开贝模板构造器:更好的构造开发模板 五种PHP设计模..

设计模式只是为 Java 架构师准备的 —— 至少您可能一直这样认为。实际上，设计模式对于每个人都非常有用。如果这些工具不是 “架构太空人” 的专利，那么它们又是什么？为什么说它们在 PHP 应用程序中非常有用？本文解释了这些问题。

设计模式 一书将设计模式引入软件社区，该书的作者是 Erich Gamma、Richard Helm、Ralph Johnson 和 John Vlissides Design（俗称 “四人帮” ）。所介绍的设计模式背后的核心概念非常简单。经过多年的软件开发实践，Gamma 等人发现了某些具有固定设计的模式，就像建筑师设计房子和建筑物一样，可以为浴室的位置或厨房的构造方式开发模板。使用这些模板或者说设计模式 意味着可以更快地设计更好的建筑物。同样的概念也适用于软件。

设计模式不仅代表着更快开发健壮软件的有用方法，而且还提供了以友好的术语封装大型理念的方法。例如，您可以说您正在编写一个提供松散耦合的消息传递系统，也可以说你正在编写名称为观察者 的模式。

用较小的示例展示模式的价值是非常困难的。这往往有些大材小用的意味，因为模式实际上是在大型代码库中发挥作用的。本文不展示大型应用程序，所以您需要思索的是在您自己的大型应用程序中应用示例原理的方法 —— 而不是本文演示的代码本身。这不是说您不应该在小应用程序中使用模式。很多良好的应用程序都以小应用程序为起点，逐渐发展到大型应用程序，所以没有理由不以此类扎实的编码实践为基础。 既然您已经了解了设计模式以及它们的有用之处，现在我们来看看 PHP V5 的五种常用模式。

工厂模式

最初在设计模式 一书中，许多设计模式都鼓励使用松散耦合。要理解这个概念，让我们最好谈一下许多开发人员从事大型系统的艰苦历程。在更改一个代码片段时，就会发生问题，系统其他部分 —— 您曾认为完全不相关的部分中也有可能出现级联破坏。

该问题在于紧密耦合 。系统某个部分中的函数和类严重依赖于系统的其他部分中函数和类的行为和结构。您需要一组模式，使这些类能够相互通信，但不希望将它们紧密绑定在一起，以避免出现联锁。 在大型系统中，许多代码依赖于少数几个关键类。需要更改这些类时，可能会出现困难。例如，假设您有一个从文件读取的 User 类。您希望将其更改为从数据库读取的其他类，但是，所有的代码都引用从文件读取的原始类。这时候，使用工厂模式会很方便。

工厂模式 是一种类，它具有为您创建对象的某些方法。您可以使用工厂类创建对象，而不直接使用 new。这样，如果您想要更改所创建的对象类型，只需更改该工厂即可。使用该工厂的所有代码会自动更改。

清单 1 显示工厂类的一个示列。等式的服务器端包括两个部分：数据库和一组 PHP 页面，这些页面允许您添加反馈、请求反馈列表并获取与特定反馈相关的文章。

清单 1. Factory1.php

IUser 接口定义用户对象应执行什么操作。IUser 的实现称为 User，UserFactory 工厂类则创建 IUser 对象。此关系可以用图 1 中的 UML 表示。

如果您使用 php 解释器在命令行上运行此代码，将得到如下结果：

测试代码会向工厂请求 User 对象，并输出 getName 方法的结果。

有一种工厂模式的变体使用工厂方法。类中的这些公共静态方法构造该类型的对象。如果创建此类型的对象非常重要，此方法非常有用。例如，假设您需要先创建对象，然后设置许多属性。此版本的工厂模式会将该进程封装在单个位置中，这样，不用复制复杂的初始化代码，也不必将复制好的代码在在代码库中到处粘贴。清单 2 显示使用工厂方法的一个示例。

清单 2. Factory2.php

这段代码要简单得多。它仅有一个接口 IUser 和一个实现此接口的 User 类。User 类有两个创建对象的静态方法。此关系可用图 2 中的 UML 表示。

在命令行中运行脚本产生的结果与清单 1 的结果相同，如下所示：

如上所述，有时此类模式在规模较小的环境中似乎有些大材小用。不过，最好还是学习这种扎实的编码形式，以便应用于任意规模的项目中。单元素模式

某些应用程序资源是独占的，因为有且只有一个此类型的资源。例如，通过数据库句柄到数据库的连接是独占的。您希望在应用程序中共享数据库句柄，因为在保持连接打开或关闭时，它是一种开销，在获取单个页

面的过程中更是如此。

单元素模式可以满足此要求。如果应用程序每次包含且仅包含一个对象，那么这个对象就是一个单元素（Singleton）。清单 3 中的代码显示了 PHP V5 中的一个数据库连接单元素。

清单 3. Singleton.php

此代码显示名为 DatabaseConnection 的单个类。您不能创建自己的 DatabaseConnection，因为构造函数是专用的。但使用静态 get 方法，您可以获得且仅获得一个 DatabaseConnection 对象。此代码的 UML 如图 3 所示。

在两次调用间，handle 方法返回的数据库句柄是相同的，这就是最好的证明。您可以在命令行中运行代码来观察这一点。

返回的两个句柄是同一对象。如果您在整个应用程序中使用数据库连接单元素，那么就可以在任何地方重用同一句柄。

您可以使用全局变量存储数据库句柄，但是，该方法仅适用于较小的应用程序。在较大的应用程序中，应避免使用全局变量，并使用对象和方法访问资源。观察者模式

观察者模式为您提供了避免组件之间紧密耦合的另一种方法。该模式非常简单：一个对象通过添加一个方法（该方法允许另一个对象，即观察者 注册自己）使本身变得可观察。当可观察的对象更改时，它会将消息发送到已注册的观察者。这些观察者使用该信息执行的操作与可观察的对象无关。结果是对象可以相互对话，而不必了解原因。 一个简单示例是系统中的用户列表。清单 4 中的代码显示一个用户列表，添加用户时，它将发送出一条消息。添加用户时，通过发送消息的日志观察者可以观察此列表。

清单 4. Observer.php

测试代码为两个策略运行同一用户列表，并显示结果。在第一种情况中，策略查找排列在 J 后的任何名称，所以您将得到 Jack、Lori 和 Megan。第二个策略随机选取名称，每次会产生不同的结果。在这种情况下，结果为 Andy 和 Megan。

策略模式非常适合复杂数据管理系统或数据处理系统，二者在数据筛选、搜索或处理的方式方面需要较高的灵活性。

结束语

本文介绍的仅仅是 PHP 应用程序中使用的几种最常见的设计模式。在设计模式 一书中演示了更多的设计模式。不要因架构的神秘性而放弃。模式是一种绝妙的理念，适用于任何编程语言、任何技能水平。

2008-9-4 23:52:04

疯狂代码 http://CrazyCoder.cn/