SPRING SUPPORT
FOR RESILIENCY USING RESILIENCE4J

CIRCUIT BREAKER PATTERN

Pom.xml

```
<dependency>
<groupId>io.github.resilience4j</groupId>
<artifactId>resilience4j-spring-boot2</artifactId>
</dependency>
<dependency>
<groupId>io.github.resilience4j</groupId>
<artifactId>resilience4j-circuitbreaker</artifactId>
</dependency>
<dependency>
<groupId>io.github.resilience4j</groupId>
<artifactId>resilience4j-timelimiter</artifactId>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-aop</artifactId>
</dependency>
```

Application.properties

```
resilience4j.circuitbreaker.configs.default.registerHealthIndicator= true
resilience4j.circuitbreaker.instances.detailsForCustomerSupportApp.minimumNumberOfCalls= 5
resilience4j.circuitbreaker.instances.detailsForCustomerSupportApp.failureRateThreshold= 50
resilience4j.circuitbreaker.instances.detailsForCustomerSupportApp.waitDurationInOpenState= 30000
resilience4j.circuitbreaker.instances.detailsForCustomerSupportApp.permittedNumberOfCallsInHalfOpenState=2
```

AccountsController.java

```
public class AccountsController{

@PostMapping("/myCustomerDetails")
@CircuitBreaker(name = "detailsforCustomerDetailsApp", fallbackMethod = "mycustomerDetailsFallback")
public CustomerDetails myCustomerDetails(@RequestBody Customer customer) {
Accounts accounts = accountRepository.findByCustomerId(customer.getCustomerId());
List<Loans> loans = loansFeignClient.getLoansDetails(customer);
List<Cards> cards = cardsFeignClient.getCardDetails(customer);

CustomerDetails customerDetails = new CustomerDetails();
customerDetails.setAccounts(accounts);
customerDetails.setLoans(loans);
customerDetails.setCards(cards);

return customerDetails;
}
}

private CustomerDetails mycustomerDetailsFallback(Customer customer, Throwable t) {
Accounts accounts = accountRepository.findByCustomerId(customer.getCustomerId());
List<Loans> loans = loansFeignClient.getLoansDetails(customer);
CustomerDetails cd = new CustomerDetails();
cd.setAccounts(accounts);
cd.setLoans(loans);
return cd;
}
```

