

16个PHP设计模式详解



实验楼 (/u/544f4bcb9d36) [+ 关注](#)

2016.12.16 15:42* 字数 2247 阅读 9130 评论 3 喜欢 44

(/u/544f4bcb9d36)

说明：该教程全部截选自实验楼教程【16个PHP设计模式详解】
(<https://link.jianshu.com?t=https://www.shiyanlou.com/courses/699>)：主要介绍16个常用的设计模式的基础概念和技术要点，通过UML类图帮助理解设计模式中各个类之间的关联关系，针对每种设计模式都使用PHP完成了一个代码示例，让你跟随实例轻松入门设计模式。

(https://
click.yc
slot=3C
3c5f-4!
51539;
46716

(http://

一、工厂模式

工厂模式具体可分为三类模式：简单工厂模式，工厂方法模式，抽象工厂模式；


1.简单工厂模式

又称为静态工厂方法(Static Factory Method)模式，它属于类创建型模式。在简单工厂模式中，可以根据参数的不同返回不同类的实例。简单工厂模式专门定义一个类来负责创建其他类的实例，被创建的实例通常都具有共同的父类。

角色：

- Factory类：负责创建具体产品的实例
- Product类：抽象产品类，定义产品子类的公共接口
- ConcreteProduct 类：具体产品类，实现Product父类的接口功能，也可添加自定义的功能

UML类图：

 此处输入图片的描述 此处输入图片的描述

示例代码：



```
<?php
//简单工厂模式
class Cat
{
    function __construct()
    {
        echo "I am Cat class <br>";
    }
}
class Dog
{
    function __construct()
    {
        echo "I am Dog class <br>";
    }
}
class Factory
{
    public static function CreateAnimal($name){
        if ($name == 'cat') {
            return new Cat();
        } elseif ($name == 'dog') {
            return new Dog();
        }
    }
}

$cat = Factory::CreateAnimal('cat');
$dog = Factory::CreateAnimal('dog');
```

(https://
click.yc
slot=30
3c5f-4!
51539;
467160

(http://

简单工厂模式最大的优点在于实现对象的创建和对象的使用分离，将对象的创建交给专门的工厂类负责，但是其最大的缺点在于工厂类不够灵活，增加新的具体产品需要修改工厂类的判断逻辑代码，而且产品较多时，工厂方法代码将会非常复杂。


2.工厂方法模式

此模式中，通过定义一个抽象的核心工厂类，并定义创建产品对象的接口，创建具体产品实例的工作延迟到其工厂子类去完成。这样做的好处是核心类只关注工厂类的接口定义，而具体的产品实例交给具体的工厂子类去创建。当系统需要新增一个产品是，无需修改现有系统代码，只需要添加一个具体产品类和其对应的工厂子类，是系统的扩展性变得很好，符合面向对象编程的开闭原则；

角色：


- Product：抽象产品类
- ConcreteProduct：具体产品类
- Factory：抽象工厂类
- ConcreteFactory：具体工厂类

UML类图：

 此处输入图片的描述 此处输入图片的描述

示例代码：

^



```

<?php
interface Animal{
    public function run();
    public function say();
}
class Cat implements Animal
{
    public function run(){
        echo "I ran slowly <br>";
    }
    public function say(){
        echo "I am Cat class <br>";
    }
}
class Dog implements Animal
{
    public function run(){
        echo "I'm running fast <br>";
    }
    public function say(){
        echo "I am Dog class <br>";
    }
}
abstract class Factory{
    abstract static function createAnimal();
}
class CatFactory extends Factory
{
    public static function createAnimal()
    {
        return new Cat();
    }
}
class DogFactory extends Factory
{
    public static function createAnimal()
    {
        return new Dog();
    }
}

$cat = CatFactory::createAnimal();
$cat->say();
$cat->run();

$dog = DogFactory::createAnimal();
$dog->say();
$dog->run();

