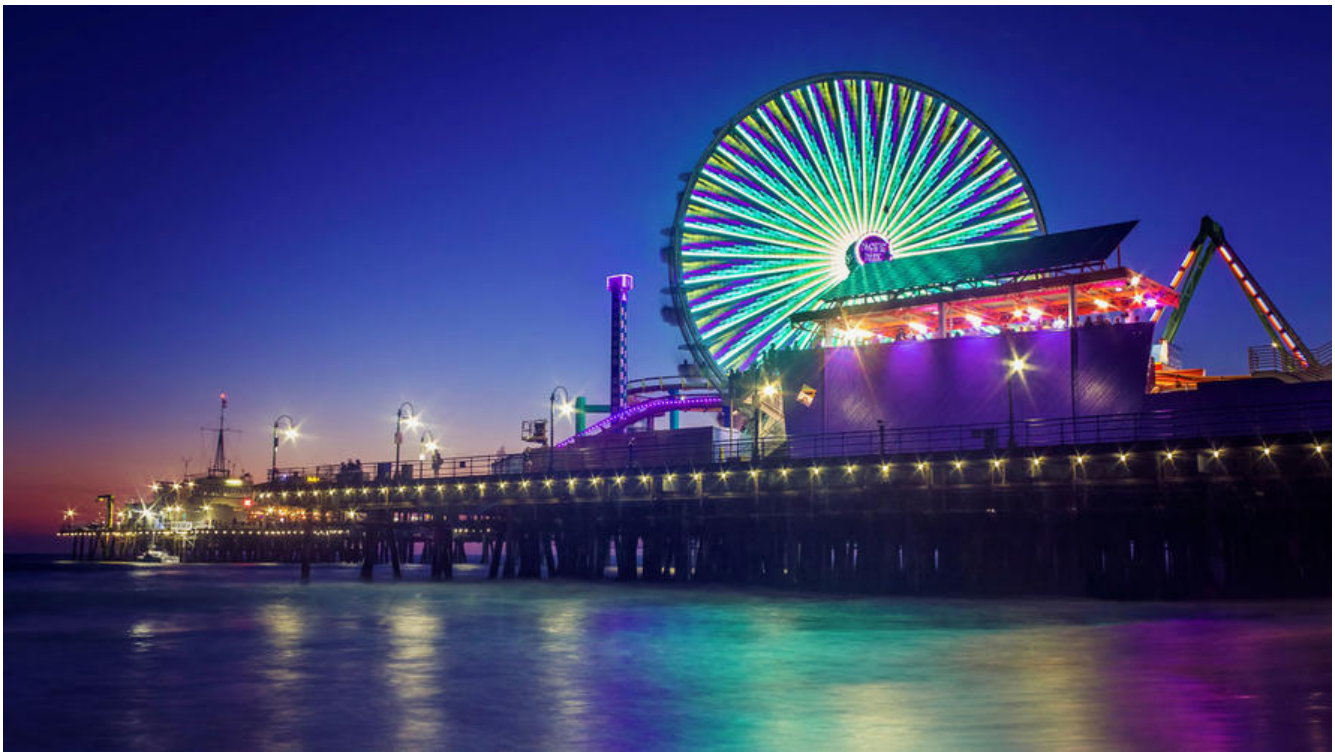
链滴

策略模式

作者：wanmisy
原文链接：https://ld246.com/article/1534841491340
来源网站：链滴



策略模式

定义了算法族，分别封装起来，让他们之间可以互相替换，此模式让算法的变化独立于使用算法的客，属于行为型模式。

意图

定义一系列的算法，把他们一个个封装起来，使他们可以相互替换

主要解决

在有多种算法相似的情况下，使用if...else所带来的复杂和难以维护

何时使用

一个系统有许多许多类，而区分他们的只是他们的直接行为

关键代码

实现一个接口

案例

1. 计算器的算法
2. 实现一个项目的多个主题
3. shiro

原文链接：策略模式

计算器demo

1. 定义接口

```
package caculator.service;
public interface Operation {
public int doOperation(int args1, int args2);
}
```

2. 实现

```
package caculator.service.Impl;
import caculator.service.Operation;
public class OperationAdd implements Operation {

@Override
public int doOperation(int args1, int args2) {
    return args1 + args2;
}
}
package caculator.service.Impl;
import caculator.service.Operation;
public class OperationSubstract implements Operation {

@Override
public int doOperation(int args1, int args2) {
    return args1 - args2;
}
}
```

3. 调用

```
package caculator.biz;
import caculator.service.Operation;
public class Caculator {
private Operation operation;

public void setOperation(Operation operation) {
    this.operation = operation;
}

public int doOperation(int args1, int args2) {
    return operation.doOperation(args1, args2);
}

public Caculator() {
    super();
}
```

```
}
```

4. 测试

```
package caculator.test;
import caculator.biz.Caculator;
import caculator.service.Impl.OperationAdd;
public class CaculatorTest {
public static void main(String[] args) {
Caculator caculator = new Caculator();
caculator.setOperation(new OperationAdd());
System.out.println(caculator.doOperation(15, 3));;
}
}
```

优点

遵循开闭原则，扩展性好

缺点

- 随着策略增加，类越来越多
- 所有的策略类都要暴露出去

原文链接：策略模式