http://localhost:8081/actuator/

```
{
  "_links": {
    "self": {
      "href": "http://localhost:8081/actuator",
      "templated": false
    },
    "bulkheads": {
      "href": "http://localhost:8081/actuator/bulkheads",
      "templated": false
    },
    "bulkheadevents-bulkheadName": {
      "href": "http://localhost:8081/actuator/bulkheadevents/{bulkheadName}",
      "templated": true
    },
    "bulkheadevents": {
      "href": "http://localhost:8081/actuator/bulkheadevents",
      "templated": false
    },
    "bulkheadevents-bulkheadName-eventType": {
      "href": "http://localhost:8081/actuator/bulkheadevents/{bulkheadName}/{eventType}",
      "templated": true
    },
    "circuitbreakers-name": {
      "href": "http://localhost:8081/actuator/circuitbreakers/{name}",
      "templated": true
    },
    "circuitbreakers": {
      "href": "http://localhost:8081/actuator/circuitbreakers",
      "templated": false
    },
    "circuitbreakerevents-name-eventType": {
      "href": "http://localhost:8081/actuator/circuitbreakerevents/{name}/{eventType}",
      "templated": true
    },
    "circuitbreakerevents": {
      "href": "http://localhost:8081/actuator/circuitbreakerevents",
      "templated": false
    },
    "circuitbreakerevents-name": {
      "href": "http://localhost:8081/actuator/circuitbreakerevents/{name}",
      "templated": true
    },
    "ratelimiters": {
      "href": "http://localhost:8081/actuator/ratelimiters",
      "templated": false
    },
    "ratelimiterevents": {
      "href": "http://localhost:8081/actuator/ratelimiterevents",
      "templated": false
    },
    "ratelimiterevents-name-eventType": {
      "href": "http://localhost:8081/actuator/ratelimiterevents/{name}/{eventType}",
      "templated": true
    },
    "ratelimiterevents-name": {
      "href": "http://localhost:8081/actuator/ratelimiterevents/{name}",
    "templated": true
  },
  "retries": {
    "href": "http://localhost:8081/actuator/retries",
    "templated": false
  },
  "retryevents": {
    "href": "http://localhost:8081/actuator/retryevents",
    "templated": false
  },
  "retryevents-name": {
    "href": "http://localhost:8081/actuator/retryevents/{name}",
    "templated": true
  },
  "retryevents-name-eventType": {
    "href": "http://localhost:8081/actuator/retryevents/{name}/{eventType}",
    "templated": true
  },
  "timelimiters": {
    "href": "http://localhost:8081/actuator/timelimiters",
    "templated": false
  },
  "timelimiterevents": {
    "href": "http://localhost:8081/actuator/timelimiterevents",
    "templated": false
  },
  "timelimiterevents-name": {
    "href": "http://localhost:8081/actuator/timelimiterevents/{name}",
    "templated": true
  },
  "timelimiterevents-name-eventType": {
    "href": "http://localhost:8081/actuator/timelimiterevents/{name}/{eventType}",
    "templated": true
  },
  "beans": {
    "href": "http://localhost:8081/actuator/beans",
    "templated": false
  },
  "caches-cache": {
    "href": "http://localhost:8081/actuator/caches/{cache}",
    "templated": true
  },
  "caches": {
    "href": "http://localhost:8081/actuator/caches",
    "templated": false
  },
  "health": {
    "href": "http://localhost:8081/actuator/health",
    "templated": false
  },
  "health-path": {
    "href": "http://localhost:8081/actuator/health/{*path}",
    "templated": true
  },
  "info": {
    "href": "http://localhost:8081/actuator/info",
    "templated": false
  },
"conditions": {
  "href": "http://localhost:8081/actuator/conditions",
  "templated": false
},
"shutdown": {
  "href": "http://localhost:8081/actuator/shutdown",
  "templated": false
},
"configprops": {
  "href": "http://localhost:8081/actuator/configprops",
  "templated": false
},
"configprops-prefix": {
  "href": "http://localhost:8081/actuator/configprops/{prefix}",
  "templated": true
},
"env": {
  "href": "http://localhost:8081/actuator/env",
  "templated": false
},
"env-toMatch": {
  "href": "http://localhost:8081/actuator/env/{toMatch}",
  "templated": true
},
"loggers": {
  "href": "http://localhost:8081/actuator/loggers",
  "templated": false
},
"loggers-name": {
  "href": "http://localhost:8081/actuator/loggers/{name}",
  "templated": true
},
"heapdump": {
  "href": "http://localhost:8081/actuator/heapdump",
  "templated": false
},
"threaddump": {
  "href": "http://localhost:8081/actuator/threaddump",
  "templated": false
},
"metrics-requiredMetricName": {
  "href": "http://localhost:8081/actuator/metrics/{requiredMetricName}",
  "templated": true
},
"metrics": {
  "href": "http://localhost:8081/actuator/metrics",
  "templated": false
},
"scheduledtasks": {
  "href": "http://localhost:8081/actuator/scheduledtasks",
  "templated": false
},
"mappings": {
  "href": "http://localhost:8081/actuator/mappings",
  "templated": false
},
"refresh": {
  "href": "http://localhost:8081/actuator/refresh",
            "templated": false
        },
        "features": {
            "href": "http://localhost:8081/actuator/features",
            "templated": false
        },
        "serviceregistry": {
            "href": "http://localhost:8081/actuator/serviceregistry",
            "templated": false
        }
    }
}
```

