

# 重试机制

---

## 重试设计

重试条件

重试时间

停止重试

重试后做什么

## 开发实现

### 支持配置和扩展重试策略

1. 定义扩展策略常量
2. 使用工厂模式+spi机制
3. 在rpcconfig全局配置中新增配置

## 应用重试功能

# 重试设计

## 重试条件

什么情况开启重试机制。由于网络原因导致异常情况发生，开启重试

## 重试时间

1. 固定重试间隔，比如隔1秒重试1次
2. 指数退避重试，比如第一次是1秒，第二次是2秒，第三次是4秒，第四次是8秒
3. 随机间隔重试，随机隔多少秒重试
4. 可变延迟重试，根据上一次的响应调整重试时间

## 停止重试

1. 重试几次后停止
2. 重试多长时间后停止

## 重试后做什么

1. 通知告警
2. 降级容错

## 开发实现

1. 编写重试策略接口

```
Java |  
  
1  package com.yupi.yurpc.fault.retry;  
2  
3  import com.yupi.yurpc.model.RpcResponse;  
4  
5  import java.util.concurrent.Callable;  
6  
7  /**  
8   * 重试策略  
9   *  
10  * @author <a href="https://github.com/liyupi">程序员鱼皮</a>  
11  * @learn <a href="https://codefather.cn">鱼皮的编程宝典</a>  
12  * @from <a href="https://yupi.icu">编程导航学习圈</a>  
13  */  
14 public interface RetryStrategy {  
15  
16     /**  
17      * 重试  
18      *  
19      * @param callable  
20      * @return  
21      * @throws Exception  
22      */  
23     RpcResponse doRetry(Callable<RpcResponse> callable) throws Exception;  
24 }  
25
```

2. 重试策略实现

```
1 <!-- https://github.com/rholder/guava-retrying -->
2 <dependency>
3     <groupId>com.github.rholder</groupId>
4     <artifactId>guava-retrying</artifactId>
5     <version>2.0.0</version>
6 </dependency>
```

## 不重试 – 重试策略

```
1 package com.yupi.yurpc.fault.retry;
2
3 import com.yupi.yurpc.model.RpcResponse;
4 import lombok.extern.slf4j.Slf4j;
5
6 import java.util.concurrent.Callable;
7
8 /**
9  * 不重试 – 重试策略
10  *
11  * @author <a href="https://github.com/liyupi">程序员鱼皮</a>
12  * @learn <a href="https://codefather.cn">鱼皮的编程宝典</a>
13  * @from <a href="https://yupi.icu">编程导航学习圈</a>
14  */
15 @Slf4j
16 public class NoRetryStrategy implements RetryStrategy {
17
18     /**
19      * 重试
20      *
21      * @param callable
22      * @return
23      * @throws Exception
24      */
25     public RpcResponse doRetry(Callable<RpcResponse> callable) throws Exception {
26         return callable.call();
27     }
28
29 }
30
```

```
1  package com.yupi.yurpc.fault.retry;
2
3  import com.github.rholder.retry.*;
4  import com.yupi.yurpc.model.RpcResponse;
5  import lombok.extern.slf4j.Slf4j;
6
7  import java.util.concurrent.Callable;
8  import java.util.concurrent.ExecutionException;
9  import java.util.concurrent.TimeUnit;
10
11  /**
12   * 固定时间间隔 - 重试策略
13   *
14   * @author <a href="https://github.com/liyupi">程序员鱼皮</a>
15   * @learn <a href="https://codefather.cn">鱼皮的编程宝典</a>
16   * @from <a href="https://yupi.icu">编程导航学习圈</a>
17   */
18  @Slf4j
19  public class FixedIntervalRetryStrategy implements RetryStrategy {
20
21      /**
22       * 重试
23       *
24       * @param callable
25       * @return
26       * @throws ExecutionException
27       * @throws RetryException
28       */
29      public RpcResponse doRetry(Callable<RpcResponse> callable) throws ExecutionException, RetryException {
30          Retryer<RpcResponse> retryer = RetryerBuilder.<RpcResponse>newBuilder()
31              .retryIfExceptionOfType(Exception.class)
32              .withWaitStrategy(WaitStrategies.fixedWait(3L, TimeUnit.SECONDS))
33              .withStopStrategy(StopStrategies.stopAfterAttempt(3))
34              .withRetryListener(new RetryListener() {
35                  @Override
36                  public <V> void onRetry(Attempt<V> attempt) {
37                      log.info("重试次数 {}", attempt.getAttemptNumber());
38                  }
39              })
40              .build();
41          return retryer.call(callable);
42      }
43  }
```

```
42     }
43
44 }
45
```

上述代码中，重试策略如下：

- 重试条件：使用 `retryIfExceptionOfType` 方法指定当出现 `Exception` 异常时重试。
- 重试等待策略：使用 `withWaitStrategy` 方法指定策略，选择 `fixedWait` 固定时间间隔策略。
- 重试停止策略：使用 `withStopStrategy` 方法指定策略，选择 `stopAfterAttempt` 超过最大重试次数停止。
- 重试工作：使用 `withRetryListener` 监听重试，每次重试时，除了再次执行任务外，还能够打印当前的重试次数。

