Creación de un módulo de PrestaShop

Tabla de contenidos

Creación de un módulo de PrestaShop

¿Qué es un módulo de PrestaShop?

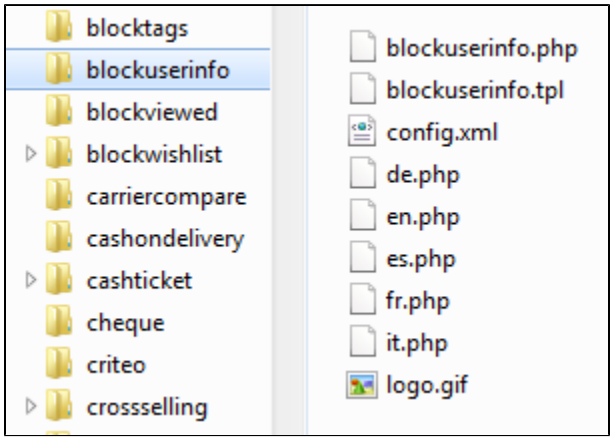
La extensibilidad de PrestaShop gira en torno a los módulos, los cuales son pequeños programas que hacen uso de las funciones de PrestaShop y los cambia o agrega para que el uso de PrestaShop sea más sencillo y personalizado.

Principios técnicos en que se basa un módulo

Un módulo de PrestaShop se compone de:

- Una carpeta raíz, llamada como el módulo, que mantendrá la totalidad de los archivos del módulo y se encontrará en la carpeta `/modules` de PrestaShop.
- Un archivo principal de PHP, llamado como el módulo, localizado en la carpeta raíz. Este archivo PHP debe tener el mismo nombre que su carpeta raíz.
- Un archivo de icono, llamado `logo.gif`, representando este módulo.
- Opcional: algunos archivos `.tpl`, que contienen el tema del módulo.
- Opcional: archivos de idioma, si el módulo o su tema cuentan con texto para mostrar (por lo tanto, este debe ser traducible).
- Opcional: en la carpeta `/themes/modules` , una carpeta con el mismo nombre que el módulo que contenga `.tpl` y los archivos de idioma si es necesario. Esta última carpeta es esencial durante las modificaciones de módulo existente, de modo que pueda adaptarse sin tener que manipular sus archivos originales. Particularmente, esta carpeta le permite manejar la visualización del módulo de distintas maneras, de acuerdo con el tema actual.

Veamos un ejemplo con el módulo **blockuserinfo** de PrestaShop:



Cualquier módulo de PrestaShop, una vez instalado en una tienda virtual, puede interactuar con uno o varios "hooks". Los Hooks le permiten "enganchar" su código a la vista actual al momento del análisis de código (es decir, al mostrar el carrito o la ficha de producto o al mostrar el stock actual...). En concreto, un hook es un acceso directo a los diferentes métodos disponibles del objeto del Módulo, según lo asignado a ese hook.

Principios de funcionamiento de los módulos

Los módulos son la manera ideal para dejar que su talento e imaginación como desarrollador se expresen por sí mismos, ya que las posibilidades creativas son muchas.

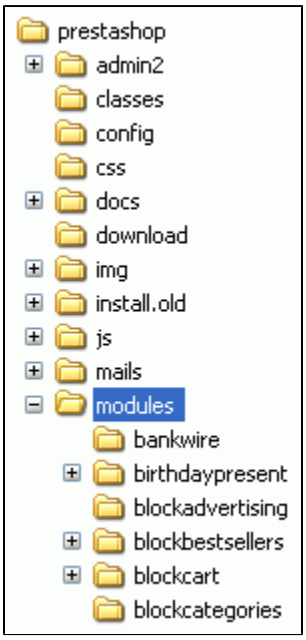
Estos pueden mostrar una gran variedad de contenido (bloques, texto, etc.), realizar muchas tareas (actualización por lote, importación, exportación, etc.) y funcionar en conjunto con otras herramientas...

Los módulos pueden ser tan configurables como sea necesario, mientras más configurable sea, más sencillo será de utilizar y por lo tanto será capaz de responder a la necesidad de una amplia variedad de usuarios.

Quizás el interés principal de un módulo es agregar funciones a PrestaShop sin tener que editar sus archivos principales, haciéndolo más fácil para realizar una actualización sin necesidad de transponer todos los cambios fundamentales.

Esa es la forma en que debe tratar de mantenerse alejado de los archivos centrales cuando construya un módulo, aunque esto puede resultar difícil de realizar en algunas situaciones...

Árbol de archivos del módulo



Todos los Módulos están localizados en la carpeta `/modules`, la que se encuentra en la raíz de la carpeta principal de PrestaShop. Esto es cierto tanto para los módulos por defecto (incluidos con PrestaShop) y los módulos de terceros que se instalan posteriormente.

Cada módulo tiene su propia sub-carpeta dentro de la carpeta `/modules`: `/bankwire`, `/birthdaypresent`, etc.

Estructura básica de un módulo

Todos los módulos utilizan la misma estructura básica, lo que hace que sea más fácil el aprendizaje mediante la observación de códigos de fuente de módulos existentes.

Vamos a crear un primer módulo sencillo, lo que nos permitirá describir mejor su estructura. Lo llamaremos "My module".

Primero vamos a crear la carpeta del módulo. Debe tener el mismo nombre que el módulo, sin espacio, sólo caracteres alfanuméricos, el guion y el guion bajo, todo en minúsculas: `/mymodule`.

Esta carpeta debe contener un archivo PHP del mismo nombre, que se encargará de la mayor parte del procesamiento: `mymodule.php`.

Eso es suficiente para un módulo muy básico, pero evidentemente, más archivos y carpetas pueden complementarlo.

La parte de front-office del módulo es definida en un archivo `.tpl` colocado en la raíz de la carpeta del módulo. Los archivos TPL pueden utilizar casi cualquier nombre. Si sólo existe un archivo como este, sería bueno ofrecerle el mismo nombre que la carpeta y el archivo principal: `mymodule.tpl`.

El archivo `mymodule.php` debe comenzar con la siguiente prueba:

```
if ( !defined( '_PS_VERSION_' ) )
  exit;
```

Esto comprueba la existencia de una constante de PHP, y si no existe, abandona. El único propósito de esta prueba es evitar que los visitantes carguen este archivo directamente.

El archivo también debe contener la clase del módulo. PrestaShop utiliza la programación Orientada a Objetos y también la utilizan sus módulos.

