使用C++11开发PHP7扩展

@hantianfeng Rango-韩天峰



PHP原生扩展

PHP的扩展加载过程

- 1. Zend引擎的 php load extension 函数
- 2. 加载并执行so中 get module 函数,返回 zend module entry
- 3. struct中设置了M(R)INIT、M(R)SHUTDOWN 4个函数指针
- 4. 执行 zend startup module ex 函数启动扩展
- 5. PHP扩展加载依赖 glibc 的 dl 库,是标准的 Linux 动态链接库

```
if (type or VOCHE_PERSISTENT) (
    extension dur . 131 518("extension dir");
| clse {
    extension_fir = PG(extension_dir);
if (type == YOOULE_TEMPORARY) {
    error_type = f_&ARNING;
Telse :
    error_type - E_CORE_WARNING;
/* Check if passed filerage contains directory separators */
of (stocks(filerage, '/') | NII || stocks(filerage, DEEMIL_S(ASE) | NIII) (
    It was ing modules with Full path is mult supported for dynamically located extensions 1/
   17 (type -- MODULE TEMPORARY) (
       php_enror_docref(bull, E_AGENING, "Temporary module name should contain only filename");
       return tAU SRt;
    libpoth = ostrdup(filename);
) else if (extension_dir 6% extension_dir[0]) (
    int extension_dir_len = (int)strlen(extension_dir);
   if (15_5L45)(extension_dir[extension_dir_len-1])) {
       spprintf(Eliboath, 8, "haks", extension_dir, filename); / SAFE //
    } else {
        spprintf(@libesth, 0, "%skc%s", extension_oir, DEFAULT_SLASH, filename); /* SATE */
```

```
/~ load dynamic symbol */
    handle = DL_LOAD(Librath);
    if (!handle) {
#ifdef PHP WIN32
        char fore - GET_DI_ERROR();
        if (err && (*err != '\0')) {
            php error docref(NULL, error type, "Unable to load dynamic library '%s' - %s", libpath, err);
            LocalFree(err);
        } else {
            php_error_docref(NULL, error_type, "Unable to load dynamic library '%s' - %s", libpath, "Unknown reason");
Celse
        php_error_docref(NULL, error_type, "Unable to load dynamic library "%s" - %s", libpath, GET_DL_ERROR());
        GET DI ERROR(); /* free the buffer storing the error */
Vendit
        efree(libpath);
        roturn FAILURE;
    efree(libpath);
```

```
get_module = (zend_module_entry *(*)(void)) Dt_FETCH_SYMBOL(handle, "get_module");
/* Some US prepend to symbol names while their dynamic linker
 * does not do that automatically. Thus we check manually for
* get module. */
if (!get module) (
    get module = (zend module entry *(*)(void)) Ot FETCH SYMBOL(handle, " get module");
if (!get module) (
    i+ (DL FETCH SYMBOL(handle, "zend extension entry") || DL FETCH SYMBOL(handle, " zend extension entry")) {
        DI UNI GAD (handle);
        php error decref(NULL, error type, "Invalid library (appears to be a Zend Extension, try loading using z
       return FAILURE;
    DI UNI GAD (handle);
    php error docref(NULL, error type, 'Invalid library (maybe not a PHP library) '%s'', filename);
    return FAILURE:
module_entry = get_module();
```

```
if ((module_entry = zend_register_module_ex(module_entry)) == NULL) {
    DL UNLOAD(handle);
    return FAILURE;
if ((type == MODULE_TEMPORARY || start_now) && zend_startup_module_ex(module_entry) == FAILURE) {
    DL_UNLOAD(handle);
    return [AILURE;
if ((type == MODBUE LEMPORARY || start now) && module entry->request startup func) {
    if (module_entry->request_startup_func(type, module_entry->module_number) == FAILURE) {
        php_error_docret(NULL, error_type, "Unable to initialize module '%s'", module_entry->name);
        DL_UNLOAD(handle);
        return FAILURE;
return SUCCESS;
```

