

简单工厂模式 (SimpleFactoryPattern) - 最易懂的设计模式解析



Carson_Ho (/u/383970bef0a0) [+ 关注](#)

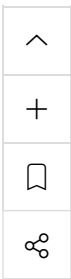
2016.08.16 16:47* 字数 1407 阅读 7226 评论 3 喜欢 29

(/u/383970bef0a0)



前言

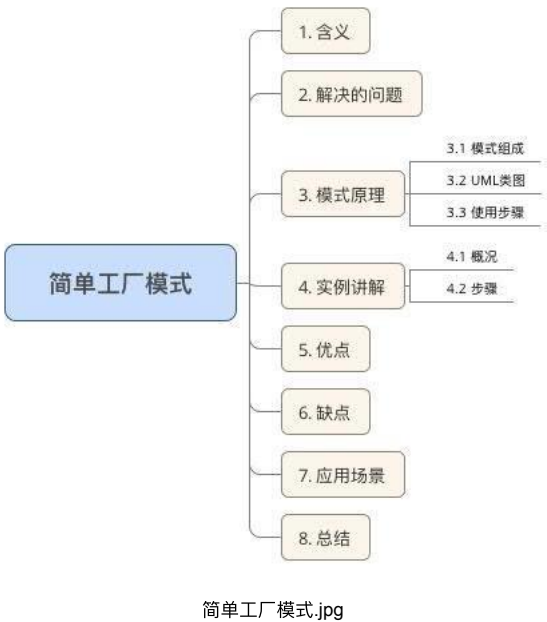
今天我来全面总结一下Android开发中最常用的设计模式 - 简单工厂模式。



其他设计模式介绍

- 1分钟全面了解“设计模式” (<https://www.jianshu.com/p/6e5eda3a51af>)
- 单例模式（Singleton） - 最易懂的设计模式解析 (<https://www.jianshu.com/p/b8c578b07fbc>)
- 简单工厂模式（SimpleFactoryPattern） - 最易懂的设计模式解析 (<https://www.jianshu.com/p/e55fbddc071c>)
- 工厂方法模式（Factory Method） - 最易懂的设计模式解析 (<https://www.jianshu.com/p/d0c444275827>)
- 抽象工厂模式（Abstract Factory） - 最易懂的设计模式解析 (<https://www.jianshu.com/p/7deb64f902db>)
- 策略模式（Strategy Pattern） - 最易懂的设计模式解析 (<https://www.jianshu.com/p/0c62bf587b9c>)
- 适配器模式（Adapter Pattern） - 最易懂的设计模式解析 (<https://www.jianshu.com/p/9d0575311214>)
- 代理模式（Proxy Pattern） - 最易懂的设计模式解析 (<https://www.jianshu.com/p/a8aa6851e09e>)
- 模板方法模式（Template Method） - 最易懂的设计模式解析 (<https://www.jianshu.com/p/a3474f4fee57>)
- 建造者模式（Builder Pattern） - 最易懂的设计模式解析 (<https://www.jianshu.com/p/be290ccea05a>)
- 外观模式（Facade Pattern） - 最易懂的设计模式解析 (<https://www.jianshu.com/p/1b027d9fc005>)

目录



1. 含义

- 简单工厂模式又叫静态方法模式（因为工厂类定义了一个静态方法）
- 现实生活中，工厂是负责生产产品的；同样在设计模式中，简单工厂模式我们可以理解为负责生产对象的一个类，称为“工厂类”。

