Manual de Programación

Programación de Plantillas

ÍNDICE

Autor Koldo G. Castillo
Modificado el 18 de enero de 2007

1. ¿Qué es una plantilla?

Las plantillas son una forma de separar el código de procesos programados en PHP del código HTML que deben usar para generar un código HTML final.

Para evitar código en PHP como el del ejemplo siguiente:

```
$dato1 = 50; $dato2=20; $dato=1;
If ($dato==1)
  echo "<html><body>".$dato1." y ".$dato2." son los resultados</body></html>";
else
  echo "<html><body>Nos ha contado ".$dato1." que te llamas ".$dato2.". </body></html>";
```

Con los problemas que crea este tipo de programación cuando hay que hacer cambios de diseño.

Un ejemplo de plantillas que se usan tanto en el MerkaWeb® como en el MerkaGest®, podría ser el siguiente:

ejemplo.tpl

```
<html><body>{item field=/dato1}</body></html>
```

ejemplo.php

```
Plantilla = new MKWTemplate();
$Datos["dato1"] = "Texto de inicio";
$resultado = Plantilla->Template($Datos);
echo $resultado;
```

Resultado:

```
<html><body>Texto de inicio</body></html>
```

De esta manera, siempre tendremos separado el trabajo de programación de los PHP, del trabajo del técnico encargado de crear y modificar la plantilla sin necesidad de que entre en el código en PHP para hacerlo.

El ejemplo escrito anteriormente no es correcto con el nivel de la clase "*MKWTemplate*" actual. Pero nos vale para ilustrar como se maneja. Hay que tener en cuenta siempre, que esta clase forma parte de las clases de la API del gestor, con lo que espera recibir información formateada como se prepara en la API. Para más información puedes consultar el manual de programación de la API.

2. ¿Dónde se crean y como se usan?

Si tenemos un proyecto MerkaWeb® MultiWeb o MerkaGest® MultiWeb, tenemos la posibilidad de dar de alta nuevas Websites que administrar desde nuestro gestor.

En la ficha de cada Website hay un campo en el que se indica la carpeta en la que se guardan las plantillas que se usarán para esa Website. Que a su vez estará en la carpeta "*templates*" del gestor (merkaweb, multiweb, merkagest, …).

Con lo que tendríamos algo como:

```
/multiweb/templates/*
        /web1/*
                /main.tpl
                /contenido.tpl
                /archivos/*
                        /img1.gif
                        /css/*
                                /clase.css
                                /…
                        /…
                /…
        /web2/*
        /…
```

Las plantillas se pueden emplear para generar las Webs, pero también para obtener otros códigos html. Como por ejemplo emails que se quieran enviar desde la Web.

2.1. Plantillas para la Web:

Cuando un usuario quiere ver una Web y se la pide al gestor, éste carga los datos que tiene que mostrar, y después emplea la plantilla adecuada para generar la Web antes de mostrarla.

En este caso los nombres de las plantilla están ya definidos en unas constantes, que puedes consultar tanto en el manual de programación de la API como al final de este documento.

2.2. Plantillas para otros elementos:

Tiene la posibilidad de indicarle el nombre de la plantilla que quieres ejecutar o incluso pasarle el código de plantilla y pasarle los datos que debe emplear para que te devuelva un código HTML.

Por ejemplo, se pueden usar las plantillas para tener formatos de emails en la gestión de contenidos, y luego ser enviados por correo usando los datos que se quieran enviar:

1. Emails de configuración de registro.
2. Emails de recordatorio de contraseña.
3. Emails de contacto, sugerencias, …
4. …

3. Nombres y extensiones

Como se ha indicado anteriormente, aunque puedes indicar la plantilla que quieres que use, la clase "*MKWTemplate*" de la API del gestor tiene una lista de plantillas y en función al contenido que quieras mostrar y al estilo de la sección y del contenido escoge la plantilla.

La búsqueda de la plantilla sigue estos pasos:

1. Coge el nombre de la plantilla y sino se lo han dado mira en la lista a ver cual corresponde al tipo de contenido que se quiere mostrar.
2. Le añade el número de estilo y la extensión.
3. Si no existe ese archivo cambia el nombre quitando el estilo.
4. Si existe el archivo carga esa plantilla.

Por ejemplo:

1. Para cargar la plantilla de contenido tiene el nombre "contenido".
2. El contenido que quiere mostrarse tiene el estilo 5, por lo que el archivo que busca es "contenido5.tpl".
3. Si no existe busca el archivo "contenido.tpl".
4. Si existe carga la plantilla.

Hay unas constantes al final del documento donde viene todo lo predefinido en este campo. Como la extensión, la carpeta de archivos, la lista de plantillas, … .

