### 1 简介

工厂模式是常用的实例化类对象的设计模式，是new方法的取代品。工厂方法的使用可以增强程序的扩展性和维护性。

### 2 问题

在面向对象编程中, 最通常的方法是一个new操作符产生一个对象实例,new操作符就是用来构造对象实例的。但是在一些情况下, new操作符直接生成对象会带来一些问题。举例来说, 许多类型对象的创造需要一系列的步骤: 你可能需要计算或取得对象的初始设置; 选择生成哪个子对象实例; 或在生成你需要的对象之前必须先生成一些辅助功能的对象。 在这些情况, 新对象的建立就是一个 “过程”，不仅是一个操作，像一部大机器中的一个齿轮传动。  
问题  
你如何能轻松方便地建立这么" 复杂 " 的对象即操作中不需要粘贴复制呢?  
  
解决方法  
建立一个工厂（一个函数或一个类方法）来制造新的对象。

### 3 好处

A.应用程序的设计中把对象的的创建集中在一个地方创建或者统一由某类来管理（spring）

B.在不改动应用程序的基础上可以直接添加对象，同时也利于对象的维护。

### 4 具体介绍

##### 简单工厂（静态工厂）

###### 4.1.1 简单介绍

根据不同的参数创建不同的产品并返回。

###### 4.1.2 角色

1. 一个产品接口
2. N个实现产品接口的产品
3. 一个工厂方法，工厂方法根据参数生产产品

###### 4.1.3 优劣势

优点：责任分割，封装创建实例细节，简单易用，添加新的产品时不用修改已有的客户端代码。

缺点：1、我们每次想要增加一种新产品的时候，都必须修改SimpleFactory的原代码

2、当我们拥有很多很多产品的时候，而且产品之间又存在复杂的层次关系的时候，这个类必须拥有复杂的逻辑判断能力，其代码量也将不断 地激增，这对以后的维护简直就是恐怖两个字...

3、整个系统都严重依赖SimpleFactory类，只要SimpleFactory类一出问题，系统就进入不能工作的状态，这也是最为致命的一点....

###### 4.1.4 适用场景

1. 工厂类创建的对象较少
2. 客户端只知道传入工厂类的参数，不关心如何创建对象。

##### 4.2 工厂方法（多态工厂）

###### 4.2.1 简单介绍

###### 4.2.2 角色

1. 一个产品接口
2. 一个工厂接口
3. N个实现产品接口的产品
4. N个实现工厂接口、生产特定产品的实现类

###### 4.2.3 优劣

优点：责任分割，封装创建实例细节，多态性设计，符合开闭原则

缺点：当有数百种甚至上千种产品的时候，也必须要有对应的上百成千个工厂类，这就出现了传说的类爆炸，对于以后的维护来说，简直就是一场灾难.....

###### 4.2.4 适用场景

1. 动态创建
2. 不关心具体产品，只关心工厂创建的类的作用。

如：日志记录器，数据库访问

 日志记录器：记录可能记录到本地硬盘、系统事件、远程服务器等，用户可以选择记录日志到什么地方。 2、数据库访问，当用户不知道最后系统采用哪一类数据库，以及数据库可能有变化时。 3、设计一个连接服务器的框架，需要三个协议，"POP3"、"IMAP"、"HTTP"，可以把这三个作为产品类，共同实现一个接口。

##### 4.3 抽象工厂（工具箱）

###### 4.3.1 简单介绍

创建一系列相互依赖对象的接口，而无需指定具体的类。

###### 4.3.2 角色

1. 一个产品接口
2. 一个工厂接口
3. N个实现产品接口的产品
4. N个实现工厂接口、生产特定类产品的实现类

###### 4.2.3 优劣

优点：责任分割，封装创建对象细节，隔离具体类的生成，使用户不需要指定什么被创建，当一个产品族的多个对象被设计成一起工作时，它能够保证客户端始终使用同一个产品族中的对象

缺点：添加新的产品对象时，难以扩展新的抽象工厂以便生成新种类的产品。

###### 4.2.4 适用场景

产品有相互依赖关系时。如，一套衣服（上衣，裤子），整套qq皮肤。