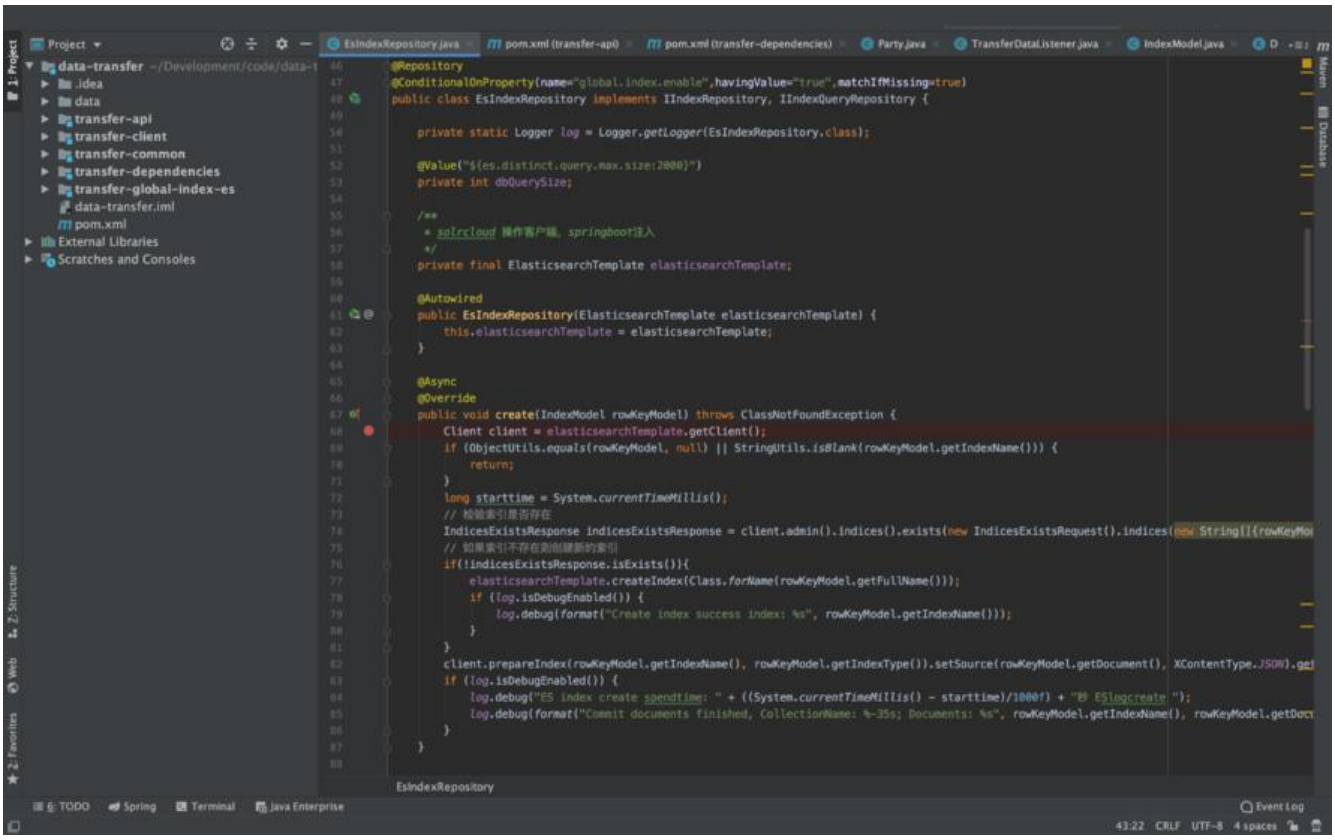
记一次多条件 if...else... 选择语句代码优化项目实战

作者：MrBox-boot

原文链接：https://ld246.com/article/1576051026754

来源网站：链滴



最近在写业务代码时，发现有一段业务的逻辑非常蛋疼，由于选择逻辑太多，以需要靠多个if...else...来实现。优化前大致是这样的：

```
@Override
public CustomerEntry4App customerHandel(String customerCode, String customerStatus){
    if (CustStatusEnum.POTENTIAL.getCode().equals(customerStatus)){
        // TODO do something
    } else if (CustStatusEnum.NORMAL.getCode().equals(customerStatus)){
        // TODO do something
    } else if (CustStatusEnum.NON_EFFECTIVE.getCode().equals(customerStatus)){
        // TODO do something
    }
    return null;
}
```

优化后代码：

```
@Override
public CustomerEntry4App customerHandel(String custCode, String custStatusId) throws Exc
ption{
    InnerCommand innerCommand = context.getInstance(custStatusId);
    return innerCommand.queryCustDetailByCustCode(custCode, custStatusId);
}
```