Las plantillas están en una carpeta concreta, pero el código HTML también incluye otros archivos como CSS, Imágenes o JavaScript. Todos estos archivos deben estar guardados dentro de una carpeta de archivos en la plantilla. El nombre de esta carpeta es otra constante predefinida para todas las plantillas.

En el manual de programación de la API y hay más información acerca del comportamiento de la clase "*MKWTemplate*".

4. Procesos en las plantillas

Cuando se va ha cargar una Web la API sigue el siguiente algoritmo:

1. Carga la configuración de la Web a mostrar.
2. Carga los datos en memoria (usuario registrado, idiomas, parámetros enviados, …).
3. Si la carpeta de la plantilla tiene un archivo de procesos:
   a. Importa sus funciones.
   b. Si existe la función "*template_preprocess*", procede a realizar una llamada a dicha función pasándole como parámetro un enlace a si mismo.
4. Procesa la petición para obtener los datos que hay que enviar a la plantilla (los datos de la sección y del contenido).
5. Si existe la función "*template_posprocess*", procede como se ha indicado antes.
6. Se mandan los datos a la plantilla y se obtiene el resultado.

Si necesitamos manejar la información que genera la API, antes o después de que la procese, tenemos que crear en la carpeta de la plantilla de la Website el archivo de procesos. Y crear la función o funciones que queramos emplear que reciben como parámetro por referencia el enlace a la API.

Por Ejemplo:

```
include "formularios.inc";
define (FRONT_REG_PAGINA, 10);

function template_preprocess($MKW) {  return; }

function template_posprocess($MKW) {
  $seccion = $MKW->GetDataSeccion();
  $accion = $MKW->GetDataAccion();
  $tipo_conte = $MKW->GetDataTipo();
  $orden = $MKW->GetDataOrden();
  $filtro = $MKW->ctrlseccion;
  $idioma = $MKW->GetDataIdioma();
  $get = $MKW->GetDataGet();
  If ($seccion[CAMPO_NOMBRE]=="PORTADA") {
    …
  }
}
```

Nos sirve para poder:

1. Cargar más datos en el contenido que se va ha mostrar.
2. Cargar o ampliar los datos de la portada.
3. Registrar un formulario.
4. Realizar una acción (como enviar un email).
5. …

La API solo procesa la información que se quiere mostrar y después la pasará por una plantilla, pero todas las acciones que queremos que nuestra Web realice deben ser programas desde una de estas funciones.

Por ejemplo:

1. Recomendar la Web mandando un email a un amigo.
2. Mostrar y enviar el formulario de contacto.
3. Registrarse en el boletín.
4. …

5. Lenguaje de programación "Meta-Ordenes"

Como se indica en el título, más que un lenguaje de programación en realidad es un conjunto de "*Meta-Ordenes*" embebido en código HTML.

El procesador de plantillas coge el código de la plantilla y va buscando las "*Meta-Ordenes*" y las sustituye por su valor equivalente después de procesarlas. Finalmente devuelve un código HTML donde ha sustituido todas las "*Meta-Ordenes*" por su valor adecuado según la lista de datos que le han pasado.

Todo esto desde la clase "*MKWTemplate*" que puedes consultar en el manual de programación de la API.

Los nombres de los campos empleados en las explicaciones, así como todos los inicios y finales de bloque, …, pueden cambiar ya que son constantes definidas en la "*APIDefines.inc*" y listadas al final de este manual.

Al crear las "*Meta-Ordenes*", los valores de los campos que no se especifican cogen por defecto sus valores nulos (0) o vacíos.

5.1. META-ORDENES:

5.1.1. Bloques:

Los bloques son "Meta-Ordenes" compuestas por una orden de inicio y una de fin, y cuya finalidad es actuar sobre el código que hay entre ambas órdenes. Puede haber bloques anidados así que es muy importante darles un nombre coherente a cada uno de ellos.

Los bloques empiezan por: "{sentencia nombre=ejemplo "+campos+"}"
Y finalizan por: "{/sentencia nombre=ejemplo}"

Y en medio está el código que hay que procesar en el bloque.

5.1.1.1 {block}{/block}:

Se emplea cuando se quiere procesar el código por cada elemento que se indica en el bloque, o cuando se quiere acceder a los datos de un único elemento y no estar poniendo continuamente el nombre del elemento.