Esa clase debe tener el mismo nombre que el módulo y su carpeta, en CamelCase: `MyModule`.
Por otra parte, la clase debe extender la clase `Module` y por lo tanto hereda todos los métodos y atributos. Esta puede igualmente extender cualquier clase derivada del `Module`: `PaymentModule`, `ModuleGridEngine`, `ModuleGraph`...

mymodule.php

```
<?php
if ( !defined( '_PS_VERSION_' ) )
  exit;

class MyModule extends Module
  {
  public function __construct()
    {
    $this->name = 'mymodule';
    $this->tab = 'Test';
    $this->version = 1.0;
    $this->author = 'Firstname Lastname';
    $this->need_instance = 0;

    parent::__construct();

    $this->displayName = $this->l( 'My module' );
    $this->description = $this->l( 'Description of my module.' );
    }

  public function install()
    {
    if ( parent::install() == false )
      return false;
    return true;
    }
  }
?>
```

Vamos a examinar cada línea de nuestro objeto `MyModule`...

```
public function __construct()
```

Define el constructor de clase.

```
$this->name = 'mymodule';
$this->tab = 'Test';
$this->version = 1.0;
$this->author = 'PrestaShop';
```

Esta sección asigna unos cuantos atributos de la instancia de clase (`this`):

- Un atributo 'name'. Este es un identificador interno, así que hágalo único, sin caracteres especiales o espacios, y en minúsculas.
- Un atributo 'tab'. Este es el título del cuadro que deberá contener este módulo en la lista de módulos del back office de PrestaShop. Usted puede utilizar un nombre ya existente, como `Products`, `Blocks` or `Stats`, o uno personalizado, como lo hicimos aquí. En este último caso, un nuevo cuadro se ha creado con su título.
- Un número de versión para el módulo, aparece en la lista de módulos.
- Un atributo 'author'. Este se muestra en la lista de módulos de PrestaShop.

```
$this->need_instance = 0;
```

La bandera `need_instance` indica si se debe cargar la clase del módulo cuando se muestran los "Módulos", en el back-office. Si se establece en 0, el módulo no cargará, por lo tanto gastará menos recursos para generar el módulo de página. Si sus módulos necesitan mostrar un mensaje de advertencia en la página "Módulos", entonces debe establecer este atributo en 1.

```
parent::__construct();
```

Llamar al constructor padre. Esto se debe realizar antes de cualquier uso del método `$this->l()` y después de la creación de `$this->name`.

```
$this->displayName = $this->l( 'My module' );
```

Asignar un nombre público para el módulo, el cual será mostrado en la lista de módulos de back-office. El método `l()` es parte de las herramientas de traducción de PrestaShop, y se explica más adelante.

```
$this->description = $this->l( 'Description of my module.' );
```

Asignación de una descripción pública para el módulo, la cual se mostrará en la lista de módulos del back-office.

```
public function install()
  {
  return ( parent::install() );
  }
```

En esta primera encarnación extremadamente simple, este método no es útil, ya que lo único que realiza es comprobar el valor devuelto por el método install() class de Módulo. Además, si no hubiéramos creado este método, el método superclase hubiera sido llamado en su lugar de todas formas, logrando que el resultado final sea idéntico.
Sin embargo, debemos mencionar que este método será muy útil una vez que tengamos que realizar las pruebas y acciones durante el proceso de instalación del módulo: crear tablas de SQL, copiar archivos, crear variables de configuración, etc.

Asimismo, el módulo debe contener un método `uninstall()` para contar con un proceso de desinstalación personalizado. Este método podría realizarse algo así como:

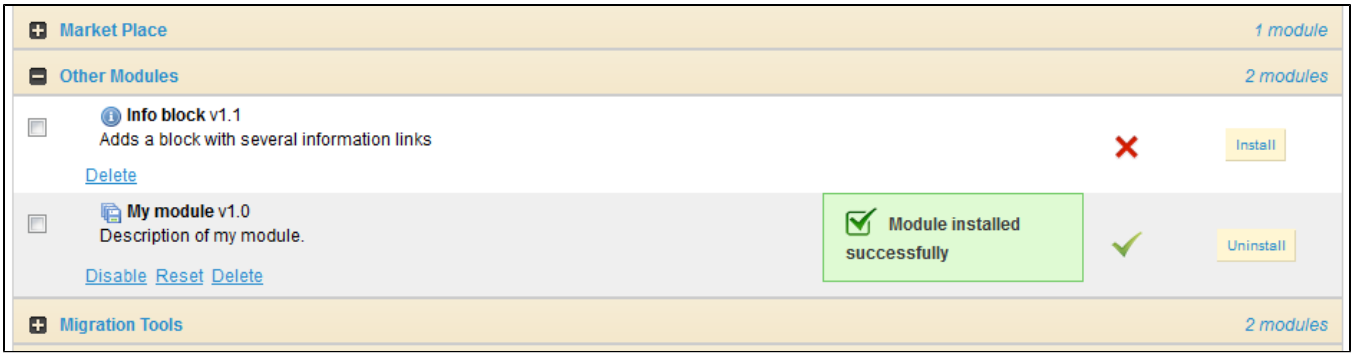
```
public function uninstall()
  {
  if ( !parent::uninstall() )
    Db::getInstance()->Execute( 'DELETE FROM `' . _DB_PREFIX_ . 'mymodule`' );
  parent::uninstall();
  }
```

Para poner "el toque final" a este módulo básico, podemos agregar un icono, que se mostrará junto al nombre del módulo en la lista de módulos de back-office.
El archivo de icono debe respetar los siguientes requisitos:

- imágenes 16*16.
- nombrado `logo.gif`.
- colocado en la carpeta principal del módulo.

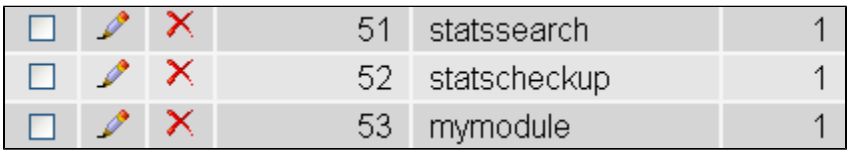
Puede encontrar un excelente conjunto de iconos gratuitos para elegir en el sitio web de FamFamFam.

Ahora que todos los fundamentos están en su lugar, coloque la carpeta del módulo en la carpeta `/modules` de su instalación de prueba de PrestaShop, abra PrestaShop, y en la pestaña "Módulos", bajo "Otros módulos", debería encontrar su módulo. Instálelo para ser capaz de manejarlo por el resto de esta guía.



PrestaShop crea automáticamente un pequeño archivo `config.xml` en la carpeta del módulo, el cual almacena un poco de información de la configuración. **NUNCA** la debe editar a manualmente.

Durante la instalación, PrestaShop también agrega una línea al cuadro de SQL `ps_module`.



Conexión de un módulo a través de Hooks

Para que un módulo sea "conectado" o "enganchado" a una ubicación en el front y back office, necesita ofrecerle acceso a uno de los hooks de PrestaShop, descritos anteriormente en esta guía.

En ese caso, cambiaremos el código del módulo, y agregaremos estas líneas:

mymodule.php (partial)

```
public function install()
  {
  if ( parent::install() == false OR !$this->registerHook( 'leftColumn' ) )
    return false;
  return true;
  }

...

public function hookLeftColumn( $params )
  {
  global $smarty;
  return $this->display( __FILE__, 'mymodule.tpl' );
  }

public function hookRightColumn( $params )
  {
  return $this->hookLeftColumn( $params );
  }
```

Vamos a explorar estas nuevas líneas.

```
if ( parent::install() == false OR !$this->registerHook( 'leftColumn' ) )
  return false;
return true;
```

Hemos cambiado la línea original para agregar una segunda prueba.

Este código comprueba:

- el valor booleano devuelto por el método `install()` de la clase `Module`: si `true`, significa que el módulo está instalado y puede ser utilizado.
- el valor booleano devuelto por `registerHook()` para el hook `leftColumn`: si `true`, significa que el módulo se encuentra registrado en el hook que necesita y puede ser utilizado.

Si cualquiera de estos dos valores booleanos es `false`, `install()` devolverá también `false` , y el módulo no podrá ser instalado. Ambos valores tienen que ser `true` para que el módulo sea considerado como instalado.

