Bisoños Usuarios de GNU/Linux de Mallorca y Alrededores | Bergantells Usuaris de GNU/Linux de Mallorca i Afegitons

Fulls d'estil dinàmics amb PHP (8483 lectures)

Per **Joan Miquel**, ***Joanmi*** (http://www.mallorcaweb.net/joanmiquel)

Creado el 01/12/2003 20:23 modificado el 01/12/2003 20:23

CSS

En aquest article explicarem una mica què son els fulls d'estil i com podem utilitzar-los per modelar la nostra plana web.

També explicarem com podem conseguir generar-los dinàmicament amb PHP per poder parametitzar-los o, fins i tot, crear el nostre propi sistema de "skins css" per al nostre portal web.

Índex:

Què son els fulls d'estil:

Què son?

Els fulls d'estil son una manera senzilla de separar la part estètica d'un document html del seu contingut real.

Permeten definir de quina manera s'ha de presentar la informació dels elements **html** mitjançant la identificació d'aquests en el document.

A més, es poden crear fulls d'estil diferents pels diferents mitjans físics de representació com pantalla, impresora, PDA... de manera que a l'hora de generar els continguts haurem d'escriure un únic document html i el propi navegador

sabrà com s'ho ha de fer per representar-lo correctament en cada medi.

Un full d'estil es compon d'una sèrie de regles que identifiquen un conjunt d'un o més elements i li associen unes característiques estètiques pròpies.

La identificació es pot fer en base al tipus d'element:

```
body
table
a
```

o identificar elements prèviament "marcats" al document html amb l'atribut *id*. Per exemple:
`table#tarifa`

afectaria només al tag html
`<table id="tarifa"...`

També podem crear regles per aplicar-les a un conjunt o *classe* d'objectes. L'identificador d'una clase es representa per un punt (.) seguit del nom de la classe i per indicar que un determinat tag pertany a una classe concreta utilitzarem l'atribut *class*. Per exemple, una regla definida per l'identificador:

`.equipdelstigres`

afectaria a tots aquests tags:

```
<p class="equipdelstigres">
<table width="75%" class="equipdelstigres">
<li class="equipdelstigres">
```

El bloc que identifica el o els elements s'anomena "selector". Després del selector van les "declaracions" que son les que defineixen els valors de cada propietat que volem fixar.

`color:#ff0000`

...que seria el mateix que "red" però a mi m'agrada més així (després hi ha navegadors que no entenen d'idiomes...).

Les declaracións s'escriuen de la forma *propietat:valor*, es separen amb punt i coma (;) i s'enmarquen en el que s'anomena "bloc" que es delimita amb claus ({) i (}). En altres paraules:

`body {color:#ff0000; background-color:#0000ff}`

...ens generaria el mateix *horrible* efecte que:
`<body text="#ff0000" bgcolor="#0000ff">`

Ón col·loco les regles?

Fantàstic. Ara ja sabem com escriure regles CSS. Ara només ens falta saber a quina part del document (o de fora d'ell) posar-les perque sorgeixin efecte.

Tenim dues possibilitats:

- Dirèctament embegudes dins el document.
- En un fitxer apart.

Dirèctament embegudes dins el document:

O (més normalment) a les capçaleres amb l'etiqueta <style>. Per exemple amb:

```
<style type="text/css" media="screen,print">
...
</style>
```

...definiriem el full d'estil a aplicar per pantalla i impresora. També podem especificar simplement *media="all"* si volem que s'apliqui a tots els medis. Personalment preferesc que en un medi desconegut la informació es representi sense fulls d'estil. Per exemple, per complir certes restriccions sobre la pantalla d'un PDA segurament seria impossible mostrar corrèctament tota la informació.

En un fitxer apart:

Quan tenim poques regles pot resultar interessant posar-les dirèctament sobre el propi document de la manera que acabam d'explicar. Però quan el volum comença a crèixer és molt més recomanable posar-les en un fitxer apart. Sobretot perque d'aquesta manera, el client les demana com un fitxer separat i, si tenim vàries pàgines que facin servir el mateix full d'estil aquest només s'haurà de transmetre un cop perque quedarà a la caché del navegador.