```

(https://
click.yc
slot=30
3c5f-4!
51539;
467160

(http://

工厂方法模式是简单工厂模式的进一步抽象和推广。由于使用了面向对象的多态性，工厂方法模式保持了简单工厂模式的优点，而且克服了它的缺点。在工厂方法模式中，核心的工厂类不再负责所有产品的创建，而是将具体创建工作交给子类去做。这个核心类仅仅负责给出具体工厂必须实现的接口，而不负责产品类被实例化这种细节，这使得工厂方法模式可以允许系统在不修改工厂角色的情况下引进新产品。

3.抽象工厂模式

提供一个创建一系列相关或相互依赖对象的接口，而无须指定它们具体的类。抽象工厂模式又称为Kit模式，属于对象创建型模式。



此模式是对工厂方法模式的进一步扩展。在工厂方法模式中，一个具体的工厂负责生产一类具体的产品，即一对一的关系，但是，如果需要一个具体的工厂生产多种产品对象，那么就需要用到抽象工厂模式了。

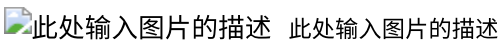
为了便于理解此模式，这里介绍两个概念：

- **产品等级结构**：产品等级结构即产品的继承结构，如一个抽象类是电视机，其子类有海尔电视机、海信电视机、TCL电视机，则抽象电视机与具体品牌的电视机之间构成了一个产品等级结构，抽象电视机是父类，而具体品牌的电视机是其子类。
- **产品族**：在抽象工厂模式中，产品族是指由同一个工厂生产的，位于不同产品等级结构中的一组产品，如海尔电器工厂生产的海尔电视机、海尔电冰箱，海尔电视机位于电视机产品等级结构中，海尔电冰箱位于电冰箱产品等级结构中。

角色：

- 抽象工厂（AbstractFactory）：担任这个角色的是抽象工厂模式的核心，是与应用系统的商业逻辑无关的。
- 具体工厂（Factory）：这个角色直接在客户端的调用下创建产品的实例，这个角色含有选择合适的产品对象的逻辑，而这个逻辑是与应用系统商业逻辑紧密相关的。
- 抽象产品（AbstractProduct）：担任这个角色的类是抽象工厂模式所创建的对象父类，或它们共同拥有的接口
- 具体产品（Product）：抽象工厂模式所创建的任何产品对象都是一个具体的产品类的实例。

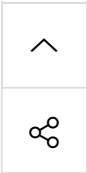
UML类图：



示例代码：

(https://click.yc slot=30 3c5f-4! 51539; 46716

(http://



```

<?php

interface TV{
    public function open();
    public function use();
}

class HaierTv implements TV
{
    public function open()
    {
        echo "Open Haier TV <br>";
    }

    public function use()
    {
        echo "I'm watching TV <br>";
    }
}

interface PC{
    public function work();
    public function play();
}

class LenovoPc implements PC
{
    public function work()
    {
        echo "I'm working on a Lenovo computer <br>";
    }
    public function play()
    {
        echo "Lenovo computers can be used to play games <br>";
    }
}

abstract class Factory{
    abstract public static function createPc();
    abstract public static function createTv();
}

class ProductFactory extends Factory
{
    public static function createTV()
    {
        return new HaierTv();
    }
    public static function createPc()
    {
        return new LenovoPc();
    }
}

$newTv = ProductFactory::createTV();
$newTv->open();
$newTv->use();

$newPc = ProductFactory::createPc();
$newPc->work();
$newPc->play();

```

(https://
click.yc
slot=30
3c5f-4!
51539;
467160

(http://

二、建造者模式



又名：生成器模式，是一种对象构建模式。它可以将复杂对象的建造过程抽象出来（抽象类别），使这个抽象过程的不同实现方法可以构造出不同表现（属性）的对象。

建造者模式是一步一步创建一个复杂的对象，它允许用户只通过指定复杂对象的类型和内容就可以构建它们，用户不需要知道内部的具体构建细节。例如，一辆汽车由轮子，发动机以及其他零件组成，对于普通人而言，我们使用的只是一辆完整的车，这时，我们需要加入一个构造者，让他帮我们把这些组件按序组装成为一辆完整的车。

