catalina.core.StandardEngine  : [main] c.d.Servi...
2022-02-27 16:11:00.356 INFO 5199 --- [           main] org.apache.catalina.core.StandardEngine  : [main] ...

```

Version Control TODO Problems Terminal Profiler Sequence Diagram Services Endpoints Build Dependencies Event Log

Resilience4j - Google Slides | Spring Initializr | localhost:80

localhost:8080/actuator/health

Apps

```

1 // 20220227161105
2 // http://localhost:8080/actuator/health
3
4 {
5   "status": "UP",
6   "components": {
7     "circuitBreakers": {
8       "status": "UNKNOWN",
9       "details": {
10         "serviceA": {
11           "status": "CIRCUIT_HALF_OPEN",
12           "details": {
13             "failureRate": "-1.0%",
14             "failureRateThreshold": "50.0%",
15             "slowCallRate": "-1.0%",
16             "slowCallRateThreshold": "100.0%",
17             "bufferedCalls": 0,
18             "slowCalls": 0,
19             "slowFailedCalls": 0,
20             "failedCalls": 0,
21             "notPermittedCalls": 0,
22             "state": "HALF_OPEN"
23           }
24         }
25       }
26     },
27     "discoveryComposite": {
28       "description": "Discovery Client not initialized",
29       "status": "UNKNOWN",

```

Resilience4j - Google Slides | Spring Initializr | localhost:8080/a | localhost:8080/actuator/health | +

localhost:8080/a

Apps

This is a fallback method for Service A

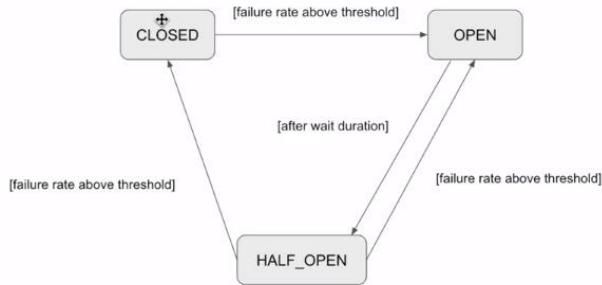
Resilience4j - Google Slides | Spring Initializr | localhost:8080/a | localhost:8080/actuator/health | +

localhost:8080/a

Apps

Service B is called from Service A

## Circuit Breaker



Next, let us now implement the Retry mechanism for our applications

ServiceA – ServiceAController.java [ServiceA]

```
private static final String SERVICE_A = "serviceA";  
  
@GetMapping  
//@CircuitBreaker(name = SERVICE_A, fallbackMethod = "serviceAFallback")  
@Retry(name = SERVICE_A)  
public String serviceA() {  
    int count=1;  
    String url = BASE_URL + "b";  
    System.out.println("Retry method called " + count++ + " times at |");  
    return restTemplate.getForObject(  
        url,  
        String.class  
);
```

Services

- Spring Boot
- Running
- ServiceApplication :8080/
- ServiceBApplication :8081/

Console

```
2022-02-27 16:11:26.348 INFO 5199 --- [nio-8081-exec-1] o.a.c.c.C ...  
2022-02-27 16:11:26.349 INFO 5199 --- [nio-8081-exec-1] o.s.web.S ...  
2022-02-27 16:11:26.350 INFO 5199 --- [nio-8081-exec-1] o.s.web.S ...
```

This screenshot shows the IntelliJ IDEA interface with the `ServiceAController.java` file open. The code implements a retry mechanism using the `@Retry` annotation from the `com.netflix.hystrix` library. The `retryCount` is set to 1. The `url` is defined as `BASE_URL + "b"`. The `System.out.println` statement logs the number of retries. Below the code editor, the `Services` panel shows two running Spring Boot applications: `ServiceApplication` and `ServiceBApplication`. The `Console` tab displays log entries for both services, indicating successful requests.

We can now add the configuration details for the Retry

ServiceA – src/.../application.yml [ServiceA]

