导航
博客园
首页
新随笔
联系
订阅
管理

公告
昵称：ejiyuan
园龄：9年3个月
粉丝：177
关注：2
+加关注

< 2016年2月 >

日	一	二	三	四	五	六
31	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	1	2	3	4	5
6	7	8	9	10	11	12

搜索
找找看

常用链接
我的随笔
我的评论
我的参与
最新评论
我的标签

最新随笔
1. Entity Framework 中的 Code First 中引入数据库函数
2. .Net下实现可扩展的编程方法简述
3. 说说数据库范式
4. 设计模式学习总结-访问者模式（Visitor Method)
5. 设计模式学习总结-备忘录模式（Memento Method)
6. 设计模式学习总结-中介者模式（Mediator Method)
7. 设计模式学习总结-解释器模式（Interpreter Method)
8. 设计模式学习总结-享元模式（Flyweight Method)
9. 设计模式学习总结-责任链模式（Chain of Responsibility Method)
10. 设计模式学习总结-状态模式（State Method)

随笔分类
ASP.NET技术(61)
C#技术(26)
HTML(2)
JavaScript技术(16)
WCF(6)
工具软件(7)
软件下载(3)
设计模式(24)
数据库技术(10)

设计模式学习总结-命令模式（Cmmand Pattern)

Posted on 2012-06-28 17:02 ejiyuan 阅读(2147) 评论(3) 编辑 收藏

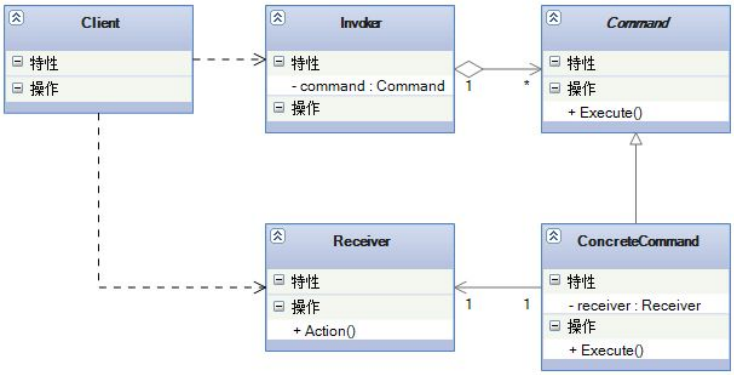
问题：
在面向对象的软件设计中，经常会遇到一个（或一系列）对象，对象本身的数据存储与对象的操作耦合在一起。例如一个对象有add(),edit(),delete()方法，这样对象支持的方法很难扩展，如果需要加入update()就必须修改代码，客户端与对象也是紧耦合的。命令模式是将一类对象的功能（行为，功能）抽象成一个命令对象，客户端在使用的时候，只与该命令对象打交道，而不用与对象打交道，分离命令的请求者和命令的执行者，降低了耦合性，可以使用不同的请求对客户进行参数化提高了程序设计的灵活性。

定义：
命令模式（Command）模式，将一个请求封装为一个对象，从而使你可用不同的请求对客户进行参数化；对请求排队或记录请求日志，以及支持可撤消的操作。
意图：

提供一个抽象的Command接口，将执行命令操作的方法封装到Command类接口中，ConcreteCommand实现这个Command接口方法，通过调用Receiver实例变量处理请求。客户端定义一个Invoker对象存储该concreteCommand对象,该invoker通过调用command对象的递交一个请求。

参与者：
•抽象命令角色（Command）：
定义命令的接口，声明执行的方法。
具体命令角色（ConcreteCommand）：
命令接口实现对象，是"虚"的实现；通常会持有接收者，并调用接收者的功能来完成命令要执行的操作。
•请求者（Invoker）：
要求命令对象执行请求，通常会持有命令对象，可以持有很多的命令对象。这个是客户端真正触发命令并要求命令执行相应操作的地方，也就是说相当于使用命令对象的入口。
•接收者（Receiver、执行者）：
接收者，真正执行命令的对象。任何类都可能成为一个接收者，只要它能够实现命令要求实现的相应功能。
•客户端（Client）：
创建具体的命令对象，并且设置命令对象的接收者。注意这个不是我们常规意义上的客户端，而是在组装命令对象和接收者，或许，把这个Client称为装配者会更好理解，因为真正使用命令的客户端是从Invoker来触发执行。