整体思路-使用策略模式：

- 所有策略使用枚举代替。

```
package com.asiainfo.customer.individual.enums;

import com.ai.bss.custcommon.enums.CustStatusEnum;
import java.util.HashMap;
import java.util.Map;

/**
 * Function:
 *
 * @author zhangds5
 * Date: 2019/11/29 18:38
 * @since JDK 1.8
 */
public enum CustStatusCommandEnum {

    POTENTIAL(CustStatusEnum.POTENTIAL.getCode(), "潜客查询", "com.asiainfo.customer.individual.service.impl.PotentialCommand"),
    NORMAL(CustStatusEnum.NORMAL.getCode(), "正式客户查询", "com.asiainfo.customer.individual.service.impl.NormalCommand"),
    NON_EFFECTIVE(CustStatusEnum.NON_EFFECTIVE.getCode(), "散客查询", "com.asiainfo.customer.individual.service.impl.PotentialCommand"),
    ;

    /** 枚举值码 */
    private final String custStatus;

    /** 枚举描述 */
    private final String desc;
    /**
     * 实现类
     */
    private final String clazz;


    /**
     * 构建一个 。
     * @param custStatus 枚举值码。
     * @param desc 枚举描述。
     */
    private CustStatusCommandEnum(String custStatus, String desc, String clazz) {
        this.custStatus = custStatus;
        this.desc = desc;
        this.clazz = clazz ;
    }

    /**
     * 得到枚举值码。
     * @return 枚举值码。
     */
    public String getCommandType() {
        return custStatus;
    }
    /**
     * 获取 class。
```

```
     * @return class。
     */
    public String getClazz() {
        return clazz;
    }

    /**
     * 得到枚举描述。
     * @return 枚举描述。
     */
    public String getDesc() {
        return desc;
    }

    /**
     * 得到枚举值码。
     * @return 枚举值码。
     */
    public String code() {
        return custStatus;
    }

    /**
     * 得到枚举描述。
     * @return 枚举描述。
     */
    public String message() {
        return desc;
    }

    /**
     * 获取全部枚举值码。
     *
     * @return 全部枚举值码。
     */
    public static Map<String,String> getAllStatusCode() {
        Map<String,String> map = new HashMap<String, String>(16) ;
        for (CustStatusCommandEnum status : values()) {
            map.put(status.getCommandType(),status.getDesc()) ;
        }
        return map;
    }

    public static Map<String,String> getAllClazz() {
        Map<String,String> map = new HashMap<String, String>(16) ;
        for (CustStatusCommandEnum status : values()) {
            map.put(status.getCommandType().trim(), status.getClazz()) ;
        }
        return map;
    }
}
```

- 定义一个 InnerCommand 接口，其中有一个 process 函数交给具体的业务实现。

```
package com.asiainfo.customer.individual.service.interfaces;

import com.asiainfo.customer.individual.entry.CustomerEntry4App;

public interface InnerCommand {
    /**
     * 执行查询
     * @param custCode
     * @param custStatusId
     */
    CustomerEntry4App queryCustDetailByCustCode(String custCode, String custStatusId)thro
s Exception;
}
```

● 不通策略均实现InnerCommand接口，并注册到Spring容器中。

```
/**
 * 正式客户查询
 * @param custCode
 * @param custStatusId
 * @return
 * @throws Exception
 */
@Service
public class NormalCommand implements InnerCommand {

    @Override
    public CustomerEntry4App queryCustDetailByCustCode(String custCode, String custStatusI
) throws Exception {
        CustomerEntry4App result = new CustomerEntry4App();
        // do something
        return result;
    }
}


/**
 * 散客户查询
 * @param custCode
 * @param custStatusId
 * @return
 * @throws Exception
 */
@Service
public class PotentialCommand implements InnerCommand {

    @Override
    public CustomerEntry4App queryCustDetailByCustCode(String custCode, String custStatusI
) throws Exception {
        CustomerEntry4App result = new CustomerEntry4App();
        // do something
        return result;
    }
}
```

- 根据指令获取对应策略的实现实例。

```
@Component
public class InnerCommandContext {

    private final static Logger LOGGER = LoggerFactory.getLogger(InnerCommandContext.class
;

    /**
     * 获取执行器实例
     * @param custStatusId 执行器实例
     * @return
     */
    public InnerCommand getInstance(String custStatusId) {

        Map<String, String> allClazz = CustStatusCommandEnum.getAllClazz();
        String clazz = allClazz.get(custStatusId);
        InnerCommand innerCommand = null;
        try {
            innerCommand = (InnerCommand) SpringApplicationContext.getInstance().getBean(Cl
ss.forName(clazz));
        } catch (Exception e) {
            LOGGER.error(e.getMessage(), e);
        }
        return innerCommand;
    }
}
```

- 具体使用，是不是看起来精简了很多。

```
@Override
public CustomerEntry4App queryCustDetailByCustCode(String custCode, String custStatusId)
hrows Exception {
    InnerCommand innerCommand = context.getInstance(custStatusId);
    return innerCommand.queryCustDetailByCustCode(custCode, custStatusId);
}
```

总结

- 当然优化方式有很多，能满足只需要两行代码就能替换以前复杂的 if else，同时也能灵活扩展。

参考

https://crossoverjie.top/2019/01/30/java-senior/design-if-else/