第三章 变量及数据类型

**世界上唯一不变的变化。**

现代编程语言中的基本元素主要有：变量，流程控制接口，函数等等。 我们能否不使用变量来编写程序呢？这显然是可以的，例如：

<?php

[echo](http://www.php.net/echo) "Hello TIPI Readers";

这个程序很简单，输出一个字符串内容。

就和我们仅仅使用二进制也能编程一样，不使用变量也能完成某些工作， 不使用变量我们的程序将会非常难以编写和维护，变量可以让我们将值存储起来， 以便在程序的其他地方使用，或者通过计算保存新的值。

变量具有三个基本组成部分：

1. **名称** 变量的标识符。就像小狗一样，主人可能会给这些小狗起个喜欢的名称。 变量命名上，PHP继承了Perl的语法风格，变量以美元符号开始，后面跟变量名。 一个有效的变量名由字母或者下划线开头，后面跟上任意数量的字母，数字，或者下划线。 PHP同时还支持复合变量，也就是类似$$a的变量，它会进行两次的解释（me：从$$衍生出可变变量）。这给PHP带来了非常灵活的动态特性。
2. **类型** 变量的类型，就像小狗的品种，不同的小狗血统可能会不一样，有的聪明， 有的会购物等等。 在很多静态语言中，变量在定义时就指定了，在程序运行过程中都不允许进行变更， 那如果你有一只能随便指定品种的小狗会不会很拉风呢;-) PHP就是这样，属于弱类型语言，可以随便赋予它任何类型的值。
3. **值内容**。 这是标识符所代表的具体内容。这就像是实实在在的小狗的这个实物。 你可以给任何一条小狗起名为：小七，在编程语言中也是如此，你可以给变量赋予它 所能表示范围的值。不过在同一时间，变量只能有一个值。

PHP中组成变量名的字母可以是英文字母 a-z，A-Z，还可以是 ASCII 字符从 127 到 255（0x7f-0xff）。 变量名是区分大小写的。

除了变量本身，在PHP中我们经常会接触到与变量相关的一些概念，比如：常量，全局变量，静态变量以及类型转换等。 本章我们将介绍这些与变量相关的实现。其中包括PHP本身的变量底层存储结构、弱类型系统的实现以及这些类型之间的相互转换等。

先看一段PHP代码：

<?php

$foo = 10;

$bar = 20;

function change() {

[global](http://www.php.net/global) $foo;#类似python

$bar = 0;

$foo++;

}

change();

[echo](http://www.php.net/echo) $foo, ' ', $bar;

运行代码会输出**11 20**。可是为什么会有这样的输出呢？变量在PHP的内部是如何实现的呢？ 变量的作用域又是怎么实现的呢？ 这是本章将对围绕**变量**这个主题展开讨论，下面我们从最基本的变量实现开始。

不是所有编程语言中的变量的值都可以改变的。想想我们学过的数学中的变量。 他们的值也是不可改变的。例如： x + y = 10; 变量x和y的值是不能发生变化的。 在某个具体场景，也就是某个方程式中只有表示特定的值，变量的值不能改变的好处是： 这样就能尽可能少的产生[副作用](http://en.wikipedia.org/wiki/Side_effect_(computer_science))， 在[Erlang语言](http://www.erlang.org/)中就是如此，它是一门函数式编程语言，非常值得学习。

**数据类型**

从类型的维度来看，编程语言可以分为三大类：

1. 静态类型语言，比如：C/Java等，在静态语言类型中，类型的检查是在编译期(compile-time)确定的， 也就是说在运行时变量的类型是不会发生变化的。
2. 动态语言类型，比如：PHP，python等各种脚本语言，这类语言中的类型是在运行时确定的， 那么也就是说类型通常可以在运行时发生变化
3. 无类型语言，比如：汇编语言，汇编语言操作的是底层存储，他们对类型毫无感知。

[« 第四节 小结](http://www.php-internals.com/book/?p=chapt02/02-04-summary)

[第一节 变量的结构和类型 »](http://www.php-internals.com/book/?p=chapt03/03-01-00-variables-structure)

# 第一节 变量的结构和类型

前言中提到变量的三个基本特性，其中有一个特性为变量的类型，变量都有特定的类型， 如：字符串、数组、对象等等。编程语言的类型系统可以分为强类型和弱类型两种：

强类型语言是一旦某个变量被申明为某个类型的变量，在程序运行过程中，就不能将该变量的类型以外的值赋予给它 （当然并不完全如此，这可能会涉及到类型的转换，后面的小节会有相应介绍），C/C++/Java等语言就属于这类。

