

我的个人小站

http://www.strolling.cn

我的微博

http://weibo.com/fujianthinking

个人资料



jiafu1115

访问：661652次

积分：9342

等级：BLOG 5

排名：第1051名

原创：176篇 转载：727篇

译文：1篇 评论：71条

文章搜索

文章分类

Design Pattern (7)

Database|Oracle (20)

Java|J2SE (445)

Linux|CentOS (26)

InfoSystem Program (4)

Education |Mobile (2)

Others (27)

Web Service (25)

Maven (26)

JVM (22)

Site Collection (4)

Concurrency (4)

Java IO (23)

Java Doc (4)

NetWork (4)

C++ (271)

IDE (4)

高并发程序设计入门    【活动】云计算行业圆桌论坛    【知识库】一张大图看懂Android架构    【征文】Hadoop十周年特别策划

——我与Hadoop不得不说的故事

## 代理模式和装饰者模式的异同点比较

标签：[decorator](#) [behavior](#) [设计模式](#) [string](#) [扩展](#)

2011-07-09 19:19    358人阅读    评论(0)

分类：[Design Pattern \(6\)](#)

1: 代理模式：

为其它对象提供一种代理以控制对目标对象的访问。即第三方对象通过访问代理对象而达到访问目标对象之目的，与此同时，代理对象在访问目标对象前后加入特定的逻辑以实现功能的扩展。

以静态代理为例：

```
package static_proxy;

public interface Movable {
    public void move();
}

package static_proxy;

import java.util.Random;

public class Tank implements Movable {
    @Override
    public void move() {
        System.out.println(" Tank is moving ");
        try {
            Thread.sleep(new Random().nextInt(1000));
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}

package static_proxy;

public class TankLogProxy implements Movable {
    private Movable obj;
    public TankLogProxy(Movable obj) {
        super();
        this.obj = obj;
    }
    @Override
```



[illegible]

a030703130: java正则表达式教程 学习地址: [http://www.java3z.com/cwbwebhome/...](http://www.java3z.com/cwbwebhome/)

replaceAll ()/appendReplacemer  
a030703130: 正则表达式 matcher类 学习地址: [http://www.java3z.com/cwbwebhome/...](http://www.java3z.com/cwbwebhome/)

#### 评论排行

- 知识都学杂了 (8)
- Java解惑笔记<不断更新: (4)
- JAVA多态与类型转化分析 (3)
- 读完《重构》的一些感想 (3)
- 关于Coding的学习与思考 (3)
- instanceOf 与 instanceof (3)
- 简单工厂模式、工厂方法 (3)
- java+方法覆盖必须不减 (3)
- Access restriction : The t (3)
- java 正斜杠与反斜杠之分 (3)

#### 好友博客

良师益友 Java Tips  
<http://www.javamex.com>

```
public void setMake_address(String make_address) {
    this.make_address = make_address;
}

public int getSpeed() {
    return speed;
}

public void setSpeed(int speed) {
    this.speed = speed;
}

public abstract void print_face();
}

package decorator;

public abstract class decorator_parent extends car_parent {
    // 装饰者父类
    protected car_parent car_parent_ref;

    public decorator_parent(car_parent carParentRef) {
        super();
        car_parent_ref = carParentRef;
    }

    @Override
    public void print_face() {
        car_parent_ref.print_face();
    }
}

package decorator;
public class decorator_audi_red extends decorator_parent {
    public decorator_audi_red(car_parent carParentRef) {
        super(carParentRef);
    }

    @Override
    public void print_face() {
        super.print_face();
        System.out.println("给 奥迪 喷涂红 - 颜色为 红色火焰");
    }
}

package decorator;
public class decorator_audi_purple extends decorator_parent {

    public decorator_audi_purple(car_parent carParentRef) {
        super(carParentRef);
    }

    @Override
    public void print_face() {
```

