链滴

蚂蚁开放性笔试题 -- 最短时间的支付方式组合

作者：sirwsl

原文链接：https://ld246.com/article/1637249162784

来源网站：链滴

题目

用户有多种支付方式（余额、红包、优惠券，代金券等），假如每种支付方式通过调用远程服务获取用性。

在外部资源环境不变情况下，请设计程序以最短响应时间获得尽可能多的可用支付方式列表。

假定支付方式可用性咨询接口定义：PaymentRemoteSerivce

接口方法：ConsultResult isEnabled(String paymentType);

返回结果：

```
public class ConsultResult {

public ConsultResult (boolean isEnable,String  errorCode){
    this.isEnable = isEnable;
    this.errorCode= errorCode;
}

/** 咨询结果是否可用*/
private boolean isEnable;

/** 错误码 */
private String errorCode;

public boolean getIsEnable(){
    return isEnable;
}

public String getErrorCode(){
    return errorCode;
}
```

```
}
```

解题思路

1、进行远程调用

2、远程调用后存缓存

3、发生变化通过socket或者mq进行通知，刷新缓存

4、定时刷新缓存、防止某时刻大量请求过来而没得缓存

开干

0、定义支付方式常量

```
package payWay.other;


/**
 * 支付方式常量
 * 余额、红包、优惠券，代金券等
 * @author sirwsl
 */
public class Constants {
    /**
     * 余额
     */
    public  static final String BALANCE = "10";

    /**
     * 红包
     */
    public static final String RED_ENVELOPE = "20";

    /**
     * 优惠券
     */
    public static final String COUPONS = "30";

    /**
     * 代金券
     */
    public static final String VOUCHERS = "40";

    /**
     * 其他
     */
    public static final String OTHER = "50";

}
```

1、定义支付方式枚举值

```
package payWay.other;


/**
 * 余额、红包、优惠券，代金券等
 * @author sirwsl
 */

public enum PayWayEnum {

    //性别枚举
    BALANCE("余额", Constants.BALANCE),
    RED_ENVELOPE("红包", Constants.RED_ENVELOPE),
    COUPONS("优惠券",Constants.COUPONS),
    VOUCHERS("代金券",Constants.VOUCHERS),
    OTHER("其他",Constants.VOUCHERS);

    private String name;

    private String value;

    PayWayEnum(String name, String code) {
        this.name = name;
        this.value = code;
    }

    public static PayWayEnum matchByValue(String value) {
        for (PayWayEnum item : PayWayEnum.values()) {
            if (item.value.equals(value)) {
                return item;
            }
        }
        return OTHER;
    }

    public static PayWayEnum matchByName(String name) {
        for (PayWayEnum item : PayWayEnum.values()) {
            if (item.name.equals(name)) {
                return item;
            }
        }
        return OTHER;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getValue() {
        return value;
```

```
    }

    public void setValue(String value) {
        this.value = value;
    }
}
```

2、将题目中的返回值封装类诺下来

```
package payWay.domain;

/**
 * @author sirwsl
 */
public class ConsultResult {

    public ConsultResult (boolean isEnable,String  errorCode){
        this.isEnable = isEnable;
        this.errorCode= errorCode;
    }

    /** 咨询结果是否可用*/
    private boolean isEnable;

    /** 错误码 */
    private String errorCode;

    public boolean getIsEnable(){
        return isEnable;
    }

    public String getErrorCode(){
        return errorCode;
    }

}
```

3、写远程调用接口

```
package payWay.service;

import payWay.domain.ConsultResult;

/**
 * @author sirwsl
 */
public interface PaymentRemoteService {

    ConsultResult isEnabled(String paymentType);
}
```

4、接口实现

```
package payWay.service.impl;

import payWay.domain.ConsultResult;
import payWay.other.Constants;
import payWay.service.PaymentRemoteService;

import java.util.concurrent.TimeUnit;

/**
 * @author sirwsl
 */
public class PaymentRemoteServiceImpl implements PaymentRemoteService {

    public static final String SUCCESS = "200";

    public static final String SERVER_ERROR = "501";


    @Override
    public ConsultResult isEnabled(String paymentType) {
        try {
            Thread.sleep(1000*3);
            switch (paymentType) {
                case Constants.BALANCE:
                case Constants.VOUCHERS:
                case Constants.RED_ENVELOPE:
                case Constants.COUPONS:
                case Constants.OTHER:
                    return new ConsultResult(true, SUCCESS);
                default:
                    return new ConsultResult(false, SUCCESS);
            }
        } catch (InterruptedException e) {
            return new ConsultResult(false, SERVER_ERROR);
        }

    }
}
```