PHP及Ruby，JavaScript等脚本语言属于弱类型语言：一个变量可以表示任意的数据类型。

PHP之所以成为一个简单而强大的语言，很大一部分的原因是它拥有弱类型的变量。 但是有些时候这也是一把双刃剑，使用不当也会带来一些问题。就像仪器一样，越是功能强大， 出现错误的可能性也就越大。

在官方的PHP实现内部，所有变量使用同一种数据结构(zval-（me：zend-value）)来保存，而这个结构同时表示PHP中的各种数据类型。 它不仅仅包含变量的值，也包含变量的类型（me：具体可参考php内部实现变量的接口定义）。这就是PHP弱类型的核心。

那zval结构具体是如何实现弱类型的呢，下面我们一起来揭开面纱。

## 一. PHP变量类型及存储结构

PHP在声明或使用变量的时候，并不需要显式指明其数据类型。

PHP是弱类型语言，这并不表示PHP没有类型，在PHP中，存在8种变量类型，可以分为三类

**标量类型**： boolean、integer、float(double)、string

**复合类型**： array、object

**特殊类型**： resource、NULL

官方PHP是用C实现的，而C是强类型的语言，那这是怎么实现PHP中的弱类型的呢？

### 1. 变量存储结构

变量的值存储到以下所示zval结构体中。 zval结构体定义在Zend/zend.h文件，其结构如下：

typedef struct \_zval\_struct zval;

...

struct \_zval\_struct {

*/\* Variable information \*/*

zvalue\_value value; */\* value \*/*

zend\_uint refcount\_\_gc;

zend\_uchar type; */\* active type \*/*

zend\_uchar is\_ref\_\_gc;

};

PHP使用这个结构来存储变量的所有数据。和其他编译性静态语言不同， PHP在存储变量时将PHP用户空间的变量类型也保存在同一个结构体中。这样我们就能通过这些信息获取到变量的类型。

zval结构体中有四个字段，其含义分别为：

| **属性名** | **含义** | **默认值** |
| --- | --- | --- |
| refcount\_\_gc | 表示引用计数 | 1 |
| is\_ref\_\_gc | 表示是否为引用 | 0 |
| value | 存储变量的值 |  |
| type | 变量具体的类型 |  |

在PHP5.3之后，引入了新的垃圾收集机制，引用计数和引用的字段名改为refcount\_\_gc和is\_ref\_\_gc。在此之前为refcount和is\_\_ref。

而变量的值则存储在另外一个结构体zvalue\_value中。值存储见下面的介绍。

PHP用户空间指的在PHP语言这一层面，而本书中大部分地方都在探讨PHP的实现。 这些实现可以理解为内核空间。由于PHP使用C实现，而这个空间的范畴就会限制在C语言。 而PHP用户空间则会受限于PHP语法及功能提供的范畴之内。   
例如有些PHP扩展会提供一些PHP函数或者类，这就是向PHP用户空间导出了方法或类。

### 2.变量类型:

zval结构体的type字段就是实现弱类型最关键的字段了，type的值可以为： IS\_NULL、IS\_BOOL、IS\_LONG、IS\_DOUBLE、IS\_STRING、IS\_ARRAY、IS\_OBJECT和IS\_RESOURCE 之一。 从字面上就很好理解，他们只是类型的唯一标示，根据类型的不同将不同的值存储到value字段。 除此之外，和他们定义在一起的类型还有IS\_CONSTANT和IS\_CONSTANT\_ARRAY。

这和我们设计数据库时的做法类似，为了避免重复设计类似的表，使用一个标示字段来记录不同类型的数据。

## 二.变量的值存储

前面提到变量的值存储在zvalue\_value联合体中，结构体定义如下：

typedef union \_zvalue\_value {

long lval; */\* long value \*/*

double dval; */\* double value \*/*

struct {

char \*val;

int len;

} str;

HashTable \*ht; */\* hash table value \*/*

zend\_object\_value obj;

} zvalue\_value;

这里使用联合体而不是用结构体是出于空间利用率的考虑，因为一个变量同时只能属于一种类型。 如果使用结构体的话将会不必要的浪费空间，而PHP中的所有逻辑都围绕变量来进行的，这样的话， 内存浪费将是十分大的。这种做法成本小但收益非常大。