UML图：



实例说明：

诺基亚手机工厂
公司（Client）通知生产部（Invoker），生产两部n8,两部n9,生产部通过"命令模式"，将生产任务交给手机工厂（Receiver），生产手机。

uml图如下：

网络杂文(10)

随笔档案

2016年2月 (1)
2012年12月 (1)
2012年9月 (1)
2012年8月 (3)
2012年7月 (13)
2012年6月 (8)
2012年2月 (1)
2011年9月 (1)
2010年11月 (1)
2010年10月 (1)
2010年6月 (2)
2010年5月 (7)
2010年4月 (7)
2010年3月 (2)
2009年12月 (2)
2009年11月 (1)
2009年10月 (1)
2009年9月 (4)
2009年8月 (2)
2009年7月 (4)
2009年6月 (3)
2009年5月 (16)
2009年4月 (3)
2009年3月 (1)
2009年2月 (2)
2009年1月 (2)
2008年11月 (1)
2008年10月 (4)
2008年9月 (6)
2008年8月 (5)
2008年6月 (1)
2008年5月 (2)
2008年3月 (8)
2008年2月 (3)
2008年1月 (4)
2007年12月 (2)
2007年11月 (11)
2007年10月 (14)
2007年9月 (8)
2007年8月 (3)
2007年7月 (1)
2007年6月 (3)
2007年4月 (4)
2007年3月 (8)
2007年1月 (2)
2006年12月 (1)
2006年11月 (14)

文章分类

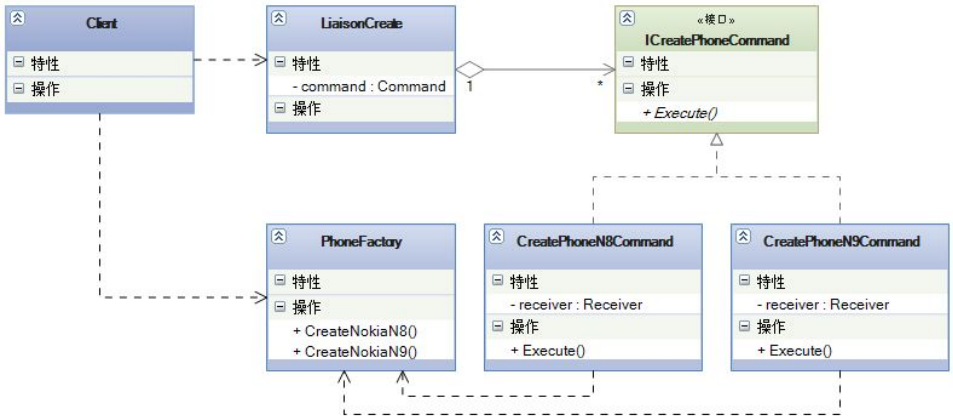
ASP.NET技术
JavaScript技术

文章档案

2007年5月 (1)

成长的足迹

ASP.NET 2.0入门教程
B# .NET Technical Community Homepage
C#多线程之三：解决多线程编程中大并发数等待唤醒的问题
codeproject
codeproject
CSDN_ASP.NET交流论坛
dynamicdrive
flash特效网



