Spring Retry

개요

Order EDA Gateway에서는 AppOrder의 콜백 API를 호출할 때, 특정 에러가 발생하는 경우 재시도를 하고 있습니다. 재시도 구현을 위해 Spring Retry를 사용했습니다. 이번 발표에서는 Spring Retry에 대해 간단히 설명하고, Order EDA Gateway에서 Spring Retry를 어떻게 사용하고 있는지 소개하도록 하겠습니다.

RetryTemplate

실패한 작업을 재시도하는 것이 실패 가능성을 줄이는데 큰 도움을 줄 수 있습니다. 특히 일시적인 오류인 경우에는 재시도를 통해서 문제를 쉽게 해결할 수도 있습니다. 예를 들어 다른 웹 서버로 요청을 보내는데 네트워크로 인해 오류가 발생하는 경우는 잠시후에 해결되는 경우가 많습니다.

Spring Retry에는 작업에 대한 재시도를 자동화하기 위한 전략 인터페이스(RetryOperations)가 있습니다. 그리고 해당 인터페이스의 구현체인 RetryTemplate을 제공합니다.

RetryTemplate를 이용하면 특정 작업에 대한 재시도를 아주 간단하고 유연하게 개발할 수 있습니다.

특정 횟수만큼 재시도

```
public void 특정횟수만큼_재시도() throws Exception {
    int maxAttempts = 3;
    RetryTemplate template = new RetryTemplate();
    SimpleRetryPolicy policy = new SimpleRetryPolicy(maxAttempts);
    template.setRetryPolicy(policy);

    template.execute(context -> {
        // Do stuff that might fail
        return result;
    });
}
```

RetryTemplate#execute 메소드를 호출할 때 재시도 해야하는 비즈니스 로직을 콜백에 넣어주면 됩니다. 콜백 메소드를 실행하는 도중에 예외가 발생하면, 재시도 정책에 따라 특정 횟수 혹은 특정 시간동안 재시도를 합니다. 콜백 메소드의 파라미터는 RetryContext 입니다. RetryContext는 대부분의 경우 사용되지 않습니다. RetryContext를 통해서 현재 시도가 몇번째 시도인지, 그리고 이전 시도에서 발생한 예외가 무엇인지등을 가져올 수 있습니다. 때로는 RetryContext를 재시도간에 데이터를 전달하는 용도로 사용하기도 합니다.

Recovery Callback

재시도가 전부 실패한 경우에 특정 콜백이 실행되도록 할 수 있습니다. 이 기능을 사용하려면 execute 메소드의 두번째 인자로 RecoveryCallback 객체를 전달해줘야 합니다.

RetryTemplate은 모든 재시도가 실패하고 더이상 재시도할 수 없는 경우, RecoveryCallback 메소드를 호출합니다. RecoveryCallback의 recover 메소드에서는 재시도가 전부 실패한 경우에 대한 대체 로직을 수행합니다.

```
int maxAttempts = 3;
retryTemplate template = new RetryTemplate();
SimpleRetryPolicy policy = new SimpleRetryPolicy(maxAttempts);
remplate.setRetryPolicy(policy);

remplate.execute(context -> {
   throw new IllegalStateException();
, context -> {
   log.error("{}번의 재시도가 전부 실패했습니다.", context.getRetryCount());
   return null;
);
```

- Recovery Callback을 전달한 경우 : 모든 재시도가 실패 시, RestTemplate#execute 메소드에서 리턴 되는 값은 RecoveryCallback에서 리턴하는 값입니다. 위의 예제의 경우 콜백 메소드를 3번 재시도 되고, RestTemplate#execute 메소드는 null을 리턴하게 됩니다.
- Recovery Callback을 전달하지 않은 경우 : 모든 재시도가 실패 시, RestTemplate#execute 메소드는 마지막 재시도시 발생한 예외를 발생시킵니다.