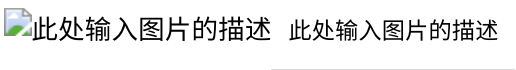
角色：

- Builder：抽象构造者类，为创建一个Product对象的各个部件指定抽象接口。
 - ConcreteBuilder：具体构造者类，实现Builder的接口以构造和装配该产品的各个部件。定义并明确它所创建的表示。提供一个检索产品的接口
 - Director：指挥者，构造一个使用Builder接口的对象。
 - Product：表示被构造的复杂对象。ConcreteBuilder创建该产品的内部表示并定义它的装配过程。
- 包含定义组成部件的类，包括将这些部件装配成最终产品的接口。

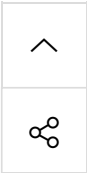
(https://
click.yc
slot=30
3c5f-4!
51539;
46716f

(http://

UML类图：



示例代码：



```

<?php
/**
 * chouxiang builer
 */
abstract class Builder
{
    protected $car;
    abstract public function buildPartA();
    abstract public function buildPartB();
    abstract public function buildPartC();
    abstract public function getResult();
}

class CarBuilder extends Builder
{
    function __construct()
    {
        $this->car = new Car();
    }
    public function buildPartA(){
        $this->car->setPartA('发动机');
    }

    public function buildPartB(){
        $this->car->setPartB('轮子');
    }

    public function buildPartC(){
        $this->car->setPartC('其他零件');
    }

    public function getResult(){
        return $this->car;
    }
}

class Car
{
    protected $partA;
    protected $partB;
    protected $partC;

    public function setPartA($str){
        $this->partA = $str;
    }

    public function setPartB($str){
        $this->partB = $str;
    }

    public function setPartC($str){
        $this->partC = $str;
    }

    public function show()
    {
        echo "这辆车由: ".$this->partA.', '.$this->partB.', 和 '.$this->partC.'组成';
    }
}

class Director
{
    public $myBuilder;

    public function startBuild()
    {
        $this->myBuilder->buildPartA();
        $this->myBuilder->buildPartB();
    }
}

```

(https://
click.yc
slot=30
3c5f-4!
51539;
467160

(http://



```
$this->myBuilder->buildPartC();
return $this->myBuilder->getResult();
}

public function setBuilder(Builder $builder)
{
    $this->myBuilder = $builder;
}
}

$carBuilder = new CarBuilder();
$director = new Director();
$director->setBuilder($carBuilder);
$newCar = $director->startBuild();
$newCar->show();
```

(https://
click.yc
slot=30
3c5f-4!
51539;
467160

三、单例模式

单例模式，也叫**单子模式**，是一种常用的软件设计模式 (<https://link.jianshu.com?t=https://zh.wikipedia.org/wiki/%E8%BD%AF%E4%BB%B6%E8%AE%BE%E8%AE%A1%E6%A8%A1%E5%BC%8F>)。在应用这个模式时，单例对象的类 (<https://link.jianshu.com?t=https://zh.wikipedia.org/wiki/%E7%B1%BB>)必须保证只有一个实例存在。许多时候整个系统只需要拥有一个的全局对象 (<https://link.jianshu.com?t=https://zh.wikipedia.org/wiki/%E5%AF%B9%E8%B1%A1>)，这样有利于我们协调系统整体的行为。

(http://

实现单例模式的思路是：一个类能返回对象一个引用(永远是同一个)和一个获得该实例的方法（必须是静态方法，通常使用getInstance这个名称）；当我们调用这个方法时，如果类持有的引用不为空就返回这个引用，如果类保持的引用为空就创建该类的实例并将实例的引用赋予该类保持的引用；同时我们还将该类的构造函数 (<https://link.jianshu.com?t=https://zh.wikipedia.org/wiki/%E6%9E%84%E9%80%A0%E5%87%BD%E6%95%B0>)定义为私有方法，这样其他处的代码就无法通过调用该类的构造函数来实例化该类的对象，只有通过该类提供的静态方法来得到该类的唯一实例。

