Synpulse8 Presentation

GitHub repo: https://github.com/Chingpo-Lin/ebank

— By Chingpo (Bob) Lin

Project Overview

1. Basic tools setup
2. User Module
3. Transaction Module
4. Swagger/Docker document

Basic Tools

1. Return Format

Custom return data

```java
@Data
@AllArgsConstructor
@NoArgsConstructor
public class JsonData {
    /**
     * status 0 means success, 1 means pending, -1 means fail */
    private Integer code;
    /**
     * data
     */
    private Object data;
    /**
     * description
     */
    private String msg;
```

2. Error Handle

Exception Handler

```java
@ControllerAdvice
@Slf4j
public class CustomExceptionHandler {

    @ExceptionHandler(value = Exception.class)
    @ResponseBody
    public JsonData handler(Exception e) {
        if (e instanceof BizException) {
            BizException bizException = (BizException) e;
            log.error("[business exception]", e);
            return JsonData.buildCodeAndMsg(bizException.getCode(), bizException.getMsg());
        } else {
            log.error("[unknown exception]", e);
            return JsonData.buildError("global unknown err");
        }
    }
}
```

Business Error

```
/**
 * account
 * */
ACCOUNT_EXISTS( code: 250001, msg: "account has already exist"),
ACCOUNT_PWD_ERROR( code: 250002, msg: "name or password error"),
ACCOUNT_NOT_LOGIN( code: 250003, msg: "account not login"),
ACCOUNT_REGISTER_FAIL( code: 250004, msg: "register fail"),
ACCOUNT_CURRENCY_EXIST( code: 250005, msg: "currency account exist"),
ACCOUNT_CURRENCY_SENDER_NOT_EXIST( code: 250006, msg: "please create an account before sending money"),
ACCOUNT_CURRENCY_RECEIVER_NOT_EXIST( code: 250007, msg: "please ask receiver to create an account"),
ACCOUNT_TRANSFER_FAIL( code: 250008,  msg: "transfer fail"),
ACCOUNT_NOT_EXIST( code: 250009,  msg: "user not found"),
```

3. Database setup

User Table

Field	Type	Length	Unsign...	Zer...
id	BIGINT	20	☐	☐
name	VARCHAR	128	☐	☐
pwd	VARCHAR	124	☐	☐
sex	TINYINT	2	☐	☐
create_ti...	DATETIME		☐	☐
mail	VARCHAR	64	☐	☐
secret	VARCHAR	32	☐	☐

User Account Table

Field	Type	Length	Unsign...	Zer...
id	BIGINT	20	☐	☐
user_id	BIGINT	20	☐	☐
balance	DECIMAL	10,2	☐	☐
currency	VARCHAR	32	☐	☐

Transaction Table

Field	Type	Length	Unsign...	Zer...	Bin...	Allow N...	K...
id	BIGINT	20	☐	☐	☐	☐	PR
currency	VARCHAR	20	☐	☐	☐	☑	
amount	DECIMAL	10,2	☐	☐	☐	☑	
account_...	VARCHAR	20	☐	☐	☐	☑	
value_date	DATETIME		☐	☐	☐	☑	
description	VARCHAR	128	☐	☐	☐	☑	
from	BIGINT	20	☐	☐	☐	☑	
to	BIGINT	20	☐	☐	☐	☑	
status	INT	11	☐	☐	☐	☑	

4. Lombok Library

1. @Data
2. log.info() / log.error()

```
<!--https://mvnrepository.com/artifact/org.projectlombok/lombok/1.18.16-->
<!--scope=provided, means only work in compile, do not need in build
, Lombok come with lombok annotation, Java can successfully compile to class file-->
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <version>${lombok.version}</version>
    <!--                <scope>provided</scope>-->
</dependency>
```

5. Encapsulation Rule

1.	XXDO	for database table object
2.	XXVO	for view object return to client
3.	XXRequest	for user request object
4.	Common model	object that use globally

6. Mybatis Plus Generator

```
// set service name
.setServiceName("%sService")
// set entity name
.setEntityName("%sDO")
```

