2.2 变量的值

PHP内核提供了三个基础宏来方便我们对变量的值进行操作，这几个宏同样以Z_开头，并且P结尾和PP结尾的同上一节中的宏一样，
分别代表这参数是指针还是指针的指针。

此外，为了进一步方便我们的工作，内核中针对具体的数据类型分别定义了相应的宏。
如针对IS_BOOL型的BVAL组合(Z_BVAL、Z_BVAL_P、Z_BVAL_PP)和针对IS_DOUBLE的DVAL组合(Z_DVAL、ZDVAL_P、ZDVAL_PP)等等。
我们通过下面这个例子来应用一下这几个宏:

```
void display_value(zval zv,zval *zv_p,zval **zv_pp)
{
    if( Z_TYPE(zv) == IS_NULL )
    {
        php_printf("类型是 IS_NULL!\n");
    }

    if( Z_TYPE_P(zv_p) == IS_LONG )
    {
        php_printf("类型是 IS_LONG，值是：%ld" , Z_LVAL_P(zv_p));
    }

    if(Z_TYPE_PP(zv_pp) == IS_DOUBLE )
    {
        php_printf("类型是 IS_DOUBLE,值是：%f" , Z_DVAL_PP(zv_pp) );
    }
}
```

string型变量比较特殊，因为内核在保存String型变量时，不仅保存了字符串的值，还保存了它的长度，
所以它有对应的两种宏组合STRVAL和STRLEN，即：Z_STRVAL、Z_STRVAL_P、Z_STRVAL_PP与Z_STRLEN、Z_STRLEN_P、Z_STRLEN_PP。
前一种宏返回的是char *型，即字符串的地址；后一种返回的是int型，即字符串的长度。

```
void display_string(zval *zstr)
{
    if (Z_TYPE_P(zstr) != IS_STRING) {
        php_printf("这个变量不是字符串!\n");
        return;
    }
    PHPWRITE(Z_STRVAL_P(zstr), Z_STRLEN_P(zstr));
    //这里用了PHPWRITE宏，只要知道它是从Z_STRVAL_P(zstr)地址开始，输出Z_STRLEN_P(zstr)长度的字符就可以了。
}
```

Array型变量的值其实是存储在C语言实现的HashTable中的，
我们可以用ARRVAL组合宏（Z_ARRVAL, Z_ARRVAL_P, Z_ARRVAL_PP）这三个宏来访问数组的值。
如果你看旧版本php的源码或者部分pecl扩展的源码，可能会发现一个HASH_OF()宏，这个宏等价于Z_ARRVAL_P()。
但不推荐在新代码中再使用了。

对象是一个复杂的结构体（zend_object_value结构体），不仅存储属性的定义、属性的值，还存储着访问权限、方法等信息。
内核中定义了以下组合宏让我们方便的操作对象：
OBJ_HANDLE：返回handle标识符，
OBJ_HT：handle表，
OBJCE：类定义，
OBJPROP：HashTable的属性，
OBJ_HANDLER：在OBJ_HT中操作一个特殊的handler方法。
现在不用担心这些宏对象的意思，后续有专门的章节介绍object。

资源型变量的值其实就是一个整数，可以用RESVAL组合宏来访问它，我们把它的值传给zend_fetch_resource函数，便可以得到这个资源的操作句柄，如mysql的链接句柄等。有关资源的内容我们将在第9章展开叙述。