```
resilience4j:  
  circuitbreaker:  
    instances:  
      serviceA:  
        registerHealthIndicator: true  
        eventConsumerBufferSize: 10  
        failureRateThreshold: 50  
        minimumNumberOfCalls: 5  
        automaticTransitionFromOpenToHalfOpenEnabled: true  
        waitDurationInOpenState: 5s  
        permittedNumberOfCallsInHalfOpenState: 3  
        slidingWindowSize: 10  
        slidingWindowType: COUNT_BASED
```

This screenshot shows the `application.yml` configuration file for the `ServiceA` project. It defines a `circuitbreaker` instance named `serviceA`. The configuration includes settings for `registerHealthIndicator`, `eventConsumerBufferSize`, `failureRateThreshold`, `minimumNumberOfCalls`, `automaticTransitionFromOpenToHalfOpenEnabled`, `waitDurationInOpenState`, `permittedNumberOfCallsInHalfOpenState`, `slidingWindowSize`, and `slidingWindowType`.

ServiceA - src.../application.yml [ServiceA]

```
eventConsumerBufferSize: 10
failureRateThreshold: 50
minimumNumberOfCalls: 5
automaticTransitionFromOpenToHalfOpenEnabled: true
waitDurationInOpenState: 5s
permittedNumberOfCallsInHalfOpenState: 3
slidingWindowSize: 10
slidingWindowType: COUNT_BASED

retry:
  instances:
    serviceA:
      registerHealthIndicator: true
      maxRetryAttempts: 5
      waitDuration: 10s
```

mvnw.cmd 42

Services

Spring Boot

Running

ServiceAAApplication :8080/

ServiceBApplication :8081/

Console Actuator

2022-02-27 16:17:03.562 INFO 5347 --- [on(2)-127.0.0.1] o.s.web.s

2022-02-27 16:17:03.563 INFO 5347 --- [on(2)-127.0.0.1] o.s.web.s

Version Control TODO Problems Terminal Profiler Sequence Diagram Services Endpoints Build Spring Dependencies Event Log

Build completed successfully in 844 ms (moments ago)

localhost:8080/a

localhost:8080/actuator/health

Service B is called from Service A

localhost:8080/a

localhost:8080/actuator/health

ServiceAApplication :8080/

ServiceBApplication :8081/

Console Actuator

2022-02-27 16:17:03.563 INFO 5347 --- [on(2)-127.0.0.1] o.s.web.s

Retry method called 1 times at Sun Feb 27 16:17:10 IST 2022

Version Control TODO Problems Terminal Profiler Sequence Diagram Services Endpoints Build Spring Dependencies Event Log

Build completed successfully in 844 ms (moments ago)

localhost:8080/a

localhost:8080/actuator/health

ServiceAApplication :8080/

ServiceBApplication

Process finished with exit code 130

Key Promoter X

Command Stop missed 6 time(s)

'%F2' (Disable alert for this shortcut)

Version Control TODO Problems Terminal Profiler Sequence Diagram Services Endpoints Build Spring Dependencies Event Log

Key Promoter X: Command Stop missed 6 time(s) // '%F2' // (Disable alert for this shortcut) (moments ago)

Stop service B now

localhost:8080/a

localhost:8080/actuator/health

Service B is called from Service A

Services

- Spring Boot
  - Running
    - ServiceAApplication :8080/
  - Finished
    - ServiceBApplication

Console Actuator

```
2022-02-27 16:17:46.406 INFO 5372 --- [           main] o.s.b.a.e
2022-02-27 16:17:46.445 INFO 5372 --- [           main] o.s.b.w.e
2022-02-27 16:17:46.456 INFO 5372 --- [           main] c.d.Servi
2022-02-27 16:17:46.927 INFO 5372 --- [on(2)-127.0.0.1] o.a.c.c.C
2022-02-27 16:17:46.927 INFO 5372 --- [on(2)-127.0.0.1] o.s.web.s
2022-02-27 16:17:46.928 INFO 5372 --- [on(2)-127.0.0.1] o.s.web.s
Retry method called 1 times at Sun Feb 27 16:18:16 IST 2022
Retry method called 2 times at Sun Feb 27 16:18:26 IST 2022
```

Version Control TODO Problems Terminal Profiler Sequence Diagram Services Endpoints Build Spring Dependencies Event Log

Services

- Spring Boot
  - Running
    - ServiceAApplication :8080/
    - ServiceBApplication :8081/

Console Actuator

