

工厂模式和策略模式结合使用的案例介绍



文景大大 (关注) IP属地: 广东

1 2021.08.20 21:32:47 字数 582 阅读 1,367

一、前言

在前面的文章中，我们有单独介绍过工厂模式和策略模式，这两种模式是实际开发中经常会用到的，今天来介绍下将两种模式结合起来使用的场景及案例，这种结合的模式也更加的常用，能帮助我们减少if-else的使用的同时，让代码逻辑也清晰简洁、扩展性高。

《工厂模式》

《策略模式》

二、案例

我们假设如下业务场景：

在某CRM系统中，针对不同来源（电话、短信、微信）的客户需要执行各自的名单创建逻辑。

首先，我们新建一个工程，引入相关的依赖信息：

```
1 <dependency>
2   <groupId>org.springframework.boot</groupId>
3   <artifactId>spring-boot-starter-web</artifactId>
4 </dependency>
5 <dependency>
6   <groupId>org.springframework.boot</groupId>
7   <artifactId>spring-boot-starter-test</artifactId>
8   <scope>test</scope>
9 </dependency>
10 <dependency>
11   <groupId>org.projectlombok</groupId>
12   <artifactId>lombok</artifactId>
13   <version>1.18.18</version>
14 </dependency>
15 <dependency>
16   <groupId>junit</groupId>
17   <artifactId>junit</artifactId>
18   <version>4.12</version>
19   <scope>test</scope>
20 </dependency>
```

然后，我们新建实体类，代表客户的信息：

```
1 @Data
2 @Builder
3 @NoArgsConstructor
4 @AllArgsConstructor
5 public class Customer {
6
7     private String name;
8     private Integer age;
9     private String address;
10
11 }
```

然后，按照策略模式的样子，我们新建一个抽象类代表公共的策略，然后分别创建手机、短信和微信来源策略：

```
1 @Service
2 public abstract class CommonChannelStrategy {
3
4     /**
5      * 定义公共的检查逻辑
6      * @param customer 客户信息
7      * @return true-检查通过
8      */
9     public boolean commonCheckValidate(Customer customer) {
10         return !ObjectUtils.isEmpty(customer.getName())
11             && !ObjectUtils.isEmpty(customer.getAge())
12             && !ObjectUtils.isEmpty(customer.getAddress());
13     }
14
15     /**
16      * 各个渠道特殊的检查逻辑，各自去实现
17      * @param customer 客户信息
18      * @return true-检查通过
19      */
20     public abstract boolean specificCheckValidate(Customer customer);
21
22 }
```

```
1 @Service
2 public class TelChannelStrategy extends CommonChannelStrategy{
3
4     /**
5      * 电话渠道客户检查策略：必须大于等于18岁才合法
6      * @param customer 客户信息
7      * @return true-合法
8      */
9     @Override
10     public boolean specificCheckValidate(Customer customer) {
11         return !ObjectUtils.isEmpty(customer) && customer.getAge() >= 18;
12     }
13 }
```

```
1 @Service
2 public class SmsChannelStrategy extends CommonChannelStrategy{
3
4     /**
5      * 短信渠道客户检查策略：必须大于等于12岁才合法
6      * @param customer 客户信息
7      * @return true-合法
8      */
9     @Override
10     public boolean specificCheckValidate(Customer customer) {
11         return !ObjectUtils.isEmpty(customer) && customer.getAge() >= 12;
12     }
13 }
```

热门故事

姐姐失踪两年了，但今天我却收到了她的一条短信

为了救下他的白月光，他递来了毒药，而我心甘情愿地喝下了

被骗去柬埔寨的人有多惨？器官被明码标价

被调去男子监狱工作的女医生有多累？

诈骗组织多可怕？我被绑进小黑屋遭受折磨

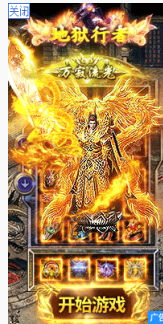
去医院却撞见老公全家陪别的女人产检，我淡定拿出老公的不育报告，...

