**Similar to Circuit Breaker**

The annotations **@Retryable** and **@Recover** come from **Spring Retry**, a powerful library used to automatically retry operations that can fail transiently (like network calls, database locks, service timeouts, etc.).

**🔁 @Retryable - Automatically Retry a Method**

It is used to **mark a method that should be retried** on specific exceptions.

**✅ Syntax:**

@Retryable(

value = { SomeTransientException.class }, // Exception(s) to retry on

maxAttempts = 3, // How many times to retry

backoff = @Backoff(delay = 2000) // Delay between retries

)

public String someRemoteCall() {

// logic here

}

**🔍 Explanation:**

* value: Exceptions that trigger a retry.
* maxAttempts: Total attempts (including the first one).
* backoff: Delay (in ms) between retries. You can also configure multiplier, maxDelay, etc.

**🛑 @Recover - Handle Failures After Retries Exhausted**

If all retry attempts fail, the method annotated with @Recover is called **as a fallback**.

**✅ Syntax:**

@Recover

public String recover(SomeTransientException e) {

// fallback logic

return "Fallback result";

}

* It must **have the same return type** as the @Retryable method.
* The first parameter must match the exception you want to recover from.

**🔁🛠️ Full Working Example:**

import org.springframework.retry.annotation.Backoff;

import org.springframework.retry.annotation.Recover;

import org.springframework.retry.annotation.Retryable;

import org.springframework.stereotype.Service;

@Service

public class MyRetryService {

@Retryable(

value = { RuntimeException.class },

maxAttempts = 3,

backoff = @Backoff(delay = 1000)

)

public String callUnstableService() {

System.out.println("Trying operation...");

throw new RuntimeException("Temporary issue");

}

@Recover

public String recover(RuntimeException e) {

System.out.println("Recovering after failure: " + e.getMessage());

return "Default response after retries failed";

}

}

**🧩 Enabling Spring Retry**

Add the necessary dependencies:

<!-- Maven -->

<dependency>

<groupId>org.springframework.retry</groupId>

<artifactId>spring-retry</artifactId>

</dependency>

<dependency>

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

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

</dependency>

Then, enable retry support in your configuration:

@SpringBootApplication

@EnableRetry

public class MyApp { }

**📌 Summary**

| Annotation | Purpose |  
| ------------ | ------------------------------------- |  
| @Retryable | Retry a method when it fails |  
| @Recover | Fallback logic after all retries fail |

Would you like:

* A Spring Boot project with this implemented?
* Examples with exponential backoff and custom retry policies?