Programmer avec SPIP

DOCUMENTATION À L'USAGE DES DÉVELOPPEURS ET WEBMESTRES

Version SPIP 2.1.0+ / mayo 2010

SPIP es un sistema de publicación y una plataforma de desarrollo. Después de un panorama rápido de SPIP, describiremos su funcionamiento técnico y explicaremos como desarrollar con SPIP, mostrando ejemplos útiles a los programadores.

Esta documentación esta destinada a un publico de webmasters con conocimientos en PHP, SQL, HTML, CSS y Javascript.

Sommaire

Introducción

Presentación de SPIP y de su funcionamiento general.

¿ Que es SPIP ?

SPIP 2.0 es un programa libre desarrollado bajo licencia GNU/GPL3. Históricamente utilizado como un sistema de publicación de contenido, el se vuelve progresivamente en una plataforma de desarrollo que permite de crear interfaces mantenibles y extensibles cual que sea la estructura de datos manejada.

¿ Que podemos hacer con SPIP ?

SPIP es particularmente adaptado para portales editoriales pero puede igualmente estar utilizado como sistema de auto-publicación (blog), de wiki, de red social, o para manejar todo dato de una base de datos MySQL, PostGres o SQLite. Unas extensiones proponen también interacciones con XML.

¿ Como funciona eso ?

SPIP 2.0 necesita a mínima PHP 4.1 (y 10 MB de memoria para PHP) así como una base de datos (MySQL, PostGres ou SQLite).

El posee una interfaz pública (front-office), visible de todos los visitantes del sitio (o en función de autorizaciones particulares) y una interfaz privada (back-office) solamente accesible a las personas autorizadas y que permite de administrar el programa y los contenidos del sitio.

Unos patrones llamados «esqueletos»

Toda la interfaz pública (en la carpeta squelettes-dist) u una parte de la interfaz privada (en la carpeta prive) utilizan, para ser visualizadas, unos patrones llamados «esqueletos», mixtura de código a producir (el más seguido HTML) y de sintaxis SPIP.

Cuando un visitante pide a visualizar la página de inicio del sitio, SPIP creara una página HTML gracias a un esqueleto nombrado `sommaire.html`. Cada tipo de página posee un esqueleto particular, tales como `article.html` (artículo), `rubrique.html` (sección)...

Estos esqueletos están analizados y compilados en lenguaje PHP. Este resultado esta puesto en caché. Son estos archivos PHP que sirven a producir luego las páginas HTML reenviadas a los visitantes de un sitio. Páginas que son también puestas en caché.

Sencillo y rápido

Los bucles `<BOUCLE>` seleccionen unos contenidos, las balizas `#BALISE` les muestran.

Lista de los 5 últimos artículos:

```
<B_art>
  <ul>
    <BOUCLE_art(ARTICLES){!par date}{0,5}>
      <li><a href="#URL_ARTICLE">#TITRE</a></li>
    </BOUCLE_art>
  </ul>
</B_art>
```

En este ejemplo, el bucle identificado `_art` hace una selección en la tabla SQL nombrada `ARTICLES`. Ella ordena los datos por fecha anti-cronológica `{!par date}` y selecciona los 5 primeros resultados `{0,5}`. La baliza `#URL_ARTICLE` muestra un vínculo hasta la página que presenta el artículo completo, mientras que la baliza `#TITRE` muestra el título del artículo.

Resultado:

```
<ul>
    <li><a href="Recursividad,246">Recursividad</a></li>
    <li><a href="Parametro">Parámetro</a></li>
    <li><a href="Argumento">Argumento</a></li>
    <li><a href="Adaptar-todos-sus-esqueletos-en-una">Adaptar
todos sus esqueletos en una solo operación</a></li>
    <li><a href="Mostrar-un-formulario-de-edición">Mostrar un
formulario de edición, si autorizado</a></li>
</ul>
```

Escritura de los esqueletos

SPIP genera páginas HTML desde unos archivos llamados esqueletos que contienen una mixtura de código HTML, de bucles y de criterios, de balizas y de filtros. Su fuerza es de poder extraer contenidos desde bases de datos de manera simple y comprensible.

