

设计模式

设计模式
设计模式简介

工厂模式
抽象工厂模式

单例模式

建造者模式

原型模式

适配器模式

桥接模式

过滤器模式

组合模式

装饰器模式

外观模式

享元模式

代理模式

责任链模式

命令模式

解释器模式

迭代器模式

中介者模式

备忘录模式

观察者模式

状态模式

空对象模式

策略模式

模板模式

访问者模式

MVC 模式

← 迭代器模式

备忘录模式 →

中介者模式

中介者模式（Mediator Pattern）是用来降低多个对象和类之间的通信复杂性。这种模式提供了一个中介类，该类通常处理不同类之间的通信，并支持松耦合，使代码易于维护。中介者模式属于行为型模式。

介绍

意图：用一个中介对象来封装一系列的对象交互，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。

主要解决：对象与对象之间存在大量的关联关系，这样势必会导致系统的结构变得很复杂，同时若一个对象发生改变，我们也需要跟踪与之相关联的对象，同时做出相应的处理。

何时使用：多个类相互耦合，形成了网状结构。

如何解决：将上述网状结构分离为星型结构。

关键代码：对象 Colleague 之间的通信封装到一个类中单独处理。

应用实例： 1、中国加入 WTO 之前是各个国家相互贸易，结构复杂，现在是各个国家通过 WTO 来互相贸易。 2、机场调度系统。 3、MVC 框架，其中C（控制器）就是 M（模型）和 V（视图）的中介者。

优点： 1、降低了类的复杂度，将一对多转化成了一对一。 2、各个类之间的解耦。 3、符合迪米特原则。

缺点：中介者会庞大，变得复杂难以维护。

使用场景： 1、系统中对象之间存在比较复杂的引用关系，导致它们之间的依赖关系结构混乱而且难以复用该对象。 2、想通过一个中间类来封装多个类中的行为，而又不想生成太多的子类。

注意事项：不应当在职责混乱的时候使用。

实现

我们通过聊天室实例来演示中介者模式。实例中，多个用户可以向聊天室发送消息，聊天室向所有的用户显示消息。我们将创建两个类 *ChatRoom* 和 *User*。*User* 对象使用 *ChatRoom* 方法来分享他们的消息。
MediatorPatternDemo，我们的演示类使用 *User* 对象来显示他们之间的通信。

分类导航

HTML / CSS

JavaScript

服务端

数据库

数据分析

移动端

XML 教程

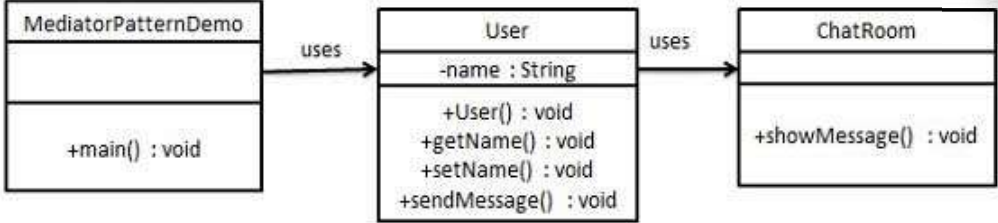
ASP.NET

Web Service

开发工具

网站建设





步骤 1

创建中介类。

ChatRoom.java

```
import java.util.Date;

public class ChatRoom {
    public static void showMessage(User user, String message){
        System.out.println(new Date().toString()
            + " [" + user.getName() +"] : " + message);
    }
}
```

步骤 2

创建 user 类。

User.java

```
public class User {
    private String name;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public User(String name){
        this.name = name;
    }

    public void sendMessage(String message){
        ChatRoom.showMessage(this,message);
    }
}
```

步骤 3

使用 *User* 对象来显示他们之间的通信。

MediatorPatternDemo.java

```
public class MediatorPatternDemo {
    public static void main(String[] args) {
        User robert = new User("Robert");
        User john = new User("John");

        robert.sendMessage("Hi! John!");
        john.sendMessage("Hello! Robert!");
    }
}
```

```
}  
}
```

步骤 4

执行程序，输出结果：

```
Thu Jan 31 16:05:46 IST 2013 [Robert] : Hi! John!  
Thu Jan 31 16:05:46 IST 2013 [John] : Hello! Robert!
```

← 迭代器模式

备忘录模式 →



4 篇笔记



写笔记

在线实例

- HTML 实例
- CSS 实例
- JavaScript 实例
- Ajax 实例
- jQuery 实例
- XML 实例
- Java 实例

字符集&工具

- HTML 字符集设置
- HTML ASCII 字符集
- JS 混淆/加密
- PNG/JPEG 图片压缩
- HTML 拾色器
- JSON 格式化工具
- 随机数生成器

最新更新

- Vue3 创建单文件...
- Vue3 指令
- Matplotlib imre...
- Matplotlib imsa...
- Matplotlib imsh...
- Matplotlib 直方图
- Python object(...

站点信息

- 意见反馈
- 免责声明
- 关于我们
- 文章归档

关注微信



^

QR code

★