http://localhost:8081/myCustomerDetails

```
{
    "accounts": {
        "customerId": 1,
        "accountNumber": 100928281,
        "accountType": "saving",
        "branchAddress": "LVS",
        "createDt": "2021-05-16"
    },
    "loans": [
        {
            "loanNumber": 3,
            "customerId": 1,
            "startDt": "2021-02-14",
            "loanType": "Home",
            "totalLoan": 50000,
            "amountPaid": 10000,
            "outstandingAmount": 40000,
            "createDt": "2018-02-14"
        },
        {
            "loanNumber": 1,
            "customerId": 1,
            "startDt": "2020-10-13",
            "loanType": "Home",
            "totalLoan": 200000,
            "amountPaid": 50000,
            "outstandingAmount": 150000,
            "createDt": "2020-10-13"
        },
        {
            "loanNumber": 2,
            "customerId": 1,
            "startDt": "2020-06-06",
            "loanType": "Vehicle",
            "totalLoan": 40000,
            "amountPaid": 10000,
            "outstandingAmount": 30000,
            "createDt": "2020-06-06"
        },
        {
            "loanNumber": 4,
            "customerId": 1,
            "startDt": "2018-02-14",
        "loanType": "Personal",
        "totalLoan": 10000,
        "amountPaid": 3500,
        "outstandingAmount": 6500,
        "createDt": "2018-02-14"
      }
   ],
   "cards": null
}
```

RETRY PATTERN
FOR RESILIENCY IN MICROSERVICES

Application.properties

```
management.endpoint.health.show-details = always
resilience4j.retry.instances.retryForCustomerDetails.maxRetryAttempts=3
resilience4j.retry.instances.retryForCustomerDetails.waitDuration=2000
```

AccountsController.java

```
public class AccountsController{

@PostMapping("/myCustomerDetails")
//@CircuitBreaker(name = "detailsforCustomerDetailsApp", fallbackMethod = "mycustomerDetailsFallback")
@Retry(name = "retryforCustomerDetails",fallbackMethod = "mycustomerDetailsFallback")
public CustomerDetails myCustomerDetails(@RequestBody Customer customer) {
System.out.println("------------------");
Accounts accounts = accountRepository.findByCustomerId(customer.getCustomerId());
List<Loans> loans = loansFeignClient.getLoansDetails(customer);
List<Cards> cards = cardsFeignClient.getCardDetails(customer);

CustomerDetails customerDetails = new CustomerDetails();
customerDetails.setAccounts(accounts);
customerDetails.setLoans(loans);
customerDetails.setCards(cards);

return customerDetails;
}
}


private CustomerDetails mycustomerDetailsFallback(Customer customer, Throwable t) {
Accounts accounts = accountRepository.findByCustomerId(customer.getCustomerId());
List<Loans> loans = loansFeignClient.getLoansDetails(customer);
CustomerDetails cd = new CustomerDetails();
cd.setAccounts(accounts);
cd.setLoans(loans);
return cd;
}
```

As we mention above it will retry for three times.