```
2022-02-27 16:17:46.927 INFO 5372 --- [on(2)-127.0.0.1] o.a.c.c.C
2022-02-27 16:17:46.927 INFO 5372 --- [on(2)-127.0.0.1] o.s.web.s
2022-02-27 16:17:46.928 INFO 5372 --- [on(2)-127.0.0.1] o.s.web.s
Retry method called 1 times at Sun Feb 27 16:18:16 IST 2022
Retry method called 2 times at Sun Feb 27 16:18:26 IST 2022
Retry method called 3 times at Sun Feb 27 16:18:36 IST 2022
Retry method called 4 times at Sun Feb 27 16:18:46 IST 2022
Retry method called 5 times at Sun Feb 27 16:18:56 IST 2022
```

Version Control TODO Problems Terminal Profiler Sequence Diagram Services Endpoints Build Spring Dependencies Event Log

Services

- Spring Boot
  - Running
    - ServiceAApplication :8080/
  - Finished
    - ServiceBApplication

Console Actuator

```
2022-02-27 16:18:49.641 INFO 5398 --- [           main] c.d.Servi
2022-02-27 16:18:49.643 INFO 5398 --- [           main] c.d.Servi
2022-02-27 16:18:49.972 INFO 5398 --- [           main] o.s.b.w.e
2022-02-27 16:18:49.976 INFO 5398 --- [           main] o.apache.
2022-02-27 16:18:49.976 INFO 5398 --- [           main] org.apach
2022-02-27 16:18:49.998 INFO 5398 --- [           main] o.a.c.c.C
2022-02-27 16:18:49.999 INFO 5398 --- [           main] w.s.c.Ser
2022-02-27 16:18:50.128 INFO 5398 --- [           main] o.s.b.w.e
2022-02-27 16:18:50.128 INFO 5398 --- [           main] o.s.b.w.e
```

Version Control TODO Problems Terminal Profiler Sequence Diagram Services Endpoints Build Spring Dependencies Event Log

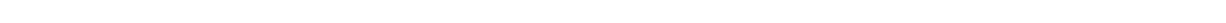
Start service B now

Resilience4J - Google Slides | Spring Initializr | localhost:8080/a | localhost:8080/actuator/health

All files are up-to-date (moments ago)

Service B is called from Service A

Service calls are now working again.



ServiceA - ServiceAController.java [ServiceA]

```
src > main > java > com > dailycodebuffer > ServiceA > controller > ServiceAController > serviceA
Project > ServiceA > .idea > .mvn > src > main > java > com.dailycod > controller > ServiceAController > ServiceAA > resources > static > templates > application.yml
```

```
27
28     @GetMapping
29     //@CircuitBreaker(name = SERVICE_A, fallbackMethod = "serviceAFallback")
30     //@Retry(name = SERVICE_A)
31     @RateLimiter(name = SERVICE_A)
32     public String serviceA() {
33
34         String url = BASE_URL + "b";
35         System.out.println("Retry method called " + count++ + " times at " + url);
36         return restTemplate.getForObject(
37             url,
38             String.class
39         );
40     }
41
42     public String serviceAFallback(Exception e) {
43         return "This is a fallback method for Service A";
44     }
45 }
```

Next, let us implement the Rate Limiter using Resilience4j

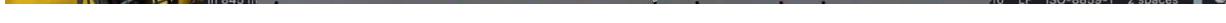


ServiceA - src/main/resources/application.yml [ServiceA]