创建 PHP 的扩展工程

- 1. 扩展骨架生成工具: ext skel
- 2. 编辑 config.m4
- 3. 修改 extension.h 头文件定义扩展的函数
- 4. 修改 extension.c 源文件,实现函数的逻辑
- 5. phpize, configure, make, make install
- 6. 建议:参考其他扩展的源码

编写 PHP 扩展 - 基本类型

```
zval a;
ZVAL_LONG(&a, 1234);
zval b;
ZVAL_DOUBLE(&b, 1234.56);
zval c;
ZVAL_STRING(&c, "hello world");
zval d;
array_init(&d);
zval e;
ZVAL_BOOL(&e, 0);
```

```
a = 1234;
$b = 1234.56;
$c = "hello world";
$d = array();
$e = false;
```

编写 PHP 扩展 - 类型推断

```
zval *value;
Z_TYPE_P(value) == IS_LONG;
Z_TYPE_P(value) == IS_STRING;
Z_TYPE_P(value) == IS_ARRAY;
Z_TYPE_P(value) == IS_DOUBLE;
Z_TYPE_P(value) == IS_TRUE;
Z_TYPE_P(value) == IS_FALSE;
Z_TYPE_P(value) == IS_OBJECT;
```

```
is_int($value);
is_float($value);
is_string($value);
is_array($value);
is_bool($value);
is_object($value);
```

编写 PHP 扩展 - 类型转换

```
convert_to_long(value);
convert_to_string(value);
convert_to_double(value);
convert_to_boolean(value);
```

```
$value = intval($value);
$value = strval($value);
$value = floatval($value);
$value = boolval($value);
```

编写 PHP 扩展 - 参数输入

```
PHP FUNCTION(swoole event add)
{
    zval *cb_read = NULL;
    zval *cb_write = NULL;
    zval *zfd;
    char 'tunc_name = NULL;
    long event flag = 0;

if (zend_parse_parameters(ZEND_NUM_ARGS() ISRMLS_CC, "z|zzl", &zfd, &cb_read, &cb_write, &event_flag) -- FAILURE)
    {
        return;
    }
}
```

编写 PHP 扩展 - 参数输入

b: 布尔值, I:整型(long), d:浮点型(double)

s:字符串(char **, long *), a:数组, z:任意PHP变量

|:可选参数,&:引用

端: 1. 字符串长度PHP7必须为long, PHP5为int

- 2. 整型必须为long,不能是int
- 3. 浮点型必须为double,不能是float

编写 PHP 扩展 - 输出返回值

- 1. 修改 return_value 实现返回值
- 2. 使用 RETURN_* 宏

```
RETURN_LONG(1234);
RETURN_BOOL(0);
RETURN_STRING("hello world");
RETURN_DOUBLE(1234.56);
```

编写 PHP 扩展 - 数组操作

```
zval d;
array_init(&d);
add_assoc_long_ex(&d, "key", sizeof("key")-1, 1234);
add_next_index_long(&d, 1234);
```

编写 PHP 扩展 - 数组添加

```
#define edo_assoc_tong(_erg, _keg, _n) ado_assoc_tong_ex(_arg, _key, strlen(_keg), _n)

#define ado_assoc_oul'(_org, _key) add_assoc_oull_cx(_org, _key, strlen(_key), _n)

#define ado assoc bool( arg. keg, r) add assoc_bool ex( arg, keg, strlen( keg), p)

#define add_assoc_resource(_org, _keg, _r) add_assoc_resource_ex(_org, _keg, strlen(_keg), _n)

#define add_assoc_boole( arg. keg, _r) add_assoc_boole ex( arg. keg, strlen(_keg), _d)

#define add_assoc_strlen(_arg, _keg, _xr) and_assoc_strlex(_arg, _keg, strlen(_keg), _str)

#define add_assoc_strling(_arg, _keg, _str) add_assoc_strling(arg, _keg, strlen(_keg), _str)

#define add_assoc_strling(_arg, _keg, _str, _length) add_assoc_strling(_ex(_arg, _keg, strlen(_keg), _#define add_assoc_strling(_keg, _keg, _strlen(_keg), _#define add_assoc_strling(_keg, _keg, _strlen(_keg), _walue)
```

```
ZEND_APT int add_next_index_long(zval *arg, zend_long n);
ZEND_APT int add_next_index_null(zval *arg);
ZEND_APT int add_next_index_null(zval *arg, in. b);
ZEND_APT int add_next_index_nescorte(zval *arg, zend_resource *n);
ZEND_APT int add_next_index_double(zval *arg, double d);
ZEND_APT int add_next_index_str(zval *arg, zend_string *str);
ZEND_APT int add_next_index_string(zval *arg, const_char *str);
ZEND_APT int add_next_index_string(zval *arg, const_char *str, size . length);
ZEND_APT int add_next_index_zval(zval *arg, zval *value);
```

