

Stroller

Life has no end beyond itself

目录视图

摘要视图

RSS 订阅

我的个人小站

http://www.strolling.cn

我的微博

http://weibo.com/fujianthinking

个人资料



jiafu1115

访问：661652次

积分：9342

等级：BLOG 5

排名：第1051名

原创：176篇 转载：727篇

译文：1篇 评论：71条

文章搜索

Q

文章分类

Design Pattern (7)

Database|Oracle (20)

JavaJ2SE (445)

Linux|CentOS (26)

InfoSystem Program (4)

Education |Mobile (2)

Others (27)

Web Service (25)

Maven (26)

JVM (22)

Site Collection (4)

Concurrency (4)

Java IO (23)

Java Doc (4)

NetWork (4)

C++ (271)

IDE (4)

高并发程序设计入门

【活动】云计算行业圆桌论坛

【知识库】一张大图看懂Android架构

【征文】Hadoop十周年特别策划——我与Hadoop不得不说的故事

抽象工厂和Builder模式区别

标签：[ibm](#) [hp](#) [class](#) [设计模式](#) [产品](#) [面试](#)

2011-08-08 12:19

982人阅读

评论(0)

分类：[Design Pattern \(6\)](#)

(1) 侧重产品生成结果，侧重产品过程

(2) 后者测试相同过程的产生，或有序列的生产过程

(3) 后面有相同的生产过程，用于被创建的对象之间有紧密的关系，前者一般不需要

因此核心要点在于：**Builder**需要有相同的生产过程，且有部件需要生产，而工厂模式一般没有，更侧重生产结果。

最近要参加面试,于是乎又把设计模式拿出来过了一遍.由于每次在看到抽象工厂和Builder模式的时候总是有点迷糊,因此这次下了狠心,翻箱倒柜的找出英文版教材,中英对照,希望能把这两种模式搞清楚.

所有的创建型模式的本质目的都是为了更好的创建对象,抽象工厂和Builder模式也是如此,另外,两种模式还有一个共同的特点,就是将对象创建过程与使用过程相分离,用户在使用时只需知道该创建什么,而无需知道对象是究竟如何创建的.这样对象创建和使用的过程之间就呈现一种松耦合的形式,当创建过程有改动的时候只需对创建过程进行无需对使用过程作出任何修改.除此之外,两种模式同样都是被用于将部件对象构成一个完整的对象.举个例子,某工厂生产IBM电脑和HP电脑,两种电脑的配置不相同,但都是由CPU、主板、内存、硬盘等部件构成,工厂模式和Builder模式都可以根据不同的配置生成IBM电脑或者HP电脑。

两种模式的共同点使得在刚刚开始学习的时候,非常容易混淆,其实仔细研究,两者之间的区别也是非常明显的,而我认为两者之间最本质的区别是,抽象工厂通过不同的构建过程生成不同的对象表示,而Builder模式通过相同的构建过程生成不同的表示。

builder 也是一个高层建筑,但是他和Abstract Factory侧重点不同, Abstract Factory侧重于创建东西的结果,而builder侧重的是创建东西的过程。当你需要做一系列有序的工作来完成创建一个对象时 builder就派上用场啦

上面的文字比较抽象，下面举个例子说明。

假设有具体工厂类IBMFactory,HPFactory和具体对象类IBM,HP,其中工厂类继承自AbstractFactory类,工厂类实现方法GetProduct(),具体对象类继承自Computer类,对象类实现方法Attach()用于装配不同部件,设备IT都继承自基类Component,演示代码如下:

```
abstract class AbstractFactory //工厂基类
...{
    abstract public Computer GetProduct(); //抽象方法
}
class IBMFactory:AbstractFactory //生产IBM电脑
...{
    public Computer GetProduct()
    ...{
```



2 of 4 16/2/29 上午12:21

a030703130: java正则表达式教程 学习地址: [http://www.java3z.com/cwbwebhome/...](http://www.java3z.com/cwbwebhome/)

replaceAll ()/appendReplacemer
a030703130: 正则表达式matcher类 学习地址: [http://www.java3z.com/cwbwebhome/...](http://www.java3z.com/cwbwebhome/)

评论排行

- | | |
|----------------------------|-----|
| 知识都学杂了 | (8) |
| Java解惑笔记<不断更新: | (4) |
| JAVA多态与类型转化分析 | (3) |
| 读完《重构》的一些感想 | (3) |
| 关于Coding的学习与思考 | (3) |
| instanceOf 与 instanceof | (3) |
| 简单工厂模式、工厂方法 | (3) |
| java+方法覆盖必须不减 | (3) |
| Access restriction : The t | (3) |
| java 正斜杠与反斜杠之分 | (3) |