前世渣男把我逼晕还叫我别怕，转世彻底黑化的我复仇反杀

妹妹过失杀人，警察来时，我捡起了那把滴血的刀

我被校霸堵在巷口，却发现他是我谈了三个月的网恋对象

我和网恋对象奔现，却发现他们是我们学校赫赫有名的海王



```
7  /*
8  @Override
9  public boolean specificCheckValidate(Customer customer) {
10      return !ObjectUtils.isEmpty(customer) && customer.getAge() >= 22;
11  }
12 }
```

这些策略如何在合适的时机使用呢？在讲策略模式的时候，我们是借助一个环境类，持有抽象策略的引用，然后初始化该环境类的时候，传进来一个具体策略对象赋值给抽象策略。

这次讲解的是整合工厂模式，使用静态工厂方法，根据入参来从内存中找到早已初始化好的具体策略对象，即枚举中的实例对象。

```
1 @AllArgsConstructor
2 @Getter
3 public enum ENUM_CUSTOMER_CHANNEL {
4     /**
5      * 电话
6      * 短信
7      * 微信
8      */
9     TEL_CHANNEL("TEL_CHANNEL", "电话来源", "telChannelStrategy"),
10    SMS_CHANNEL("SMS_CHANNEL", "短信来源", "smsChannelStrategy"),
11    WECHAT_CHANNEL("WECHAT", "微信来源", "wechatChannelStrategy");
12
13    /**
14     * 渠道编码
15     */
16    private final String channelCode;
17    /**
18     * 渠道名称
19     */
20    private final String channelName;
21    /**
22     * 渠道处理方法
23     */
24    private final String channelService;
25
26 }
```

```
1 @Slf4j
2 public class ChannelFactory {
3
4     /**
5      * 存放不同渠道来源名单的处理实例bean
6      * telChannelService
7      * smsChannelService
8      * wechatChannelService
9      */
10    private static final Map<String, String> SERVICE_BEAN_MAP = new HashMap<>();
11
12    // 系统启动时就需要将各个渠道的处理实例bean放到map中
13    static {
14        for (ENUM_CUSTOMER_CHANNEL channel : ENUM_CUSTOMER_CHANNEL.values()) {
15            SERVICE_BEAN_MAP.put(channel.getChannelCode(), channel.getChannelService());
16        }
17    }
18
19    /**
20     * 定义静态工厂方法
21     *
22     * @param channelCode 渠道编码
23     * @return CommonChannelStrategy 具体的处理该渠道客户的实例bean
24     */
25    public static CommonChannelStrategy getChannelStrategy(String channelCode) {
26        String beanName = SERVICE_BEAN_MAP.get(channelCode);
27        if (ObjectUtils.isEmpty(beanName)) {
28            log.error("渠道类型: {}错误, 请检查!", channelCode);
29            return null;
30        }
31        return (CommonChannelStrategy) SpringBeanUtils.getBean(beanName);
32    }
33
34 }
```

```
1 @Component
2 public class SpringBeanUtils implements ApplicationContextAware {
3
4     private static ApplicationContext applicationContext;
5
6     @Override
7     public void setApplicationContext(ApplicationContext applicationContext) throws BeansException {
8         if (SpringBeanUtils.applicationContext == null) {
9             SpringBeanUtils.applicationContext = applicationContext;
10        }
11    }
12
13    // 获取applicationContext
14    public static ApplicationContext getApplicationContext() {
15        return applicationContext;
16    }
17
18    // 通过name获取Bean
19    public static Object getBean(String name) {
20        return getApplicationContext().getBean(name);
21    }
22
23    // 通过class获取Bean
24    public static <T> T getBean(Class<T> clazz) {
25        return getApplicationContext().getBean(clazz);
26    }
27
28    // 通过name, 以及Class返回指定Bean
29    public static <T> T getBean(String name, Class<T> clazz) {
30        return getApplicationContext().getBean(name, clazz);
31    }
32
33 }
```

如此，我们的策略模式和静态工厂方法模式整合好了，我们写一个单元测试一下：

```
import com.example.aopdemo.AopDemoApplication;
import lombok.extern.slf4j.Slf4j;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
import org.springframework.util.ObjectUtils;

@Slf4j
@RunWith(SpringJUnit4ClassRunner.class)
@SpringBootTest(classes = AopDemoApplication.class)
public class ChannelFactoryTest {

    @Test
    public void getChannelStrategy() {
        Customer customer = getCustomer();
        CommonChannelStrategy channelStrategy = ChannelFactory.getChannelStrategy("TEL_CHANNEL");
    }
}
```

热门文章

姐姐失踪两年了，但今天我收到了她的一条短信

为了救下他的白月光，他递来了毒药，而我心甘情愿地喝下了

被骗去柬埔寨的人有多惨？器官被明码标价

被调去男子监狱工作的女医生有多累？

诈骗组织多可怕？我被绑进小黑屋遭受折磨

去医院却撞见老公全家陪别的女人产检，我淡定拿出老公的不育报告，...

