



## Uso de Componentes PHP

# Formularios a Tope

Al igual que otros lenguajes, PHP el lenguaje de guiones para la Web, está repleto de componentes prefabricados. En este artículo echaremos un vistazo a aplicaciones prácticas usando como ejemplo formularios de gestión.

POR RICHARD SAMAR

Los programadores, y sobre todo sus clientes, están siempre interesados en resultados rápidos. La necesidad de trabajar en proyectos en paralelo y unos plazos de cierre muy ajustados es particularmente duro para los desarrolladores web. Debido a esto los desarrolladores profesionales gastan más tiempo buscando como afrontar la tarea que tienen entre manos que escribiendo miles de funciones auxiliares. PHP [2] soporta esta filosofía en muchas áreas de desarrollo de aplicaciones, pero desgraciadamente esto no lo sabe todo el mundo. Y esto es algo que intentaré arreglar en este artículo.

La idea tras Pear, la “PHP Extension and Application Repository” (Repositorio de aplicaciones y extensiones de PHP) surge en paralelo con la publicación de PHP 4 en 1.999. Muchos programadores vieron esto como una analogía con el CPAN de PERL. Pero donde CPAN ofrece más cantidad que

The image shows a printed application form titled "Application for Residence in New Zealand" from the "New Zealand Immigration Service". The form is held together by a paperclip. It features a red-tinted background image of a family. The form includes sections for "Personal Details" and "Principal applicant" with various input fields and checkboxes. The form is titled "Residence" in large vertical text on the right side.

calidad, los mantenedores de la librería de clases de PHP habían puesto el punto de mira en la calidad de su colección. En la fecha en que apareció PHP 4.3m, Pear dejó el estado de desarrollo beta y ahora es un componente esencial de cualquier distribución Linux que incluya PHP.

## El navegador de paquetes muestra las categorías de aplicaciones.

Cuando en PHP se habla de componentes, se está haciendo referencia a los paquetes. Estos están organizados, por

categorías de aplicación, en un árbol, con un nombre subrayado. El denominado *navegador de paquetes* [4] en la Web muestra esta estructura de árbol (véase la figura 1). En este artículo veremos las capacidades de Pear en la práctica, utilizando el paquete *QuickForm* como ejemplo. *QuickForm* facilita el uso de formularios, la principal interfaz para el usuario y de uso obligatorio en las aplicaciones Web. *QuickForm* ha sido asignado a la categoría *HTML* y así puede ser localizado bajo *HTML\_QuickForm*. Tal y como hemos dicho con

anterioridad, se puede utilizar el navegador para determinar el ámbito de aplicación, la versión, el historial de cambios, obtener estadísticas y dependencias con otro paquetes.

## Instalando PEAR y QuickForm

Si PHP 4.3 o posterior está instalado en nuestro sistema, con casi toda seguridad que lo estará también Pear. En ese caso, no necesitaremos realizar los pasos de instalación. En el improbable caso de que PHP haya sido explícitamente configurado y compilado con la opción `--without-pear`, el administrador necesitará recompilarlo sin esta opción. Si su distribución aún usa una versión antigua, puede dar un rodeo, como alternativa a la actualización de PHP, para obtener el instalador de paquetes Pear, el cual es esencial para las tareas de configuración de paquetes. Para hacer esto, abra una conexión a Internet y teclee lo siguiente en una terminal:

```
lynx -source http://go-pear.org
| php
```

Esta línea descarga un guión PHP. En cuanto lo tengamos en nuestro disco duro, lo ejecutamos y se encargará de todo automáticamente. El así llamado `include_path` en el fichero de configuración PHP (normalmente en `/etc/php.ini`) ha de estar correctamente configurado para evitar problemas cuando se añadan componentes Pear. Si PHP está instalado en `/usr/local` lo normal es que Pear se configure para que esté en `/usr/local/lib/php/`. Si no queremos andar arreglando errores de instalación, un vistazo rápido al fichero de configuración nos revelará si las entradas están apuntando a los directorios correctos.