好友博客

良师益友 Java Tips
<http://www.javamex.com>

```
...{
    abstract public Computer Building();
}

class IBMBuilder //IBM对象的创建者
...{
    public Computer Building()
    ...{
        IBM ibm=new IBM();
        ibm.Attach(new IBM所需CPU); //装配CPU
        ibm.Attach(new IBM所需主板); //装配主板
        ibm.Attach(new IBM所需内存); //装配内存
        ibm.Attach(new IBM所需硬盘); //装配硬盘
        return ibm; //返回构建好的对象
    }
}

class HPBuilder //HP对象的创建者
...{
    public Computer Building()
    ...{
        HP hp=new HP()
        hp.Attach(new HP所需CPU); //装配CPU
        hp.Attach(new HP所需主板); //装配主板
        hp.Attach(new HP所需内存); //装配内存
        hp.Attach(new HP所需硬盘); //装配硬盘
        return hp; //返回构建好的对象
    }
}

class Director //指导者类
...{
    protected Computer computer=null; //保存具体对象
    public void Construct(Builder builder) //创建具体对象
    ...{
        computer=builder.Building();
    }
    public Computer GetComputer() //获取已经创建好的对象
    ...{
        return computer;
    }
}
```

使用过程如下:

IBMBuild ibmBuilder=new IBMBuild(); //实例化IBM创建者对象

HPBuilder hpBuilder=new HPBuilder(); //实例化HP创建者对象

Director director=new Director(); //实例化创建者对象

director.Construct(ibmBuilder); //创建IBM对象

IBM ibm=(IBM)director.GetComputer(); //获取IBM对象

irector.Construct(hpBuilder); //创建HP对象

HP hp=(HP)director.GetComputer(); //获取HP对象

从上面代码可以看出, IBM和HP对象的创建过程均有Director的Constuct()实现, 只要设置不同的Builder, 就可以创建不同的对象。也即, 用相同的创建过程创建不同的对象。

直观上的感觉, Builder模式是抽象工厂模式的再封装, 不但实现了创建过程的隐藏, 甚至连创建过程该有那个方法实现都不必再去考虑, 这样我们可以更加黑盒的去构建不同的对象。但事实上, 两种模式有本质的区别, Builder模式的使用前提是被创建的对象之间有紧密的关系, 属于同一类对象, 当组成对象类型不同的时候, Builder模式就显得力不从心了, 这是就需要配合抽象工厂模式来构建由不同类型的对象构成的对象了。

关闭

顶 踩

0 1

上一篇 [JAVA设计模式之创建者模式](#)

下一篇 [Java path](#)

我的同类文章

Design Pattern (6)

• [欢迎使用CSDN-markdown...](#)

2015-07-10

阅读 84

• [代理模式和装饰者模式的异...](#)

2011-07-09

阅读 358

• [Java设计|单例模式](#)

2011-07-09

阅读 249

• [对代理模式与Java动态代理...](#)

2011-07-30

阅读 211

• [Java设计|生成器模式](#)

2011-07-09

阅读 271

• [简单工厂模式、工厂方法模...](#)

2011-07-08

阅读 271

主题推荐

工厂模式

产品

对象

测试

class

猜你在找

- 软件测试工程师面试突击——100道试题精讲视频（
- Java开发中的23种设计模式详解
- “职业测评，企业面试技巧”课程
- Java 23种设计模式全解析之二
- 移动APP测试基础到进阶
- java设计模式
- 设计模式课程
- Java开发中的23种设计模式详解
- C/C++单元测试培训
- 设计模式之工厂方法模式



查看评论

暂无评论

您还没有登录,请[\[登录\]](#)或[\[注册\]](#)

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目

全部主题

Hadoop

AWS

移动游戏

Java

Android

iOS

Swift

智能硬件

Docker

OpenStack

VPN

Spark

ERP

IE10

Eclipse

CRM

JavaScript

数据库

Ubuntu

NFC

WAP

jQuery

BI

HTML5

Spring

Apache

.NET

API

HTML

SDK

IIS

Fedora

XML

LBS

Unity

Splashtop

UML

components

Windows Mobile

Rails

QEMU

KDE

Cassandra

CloudStack

FTC

coremail

OPhone

CouchBase

云计算

iOS6

Rackspace

Web App

SpringSide

Maemo

Compuware

大数据

aptech

Perl

Tornado

Ruby

Hibernate

ThinkPHP

HBase

Pure

Solr

Angular

Cloud Foundry

Redis

Scala

Django

Bootstrap

关闭

公司简介 | 招贤纳士 | 广告服务 | 银行汇款帐号 | 联系方式 | 版权声明 | 法律顾问 | 问题报告 | 合作伙伴 | 论坛反馈

网站客服 杂志客服 微博客服 webmaster@csdn.net 400-600-2320 | 北京创新乐知信息技术有限公司 版权所有 | 江苏乐知网络技术有限公司 提供商务支持
京 ICP 证 09002463 号 | Copyright © 1999-2014, CSDN.NET, All Rights Reserved