编写 PHP 扩展 - 数组操作

```
ZEND_API zend_bool ZEND_FASTCALL zend_hash_exists(const HashTable *ht, zend_string *key);
ZEND_API zend_bool ZEND_FASTCALL zend_hash_scr_exists(const HashTable *ht, const than *str, size_t len);
ZEND_API zend_bool ZEND_FASTCALL zend_hash_index_exists(const HashTable *ht, zend_ulong h);
```

```
ZEND API zval* ZEND FASTCALL zend hash find(const HashTable *ht, zend string *key);
ZEND_API zval* ZEND_FASTCALL zend_hash_str_find(const HashTable *ht, const char *key, size_t len);
ZEND_API zval* ZEND_FASTCALL zend_hash_index_find(const HashTable *ht, zend_ulong h);
ZEND_API zval* ZEND_FASTCALL_zend_hash_index_find(const HashTable *ht, zend_ulong h);
```

```
ZEND_API int ZEND_FASTCALL zend_hash_del(HashTable *ht, zend_string *key);
ZEND_API int ZEND_FASTCALL zend_hash_del_ind(HashTable *ht, zend_string *key);
ZEND_API int ZEND_FASTCALL zend_hash_str_cel(HashTable *ht, const char *key, size t len);
ZEND_API int ZEND_FASTCALL zend_hash_str_cel_ind(HashTable *ht, const char *key, size_t len);
ZEND_API int ZEND_FASTCALL zend_hash_index_cel(HashTable *ht, zend_ulong h);
```

编写 PHP 扩展 - 对象操作

```
zval object;
object init ex(&object, my class entry);
char *message = "hello world";
zend_update_property_string(my_class_entry, &object, "message", *irenf("nessage") - 1, message);
zval rv;
zval *res = zend_read_property(my_class_entry, &object, "message", sizeof("nessage") - 1, 1, &rv);

zval args[4];
zval func_name;
zval *retval = NULL;
zval *result = call_user_function_ex(NULL, &object, &func_name, &retval, 4, args, 0, NULL);
```

编写 PHP 扩展 - 更多

- 1. 遍历数组
- 2. 使用资源类型
- 3. 实现一个扩展类
- 4. 引用计数管理
- 5.

编写 PHP 扩展 - 坑

- 1. zend_update_property 是否要对zval增加引用计数
- 2. Immutable 数组修改导致崩溃
- 3. API函数风格不统一,有的是SUCCESS,有的是False

Zend API 的问题

- 1. 大量使用宏
- 2. API 名称太长,参数太多,无法记住
- 3. API 分散在众多 .h 和 .c 文件中
- 4. C语言开发,大量使用指针,容易出错
- 5. 没有任何教程或手册

C or C++

C语言

- 50年历史的古老编程语言
- 面向过程风格,函数夹杂着宏,封装性差
- 使用宏和万能指针
- 未提供数据结构
- 仅适合编写底层软件

C++11

- 现代编程语言
- 面向对象风格,对象属性和方法,封装性好
- 模版泛型编程
- STL容器
- 通用编程

PHP与C++

	PHP	C++
语言类型	动态,弱类型	静态,强类型
计算性能	差	强
执行方式	编译生成中间码,解释执行	编译为机器指令,直接执行
开发效率	高	低
扩展能力	弱(必须依赖扩展)	强