2. 解决的问题

简单工厂模式 (SimpleFactoryPattern) - 最易懂的设计模式解析 - 简书

将“类实例化的操作”与“使用对象的操作”分开，让使用者不用知道具体参数就可以实例化出所需要的“产品”类，从而避免了在客户端代码中显式指定，实现了解耦。

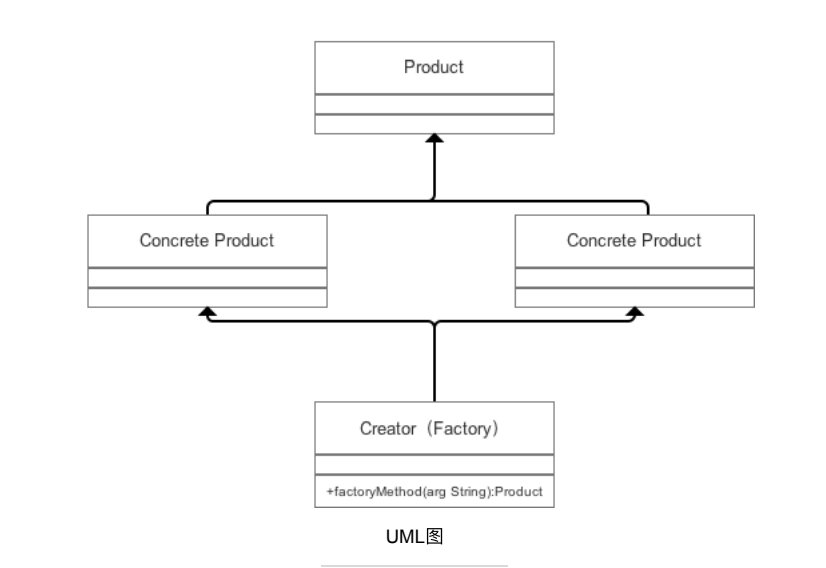
即使用者可直接消费产品而不需要知道其生产的细节

3. 模式原理

3.1 模式组成

组成（角色）	关系	作用
抽象产品（Product）	具体产品的父类	描述产品的公共接口
具体产品（Concrete Product）	抽象产品的子类；工厂类创建的目标类	描述生产的具体产品
工厂（Creator）	被外界调用	根据传入不同参数从而创建不同具体产品类的实例

3.2 UML类图



3.3 使用步骤

- 创建**抽象产品类** & 定义具体产品的公共接口；
- 创建**具体产品类**（继承抽象产品类） & 定义生产的具体产品；
- 创建**工厂类**，通过创建静态方法根据传入不同参数从而创建不同具体产品类的实例；
- 外界通过调用工厂类的静态方法，**传入不同参数从而创建不同具体产品类的实例**

4. 实例

接下来我用一个实例来对简单工厂模式进行更深一步的介绍。

4.1 实例概况

- 背景：小成有一个塑料生产厂，用来做塑料加工生意
- 目的：最近推出了3个产品，小成希望使用**简单工厂模式**实现3款产品的生产

4.2 使用步骤

实现代码如下：

^

+

🔖

🔗

步骤1. 创建抽象产品类，定义具体产品的公共接口

```
abstract class Product{
    public abstract void Show();
}
```

步骤2. 创建具体产品类（继承抽象产品类），定义生产的具体产品

```
// 具体产品类A
class ProductA extends Product{

    @Override
    public void Show() {
        System.out.println("生产出了产品A");
    }
}

// 具体产品类B
class ProductB extends Product{

    @Override
    public void Show() {
        System.out.println("生产出了产品C");
    }
}

// 具体产品类C
class ProductC extends Product{

    @Override
    public void Show() {
        System.out.println("生产出了产品C");
    }
}
```

步骤3. 创建工厂类，通过创建静态方法从而根据传入不同参数创建不同具体产品类的实例

```
class Factory {
    public static Product Manufacture(String ProductName){
        // 工厂类里用switch语句控制生产哪种商品；
        // 使用者只需要调用工厂类的静态方法就可以实现产品类的实例化。
        switch (ProductName){
            case "A":
                return new ProductA();

            case "B":
                return new ProductB();

            case "C":
                return new ProductC();

            default:
                return null;
        }
    }
}
```

步骤4. 外界通过调用工厂类的静态方法，传入不同参数从而创建不同具体产品类的实例

```
//工厂产品生产流程
public class SimpleFactoryPattern {
    public static void main(String[] args){
        Factory mFactory = new Factory();

        //客户要产品A
        try {
            //调用工厂类的静态方法 & 传入不同参数从而创建产品实例
            mFactory.Manufacture("A").Show();
        }catch (NullPointerException e){
            System.out.println("没有这一类产品");
        }

        //客户要产品B
        try {
            mFactory.Manufacture("B").Show();
        }catch (NullPointerException e){
            System.out.println("没有这一类产品");
        }

        //客户要产品C
        try {
            mFactory.Manufacture("C").Show();
        }catch (NullPointerException e){
            System.out.println("没有这一类产品");
        }

        //客户要产品D
        try {
            mFactory.Manufacture("D").Show();
        }catch (NullPointerException e){
            System.out.println("没有这一类产品");
        }
    }
}
```

结果输出：

生产出了产品A
生产出了产品C
生产出了产品C
没有这一类产品

5. 优点

- 将创建实例的工作与使用实例的工作分开，使用者不必关心类对象如何创建，实现了解耦；
- 把初始化实例时的工作放到工厂里进行，使代码更容易维护。更符合面向对象的原则 & 面向接口编程，而不是面向实现编程。