Por lo tanto, esta línea ahora se lee de esta manera: si la instalación o el "enganche" fracasa, nosotros informaremos a PrestaShop.

```
public function hookLeftColumn( $params )
  {
  global $smarty;
  return $this->display(__FILE__, 'mymodule.tpl');
  }
```

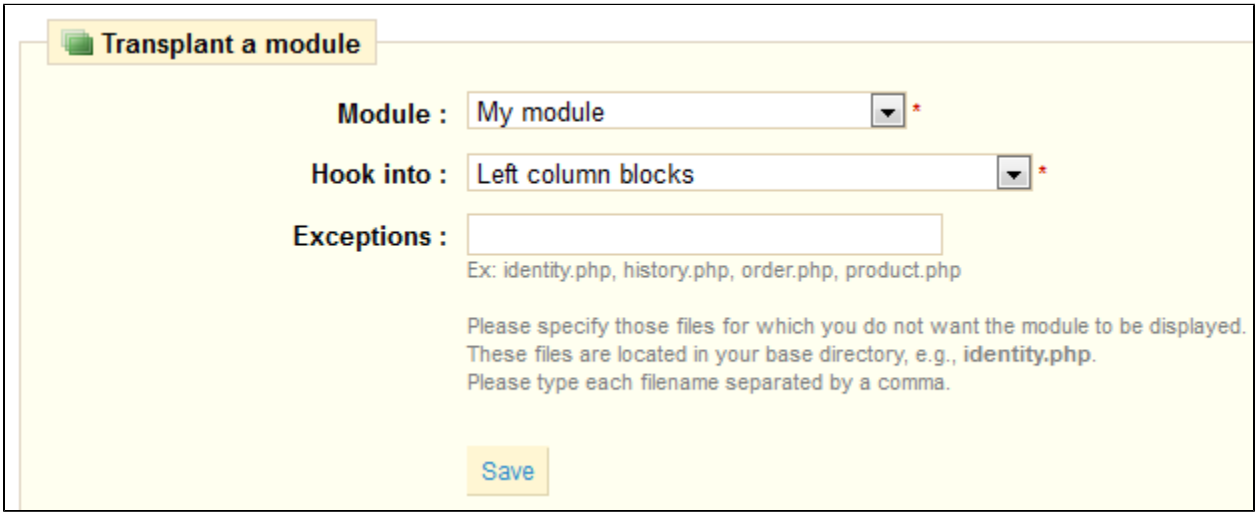
El método `hookLeftColumn()` hace posible que el módulo se "enganche" en la columna izquierda del tema.
`$smarty` es la variable global para el sistema de plantillas Smarty, el cual utiliza PrestaShop y que tenemos que acceder.
El método `display()` devuelve el contenido del archivo de plantilla `mymodule.tpl`, si es que existe.

```
public function hookRightColumn( $params )
  {
  return $this->hookLeftColumn( $params );
  }
```

Del mismo modo, `hookRightColumn()` ofrece acceso a la columna derecha del tema. En este ejemplo, llamamos al método `hookLeftColumn()` para obtener que se muestre la misma sin importar la columna.

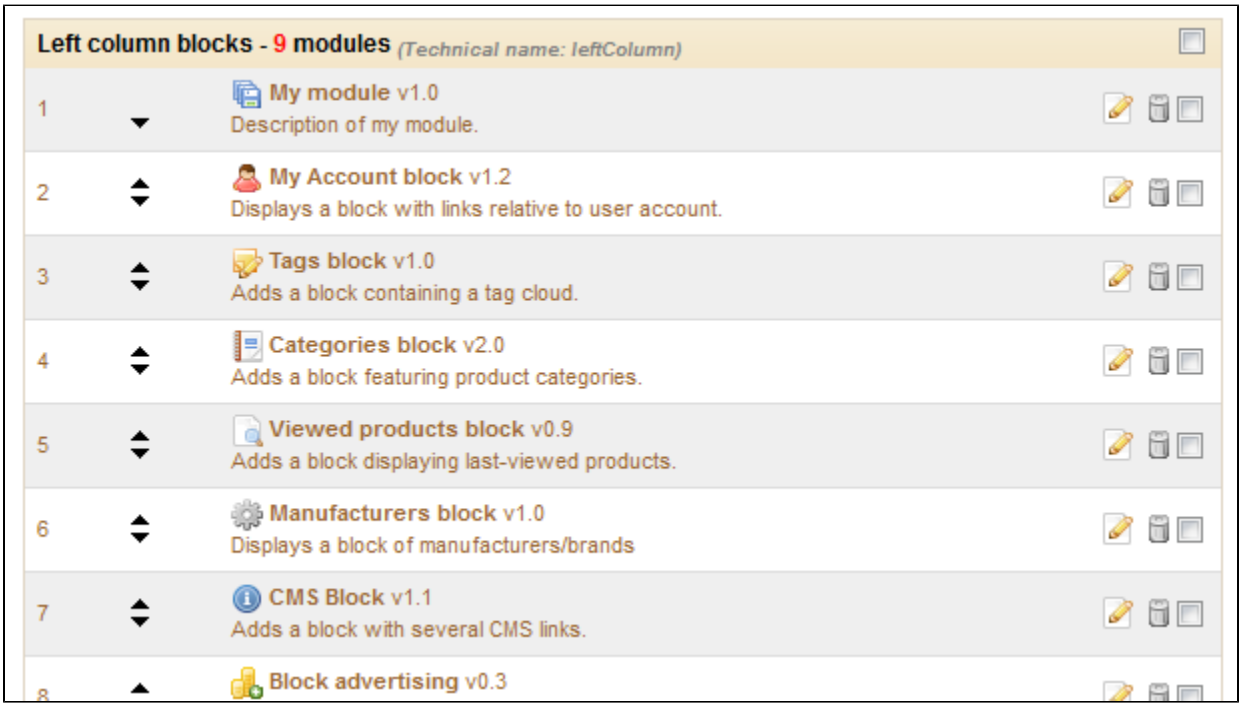
Guarde su archivo, ahora ya puede “engancharlo” con el tema, desplácelo y trasládelo: vaya a la sub-pestaña "Posiciones" en la "Módulos" del back-office, luego haga clic en el enlace “Trasladar un módulo".

En el formulario, encuentre "My module" en el menú desplegable de módulos, luego seleccione "bloques de columna izquierda" en el menú desplegable de "Hook en".



Es inútil tratar de conectar un módulo a un hook para el cual no tiene ningún método implementado.

Guarde. La página "Posiciones" debe cargar, con el siguiente mensaje: "Módulo trasladado con éxito a un hook". ¡Felicitaciones! Desplácese hacia abajo y podrá observar su módulo junto con los otros módulos de la lista "bloques de columna izquierda". Muevalo a la parte superior de la lista.



Visualización del Contenido

Ahora que tenemos acceso a la columna de la izquierda, debemos mostrar algo en la misma.

Como se dijo anteriormente, el contenido a mostrar en el tema deben ser almacenados en archivos `.tpl`. Vamos a crear el archivo `mymodule.tpl`, que fue aprobado como un parámetro del método `display()` en el código de nuestro módulo.