Por ejemplo, escribir todos los enlaces de un menú:

```
{block … field=/CAMPO_MENU:1/CAMPO_ENLACES …}
        {item … field=CAMPO_HTML …}
{/block}
```

O por ejemplo,

```
{block … field=/CAMPO_MENU:1/CAMPO_ENLACES:5 …}
        {item … field=CAMPO_ARCHIVO …}{item … field=CAMPO_TITULO …}
{/block}
```

Sería equivalente a

```
{item … field=/CAMPO_MENU:1/CAMPO_ENLACES:5/CAMPO_ARCHIVO …}
{item … field=/CAMPO_MENU:1/CAMPO_ENLACES:5/CAMPO_TITULO …}
```

Los campos que definen el comportamiento de un bloque son:

1. "**name**" : nombre del bloque.
2. "**src**" : si hay código que cargar de algún archivo.
3. "**field**" : campo de donde coger los datos que procesar con el código.
4. "**id**" : dentro de esos datos puedes indicar un solo elemento.
5. "**con**" : indica que el "id" es el contador desde el que empezar el bucle "sen=1" o es el límite al que llegar "sen=-1".

Algunos ejemplos:

{block name=ejemplo src=css field=/CAMPO_ENLACES} …código… {/block name=ejemplo}
Agregará el código en la plantilla "*css*" al código del bloque, y lo repetirá por cada elemento del campo "/CAMPO_ENLACES".

{block name=ejemplo src=css field=/CAMPO_ENLACES id=4}…
En este caso lo procesará una vez pasando como datos "/CAMPO_ENLACES:4"

{block name=ejemplo src=css field=/CAMPO_ENLACES id=4 sen=-1} …
En este caso lo repetirá por los elementos del 1 al 4 del campo "/CAMPO_ENLACES".

{block name=ejemplo src=css field=/CAMPO_ENLACES id=4 sen=1} …
En este caso lo repetirá por los elementos del 4 al final del campo "/CAMPO_ENLACES".

5.1.1.2 {if}{/if}:

Evalúa la condición que se le pasa en los campos que definen el bloque, y si se cumple incluirá el código que contiene. En caso contrarío lo quitará del código.

Los campos que definen su comportamiento son:

1. "**name**" : nombre del bloque.
2. "**field**" : campo que indica el dato que va ha ser comparado.
3. "**value**" : campo que indica el dato con el que comprar.
4. "**sen**" : condición que se debe cumplir (<>, ==, >, <, >=, <=) (0..5).

Algunos ejemplos:
{if name=ejemplo field=/CAMPO_ID sen=1 value=0} Sin Identificar {/if name=ejemplo}
Si el campo de identificación es nulo, aparecería el texto indicado.

{if name=ejemplo field=/CAMPO_ENLACES sen=0 value=vector()} … {/if name=ejemplo}
Si el campo de enlaces no es un vector vacío, procesará el código.

5.1.1.3 {php}{/php}:

Este es el único bloque que no permite bloques anidados de ningún tipo. Por que el código que contiene será ejecutado por el intérprete de PHP. Por lo tanto tampoco requiere nombre, y no tiene ningún campo que haya que definir.

Puede ser muy útil en tiempo de depuración para saber con que datos estamos llegando a un bloque.

Ejemplo:

```
{php}echo "HOLAAAAAA"; {/php}
{block …}
        {php}print_r($bloqueDatos);{/php}
{/block …}
```

5.1.2. Campos:

Los campos son "*Meta-Ordenes*" que se evalúan y son sustituidas por el valor resultante.

5.1.2.1 {item}: Busca el valor del campo especificado y lo devuelve formateado:

1. "**name**" : nombre del campo.
2. "**field**" : campo dentro de los datos que se están procesando donde está el valor a ser devuelto.
3. "**id**" : dentro de este valor a devolver, escoge un elemento concreto.
4. "**size**" : si es un texto, indica el tamaño máximo y si lo excede lo corta y le añade el texto pasado en el campo "*format*". Si es numérico, indica el número de decimales.
5. "**format**" : si es un texto que supera el campo "*size*" en tamaño, lo corta y le añade su valor al final. Si no ha sido definido el campo "*size*" supone que es una fecha, y es el formato de la fecha. Aunque si no hay campo que formatear, coge la fecha actual.
6. "**tags**" : si se indica un valor 0 quitaría las sentencias HTML que pudiera contener el valor que se está procesando. Si no está definido o vale 1, no quitará nada.
7. "**block**" : en caso de que quiera acceder a datos de un bloque superior al anidado.

Por ejemplo,

{item name=titulo field=/CAMPO_TITULO size=30 format=… block=categoria}

Si el título tiene un tamaño mayor de 30 caracteres lo cortará a esa medida y le pondrá "…" al final.

{item name=titulo field=/CAMPO_PRECIO size=2}

Saca el precio con 2 decimales.

*5.1.2.2 {enlace}:*genera un enlace según la función de formateo en "*APIFormat.inc*":

1. "**name**" : nombre del campo.
2. "**seccion**" : sección que mostrar.
3. "**ctipo**" : tipo del contenido.
4. "**contenido**" : código del contenido.
5. "**accion**" : acción que agregar al enlace.
6. "**ssl**" : usará conexión segura (SSL) si está a 1.
7. "**block**" : en caso de que quiera acceder a datos de un bloque superior al anidado.