PHPI C++

PHP与C++

- 1. PHP与C++互补性强,可实现动静结合
- 2. C/C++ 一与PHP直接在内存堆栈上实现互调用的语言
- 3. 其他编程语言与PHP互调用需要通过 RPC (远程调用)实现

最大的障碍: Zend API 太难用了



使用C++11 编写PHP扩展

PHP-X

https://github.com/swoole/PHP-X

扩展模块 (C++11)

SAPI

PHP-X (C++11)

PHP扩展

C++嵌入PHP

Variant

Array

String

Object

Resource

Zend API (C)

zval

zend_string

zend_resource

zend_array

Variant

String

Extension

ArgInfo

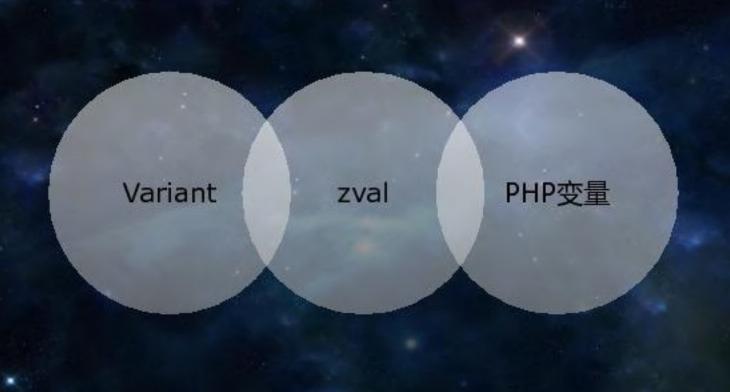
Class

Interface

Array

Object

Resource



Variant: 赋值

```
Variant a = 1234;
Variant b = 1234.56;
Variant c = "hello world";
Variant d = true;
```

Variant: Scalar类型转换

```
long v1 = value.toInt();
double v2 = value.toFloat();
bool v3 = value.toBool();
string v4 = value.toString();
CURL *v5 = value.toResouce<CURL>();
```

Variant:非Scalar类型转换

```
Array arr(var);
Object obj(var);
String str(var);
```

Variant: 类型推断

```
bool ret1 = value.isString();
bool ret2 = value.isArray();
bool ret3 = value.isObject();
bool ret4 = value.isInt();
bool ret5 = value.isFloat();
bool ret6 = value.isBool();
bool ret7 = value.isResource();
```

Variant: 更多API

```
value.type() == IS_STRING;
value.addRef();
value.length();
value.equals(var2);
Variant *v = value.dup();
value.getRefCount();
```

Variant: 生命周期

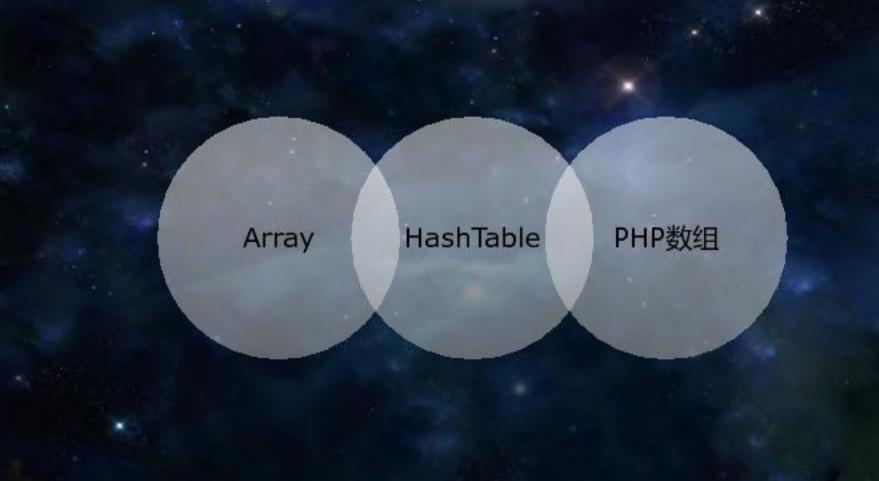
```
function test()
{
     $a = "hello world";
}
```

```
void test()
{
    Variant a("hello world");
}
```