## Visualización de Paquetes

La orden `pear list` muestra en la terminal una lista de los paquetes

```
$pear list
Installed packages:
=====
Package Version State
Archive_Tar 0.9 stable
Console_Getopt 1.0 stable
DB 1.3 stable
```

Podemos teclear `pear install HTML_QuickForm` en una terminal para instalar el paquete automáticamente. Si la conexión a Internet se viniera abajo en cualquier momento, podemos utilizar el navegador de Paquetes de la siguiente manera, para recuperar el paquete, resolver cualquier dependencia, en este caso `HTML_Common`, e instalar la colección:

```
$ pear install /Path/to/Down
load/HTML_Common-1.2.1.tgz
install ok: HTML_Common 1.2.1
$ pear install /Path/to/Down
load/HTML_QuickForm-3.0.tgz
install ok: HTML_QuickForm 3.0
```

## Listado 1: emailformular.php

```
01 <?php
02 require_once
   'HTML/QuickForm.php';
03
04 // Muestra la versión de
   QuickForm
05 print 'PEAR::HTML_QuickForm
   Version ';
06 print
   HTML_QuickForm::apiVersion() .
   '<br/><br/>';
07
08 $myForm = new
   HTML_QuickForm('Formulario de
   Email', 'POST');
09 $myForm->addElement('header',
   '', 'Datos Personales');
10
11 $myForm->addElement('text',
   'textName', 'Apellido:');
12 $myForm->addElement('text',
   'textFirstname', 'Nombre:');
13 $myForm->addElement('text',
   'textEmail', 'Email:');
14 $myForm->addElement('submitButton',
   'submitButton', 'Entrar
   datos');
15
16 $name =&
   $myForm->getElement('textName'
   );
17 $name->setMaxLength(30);
18 $name->setSize(30);
19
20 $vname =&
   $myForm->getElement('textFirst
   name');
21 $vname->setMaxLength(20);
22 $vname->setSize(30);
23
24 $email =&
   $myForm->getElement('textEmail
   ');
25 $email->setMaxLength(50);
26 $email->setSize(30);
27
28 // Añade las reglas de
   validación
29 $myForm->addRule('textName',
   'Por favor indique el apellido
   ', 'obligatorio');
30
   $myForm->addRule('textFirstnam
   e', 'Por favor indique el
   nombre', 'obligatorio');
31 $myForm->addRule('textEmail',
   'Por favor indique la
   dirección de email',
   'obligatorio');
32 $myForm->addRule('textEmail',
   'Email incorrecto', 'email');
33 $myForm->addRule('textEmail2',
   'Por favor indique la
   dirección de email ',
   'obligatorio');
34 $myForm->addRule('textEmail2',
   'Email incorrecto ', 'email');
35
36 // También es posible validar
   del lado del cliente usando
   JavaScript
37 //
   $myForm->addRule('textEmail',
   'Email invalid', 'email',
   NULL, 'client');
38
39 // El formulario se congela si
   la validación es OK
40 if ( $myForm->validate() )
41 {
42   print '¡Gracias! Sus datos
   son: ';
43
   $myForm->removeElement('submit
   Button');
44   $myForm->freeze();
45 }
46
47 // Display form
48 $myForm->display();
49
50
51 ?>
```

A continuación, tecleamos *pear list* otra vez para que muestre los paquetes instalados. Más tarde podemos introducir periódicamente *pear upgrade-all* para actualizar cualquier paquete que tengamos instalado. *pear uninstall* seguido del nombre del paquete desinstalará un paquete. Y si estamos interesado en más detalles de la configuración general de Pear, tecleamos *config-show* y se nos facilitará esa información.

## HTML\_QuickForm en Acción

El paquete *PEAR::HTML\_QuickForm* (nótese el peculiar modo de nombrar con el símbolo de los dos puntos) facilita funciones para la creación, validación y manejo de formularios en HTML. Es fácil de manejar y extremadamente flexible a la vez. además de su facilidad para obtener resultados sencillamente, el paquete también facilita interfaces para los motores de plantilla más conocidos, como Smarty [5]. el listado 1 muestra un pequeño ejemplo. El paquete Pear utiliza un enfoque de programación orientada a objetos, tal como suele entenderse. El formulario de prueba requiere al usuario que introduzca el apellido, el nombre de pila y la dirección de correo electrónico. Se realiza una validación para fallos en la entrada de datos, la experiencia diaria de los programadores revela que hay que incluir validaciones, incluso con la más simple de las estructuras. La estructura *require\_once HTML/QuickForm.php* de la línea 2 contiene el paquete Pear requerido. En este punto no es necesario un nombre de ruta absoluto, ya que PHP la buscará