Per incloure les regles en un full d'estil separat, les posarem en un fitxer amb extensió .css i posarem una línia de l'estil de:

```
<link rel="StyleSheet" type="text/css" media="all" href="estil.css" />
```

entre les etiquetes <head> i </head> del nostre document.

Altres fulls d'estil:

A més del full d'estil que nosaltres puguem definir, existeixen altres fulls d'estil que s'apliquen paral·lelament al nostre. Són el full d'estil d'usuari i el full d'estil de navegador.

- El full d'estil de navegador defineix la forma en que, per defecte, el navegador mostra cada tipus d'element d'un document html si no hi ha una regla específica al full d'estil d'usuari o al full d'estil específic creat pel dissenyador del document.
- El full d'estil d'usuari defineix alteracions sobre el full d'estil del navegador que l'usuari pot modificar per tal de modificar la forma de de representació per defecte dels elements. Com ara, per exemple, especificar un tipus o tamany de font diferenta.

Normalment, les regles definides pel dissenyador prevaleixen sobre les d'usuari i aquestes sobre les del navegador. Encara que aquest comportament es pot canviar per una determinada declaració afegint el modificador **!important** al full d'estil corresponent per forçar, per exemple, un tamany de lletra més gros si tenim dificultats de visió. Si voleu, a la secció "Més informació", trobareu enllaços a tutorials complets i documentació més específica sobre CSS.

L'anaguet lleig que ningú es mirava...

L'etiqueta **<DIV>** que sempre ens han dit que servia per delimitar blocs lògics dins un mateix document i que nosaltres només hem vist útil per fer allò de...

```
<DIV align="center"&gt
```

...quan no teniem altre lloc on ficar lo de *align="center"*. De sobte cobra una utilitat quasi màgica!!

Bé... en realitat podriem fer-ho amb qualsevol altre tag, per exemple una taula sense voreres (amb *border=0*) i d'una sola cel·la. Però aquí el **<DIV>** és "the right way" perque no necessitam una entitat tan complexa com la taula simplement per fitar un bloc de text.

Vegem un exemple:

```
<html>
<head>
<title>ProvaCSS</title>
<style type="text/css" media="screen,print">
div#menu {
position:fixed;
top:0px; left:0px;
bottom:0px; width:20%;
background:#a0a0a0;
overflow:auto;
}
div#document {
position:fixed;
top:0px; left:20%;
bottom:0px; right:0px
overflow:auto;
}
</style>
</head>
<body>

<div id="menu">
<h1>Menú:</h1>
<ul>
<li>Opció 1
<li>Opció 2
<li>Opció 3
<li>Opció 4
</ul>
</div>

<div id="document">
<p>Cos del document...</p>
</div>

</body>
</html>
```

...ni més ni menys que la típica barra de menú lateral amb la finestra de continguts laterals i amb scrolling independent. En altres paraules: "Frames" elegants sense frames.

Igual que podem fer això ens podem montar qualsevol estructura que s'ens ocorri i, fins i tot de manera molt més flexible (i avantatjosa) que si ho féssim amb frames:

- Aquí necessitam un sol document (o 2 si separam el full d'estil, però la informació *útil* queda agrupada en el document principal).
- Podem situar els elements de la manera que volguem. No és necessari fer divisions horitzontals o verticals de tota la pantalla.
- Podem fins i tot solapar àrees del document si això ens interessa. Per exemple podriem posar un títol amb lletres grosses en un amb fons transparent solapat dirèctament al damunt dels continguts. Per això, quan treballam d'aquesta manera es diu que estam utilitzant **capes**. Existeix una propietat del CSS que permet establir l'ordre d'aquestes *capes* per determinar quines estaràn damunt quines altres.
- Si el navegador no soporta fulls d'estils els continguts es mostraràn correctament un darrera l'altre (en ordre d'aparició).
- Per si fos poc, encara podem organitzar els s dins un element <table> per definir una estructura bàsica de representació pel cas de que el navegador no soporti css.