User Service

1. Register

Password Encode

```
<!--https://mvnrepository.com/artifact/commons-codec/commons-codec -->
<!-- use for encryption -->
<dependency>
    <groupId>commons-codec</groupId>
    <artifactId>codec</artifactId>
    <version>${commons.codec.version}</version>
</dependency>
```

```
userDO.setSecret("$1$" + CommonUtil.getStringNumRandom( len: 8));

String cryptPwd = Md5Crypt.md5Crypt(registerRequest.getPwd().getBytes(), userDO.getSecret());
userDO.setPwd(cryptPwd);
```

2. Login

Login Interceptor

```java
@Slf4j
public class LoginInterceptor implements HandlerInterceptor {

    public static ThreadLocal<LoginUser> threadLocal = new ThreadLocal<>();
```

User APIs to intercept

```java
@Configuration
@Slf4j
public class InterceptorConfig implements WebMvcConfigurer {

    LoginInterceptor loginInterceptor() { return new LoginInterceptor(); }

    @Override
    public void addInterceptors(InterceptorRegistry registry) {
        registry.addInterceptor(loginInterceptor())
                .addPathPatterns("/s8/user_account/*/**")
                .excludePathPatterns("/s8/user_account/*/transfer");
    }
```

Json Web Token Generate

```java
public static String geneJsonWebToken(LoginUser loginUser) {
    if (loginUser == null) {
        throw new NullPointerException("loginUser object is null");
    }

    String token = Jwts.builder().setSubject(SUBJECT)
            .claim( s: "id", loginUser.getId())
            .claim( s: "name", loginUser.getName())
            .claim( s: "mail", loginUser.getMail())
            .setIssuedAt(new Date())
            .setExpiration(new Date(System.currentTimeMillis() + EXPIRE))
            .signWith(SignatureAlgorithm.HS256, SECRET)
            .compact();
    token = TOKEN_PREFIX + token;

    log.info("decode is:{}", checkJWT(token).toString());

    return token;
}
```

Json Web Token Decode

```
public static Claims checkJWT(String token) {
    try {
        log.info("token decode:{}", token);
        final Claims claims = Jwts.parser().setSigningKey(SECRET)
                .parseClaimsJws(token.replace(TOKEN_PREFIX, replacement: ""))
                .getBody();
        return claims;
    } catch (Exception e) {
        log.error("jwt token decode fail");
        return null;
    }
}
```

3. User Account

1. Create account for specific bank
2. Transfer between account (feign call by transaction server)

Transaction Service

1. Transfer

Feign Service

1. Add EnableFeignClient in starter
2. Add feign service interface (need same path)
3. Need load balancer (Ribbon with/without Eureka) (Nacos here)

Add Dependency

```xml
<!-- feign remote call -->
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-openfeign</artifactId>
</dependency>

<!-- nacos client -->
<dependency>
    <groupId>com.alibaba.cloud</groupId>
    <artifactId>spring-cloud-starter-alibaba-nacos-discovery</artifactId>
</dependency>
```

Starter File

```java
@SpringBootApplication
@MapperScan("org.example.mapper")
@EnableTransactionManagement
@EnableFeignClients
@EnableDiscoveryClient
public class TransactionApplication {
    public static void main(String[] args) { SpringApplication.run(TransactionApplication.class, args); }
}
```

Feign Service

```java
@FeignClient(name = "ebank-user-service")
public interface UserFeignService {

    /**
     * transfer process
     * @param transferInfo
     * @return
     */
    @PostMapping("/s8/user_account/v1/transfer")
    JsonData transfer(@RequestBody TransferInfo transferInfo);

    @GetMapping("s8/user/v1/get_name/{user_id}")
    JsonData getUserName(@PathVariable("user_id") long userId);
}
```

Transactional

```
@Transactional(rollbackFor = Exception.class, propagation = Propagation.REQUIRED)
public JsonData transfer(TransferInfo transferInfo) {
```

