

Java设计模式之（十四）——策略模式

目录

- 1、什么是策略模式?
- 2、策略模式定义
- 3、策略模式通用代码
- 4、用策略模式改写if-else
 - 4.1 常规写法
 - 4.2 策略模式改写
- 4、策略模式优点
- 5、策略模式应用场景

[回到顶部](#)

1、什么是策略模式?

Define a family of algorithms, encapsulate each one, and make them interchangeable. Strategy lets the algorithm vary independently from clients that use it.

策略模式（Strategy Pattern）：定义一族算法类，将每个算法分别封装起来，让它们可以互相替换。

[回到顶部](#)

2、策略模式定义

image-20210923211729495

①、Context封装角色

它也叫做上下文角色，起承上启下封装作用，屏蔽高层模块对策略、算法的直接访问，封装可能存在的变化。

②、Strategy 抽象策略角色

策略、算法家族的抽象，通常为接口，定义每个策略或算法必须具有的方法和属性。

③、ConcreteStrategy 具体策略角色

实现抽象策略中的操作，该类含有具体的算法。

[回到顶部](#)

3、策略模式通用代码

```
1 public class Context {
2     // 抽象策略
3     private Strategy strategy = null;
4     // 构造函数设置具体策略
5     public Context(Strategy strategy) {
6         this.strategy = strategy;
7     }
8     // 封装后的策略方法
9     public void doAnything() {
10        this.strategy.doSomething();
11    }
12 }
```

个人公众号，欢迎关注交流



程序员一站式导航网站

昵称：YSOcean
园龄：6年2个月
粉丝：5798
关注：15
[+加关注](#)

我的标签

- Linux系列教程(24)
- 深入理解计算机系统(23)
- MyBatis详解系列(16)
- Java数据结构和算法(15)
- Java设计模式系列(14)
- Redis详解(13)
- Java虚拟机详解(11)
- JDK源码解析(11)
- Maven系列教程(8)
- Spring入门系列(8)
- [更多](#)

友情链接

程序员一站式导航

阅读排行榜

- 1. Nginx（三）-----nginx反向代理(576168)
- 2. Java关键字(一)——instanceof(532732)
- 3. Tomcat 部署项目的三种方法(280148)

```
11     }
12 }
```

```
1 public interface Strategy {
2     // 策略模式的运算法则
3     public void doSomething();
4 }
```

```
1 public class ConcreteStrategy1 implements Strategy{
2     @Override
3     public void doSomething() {
4         System.out.println("ConcreteStrategy1");
5     }
6 }
```

```
1 public class ConcreteStrategy2 implements Strategy{
2     @Override
3     public void doSomething() {
4         System.out.println("ConcreteStrategy2");
5     }
6 }
7
```

测试:

```
1 public class StrategyClient {
2     public static void main(String[] args) {
3         // 声明一个具体的策略
4         Strategy strategy = new ConcreteStrategy1();
5         // 声明上下文对象
6         Context context = new Context(strategy);
7         // 执行封装后的方法
8         context.doAnything();
9     }
10 }
```

[回到顶部](#)

4、用策略模式改写if-else

假设我们要处理一个office文件，分为三种类型 docx、xlsx、pptx，分别表示Word文件、Excel文件、PPT文件，根据文件后缀分别解析。

4.1 常规写法

```
1 public class OfficeHandler {
2
3     public void handleFile(String filePath) {
4         if(filePath == null) {
5             return;
6         }
7         String fileExtension = getFileExtension(filePath);
8         if(("docx").equals(fileExtension)) {
9             handlerDocx(filePath);
10        }else if(("xlsx").equals(fileExtension)) {
11            handlerXlsx(filePath);
12        }else if(("pptx").equals(fileExtension)) {
13            handlerPptx(filePath);
14        }
15    }
16 }
```

4. mybatis 详解（五）-----
动态SQL(140720)

5. Java 集合详解(139154)

最新评论

1. Re:Java数据结构和算法
(十) ——二叉树

7.2 中，删掉节点9后，把7变成了8的右节点，这样的话，这棵树就不是二叉搜索树了。所以这个删除方法是不是错了？

--sunnyair001

2. Re:Nginx（四）-----ngi
nx 负载均衡

讲的不错，看懂了。

--成佛在西天

3. Re:Java虚拟机详解（二）
-----运行时内存结构

@liu_whut 是的，没错。果然评论有人指出来...

--懒惰的星期六

4. Re:Java数据结构和算法
(一) ——简介

感谢大佬

--星空12121

5. Re:一门能让你五分钟学会
的语言-Brainfuck

@laya1211 🍌...

--YSOcean

```
15     }
16
17     public void handlerDocx(String filePath) {
18         System.out.println("处理docx文件");
19     }
20     public void handlerXlsx(String filePath) {
21         System.out.println("处理xlsx文件");
22     }
23     public void handlerPptx(String filePath) {
24         System.out.println("处理pptx文件");
25     }
26     private static String getFileExtension(String filePath) {
27         // 解析文件名获取文件扩展名,比如 文档.docx, 返回 docx
28         String fileExtension =
29             filePath.substring(filePath.lastIndexOf(".") + 1);
30         return fileExtension;
31     }
```

处理逻辑全部放在一个类中,会导致整个类特别庞大,假设我们要新增一种类型处理,比如对于2007版之前的office文件,后缀分别是 doc/xls/ppt,那我们得增加 else if 逻辑,违反了开闭原则,如何解决这种问题呢,答案就是通过策略模式。

4.2 策略模式改写

```
1 public interface OfficeHandlerStrategy {
2     void handlerOffice(String filePath);
3 }
```

```
1 public class OfficeHandlerDocxStrategy implements
2     OfficeHandlerStrategy {
3     @Override
4     public void handlerOffice(String filePath) {
5         System.out.println("处理docx");
6     }
7 }
```

// 省略 OfficeHandlerXlsxStrategy/OfficeHandlerPptxStrategy 类

```
1 public class OfficeHandlerStrategyFactory {
2     private static final Map<String, OfficeHandlerStrategy> map = new
3     HashMap<>();
4     static {
5         map.put("docx", new OfficeHandlerDocxStrategy());
6         map.put("xlsx", new OfficeHandlerXlsxStrategy());
7         map.put("pptx", new OfficeHandlerPptxStrategy());
8     }
9     public static OfficeHandlerStrategy getStrategy(String type) {
10         return map.get(type);
11     }
12 }
```

测试:

```
1 public class OfficeHandlerStrategyClient {
2     public static void main(String[] args) {
3         String filePath = "C://file/123.xlsx";
4         String type = getFileExtension(filePath);
5         OfficeHandlerStrategy strategy =
6             OfficeHandlerStrategyFactory.getStrategy(type);
7         strategy.handlerOffice(filePath);
8     }
9 }
```