Bucles

Un `bucle` permite de seleccionar contenidos desde una base de datos. Concretamente, el `bucle` sera traducido en una consulta SQL optimizada permitiendo de extraer el contenido pedido.

Sintaxis de los bucles

Entonces, un bucle declara una tabla SQL, sobre cual extraer las informaciones, y también unos `criterios` de selección.

```
<BOUCLE_nombre(TABLA){criterio}{criterio}>
 ... para cada respuesta...
</BOUCLE_nombre>
```

Un bucle posee obligatoriamente un nombre (identificador único al interior de un mismo esqueleto), este nombre esta juntado a la palabra `BOUCLE` (bucle en francés). Aquí, entonces, el nombre del bucle est «_nombre».

La tabla es definida, o por un alias (entonces escrito en mayúsculas), o por el nombre real de la tabla en su verdadera caja (mayúsculas o minúsculas), por ejemplo «spip_articles».

Los criterios son escritos después, entre llaves, por ejemplo `{par nom}` ("por nombre" en francés) para ordernar los resultados en el orden alfabético según el campo «nom» de la tabla SQL relativa.

Exemple

Este bucle lista todas las imágenes del sitio. El criterio `{extension IN jpg,png,gif}` permite de seleccionar los archivos que poseen una extensión entre las tres listadas.

```
<BOUCLE_documentos(DOCUMENTS){extension IN jpg,png,gif}>
    [(#FICHIER|image_reduire{300})]
</BOUCLE_documentos>
```

La baliza `#FICHIER` contiene la dirección del documento, al cual se aplica un `filtro` nombrado «image_reduire» ("imágen_reducir" en francés) que redimensiona el imagen automáticamente a 300 pixels si su tamaño es mas grande y devuelva una baliza HTML que permite de visualizar el imagen (baliza `<img>`)

Sintaxis completa de los bucles

Los bucles, como las balizas, poseen una sintaxis que permite múltiplas composiciones. Unas partes opcionales están visualizadas solo una vez (y no para cada elemento). Una otra parte alternativa esta visualizada solo si el bucle no devuelve ningún contenido. Así es la sintaxis (`x` es el identificador del bucle):

```
<Bx>
    una sola vez antes
<BOUCLEx(TABLA){criterio}>
    para cada elemento
</BOUCLEx>
    una sola vez después
</Bx>
    mostrar eso solo si no hay resultado
<//Bx>
```

Exemple

Este bucle selecciona los 5 últimos artículos publicados sobre el sitio. Aquí, las balizas HTML `<ul>` y `</ul>` solo serán mostrados una vez, y solo si unos resultados están encontrados para los criterios de selección. Si ningún artículo fuera publicado, las partes opcionales del bucle no serian mostradas.

```
<B_ultimos_articulos>
  <ul>
<BOUCLE_ultimos_articulos(ARTICLES){!par date}{0,5}>
  <li>#TITRE, <em>[(#DATE|affdate)]</em></li>
</BOUCLE_ultimos_articulos>
  </ul>
```

```
</B_ultimos_articulos>
```

La baliza #DATE (fecha, en francés) muestra la fecha de publicación del artículo. Un `filtro` «affdate» (muestra fecha, en francés) esta afectado a la baliza y permite de escribir la fecha en el idioma del contenido.