Por ejemplo

{enlace name=portada seccion=//CAMPO_ID ctipo=CONTENIDO_NINGUNO}

http://www.dominio.com/index.php?idioma=es&seccion=1&ctipo=0&contenido=0

5.1.2.3 {dicc}: busca en el diccionario según el idioma en el que se está viendo la web la traducción de la palabra que buscamos.

1. "**name**" : nombre del campo.
2. "**field**" : campo dentro de los datos que buscará en el diccionario.
3. "**id**" : dentro de este valor a buscar, escoge un elemento concreto.
4. "**block**" : en caso de que quiera acceder a datos de un bloque superior al anidado.

Por ejemplo

{dicc name=texto field=buscar}

Escribiría la traducción en el idioma en el que se esté viendo la Web de esa palabra.

5.1.2.4 {assign}: crea una variable global mientras se procesa la plantilla.

1. "**name**" : nombre de la variable global a generar.
2. "**value**" : campo dentro de los datos que buscará en el diccionario.
3. "**id**" : dentro de este valor a buscar, escoge un elemento concreto.
4. "**block**" : en caso de que quiera acceder a datos de un bloque superior al anidado.

Por ejemplo,

{assign name=identificador value=//CAMPO_ID}

Crearía una variable global llamada "*identificador*" y que contendría el valor de "*//CAMPO_ID*".

5.1.2.5 {value}: devuelve el valor de la variable global.

1. "**name**" : nombre de la variable global a devolver.

Por ejemplo,

{value name=identificador }

Presentaría en pantalla el valor que habíamos guardado en esa variable.

5.2. DATOS:

Cuando tenemos que especificar el valor que queramos que cojan los campos, podemos acceder a datos de las siguientes maneras:

5.2.1. Literales:

Siempre podemos especificar datos literales, es decir, números, textos y constantes existentes en "*APIDefines.inc*". En el caso de los texto hay que indicar con comillas simples que lo es, excepto en el campo "*format*" cuando es el final de un texto cortado, que como ya supone que es un texto no hace falta agregar las comillas.

No puede usarse espacio en blanco, tampoco entre comillas, ya que el espacio en blanco es el carácter usado para identificar la separación entre campos.

5.2.2. Datos del bloque en proceso:

El bloque que se está procesando aporta unos datos, y para acceder a ellos se inicia el nombre del campo con el texto "/". Como hemos estado viendo en los ejemplos anteriores.

Por ejemplo, "/CAMPO_ID" accederá a $bloqueDatos[CAMPO_ID], suponiendo que $bloqueDatos es la variable que contiene los datos del bloque.

Los datos que se están procesando pueden contener vectores de datos, por lo que en el nombre se puede construir una variable compleja. Como hacerlo se explica en un punto posterior.

5.2.3. Datos del bloque inicial:

Para acceder a los datos del primer bloque de datos se inicia el nombre del campo con el texto "//". Como hemos podido ver en algún ejemplo anterior.

Por ejemplo, "//CAMPO_ID" accederá a $this->datos[CAMPO_ID], suponiendo que $ this->datos es la variable que contiene los datos iniciales antes de empezar a procesas la plantilla.

Igual que en el punto anterior, los datos iniciales pueden contener vectores de datos, por lo que en el nombre se puede construir una variable compleja. Como hacerlo se explica en un punto posterior.

5.2.4. Variables globales del proceso:

Si hemos creado variables globales durante el proceso, podemos acceder a ellos tan solo escribiendo su nombre, como si fueran constantes.

5.2.5. Un elemento del gestor:

Si queremos buscar un elemento del gestor, podemos hacerlo iniciando el nombre con el texto "#". En el campo "*nombre*" buscaría el tipo de dato que quieres buscar (por constante o valor numérico), y en el campo "*id*" el identificador del elemento.

Por ejemplo, "#CONTENIDO_CONTENIDO_CONTENIDO" e "id=5", recogería los datos del contenido número 5.

5.2.5. Variables de la función de evaluación:

1. **$migas** : los datos procesando la API.
2. **$idioma** : abreviatura del idioma en actual;
3. **$idiomas** : listado de idiomas disponibles, con abreviatura y nombre.
4. **$seccion** : código de la sección actual.
5. **$contenido** : código del contenido actual.
6. **$tipo_conte** : tipo de contenido.
7. **$accion** : acción.
8. **$orden** : orden.
9. **$get** : variables recibidas por el método GET.
10. **$post** : variables recibidas por el método POST.
11. **$usuario** : datos del usuario identificado en la web.

5.2.6. Variables de la clase MKWTemplate():

Si es necesario, se puede acceder a las variables propias de la clase a través de la sentencia "*$this->nombre_variable*";

Para obtener el listado completo, consultar el manual de programación de la API donde se da información de esta clase.