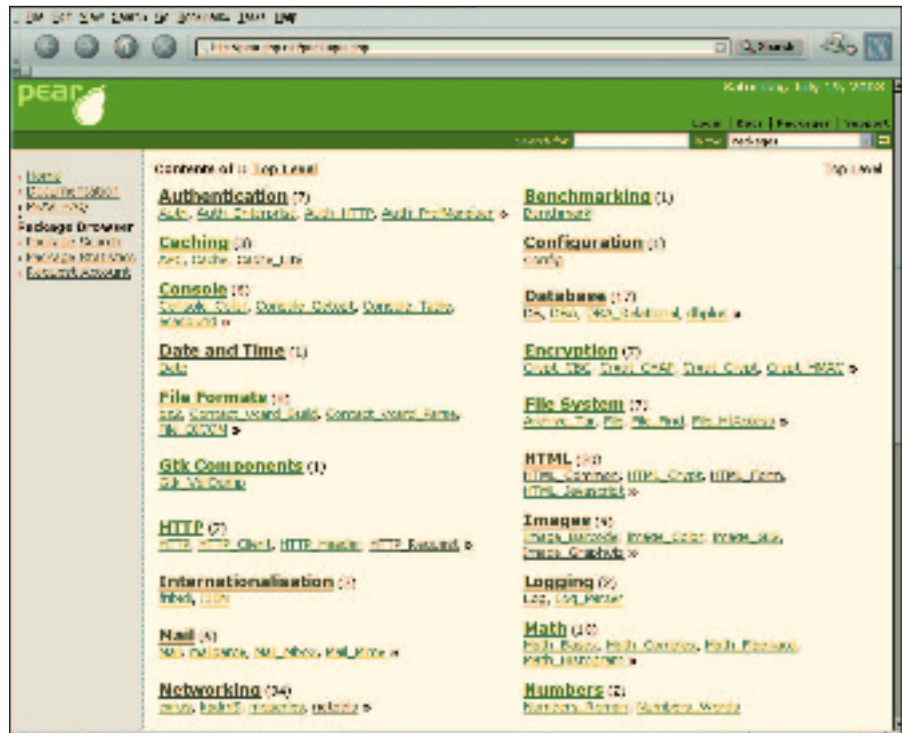


Figura 1: En la jerga de PHP los componentes se denominan paquetes. El navegador de paquetes basado en Web muestra los paquetes en una vista de árbol, ordenados por categorías de aplicaciones.

automáticamente en el *include\_path* configurado anteriormente. De nuevo esto es un reflejo de la estructura de Pear: *QuickForm* ocupa una posición debajo de *HTML* en la jerarquía y *HTML* es un directorio físico.

## Un Derroche de Funciones

El API *QuickForm* es de rango amplio y extremadamente funcional, especialmente desde la versión 3. Por tanto tiene sentido conseguir la salida de *apiVersion()* en la línea 6. Esto es conocido como método estático, algo que debería resultar familiar a la mayoría de los desarrolladores que usan un lenguaje orientado a objetos: Cuando usan un operador con doble signo de dos puntos para llamar a un método de esta clase, no se necesita inicialización. Se pasan dos parámetros a la estructura en la línea 8: *EmailFormular* identifica el formulario y posteriormente se utilizará como el nombre del formulario en el código HTML. *POST* especifica el método *HTTP POST* para la transmisión de datos. Se pueden incluir otros parámetros y modificadores si son necesarios. Por ejemplo, puede usarse un tercer parámetro para especificar el documento objeto que será generado. Como en este caso el parámetro ha sido

omitido, se supondrá el documento actual como el objetivo.