Resultado :

```
<ul>
  <li>Contenido de un archivo exec (esqueleto), <em>13 de
octubre de 2009</em></li>
  <li>Liens AJAX, <em>1er octobre 2009</em></li>
  <li>Obligar el idioma según el visitor, <em>27 de
septiembre de 2009</em></li>
  <li>Definition, <em>27 September 2009</em></li>
  <li>List of current pipelines, <em>27 September
2009</em></li>
</ul>
```

Los bucles imbricados

Es siguiente útil de imbricar un bucle en un otro para mostrar lo que queremos. Estas imbricaciones permiten de utilizar unos valores del primer bucle como criterio de selección del segundo.

```
<BOUCLEx(TABLA>){criterios}>
    #ID_TABLA
    <BOUCLEy(OTRA_TABLA){id_tabla}>
        ...
    </BOUCLEy>
</BOUCLEx>
```

Exemple

Aquí, listamos los artículos contenidos en las primeras secciones del sitio gracias al criterio `{racine}` (raíz, en francés) que selecciona las secciones de primer nivel (a la raíz del sitio), que llamamos generalmente «sector» :

```
<B_secs>
  <ul class='rubriques'>
  <BOUCLE_secs(RUBRIQUES){racine}{par titre}>
    <li>#TITRE
      <B_arts>
        <ul class='articles'>
        <BOUCLE_arts(ARTICLES){id_rubrique}{par titre}>
          <li>#TITRE</li>
        </BOUCLE_arts>
        </ul>
      </B_arts>
    </li>
  </BOUCLE_secs>
  </ul>
</B_secs>
```

El bucle `ARTICLES` (artículos, en francés) contiene simplemente un criterio de clasificación `{par titre}` (por titulo, en francés) y un criterio `{id_rubrique}` (identificador de la sección). Este ultimo criterio indica de seleccionar los artículos que pertenecen a la misma sección.

Resultado:

```
<ul class='rubriques'>
    <li>en
    </li>
    <li>fr
        <ul class='articles'>
            <li>Notes sur cette documentation</li>
            <li>Autre article</li>
        </ul>
    </li>
</ul>
```

Los bucles recursivos

Un bucle recursivo (n), contenido en un bucle pariente (x), permite de ejecutar el bucle (x) una nueva vez, transmitiendo automáticamente los parámetros necesarios. Entonces, dentro del bucle (x), llamamos este mismo bucle (eso se llama la recursividad) con otros argumentos. Este proceso se repite mientras que el bucle llamado sigue volviendo resultados.

```
<BOUCLEx(TABLA){id_parent}>
  ...
  <BOUCLEn(BOUCLEx) />
  ...
</BOUCLEx>
```

Cuando un sitio posee numerosas subsecciones, o numerosos mensajes de foro, estos bucles recursivos están utilizados seguido. Se pueden también fácilmente mostrar elementos idénticos jerarquizados.

Exemple

De esta manera vamos a mostrar la lista completa de secciones del sitio. Para eso, hacemos un primer bucle sobre las secciones, con un criterio para seleccionar las secciones hijas de la secciones en curso: `{id_parent}` (identificador del pariente). Ordenamos también por numero (un orden dado a las secciones para mostrar las voluntariamente en un cierto orden), luego por titulo de sección.

```
<B_secs>
  <ul>
  <BOUCLE_secs(RUBRIQUES){id_parent}{par num titre,
titre}>
    <li>#TITRE
    <BOUCLE_sub_secs(BOUCLE_rubs) />
    </li>
  </BOUCLE_secs>
  </ul>
</B_secs>
```

Al primer pasaje en el bucle, `id_parent` listara las secciones a la raíz del sitio. Para estas secciones, el campo SQL `id_parent` vale cero. Una vez la primer sección mostrada, el bucle recursivo esta llamado. SPIP llama de nuevo el bucle «_secs». Esta vez la selección `{id_parent}` no es la misma porque este criterio lista las secciones hijas de la sección en curso. Si hay sub-secciones, la primera esta mostrada. Y en seguida, y una nueva vez, pero en esta sub-sección, el bucle «_secs» esta ejecutado. Mientras que hay sub-secciones a mostrar, este proceso recursivo reempieza.