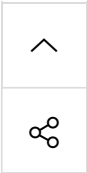
---维基百科


单例模式的要点有：某个类只能有一个实例；它必须自行创建本身的实例；它必须自行向整个系统提供这个实例。单例模式是一种对象创建型模式。

角色：

- Singleton：单例类

UML 类图：



 此处输入图片的描述 此处输入图片的描述

示例代码：

```
<?php

class Singleton
{
    private static $instance;
    //私有构造方法，禁止使用new创建对象
    private function __construct(){}

    public static function getInstance(){
        if (!isset(self::$instance)) {
            self::$instance = new self;
        }
        return self::$instance;
    }
    //将克隆方法设为私有，禁止克隆对象
    private function __clone(){}

    public function say()
    {
        echo "这是用单例模式创建对象实例 <br>";
    }
    public function operation()
    {
        echo "这里可以添加其他方法和操作 <br>";
    }
}

// $shianlou = new Singleton();
$shianlou = Singleton::getInstance();
$shianlou->say();
$shianlou->operation();

$newShianlou = Singleton::getInstance();
var_dump($shianlou === $newShianlou);
```

(https://
click.y
slot=30
3c5f-4!
51539
46716

(http://

上述的五个模式均属于创建型模式，关于结构型模式，点击【16个PHP设计模式详解】
(<https://link.jianshu.com?t=https://www.shiyanlou.com/courses/699>)即可查看了.....

来啊，来shiyanlou.com学IT呀，反正有大把时间~

赞赏支持

 PHP (/nb/3349190) 举报文章 © 著作权归作者所有



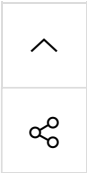
实验楼 (/u/544f4bcb9d36)

写了 607432 字，被 6012 人关注，获得了 10050 个喜欢

(/u/544f4bcb9d36)

+ 关注

实验楼是专业的IT在线实训平台，不但提供海量的IT教程，更有在线开发环境，可以随时动手操作，实战式...



喜欢 | 44



更多分享



(/apps/redirect?utm_source=note-bottom-click)

(https://
click.yc
slot=3C
3c5f-4!
51539;
46716

被以下专题收入，发现更多相似内容

IT课程分享 (/c/0e47180b3a1b?utm_source=desktop&utm_medium=notes-included-collection)

PHP (/c/27de7c36903c?utm_source=desktop&utm_medium=notes-included-collection)

开发模式 (/c/3d83df5dda0f?utm_source=desktop&utm_medium=notes-included-collection)

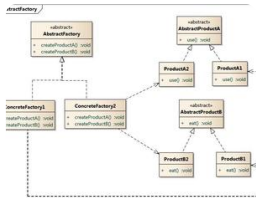
(http://

设计模式汇总 (/p/f7f9553ba2ba?utm_campaign=maleskine&utm_conte...

设计模式汇总 一、基础知识 1. 设计模式概述 定义：设计模式（Design Pattern）是一套被反复使用、多数人知晓的、经过分类编目的、代码设计经验的总结，使用设计模式是为了可重用代码、让代码更容易被他人...

MinoyJet (/u/9c0529da1861?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommend

(/p/2201b4be119d?



utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommend
学习设计模式 (2) (/p/2201b4be119d?utm_campaign=maleskine&utm_...

创建型模式 抽象工厂模式 (abstract factory) 3.1模式动机 在工厂方法模式中具体工厂负责生产具体的产品，每一个具体工厂对应一种具体产品，工厂方法也具有唯一性，一般情况下，一个具体工厂中只有一个...


僚机KK (/u/4b35ebf9dedb?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommend



(/p/e873855e88a0?
utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommend

【创建型模式三】抽象工厂(Abstract Factory) (/p/e87...

1 场景问题# 1.1 选择组装电脑的配件## 举个生活中常见的例子——组装电脑，我们在组装电脑的时候，通常需要选择一系列的配件，比如：CPU、硬盘、...