Stateless Retry

가장 간단한 재시도는 단순한 while 루프입니다. RetryTemplate는 성공하거나 혹은 실패할 때까지 계속해서 재시도할 수 있습니다. stateless 재시도에서는 콜백이 항상 동일한 스레드에서 실행됩니다.

> Stateless Retry 말고도 Stateful Retry가 있습니다. Order EDA Gateway에서 사용하고 있는 Retry는 Stateless Retry이기 때문에 따로 설명하지 않겠습니다.

Retry Policies

RetryTemplate은 콜백 메소드에서 예외가 발생한 경우에 재시도을 할지 말지에 대한 결정을 RetryPolicy에 위임합니다. 따라서 어떤 RetryPolicy를 사용하느냐에 따라 재시도 여부가 결정됩니다.

Spring Retry는 SimpleRetryPolicy 및 TimeoutRetryPolicy와 같은 간단한 stateless RetryPolicy 구현체를 제공합니다. SimpleRetryPolicy는 특정 예외가 발생한 경우에만 재시도를 하며, 재시도 최대 횟수를 제한합니다.

> 참고로 Order EDA Gateway에서는 SimpleRetryPolicy를 사용하고 있습니다.

모든 예외를 재시도하는 것은 비효율적입니다. 따라서 모든 예외에 대해서 재시도하는 것보다는, 재시도 했을때 성공할 수 있는 예외에 대해서만 재시도하는 것이 좋습니다.

Backoff Policies

오류가 발생하여 재시도를 할 때 재시도를 하기전에 잠깐 기다리는 것이 유용 할 때가 많습니다. 일반적으로 오류는 잠깐 동안 기다리기만 해도 해결되는 경우가 많습니다. RetryCallback이 실패하면 RetryTemplate은 BackoffPolicy에 따라 실행을 일시적으로 중지할 수 있습니다.

Spring Retry는 유용한 BackoffPolicy 구현체를 제공하고 있습니다. backoff 시간을 기하급수적으로 증가시키고 싶은 경우 ExponentialBackoffPolicy를 사용하면 됩니다. 고정된 시간으로 backoff 시키고자 한다면 FixedBackOffPolicy를 사용하면 됩니다.

> 참고로 Order EDA Gateway에서는 FixedBackOffPolicy를 사용하고 있습니다.

Listener

Spring Retry는 RetryListener 인터페이스를 제공하고 있습니다. 이 인터페이스의 구현체를 RestTemplate에 등록할 수 있습니다.

```
public interface RetryListener {

    void open(RetryContext context, RetryCallback<T> callback);

    void onError(RetryContext context, RetryCallback<T> callback, Throwab
le e);

    void close(RetryContext context, RetryCallback<T> callback, Throwable
 e);
}
```

- open(RetryContext context, RetryCallback<\T> callback) : 첫번째 시도 전에 호출이 됩니다. 메소드의 인자로 RetryContext 객체와 실행될 콜백 메소드가 전달됩니다. 만약에 등록된 RetryListener 중에서 open 메소드의 결과로 false을 리턴한 리스너가 하나라도 있다면, TerminatedRetryException이 발생 합니다. 그리고 예외가 발생한 경우에는 한번도 시도 하지 않습니다.
- void close(RetryContext context, RetryCallback<\T> callback, Throwable e) : 성공 실패 여부와 상관없이 재시도가 모두 끝나면 close 메소드가 호출 됩니다. 만약에 모든 재시도가 실패한 경우, 마지막에 발생한 예외가 두번째 인자로 전달 됩니다.
- void onError(RetryContext context, RetryCallback<\T> callback, Throwable e) : 재시도 중 에러가 발생할 때마다 onError가 호출됩니다.

> 참고로 RetryListener는 하나 이상 등록될 수 있으며, RetryListner는 리스트에 저장됩니다.

Order EDA Gateway

Order EDA Gateway에서 어떻게 Retry하고 있는지는 Wiki를 참고하시길 바랍니다.