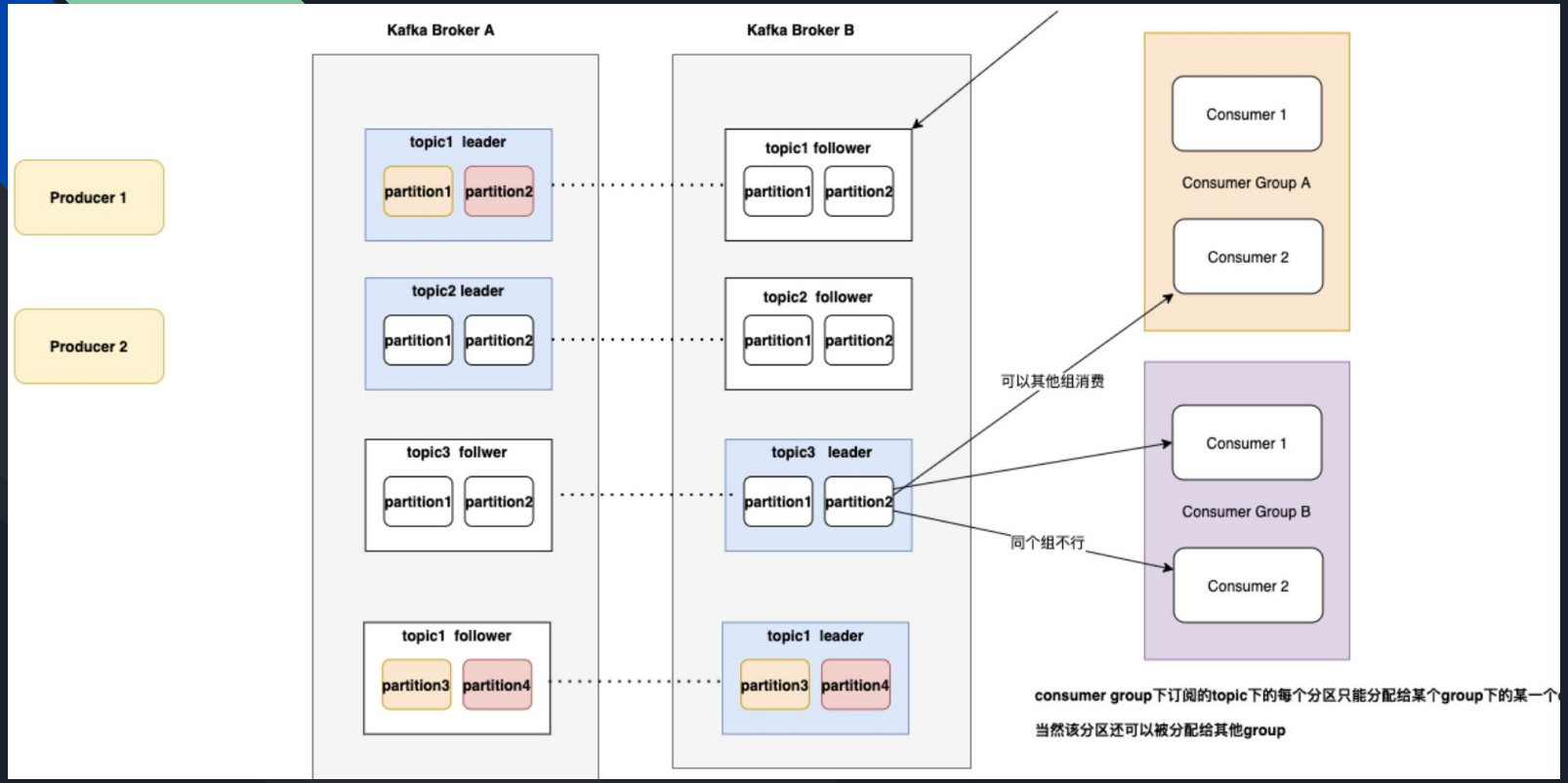
2. Page Transactions

Mybatis Plus Pagination

```java
@Bean
public MybatisPlusInterceptor mybatisPlusInterceptor() {
    MybatisPlusInterceptor mybatisPlusInterceptor = new MybatisPlusInterceptor();
    mybatisPlusInterceptor.addInnerInterceptor(new PaginationInnerInterceptor());

    return mybatisPlusInterceptor;
}
```