## Una cabecera y cajas de texto

Una cabecera, como la de la línea 9, es útil para identificar más fácilmente el documento y conseguir formularios impecables. Después vienen tres elementos del tipo de formulario *text*, que cualquier navegador Web dibujará como una caja de texto HTML. El segundo parámetro identifica el elemento, mientras que el tercero especifica el texto que aparecerá junto a la cajas en la página Web; esta es a la que se refiere como una etiqueta. Para el botón de enviar de la línea 14 están disponibles los siguientes tipos de elementos:

- caja de verificación
- imagen
- oculto
- contraseña
- radio
- seleccionar
- texto
- área de texto

Normalmente, en un formulario de texto, necesitaremos especificar la longitud de los campos. Para hacer esto así los desarrolladores pueden usar *getElement()* que devuelve una referencia a cada elemento y los asigna a la correspondiente variable. En

Tabla 1: Tipos de validaciones para *addRule()*

Entrada	Significa
required	El campo debe ser rellenado
maxlength	Longitud máxima del campo
minlength	Longitud mínima del campo
email	Dirección válida de correo electrónico
lettersonly	La entrada solo debe contener letras
numeric	La entrada solo debe contener números
regex	La entrada debe coincidir con una expresión regular

nuestro ejemplo, de la línea 16 a la 26 se ajusta la longitud visible de todos los campos a 30 caracteres, aunque la longitud máxima sea distinta.

## Validación de Campos

Gracias a *QuickForm* es bastante fácil validar las entradas de los usuarios. El método *addRule()* le permite asignar un número arbitrario de reglas a cada elemento. Los elementos se identifican por nombres únicos.

Las facilidades de validación son extremadamente amplias, ya que el método tiene muchos parámetros opcionales. Las líneas de la 29 hasta la 34 en el listado 1 muestra un caso muy simple, pasando el texto de error como el segundo y el tipo de validación como tercer parámetro. La tabla 1 muestra una visión general de los tipos de validación más importantes (véase la Tabla 1). Dependiendo del tipo, el programador deberá pasar al método *addRule()* un valor numérico, una expresión regular o nada como cuarto parámetro.

Nuestro ejemplo implemente la validación de campos desde el lado del

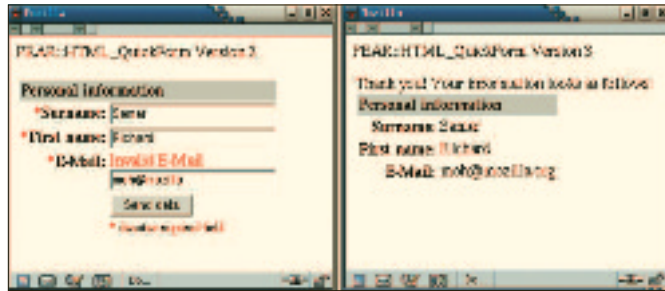


Figura 2: Gracias a *QuickForm*, validar direcciones de correo electrónico es una tarea fácil. Los campos obligatorios se especifican como tales y una entrada errónea genera el correspondiente mensaje de error.

servidor. El comentario en la línea 37 también indica la posibilidad de realizar la validación del lado del cliente usando Javascript. Entonces la instrucción *if* usa *validate()* para comprobar si cada campo ha sido correctamente validado. Luego, *validate()*, devuelve *TRUE* y congela todo el formulario. La característica de *freeze()* es que muestra los datos de entrada, sin permitir editarlos.

Esto es muy práctico, ya que permite al usuario crear una copia impresa, después de completar la entrada de datos. El botón *Enviar* se elimina dentro de la directiva *if*, pues no tendría ningún sentido mostrar el botón después de que la entrada de datos se haya terminado satisfactoriamente. Finalmente el

método más importante aparece en la línea 49: Se llama a *display()* para que muestre el formulario. La figura 2 se puede ver el resultado de todo este esfuerzo, con y sin errores.

## Más Flexibilidad

La funcionalidad proporcionada por el Listado 1 es aceptable para lo que es, pero el aspecto del formulario (véase la Figura 2) definitivamente no tiene nada de especial. Además, puede que prefiramos personalizarlo en vez de utilizar los mensajes convencionales. Esto es fácil de hacer gracias al alto grado de flexibilidad que facilita *QuickForm*, tal como se muestra en otro ejemplo en el Listado 2. Éste crea un simple for-