Por consiguiente, crearemos el archivo `mymodule.tpl` y le agregaremos algunas líneas de código.

mymodule.tpl

```
<!-- Block mymodule -->
<div id="mymodule_block_left" class="block">
  <h4>Welcome!</h4>
  <div class="block_content">
    <ul>
      <li><a href="{$base_dir}modules/mymodule/mymodule_page.php" title="Click this link">Click me!</a></li>
    </ul>
  </div>
</div>
<!-- /Block mymodule -->
```

Guarde el archivo en la carpeta raíz del módulo, recargue la página principal de su tienda: debe aparecer en la parte superior de la columna de la izquierda, justo debajo del logotipo de la tienda.

El enlace que aparece no conduce a nada por ahora. Si usted necesita probarlo, agregue el archivo necesario `mymodule_page.php` en la carpeta del módulo, con un contenido mínimo, como "Welcome to my shop!" La página resultante será muy simple, a ver si intentamos utilizar el estilo del tema en su lugar.

Como era de esperar, tenemos que crear un archivo TPL para utilizar el estilo del tema. Vamos a crear el archivo `mymodule_page.tpl` que contendrá la línea básica y llamaremos a ese archivo `mymodule_page.php`, el cual agregará el encabezado del tema, pie de página, etc.

> Debe esforzarse por utilizar nombres explícitos y reconocibles para sus archivos TPL, para que pueda encontrarlos rápidamente en el back-office – los que son una necesidad cuando se utiliza la herramienta de traducción.

mymodule_page.tpl

```
Welcome to my shop!
```

mymodule_page.php

```
<?php
global $smarty;
include( '../../config/config.inc.php' );
include( '../../header.php' );

$smarty->display( dirname(__FILE__) . '/mymodule_page.tpl' );

include( '../../footer.php' );
?>
```

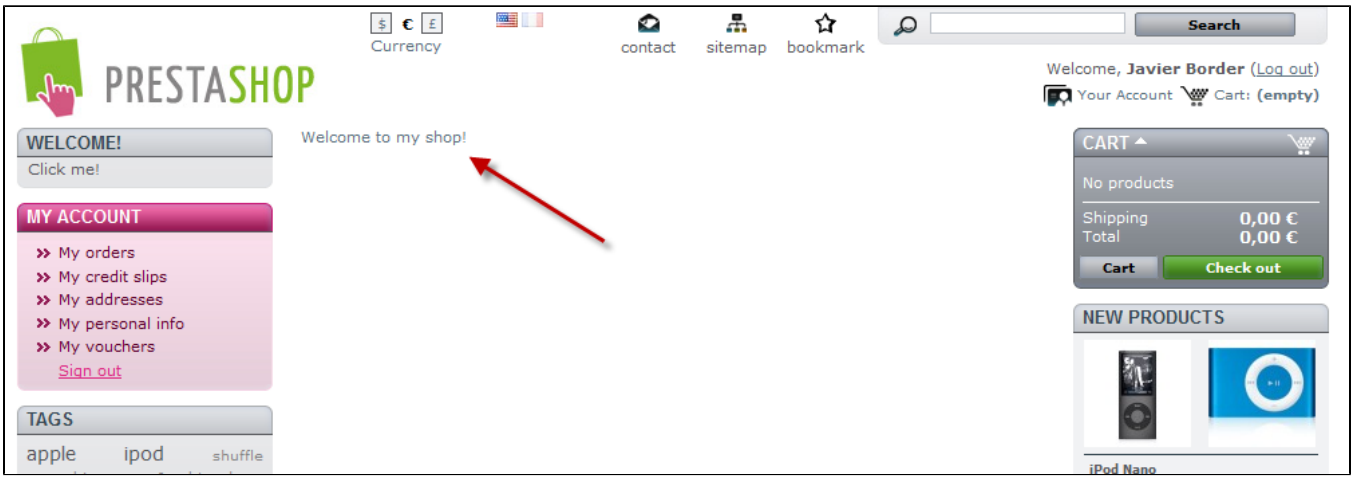
Primero, cargaremos la instancia Smarty actual. Esto debe realizarse antes de mencionar al método d`isplay()`.

Los diversas menciones `include()` en el archivo nos permite cargar:

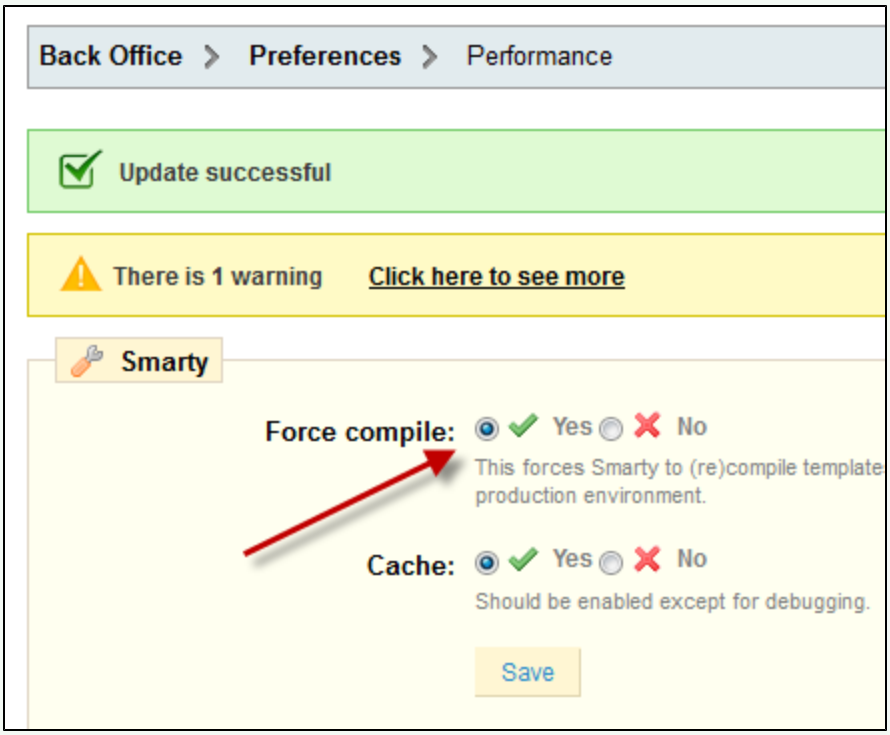
- la configuración actual de PrestaShop.
- el archivo de encabezado del tema (mediante `header.php`, que actúa como un archivo de carga).
- el archivo de pie de página del tema (mediante `footer.php`, que actúa como un archivo de carga).

Colocaremos su archivo de plantilla en medio de estos, cuya única acción será la de mostrar la línea "Welcome to my shop!".

Guarde todos los archivos y recargue la página de inicio de su tienda: con sólo unas pocas líneas, el resultado final es mucho mejor, ¡Con nuestra línea "Welcome" perfectamente colocada entre el encabezado, el pie de página y las columnas!



Si realiza múltiples cambios y recargas a su página de inicio, puede parecer que dichos cambios no son aplicados. Esto se debe a que Smarty almacena una versión compilada de la página de inicio. Para forzar Smarty a recompilar plantillas en cada invocación, diríjase a la sub-pestaña de “Rendimiento” en la pestaña de "Preferencias" y elija "Sí" en la opción "Forzar la compilación".



¡**No fuerce la compilación en sitios de producción**, ya que retrasa todo gravemente!

Utilización de Smarty

Smarty es un motor de plantillas PHP y es utilizado por el sistema de creación de temas de PrestaShop.