Més informació:

Si voleu saber més sobre css, aquí vos poso alguns enllaços interessants:

Url:	Comentari:
http://www.tierradenomadas.com/tw007.phtml[16]	Tutorial molt bò de CSS (en castellà).
http://www.w3.org/TR/REC-CSS2/cover.html[17]	Especificacións del CSS2
http://www.sidar.org/recur/desdi/traduc/es/css/cover.html[18]	Traducció al castellà de les especificacións de CSS.

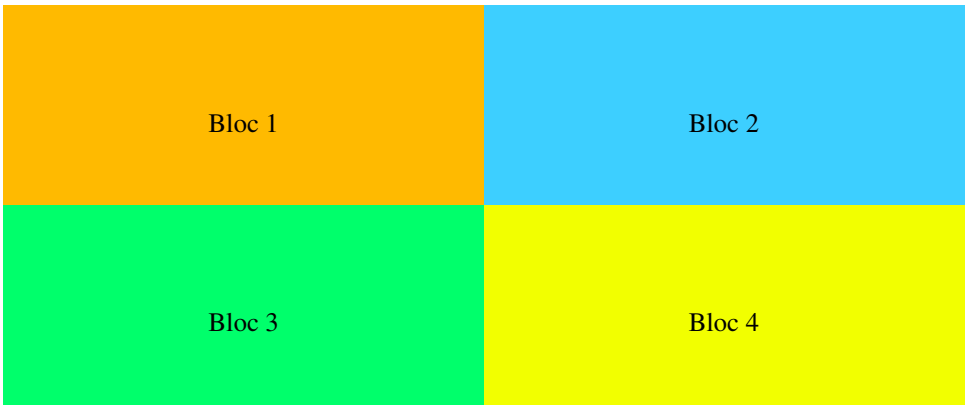
Generació dinàmica de fulls d'estil amb PHP:

Perquè?

Amb els fulls d'estil aconseguim per fi separar del tot la part estètica dels continguts pròpiament dits del document. Però si volem definir fulls d'estil complicats, moltes vegades el resultat és un document infumable ple de regles que no sabem com ordenar i entre les quals no podem ni insertar un trist comentari.

...tot això per no parlar de poder iterar l'escriptura de regles repetitives o generar automàticament un (o part d'un) full d'estil a partir de premises més simples. En altres paraules, no podem **programar** els fulls d'estil. ...O si?

Imaginem que volem crear un portal amb un full d'estil que ens defineixi les següents arees:



...resulta que si volem alterar el tamany del Bloc 1, aleshores ens veim obligats a canviar les coordenades i dimensions de tots els altres blocs!! ...quan en un script normal en PHP ho resoldriem amb una simple (o dues) constant.

Per aquesta raó s'em va ocórrer la possibilitat de generar **també** el full d'estil amb PHP.

Còm?

Sistema "inline":

El sistema més obvi per generar el codi CSS desde PHP és posar el CSS "inline" en el propi script PHP. Per exemple, i per tenir-ho separadet en un fitxer apart podriem fer ús de la sentència "*include*":

```
<style type="text/css" media="screen,print">
<?PHP include "css.inc"; ?>
</style>
```

L'avantatge d'aquest sistema és que a cada recàrrega, si volem, podem generar un full d'estil totalment diferent en funció, per exemple, dels paràmetres rebuts a la url o de l'estat de cada moment si estam utilitzant sesions. Però té l'inconvenient de que cada vegada transferim de nou tot el full d'estil amb el consegüent **overhead** que això suposa quan, en canvi, amb un full d'estil separat, aprofitam la capacitat de caché que tenen tots els navegadors cada cop que es demana el mateix fitxer.