### Listado 2: creditcard.php

```

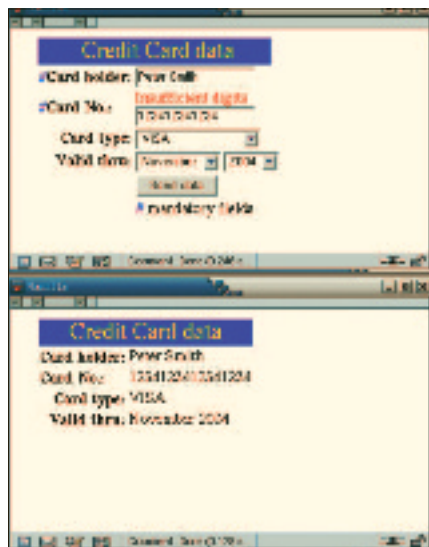
01 <?php
02 require_once
   'HTML/QuickForm.php';
03
04 // Plantilla para la cabecera
05 $headerTemplate = '<tr><td
   style="white-space: nowrap;
   background-color: blue;"
   align="center" ';
06 $headerTemplate .=
   ' valign="top"
   colspan="2"><font size="5"
   color="yellow"><header></font>
   </td></tr>';
07
08 // Plantilla para los campos
   de texto (miembro de y número)
09 $elTemplate = '<tr><td
   align="right" valign="top">';
10 $elTemplate .= '<!-- BEGIN
   required --><font
   color="blue"><b>#</b></font><!--
   END required
   --><b>{label}</b></td>';
11 $elTemplate .= '<td
   valign="top" align="left"><!--
   BEGIN error --><span
   style="color: #ff0000">';
12 $elTemplate .=
   '{error}</span><br /><!-- END
   error --></td></tr>';
13
14 // Instanciar el formulario
15 $myForm = new
   HTML_QuickForm('CreditcardForm
   ', 'POST');
16 // Nuevo texto para los campos
   obligatorios
17 $myForm->setRequiredNote('<font
   color="blue"><b>#</b></font>
   are mandatory');
18
19 // Añadir la cabecera y
   configurar la nueva plantilla
20 $myForm->addElement('header',
   '', 'Creditcarddata');
21
22 $myForm->setHeaderTemplate($he
   aderTemplate);
23 // Añadir campos de texto y
   configurar la nueva plantilla
   para cada uno
24 $myForm->addElement('text',
   'textCardholder',
   'Cardholder:');
25 $myForm->addElement('text',
   'textCardnumber',
   'Cardnumber:');
26
27 $myForm->setElementTemplate($e
   lTemplate, 'textCardholder');
28
29 $myForm->setElementTemplate($e
   lTemplate, 'textCardnumber');
29 // Matriz para los tipos de
   tarjeta de credito
30 $cardtypes = array( 'visa' =>
   'VISA', 'master' =>
   'EuroCard', 'amex' =>
   'American Express',);
31 // Menu desplegable para los
   tipos de tarjeta de credito
32 $myForm->addElement('select',
   'selectCardtypes',
   'cardtype:', $cardtypes);
33
34 // Matriz para mese y años
35 $months = array (

```

## ¿Qué es Pear? ¿Qué es PECL?

PHP es un lenguaje extremadamente universal que puede fácilmente integrar librerías de programación y así implementar interfaces a bases de datos, herramientas gráficas, analizadores gramáticos (parsers) para XML y muchas otras cosas. Esto que ha hecho a PHP popular y que la mayoría de la comunidad comenzara a ponerse a trabajar en el desarrollo de extensiones, ésta es una de las ventajas más obvias del Código Abierto! Pero para mantener la extensión del paquete de PHP en un tamaño razonable, solo se proporcionan de manera predeterminada con PHP las extensiones más importantes y más usadas.

mulario para la entrada de los datos de una tarjeta de crédito. Los desarrolladores pueden utilizar plantillas para modificar el color, formato y el orden de los elementos individuales del formulario a su gusto. Estas plantillas están definidas en dos variables de cadena, para la cabecera y para las cajas de texto, al comienzo del Listado 2. La



**Figura 3:** *QuickForm* es extremadamente flexible cuando hay que alinear campos y asignar etiquetas y colores. Las plantillas se pueden usar para complementar casi cualquier requisito.

cadena entre corchetes define la parte del texto del elemento actual y puede ser colocado donde quiera. Para campos de texto, `{label}` indica el nombre, como *Miembro de*, y `{elemento}` el pro-

pio campo de texto. Las áreas encerradas en los comentarios HTML contienen los mensajes de error para los campos obligatorios y los símbolos. En el último caso el asterisco rojo es reemplazado por un diamante azul. Como un formulario está, predeterminadamente, empotrado en una tabla, solo tiene que asegurarse que las etiquetas internas de la tabla están bien configuradas. No obstante, podría plantearse usar una plantilla de formulario sin una tabla. El método `setRequiredNote()` de la línea 17 reemplaza los mensajes estándar con mensajes personalizados. Adicionalmente `setHeaderTemplate()` y `setElementTemplate()` registran las plantillas definidas con los elementos.