```
src > main > resources > application.yml
```

```
28     slidingWindowType: COUNT_BASED
29     retry:
30         instances:
31             serviceA:
32                 registerHealthIndicator: true
33                 maxRetryAttempts: 5
34                 waitDuration: 10s
35     ratelimiter:
36         instances:
37             serviceA:
38                 registerHealthIndicator: false
39                 limitForPeriod: 10
40                 limitRefreshPeriod: 10s
41                 timeoutDuration: 3s
```

If you have multiple instances, you can define the default configuration too

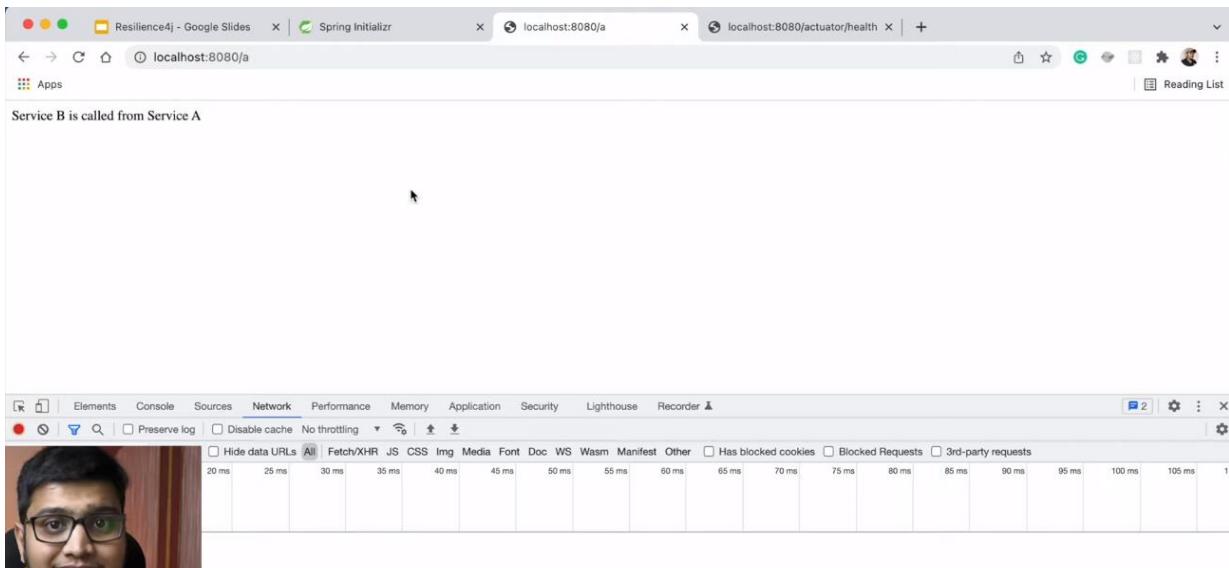