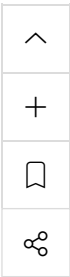
6. 缺点

- 工厂类集中了所有实例（产品）的创建逻辑，一旦这个工厂不能正常工作，整个系统都会受到影响；
- 违背“开放 - 关闭原则”，一旦添加新产品就不得不修改工厂类的逻辑，这样就会造成工厂逻辑过于复杂。
- 简单工厂模式由于使用了静态工厂方法，静态方法不能被继承和重写，会造成工厂角色无法形成基于继承的等级结构。

7. 应用场景

在了解了优缺点后，我们知道了简单工厂模式的应用场景：

- 客户如果只知道传入工厂类的参数，对于如何创建对象的逻辑不关心时；
- 当工厂类负责创建的对象（具体产品）比较少时。



8. 总结

本文主要对简单工厂模式进行了全面介绍，接下来将介绍工厂方法模式 & 其他设计模式，有兴趣可以继续关注Carson_Ho的安卓开发笔记
(https://www.jianshu.com/users/383970bef0a0/latest_articles)!!!!

请点赞！因为你的鼓励是我写作的最大动力！

相关文章阅读

单例模式（Singleton） - 最易懂的设计模式解析
(<https://www.jianshu.com/p/b8c578b07fbc>)
简单工厂模式（SimpleFactoryPattern） - 最易懂的设计模式解析
(<https://www.jianshu.com/p/e55fbddc071c>)
工厂方法模式（Factory Method） - 最易懂的设计模式解析
(<https://www.jianshu.com/p/d0c444275827>)
抽象工厂模式（Abstract Factory） - 最易懂的设计模式解析
(<https://www.jianshu.com/p/7deb64f902db>)
策略模式（Strategy Pattern） - 最易懂的设计模式解析
(<https://www.jianshu.com/p/0c62bf587b9c>)
适配器模式（Adapter Pattern） - 最易懂的设计模式解析
(<https://www.jianshu.com/p/9d0575311214>)
代理模式（Proxy Pattern） - 最易懂的设计模式解析
(<https://www.jianshu.com/p/a8aa6851e09e>)
模板方法模式（Template Method） - 最易懂的设计模式解析
(<https://www.jianshu.com/p/a3474f4fee57>)
建造者模式（Builder Pattern） - 最易懂的设计模式解析
(<https://www.jianshu.com/p/be290ccea05a>)
外观模式（Facade Pattern） - 最易懂的设计模式解析
(<https://www.jianshu.com/p/1b027d9fc005>)

欢迎关注Carson_Ho

(https://www.jianshu.com/users/383970bef0a0/latest_articles)的简书！

不定期分享关于安卓开发的干货，追求短、平、快，但却不缺深度。





小礼物走一走，来简书关注我

赞赏支持

设计模式 (/nb/5752111) 举报文章 © 著作权归作者所有



Carson_Ho (/u/383970bef0a0) 

写了 285959 字，被 24486 人关注，获得了 17159 个喜欢
(/u/383970bef0a0)

+ 关注

简书认证作者、CSDN签约作者、稀土掘金专栏作者 定位：分享 Android开发 干货 Github：https://github.c...



喜欢 | 29



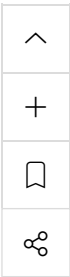
更多分享

(http://cwb.assets.jianshu.io/notes/images/4517690

被以下专题收入，发现更多相似内容

- + 收入我的专题
-  Android知识 (/c/3fde3b545a35?utm_source=desktop&utm_medium=notes-included-collection)
-  Android开发 (/c/d1591c322c89?utm_source=desktop&utm_medium=notes-included-collection)
-  Android... (/c/58b4c20abf2f?utm_source=desktop&utm_medium=notes-included-collection)
-  程序员 (/c/NEt52a?utm_source=desktop&utm_medium=notes-included-collection)
-  架构算法设计模... (/c/c568ddab391a?utm_source=desktop&utm_medium=notes-included-collection)
-  设计模式 (/c/b9af413630c3?utm_source=desktop&utm_medium=notes-included-collection)
-  Android开发 (/c/0dc880a2c73c?utm_source=desktop&utm_medium=notes-included-collection)

展开更多




(/p/d0c444275827?



utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)
工厂方法模式（Factory Method） - 最易懂的设计模式解析 (/p/d0c444275...

前言 在上文提到的最易懂的设计模式系列解析：简单工厂模式，发现简单工厂模式存在一系列问题： 工厂类集中了所有实例（产品）的创建逻辑，一旦这个工厂不能正常工作，整个系统都会受到影响； 违背“开放 - ...