Resultado:

```
<ul>
<li>en
    <ul>
        <li>Introducción</li>
        <li>Los esqueletos
            <ul>
                <li>Bucles</li>
            </ul>
        </li>
        <li>Extender SPIP
            <ul>
                <li>Introducción</li>
                <li>Pipelines</li>
                ...
            </ul>
        </li>
        ...
    </ul>
</li>
<li>fr
    <ul>
        <li>Introduction</li>
        <li>Écriture des squelettes
            <ul>
                <li>Boucles</li>
                <li>Balises</li>
                <li>Critères de boucles</li>
                ...
            </ul>
        </li>
        ...
```

```
    </ul>
</li>
</ul>
```

En savoir plus !

Entender los principios de la recursividad en programación puede ser difícil. Si lo que fue explicado aquí le deja perplejo, lee el artículo dedicado de SPIP.net que explica eso con otras palabras: http://www.spip.net/es_article914.html

Bucle sobre una tabla ausenta

Cuando preguntamos a SPIP de interrogar una tabla que no existe, este muestra un error sobre la página para todos los administradores del sitio.

Sin embargo esta ausencia puede a veces estar justificada, por ejemplo si interrogamos una tabla de un plugin que puede estar activo o no. Para eso un punto de interrogación ubicado justo antes del fin de la paréntesis permite de indicar que la ausencia de esta tabla esta tolerada:

```
<BOUCLE_table(TABLA ?){criterios}>
   ...
</BOUCLE>
```

Exemple

Si un esqueleto utiliza el plugin «Agenda» (que propone la tabla EVENEMENTS (eventos)), pero que este esqueleto tiene que funcionar aún en ausencia de este plugin, es posible de escribir estes bucles:

```
<BOUCLE_eventos(EVENEMENTS ?){id_article}{!par date}>
   ...
</BOUCLE_eventos>
```

Balizas

Las balizas sirven generalmente para visualizar o calcular contenidos. Estos contenidos pueden provenir de varias fuentes: * del entorno del esqueleto, o sea de algunos parámetros trasmitidos al esqueleto ; en este caso hablamos del contexto de compilación. * del contenido de una tabla SQL al interior de un bucle. * de una otra fuente especifica. En este caso, las balizas y sus acciones deben sí o sí estar indicadas a SPIP mientras las dos fuentes precedentes pueden estar calculadas automáticamente.

Sintaxis completa de las balizas

Como los bucles, las balizas tiene partes opcionales, y pueden tener argumentos. Los asteriscos cancelan tratamientos automáticos.

```
#BALIZA
#BALIZA{argumento}
#BALIZA{argumento, argumento, argumento}
#BALIZA*
#BALIZA**
[(#BALIZA)]
[(#BALIZA{argumento})]
[(#BALIZA*{argumento})]
[ antes (#BALIZA) después ]
[ antes (#BALIZA{argumento}|filtro) después ]
[ antes (#BALIZA{argumento}|filtro{argumento}|filtro) después
]
...
```

Regla de corchetes

La escritura completa, con paréntesis y corchetes es obligatoria en cuanto uno de los argumentos de la baliza utiliza también paréntesis y corchetes o cuando la baliza contiene un filtro.

```
// riesgo de malas sorpresas:
#BALIZA{[(#BALIZA|filtro)]}
// interpretación siempre correcta:
[(#BALIZA{[(#BALIZA|filtro)]})]
// aún si esta escritura funciona en SPIP 2.0, no esta
garantizada:
#BALIZA{#BALIZA|filtro}
// la utilización de un filtro exige corchetes y paréntesis:
```

```
[(#BALIZA|filtro)]
```

Exemple

Mostrar un vínculo hasta la página de inicio del sitio:

```
<a href="#URL_SITE_SPIP">#NOM_SITE_SPIP</a>
```

(URL_SITE_SPIP: URL sitio SPIP en francés, NOM_SITE_SPIP: nombre sitio SPIP en francés)

Mostrar una baliza HTML `<div>` y el contenido de un `#SOUSTITRE` (subtitulo en francés) si existe:

```
[<div class="soustitre">(#SOUSTITRE)</div>]
```

El entorno #ENV

Llamamos entorno el conjunto de parámetros que son transmitidos a un esqueleto dado. Hablaremos también de contexto de compilación.