Kafka – – asynchronous & clipping



Send Page with Kafka

```
private void sendTransferKafkaMsg(TransferInfo transferInfo, String topic) {
    kafkaTemplate.send(topic, JSON.toJSON(transferInfo).toString()).addCallback(success -> {
        String topicName = success.getRecordMetadata().topic();
        int partition = success.getRecordMetadata().partition();
        long offset = success.getRecordMetadata().offset();
        log.info("succussfully send by kafka with topic:{}, " +
                "partition:{}, offset:{}", topicName, partition, offset);
    }, failure -> {
        log.info("fail to send to kafka:{}", failure.getMessage() );
    });
}
```

Consume Msg

```java
@KafkaListener(topics = {"ebank-transfer-topic"},groupId = "ebank-group2")
public void onMessage2(ConsumerRecord<?, ?> record, Acknowledgment ack,
                       @Header(KafkaHeaders.RECEIVED_TOPIC) String topic){

    log.info("listen to kafka with topic:{}, in partition:{}, with value:{}",
            topic, record.partition(), record.value());
    JSONObject jsonObject = JSON.parseObject(record.value().toString());

    try {
        boolean flag = transactionService.transferConsume(jsonObject);
        if (flag) {
            ack.acknowledge(); // this will submit msg
        } else {
            throw new BizException(BizCodeEnum.ACCOUNT_TRANSFER_FAIL);
        }
    } catch (Exception e) {
        log.error("fail", jsonObject);
    }
}
```

Document

1. Swagger

Swagger Config

```
@Bean
public Docket webApiDoc() {
    return new Docket(DocumentationType.OAS_30)
            .groupName("client side doc") Docket
            .pathMapping("/")
            // if open swagger, false is close, can change by var, online close
            .enable(true)
            // config documentation info
            .apiInfo(apiInfo())
            .select() ApiSelectorBuilder
            .apis(RequestHandlerSelectors.basePackage("org.example"))
            // select all path under api
            .paths(PathSelectors.ant( antPattern: "/s8/**"))
            .build() Docket
            // swagger 3.0 config
            .globalRequestParameters(globalRequestParameters())
            .globalResponses(HttpMethod.GET, getGlobalResponseMsg())
            .globalResponses(HttpMethod.POST, getGlobalResponseMsg());
```

2. Postman

Postman Link

https://github.com/Chingpo-Lin/ebank/blob/master/ebank.postman_collection.json

3. Docker Image

Dockerfile

```
#FROM  adoptopenjdk/openjdk11:ubi
# jre take less space
FROM  adoptopenjdk/openjdk11:jre11u-nightly
VOLUME /tmp
ARG JAR_FILE
COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java","-jar","/app.jar"]
```

```
REPOSITORY                    TAG                 IMAGE ID            CREATED             SIZE
xd-cloud/ebank-user           latest              a42fc81e45cb        9 days ago          318 MB
```

4. Kubernetes File

Kubernetes deploy

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ebank-transaction-deployment
  labels:
    app: ebank-transaction
spec:
  replicas: 1
  selector:
    matchLabels:
      app: ebank-transaction
  template:
    metadata:
      labels:
        app: ebank-transaction
    spec:
      containers:
      - name: ebank-transaction
        image: ebank-transaction
        ports:
        - containerPort: 7081
```

Demo

Things to improve

1. Lock Transfer

1. Use Redis to save unique key to avoid redundant transfer in short time

2. Confirmation Code

1. Need to send confirmation code to user devices when transfer amount is large

3. Flow Control

1. Prevent multiple requests attack

Thanks for listening

Any Questions?

My contact info:

LinkedIn: https://www.linkedin.com/in/chingpo-bob-lin
IG: @bob.lin.92
Wechat: ljb_a_552
Line: linbob92
Github: https://github.com/Chingpo-Lin
Email: ljb199992@gmail.com
Resume: Ask me for latest version