



testFest 2008

Costa Rica PHP

Pablo Víquez ZCEPHP5
<pablovrogers@gmail.com>



teamwork

teamwork

Eventos

testFest 2008
Building the best

- Reunion mensual del grupo
 - Auditorio de la Universidad Latina
 - Martes 20 de Mayo, 6pm

¿Porque probar?

- Posiblemente ya lo este haciendo.
- Ayuda a encontrar errores en el código y en la aplicación.
- Las pruebas hacen que se escriba mejor código.
- Ahorran tiempo al final.
- Proveen una documentación mínima de lo que se espera que el código haga.

¿Porque probar?

- Hacemos reutilización de código y esta reutilización normalmente requiere cambios.
- “Darle una ojeada” no es suficiente.
- Muchas veces el código esta hecho para una situación especifica.
 - Cuando se expande o se cambia algo falla.
- Interacción del código se subestima.
 - Errores en una función puede afectar otras.

Pruebas automáticas

- Pruebas de integración del sistema.
 - ¿Funciona el sistema?
- Pruebas de funciones
 - ¿Funciona el API?
- Unidades de prueba
 - ¿Funciona este código?
- Pruebas de aceptación
 - ¿Hace lo que el cliente desea?
 - Genera pruebas con base en los requerimientos.
- Pruebas de regresión
 - ¿El sistema todavía funciona como se espera?
 - Verifica las entradas contra las salidas esperadas.

Introducción a phpt

- ¿Que son?
 - Un archivo phpt es un script pequeño usado para probar la funcionalidad de PHP. Son usadas para:
 - Asegurar que las nuevas versiones funcionen y hagan las cosas que las versiones pasadas hacen.
 - Ayudar a encontrar “pulgas” en las versiones actuales.
- ¿Que pruebo?
 - Se escriben pruebas a una o mas funciones disponibles en PHP, ya sea a las funciones básicas del lenguaje o a funciones en alguna de las numerosas extensiones de PHP.
- ¿Que se espera que haga?
 - Básicamente tratar de quebrar la función. Debería no solo probar con los parámetros normales, sino también por casos extremos. Se pueden generar errores intencionalmente.

Convención de nombres

- Pruebas a “pulgas” conocidas

- bug{bug_id}.phpt *bug36428.phpt*

- El “bug” 36428 hace una prueba para PDO, donde este genera un mensaje de error incorrecto en la función PDO::fetchAll().

- Prueba funciones de PHP

- {nombreFuncion}.phpt *empty.phpt*

- Pruebas generales a las extensiones

- {nombreExt}_{num}.phpt *pdo_003.phpt*


Formato básico

- Una archivo phpt con una prueba sencilla consiste en 3 secciones.
 - Nombre
 - Entrada
 - Salida esperada.

TIP: Asegúrese que la salida sea verificable

Prueba básica - EXPECT

```
--TEST--  
strstr() function - basic test for strstr()  
--FILE--  
<?php  
$trans =  
    array(  
        "hello" => "hola",  
        "I"      => "Yo",  
        "all"    => "todos",  
        "world" => "mundo",  
        "said"  => "dije",  
        "hi"    => "tuanis");  
var_dump(strstr("# hi all, I said hello world! #", $trans)); ?>  
--EXPECT--  
string(32) "# tuanis todos, Yo dije hola mundo! #"
```



TIP: Siempre
deje una línea
en blanco al
final del archivo.
CVS friendly

Pruebas avanzadas - EXPECTF

```
--TEST--
```

```
Testing str_shuffle.
```

```
--FILE--
```

```
<?php
```

```
$s = '123';
```

```
var_dump(str_shuffle($s));
```

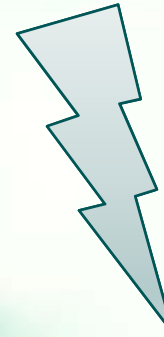
```
var_dump($s);
```

```
?>
```

```
--EXPECTF--
```

```
string(3) "%s"
```

```
string(3) "123"
```



str_shuffle:

Mezcla un string aleatoriamente.

string str_shuffle (string str)

%a para N caracteres (al menos uno)

%i enteros

%d solo numeros

%f punto flotante

%c un carácter

%x valores hexadecimales

%w cualquier numero con espacios

%e para DIRECTORY_SEPARATOR ('\ o '/')

