

公告

总访问量：

昵称:菜鸟奋斗史
园龄:4年5个月
粉丝:131
关注:23
+加关注

<	2018年12月											>
日	一	二	三	四	五	六						
25	26	27	28	29	30	1						
2	3	4	5	6	7	8						
9	10	11	12	13	14	15						
16	17	18	19	20	21	22						
23	24	25	26	27	28	29						
30	31	1	2	3	4	5						

搜索

找找看

谷歌搜索

常用链接

我的随笔
我的评论
我的参与
最新评论
我的标签

随笔分类(161)

algorithm(4)
Git(1)
hadoop(1)
HTML+CSS+JavaScript(1)
Java NIO(2)
JavaSE(51)
Java编程思想(6)
Java并发编程(转载文章)(18)
Java单元测试(1)
Java设计模式(14)
JDBC(10)
linux(1)
mybatis(1)
MySQL数据库(3)
Servlet+JSP(4)
Spring(6)
SQL Server 2008(1)
Sqlite数据库(1)
SwordForOffer(1)
XML+AJAX(3)
操作系统(1)
工作总结(1)
计算机组成原理(1)
零碎代码(5)
数据结构(11)
项目练习(6)
小知识(5)
云计算(1)

随笔档案(150)

2018年11月 (1)
2018年9月 (1)
2018年4月 (1)
2017年6月 (2)
2017年2月 (1)
2016年11月 (2)
2016年10月 (26)
2016年9月 (8)
2016年8月 (1)
2016年6月 (1)
2016年5月 (19)
2016年4月 (38)
2016年3月 (46)
2015年10月 (3)

积分与排名

积分 - 128902
排名 - 2976

Java设计模式系列之中介者模式

中介者模式(Mediator)的定义

用一个中介对象来封装一系列的對象交互。中介者使各对象不需要显式地相互引用, 从而使其耦合松散, 而且可以独立地改变它们之间的交互。

中介者模式(Mediator)的适用性

- 1.一组对象以定义良好但是复杂的方式进行通信, 产生的相互依赖关系结构混乱且难以理解。
- 2.一个对象引用其他很多对象并且直接与这些对象通信,导致难以复用该对象。
- 3.想定制一个分布在多个类中的行为, 但又不想生成太多的子类。

中介者模式(Mediator)的参与者

1.Mediator

中介者定义一个接口用于与各同事(Colleague)对象通信。

2.ConcreteMediator

具体中介者通过协调各同事对象实现协作行为, 了解并维护它的各个同事。

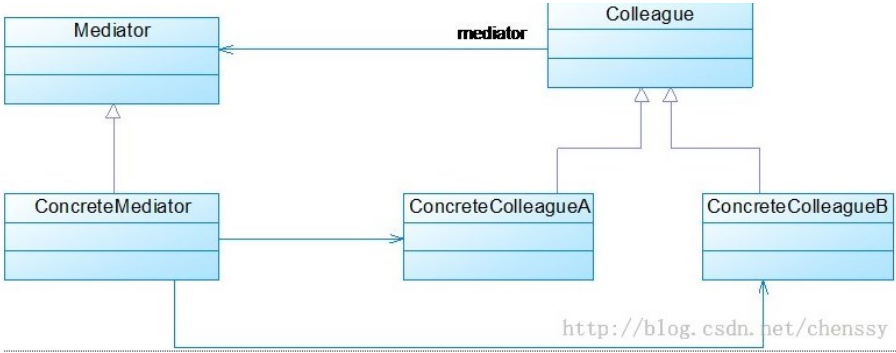
3.Colleague:

抽象同事类。

4.Colleagueclass

具体同事类。每个具体同事类都只需要知道自己的行为即可, 但是他们都需要认识中介者

中介者模式的UML类图



具体代码实现：

第一步:定义Mediator

```
1 //定义抽象Mediator,可以与同时们进行联络
2 public abstract class Mediator {
3     public abstract void contact(String content,Colleague coll);
4 }
```

第二步:定义抽象Colleague

```
1 public class Colleague {
2     protected String name;
3     protected Mediator mediator;
4
5     public Colleague(String name, Mediator mediator) {
6         this.name = name;
7         this.mediator = mediator;
8     }
9 }
```

第三步:定义具体Colleagueclass

```
1 public class ColleagueA extends Colleague {
2
3     // 具体同事类继承自Colleague,此刻就可以与中介者mediator进行通信了
4     public ColleagueA(String name, Mediator mediator) {
5         super(name, mediator);
6     }
7     public void getMessage(String message){
8         System.out.println("同事A"+name+"获得信息"+message);
9     }
10    //同事A与中介者通信
11    public void contact(String message){
12        mediator.contact(message, this);
13    }
14 }
15 public class ColleagueB extends Colleague {
```

最新评论

1. Re:JDBC学习笔记(4)——

PreparedStatement的使用

PreparedStatement说的很好,到位.值得收藏的宝贵资料.