5.2.7. Variables de la API:

La clase "*MKWTemplate*", tiene una variable que de enlace con la API que lo ha creado. Se puede acceder a través de dicha variable a las variables y funciones de la API. Para ello emplearíamos una sentencia como "*$this->MKW->nombre_variable*".

5.3. VARIABLES COMPLEJAS:

Con los datos que se están procesando contienen vectores de datos, se puede construir una variable completa que recorra el vector para obtener el dato que precisamos.

Para bajar de niveles por el vector usaremos el carácter "/" y para escoger un elemento del vector el carácter ":" seguido del número. Por ejemplo, para una estructura como esta:

```
Datos = "nombre" => "prueba"
        "nivel1" =>
            "1" =>
                "nombre" => "nivel 1.1"
                "nivel2" =>
                    "nombre" => "nivel 1.1.2"
            "2" =>
                "nombre" => "nivel 1.2"
```

Los siguientes accesos darían como resultado:

/nombre	prueba
/nivel1:1/nombre	nivel 1.1
/nivel1:1/nivel2/nombre	nivel 1.1.2
/nivel1:2/nombre	nivel 1.2

6. Resumen de "Meta-Ordenes".

Sentencias y sus campos:

{block name=a field=b id=c sen=d src=e block=x**}**…código…**{/block name=a}**

{if name=a field=b value=c con=d block=x**}…código…{/block name=a}**

{php}…código…**{/php}**

{item name=i field=a id=b size=c format=d tags=e block=x**}**

{dicc name=d field=a id=b block=x**}**

{assign name=d value=a id=b block=x**}**

{value name=d**}**

Colores:

Código a procesar
Campos opcionales
Campos obligatorios

Datos a los que se pueden acceder:

Literales

/	(datos del bloque de datos local)
/campo:1/campo	
//	(datos del bloque de datos inicial)
//campo/campo:1/campo	
#	(buscar un elemento: 'nombre' es el tipo e id el código)
nombre_variable	(para las variables globales)
$this->variable_clase	(variables de la clase)
$this->MKW->variable_api	(variables de la API)
$migas, $idioma, $idiomas, $seccion, $contenido, $tipo_conte, $accion, $orden, $usuario, $get, $post	(variables de la función de procesado)

7. Constantes de la APIDefines.inc para las plantillas.

Además de toda la información a la que puede acceder la plantilla, tiene unas constantes creadas para definir su comportamiento:

```
define(TEMPLATE_PLANTILLAS_SEPARADOR, "|");
define(TEMPLATE_PLANTILLAS_EXTENSION, ".tpl");
define(TEMPLATE_PLANTILLAS_PROCESOS, "process.inc");

define(TEMPLATE_PLANTILLAS_LISTADO_CONTENIDO,
        "main".TEMPLATE_PLANTILLAS_SEPARADOR.
        "menu".TEMPLATE_PLANTILLAS_SEPARADOR.
        "contenido".TEMPLATE_PLANTILLAS_SEPARADOR.
        "documento".TEMPLATE_PLANTILLAS_SEPARADOR.
        "imagen".TEMPLATE_PLANTILLAS_SEPARADOR.
        "album".TEMPLATE_PLANTILLAS_SEPARADOR.
        "boletin".TEMPLATE_PLANTILLAS_SEPARADOR.
        "canal");

define(TEMPLATE_PLANTILLAS_LISTADO_LISTADOS,
        "main".TEMPLATE_PLANTILLAS_SEPARADOR.
        "lstmenus".TEMPLATE_PLANTILLAS_SEPARADOR.
        "lstcontenidos".TEMPLATE_PLANTILLAS_SEPARADOR.
        "lstdocumentos".TEMPLATE_PLANTILLAS_SEPARADOR.
        "lstimagenes".TEMPLATE_PLANTILLAS_SEPARADOR.
        "lstalbunes".TEMPLATE_PLANTILLAS_SEPARADOR.
        "lstboletines".TEMPLATE_PLANTILLAS_SEPARADOR.
        "lstcanales");

define(TEMPLATE_DELIMIT_IN_INI, "{");
define(TEMPLATE_DELIMIT_IN_FIN, "}");
define(TEMPLATE_DELIMIT_OUT_INI, "{/");
define(TEMPLATE_DELIMIT_OUT_FIN, "}");
define(TEMPLATE_DELIMIT_SEPARATOR, " ");
define(TEMPLATE_DELIMIT_ASIGN, "=");

define(TEMPLATE_DEFINE_CAMPO_INICIO, "//");
define(TEMPLATE_DEFINE_CAMPO_LOCAL, "/");
define(TEMPLATE_DEFINE_CAMPO_BUSCAR, "#");
define(TEMPLATE_DEFINE_CAMPOS_CAMPOS, "/");
define(TEMPLATE_DEFINE_CAMPOS_ID, ":");

define(TEMPLATE_DEFINE_BLOQUE, "block");
define(TEMPLATE_DEFINE_ITEM, "item");
define(TEMPLATE_DEFINE_ENLACE, "enlace");
define(TEMPLATE_DEFINE_PHP, "php");
define(TEMPLATE_DEFINE_IF, "if");
define(TEMPLATE_DEFINE_DICC, "dicc");
define(TEMPLATE_DEFINE_ASSIGN, "assign");
define(TEMPLATE_DEFINE_VALUE, "value");

define(TEMPLATE_CAMPO_NOMBRE, "name");
define(TEMPLATE_CAMPO_CAMPO, "field");
define(TEMPLATE_CAMPO_PADRE, "block");
define(TEMPLATE_CAMPO_NUMERO, "id");
define(TEMPLATE_CAMPO_SRC, "src");
```

```
define(TEMPLATE_CAMPO_SENTIDO, "sen");
define(TEMPLATE_CAMPO_FORMAT, "format");
define(TEMPLATE_CAMPO_SIZE, "size");
define(TEMPLATE_CAMPO_CONDICION, "con");
define(TEMPLATE_CAMPO_VALOR, "value");
define(TEMPLATE_CAMPO_TAGS, "tags");

define(TEMPLATE_CAMPO_CONDICION_DES, "0");
define(TEMPLATE_CAMPO_CONDICION_IGU, "1");
define(TEMPLATE_CAMPO_CONDICION_MAY, "2");
define(TEMPLATE_CAMPO_CONDICION_MEN, "3");
define(TEMPLATE_CAMPO_CONDICION_MAI, "4");
define(TEMPLATE_CAMPO_CONDICION_MEI, "5");


define(TEMPLATE_BLOQUE_MAIN, "main");
define(TEMPLATE_ARCHIVO_MAIN, "main");
define(TEMPLATE_BLOQUE_CONTENIDO, "contenido");
```