前世渣男把我迷晕还叫我别怕，转世彻底黑化的我复仇反杀

妹妹过失杀人，警察来时，我捡起了那把滴血的刀

我被校霸堵在巷口，却发现他是我谈了三个月的网恋对象

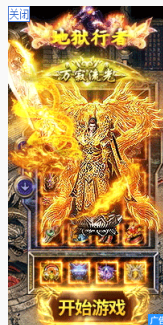
我和网恋对象奔现，却发现他是我们学校赫赫有名的海王

7赞

赞赏

回

更多好文



7赞

赞赏

更多好文

```
24 }
25     log.info("该客户{}电话渠道合法！", customer.getName());
26 }
27
28 private Customer getCustomer(){
29     return Customer.builder()
30         .name("张三")
31         .age(19)
32         .address("上海市")
33         .build();
34 }
```

运行后成功执行了电话渠道客户策略的行为。

三、总结

为什么要使用这种策略模式和静态工厂方法模式结合的方案呢？

- 即减少了if-else代码；
- 可扩展性高了；
- 避免了自己new对象；
- 不需要环境类以及新建环境类对象；
- 大部分复杂业务场景的系统都会选择使用这种方案，比较成熟。

7人点赞 >

设计模式

更多精彩内容，就在简书APP



“小礼物走一走，来简书关注我”

赞赏支持

还没有人赞赏，支持一下

文景大大 一个怀有激情和梦想的开发者，爱好知识分享、阅读交流、运动健身... 总资产65 共写了39.0W字 获得570个赞 共271个粉丝

人面猴

序言：七十年代末，一起剥皮案震惊了整个滨河市，随后出现的几起案子，更是在滨河造成了极大的恐慌。老刑警刘岩，带你破解...

沈念sama 阅读 20372 评论 1 赞 46

前方高能预警

1. 周嘉洛拥有一个不好说有没有用的异能。异能管理局的人给他的异能起名为「前方高能预警」。2. 周嘉洛第一次发...

沈念sama 阅读 10176 评论 1 赞 51

死咒

序言：滨河连续发生了三起死亡事件，死亡现场离奇诡异，居然都是意外死亡，警方通过查阅死者的电脑和手机，发现死者居然都...

沈念sama 阅读 6106 评论 1 赞 55

救了他两次的神仙让他今天三更去死

文/潘晓璐 我一进店门，熙熙攘攘的掌柜王于贵愁眉苦脸地迎上来，“王大人，你说我怎么就摊上这事。”“怎么了？”“我有些...

开封第一讲书人 阅读 5292 评论 0 赞 13

道士缉凶录：失踪的卖菜人

文/不坏的土叔 我叫张陵，是天一观的道长。经常有香客问我，道长，这世上最难降的妖魔是什么？我笑而不...

开封第一讲书人 阅读 4318 评论 0 赞 5

港岛之恋（遗憾婚礼）

正文 为了忘掉前任，我火速办了婚礼，结果婚礼上，老公的妹妹穿的比我还像新娘。我一直安慰自己，他们只是感情好，可当我...

茶点故事 阅读 6548 评论 0 赞 54

恶毒庶女顶嫁案：这布局不是一般人想出来的

文/花漫 我一把揭开白布。她就那样静静地躺着，像睡着了一般。火红的嫁衣衬着肌肤如雪。梳的纹丝不乱的头发上，一...

开封第一讲书人 阅读 8849 评论 0 赞 5

城市分裂传说

那天，我端着相机与录音，去河边找鬼。笑死，一个胖子当着我的面吹牛，可吹牛的内容都是我干的。我是一名探鬼主播，决...

沈念sama 阅读 2525 评论 1 赞 70

双鸳鸯连环套：你想象不到人心有多黑

文/苍兰青墨 我猛地睁开眼，长吁一口气：“原来是场噩梦啊……”“哼！你这毒妇竟也来了？”一声冷哼从身侧响起，我...

开封第一讲书人 阅读 2468 评论 0 赞 0

万荣杀人案实录

热门故事

姐姐失踪两年了，但今天我却收到了她的一条短信

为了救下他的白月光，他递来了毒药，而我心甘情愿地喝下了

被骗去柬埔寨的人有多惨？器官被明码标价

被调去男子监狱工作的女医生有多累？

诈骗组织多可怕？我被绑进小黑屋遭受折磨

去医院却撞见老公全家陪别的女人产检，我淡定拿出老公的不育报告，...

前世渣男把我迷晕还叫我别怕，转世彻底黑化的我复仇反杀

妹妹过失杀人，警察来时，我捡起了那把滴血的刀

我被校霸堵在巷口，却发现他是我谈了三个月的网恋对象

我和网恋对象奔现，却发现他们是我们学校赫赫有名的海王