Por ejemplo, cuando un visitante pide la visualización del articulo 92, el identificado del articulo (92) esta transmitido al esqueleto `article.html` (artículo en francés). Dentro de este esqueleto, se puede recuperar este valor gracias a una baliza especial: `#ENV` (en francés, "entorno" se dice "environnement", por eso su contracción es "ENV"). Así que `#ENV{id_article}` mostraría "92".

Algunos parámetros están transmitidos de manera automática al esqueleto, por ejemplo la fecha actual (al momento del calculo de la página) que se puede visualizar con `#ENV{date}` ("fecha" en francés). De la misma manera, si llamamos un esqueleto con unos argumentos en el URL de la página, ellos están transmitidos al entorno.

Exemple

El URL `spip.php?page=albumes&tipo=clasico` cargara un esqueleto `albumes.html`. Dentro de este esqueleto, `#ENV{tipo}` permitirá de recuperar el valor transmitido, aquí «clasico».

Contenido de los bucles

El contenido extraído de las selecciones realizadas con bucles SPIP está mostrado gracias a las balizas. De manera automática, cuando una tabla posee un campo SQL «x», SPIP podrá mostrar su contenido si escribimos `#X`.

```
<BOUCLEx(TABLAS)>
#X - #NOMBRE_DEL_CAMPO_SQL - #CAMPO_INEXISTENTE<br />
</BOUCLEx>
```

SPIP no creara una consulta SQL de selección total (`SELECT * ...`) para recuperar las informaciones pedidas, pero, cada vez, selecciones especificas: aquí seria `SELECT x, nombre_del_campo_sql FROM spip_tables`.

Cuando un campo no existe en la tabla SQL, como aquí «campo_inexistente», SPIP no le pide en la consulta, pero prueba de recuperarlo en un bucle pariente – si hay uno – cuando el campo existe en la tabla SQL concernida. Si ningún bucle pariente posee un tal campo, SPIP le busca en el entorno, como si se escribía `#ENV{campo_inexistente}`.

Exemple

Imaginamos una tabla SQL "gatos" que contiene 5 columnas «id_gato», «raza», «nombre», «edad», «color». Podremos listar su contenido de la manera siguiente:

```
<B_gatos>
  <table>
    <tr>
      <th>Nombre</th><th>Edad</th><th>Raza</th>
    </tr>
```

```
    <BOUCLE_gatos(GATOS){par nombre}>
      <tr>
        <td>#NOMBRE</td><td>#EDAD</td><td>#RAZA</td>
      </tr>
    </BOUCLE_gatos>
  </table>
</B_gatos>
```

De manera automática, SPIP, analizando el esqueleto, entenderá que debe recuperar los campos `nombre`, `edad` y `raza` en la tabla SQL `gatos`. Sin embargo, no ira a recuperar los campos que no necesita (aquí `id_gato` y `color`), lo que evita de sobrecargar el servidor de base de datos con pedidas de campos inútiles .

Contenido de bucles parientes

A veces es útil de querer recuperar el contenido de un bucle pariente de lo corriente, a través de una baliza. SPIP disponga de una escritura para eso (n siendo el identificador del bucle querido):

```
#n:BALIZA
```

Exemple

Mostrar de manera sistemática el título de la sección al mismo tiempo que el título del artículo:

```
<BOUCLE_secs(RUBRIQUES)>
  <ul>
    <BOUCLE_arts(ARTICLES){id_rubrique}>
      <li>#_secs:TITRE - #TITRE</li>
    </BOUCLE_arts>
  </ul>
</BOUCLE_secs>
```

Balizas predefinidas

Como le hemos visto, con las balizas podemos extraer contenidos desde el entorno o desde una tabla SQL. Existen otras balizas que tienen acciones especiales explícitamente definidas.