Este analiza archivos TPL, buscando elementos dinámicos para sustituir por sus datos equivalentes, luego muestra el resultado generado. Estos elementos dinámicos se indican entre llaves: { ... }. El programador puede crear nuevas variables y utilizarlas en los archivos TPL.

Por ejemplo, en nuestro `mymodule_page.php`, podemos crear una variable:

mymodule_page.php

```
<?php
global $smarty;

include( '../../config/config.inc.php' );
include( '../../header.php' );

$mymodule = new MyModule();
$message = $mymodule->l( 'Welcome to my shop!' );
$smarty->assign( 'messageSmarty', $message ); // creation of our variable
$smarty->display( dirname(__FILE__) . '/mymodule_page.tpl' );

include( '../../footer.php' );
?>
```

A partir de ahí, podemos pedir a Smarty que muestre el contenido de esta variable en nuestro archivo TPL.

mymodule_page.tpl

```
{$messageSmarty}
```

PrestaShop incluye una serie de variables. Por ejemplo, {$HOOK_LEFT_COLUMN} será remplazado por el contenido de la columna izquierda, es decir, el contenido de todos los módulos que han sido unidos al hook de la columna izquierda.

Todas las variables de Smarty son globales. Por lo tanto, debe prestar atención a no nombrar su propia variable con el nombre de una variable de Smarty existente, a fin de evitar sobrescribirla. Le aconsejamos evitar los nombres demasiado simples, tales como `products`, y anteponerlo con el nombre de su módulo, o incluso su propio nombre, como por ejemplo: {$mark_mymodule_product}.

Aquí le ofrecemos una lista de variables de Smarty que son comunes a todas las páginas:

Archivo / carpeta	Descripción
img_ps_dir	URL de la carpeta de imágenes de PrestaShop.
img_cat_dir	URL de la carpeta de imágenes de categorías.
img_lang_dir	URL de la carpeta de imágenes de idiomas.
img_prod_dir	URL de la carpeta de imágenes de productos.
img_manu_dir	URL de la carpeta de imágenes de los fabricantes.

img_sup_dir	URL de la carpeta de imágenes de los proveedores.
img_ship_dir	URL de la capeta de las imágenes de los transportistas.
img_dir	URL de la carpeta de las imágenes del tema.
css_dir	URL de la carpeta del tema CSS.
js_dir	URL de la carpeta del tema JavaScript.
tpl_dir	URL de la carpeta del tema actual.
modules_dir	URL de la carpeta de módulos.
mail_dir	URL de la carpeta de plantillas de correo.
pic_dir	URL de la carpeta de las fotos subidas.
lang_iso	Código ISO del idioma actual.
come_from	URL de origen del visitante.
shop_name	Nombre de la tienda.
cart_qties	Número de productos en el carrito.
cart	El carrito de compras.
currencies	Las monedas disponibles.
id_currency_cookie	ID de la moneda actual.
currency	Objeto de moneda (moneda utilizada actualmente).
cookie	Cookies de usuario.
languages	Los diferentes idiomas disponibles.
logged	Indica si el visitante se ha conectado a una cuenta de cliente.
page_name	Nombre de la página.
customerName	Nombre del cliente (si está conectado).
priceDisplay	Método de visualización del precio(con o sin impuestos...).
roundMode	Método de redondeo en uso.
use_taxes	Indica si los impuestos se encuentran habilitados.

Si necesita que se muestren todas las páginas actuales de variables de Smarty, agregue la siguiente función:

```
{debug}
```

Los comentarios se basan en asterisco:

```
{* This string is commented out *}

{*
This string is too!
*}
```

A diferencia de los comentarios HTML, el código “commented out” de Smarty no se encuentra presente en el archivo final de salida.

Traducción de un Módulo

Las cadenas de texto de nuestro módulo están escritos en Inglés, pero también pudiéramos permitir a propietarios de tienda españoles a utilizar nuestro módulo. Por lo tanto, tenemos que traducir las cadenas al español, tanto en el front como en el back-office. Esto podría ser una tarea tediosa, pero la traducción propia de Smarty y PrestaShop hacen que sean mucho más fáciles.

Las cadenas en los archivos PHP necesitarán ser mostradas a través del método `l()`, de la clase abstracta `Module.php`.

mymodule.php (partial)

```
...
$this->displayName = $this->l( 'My module' );
$this->description = $this->l( 'Description of my module.' );
...
```

Las cadenas en los archivos TPL tendrán que ser convertidas en contenido dinámico, que Smarty remplazará por la traducción del idioma elegido. En nuestro módulo de muestra, este archivo:

mymodule.tpl (partial)

```
<li>
  <a href="{$base_dir}modules/mymodule/mymodule_page.php" title="Click this link">Click me!</a>
</li>
```

...se convierte en:

mymodule.tpl (partial)

```
<li>
  <a href="{$base_dir}modules/mymodule/mymodule_page.php" title="{l s='Click this link' mod='mymodule'}">{l
s='Click me!' mod='mymodule'}</a>
</li>
```

...y este otro:

mymodule_page.tpl

```
<h4>Welcome!</h4>
...
Click me!
```

...se convierte en:

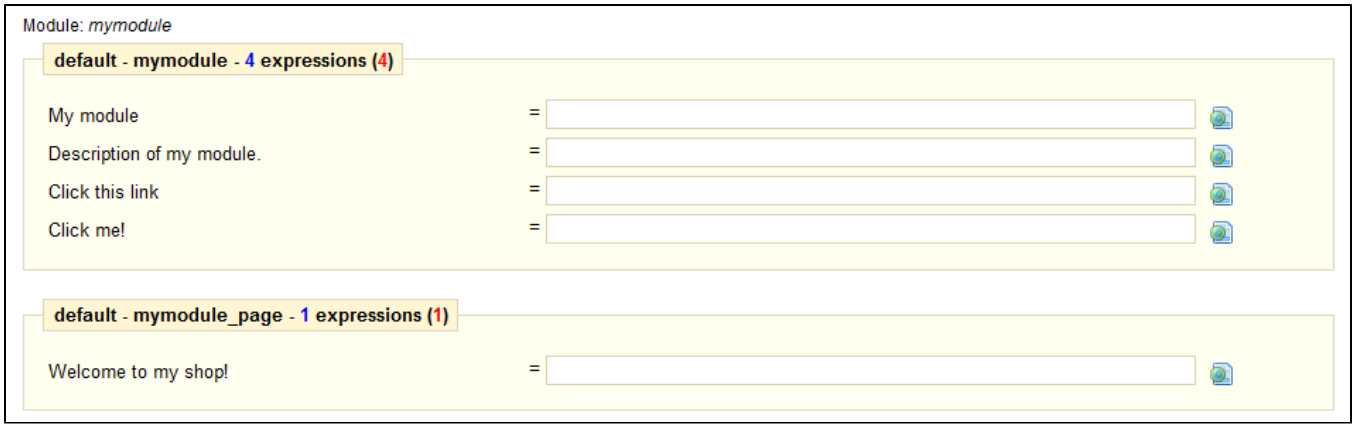
mymodule.tpl

```
<h4>{l s='Welcome!' mod='mymodule'}</h4>
...
{l s='Click me!' mod='mymodule'}
```

La herramienta de traducción necesita del parámetro `mod` para coincidir la cadena a traducir con su traducción.
Las cadenas se delimitan con comillas simples. Si una cadena contiene comillas simples, deben ser escapados utilizando una barra inversa ().