Variant: 堆内存

```
void test()
{
    Variant *a = new Variant("hello world");
}
```

```
void test2(Variant *a)
{
    delete a;
}
```



Array: List

```
Array arr;
arr.append(1234);
arr.append(1234.56);
arr.append(true);
arr.append("hello");
```

Array: Map

```
Array arr;
arr.set("key1",
                 1234);
arr.set("key2", 1234.56);
arr.set("key3",
                false);
arr.set("key4", "world");
```

Array:遍历

```
for (int i = 0; i < arr.count(); i++)
    cout << "key: " << i;
    cout << "value: " << arr[i].toString() << endl;</pre>
for (auto i = map.begin(); i != map.end(); i++)
    Variant key = i.key();
    Variant value = i.value();
    cout << "key: " << key.toString();</pre>
    cout << "value: " << arr[i].toString() << endl;</pre>
```

Array:合并

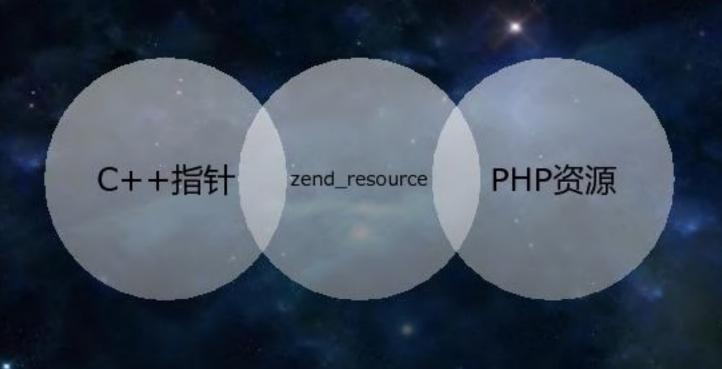
```
Array arr1;
Array arr2;
arr2.append(123);
arr2.append("hello world");
arr2.append(123.05);
arr1.merge(arr2);
```

Array:排序与查找

```
Array arr;
arr.append(123);
arr.append(456);
arr.append(150);
arr.sort();
var_dump(arr);
```

Array:更多API

```
arr.remove("key4");
Variant v1 = arr["key1"];
Variant v2 = arr[0];
int c = arr.count();
bool e = arr.empty();
bool f = arr.exists(1234);
bool g = arr.exists("key5");
```



Resource: 打通 PHP 与 C++

- 1. 将 C++ 指针可保存到 PHP 变量中
- 2. 在 PHP 的函数中互相传递资源变量
- 3. PHP 中调用 C++ 扩展函数,将资源变量传给 C++

Resource: 定义

```
void defineResource()
    PHP::registerResource("String", String_dtor);
static void String_dtor(zend_resource *res)
    String *str = static_cast<String*>(res->ptr);
   delete str;
```

Resource:创建/获取

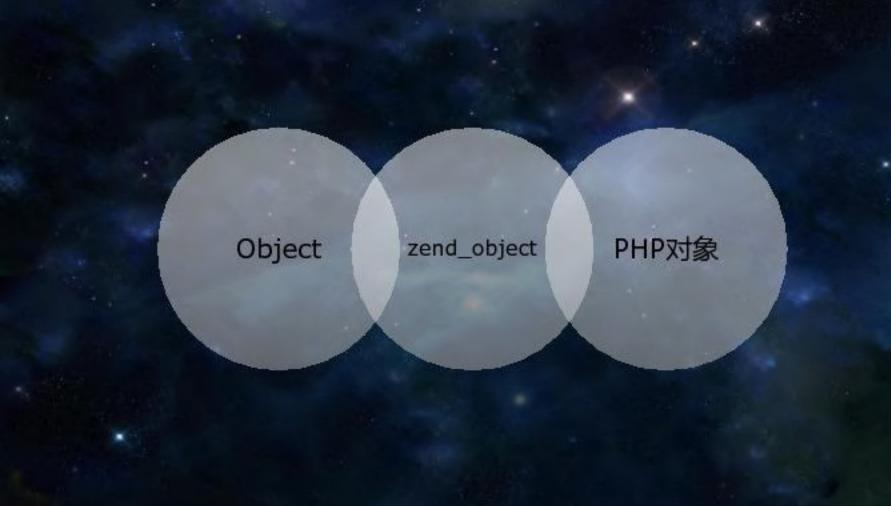
```
void setResource(Object object)
    String *str = new String("hello world")
    Variant var = PHP::newResource("String", str);
   object.set("resource", var);
void getResource(Object object)
    Variant var = object.get("resource");
    String *str = var.toResource<String>("String");
```

