

iranphp articles

توسعه PHP
امید متقی راد
omid@oxygenws.com
.....

عنوان مقاله :
نگارنده :
آدرس پست الکترونیک :
تاریخ نگارش :

توسعه PHP :

چکیده :

در این مقاله به بررسی نحوه توسعه PHP می پردازیم. یکی از بزرگترین محاسن برنامه های open source از جمله PHP ، باز بودن متن برنامه آنهاست که به توسعه دهندگان این امکان را می دهد تا برنامه مورد نظر را طبق درخواست ها و نیاز های خودشان تغییر داده و از آن استفاده کنند. پس از خواندن این مقاله شما خواهید توانست توسعه های ساده ای برای PHP نوشته و از آنها در پروژه های خود استفاده کنید.

کلمات کلیدی :

PHP، برنامه نویسی، لینوکس، open source

۱- مقدمه

در این مقاله به بررسی نحوه توسعه PHP می پردازیم. یکی از بزرگترین محاسن برنامه های open source از جمله PHP ، باز بودن متن برنامه آنهاست که به توسعه دهندگان این امکان را می دهد تا برنامه مورد نظر را طبق درخواست ها و نیاز های خودشان تغییر داده و از آن استفاده کنند. به عنوان مثال از تاریخ شمسی در PHP پشتیبانی نمی شود و شما می توانید با توسعه این زبان، برای پشتیبانی این تقویم، آن را برای خود مناسب و قابل استفاده کنید .

پس از خواندن این مقاله شما خواهید توانست توسعه های ساده ای برای PHP نوشته و از آنها در پروژه های خود استفاده کنید. در این مقاله با هم توسعه ای ساده برای PHP خواهیم نوشت .

برای درک مطالب این مقاله احتیاج به داشتن اطلاعاتی در مورد زبان برنامه نویسی C و PHP و روش کامپایل کردن در لینوکس دارید. زبان برنامه نویسی این مقاله C بوده و محیط کاری لینوکس می باشد. پیش فرض آن است که شما توانایی نصب برنامه های مختلف تحت لینوکس را دارید. تمامی کار های این مقاله تحت خط فرمان یا command line انجام می شود و شما باید تسلط و درک کافی در این مورد را نیز داشته باشید .

۲- برنامه های مورد نیاز

برای توسعه PHP احتیاج به نرم افزار هایی داریم که مسئولیت عملیات کامپایل کردن در لینوکس را دارند. لیستی از برنامه های مورد نیاز را در زیر می بینید، این نرم افزار ها را می توانید از آدرس www.gnu.org گرفته و روی سیستم لینوکس تان نصب کنید. در صورتی که از توسعه های معروف لینوکس استفاده می کنید اکثر این برنامه ها را نصب شده دارید .

bison
flex
m4
autoconf
automake
libtool

gcc یا هر نوع کامپایلر دیگر
make

cvs از سایت www.cvshome.org

پس از نصب برنامه های فوق، احتیاج دارید نسخه قابل توسعه PHP یعنی نسخه CVS یا Concurrent Version System آن را از آدرس <http://cvs.php.net> بگیرید و آماده توسعه شوید. برای این کار باید تحت خط فرمان لینوکس تان دستوری مشابه دستور زیر تایپ کرده و به جای X، Y و Z شماره نگارش PHP ای که قصد گرفتن آن را دارید را وارد کنید. با اجرای این دستور متن برنامه های PHP از اینترنت گرفته می شود و با شاخه ای به نام php-src روی دیسک شما ذخیره می شود

```
cvs -d :pserver:cvsread@cvs.php.net:/repository checkout -r php_X_Y_Z php-src
```

۳- توسعه PHP

۳-۱: هدف برنامه و آماده کردن محیط

می خواهیم تابعی به PHP اضافه کنیم به شکل `salam()` که به عنوان ورودی یک رشته را دریافت کرده و رشته ای به صورت `salam STRING` را برگرداند.

به عنوان اولین قدم، به شاخه PHP رفته و سپس به شاخه `ext` آن می رویم و فایلی را در مورد نوع توابعی که قصد توسعه داریم می سازیم. در این فایل که ما نام آن را `salam.def` می گذاریم به ترتیب نوع خروجی توابع، نام تابع، نام و نوع پارامترهای ورودی، یک " " فاصله) به عنوان جدا کننده و رشته ای که توضیح مختصری از آن تابع باشد را می نویسیم. در این فایل، هر تابع را باید در یک خط بنویسیم. در زیر به نمونه فایل مورد نیاز برای توسعه تابع مان توجه می کنیم:

```
string salam(string arg) return "salam ARG"
```

توجه کنید که تمامی مقادیر بعد از پرانتز بسته تا آخر خط، توضیحات تابع می باشند.