代码：

```
/// <summary>
/// 手机生产命令接口（Command）
/// </summary>
public interface ICreatePhoneCommand
{
    void Execute();
}
/// <summary>
/// N8手机生产具体命令类（ConcreteCommand）
/// </summary>
public class CreateNokiaN8Command : ICreatePhoneCommand
{
    PhoneFactory phoneFactory = null;

    public CreateNokiaN8Command(PhoneFactory _phoneFactory)
    {
        phoneFactory = _phoneFactory;
    }
    public void Execute()
    {
            phoneFactory.CreateNokiaN8();
    }
}
/// <summary>
/// N8手机生产具体命令类（ConcreteCommand）
/// </summary>
public class CreateNokiaN9Command : ICreatePhoneCommand
{
    PhoneFactory phoneFactory = null;
    public CreateNokiaN9Command(PhoneFactory _phoneFactory)
    {
        phoneFactory = _phoneFactory;
    }
    public void Execute()
    {
        phoneFactory.CreateNokiaN9();
    }
}
/// <summary>
/// 手机生产工厂（Receiver）具体的手机生产
/// </summary>
public class PhoneFactory
{
    public void CreateNokiaN8()
    {
        System.Console.WriteLine("一部Nokia N8 生产完成");
    }
    public void CreateNokiaN9()
    {
        System.Console.WriteLine("一部Nokia N9 生产完成");
    }
}
/// <summary>
/// 生产部对象（Invoker）接收生产信息，制定生产清单。通知PhoneFactory生产
/// </summary>
public class LiaisonCreate
{
    List<ICreatePhoneCommand> createPhoneCommandList = new List<ICreatePhoneCommand>();
    /// <summary>
    /// 添加生产任务
```

```
    /// </summary>
    /// <param name="_createPhoneCommand"></param>
    public void AddCreatePhoneTask(ICreatePhoneCommand _createPhoneCommand)
    {
        createPhoneCommandList.Add(_createPhoneCommand);
    }
    /// <summary>
    /// 撤销生产任务
    /// </summary>
    /// <param name="_createPhoneCommand"></param>
    public void CancelCreatePhoneTask(ICreatePhoneCommand _createPhoneCommand)
    {
        createPhoneCommandList.Remove(_createPhoneCommand);
    }
    /// <summary>
    /// 执行生产
    /// </summary>
    public void CreatePhone()
    {
        foreach (var createPhoneCommand in createPhoneCommandList)
        {
            createPhoneCommand.Execute();
        }
    }
}
public  void CommandTest()
{
    //初始化生产部联系人
    LiaisonCreate liaisonCreate = new LiaisonCreate();
    //初始化生产工厂
    PhoneFactory phoneFactory = new PhoneFactory();
    //设置生产清单
    liaisonCreate.AddCreatePhoneTask(new CreateNokiaN8Command(phoneFactory));
    liaisonCreate.AddCreatePhoneTask(new CreateNokiaN8Command(phoneFactory));
    liaisonCreate.AddCreatePhoneTask(new CreateNokiaN9Command(phoneFactory));
    liaisonCreate.AddCreatePhoneTask(new CreateNokiaN9Command(phoneFactory));
    //取消一部N9的生产
    liaisonCreate.CancelCreatePhoneTask(new CreateNokiaN9Command(phoneFactory));

    //开始执行生产
    liaisonCreate.CreatePhone();
    System.Console.Read();
}
```

优点:
•命令模式将发出命令的责任和执行命令的责任分割开，降低系统的耦合度。
•新的命令可以很容易地加入到系统中。只要实现了抽象命令接口的具体命令类就可以与接收者相关联。
•可以比较容易地设计一个组合命令，形成一个轻量级的事件队列
•命令模式使请求本身成为一个对象，这个对象和其他对象一样可以被存储和传递。
•请求方不必知道接收请求的接口，执行命令的细节（只需客户端为concreteCommand对象指定一个receiver对象即可）起到了很好的封装隔离作用.

缺点:

•每一个命令都需要设计一个具体命令类，使用命令模式会导致某些系统有过多的具体命令类。

应用情景:
•系统需要将请求调用者和请求接收者解耦，使得调用者和接收者不直接交互。
•系统需要在不同的时间指定请求、将请求排队和执行请求。
•系统需要支持命令的撤销(Undo)操作和恢复(Redo)操作。
•系统需要将一组操作组合在一起，即支持宏命令。

分类: 设计模式

« 上一篇: 设计模式学习总结-迭代器模式（Iterator Pattern）
» 下一篇: 设计模式学习总结-适配器模式（Adapter Pattern）

Feedback

#1楼 回复 引用

ejiyuan,您好！我浏览过您发的反编译文章，十分感兴趣。我向您发了一个信息求教一个问题，烦请及时回复。谢谢！

--luoli519

5. Re:关于LinQ的动态Or查询

感谢大神。

--影像。

2012-06-28 17:20 by HU9HJI1

学习了.不错.

支持(0) 反对(0)

#2楼 回复 引用

2012-06-28 17:53 by 「初见」

也学习了、赞一个、

支持(0) 反对(0)

#3楼 回复 引用

2013-02-22 11:03 by 风影极光

想问楼主的UML图使用的哪个工具绘制的?

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

发表评论

昵称: Aldridge1

评论内容:

提交评论 注销 订阅评论

[Ctrl+Enter快捷键提交]