常用 API

```
php::global("_GET");
php::constant("PHP_VERSION");
php::echo("hello world");
php::error(E_NOTICE, "notice");
php::var_dump(var);
php::exec("json_encode", arr);
```

调用 PHP 函数

```
Variant ret = php::exec("test");
if (ret.isArray()) {
    Array arr(ret);
} else if (ret.isObject()) {
    Object obj(obj);
```



Object:创建对象/执行方法

```
Object redis = php::newObject("redis");
Variant ret1 = redis.exec("connect", "127.0.0.1");
if (ret1.toBool())
{
    auto ret2 = redis.exec("get", "key");
    cout << "value=" << ret2.toString() << endl;
}</pre>
```

Object:读写对象属性

```
Object object = php::newObject("stdClass");
object.set("name", "Rango");
Varaint name = object.get("name");
```

Object:读写C++指针

```
object.oSet("str", "String", new String("php"));
String *str = object.oGet<CppObject>("str", "String");
```

Object:更多API

```
uint32_t id = object.getId();
string name = object.getClassName();
bool ret1 = object.methodExists("methodName");
bool ret2 = object.propertyExists("propertyName");
```

```
Variant value = Class::get("MyClass", "name");
Class::set("MyClass", "name", Array());
```

编写 PHP7 扩展

```
PHPX_EXTENSION()
    Extension *extension = new Extension("cpp_ext", "0.0.1");
    extension->onStart = [extension]()
        extension->registerConstant("CPP_EXT_VERSION", "0.0.1");
        extension->registerClass(c);
    };
    extension->registerFunction(PHPX_FN(cpp_ext_test));
    return extension;
```

实现扩展函数

```
PHPX_FUNCTION(cpp_ext_test)
    for (int i = 0; i < args.count(); i++) {</pre>
        php::echo("arg[%d] type is %d\n", i, args[i].type());
    Variant v1 = args[0];
    Array arr(v1);
    arr.set(1, "efg");
    retval = arr;
```

实现扩展类

```
Class *c = new Class("myClass");
c->addMethod(PHPX_ME(myClass, test), STATIC);
c->addMethod(PHPX_ME(myClass, test2));
c->addMethod(PHPX_ME(myClass, construct), CONSTRUCT);
c->addProperty("name", "Rango");
c->addConstant("VERSION", 1002);
extension->registerClass(c);
```

实现类方法

```
PHPX_METHOD(myClass, test2)
{
    Variant res = newResource("String", new String("hello"));
    _this.set("resource", res);
    php::error(E_WARNING, "error message.");
}
```

注入phpinfo

gtk					
е, к юдент	Leidene				
suther version	3.0.1				

PHP-X 宏

```
PHPX_FUNCTION(cpp_ext_test)
void cpp_ext_test(Args &args, Variant &retval);

PHPX_METHOD(MyClass, test)
void MyClass_test(Object &_this, Args &args, Variant &retval);
```

```
PHPX_EXTENSION()
Extension* get_module()

PHPX_FN(cpp_ext_test)
"cpp_ext_test", cpp_ext_test

PHPX_ME(myClass, test)
"test", MyClass_test
```

PHP-X 宏

```
PHPX_EXTENSION()
Extension* get_module()
PHPX_FN(cpp_ext_test)
"cpp_ext_test", cpp_ext_test
PHPX_ME(myClass, test)
"test", MyClass_test
```

