

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

3. Abstract Factory 抽象工厂（创建型模式）

李建忠

www.lijianzhong.com

上海祝成科技 高级培训讲师

new的问题

常规的对象创建方法：

```
// 创建一个Road 对象  
Road road=new Road();
```

new的问题：

- 实现依赖，不能应对“具体实例化类型”的变化。

解决思路：

- 封装变化点 —— 哪里变化，封装哪里
- 潜台词：如果没有变化，当然不需要额外的封装！

工厂模式的缘起

- 变化点在“对象创建”，因此就封装“对象创建”
- 面向接口编程——依赖接口，而非依赖实现
- 最简单的解决方法：

```
class RoadFactory {  
    public static Road CreateRoad()  
    {  
        return new Road();  
    }  
}
```

```
// 创建一个Road 对象  
Road road=  
    roadFactory.CreateRoad();
```

创建一系列相互依赖的对象

假设一个游戏开发场景：

我们需要构造“道路”、
“房屋”、“地道”、“丛林”……等等对象

```
Road road=  
roadFactory.CreateRoad(  
);  
...
```

```
Building building=  
roadFactory.CreateBuildi  
ng();  
...
```

```
class RoadFactory {  
    public static Road CreateRoad()  
    {  
        return new Road();  
    }  
    public static Building  
CreateBuilding ()  
    {  
        return new Building();  
    }  
    public static Tunnel  
CreateTunnel ()  
    {  
        return new Tunnel();  
    }  
    public static Jungle  
CreateJungle()  
    {  
        return new Jungle();  
    }  
}
```

简单工厂的问题

简单工厂的问题：

- 不能应对“不同系列对象”的变化。比如有不同风格的游戏场景——对应不同风格的道路、房屋、地道……

如何解决：

- 使用面向对象的技术来“封装”变化点

动机 (Motivation)

在软件系统中，经常面临着“一系列相互依赖的对象”的创建工作；同时，由于需求的变化，往往存在更多系列对象的创建工作。

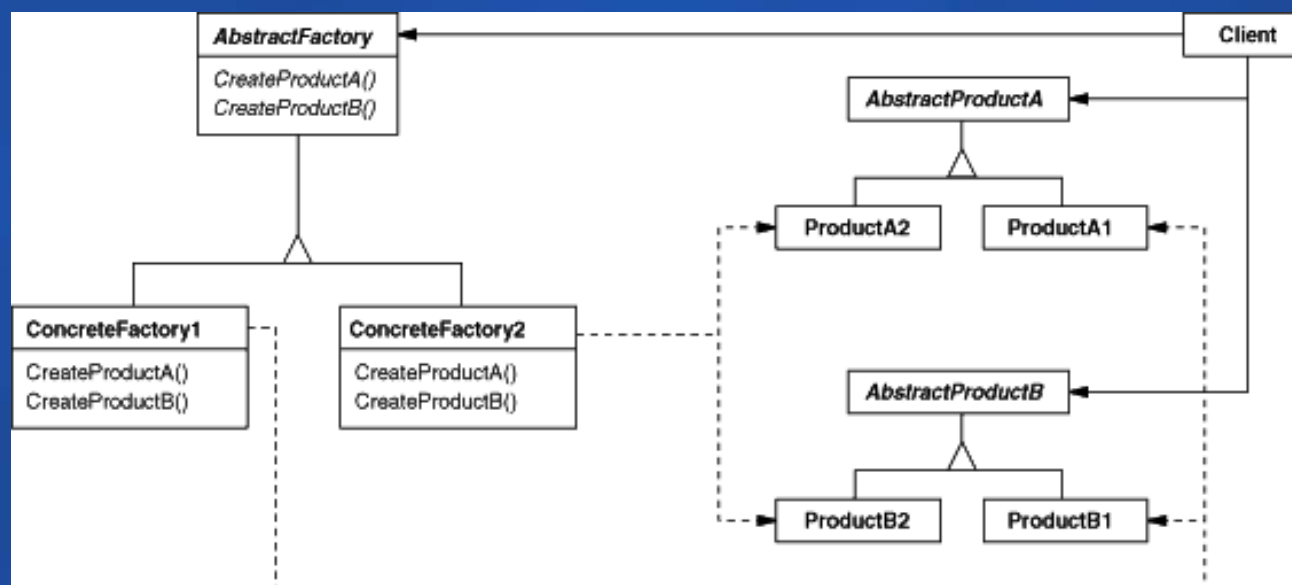
如何应对这种变化？如何绕过常规的对象创建方法(new)，提供一种“封装机制”来避免客户程序和这种“多系列具体对象创建工作”的紧耦合？

意图 (Intent)

提供一个接口，让该接口负责创建一系列“相关或者相互依赖的对象”，无需指定它们具体的类。

——《设计模式》GoF

结构 (Structure)



您的潜力，我们的动力

Microsoft
微软(中国)有限公司

游戏框架中的Abstract Factory应用

Codes in VS.NET

Abstract Factory模式的几个要点

- 如果没有应对“多系列对象构建”的需求变化，则没有必要使用Abstract Factory模式，这时候使用简单的静态工厂完全可以。
- “系列对象”指的是这些对象之间有相互依赖、或作用的关系，例如游戏开发场景中的“道路”与“房屋”的依赖，“道路”与“地道”的依赖。
- Abstract Factory模式主要在于应对“新系列”的需求变动。其缺点在于难以应对“新对象”的需求变动。
- Abstract Factory模式经常和Factory Method模式共同组合来应对“对象创建”的需求变化。

您的潜力，我们的动力

Microsoft
微软(中国)有限公司

.NET框架中的Abstract Factory应用


Codes in VS.NET

推荐参考书

- 《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