De esta manera, las cadenas pueden ser traducidas directamente dentro de PrestaShop: trasládese a la sub-pestaña "Traducciones" de la pestaña "Herramientas" y en el menú desplegable de la sección "Modificar traducciones", elija la opción "Traducciones de los módulos", a continuación, haga clic en la bandera española para traducir los módulos al español.

La siguiente página muestra todas las cadenas de todos los módulos instalados actualmente. Los módulos que tienen todas sus cadenas ya traducidas contienen sus elementos cerrados, mientras que si por lo menos una cadena no se encuentra en la traducción de un módulo, sus elementos son expandidos.
Para traducir las cadenas de su módulo (las que fueron "marcadas" utilizando el método `l()`), sólo tiene que encontrar su módulo en la lista (use el buscador en la página de su navegador) y rellene los campos vacíos.



Una vez que todas las cadenas para el módulo se encuentran correctamente traducidas, haga clic en el botón "Actualizar traducción", ya sea en la parte superior o inferior de la página.

Cada campo tiene un icono a la derecha. Esto le permite obtener una sugerencia del Traductor de Google. Puede desplazar el ratón sobre él para ver la traducción y haga clic en él para rellenar el campo con la traducción.

La traducciones automáticas no siempre son exactas, use con precaución.

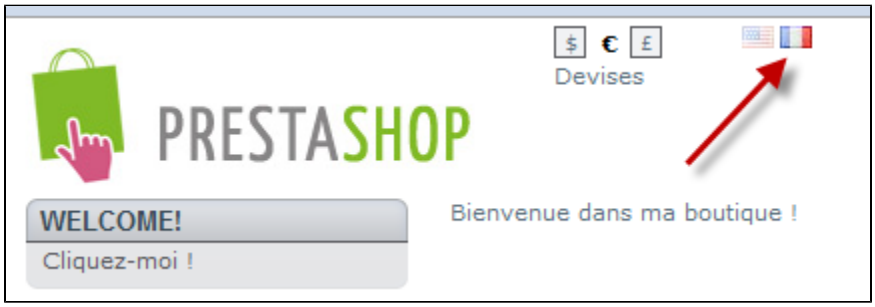
Las traducciones se guardan en un archivo nuevo, `es.php` (o `languageCode.php`, el cual es generado por PrestaShop y luce así:

mymodule.tpl

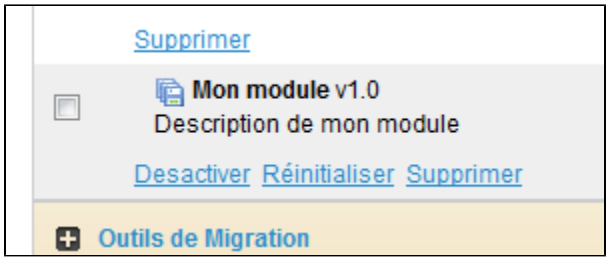
```
<?php

global $_MODULE;
$_MODULE = array();
$_MODULE['<{mymodule}prestashop>mymodule_2ddddc2a736e4128ce1cdfd22b041e7f'] = 'Mon module';
$_MODULE['<{mymodule}prestashop>mymodule_d6968577f69f08c93c209bd8b6b3d4d5'] = 'Description de mon module';
$_MODULE['<{mymodule}prestashop>mymodule_c66b10fbf9cb6526d0f7d7a602a09b75'] = 'Cliquez sur ce lien';
$_MODULE['<{mymodule}prestashop>mymodule_f42c5e677c97b2167e7e6b1e0028ec6d'] = 'Cliquez-moi \!';
$_MODULE['<{mymodule}prestashop>mymodule_page_c0d7cffa0105851272f83d5c1fe63a1c'] = 'Bienvenue dans ma boutique
\!';
```

¡Este archivo no debe ser editado manualmente! Sólo puede ser editado a través de la herramienta de traducción de PrestaShop.

Ahora que tenemos una traducción, podemos hacer clic en la bandera española en el front-office (siempre que el idioma ha sido instalado) y obtener el resultado esperado: las cadenas del módulo se encuentran ahora en español.



Y podrán ser traducidas al español cuando el Back Office se encuentre en español.



Sólo las cadenas traducidas son tomadas en cuenta por la herramienta de PrestaShop, los archivos de PHP y TPL deben ser ubicados en la raíz de la carpeta del módulo.

Creación de la pestaña de back office del módulo y su clase

En esta sección usted aprenderá cómo ofrecer a su módulo, su propia pestaña o sub-pestaña en cuestión de minutos.

Siga estos pasos:

1. Agregar un nuevo cuadro a su base de datos de PrestaShop, llamado `ps_test`. Ofrézcale dos campos:
    - `id_test` (INT 11)
    - `test` (VARCHAR 32)

2. Crear un archivo en blanco llamado `Test.php` en la carpeta `/classes` de PrestaShop.
3. Agregar las siguientes líneas a este archivo:

Test.php

```
<?php
class Test extends ObjectModel
  {
  /** @var string Name */
  public $test;

  protected $fieldsRequired = array( 'test' );
  protected $fieldsSize = array( 'test' => 64 );
  protected $fieldsValidate = array( 'test' => 'isGenericName' );
  protected $table = 'test';
  protected $identifier = 'id_test';

  public function getFields()
    {
    parent::validateFields();
    $fields[ 'test' ] = pSQL( $this->test );
    return $fields;
    }
  }
?>
```

1. Crear un archivo en blanco llamado `AdminTest.php` en `/admin/tabs` de PrestaShop.
2. Agregar las siguientes líneas a este archivo:

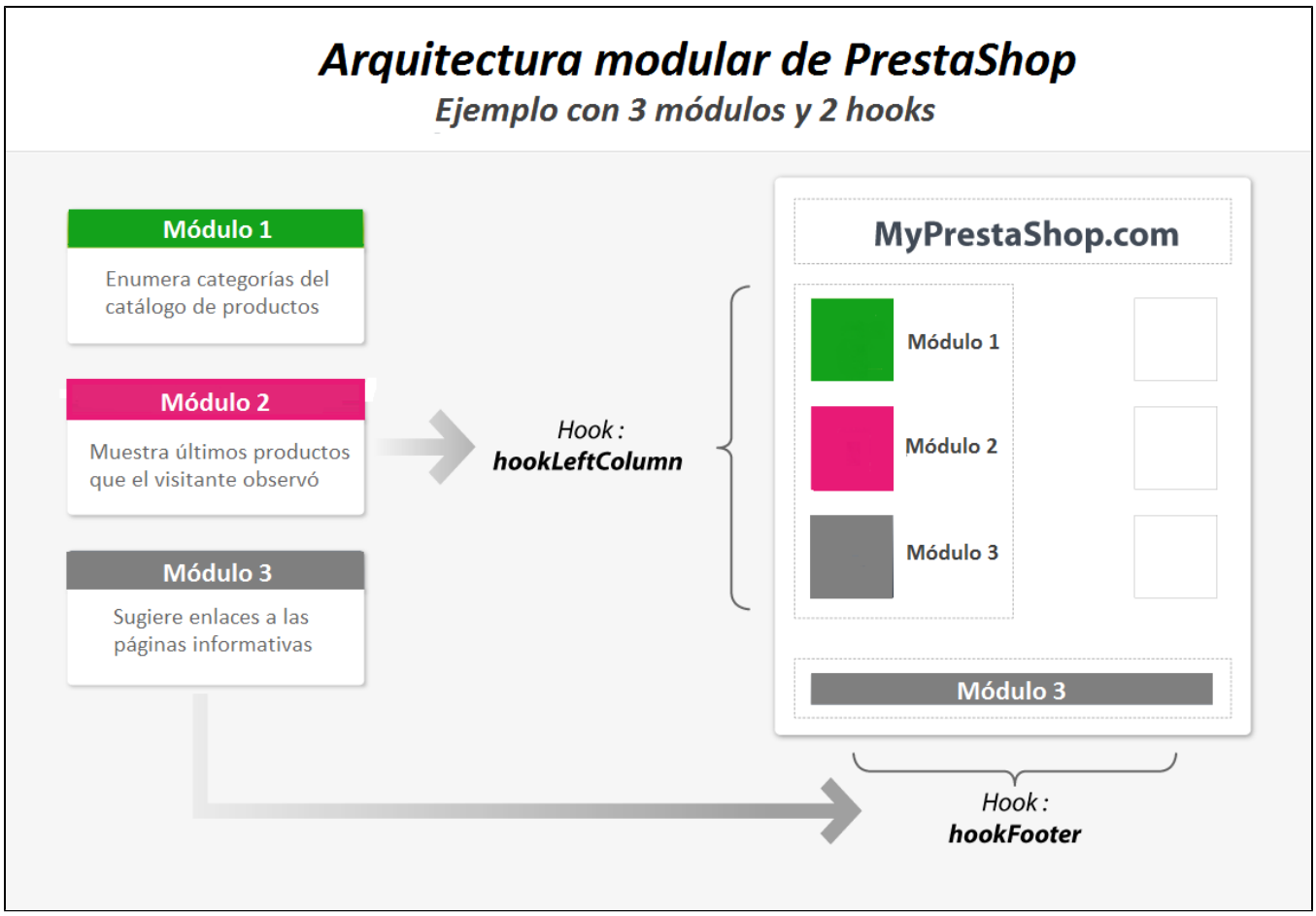
AdminTest.php

```php
<?php
include_once( PS_ADMIN_DIR . '/../classes/AdminTab.php' );

class AdminTest extends AdminTab
  {
  public function __construct()
    {
    $this->table = 'test';
    $this->className = 'Test';
    $this->lang = false;
    $this->edit = true;
    $this->delete = true;
    $this->fieldsDisplay = array(
      'id_test' => array(
        'title' => $this->l( 'ID' ),
        'align' => 'center',
        'width' => 25 ),
      'test' => array(
        'title' => $this->l( 'Name' ),
        'width' => 200 )
    );

    $this->identifier = 'id_test';

    parent::__construct();
    }

  public function displayForm()
    {
    global $currentIndex;

    $defaultLanguage = intval( Configuration::get( 'PS_LANG_DEFAULT' ) );
    $languages = Language::getLanguages();
    $obj = $this->loadObject( true );

    echo '
      <script type="text/javascript">
        id_language = Number('.$defaultLanguage.');
      </script>';

    echo '
      <form action="' . $currentIndex . '&submitAdd' .  $this->table . '=1&token=' . $this->token . '" method="
post" class="width3">
        ' . ($obj->id ? '<input type="hidden" name="id_' . $this->table . '" value="' . $obj->id . '" />' :
'').'
        <fieldset><legend><img src="../img/admin/profiles.png" />' . $this->l( 'Profiles' ) . '</legend>
          <label>'.$this->l( 'Name:' ).' </label>
          <div class="margin-form">';
    foreach ( $languages as $language )
      echo '
          <div id="name_' . $language['id_lang'|'id_lang'] . '" style="display: ' . ($language
['id_lang'|'id_lang'] == $defaultLanguage ? 'block' : 'none') . '; float: left;">
            <input size="33" type="text" name="name_' . $language['id_lang'|'id_lang'] . '" value="' .
htmlentities( $this->getFieldValue( $obj, 'name', intval( $language['id_lang'|'id_lang'] ) ), ENT_COMPAT, 'UTF-
8' ) . '" /><sup>*</sup>
          </div>';
    $this->displayFlags( $languages, $defaultLanguage, 'name', 'name' );
    echo '
          <div class="clear"></div>
        </div>
        <div class="margin-form">
          <input type="submit" value="' . $this->l( ' Save ' ) . '" name="submitAdd' . $this->table . '" class="
button" />
        </div>
        <div class="small"><sup>*</sup> ' . $this->l( 'Required field' ) . '</div>
      </fieldset>
    </form> ';
    }
  }
?>
```

Coloque los archivos en línea, luego, cree la pestaña yendo a la sub-pestaña "Pestañas" en la pestaña “Empleados”. Haga clic en "Añadir nuevo" y rellene los campos con el nombre de la clase," AdminTest ". ¡No confunda "clase" con "módulos"! Elija un icono (como uno del paquete de FamFamFam), seleccione donde debe ir la pestaña y guarde. ¡Ya está listo! Ahora comience a personalizarlo a sus necesidades!

Listado de hooks de PrestaShop

He aquí un resumen de la arquitectura del módulo de PrestaShop:



Cuando una de las páginas del sitio es cargada, el motor de PrestaShop revisa los módulos que deben ser invocados para a cada uno de los hooks que conforman la página.

Le presentamos una lista de 53 hooks, disponible en PrestaShop.

Front-office

Página de inicio y artículos generales del sitio

Nombre del Hook	Ubicación del Archivo	Visible	Descripción
header	header.php	No	Llamado entre las etiquetasHEAD. Ubicación ideal para agregar archivos JavaScript y CSS.

top	header.php	Sí	Llamado en el encabezado de la página.
leftColumn	header.php	Sí	Llamado al cargar la columna izquierda.
rightColumn	footer.php	Sí	Llamado cuando se carga la columna derecha.
footer	footer.php	Sí	Llamado en el pie de la página.
home	index.php	Sí	Llamado en el centro de la página de inicio.

Ficha de Producto

Nombre del Hook	Ubicación del Archivo	Visible	Descripción
extraLeft	product.php	Sí	Llamado justo antes del enlace "Imprimir", debajo de la foto.
extraRight	product.php	Sí	Llamado justo después del bloque del botón "Añadir al carrito".
productActions	product.php	Sí	Llamado dentro del bloque del botón "Añadir al carrito", después de ese botón.
productOutOfStock	product.php	Sí	Llamado dentro del bloque del botón "Añadir al carrito", después de la información de "disponibilidad".
productfooter	product.php	Sí	Llamado antes de las pestañas.
productTab	product.php	Sí	Llamado en la lista de pestañas, como "Más información", "Hoja de datos", "Accesorios"... Ubicación ideal para una pestaña extra, cuyo contenido es manejado por el hook `productTabContent`.
productTabContent	product.php	Sí	Llamado cuando se hace clic en una pestaña. Ubicación ideal para el contenido de una pestaña extra, que ha sido definida utilizando el hook `productTab`.