## 支持配置和扩展重试策略

### 1. 定义扩展策略常量

```
1  package com.yupi.yurpc.fault.retry;
2
3  /**
4   * 重试策略键名常量
5   *
6   * @author <a href="https://github.com/liyupi">程序员鱼皮</a>
7   * @learn <a href="https://codefather.cn">鱼皮的编程宝典</a>
8   * @from <a href="https://yupi.icu">编程导航学习圈</a>
9   */
10 public interface RetryStrategyKeys {
11
12     /**
13      * 不重试
14      */
15     String NO = "no";
16
17     /**
18      * 固定时间间隔
19      */
20     String FIXED_INTERVAL = "fixedInterval";
21
22 }
23
```

## 2. 使用工厂模式+spi机制

```
Java |  
1 package com.yupi.yurpc.fault.retry;  
2  
3 import com.yupi.yurpc.spi.SpiLoader;  
4  
5 /**  
6  * 重试策略工厂（用于获取重试器对象）  
7  *  
8  * @author <a href="https://github.com/liyupi">程序员鱼皮</a>  
9  * @learn <a href="https://codefather.cn">编程宝典</a>  
10 * @from <a href="https://yupi.icu">编程导航知识星球</a>  
11 */  
12 public class RetryStrategyFactory {  
13  
14     static {  
15         SpiLoader.load(RetryStrategy.class);  
16     }  
17  
18     /**  
19      * 默认重试器  
20      */  
21     private static final RetryStrategy DEFAULT_RETRY_STRATEGY = new NoRetr  
yStrategy();  
22  
23     /**  
24      * 获取实例  
25      *  
26      * @param key  
27      * @return  
28      */  
29     public static RetryStrategy getInstance(String key) {  
30         return SpiLoader.getInstance(RetryStrategy.class, key);  
31     }  
32  
33 }  
34
```

## 3. 在rpcconfig全局配置中新增配置

```
1  @Data
2  public class RpcConfig {
3      /**
4       * 重试策略
5       */
6      private String retryStrategy = RetryStrategyKeys.N0;
7  }
8
```

## 应用重试功能

```

1  // 使用重试机制
2  RetryStrategy retryStrategy = RetryStrategyFactory.getInstance(rpcConfig.getRetryStrategy());
3  RpcResponse rpcResponse = retryStrategy.doRetry(() ->
4      VertxTcpClient.doRequest(rpcRequest, selectedServiceMetaInfo)
5  );
6  /**
7   * 服务代理 (JDK 动态代理)
8   *
9   * @author <a href="https://github.com/liyupi">程序员鱼皮</a>
10  * @learn <a href="https://codefather.cn">编程宝典</a>
11  * @from <a href="https://yupi.icu">编程导航知识星球</a>
12  */
13  public class ServiceProxy implements InvocationHandler {
14
15      /**
16       * 调用代理
17       *
18       * @return
19       * @throws Throwable
20       */
21      @Override
22      public Object invoke(Object proxy, Method method, Object[] args) throws Throwable {
23          // 指定序列化器
24          final Serializer serializer = SerializerFactory.getInstance(RpcApplication.getRpcConfig().getSerializer());
25
26          // 构造请求
27          String serviceName = method.getDeclaringClass().getName();
28          RpcRequest rpcRequest = RpcRequest.builder()
29              .serviceName(serviceName)
30              .methodName(method.getName())
31              .parameterTypes(method.getParameterTypes())
32              .args(args)
33              .build();
34          try {
35              // 从注册中心获取服务提供者请求地址
36              RpcConfig rpcConfig = RpcApplication.getRpcConfig();
37              Registry registry = RegistryFactory.getInstance(rpcConfig.getRegistryConfig().getRegistry());
38              ServiceMetaInfo serviceMetaInfo = new ServiceMetaInfo();
39              serviceMetaInfo.setServiceName(serviceName);
40              serviceMetaInfo.setServiceVersion(RpcConstant.DEFAULT_SERVICE_VERSION);

```



```

41         List<ServiceMetaInfo> serviceMetaInfoList = registry.serviceDiscovery(
42             serviceMetaInfo.getServiceKey());
43         if (CollUtil.isEmpty(serviceMetaInfoList)) {
44             throw new RuntimeException("暂无服务地址");
45         }
46
47         // 负载均衡
48         LoadBalancer loadBalancer = LoadBalancerFactory.getInstance(
49             rpcConfig.getLoadBalancer());
50         // 将调用方法名（请求路径）作为负载均衡参数
51         Map<String, Object> requestParams = new HashMap<>();
52         requestParams.put("methodName", rpcRequest.getMethodName());
53         ServiceMetaInfo selectedServiceMetaInfo = loadBalancer.select(
54             requestParams, serviceMetaInfoList);
55
56         // rpc 请求
57         // 使用重试机制
58         RetryStrategy retryStrategy = RetryStrategyFactory.getInstance(
59             rpcConfig.getRetryStrategy());
60         RpcResponse rpcResponse = retryStrategy.doRetry(() ->
61             VertxTcpClient.doRequest(rpcRequest, selectedServiceMetaInfo));
62         return rpcResponse.getData();
63     } catch (Exception e) {
64         throw new RuntimeException("调用失败");
65     }

```