OutPut

```
------------------
Hibernate: select accounts0_.account_number as account_1_0_, accounts0_.account_type as account_2_0_,
accounts0_.branch_address as branch_a3_0_, accounts0_.create_dt as create_d4_0_, accounts0_.customer_id as
customer5_0_ from account accounts0_ where accounts0_.customer_id=?
2022-07-10 16:38:03.396  WARN 9076 --- [nio-8081-exec-1] o.s.c.l.core.RoundRobinLoadBalancer      : No servers available for
service: cards
```

```
2022-07-10 16:38:03.400  WARN 9076 --- [nio-8081-exec-1] .s.c.o.l.FeignBlockingLoadBalancerClient : Load balancer does not
contain an instance for the service cards
------------------
Hibernate: select accounts0_.account_number as account_1_0_, accounts0_.account_type as account_2_0_,
accounts0_.branch_address as branch_a3_0_, accounts0_.create_dt as create_d4_0_, accounts0_.customer_id as
customer5_0_ from account accounts0_ where accounts0_.customer_id=?
2022-07-10 16:38:04.113  WARN 9076 --- [nio-8081-exec-1] o.s.c.l.core.RoundRobinLoadBalancer      : No servers available for
service: cards
2022-07-10 16:38:04.113  WARN 9076 --- [nio-8081-exec-1] .s.c.o.l.FeignBlockingLoadBalancerClient : Load balancer does not
contain an instance for the service cards
------------------
Hibernate: select accounts0_.account_number as account_1_0_, accounts0_.account_type as account_2_0_,
accounts0_.branch_address as branch_a3_0_, accounts0_.create_dt as create_d4_0_, accounts0_.customer_id as
customer5_0_ from account accounts0_ where accounts0_.customer_id=?
2022-07-10 16:38:04.784  WARN 9076 --- [nio-8081-exec-1] o.s.c.l.core.RoundRobinLoadBalancer      : No servers available for
service: cards
2022-07-10 16:38:04.785  WARN 9076 --- [nio-8081-exec-1] .s.c.o.l.FeignBlockingLoadBalancerClient : Load balancer does not
contain an instance for the service cards
```

http://localhost:8081/myCustomerDetails

```
{
  "accounts": {
    "customerId": 1,
    "accountNumber": 100928281,
    "accountType": "saving",
    "branchAddress": "LVS",
    "createDt": "2021-05-16"
  },
  "loans": [
    {
      "loanNumber": 3,
      "customerId": 1,
      "startDt": "2021-02-14",
      "loanType": "Home",
      "totalLoan": 50000,
      "amountPaid": 10000,
      "outstandingAmount": 40000,
      "createDt": "2018-02-14"
    },
    {
      "loanNumber": 1,
      "customerId": 1,
      "startDt": "2020-10-13",
      "loanType": "Home",
      "totalLoan": 200000,
      "amountPaid": 50000,
      "outstandingAmount": 150000,
      "createDt": "2020-10-13"
    },
    {
      "loanNumber": 2,
      "customerId": 1,
      "startDt": "2020-06-06",
      "loanType": "Vehicle",
      "totalLoan": 40000,
      "amountPaid": 10000,
      "outstandingAmount": 30000,
        "createDt": "2020-06-06"
      },
      {
        "loanNumber": 4,
        "customerId": 1,
        "startDt": "2018-02-14",
        "loanType": "Personal",
        "totalLoan": 10000,
        "amountPaid": 3500,
        "outstandingAmount": 6500,
        "createDt": "2018-02-14"
      }
    ],
    "cards": null
}
```

<u>RATE LIMITTER PATTERN</u>

The rate limiter pattern will help to stop overloading the service with more calls more than it can consume in a given time. This is
an imperative technique to prepare our API for high availability and reliability.

Application.properties

```
resilience4j.ratelimiter.configs.default.registerHealthIndicator= true
resilience4j.ratelimiter.instances.sayHello.timeoutDuration=5000
resilience4j.ratelimiter.instances.sayHello.limitRefreshPeriod=5000
resilience4j.ratelimiter.instances.sayHello.limitForPeriod=1
```

AccountsController.class

```
public class AccountsController{

        @GetMapping("/sayHello")
        @RateLimiter(name = "sayHello", fallbackMethod = "sayHelloFallback")
        public String sayHello() {
                return "Hello, Welcome";
        }

        private String sayHelloFallback(Throwable t) {
                return "Max time request";
        }
}
```

http://localhost:8081/sayHello

Hello, Welcome

http://localhost:8081/sayHello

Max time request

<u>BULKHEAD PATTERN</u>

A ship is split into small multiple compartments using Bulkheads. Bulkheads are used to seal parts of the ship to prevent entire ship fromsinking in case of flood.