各种类型的数据会使用不同的方法来进行变量值的存储，其对应赋值方式如下：

* 一般类型

| **变量类型** | **宏** |  |
| --- | --- | --- |
| boolean | ZVAL\_BOOL | 布尔型/整型的变量值存储于(zval).value.lval中，其类型也会以相应的IS\_\*进行存储。  Z\_TYPE\_P(z)=IS\_BOOL/LONG; Z\_LVAL\_P(z)=((b)!=0); |
| integer | ZVAL\_LONG |
| float | ZVAL\_DOUBLE |
| null | ZVAL\_NULL | NULL值的变量值不需要存储，只需要把(zval).type标为IS\_NULL。  Z\_TYPE\_P(z)=IS\_NULL; |
| resource | ZVAL\_RESOURCE | 资源类型的存储与其他一般变量无异，但其初始化及存取实现则不同。  Z\_TYPE\_P(z) = IS\_RESOURCE; Z\_LVAL\_P(z) = l; |

* 字符串String

字符串的类型标示和其他数据类型一样，不过在存储字符串时多了一个字符串长度的字段。

struct {

char \*val;

int len;

} str;

C中字符串是以\0结尾的字符数组，这里多存储了字符串的长度，这和我们在设计数据库时增加的冗余字段异曲同工。 因为要实时获取到字符串的长度的时间复杂度是O(n)，而字符串的操作在PHP中是非常频繁的，这样能避免重复计算字符串的长度， 这能节省大量的时间，是空间换时间的做法。   
这么看在PHP中strlen()函数可以在常数时间内获取到字符串的长度。 计算机语言中字符串的操作都非常之多，所以大部分高级语言中都会存储字符串的长度。

* 数组Array

数组是PHP中最常用，也是最强大变量类型，它可以存储其他类型的数据，而且提供各种内置操作函数。数组的存储相对于其他变量要复杂一些， 数组的值存储在zvalue\_value.ht字段中，它是一个HashTable类型的数据。 PHP的数组使用哈希表来存储关联数据。哈希表是一种高效的键值对存储结构。PHP的哈希表实现中使用了两个数据结构HashTable和Bucket。 PHP所有的工作都由哈希表实现，在下节HashTable中将进行哈希表基本概念的介绍以及PHP的哈希表实现。

* 对象Object

在面向对象语言中，我们能自己定义自己需要的数据类型，包括类的属性，方法等数据。而对象则是类的一个具体实现。 对象有自身的状态和所能完成的操作。

PHP的对象是一种复合型的数据，使用一种zend\_object\_value的结构体来存放。其定义如下：

typedef struct \_zend\_object\_value {

zend\_object\_handle handle; *// unsigned int类型，EG(objects\_store).object\_buckets的索引*

zend\_object\_handlers \*handlers;

} zend\_object\_value;

PHP的对象只有在运行时才会被创建，前面的章节介绍了EG宏，这是一个全局结构体用于保存在运行时的数据。 其中就包括了用来保存所有被创建的对象的对象池，EG(objects\_store)，而object对象值内容的zend\_object\_handle域就是当前 对象在对象池中所在的索引，handlers字段则是将对象进行操作时的处理函数保存起来。 这个结构体及对象相关的类的结构\_zend\_class\_entry，将在第五章作详细介绍。

PHP的弱变量容器的实现方式是兼容并包的形式体现，针对每种类型的变量都有其对应的标记和存储空间。 使用强类型的语言在效率上通常会比弱类型高，因为很多信息能在运行之前就能确定，这也能帮助排除程序错误。 而这带来的问题是编写代码相对会受制约。

PHP主要的用途是作为Web开发语言，在普通的Web应用中瓶颈通常在业务和数据访问这一层。不过在大型应用下语言也会是一个关键因素。 facebook因此就使用了自己的php实现。将PHP编译为C++代码来提高性能。不过facebook的hiphop并不是完整的php实现， 由于它是直接将php编译为C++，有一些PHP的动态特性比如eval结构就无法实现。当然非要实现也是有方法的， hiphop不实现应该也是做了一个权衡。

[« 第三章 变量及数据类型](http://www.php-internals.com/book/?p=chapt03/03-00-variable-and-data-types)

[哈希表(HashTable) »](http://www.php-internals.com/book/?p=chapt03/03-01-01-hashtable)

参考：<http://www.php-internals.com/book/?p=chapt03/03-01-00-variables-structure>