8. Ejemplo de plantilla para una sección.

```
<html>
…
<meta name="language" content="{item name=idioma field=$idioma}">
…
{block name=metas field=/CAMPO_METAS}
        {item name=html field=/CAMPO_HTML}
{/block name=metas}
…
<link href="./archivos/css/estilos.css" rel="stylesheet" type="text/css" media="screen" />
…
<link rel="alternate" title="rss" type="application/rss+xml" title="RSS 2.0" href="{enlace name=enlace seccion=19
ctipo=CONTENIDO_CANAL_CANAL contenido=1 accion='' ssl=0}" target="_blank"/>
…
<a href="." alt="{dicc name=dicc value='titulo'}" title="{dicc name=dicc value='titulo'}">
        <img alt="{dicc name=dicc value='titulo'}" title="{dicc name=dicc value='titulo'}"
src="./archivos/img/logo_alimentatec_blanco.gif" width="229" height="91" alt="" hspace="0" vspace="0" border="0" />
</a>
{item name=banner field=/CAMPO_GRAFICO/CAMPO_HTML}
…
{block name=menu field=/CAMPO_MENUS id=1}
<ul id="navlist">
{block name=submenu field=/CAMPO_ENLACES}
        <li id="active">
                {item name=html field=/CAMPO_HTML}
                <ul style="display: block;" id="subnavlist{item name=html field=/CAMPO_ID}">
                {block name=enlaces field=/CAMPO_ENLACES}
                        <li>{item name=html field=/CAMPO_HTML}</li>
                {/block name=enlaces}
                </ul>
        </li>
{/block name=submenu}
</ul>
{/block name=menu}
…
{block name=menu field=/CAMPO_MENUS:2}
<ul id="navlist">
{block name=submenu field=/CAMPO_ENLACES}
        <li id="active">
                {item name=html field=/CAMPO_HTML}
                <ul style="display: block;" id="subnavlist{item name=html field=/CAMPO_ID}">
                {block name=enlaces field=/CAMPO_ENLACES}
                        <li>{item name=html field=/CAMPO_HTML}</li>
                {/block name=enlaces}
                </ul>
        </li>
{/block name=submenu}
</ul>
{/block name=menu}
…
{if name=estilo field=$accion sen=0 value='extender'}
        <div id="contenido">
{/if name=estilo}
{if name=estilo field=$accion sen=1 value='extender'}
        <div id="contenidocompleto">
{/if name=estilo}
…
{item name=contenido field=//CAMPO_CONTENIDO_HTML}

{if name=solo_seccion field=$tipo_conte sen=1 value=0}
…
{/if name=solo_seccion}

</html>
```

