Java设计模式之（十四）——策略模式

目录

回到顶部

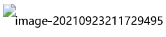
1、什么是策略模式?

> 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



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

```
public class Context {
    // 抽象策略
    private Strategy strategy = null;
    // 构造函数设置具体策略
    public Context(Strategy strategy) {
        this.strategy = strategy;
    }
    // 封装后的策略方法
    public void doAnything(){
        this.strategy.doSomething();
```

```
    }
}
```

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

```
public class ConcreteStrategy1 implements Strategy{
    @Override
    public void doSomething() {
        System.out.println("ConcreteStrategy1");
    }
}
```

```
public class ConcreteStrategy2 implements Strategy{
    @Override
    public void doSomething() {
        System.out.println("ConcreteStrategy2");
    }
}

```

测试：

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

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

```
public class OfficeHandler {

    public void handleFile(String filePath){
        if(filePath == null){
            return;
        }
        String fileExtension = getFileExtension(filePath);
        if(("docx").equals(fileExtension)){
            handlerDocx(filePath);
        }else if(("xlsx").equals(fileExtension)){
            handlerXlsx(filePath);
        }else if(("pptx").equals(fileExtension)){
            handlerPptx(filePath);
        }
```

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）------nginx 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

```
    }

    public void handlerDocx(String filePath){
        System.out.println("处理docx文件");
    }
    public void handlerXlsx(String filePath){
        System.out.println("处理xlsx文件");
    }
    public void handlerPptx(String filePath){
        System.out.println("处理pptx文件");
    }
    private static String getFileExtension(String filePath){
        // 解析文件名获取文件扩展名,比如 文档.docx，返回 docx
        String fileExtension = filePath.substring(filePath.lastIndexOf(".")+1);
        return fileExtension;
    }
}
```

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

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

```
public class OfficeHandlerDocxStrategy implements OfficeHandlerStrategy {
    @Override
    public void handlerOffice(String filePath) {
        System.out.println("处理docx");
    }
}
```

// 省略 OfficeHandlerXlsxStrategy/OfficeHandlerPptxStrategy 类

```
public class OfficeHandlerStrategyFactory {
    private static final Map<String,OfficeHandlerStrategy> map = new HashMap<>();
    static {
        map.put("docx",new OfficeHandlerDocxStrategy());
        map.put("xlsx",new OfficeHandlerXlsxStrategy());
        map.put("pptx",new OfficeHandlerPptxStrategy());
    }
    public static OfficeHandlerStrategy getStrategy(String type){
        return map.get(type);
    }
}
```

测试：

```
public class OfficeHandlerStrategyClient {
    public static void main(String[] args) {
        String filePath = "C://file/123.xlsx";
        String type = getFileExtension(filePath);
        OfficeHandlerStrategy strategy = OfficeHandlerStrategyFactory.getStrategy(type);
        strategy.handlerOffice(filePath);
```