有关值操作的宏都定义在./Zend/zend_operators.h文件里：

```
//操作整数的
#define Z_LVAL(zval)             (zval).value.lval
#define Z_LVAL_P(zval_p)         Z_LVAL(*zval_p)
#define Z_LVAL_PP(zval_pp)       Z_LVAL(**zval_pp)

//操作IS_BOOL布尔型的
#define Z_BVAL(zval)             ((zend_bool)(zval).value.lval)
#define Z_BVAL_P(zval_p)         Z_BVAL(*zval_p)
#define Z_BVAL_PP(zval_pp)       Z_BVAL(**zval_pp)

//操作浮点数的
#define Z_DVAL(zval)             (zval).value.dval
#define Z_DVAL_P(zval_p)         Z_DVAL(*zval_p)
#define Z_DVAL_PP(zval_pp)       Z_DVAL(**zval_pp)

//操作字符串的值和长度的
#define Z_STRVAL(zval)           (zval).value.str.val
#define Z_STRVAL_P(zval_p)       Z_STRVAL(*zval_p)
#define Z_STRVAL_PP(zval_pp)         Z_STRVAL(**zval_pp)

#define Z_STRLEN(zval)           (zval).value.str.len
#define Z_STRLEN_P(zval_p)       Z_STRLEN(*zval_p)
#define Z_STRLEN_PP(zval_pp)         Z_STRLEN(**zval_pp)

#define Z_ARRVAL(zval)           (zval).value.ht
#define Z_ARRVAL_P(zval_p)       Z_ARRVAL(*zval_p)
#define Z_ARRVAL_PP(zval_pp)         Z_ARRVAL(**zval_pp)

//操作对象的
#define Z_OBJVAL(zval)           (zval).value.obj
#define Z_OBJVAL_P(zval_p)       Z_OBJVAL(*zval_p)
#define Z_OBJVAL_PP(zval_pp)         Z_OBJVAL(**zval_pp)

#define Z_OBJ_HANDLE(zval)       Z_OBJVAL(zval).handle
#define Z_OBJ_HANDLE_P(zval_p)       Z_OBJ_HANDLE(*zval_p)
#define Z_OBJ_HANDLE_PP(zval_p)      Z_OBJ_HANDLE(**zval_p)

#define Z_OBJ_HT(zval)           Z_OBJVAL(zval).handlers
#define Z_OBJ_HT_P(zval_p)       Z_OBJ_HT(*zval_p)
#define Z_OBJ_HT_PP(zval_p)      Z_OBJ_HT(**zval_p)

#define Z_OBJCE(zval)            zend_get_class_entry(&(zval) TSRMLS_CC)
#define Z_OBJCE_P(zval_p)        Z_OBJCE(*zval_p)
#define Z_OBJCE_PP(zval_pp)      Z_OBJCE(**zval_pp)

#define Z_OBJPROP(zval)          Z_OBJ_HT((zval))->get_properties(&(zval) TSRMLS_CC)
#define Z_OBJPROP_P(zval_p)      Z_OBJPROP(*zval_p)
#define Z_OBJPROP_PP(zval_pp)        Z_OBJPROP(**zval_pp)

#define Z_OBJ_HANDLER(zval, hf)      Z_OBJ_HT((zval))->hf
#define Z_OBJ_HANDLER_P(zval_p, h)  Z_OBJ_HANDLER(*zval_p, h)
#define Z_OBJ_HANDLER_PP(zval_p, h)      Z_OBJ_HANDLER(**zval_p, h)
```

```
#define Z_OBJDEBUG(zval,is_tmp)     (Z_OBJ_HANDLER((zval),get_debug_info)?  \
                          Z_OBJ_HANDLER((zval),get_debug_info)(&(zval),&is_tmp TSRMLS_CC): \
                          (is_tmp=0,Z_OBJ_HANDLER((zval),get_properties)?Z_OBJPROP(zval):NULL))
#define Z_OBJDEBUG_P(zval_p,is_tmp) Z_OBJDEBUG(*zval_p,is_tmp)
#define Z_OBJDEBUG_PP(zval_pp,is_tmp)   Z_OBJDEBUG(**zval_pp,is_tmp)

//操作资源的
#define Z_RESVAL(zval)           (zval).value.lval
#define Z_RESVAL_P(zval_p)      Z_RESVAL(*zval_p)
#define Z_RESVAL_PP(zval_pp)         Z_RESVAL(**zval_pp)
```

links