9. Ejemplo de plantilla para un contenido.

```
{assign name=idcontenido value=/CAMPO_ID}
{assign name=idpagina value=0}
{assign name=enlaces value=0}

<div class="rastromigas"><a id="arriba" name="arriba"></a>
        <a href="." alt="{dicc name=dicc value='inicio'}" title="{dicc name=dicc value='inicio'}">{dicc name=dicc
value='inicio'}</a> >
{if name=seccion field=$tipo_conte sen=0 value=0}{if name=contenido_seccion field=//CAMPO_ID sen=0 value=14}
        <a href="{enlace name=enlace seccion=//CAMPO_ID ctipo=CONTENIDO_NINGUNO contenido=0 accion='' ssl=0}"
alt="{item name=campo field=//CAMPO_TITULO}" title="{item name=campo field=//CAMPO_TITULO}">{item
name=campo field=//CAMPO_TITULO size=20}</a> >
{/if name=contenido_seccion}{/if name=seccion}
{item name=titulo field=/CAMPO_TITULO size=20 format=...}</div>

<div class="contenidohomesa" >
        <div class="barratitulo_sa">{if name=sec_por field=//CAMPO_ID sen=0 value=14}{item name=titulo
field=//CAMPO_TITULO size=30 format=...}{/if name=sec_por}{if name=sec_por field=//CAMPO_ID sen=1
value=14}{item name=titulo field=/CAMPO_TITULO size=30 format=...}{/if name=sec_por}</div>
        <div>
        <H5>{if name=sec_por field=//CAMPO_ID sen=0 value=14}{item name=titulo field=//CAMPO_TITULO} -
 “{/if name=sec_por}{item name=titulo field=/CAMPO_TITULO}{if name=sec_por field=//CAMPO_ID sen=0
value=14}”{/if name=sec_por}</H5>
        <p>
                {if name=grafico field=/CAMPO_GRAFICO sen=0 value=vector()}
                <IMG src="{item name=titulo field=/CAMPO_GRAFICO/CAMPO_ARCHIVO}" alt="{item name=titulo
field=/CAMPO_TITULO}" title="{item name=titulo field=/CAMPO_TITULO}" align=left hspace=5 vspace=5 border=0>
                {/if name=grafico}
                {if name=resumen field=/CAMPO_RESUMEN sen=0 value=''}
                <P class=fondosa align=left>{item name=titulo field=/CAMPO_RESUMEN}</p>
                {/if name=resumen}
                {item name=titulo field=/CAMPO_TEXTO}
        </p>
```