Fitxer .css.php:

El segon que s'ens acudeix és redactar un script PHP apart que ens generi el codi css a la sortida. De manera que teòricament ens hauria de bastar amb una cosa com:

```
<link rel="StyleSheet" type="text/css" media="all" href="estil.css.php">
```

L'exemple següent mostra com no només podem generar el css amb el php, sino que també podem passar-li paràmetres per obtenir distints resultats:

index.php

```
<html>
<head>
<title>Prova CSS</title>

<script language=php>
$color && $c = "?c=$color";
echo "<link rel=\"StyleSheet\"
type=\"text/css\" media=\"all\"
href=\"dinamic.css.php$c\">\n";
</script>

</head>
<body>
<h1>
<script language=PHP>
$cad = array ( 78, 79, 32, 97, 108,
32, 112, 108, 97, 32, 100, 101,
32, 99, 97, 114, 114, 101, 116,
101, 114, 101, 115, 33);
for ($i = 0; $cad[$i]; $i++)
  echo chr($cad[$i]);
</script>
</h1>
</body>
</html>
```

dynamic.css.inc:

```
<script language=php>
$c || $c="f08f00";
echo "body
{background-color:#$c}";
</script>
```

Cada cop que solicitem un full d'estil que ja hem demanat abans (amb els mateixos paràmetres) el navegador farà servir el que ja té en caché. En canvi si en sol·licitam un amb unes especificacions distintes, el reclamarà del servidor.

ADVERTÈNCIA: Si utilitzau un fitxer css extern que executa codi PHP i cometeu errors (per exemple de sintaxi) PHP escriurà els advertiments d'error PHP a la seva sortida standard (com sempre); és a dir: al fitxer *estil.css.php*. De manera que no els veurem ni al navegador ni al "font del document" que la majoria de navegadors ens deixen inspeccionar.

Per solucionar aquest problema a l'exemple anterior, podriem implementar un senzill commutador a partir d'una constant:

```
<script language=php>
define (__debug_css__, 0);
$color && $c = "?c=$color"; // Si el nom de la variable no coïncidís amb l'usat a dinamic.css.inc hauriem de solucionar el problema del pas de paràmetres dins el bloc condicional.
if (__debug_css__) {
  echo "<pre>\n";
  include "dinamic.css.php";
  echo "</pre>\n";
} else {
  echo "<link rel=\"StyleSheet\" type=\"text/css\" media=\"all\" href=\"dinamic.css.php$c\">\n";
};
</script>
```

D'aquesta manera, en observar comportaments extranys, podriem canviar el valor de la constant __debug_css__ per llegir còmodament la informació que ens donen els missatges d'error.

Una altra solució seria invocar dirèctament la url del fitxer css des del navegador passant-li els paràmetres pertinents. Aquesta pot ser una alternativa més pràctica sobretot si tenim pocs o cap paràmetre a pasar.

Generació prèvia.

Amb el mètode anterior podem programar un generador de fulls d'estil capaç de treure una sortida en funció de les necessitats de cada moment concret. Però de vegades no necessitam res de tot això i per l'únic que volem generar el CSS amb PHP és per la comoditat de poder automatitzar la generació del CSS en funció de paràmetres que podem modificar segons les necessitats que tenguem o, simplement, per poder anar provant amb diferentes combinacions de determinats paràmetres.

En canvi, pot ser que la velocitat en que es serveix la pàgina sigui crítica. I si un full d'estil que al final (posat en serveï) ha de ser sempre el mateix s'ha de generar de nou a cada petició, sempre tardarà més que si servissim un fitxer estàtic.

Per solucionar aquest problema el que podem fer és utilitzar un full d'estil estàtic, però generat *prèviament* amb PHP. La idea es basa en el fet de que mentres el navegador va rebent la pàgina (codi html resultat de la sortida de l'execució del fitxer PHP), en el moment en que arriba a la línia html que enllaça amb el fitxer que conté el CSS, tot el codi PHP que pogués existir abans d'aquesta línia en el fitxer php original, ja s'haurà executat.

Una afirmació que, certament, sembla òbvia; però que ens du a la conclusió de que podem fer l'*estupidesa* ((C) Gilipolles 2003) d'escriure un fitxer CSS "estàtic" _abans_ de que el navegador tengui ocasió de requerir-lo al servidor.