Services

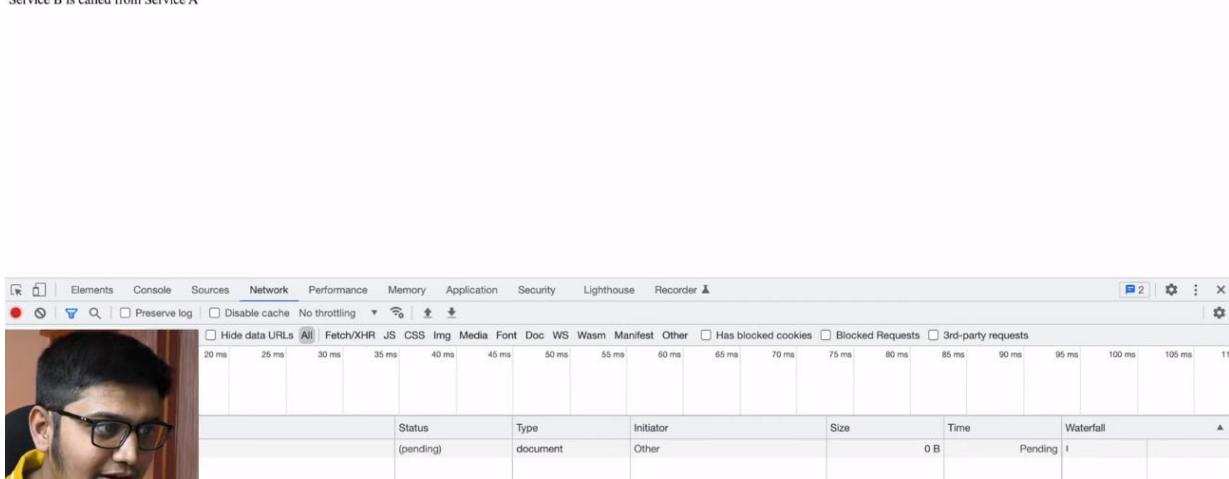
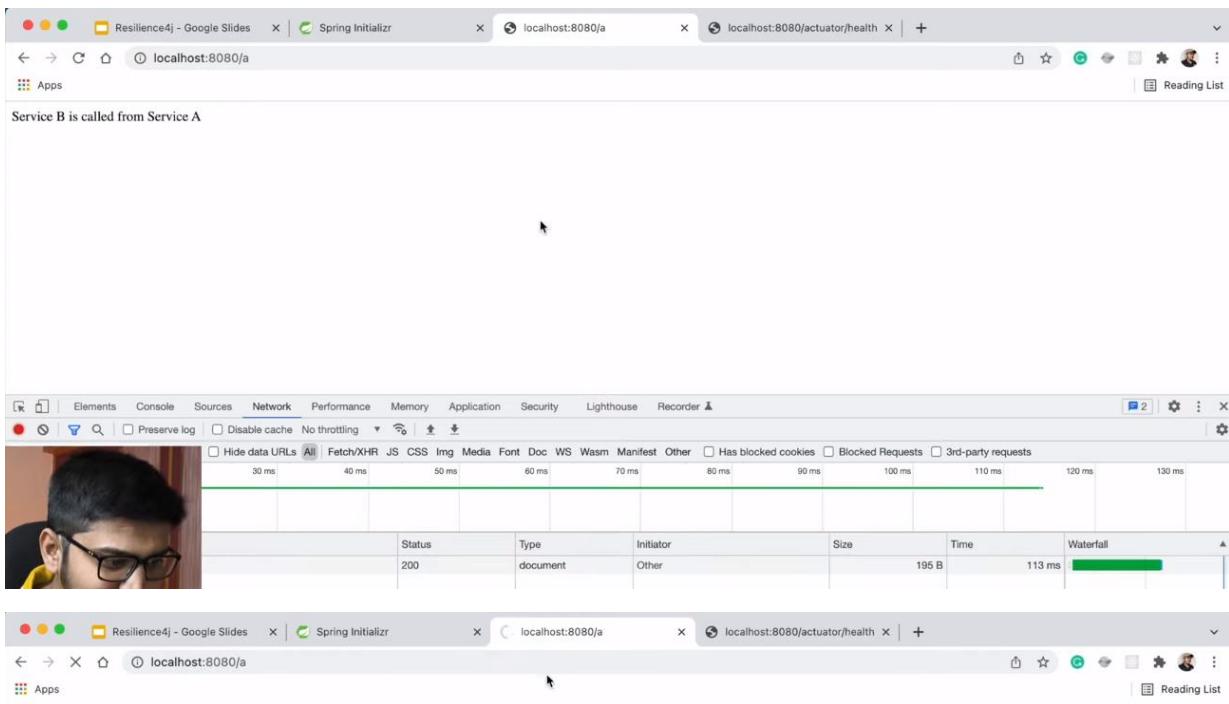
```
Spring Boot > Application :8080/ > Running
```

Console

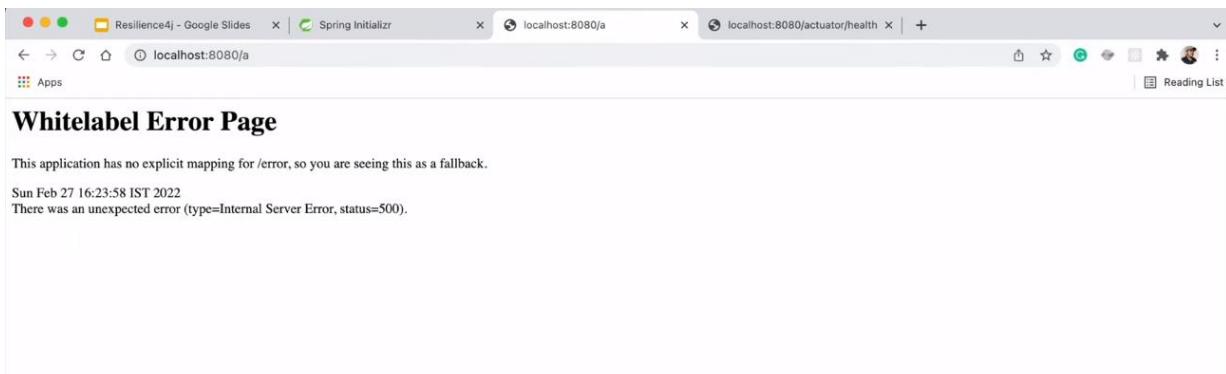
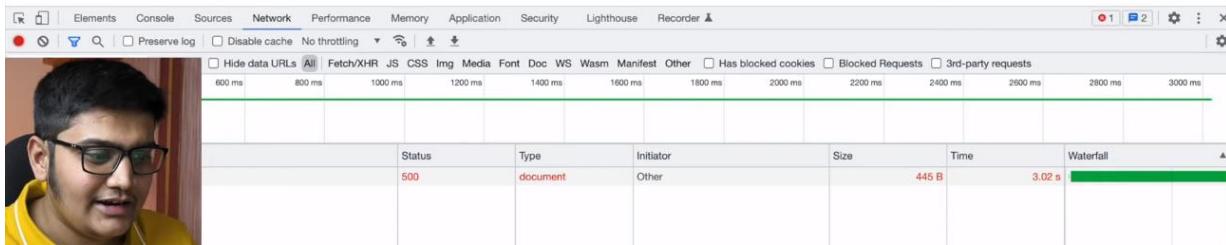
```
2022-02-27 16:22:20.631 INFO 5504 --- [main] o.a.c.c.C
2022-02-27 16:22:20.631 INFO 5504 --- [main] w.s.c.Ser
2022-02-27 16:22:21.216 INFO 5504 --- [main] o.s.b.a.e
2022-02-27 16:22:21.256 INFO 5504 --- [main] o.s.b.w.e
2022-02-27 16:22:21.267 INFO 5504 --- [main] c.d.Servi
2022-02-27 16:22:21.580 INFO 5504 --- [on(2)-127.0.0.1] o.a.c.c.C
2022-02-27 16:22:21.581 INFO 5504 --- [on(2)-127.0.0.1] o.s.web.s
