webclient 호출 및 reactive 응답처리

- 이벤트 처리
    - doOnError() : 예외가 발생했을 경우, 특정 행위를 실행시킬 경우 사용
        - 에러 시, 처리 후 리턴되는지 다음건으로 이동되는지 확인 필요
        - 2xx 이 아닌 경우와 Exception 인 경우 call 됨
    - doOnSuccess() : 발송 성공 시, 후처리
    - onErrorReturn : 예외가 발생했을 때 특정 값을 Return 함
        - 정상과 같은 객체를 생성해서 특정값을 저장하고 중지 후 리턴
    - onErrorResume : 예외가 발생했을 때 다른 Flux형태로 Return 함
        - 정상과 같은 객체를 생성해서 특정값을 저장하고 다음 건 처리
    - onErrorContinue : 예외가 발생했을 때 멈추지 않고 해당 영역만 skip해서 동작함.
        - 에러처리 후 해당 건 스킵 및 다음 건 처리

```
public Mono<Object> doExec(
                ServerHttpRequest request, ServerHttpResponse
clientResponse, RbcAuth rbcAuth) {

        String finalAuthToken = ...;

        try{
                return webClient.get()
                                .uri(rbcUrl.getCategories())
.accept(MediaType.APPLICATION_JSON)
                                   .headers(httpHeaders ->
httpHeaders.setBearerAuth(finalAuthToken))
                                .retrieve()
                                .onStatus(HttpStatus::isError,
res -> doError(res))
.bodyToMono(CustomResult.class)
                                              .doOnError(ex ->
doExcept(clientResponse, ex))
                                               .flatMap(res ->
doSucc(clientResponse, res))
                                           .doOnSuccess(res ->
doFinish(clientResponse, res))
                                .subscribe()
                                .log()
```

```
                                ;
        }catch (Exception e){
                        log.error("webClient fail. e={}",
e.getMessage(), e);
                return doExcept(clientResponse, e);
        }
}

private Mono<Object> doFinish(ServerHttpResponse response,
CustomResult res) {
        ...
        return Mono.just("finish");
}

private Mono<Object> doSucc(ServerHttpResponse response,
CustomResult res) {
        ...
        return Mono.just("success");
}

private Mono<Object> doExcept(ServerHttpResponse response,
Throwable ex) {
        ...
        return Mono.just("fail. e={}", e.getmessage());
}

private Mono<Exception> doError(ClientResponse res) {
        if (res.statusCode().isError()) {
                                return  Mono.error(new
RuntimeException(res.toString()));
        }
        else{
                                return  Mono.error(new
RuntimeException(res.toString()));
        }
}
```