Certament sembla una estupidesa, ja que no deixarà de consumir temps en la seva execució. Però ara podem condicionar l'execució d'aquest codi, per exemple, al valor d'una constant que podem ubicar, per exemple, en un fitxer de configuració; i que determini si s'ha de generar o no el full d'estil.

D'aquesta manera, podem tenir habilitada la generació del CSS mentres estam desenvolupant el projecte i canviar el valor d'aquesta constant abans de pujar el projecte al servidor per tal de que la versió en serveï s'estalvii la generació d'un fitxer que no canvia.

```
include "config.inc" // Fitxer ón definim __css_souce_sw__ i __css_dst_file__.

[...]

if (__css_souce_sw__ == "generated") {
  include "css/index.css.inc";
};

[...]

echo "<link rel=\"STYLESHEET\" href=\"" . __css_dst_file__ . "\" type=\"text/css\"></link>\n";
```

Generació híbrida.

Hem presentat tres mètodes de generació de fulls d'estil. Des d'un totalment dinàmic fins al darrer gairebé estàtic; cadascun amb els seus avantatges i els seus inconvenients. En un projecte real, normalment ens trobarem que una part del css ens pot interessar que sigui dinàmica mentre una altra bona part pugui ser estàtica.

Res ens impedeix combinar els sistemes anteriors per obtenir el resultat desitjat i, fins i tot, podem definir regles "genèriques" per determinats elements en fulls d'estil "estàtics" que després siguin contradites per regles més "dinàmiques" amb major preferència. L'únic que hem de tenir en compte, precisament, és l'ordre de prevalència de les regles que va en funció del lloc ón estàn definides i la posició que ocupen en aquest lloc.

En aquest sentit, cal tenir clar quin és el criteri de preferència que segueixen els navegadors a l'hora de decidir quina regla d'estil aplicar quan n'hi ha vàries que serien aplicables al mateix element.

Criteris de prevalència de les regles CSS:

1. **Origen:** Com ja s'ha explicat al principi del document, la informació del full d'estil de l'aplicació és substituïda per la de l'usuari i aquesta, al seu torn per la d'autor.
2. **Especificitat:** Si l'origen és el mateix, les regles més específiques substituëixen les més generals.
3. **Ordre d'aparició:** Finalment, si les dues regles tenen el mateix origen (òbviament ens interessa un únic origen: les regles d'autor), aleshores mana l'ordre d'aparició. Sent la darrera en haver estat declarada la que preval ("sobreescriu") sobre les anteriors.

Creació d'un sistema de pells personalitzables:

Ara que tenim les eines per automatitzar la generació dels fulls d'estil, resulta fàcil pensar en la possiblitat de crear "skins" o "pells" personalitzables pel nostre portal web.

Aprofitant la funció parse_ini_file[20] de PHP4, podem utilitzar els clàssics fitxers ".ini" per emmagatzemar els paràmetres de les diferentes àrees de la nostra pàgina.

Més encara, podem fins i tot crear capes automàtiques que es generin a partir d'un d'aquests fitxers ".ini". Això pot ser especialment útil si volem crear una capa de fons (o de "màscara" sobreposada als continguts) a base de diversos s amb imatges de fons. Podem definir la ubicació, dimensions, etc... de cada un en una secció d'un fitxer .ini i generar a partir d'aquest el codi css i el html que farà aparèixer el .

Una altra opció interessant pot ser definir les regles de cada bloc ( ) de la forma *propietat* = *valor* en una secció d'un fitxer .ini i:

- Utilitzar propietats "literals" de CSS com background, color, etc...
- Decidir quines propietats CSS permetem alterar i quines no.

- Definir altres tipus de propietats relatives al mateix objecte que no tenguin res a veure amb el CSS. És a dir: tenim agrupats tots els paràmetres de configuració d'aquell element o bloc de la pàgina.