10. Ejemplo de process.inc.

```
include "formularios.inc";

define (FRONT_REG_PAGINA, 10);

function template_preprocess($MKW) {
	return;
}

function template_posprocess($MKW) {

$seccion = $MKW->GetDataSeccion();
$accion = $MKW->GetDataAccion();
$tipo_conte = $MKW->GetDataTipo();
$orden = $MKW->GetDataOrden();
$filtro = $MKW->ctrlseccion;
$idioma = $MKW->GetDataIdioma();
$get = $MKW->GetDataGet();

if ($seccion[CAMPO_NOMBRE]=="PORTADA") {
	for ($i=2; $i<=4; $i++) {
		$menu = $seccion[CAMPO_MENUS][6][CAMPO_ENLACES][$i];
		for ($j=1; $j<=(integer)$menu[CAMPO_ENLACES][CAMPO_NUMERO]; $j++)
			$menu[CAMPO_ENLACES][$j]["enlace"] = $MKW-
>GetContenido($menu[CAMPO_ENLACES][$j][CAMPO_CONTENIDO]);
		$seccion[CAMPO_MENUS][6][CAMPO_ENLACES][$i] = $menu;
	}
	$MKW->SetDataSeccion($seccion);
}

switch ($orden) {
	case "contacto":
		$contenido = $MKW->GetContenido("EMAIL/contacto");
		$resOk =  $MKW->EnviarEmail ($MKW->boletin_email,$MKW->boletin_entidad,$MKW-
>boletin_email,$MKW->boletin_entidad,$contenido,$MKW->Migas[CAMPO_MIGAS_PARAMETROS][CAMPO_MIGAS_POST]);
		if ($resOk) {
			$contenido = $MKW->GetContenido("MSG/envio_ok");
		} else {
			$contenido = $MKW->GetContenido("MSG/envio_no");
			$accion="contacto";
		}
		$MKW->SetDataContenido($contenido);
	break;
	case "enviar":
		$contenido = $MKW->GetContenido("EMAIL/enviar_amigo");
		$toname = $MKW->Migas[CAMPO_MIGAS_PARAMETROS][CAMPO_MIGAS_POST]["nombredestino"];
		$to = $MKW->Migas[CAMPO_MIGAS_PARAMETROS][CAMPO_MIGAS_POST]["emaildestino"];
		if ($toname=="" || $to=="")
			$resOk=false;
		else
			$resOk =  $MKW->EnviarEmail ($MKW->boletin_email,$MKW-
>boletin_entidad,$to,$toname,$contenido,$MKW->Migas[CAMPO_MIGAS_PARAMETROS][CAMPO_MIGAS_POST]);
		if ($resOk) {
			$contenido = $MKW->GetContenido("MSG/envio_ok");
		} else {
			$contenido = $MKW->GetContenido("MSG/envio_no");
			$accion="enviar";
		}
		$MKW->SetDataContenido($contenido);
	break;
	default:
}

$contenido=$MKW->GetDataContenido();
switch ($accion) {
	case "registro":
		$contenido = $MKW->GetContenido("INFO/contacto");
	case "contacto":
		$contenido[CAMPO_TEXTO] .= FormContacto();
```

```
                break;
        case "enviar":
                $contenido[CAMPO_TEXTO] .= FormEnviarAmigo();
                break;
        case "buscar":
                $contenido=ResultadoBusqueda(&$MKW, $filtro, $idioma);
        break;
}
$MKW->SetDataContenido($contenido);
return;
}

// ********************************************************************************
// Esta función podría haberse creado usando una plantilla
// ********************************************************************************

function ResultadoBusqueda($MKW, $filtro, $idioma) {

$busqueda.="<div id=\"resultadosLotus\"><img border=\"0\" width=\"60\" src=\"./img/documento.gif\"
alt=\"Representación de un documento\" style=\"float:left\"/></a>Se han encontrado <strong>0 referencias</strong> en
las Bases de Datos de Actualidad Científica. En la parte inferior puedes visualizar las referencias (contenidos) encontrados
para ese término en el resto de secciones de alimentatec.<div class=\"spacer\"></div></div>";


$keyword = $MKW->Migas[CAMPO_MIGAS_PARAMETROS][CAMPO_MIGAS_POST]["keyword"];
if ($keyword=="")
        $keyword = $MKW->Migas[CAMPO_MIGAS_PARAMETROS][CAMPO_MIGAS_GET]["keyword"];

$resultado=$MKW->GetContenidosCriterio($keyword,"conFecMod DESC", $MKW-
>Migas[CAMPO_MIGAS_PARAMETROS][CAMPO_MIGAS_GET]["pagina"], FRONT_REG_PAGINA);

$ini = ((($resultado[CAMPO_PAGINA]-1)*FRONT_REG_PAGINA)+1);
$fin = $ini + ($resultado[CAMPO_NUMERO])-1;
$busqueda.="<div  class=\"paginacion\">Encontados ".$resultado[CAMPO_TOTAL]." registros. Mostrando del $ini a
$fin<br/>Página ";

for ($i=1; $i<=$resultado[CAMPO_PAGINAS]; $i++) {
        if ($i==$resultado[CAMPO_PAGINA])
                $busqueda.=(($i<10)?"0":"").$i;
        else
                $busqueda.="<a href='".FormatHref($idioma, 14, CONTENIDO_CONTENIDO_CONTENIDO, 0,
'buscar&keyword='.$keyword.'&pagina='.$i)."' title='página".$i."' alt='página".$i.">".(($i<10)?"0":"").$i."</a>";
        if ($i<$resultado[CAMPO_PAGINAS]) $busqueda.=" ";
}
$busqueda.="</div><br /><br />";

for ($i=1; $i<=$resultado[CAMPO_NUMERO]; $i++) {
        $campos = $resultado[$i];
        $busqueda.="<h5 class=\"buscador\"><a href='".FormatHref($idioma, $campos[CAMPO_SECCION],
CONTENIDO_CONTENIDO_CONTENIDO, $campos[CAMPO_ID])."' title='".$campos[CAMPO_TITULO]."'
alt='".$campos[CAMPO_TITULO]."'>- ".$campos[CAMPO_TITULO]."</a></h5>";
        $resumen = strip_tags($campos[CAMPO_RESUMEN]);
        if (strlen($resumen)>300)
                $resumen = substr($resumen, 0, 300)."...";
        $busqueda.="<p>".$resumen."</p>";
}

$contenido = $MKW->GetContenido($filtro."/MSG/busqueda");
$contenido[CAMPO_TEXTO] .= $busqueda;
return ($contenido);
}

// ********************************************************************************
// Podría haberse hecho de la siguiente manera:
//      1. Crear una instancia de la clase MKWTemplate y configurarla para que coja una plantilla
//              de nombre "resultado", por ejemplo.
//      2. Ir recogiendo el resultado de la búsqueda en una variable de tipo vector.
//      3. Procesar la plantilla con la variable resultante.
//      4. Devolver el resultado.
// ********************************************************************************
```