2022-02-27 16:22:21.582 INFO 5504 --- [on(2)-127.0.0.1] o.s.web.s
```



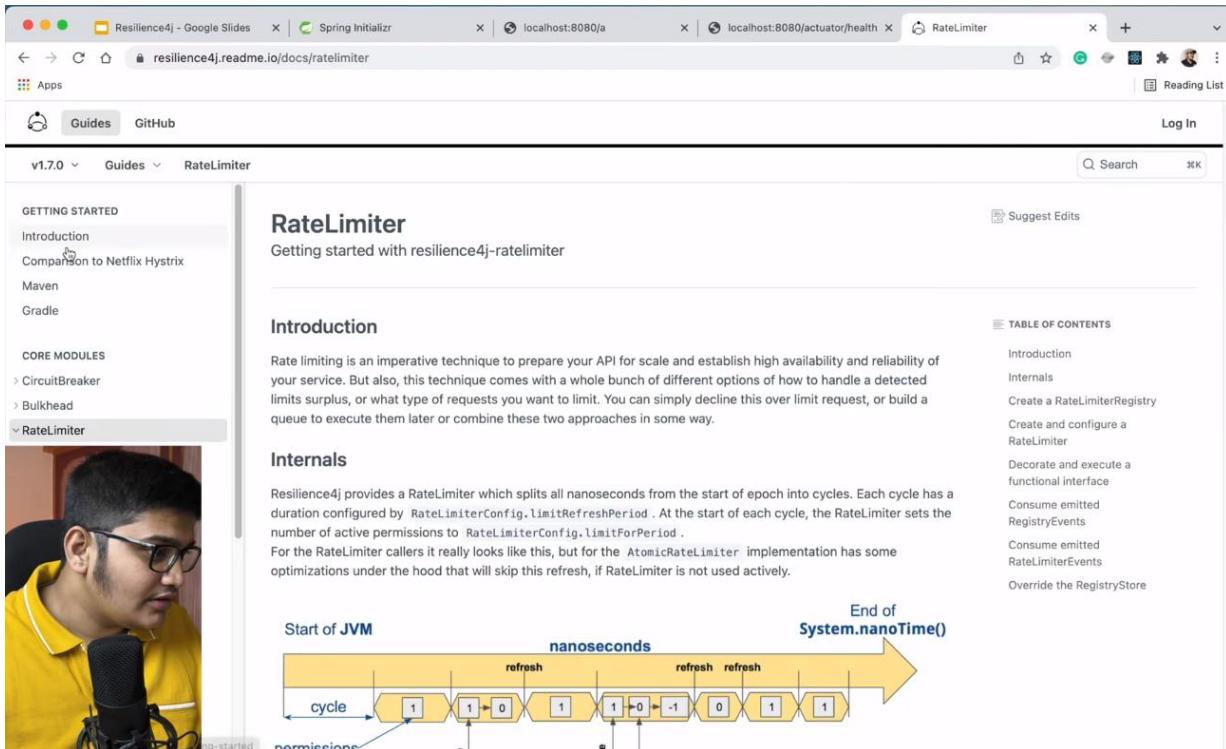
Only 10 requests are allowed within a 10s interval, a delay of 3s will occur if you try to exceed that limit



It is now waiting because we have sent 10 requests within the 10s interval

After 3s, we will then get an error because we have not defined/used a fallback method for the rate limiter configuration



<https://www.resilience4j.readme.io/docs/getting-started>

Introduction

Comparison to Netflix Hystrix  
Maven  
Gradle

CORE MODULES

> CircuitBreaker  
> Bulkhead  
> RateLimiter  
> Retry  
TimeLimiter  
Cache

Java