Com a exemple, la següent funció ens extrauria les propietats desitjades d'un element ignorant les altres (que també podrien haver sigut tractades específicament amb una clàusula "case" a l'efecte i, a més, permet el valor 'default' que pot resultar còmode a l'hora de no definir certes propietats sense oblidar-nos que estàn permeses en futurs canvis a la *pell* (skin):

```
function expand_css_at (&$source) {
  $text = "";
  foreach($source as $key => $value) {
    switch (strtolower ($key)) {
      case "top": case "bottom": case "right": case "left":
      case "width": case "height": case "font-size": case "color":
      case "text-align": case "z-index":
        if (strtolower ($value) != "default") {
          $text .= "$key:$value; ";
        };
        break;
    }; // Another cases will be manually referenced.
  };
  return $text;
};
```

Un regalet ;-)

En realitat això no ho he utilitzat per implementar les pells, sino que és una funció que vaig crear per unes llibreries de generació de taules que, de moment, posen el codi css "inline". Però és una d'aquestes funcions *xorres* que fins que un no la troba en falta no se li acudeix implementar-la però després li entren ganes de modificar tot el codi que té fet per incloure-la.

Serveix per generar una regla css a partir d'una cadena buïda a la que anam afegint opcions.

```
function styleset (&$s) {
  if (func_num_args() > 1) { // Add style rule:
    $prop = func_get_arg(1);
    $val = func_get_arg(2);
    if (func_num_args() >= 4) { // Translation stablished.
      $trans = func_get_arg(3);
      $trans[$val] && $val = $trans[$val];
    };
    if (! $s) { // Initialize style string:
      $s = " style=\"";
    } else {
      // Remove last end-quote:
      $s = substr ($s, 0, strlen ($s) -1) . ";";
    };
    $s .= "$prop:$val\"";
  } else { // Initialize style rule:
    $s = "";
  };
  return 0;
};
```

Fins aquí aquest testament. Si heu arribat a aquest punt, o vos interesa de veres el tema o teniu molta moral ;X-D

Lista de enlaces de este artículo:

1. http://bulma.net/body.phtml?nIdNoticia=1923&nIdPage=2#c1
2. http://bulma.net/body.phtml?nIdNoticia=1923&nIdPage=2#c1.1
3. http://bulma.net/body.phtml?nIdNoticia=1923&nIdPage=2#c1.2
4. http://bulma.net/body.phtml?nIdNoticia=1923&nIdPage=2#c1.3
5. http://bulma.net/body.phtml?nIdNoticia=1923&nIdPage=2#c1.4
6. http://bulma.net/body.phtml?nIdNoticia=1923&nIdPage=2#c1.5
7. http://bulma.net/body.phtml?nIdNoticia=1923&nIdPage=3#c2
8. http://bulma.net/body.phtml?nIdNoticia=1923&nIdPage=3#c2.1
9. http://bulma.net/body.phtml?nIdNoticia=1923&nIdPage=3#c2.2
10. http://bulma.net/body.phtml?nIdNoticia=1923&nIdPage=3#c2.2.1
11. http://bulma.net/body.phtml?nIdNoticia=1923&nIdPage=3#c2.2.2
12. http://bulma.net/body.phtml?nIdNoticia=1923&nIdPage=3#c2.2.3
13. http://bulma.net/body.phtml?nIdNoticia=1923&nIdPage=3#c2.2.4
14. http://bulma.net/body.phtml?nIdNoticia=1923&nIdPage=4#c3
15. http://bulma.net/body.phtml?nIdNoticia=1923&nIdPage=4#c3.1
16. http://www.tierradenomadas.com/tw007.phtml
17. http://www.w3.org/TR/REC-CSS2/cover.html
18. http://www.sidar.org/recur/desdi/traduc/es/css/cover.html
19. http://bulma.net/
20. http://es2.php.net/manual/en/function.parse-ini-file.php

E-mail del autor: joanmi _ARROBA_ bulma.net
Podrás encontrar este artículo e información adicional en: http://bulma.net/body.phtml?nIdNoticia=1923