

您的潜力，我们的动力

**Microsoft**  
微软(中国)有限公司

# C#面向对象设计模式纵横谈

## 9. Composite 组合（结构型模式）

李建忠

[jianzhong.lee@gmail.com](mailto:jianzhong.lee@gmail.com)

设计模式论坛:

[forum.softcompass.com](http://forum.softcompass.com)

上海祝成科技 高级培训讲师

[www.softcompass.com](http://www.softcompass.com)



**MSDN Webcasts**

您的潜力, 我们的动力

**Microsoft**  
微软(中国)有限公司



组团给大家拜年J 从俄罗斯套娃谈起...



**MSDN Webcasts**

# 对象容器的问题

在面向对象系统中，我们常会遇到一类具有“容器”特征的对象——即它们在充当对象的同时，又是其他对象的容器。

```
public class SingleBox: IBox {  
    public void process() { .....}  
}
```

```
public class ContainerBox :IBox {  
    public void process(){.....}  
    public ArrayList getBoxes(){.....}  
}
```

如果我们要对这样的对象容器进行处理：

```
IBox box=Factory.GetBox();  
If (box is ContainerBox){  
    box.process();  
    ArrayList list= ((ContainerBox) box).GetBoxes();  
    ..... // 将面临比较复杂的递归处理  
}else if( box is SingleBox){  
    box.process();  
}
```



# 动机 (Motivation)

上述描述的问题根源在于：客户代码过多地依赖于对象容器复杂的内部实现结构，对象容器内部实现结构（而非抽象接口）的变化将引起客户代码的频繁变化，带来了代码的维护性、扩展性等弊端。