--好男儿2017

2. Re:JDBC学习笔记(4)——

PreparedStatement的使用

写的好

--技术朱

3. Re:坦克大战(版本2.5-版本2.9)

能否打包源码发给小弟拜读, 谢谢啦

396125824@qq.com

--yczhang11

4. Re:JDBC学习笔记(10)——调用函数&

存储过程

看了一下, 哥们好像也是尚硅谷的课程啊

--蓝正满

5. Re:Spring学习笔记--Spring简介

博主写的非常好, 这里也整理了一个

spring系列文章, 欢迎多多交流:

--小知哥

阅读排行榜

1. Java单元测试初体验(JUnit4)

(59333)

2. JDBC学习笔记(4)——

PreparedStatement的使用(31543)

3. Java设计模式系列之状态模式

(21892)

4. Spring学习笔记--Spring简介

(16122)

5. Java设计模式系列之责任链模式

(14412)

评论排行榜

1. 坦克大战(版本2.5-版本2.9)(8)

2. 坦克大战(版本1.7-版本2.4)(4)

3. 坦克大战(版本1.0-版本1.6)(4)

4. Java设计模式系列之状态模式(4)

5. 剑指offer题目java实现(3)

推荐排行榜

1. 坦克大战(版本2.5-版本2.9)(8)

2. Java单元测试初体验(JUnit4)(6)

3. JDBC学习笔记(4)——

PreparedStatement的使用(5)

4. 坦克大战(版本1.7-版本2.4)(4)

5. 坦克大战(版本0.1-版本0.9)(4)

```
16
17     public ColleagueB(String name, Mediator mediator) {
18         super(name, mediator);
19     }
20     public void getMessage(String message) {
21         System.out.println("同事B"+name+"获得信息"+message);
22     }
23     //同事B与中介者通信
24     public void contact(String message) {
25         mediator.contact(message, this);
26     }
27 }
```

第四步:定义具体中介者ConcreteMediator,具体中介者通过协调各同事对象实现协作行为,了解并维护它的各个同事。

```
1 //定义具体中介者ConcreteMediator,具体中介者通过协调各同事对象实现协作行为,了解并维护它的各个同事。
2 public class ConcreteMediator extends Mediator {
3     ColleagueA collA;
4     ColleagueB collB;
5
6     public ColleagueA getCollA() {
7         return collA;
8     }
9
10    public void setCollA(ColleagueA collA) {
11        this.collA = collA;
12    }
13
14    public ColleagueB getCollB() {
15        return collB;
16    }
17
18    public void setCollB(ColleagueB collB) {
19        this.collB = collB;
20    }
21
22    @Override
23    public void contact(String content, Colleague coll) {
24        if (coll==collA) {
25            collB.getMessage(content);
26        } else {
27            collA.getMessage(content);
28        }
29    }
30 }
```

第五步:定义Client,测试中介者模式的使用

```
1 public class Client {
2
3     /**
4      * @param args
5      */
6     // 中介者, ColleagueA, ColleagueB
7     public static void main(String[] args) {
8         // 定义中介者
9         ConcreteMediator mediator = new ConcreteMediator();
10        // 定义具体同事类
11        ColleagueA colleagueA = new ColleagueA("张三", mediator);
12        ColleagueB colleagueB = new ColleagueB("李四", mediator);
13        // 中介者知晓每一个具体的Colleague类
14        mediator.setCollA(colleagueA);
15        mediator.setCollB(colleagueB);
16        colleagueA.contact("我是A,我要和同事B说说工作的事情");
17        colleagueB.contact("我是B,我下午有时间,下午商量吧");
18    }
19
20 }
```

运行结果:

```
同事B李四获得信息:我是A,我要和同事B说说工作的事情
同事A张三获得信息:我是B,我下午有时间,下午商量吧
```

总结:

中介者就是一个处于众多对象中间,并恰当地处理众多对象之间相互之间的联系的角色。以上代码中只有两个参与者类,但是这些我们都可以根据中介者模式的宗旨进行适当地扩展,即增加参与者类,然后中介者就得担负更加重的任务了,我们看到上面具体中介者类Mediator中的方法比较多而且有点乱。所以,在解耦参与者类之间的联系的同时,中介者自身也不免任务过重,因为几乎所有的业务逻辑都交代到中介者身上了,可谓是“万众期待”的一个角色了。这就是中介者模式的不足之处了。此外,上面这个代码例子的参与者的属性和方法都是一样的,我们可以抽取一个抽象类出来,减少代码,但是有时候我们根本抽取不了多个“参与者”之间的共性来形成一个抽象类,这也大大增加了中介者模式的使用难度。

分类: [Java设计模式](#)



菜鸟奋斗史

关注 - 23

粉丝 - 131

+加关注

« 上一篇:Java设计模式系列之命令模式
> 下一篇:(转载)Java之外观模式 (Facade Pattern)

0


 推荐

0

 反对

posted @ 2016-04-20 19:09 菜鸟奋斗史 阅读(5524) 评论(0) 编辑 收藏

[刷新评论](#) [刷新页面](#) [返回顶部](#)

 注册用户登录后才能发表评论, 请 [登录](#) 或 [注册](#), [访问网站首页](#)。

相关博文:

- [哈希\(不可逆\)加密通用类库函数](#)
- [别人的签名](#)
- [例外管理与意料之中的常例](#)
- [ASP.NET中如何防范SQL注入式攻击](#)
- [算法:移动中的碰撞侦测](#)

最新新闻:

- [B站发布2018年度弹幕:“前方高能”排年度热词第二](#)
 - [日本智能手机销售排行榜:苹果iPhone占据前三甲](#)
 - [中国年度航天发射次数首次超美国, 马斯克发推特感叹](#)
 - [荣耀总裁赵明:全球化手机品牌最后只会剩四五家](#)
 - [2018年度趋势:崩溃](#)
- > [更多新闻...](#)