Carrito

Nombre del Hook	Ubicación del Archivo	Visible	Descripción
cart	Class: Cart.php	No	Llamado después de la creación o actualización de un carrito.
shoppingCart	order.php	Sí	Llamado debajo del cuadro de artículos del carrito.
shoppingCartExtra	order.php	Sí	Llamado después del cuadro de artículos del carrito, encima de los botones de navegación.
createAccountTop	authentication.php	Sí	Llamado dentro del formulario de creación de cuentas de clientes, encima del bloque "Su información personal".
createAccountForm	authentication.php	Sí	Llamado dentro del formulario de creación de cuentas de clientes antes del botón "Registrar".
createAccount	authentication.php	No	Llamado después de la creación de la cuenta del cliente.
customerAccount	my-account.php	Sí	Llamado en la cuenta del cliente de la página de inicio, después de la lista de enlaces disponibles. Ubicación ideal para agregar un enlace a esta lista.

myAccountBlock	Module: blockmyaccount.php	Sí	Llamado dentro del bloque "Mi cuenta", en la columna izquierda, debajo de la lista de enlaces disponibles. Ubicación ideal para agregar un enlace a esta lista.
authentication	authentication.php	No	Llamado después de la identificación del cliente, sólo si la autenticación es válida (e-mail y contraseña son CORRECTOS).

Búsqueda

Nombre del Hook	Ubicación del Archivo	Visible	Descripción
search	Class: Search.php	No	Llamado después que se realiza una búsqueda. Ubicación ideal para analizar y/o manipular la consulta de búsqueda y los resultados.

Elección del Transportista

Nombre del Hook	Ubicación del Archivo	Visible	Descripción
extraCarrier	order.php	Sí	Llamado después de la lista de transportistas disponibles, durante el procesamiento del pedido. Ubicación ideal para agregar un transportista, agregado por un módulo.

Pago

Nombre del Hook	Ubicación del Archivo	Visible	Descripción
payment	order.php	Sí	Llamado cuando necesita construir una lista de soluciones de pago disponibles, durante el proceso del pedido. Ubicación ideal para permitir la elección de un módulo de pago que usted ha desarrollado.
paymentReturn	order-confirmation.php	Sí	Llamado cuando el usuario es regresado a la tienda después de haber pagado en una página externa. Ubicación ideal para mostrar una confirmación u ofrecer detalles sobre el pago.
orderConfirmation	order-confirmation.php	Sí	Un duplicado de `paymentReturn`.
backBeforePayment	order.php	No	Llamado al mostrar la lista de soluciones de pago disponibles. Ubicación ideal para redirigir al usuario en lugar de mostrar dicha lista (p.ej., finalización de 1-clic de PayPal).

Devoluciones de Mercancías

Nombre del Hook	Ubicación del Archivo	Visible	Descripción
orderReturn	order-follow.php	No	Llamado cuando el clientes solicita devolver su mercancía a la tienda, y por si se produce un nuevo error.
PDFInvoice	Class: PDF.php	Sí	Llamado al mostrar la factura en formato PDF. Ubicación ideal para mostrar contenido dinámico o estático dentro de la factura.

Back-office

General

Nombre del Hook	Ubicación del Archivo	Visible	Descripción
backOfficeTop	header.inc.php	Sí	Llamado dentro del encabezado, encima de las pestañas.
backOfficeHeader	header.inc.php	No	Llamado entre las etiquetas HEAD. La ubicación es ideal para agregar archivos de JavaScript y CSS.
backOfficeFooter	footer.inc.php	Sí	Llamado dentro del pie de página, encima de la línea "Creado por PrestaShop".
backOfficeHome	index.php	Sí	Llamado en el centro de la página de inicio.

Pedidos y detalles del pedido

Nombre del Hook	Ubicación del Archivo	Visible	Descripción
newOrder	Class: PaymentModule.php	No	Llamado durante el proceso de creación de un nuevo pedido, después de que ha sido creado.
paymentConfirm	Class: Hook.php	No	Llamado cuando el estado de un pedido se convierte en "pago aceptado".
updateOrderStatus	Class: OrderHistory.php	No	Llamado cuando el estado de un pedido se cambia, antes de ser cambiado.
postUpdateOrderStatus	Class: OrderHistory.php	No	Llamado cuando el estado de un pedido se cambia, después de ser cambiado.
cancelProduct	AdminOrders.php	No	Llamado cuando un elemento es eliminado de un pedido, después de la eliminación.
invoice	AdminOrders.php	Sí	Llamado cuando los detalles de un pedido son mostrados, encima del bloque de Información del Cliente.
adminOrder	AdminOrders.php	Sí	Llamado cuando los detalles de un pedido son mostrados, debajo del bloque de Información del Cliente.
orderSlip	AdminOrders.php	No	Llamado durante la creación de una nota de crédito, después que ha sido creada.

Productos

Nombre del Hook	Ubicación del Archivo	Visible	Descripción
addproduct	AdminProducts.php	No	Llamado cando un producto es creado o duplicado, después de dicha creación/duplicación.
updateproduct	AdminProducts.php	No	Llamado cuando un producto se actualiza con una nueva imagen, después de dicha actualización.
deleteproduct	Class: Product.php	No	Llamado cuando un producto se elimina, antes de dicha eliminación.
updateQuantity	Class: PaymentModule.php	No	Llamado durante la validación de un pedido, cuyo estado es diferente a "cancelado" o "error de Pago", para cada uno de los artículos del pedido.

updateProductAttribute	Class: Product.php	No	Llamado cuando una declinación de producto se actualiza, después de dicha actualización.
watermark	AdminProducts.php	No	Llamado cuando una imagen es agregada a un producto, después de dicha agregación.

Estadísticas

Nombre del Hook	Ubicación del Archivo	Visible	Descripción
GraphEngine	Class: ModuleGraph.php	Sí	Llamado cuando un gráfico de estadísticas es mostrado.
GridEngine	Module: GridEngine.php	Sí	Llamado cuando la red de estadísticas es mostrada.
AdminStatsModules	AdminStatsTab.php	Sí	Llamado cuando la lista de los módulos de estadísticas es mostrada.

Clientes

Nombre del Hook	Ubicación del Archivo	Visible	Descripción
adminCustomers	AdminCustomers.php	Sí	Llamado cuando los detalles de un cliente son mostrados, después de la lista de grupos a la que el cliente actual pertenece.

Transportistas

Nombre del Hook	Ubicación del Archivo	Visible	Descripción
updateCarrier	AdminCarriers.php	No	Llamado durante la actualización de un transportista, después de dicha actualización.

Solución de problemas

Si el módulo no funciona como se espera, aquí le presentamos algunas maneras de encontrar ayuda.

Foro oficial de PrestaShop

Participe en nuestro foro en http://www.prestashop.com/forums/, y busque una respuesta utilizando palabras clave relevantes. Si su búsqueda necesita ser más específica, utilice el formulario de búsqueda avanzado. Y si su búsqueda no produce nada útil, cree un nuevo hilo, donde puede utilizar cuantas palabras sean necesarias para escribir su pregunta, para eso primero deberá registrarse.

Algunos foros mantienen ciertos hilos pegados a la parte superior de todas las discusiones, por que contienen información útil, así que asegúrese de leer detalladamente.

Seguimiento de errores

Si resulta que su problema se debe a un error de PrestaShop en lugar de su código, envíe el problema al “bug-tracker” de PrestaShop: http://forge.prestashop.com/ (es necesario registrarse). Esto le permite discutir el problema directamente con los desarrolladores de PrestaShop.

Sitios web oficiales de PrestaShop

URL	Descripción
http://www.prestashop.com	Sitio web oficial de la herramienta de PrestaShop, su comunidad e información de la compañía.
http://addons.prestashop.com	Bazar para temas y módulos
http://www.prestabox.com	¡Hospede su tienda con nosotros!