关闭

```
        super.print_face();
        System.out.println("给  奥迪  喷涂鸦  -  颜色为  紫色霞光");
    }
}

package decorator;

public class main_run {
    public static void main(String[] args) {

        car_parent audi_sub_ref = new audi_sub();
        audi_sub_ref.setMake_address("北京市朝阳区");
        audi_sub_ref.setSpeed(200);

        decorator_audi_red decorator_audi_red_ref = new decorator_audi_red(audi_sub_ref);

        decorator_audi_purple decorator_audi_purple_ref = new decorator_audi_purple(decorator_audi_red_ref);
        decorator_audi_purple_ref.print_face();
    }
}
```

装饰者模式：装饰者和被装饰者应继承或实现同一父类或接口，以表示它们为同一类对象（非必须）；装饰者对象对被装饰者对象增加一定的职责；可多次装饰。

可以发现：代理模式和装饰者模式上在语法形式上几乎完全一样，那么它们的区别在哪里呢？

装饰者模式：动态地给一个对象添加一些额外的职责。就增加功能来说，装饰模式相比生成子类更加灵活

代理模式：为其它对象提供一种代理以控制对这个对象的访问。

其实，它们的着重点一个在于“增加”职责，另一个在于“控制”访问。这是它们最本质的区别。

由此可以看到：学习设计模式重点在于“语义”上把握，而不是追求它的“形式”。

两者的定义

装饰器模式：能动态的新增或组合对象的行为。

代理模式：为其他对象提供一种代理以控制对这个对象的访问。

装饰模式是“新增行为”，而代理模式是“控制访问”。关键就是我们如何判断是“新增行为”还是“控制访问”。

原句：I think we're all aware that the decorator pattern is used to add behavior to existing code. This does not stop at one kind of behavior but any number of different things that we want to do.

意译：装饰类只能新增行为，不能跳过其他的任何一个行为。

顶 踩  
0 0

关闭

上一篇 [Java设计|生成器模式](#)

下一篇 [泛型方法](#)

我的同类文章

Design Pattern (6)

• 欢迎使用CSDN-markdown...2015-07-10 阅读 84

• 对代理模式与Java动态代理...2011-07-30 阅读 211

• Java设计|单例模式2011-07-09 阅读 249

• 抽象工厂和Builder模式区别2011-08-08 阅读 978

• Java设计|生成器模式2011-07-09 阅读 271

• 简单工厂模式、工厂方法模...2011-07-08 阅读 1507

猜你在找

设计模式课程

Android设计模式精解(第7课)：Adapter模式

Android设计模式精解(第10课)：状态模式(State pat

Python自动化开发基础 装饰器-异常处理-面向对象编程

C语言系列之 进程通讯与相关设计模式

智能运动手环

达内的

全息投影手机

web前端工程

宝马3系促销

ios工程师月薪

小米运动手环

查看评论

暂无评论

您还没有登录,请[\[登录\]](#)或[\[注册\]](#)

\* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目

全部主题

HadoopAWS移动游戏JavaAndroidiOSSwift智能硬件DockerOpenStackVPNSparkERPIE10EclipseCRMJavaScript数据库UbuntuNFCWAPjQueryBIHTML5SpringApache.NETAPIHTMLSDKIISFedoraXMLLBSUnitySplashtopUMLcomponentsWindows MobileRailsQEMUKDECassandraCloudStackFTCcoremailOPhoneCouchBase云计算iOS6RackspaceWeb AppSpringSideMaemoCompuware大数据aptechPerlTornadoRubyHibernateThinkPHPHBasePureSolrAngularCloud FoundryRedisScalaDjangoBootstrap

公司简介 | 招贤纳士 | 广告服务 | 银行汇款帐号 | 联系方式 | 版权声明 | 法律顾问 | 问题报告 | 合作伙伴 | 论坛反馈

网站客服

杂志客服

微博客服

webmaster@csdn.net

400-600-2320

北京创新乐知信息技术有限公司 版权所有 | 江苏乐知网络技术有限公司 提供商务支持

京 ICP 证 09002463 号 | Copyright © 1999-2014, CSDN.NET, All Rights Reserved

关闭

5 of 5

16/2/29 上午12:21