## Menús Desplegables

Los menús desplegables (elemento tipo `select`) se definen como valores en las áreas apropiadas desde la línea 30 hasta la 41. El área `$cardtypes` es el cuarto parámetro de `addElement()` en la línea 32. Este mecanismo posibilita añadir nuevos valores rápidamente en un momento posterior sin sacrificar la

## Listado 2: creditcard.php (continuación)

```

36 '01' => 'Enero', '02' =>
    'Febrero', '03' => 'Marzo',
37 '04' => 'Abril', '05' =>
    'Mayo', '06' => 'Junio',
38 '07' => 'Julio', '08' =>
    'Agosto', '09' =>
    'Septiembre',
39 '10' => 'Octubre', '11' =>
    'Noviembre', '12' =>
    'diciembre'
40 );
41 $years = array ( '2003' =>
    '2003', '2004' => '2004',
    '2005' => '2005' );
42
43 // Crear el grupo de elementos
    para meses y años
44 $validTo[] =
    &HTML_QuickForm::createElement
    ('select', 'selectValidMonth',
    NULL, $months);
45 $validTo[] =
    &HTML_QuickForm::createElement
    ('select', 'selectValidYear',
    NULL, $years);
46
47 // Grupo de elementos creados
    para meses y años
48 $myForm->addGroup($validTo,
    'validToGroup', 'Valid to:');
49
50 // Add Submit button
51 $myForm->addElement('submit',
    'submitButton', 'Submit Data');
52
53 // El número de la tarjeta de
    credito tiene 16 dígitos
54 $cardnumber =&
    $myForm->getElement('textCardn
    umber');
55 $cardnumber->setMaxLength(16);
56
57 // Reglas de validación: Los
    dos campos de texto debes
    estar ocupados
58 // El número de la tarjeta de
    credito debe ser numérico y
    comprende 16 dígitos
59
60 $myForm->addRule('textCardhold
    er', 'Por favor introduzca
    miembro de', 'Obligatorio');
61
62 $myForm->addRule('textCardnumb
    er', 'Número de tarjeta
    invalido', 'numérico');
63
64 $myForm->addRule('textCardnumb
    er', 'Número de tarjeta
    demasiado corto', 'longitud
    mínima', 16);
65
66 // Freeze form if validation
    succeeds
67 if ( $myForm->validate() )
68 {
69     $myForm->removeElement('submit
    Button');
70     $myForm->freeze();
71 }
72 // Display form
73 $myForm->display();
74 ?>

```

legibilidad. El paquete *QuickForm* también facilita los grupos, donde los elementos individuales se sitúan juntos. En el caso del formulario de la tarjeta de crédito (véase la Figura 3), tiene más sentido organizar la validez de los datos (que es la combinación de los meses y de los años que forman la fecha) de esta manera.

Para hacer esto, las líneas 44 y 45 añaden dos menús desplegables a la matriz `$validTo[]`. Estos elementos individuales no contienen etiquetas y esto es porque el tercer parámetro en el método estático `createElement()` es `NULL`. Se especifica una etiqueta para todo el grupo en la línea 48 usando `addGroup()`; la matriz será el primer parámetro que se le pasará. Las restantes instrucciones están en el Listado 1.

## Ejemplos Sencillos

Como no es necesario escribir el software cliente y existe un navegador Web para casi cada sistema operativo, más y

más programadores están adoptando la metodología de aplicaciones basadas en servidor Web para desarrollar aplicaciones. También se pueden estimar los costes operativos, pues los costes de mantenimiento del software y similares son más bajos que para las soluciones tradicionales cliente/servidor, debido al hecho de que el software solamente necesita ser actualizado del lado del servidor. Ambos ejemplos de PHP mostrados aquí son muy simples, pero demuestran que ambas aplicaciones Web se desarrollan fácilmente. Y al mismo tiempo es fácil acostumbrarse a evitar los errores de entrada. Los sistemas de gestión de flujo de trabajo son un uso típico que implica un montón de formularios y son un campo perfecto para el uso de *QuickForm*. La biblioteca presentada en este artículo proporciona una amplia gama de las funciones adicionales satisfaciendo varios campos de aplicación, tales como procesado de carga de ficheros, mejoras personalizadas, funciones especializadas de