```
// Create a CircuitBreaker with default configuration
CircuitBreaker circuitBreaker = CircuitBreaker
    .ofDefaults("backendService");

// Create a Retry with default configuration
// 3 retry attempts and a fixed time interval between retries of 500ms
Retry retry = Retry
    .ofDefaults("backendService");

// Create a Bulkhead with default configuration
Bulkhead bulkhead = Bulkhead
    .ofDefaults("backendService");

Supplier<String> supplier = () -> backendService
    .doSomething(param1, param2)

// Decorate your call to backendService.doSomething()
// with a Bulkhead, CircuitBreaker and Retry
// **note: you will need the resilience4j-all dependency for this
Supplier<String> decoratedSupplier = Decorators.ofSupplier(supplier)
    .withCircuitBreaker(circuitBreaker)
    .withBulkhead(bulkhead)
    .withRetry(retry)
    .decorate();

// Execute the decorated supplier and recover from any exception
String result = Try.ofSupplier(decoratedSupplier)
    .recover(Throwable -> "Hello from Recovery").get();

// When you don't want to decorate your lambda expression,
// but just execute it and protect the call by a CircuitBreaker.
String result = circuitBreaker
    .executeSupplier(backendService::doSomething);

// You can also run the supplier asynchronously in a ThreadPoolBulkhead
ThreadPoolBulkhead threadPoolBulkhead = ThreadPoolBulkhead
```

TABLE OF CONTENTS

- Sneak preview
- Modularization
- All core modules and the Decorators class
- Core modules
- Add-on modules
- Frameworks modules
- Reactive modules
- Metrics modules
- 3rd party modules

All the possible modules we can implement are given on the left pane for spring-boot, reactor, etc