如何将“客户代码与复杂的对象容器结构”解耦？让对象容器自己来实现自身的复杂结构，从而使得客户代码就像处理简单对象一样来处理复杂的对象容器？

# 意图 (Intent)

将对象组合成树形结构以表示“部分-整体”的层次结构。**Composite**使得用户对单个对象和组合对象的使用具有一致性。

——《设计模式》GoF

# 例说Composite应用

您的潜力，我们的动力

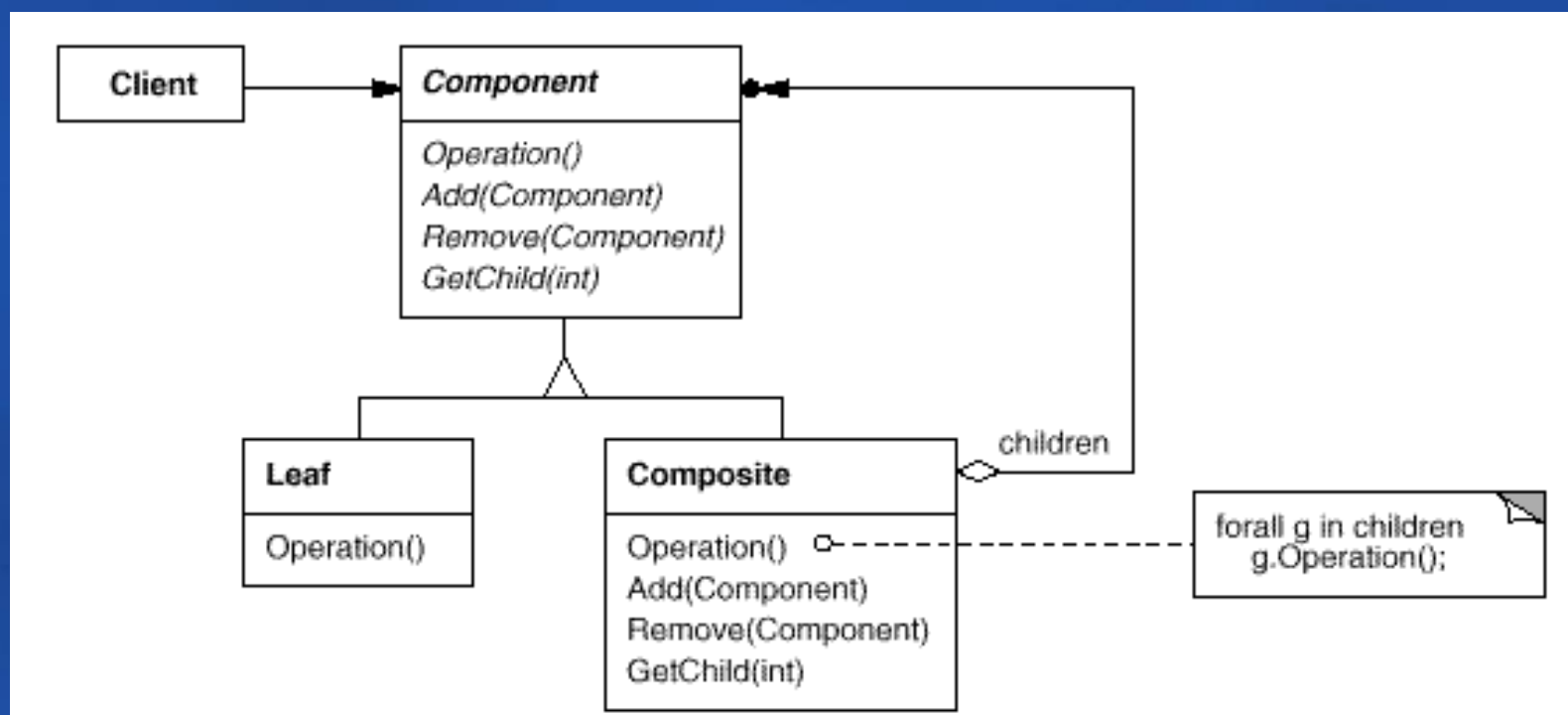
**Microsoft**  
微软(中国)有限公司

## Codes in VS.NET



**MSDN Webcasts**

# 结构 (Structure)





# Composite模式的几个要点

- **Composite**模式采用树形结构来实现普遍存在的对象容器，从而将“一对多”的关系转化为“一对一”的关系，使得客户代码可以一致地处理对象和对象容器，无需关心处理的是单个的对象，还是组合的对象容器。
- 将“客户代码与复杂的对象容器结构”解耦是**Composite**模式的核心思想，解耦之后，客户代码将与纯粹的抽象接口——而非对象容器的复内部实现结构——发生依赖关系，从而更能“应对变化”。
- **Composite**模式中，是将“Add和Remove等和对象容器相关的方法”定义在“表示抽象对象的**Component**类”中，还是将其定义在“表示对象容器的**Composite**类”中，是一个关乎“透明性”和“安全性”的两难问题，需要仔细权衡。这里有可能违背面向对象的“单一职责原则”，但是对于这种特殊结构，这又是必须付出的代价。**ASP.NET**控件的实现在这方面为我们提供了一个很好的示范。
- **Composite**模式在具体实现中，可以让父对象中的子对象反向追溯；如果父对象有频繁的遍历需求，可使用缓存技巧来改善效率。



您的潜力，我们的动力

**Microsoft**  
微软(中国)有限公司

# .NET框架中的Composite应用

## ASP.NET控件结构分析




**MSDN Webcasts**

# 推荐资源

- 《设计模式：可复用面向对象软件的基础》 GoF
- 《面向对象分析与设计》 Grady Booch
- 《敏捷软件开发：原则、模式与实践》 Robert C. Martin
- 《重构：改善既有代码的设计》 Martin Fowler
- 《Refactoring to Patterns》 Joshua Kerievsky



# Question & Answer

如需提出问题，请单击“提问”按钮并在随后显示的浮动面板中输入问题内容。一旦完成问题输入后，请单击“提问”按钮。

 **问题和解答 (无问题)** ▲ ×

在此会议中尚未解答任何问题。

要向演示者提问，请在此处键入问

提问(A)

删除(D)

问题管理器(Q)

您的潜力，我们的动力

**Microsoft®**  
微软(中国)有限公司

**Microsoft®**

**msdn**  


**MSDN Webcasts**