## Peras y Adobos Gratis

Con una gama tan grande de extensiones disponibles, hay que poner un cierto esfuerzo para promover una librería. Es por esto que el promotor de Pear, Stig Saether Bakken, distribuye peras y adobos a los asistentes a sus presentaciones y talleres de Pear. A pesar de que la virtud de este dúo culinario es discutible, atrae la atención de la audiencia sobre los nombres Pear y PECL (se dice "pickle", "adobo" en inglés). PECL es el acrónimo de PHP Extension Code Library, que abarca un subconjunto de Pear y contiene las extensiones PHP programadas en C. Si se busca extensiones más exóticas, que faciliten Mono, el clon del marco de trabajo de Microsoft .NET, deberemos examinar PECL. La instalación es manejada por el instalador de paquetes, como con cualquier otro paquete Pear. En el caso de los paquetes PECL, nos aseguraremos de que cualquier dependencia del sistema que surja se resuelva. Si se está instalando el paquete Mono, por supuesto que se necesitará el marco de trabajo Mono.

## No es necesario el escepticismo

Los desarrolladores tienden a ser meticulosos y desconfiados con el código de los demás. Esta es la razón por la cual muchos desarrolladores ven Pear con un cierto grado de escepticismo. Pero esta clase de resistencia a la librería de las clases de PHP está injustificada. Si se confía en PHP, también debemos estar preparados para confiar en la calidad de Pear, al menos en lo que se refiere a los paquetes estables de la Pear Foundation Classes (PFC).

## INFO

- [1] Listados de este artículo: <http://www.linuxmagazine.com/es/descargas>
- [2] Sitio oficial de PHP: <http://www.php.net>
- [3] Sitio oficial de Pear: <http://pear.php.net>
- [4] Navegador de paquetes Pear: <http://pear.php.net/packages.php>
- [5] Motor de plantillas Smarty: <http://smarty.php.net/>
- [6] Protocolo de la reunión Pear del 2003: <http://pear.php.net/news/meeting-2003-summary.php>

"callback" o filtros inteligentes para entrada de campos. Pero *QuickForm* no es solamente la librería facilitada por Pear para simplificar las tareas de los programadores. La transferencia de datos de una tarjeta de crédito, la cuál podría considerarse como una extensión al segundo ejemplo, se maneja normalmente a través de HTTP o Soap. De nuevo los paquetes de Pear reducen este proceso a apenas algunas líneas de código y ahorran a los desarrolladores el esfuerzo de implementar protocolos estándar. Gracias a Pear, no es necesario reinventar la rueda para cada campo de aplicación en el que se aventure. ■

## Calidad con C mayúscula

Estos paquetes de alta calidad están sujetos a medidas rigurosas de control de calidad. Cuando se agregan nuevas características, o cambian las necesidades de una implementación, esos cambios son discutidos por un equipo de experimentados desarrolladores usando una lista de correo. Los paquetes de Pear cumplen con una codificación estándar que define con precisión como deben programar los desarrolladores, que incluye como hay que sangrar el código, como asignar nombres de variables. Esta metodología permite que los desarrolladores de Pear y recién llegados al lenguaje se hagan rápidamente con el código fuente. Además, es un requisito facilitar una buena documentación para el código. Gracias a este estándar, la documentación del API para el paquete puede crearse automáticamente. Y es por esta razón que la página Web de Pear en [3] contiene abundantes descripciones de este tipo. Muchos de los autores de paquetes Pear son miembros respetados y bien conocidos de la comunidad de desarrolladores que también han contribuido en otras partes de PHP. Para descubrir cuánta sangre, sudor y lágrimas se ha derramado para producir código de calidad, solo se debe dedicar unos minutos a leer el resumen de la última reunión de Pear [6].

## EL AUTOR

*Richard Samar, <http://richard-samar.de>, es un consultor independiente de TI que vive cerca de Frankfurt, Alemania. Es un entusiasta de PHP desde hace muchos años y es bien conocido en la comunidad alemana de PHP.*

