代理模式

2020年2月12日 10:40

1. 概念

给目标对象提供一种代理，以控制对这个对象的访问。
代理对象在客户端和目标对象之间起到中介的作用。

2. 类型

1. 静态代理：在代码中显式地定义代理类的代理，代理关系在编译期间就已经确定，在代理类中对同名的业务方法进行包装，客户端通过调用代理类的业务方法来调用被代理类的方法，并进行增强。适合代理类较少的情况。
2. 动态代理：代理类在程序运行时被动态创建，是对接口的代理，不可对某个具体类进行代理。通过接口中方法名，在动态生成的新的代理类中调用同名的业务方法。
3. CGLib代理：可针对类实现进行代理，通过继承实现，生成的代理类为被代理类的子类，通过重写业务方法进行扩展。使用时需要引用第三方框架。

3. UML结构图

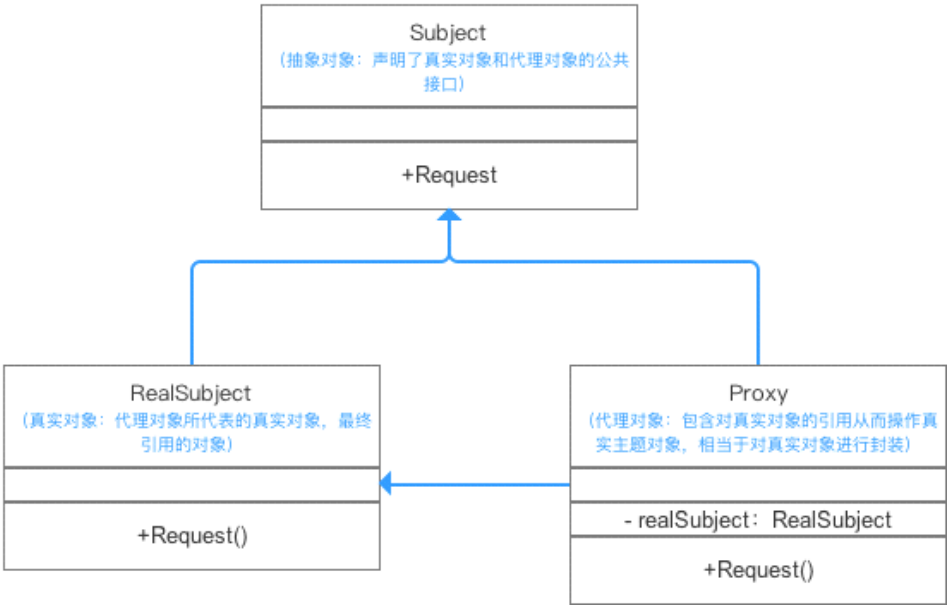

图2.1 代理模式UML结构图

4. 代理模式实例

小成希望买一台最新的顶配Mac电脑，但是由于国内还没上，只有美国才有，遂只能寻找代购进行购买，其中代购者（代理对象） 代替 我（真实对象） 去买Mac（间接访问的操作）。

1. 抽象接口类

```
public interface Subject {
  public void buyMac();
}
```

2. 被代理类

```
public class RealSubject implement Subject{
    @Override
    public void buyMac() {
        System.out.println("买一台Mac");
    }
}
```

3. 代理类

```
public class Proxy  implements Subject{
    private RealSubject realSubject ;
    @Override
    public void buyMac{

      //引用并创建真实对象实例，即”我“
      realSubject = new RealSubject();
      //调用真实对象的方法，进行代理购买Mac
      realSubject.buyMac () ;
      //代理对象额外做的操作
      this.WrapMac();
    }
     public void WrapMac(){
      System.out.println(”用盒子包装好Mac“);
    }
}
```

4. 客户端

```
public class ProxyPattern {
    public static void main(String[] args){
    Subject proxy = new Proxy () ;
    proxy.buyMac();
    }
}
```

5. 结果

```
买一台Mac
用盒子包装好Mac
```

5. 优/缺点

1. 优点

- 代理模式能将代理对象与真实被调用的目标对象分离
- 在一定程度上降低了系统的耦合度，扩展性好
- 保护目标对象
- 增强目标对象

2. 缺点

- 造成系统中类的数目增加
- 在客户端和目标对象增加一个代理对象，会造成请求处理速度变慢
- 增加了系统的复杂度

6. 适用场景

1. 保护目标对象
2. 增强目标对象

7. 扩展 - Spring代理选择

1. 当Bean有实现接口时，Spring会使用JDK动态代理
2. 当Bean没有实现接口时，Spring使用CGLib
3. 可强制使用CGLib

8. 参考资料

[1] https://www.jianshu.com/p/a8aa6851e09e