En estos casos, ellas son declaradas (en SPIP) o en el archivo ecrire/public/balises.php, o en la carpeta ecrire/balise/

Aquí están algunos ejemplos:

- `#NOM_SITE_SPIP`: retorna el nombre del sitio
- `#URL_SITE_SPIP`: retorna el URL del sitio (sin el / final)
- `#CHEMIN`: retorna el camino de un archivo `#CHEMIN{javascript/jquery.js}`
- `#CONFIG`: permite de recuperar informaciones sobre la configuración del sitio (almacenada por parte en la table SQL «spip_meta»). `#CONFIG{version_installee}` ("versión instalada" en francés)
- `#SPIP_VERSION`: muestra la versión de SPIP
- ...

Veremos muchas otras balizas de este tipo después.

Balizas genericas

SPIP dispone de medios poderosos para crear balizas particulares que se pueden adaptar al contexto de la página, del bucle o simplemente al nombre de la baliza.

Así, es posible de declara balizas que tendrán todas el mismo prefije y efectuaron así un tratamiento común propio a cada tipo de baliza.

Estos tipos de balizas están declarados en la carpeta ecrire/balise/. Son los archivos `*_.php`.

Así podemos encontrar:

- `#LOGO_` para mostrar logotipos de artículos, de sección o otro: `#LOGO_ARTICLE` (logotipo de un artículo)
- `#URL_` para determinar un URL de un objeto SPIP, como `#URL_MOT` (URL de una palabra-clave) al interior de un bucle `MOTS` (palabras clave)

- `#FORMULAIRE_` para mostrar un formulario definido en la carpeta `/formulaires` como `#FORMULAIRE_INSCRIPTION` (formulario inscripción)

Tratamientos automáticos de balizas

La mayoría de las balizas SPIP, en particular todas las que son leídas en la base de datos, efectúan tratamientos automáticos para bloquear códigos malintencionados que podrían estar añadidos por redactores al momento de la escritura del articulo (código PHP o scripts JavaScript).

Además de estos tratamientos, otros pueden estar definidos para cada campo SQL a fin de hacer aplicar automáticamente los tratamientos sobre este campo. Estas operaciones están definidas en el archivo ecrire/public/interfaces.php por una tabla global `$table_des_traitements` ("tabla de los tratamientos" en francés). La clave de la tabla es el nombre de la baliza, y el valor es una tabla asociativa: su clave «0» define un tratamiento para cualquier tabla, una clave «nombre_de_la_tabla» (sin el prefije) define un tratamiento para una baliza de una tabla especifica.

Los tratamientos están dados por una cadenas de caracteres `funcion(%s)` explicitando las funciones a aplicar. Adentro, «%s» sera remplazado por el contenido del campo.

```
$table_des_traitements['BALISE'][]= 'tratamiento(%s)';
$table_des_traitements['BALISE']['objets']=
'tratamiento(%s)';
```

Dos usos frecuentes de filtros automáticos están definidos por unas constantes que pueden estar utilizadas:

- `_TRAITEMENT_TYPO` aplica tratamientos tipográficos,
- `_TRAITEMENT_RACCOURCIS` aplica los tratamientos tipográficos y las traducciones de atajos SPIP.

Exemple

Las balizas `#TITRE` ("título" en francés) y `#TEXTE` ("texto" en francés) reciben unos tratamientos, que se apliquen para cualquier bucle, definidos de esta manera:

```
$table_des_traitements['TEXTE'][]=
_TRAITEMENT_RACCOURCIS;
$table_des_traitements['TITRE'][]= _TRAITEMENT_TYPO;
```

La baliza `#FICHIER` ("archivo" en francés) efectúa un tratamiento solo en los bucles de documentos:

```
$table_des_traitements['FICHIER']['documents']=
'get_spip_doc(%s)';
```

Impedir los tratamientos automáticos

Los tratamientos de seguridad y los tratamientos definidos se aplican automáticamente sobre las balizas, pero es posible de evitar eso para algunas particularidades de un esqueleto. La extensión «asterisco» de una baliza es concebida para:

```
// todos los tratamientos
#BALIZA
// sin los tratamientos definidos
#BALIZA*
// sin aún los tratamientos de seguridad
#BALIZA**
```