 Carson_Ho (/u/383970bef0a0?


utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

(/p/7deb64f902db?



utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)
抽象工厂模式（Abstract Factory） - 最易懂的设计模式解析 (/p/7deb64f90...

前言 在上文提到的最易懂的设计模式系列解析：工厂方法模式，发现工厂方法模式存在一个严重的问题： 一个具体工厂只能创建一类产品 而在实际过程中，一个工厂往往需要生产多类产品。为了解决上述的问题， ...

 Carson_Ho (/u/383970bef0a0?


utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

(/p/be290ccea05a?



utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)
建造者模式（Builder Pattern） - 最易懂的设计模式解析 (/p/be290ccea05a...

前言 今天我来全面总结一下Android开发中最常用的设计模式 -建造者模式。 其他设计模式介绍1分钟全面了解“设计模式”单例模式（Singleton） - 最易懂的设计模式解析简单工厂模式（SimpleFactoryPattern） - 最...

 Carson_Ho (/u/383970bef0a0?


utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

(/p/9d0575311214?



utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)
适配器模式（Adapter Pattern） - 最易懂的设计模式解析 (/p/9d057531121...

前言 今天我来全面总结一下Android开发中最常用的设计模式 - 适配器模式。 其他设计模式介绍1分钟全面了解“设计模式”单例模式（Singleton） - 最易懂的设计模式解析简单工厂模式（SimpleFactoryPattern） - 最...


 Carson_Ho (/u/383970bef0a0?

utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

(/p/b8c578b07fbc?

utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)
单例模式（Singleton） - 最易懂的设计模式解析 (/p/b8...

前言 今天我来全面总结一下Android开发中最常用的设计模式 - 单例模式。 其他设计模式介绍1分钟全面了解“设计模式”单例模式（Singleton） - 最易懂的设计...

 Carson_Ho (/u/383970bef0a0?

utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)




(/p/2e45f8ade8f5?



utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)
幽兰含香竹萧萧 (/p/2e45f8ade8f5?utm_campaign=maleskine&utm_cont...

扬州郑板桥纪念馆上半年开馆了，在天宁寺北面几间禅房内。我跑去看。郑板桥是江苏兴化人，乾隆进士，在山东潍县做过县令。后来辞官来扬州以卖画为生，是扬州八怪代表人物之一。他说他一生只做两件事，...

 好风明月暗知心 (/u/104bd675c575?


utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

(/p/45de0e3c8ea5?



utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)
禅绕图样02~气泡酒 (/p/45de0e3c8ea5?utm_campaign=maleskine&utm_...

今天画的这个图样，叫做“气泡酒”。虽然很简单，但是很有气场，也是极考验耐心的。画的时候一定要轻轻下笔，慢慢地画，时刻告诉自己“慢就是快”。准备：1. 观察自己的坐姿，然后调整，让自己舒服2. 观察自...

 水草鱼儿 (/u/0c6e0324599e?


utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

(/p/2320d78a9118?



utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)
教师节，我们要哪般 (/p/2320d78a9118?utm_campaign=maleskine&utm_...


“教师节要到了，你家准备多少红票票？”这天一上班，就有人颇有些神秘地凑上来问小张。小张则一脸的懵圈，“什么情况？我不知道啊。”同事张大了嘴，先是愕然，继而尴尬地笑笑，“那，就当我说。”教师...

 丫丫沛 (/u/f469abf90894?

utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

离婚为什么越来越容易？ (/p/4ca43d67d12a?utm_campaign=maleskine&...

和闺蜜出去见朋友，一个很漂亮的美女。跟我们是同龄人，打扮优雅有气质。一看就是个很幸福的人，就是那种家庭条件好，爸妈疼，老公爱的人，才能有这等面容和自信温婉的气质。可是我俩都想错了，她两年...

 夏花落 (/u/6c2dc8c885bd?


utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

(/p/9f715815d8f6?



utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)
什么是UI设计？学习UI设计需要掌握哪些知识体系？ UI 设计初学者应该怎样...

写在最前：想一起学习交流的童鞋加我微信：XY521JS，我邀请你进UI设计微信学习群。另外每人可以免费领取100多本设计书籍和零基础视频学习教程一套！更多关于UI设计的学习方法和经验总结请关注微信公众...

 UI设计院 (/u/4a57e95cfb55?

utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

^

+

🔖

🔗