# 工厂模式

## **关于工厂方法模式的误会**

不太在意设计模式的同事会对工厂模式有极深的误解，总会把“工厂模式”与“静态工厂方法”混为一谈，什么是静态工厂方法？看一个简单的例子：

public class SimpleClientFactory {

public static Client createClient(){

return new Client();

}

}

通过一个静态方法来创建实例，这种方式在代码中比较常见，但这并不是我们今天要说的工厂模式，它只是一个“静态工厂方法”。   
个人觉得很难给模式一个语句上的定义，因为这些模式本身只是一些帮助我们养成好的代码习惯的一些建议，它们甚至算不上是一种规范。对于工厂模式，我觉得某一段定义说的是比较准确的。

****工厂模式：父类定义了创建对象的接口，但是由子类来具体实现，工厂方法让类把实例化的动作推迟到了子类当中。****

****（1） 保证了创建Bean的方式的多样性****   
Bean工厂有很多种，它们负责创建各种各样不同的Bean，比如Map类型的Bean，List类型的Bean，Web服务Bean，子类们不需要关心单例或非单例情况下是否需要额外操作，只需要关心如何创建Bean，并且创建出来的Bean是多种多样的。

****（2） 严格规定了Bean创建前后的其它动作****   
虽然子类可以自由的去创建Bean，但是创建Bean之前的准备工作以及创建Bean之后对Bean的处理工作是AbstractFactoryBean设定好了的，子类不需要关心，也没权力关心，在这个例子中父类只负责一些前置判断工作。

工厂方法模式非常的有趣，它给了子类创建实例的自由，又严格的规定了实例创建前后的业务流程。

## **依赖倒置原则**

工厂方法模式非常好的诠释了面向对象六大设计原则之一的依赖倒置原则：要依赖抽象，不要依赖具体类。

按理说，高层组件应该依赖于低层组件，低层组件为高层组件提供一些最基础的服务，但是工厂模式倒置了这一依赖现象，让低层组件反而要依赖于统一的抽象接口。   
****工厂模式让高层组件（WebCrawler）和低层组件（ATypeClient|BTypeClient|……）都依赖于共同的接口（HttpClient），这倒置了原本的依赖模型，解除了高层组件和低层组件之间的强依赖关系。****

****工厂模式分为：工厂方法模式（factory method）和抽象工厂模式（abstract factory）。****

****结论1：简单工厂通过构造时传入的标识来生产产品，不同产品都在同一个工厂中生产，这种判断会随着产品的增加而增加，给扩展和维护带来麻烦****

在工厂模式中，已经将工厂类分开，不再将所有产品在同一工厂中生产，解决了简单工厂中不断增加的switch case的问题。如再新增一个C产品，那么只需写一个C工厂和C产品，在调用时用C工厂生产C产品即可，A和B工厂和产品完全不受影响

****结论2：工厂模式无法解决产品族和产品等级结构的问题****

在工厂模式中，一个工厂生产一个产品，所有的具体产品是由同一个抽象产品派生的，不存在产品等级结构和产品族的概念；而在抽象工厂中，同一个等级的产品是派生于一个抽象产品（即产品接口），一个抽象工厂派生不同的具体工厂，每个具体工厂生产自己的产品族（包含不同产品等级）

****结论3：工厂模式中，一个工厂生产一个产品，所有产品派生于同一个抽象产品（或产品接口）；而抽象工厂模式中，一个工厂生产多个产品，它们是一个产品族，不同的产品族的产品派生于不同的抽象产品（或产品接口）****

## 三种工厂模式关键点

三种工厂的实现是越来越复杂的

简单工厂通过构造时传入的标识来生产产品，不同产品都在同一个工厂中生产，这种判断会随着产品的增加而增加，给扩展和维护带来麻烦

工厂模式无法解决产品族和产品等级结构的问题

抽象工厂模式中，一个工厂生产多个产品，它们是一个产品族，不同的产品族的产品派生于不同的抽象产品（或产品接口）