Exemple

Retardar la aplicación de los tratamientos tipográficos y de los atajos SPIP sobre el texto de una página (el filtro `propre` ("limpio" en francés) esta aplicada normalmente automáticamente), para añadir, antes, un filtro efectuando una acción del montón:

```
[<div
class="texte">(#TEXTE*|filtro_del_monton|propre)</div>]
```

Balizas a conocer

Dentro del juego de balizas especificas que posee SPIP por omisión, una parte están utilizadas bastante seguido, y tienen entonces que estar presentadas aquí.

Nom	Description
#INSERT_HEAD_CSS (p.26)	Baliza de inserción de CSS en el para plugins.
#SESSION (p.26)	Recuperar una información de sesión
#SESSION_SET (p.27)	Definir variables de sesión
#VAL (p.27)	Devuelve un valor

#INSERT_HEAD_CSS

`#INSERT_HEAD_CSS` plazado entre las etiquetas HTML `<head>` y `</head>` permite a los plugins de añadir scripts CSS utilizando el pipeline insert_head_css (p.0). Si esta baliza no esta presente en el esqueleto, `#INSERT_HEAD` añadira el contenido del pipeline ella-misma.

En los esqueletos por omisión de SPIP, esta baliza es insertada antes del archivo CSS `habillage.css` en squelettes-dist/inc-head.html. Así, temas gráficos que sobrecargan este archivo `habillage.css` pueden también sobrecargar, en CSS, las declaraciones añadidas antes por los plugins.

#SESSION

`#SESSION{parametro}` muestra informaciones sobre el visitante conectado. Una sesión puede ser considerada como informaciones individuales, conservadas sobre el servidor durante el tiempo de visita del visitante. Así, estas informaciones pueden estar encontradas y devueltas cuando este visitante cambia de página.

La presencia de esta baliza, como para la baliza `#AUTORISER`, genera un cache diferente por visitante autentificado sobre el sitio, y un cache para los visitantes non-autentificados.

Exemple

Mostrar el nombre ("nom" en francés) del visitante si esta conocido:

```
#SESSION{nom}
```

Mostrar una información si el visitante esta autentificado sobre el sitio, es decir si el posee un `id_auteur` ("id_autor", en francés):

```
[(#SESSION{id_auteur}|oui) Usted esta autentificado ]
```

#SESSION_SET

La baliza `#SESSION_SET{parametro, valor}` permite de definir variables de sesión para un visitante, que podrán estar recuperadas por `#SESSION{parametro}`.

Exemple

Definir un perfume de vainilla !

```
#SESSION_SET{perfume,vainilla}
#SESSION{perfume}
```

#VAL

`#VAL{valor}` permite de devolver el valor que le damos, simplemente. Esta baliza sirve principalmente para enviar un primer argumento a filtros existentes.

```
#VAL{Este texto sera devuelto}
```

Exemple

Devolver un carácter con la función PHP `chr` :

```
[(#VAL{91}|chr)]   // [
[(#VAL{93}|chr)]   // ]
```

A veces el compilador de SPIP se confunde entre los corchetes que queremos escribir, y los corchetes de apertura o de cerrada de las balizas. Un ejemplo frecuente es el envío de un parámetro tabla en un formulario (`name="campo[]"`), cuando el campo esta incluido en una baliza:

```
// problema : el ] de campo[] esta confundido
// con la cerrada de la baliza #ENV
[(#ENV{mostrar}|oui)
<input type="hidden" name="campo[]" value="valor" />
]
// ningún problema aquí
[(#ENV{mostrar}|oui)
<input type="hidden"
name="campo(#VAL{91}|chr)][(#VAL{93}|chr)]" value="valor"
/>
]
```

Index

Symboles

A

D

E

F

I

N

O

P

R

S

Table des matières