5、模拟缓存

```
package payWay.other;

import java.util.HashMap;
import java.util.Map;

/**
 * 模拟缓存操作
 * @author sirwsl
 */
public class CacheMock<T,V> {
    Map<T,V> cache = new HashMap<>();

    public synchronized boolean set(T key,V value){
```

```
        cache.put(key,value);
        return true;
    }

    public synchronized boolean set(T key,V value,long secconds){
        cache.put(key,value);
        return true;
    }

    public V get(T key){
        if (cache.containsKey(key)){
            return cache.get(key);
        }
        return null;

    }

}
```

6、实现测试、封装获取方式

```
package payWay;

import jdk.nashorn.internal.runtime.logging.Logger;
import org.omg.CORBA.BAD_CONTEXT;
import payWay.domain.ConsultResult;
import payWay.other.CacheMock;
import payWay.other.Constants;
import payWay.other.PayWayEnum;
import payWay.service.PaymentRemoteService;
import payWay.service.impl.PaymentRemoteServiceImpl;
import sun.rmi.runtime.Log;

import javax.management.StringValueExp;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.Objects;
import java.util.concurrent.Executors;
import java.util.concurrent.ThreadPoolExecutor;
import java.util.concurrent.TimeUnit;

/**
 * 业务处理
 */
public class Main {
    static CacheMock<String,ConsultResult> cacheMock = new CacheMock<>();
    static PaymentRemoteService paymentRemoteService = new PaymentRemoteServiceImpl();
    //测试
    public static void main(String[] args) {
        long start = System.currentTimeMillis();
        List<String> test1 = getPayWayList();
        System.out.println("================第 1 次时间耗时:"+ (System.currentTimeMillis
```

```
) - start)+"================\n结果： ");
        test1.stream()
                .map(li -> PayWayEnum.matchByValue(li).getName())
                .forEach(li -> System.out.println(li +" "));

        start = System.currentTimeMillis();
        List<String> test2 = getPayWayList();
        System.out.println("================第 2 次时间耗时:"+ (System.currentTimeMillis
) - start)+"================\n结果： ");
        test2.stream()
                .map(li -> PayWayEnum.matchByValue(li).getName())
                .forEach(li -> System.out.println(li +" "));

        start = System.currentTimeMillis();
        List<String> test3 = getPayWayList();
        System.out.println("================第 3 次时间耗时:"+ (System.currentTimeMillis
) - start)+"================\n结果： ");
        test3.stream()
                .map(li -> PayWayEnum.matchByValue(li).getName())
                .forEach(li -> System.out.println(li +" "));


    }

    /**
     * 获取服务排列组合
     * @return : 可用支付方式集合
     */
    public static List<String> getPayWayList(){
        //定义收集器
        List<String> useWayCollect = new ArrayList<>(Arrays.asList(Constants.BALANCE,Consta
ts.RED_ENVELOPE,
                Constants.COUPONS,Constants.VOUCHERS,Constants.OTHER));
        List<String> canUseWayCollect = new ArrayList<>(4);
        //多线程从缓存或者远程调用接口
        ThreadPoolExecutor executor = (ThreadPoolExecutor) Executors.newFixedThreadPool(5);
        for(int i = 0; i < useWayCollect.size();i++) {
            String li = useWayCollect.get(i);
            Runnable runnable = ()-> {
                //缓存中拿值
                ConsultResult hasUse = cacheMock.get(li);
                //值为空则远程调用
                if (Objects.isNull(hasUse)) {
                    //超时熔断机制，此处不方便搞
                    hasUse = paymentRemoteService.isEnabled(li);
                    //写缓存
                    cacheMock.set(li, hasUse);
                }
                //去除不能用的
                if (hasUse.getIsEnable()) {
                    canUseWayCollect.add(li);
                }
            };
            executor.execute(runnable);
```

```
        }
        try {
            executor.shutdown();
            if(!executor.awaitTermination(3000L, TimeUnit.MILLISECONDS)){
                executor.shutdownNow();
            }
        }catch (InterruptedException e) {
            executor.shutdownNow();
            Thread.currentThread().interrupt();
        }
        //返回结果
        return canUseWayCollect;
    }


}
```

结束后发现的缺陷

1、没有对MQ、或者socket进行模拟实现

2、模拟缓存、没设置失效时间

3、没有定时任务实时刷新缓存

4、采用原生java写，没用maven进行管理，有些地方有很大优化

5、由于没有对mq进行模拟实现，如果数据变化后，就会造成fegin与cache数据不一致情况

6、因为说的是远程调用，没有考虑熔断降级的情况