PHP برنامه ای به نام `ext_skel` برای آماده سازی مقدمات توسعه دارد، که در شاخه `ext` قرار دارد و شما می توانید به عنوان دومین قدم توسعه، از آن استفاده کنید. برای ادامه توسعه احتیاج داریم از خط زیر استفاده کرده تا مجموعه فایل های مورد نیاز توسط این برنامه ساخته شود.

```
./ext_skel -extname=salam -proto=salam.def
```

با اجرای این دستور شاخه ای به نام مقدار جلوی `-extname` ساخته شده و فایل های مورد نیاز نیز در آن قرار می گیرد. این فایل ها عبارتند از:

```
config.m4
CREDITS
EXPERIMENTAL
salam.c
salam.php
Makefile.in
php_salam.h
شاخه tests
```

۳-۲: توسعه فایل ها

یکی از مهمترین و اصلی ترین فایل هایی که لازم است تغییر دهیم و تمامی کدهای اصلی برنامه ما در آن قرار دارد، فایل `salam.c` می باشد. با باز کردن این فایل، اولین قسمت مهمی که مشاهده می کنید، خطوط زیر می باشد:

```
/* {{{ salam_functions[]
 *
 * every user-visible function must have an entry in salam_functions[]
 */
function_entry salam_functions[] = {
    PHP_FE(confirm_salam_compiled, NULL) /* for testing; remove later */
    PHP_FE(salam, NULL)
    {NULL, NULL, NULL} /* must be the last line in salam_functions[] */
};
/* }}} */
```

این مقادیر توسط فایل `salam.def` که تعریف کرده بودیم ساخته شده و در صورتی که فایل `salam.def` را بدون نقص نوشته باشیم، احتیاجی به ایجاد تغییر در این قسمت نداریم. توجه کنید که اولین تابع معرفی شده یعنی `confirm_salam_compiled` یک تابع برای تست کامپایل شدن یا نشدن سری توابع مان بوده که پس از موفقیت در کامپایل، می توانیم خط مربوطه را حذف کنیم. کدهای مهم بعدی، مشابه بخش زیر خواهد بود که شامل توابع پیش فرض زمان فراخوانی و اتمام کار توابع مورد نظر می باشد. این بخش، اطلاعاتی در مورد نوع، نگارش و نام توسعه ما را نیز داراست.

```
zend_module_entry salam_module_entry = {
```

```
STANDARD_MODULE_HEADER,  
"salam",  
salam_functions,  
PHP_MINIT(salam),  
PHP_MSHUTDOWN(salam),  
PHP_RINIT(salam), /* replace with NULL if no request init code */  
PHP_RSHUTDOWN(salam), /* replace with NULL if no request shutdown code */  
PHP_MINFO(salam),  
"0.1", /* replace with version number for your extension */  
STANDARD_MODULE_PROPERTIES  
};
```

برای تابعی که ما می خواهیم توسعه دهیم باید مقادیر `PHP_RINIT(salam)` و `PHP_RSHUTDOWN(salam)` را با `NULL` جایگزین کنیم، پس کد شما چیزی شبیه مقادیر زیر خواهد شد:

```
zend_module_entry salam_module_entry = {  
    STANDARD_MODULE_HEADER,  
    "salam",  
    salam_functions,  
    PHP_MINIT(salam),  
    PHP_MSHUTDOWN(salam),  
    NULL,  
    NULL,  
    PHP_MINFO(salam),  
    "0.1", /* replace with version number for your extension */  
    STANDARD_MODULE_PROPERTIES  
};
```

سپس توابع `PHP_MINIT(salam)` و `PHP_MSHUTDOWN(salam)` را پیدا کرده و تعیین می کنیم که مقدار `SUCCESS` را برگردانند. پس کد این بخش برنامه ما شبیه کد زیر می شود:

```
PHP_MINIT_FUNCTION(salam) {  
    return SUCCESS;  
}  
PHP_MSHUTDOWN_FUNCTION(salam) {  
    return SUCCESS;  
}
```

سپس به اصلی ترین تابع می رسیم که باید تمام وظایف مورد نظر برای تابع `salam` را در آن توسط زبان C نوشته تا خروجی مناسب را برای ما تولید کند. کدهای شما باید بعد از خط `return` بیاید و خط های بالای آن بدون تغییر بماند (این بدون تغییر ماندن به دلیل آگاهی نداشتن ما برای تغییر آنهاست. صحبت کامل در مورد این خطوط از حوصله این مقاله خارج بوده و فقط کفایت بدانی که این کدها شامل مراحل پردازش مقادیر ورودی تابع است)

```
/* {{{ proto string salam(string arg)  
   return \"salam ARG\" */  
PHP_FUNCTION(salam)  
{  
    char *arg = NULL;  
    int argc = ZEND_NUM_ARGS( );  
    int arg_len;  
  
    if (zend_parse_parameters(argc TSRMLS_CC, "s", &arg, &arg_len)  
        == FAILURE)  
        return;  
  
    // YOUR CODES SHOULD COME HERE  
}
```

```
/* }}} */
```

تابع `PHP_FUNCTION` برای `PHP` نام یک تابع را مشخص می کند و مقادیر داخل این تابع، اعمالی را که تابع `PHP` در هنگام فراخوانی آن در برنامه های `PHP` باید انجام دهد را نشان می دهد. همانطور که گفته شد، بعد از چند خط اول، کدهای ما قرار خواهد گرفت و برنامه ما به صورت زیر در خواهد آمد:

```
PHP_FUNCTION(salam) {
    char *arg = NULL, *sout;
    int argc = ZEND_NUM_ARGS( );
    int arg_len, len;

    if (zend_parse_parameters(argc TSRMLS_CC, "s/", &arg, &arg_len)
        == FAILURE)
        return;

    strcpy(sout, "salam ");
    strcat(sout, arg);
    len = strlen(sout);
    RETURN_STRINGL(sout, len, 1);
}
```

ابتدا متغیرهایی به نام `sout` از نوع کاراکتر و `len` از نوع عددی تعریف کرده ایم. در خط بعد از `return` مقدار `"salam "` را در متغیر `sout` کپی کرده و سپس مقادیر `sout` و `arg` که رشته ورودی ما می باشد را به هم متصل می کنیم و به `sout` انتساب می دهیم. سپس طول `sout` را به دست آورده و از طریق تابع `(RETURN_STRINGL)` این تابع جزو توابع `ZEND API` بوده و برای برگرداندن مقادیر استفاده می شوند، این توابع محدود و تعیین شده می باشند) مقدار رشته و طول آنرا بر می گردانیم. سومین متغیر در تابع `RETURN_STRINGL` مشخص کننده صحت خروجی یا وجود اشکال در خروجی می باشد.

هم اکنون توسعه تابع ما به پایان رسیده، اما برای امکان کامپایل تابع توسعه داده شده لازم است مقادیر لازم برای درست کامپایل شدن تابع را تصحیح و کامل کنیم. پس فایل `salam.c` را ذخیره کرده و می بندیم و فایل `config.m4` را برای اعمال تغییرات باز می کنیم. در این فایل، از قبل، خطوط لازم وجود دارند ولی به صورت توضیح یا `comment` در آمده اند که شما باید این خطوط را از این حالت خارج کنید. پس از این کار خطوطی دقیقاً مشابه خطوط زیر خواهید داشت (توجه کنید که هیچ علامتی در ابتدای این خطوط نباشند):

```
PHP_ARG_ENABLE(salam, whether to enable salam support,
[ --enable-salam Enable salam support])
```

توسعه تابع ما به پایان رسید، حال نوبت کامپایل کردن `PHP` است تا بتوانید از تابعی که نوشتیم در آن استفاده کنیم. در بخش بعد به این مسئله می پردازیم.

۴- کامپایل PHP

قبل از اجرای دستورات کامپایل، برای اینکه `PHP`، توسعه جدید ما را بشناسد، لازم است دستور `buildconf` را که در شاخه اصلی `PHP` قرار دارد، اجرا کنیم. این دستور را باید به شکل زیر وارد کنیم:

```
./buildconf
```

سپس باید برنامه را با توسعه ای که نوشتیم کامپایل کنیم. این کار در `PHP` شبیه خیلی از برنامه های دیگر تحت لینوکس انجام می شود. به این صورت که شما باید مقداری مشابه مقدار `--enable-salam` را در هنگام `configure` کردن `PHP` اعمال کنید. سپس سری دستورات `make` را برای نصب `PHP` اجرا می کنیم. دستوراتی که اجرا می کنیم به ترتیب، شبیه دستورات زیر خواهد بود، توجه کنید که تمامی این دستورات باید در شاخه اصلی `PHP` اجرا شوند:

```
./configure --enable-salam
make
make install
```

اکنون PHP شما با تابع شما توسعه پیدا کرده است. حال نوبت آزمایش تابع است تا مشاهده کنیم که درست کار می کند. برای این منظور فایلی با توسعه php نام test ساخته و مقادیر زیر را در آن می نویسیم:

```
<?php
    echo salam('donya');
?>
```

سپس، در خط فرمان دستور زیر را اجرا کنید. می بینید که مقدار "salam donya" در خروجی چاپ خواهد شد.

```
/usr/bin/php -q test.php
```

۵- جمع بندی

در این مقاله یاد گرفتیم که چطور از برتری های برنامه های open source استفاده کرده و از آنها برای رسیدن به اهداف مان استفاده کنیم و چطور آنها را توسعه دهیم تابعی برای PHP ساختیم، که قبلا در آن وجود نداشت و از آن در برنامه ای که با PHP نوشتیم استفاده نمودیم. به یاد داشته باشید که این مقاله، فقط توضیحات مختصری در مورد توسعه PHP ارائه کرده و قصد آشنا کردن شما با این مقوله را داشته است.