Makefile编写

```
PHP_INCLUDE = `php-config --includes`
PHP_LIBS = 'php-config --libs'
PHP_LDFLAGS = `php-config --ldflags`
PHP_INCLUDE_DIR = `php-config --include-dir'
PHP_EXTENSION_DIR = `php-config --extension-dir`
PHP_INCLUDE_DIR = "/home/htf/workspace/php-x/include"
cpp_ext.so: cpp_ext.cpp
    c++ -g -o cpp_ext.so -00 -fPIC -shared cpp_ext.cpp -std=c++11 \
    ${PHP_INCLUDE} -I${PHP_INCLUDE_DIR} -I${PHPX_INCLUDE_DIR} ${PHP_LDFLAGS}
install: cpp_ext.so
    cp cpp_ext.so ${PHP_EXTENSION_DIR}/
clean:
    rm *.so
```

编译安装

```
make
sudo make install
echo "extension=cpp_ext.so" >> /path/to/php.ini
php -m
php --ri cpp_ext
php --re cpp_ext
php -r "MyClass::test();"
```



在C++程序中嵌入 PHP ZendVM

C++嵌入动态语言

- Python (boost)
- 2. Lua (游戏)
- 3. JS (Qt)
- 4. PHP Embed (没人用)

C++嵌入PHP

```
#include "sapi/embed/php_embed.h"
int main(int argc, char * argv[])
    PHP_EMBED_START_BLOCK(argc, argv);
    char * script = " print 'Hello World!';";
    zend_eval_string(script, NULL, "Simple Hello\
        World App" TSRMLS_CC);
    PHP EMBED END BLOCK();
    return 0;
```

C++嵌入PHP会带来什么.

- 1. C++程序调用ZendVM提供的功能
- 2. C++程序调用各类PHP扩展提供的功能
- 3. C++程序调用PHP类库、框架

C++嵌入PHP

```
#include "embed.h"
```

```
VM vm(argc, argv);
vm.eval("echo 'Hello World!';");
vm.include("embed.php");
```

```
auto obj = php::newObject("Redis");
auto ret = obj.exec("connect", '127.0.0.1');
```

C++使用Composer包

```
//composer
vm.include("vendor/autoload.php");
auto c = "Illuminate\\Foundation\\Application";
auto app = php::newObject(c);
```



PHP-X实践

建议使用Eclipse



Eclipse

- 自动提示
- 自动补齐
- 语法搜索
- 自动跳转
- 代码即文档

PHP-X

- 1. rocksdb:Facebook开源KV数据库RocksDB的客户端
- 2. Gtk: Gtk图形界面库,用于开发跨平台的桌面软件
- 3. Stdext:标准库扩展,补充 Standard 和 SPL 的不足

PHP-X Gtk

```
$app = new Gtk\Application("test.glade", "window1");
$app setTitle("Power by PHP-X.");
$app->find("button1")->on("clicked", function () use ($app) {
    $app quit();
});
$app->find('button3')->on('clicked', function() use ($app) {
    $input = $app > find("entry1");
   $text = $input getText();
    $app->find('label1')->setText("Value=$text");
});
$app->find('button4')->on('clicked', function() use ($app) {
    $input = $app find("entry1");
    $input > setText('hello world');
});
$app run();
```

PHP-X Gtk

	Power b	y PHP-X.		
	b.e	en:		
PHP大会				
	Value=F	HF大会		
		双取输入信息	2.H	退出程序

PHP-X 想象力

- 1. PHP 可以用 C++ 扩展实现多线程
- 2. PHP-X 支持 Windows 平台,可以开发Windows的PHP扩展
- 3. 程序中大量运算的逻辑可以改为 C++ 扩展实现
- 4. 使用 PHP-X 开发商业软件,避免源码泄漏
- 利用 Facebook/Google/Microsoft/Inter/Tencent 等巨头开源的C++库来扩展 PHP

THANK YOU

- 2018年再见 -

Q & A