Pruebas avanzadas - EXPECTREGEX

```
--TEST--
```

```
Mas pruebas con str_shuffle
```

```
--FILE--
```

```
<?php
```

```
$s = '123';
```

```
var_dump(str_shuffle($s));
```

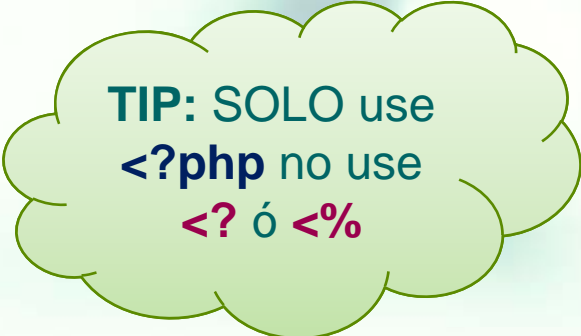
```
var_dump($s);
```

```
?>
```

```
--EXPECTREGEX--
```

```
string\(3\) "[123]{3}"
```

```
string\(3\) "123"
```



TIP: SOLO use
<?php no use
<? ó <%

Pruebas avanzadas

```
--TEST--
Validacion de salidas con md5
--FILE--
<?php
$dest= 'test.png' ;
@unlink($dest);
imagepng(imageCreateTruecolor(640, 100), $dest);
var_dump(md5_file($dest));
@unlink($dest);
?>
--EXPECT--
string(32) "13e1f304542aec94b80ef8271ce272b1"
```

TIP: Salidas grandes pueden ser verificadas con MD5.

Quando use archivos, asegúrese de borrarlos antes y después

¿Cómo ejecutar las pruebas?

- Se ejecutan con “run-test.php”
 - Este se encuentra en la distribución del código fuente.
- Este ejecuta el archivo phpt y da su resultado.
- En caso de encontrar una “pulga” automáticamente genera un reporte.
 - Siempre acepte enviar el resultado si le pregunta.

Generación Automática de Pruebas

- Necesita:
 - Copia del código fuente de PHP.
- Dentro de los archivos fuente:
 - `/scripts/dev/generate_phpt.php`

Generación Automática de Pruebas

```
--TEST--
Test cos() : basic functionality
--FILE--
<?php
/* Prototype : proto float cos(float number)
 * Description: Returns the cosine of the number in radians
 * Source code: ext/standard/math.c
 * Alias to functions:
 */

/*
 * add comment here to indicate details of what this testcase is testing in
particular
 */
echo "*** Testing cos() : basic functionality ***\n";
// Initialise all required variables
$number = 10.5;
// Calling cos() with all possible arguments
var_dump( cos($number) );
echo "Done "
?>
--EXPECTF--
Expected output goes here
Done
```

teamwork

Generación Automática de Pruebas

- Sintaxis:

- `php generate_phpt.php`
 - `-s <php_source_dir>`
 - `-f <function_name>`
 - `-b |`
 - `-e |`
 - `-v`
 - `[-i <include_file>]`

Generación Automática de Pruebas

■ Opciones

- -s – Donde se encuentra el código fuente
- -f – Nombre de la función
- -b – Genera un test básico
- -e – Genera un test de error
- -v – Un test(s) de variación
- -i – Bloque de PHP. Si esta presente no se genera código.
- -h – Imprime la ayuda (Estos comandos)

Tips

- Mantenga las pruebas simples.
 - Pruebe el min/max numero de args.
- No use include files, hace difícil el “debug”
- Escriba las pruebas de manera legible, recuerde que otras personas las van a leer.
- Si la documentación dice que el comportamiento es impredecible, no escriba una prueba para esa función.

Tips

- Mantenga las pruebas simples.
 - Pruebe el min/max numero de args.
- No use include files, hace difícil el “debug”
- Escriba las pruebas de manera legible, recuerde que otras personas las van a leer.
- Si la documentación dice que el comportamiento es impredecible, no escriba una prueba para esa función.

¡Iniciamos!

testFest 2008
Building the best

- No apunté nada! ¿que hago? ☺
 - www.costaricaphp.org/testfest/presentacion.pdf
- Esta presentación se encuentra en el desktop de las máquinas.



teamwork