 猿码道 (/u/657c611b2e07?

utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommenc

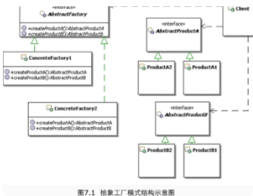
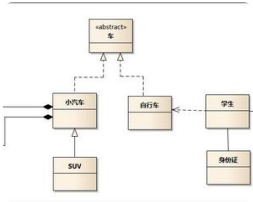


图7.1 抽象工厂模式结构示意图


(/p/7df9b2a1a315?



utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommenc

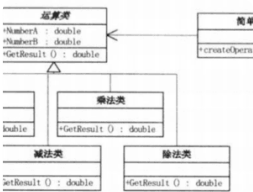
学习设计模式（1） (/p/7df9b2a1a315?utm_campaign=maleskine&utm_...

一个UML类图 类之间的关系 类的继承结构表现在UML中为：泛化(generalize)与实现(realize) 泛化关系 (generalization) 泛化关系用一条带空心箭头的直接表示； eg:汽车与SUV之间为泛化关系； 实现关系...

 僚机KK (/u/4b35ebf9dedb?

utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommenc


(/p/47c67255c842?



utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommenc

大话设计模式阅读笔记 (/p/47c67255c842?utm_campaign=maleskine&ut...

设计模式基本原则 开放-封闭原则（OCP），是说软件实体（类、模块、函数等等）应该可以拓展，但是不可修改。开-闭原则是面向对象设计中最基本的原则，曾经看过一篇文章对此归纳的非常好。单一职责原则...

 西山薄凉 (/u/b911563955c5?

utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommenc


(/p/ab0aa5803e9c?



utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommenc

入学 (/p/ab0aa5803e9c?utm_campaign=maleskine&utm_content=note...

文：哈哈世界 早在两周前女儿就开始收拾行李了，零七碎八的东西打包，大大小小的包裹靠墙排队，简直像是小搬家。女儿关键时刻掉链子的个性再次闪光，户口迁移竟然留了小尾巴。六月份没少叮嘱她户口的事...


 哈哈世界123 (/u/6e94644144df?

utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommenc

Swift/Storyboard 界面跳转/传值 (/p/533797f9ff86?utm_campaign=males...

跳转 Push时的Controller为 UINavigationControllerPresent时的Controller为 UINavigationController 使用 Storyboard Segue 方式:在Storyboard中设置Segue的Identifier，以及...



 竹菜板 (/u/c731b4ea6806?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommenc

(连载) 《美梦如璇·第八章·第八十八节》 (/p/f6cc64b79c08?utm_campai...

(连载) 《美梦如璇·第八章·第八十八节》 少年时节读过的书，行过的路，遇见的人；怀梦日子里度过的苦，流淌的泪，离散的天涯；星辰海途中渡过的光阴，回首的过往，憧憬的未来。你在或者不在，你见证...

 叶子程 (/u/241791d0b96f?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommenc

(/p/3085aa94b732?



(https://click.ycslot=303c5f-4!51539;46716

utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommenc
培训师访谈 | 从内训师到职业培训师，你要避开这些坑 (/p/3085aa94b732?...

(http://

鹏有约，聆听奋斗故事 记录别样人生 刘明源导师 嘉宾简介刘明源 首批国家认证高级人力资源管理师；刘子熙职业训练发展研究院特约高级研究；《IPTA国际职业培训师标准教程》授权导师；安徽合纵连横企业...

 鹏有约 (/u/9fc7cc1d9f40?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommenc

弱电智能化行业最新招标信息2016.10.28 (/p/9cb21e118726?utm_campai...

1、中国质量认证中心视频会议系统及会议室音频系统及配件采购项目招标公告2、北京国家会计学院摄像控制系统招标公告3、朝阳区垡头地区焦化厂棚户区改造安置房项目（第一标段）智能化工程专业招标资格预...

 中网国金 (/u/2e7e9638c071?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommenc

