От стария HTML сайт – нов с PHP

Времената се менят. HTML вече се учи в училище, а най-обикновения хостинг струва по-малко от седемдесет лева за година, заедно с домейна. При това с поддръжка на PHP и MySQL.
А как да оползотворим всичко това?

Въпросът естествено не е зададен към управителите на големите фирми, а към всички онези, които имат малки лични или фирмени сайтове и ги поддържат сами. В края на краищата компютърът е за всички, не само за тесните IT специалисти, нали!
Имали сте класически сайт с фрейм, и нямате идея за какво бихте използвали PHP и дали няма да си навлечете допълнителни главоболия при подобен експеримент.
PHP всъщност ви дава едно модерно и леко решение – резултатът е по-просто обслужване на промените в сайта. А ако сега опитате да промените нещата, ще сте направили и първата стъпка към усвояване на големите възможности, които имат не само езика PHP, но и използването на MySQL.

Начална точка

Предполагаме, че имате хостинг с поддръжка на PHP и MySQL А може би сте свалили и инсталирали на вашия компютър PHP и сървъра Apache, или сте постъпили още по-просто, като сте свалили и инсталирали EasyPHP под GPL лиценз, който ви дава достъп до PHP, MySQL и Apache едновременно. Ако ползвате EasyPHP, трябва да разположите php файловете, които искате да изпробвате в папка, която е подпапка на www. След това да стартирате EasyPHP от старт – менюто. Когато видите, че Apache и MySQL имат зелен светофар, отворете браузер и напишете в адресна лента
http://127.0.0.1/, а после натиснете Enter
Ще видите екрана на EasyPHP и подпапките на www
Може би знаете, че ако смените разширението на файл от htm или html на php и го стартирате през браузер при горните условия, ще видите същото, както и ако не бяхте сменяли разширението, а също и че кодът на php започва с <?php и завършва с ?>

Оператор include

Вместо фреймове, може да се използват включени с include файлове. Синтасисът е

```
include('myfile.php');
```

ако сте оставили файла myfile.php в същата папка.

Отделяне на постоянните и променливи части

Нека си представим, че на всички страници от сайта, отгоре се повтаря например банерът headbanner.gif, а отдолу, по същият начин - банерът footbanner.gif. В средната част на страницата съдържанието е променливо. Да „разрежем” мислено кода на три части и да ги запишем в три различни файла:
Горна, постоянна част - headpage.php
Средна, променлива част – обикновен код на html

Долна, постоянна част - footpage.php

Така страницата page.php ще съдържа:

```
<?php
include('headpage.php');
?>
<h1>Променлив текст<h1>
<?php
include('footpage.php');
?>
```

Всяка друга старница може да бъде получена от тази със замяна на променливия текст.

Горна постоянна част - headpage.php

В този вариант се визуализира горния банер и се задава кодирането за българския текст:

```
<html>
<head>
<title>Web Site Title</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
</head>
<body>
<img src="headbanner.gif">
```

Долна постоянна част - footpage.php

Тя е още по-проста, защото показва само банера отдолу.

```
<img src="footbanner.gif">
</div>
</body>
</html>
```

Ами къде е менюто?

Нека да показваме например три страници - page1.php, page2.php и page3.php.

Да напишем menu.php

```
<?php
echo "<a href=\"page1.php\">page 1</a>  ";
```

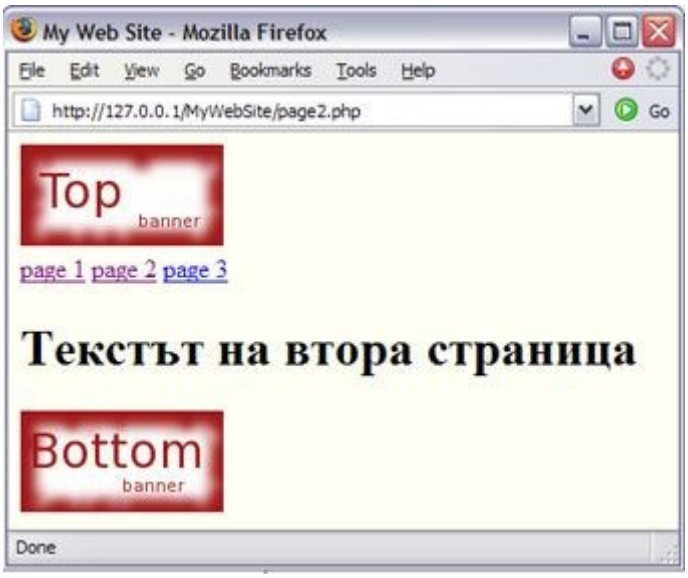
```
echo "<a href=\"page2.php\">page 2</a>  ";
echo "<a href=\"page3.php\">page 3</a>  ";
?>
```

Обратната наколнена черта тук \"page1.php\" е за да се визуализират кавичките.
И сега може би се досещате, чу ще го добавим към нашата горна постоянна част. Допишете в headpage.php следния код:

```
<?php
include('menu.php');
?>
```

Какво сглобихме до тук?

Ето това:



Не е много красиво, но работи!
Дали отгоре ще има банер или цяла таблица с текстове, банери или линкове, просто е без значение.

А защо няма index ?

1. Сега ще го сложим като използваме page.php и поправим текста.
2. После ще добавим в менюто ред

```
echo "<a href=\"index.php\">home</a>  ";
```

Току що демострирахме update на сайта, просто генерираме страницата и я добавяме в менюто.

CSS

Добрият стил на работа и промените на стандартите, отразени в сайта на консорциума www.w3.org, изискват да използвате именно стилови формати, а не определяне на цвета и големината на обектите в сайта на парче. Имайте предвид това, когато ползвате генератор на html код. За предпочитане е да използваме външен css файл, връзката към който ще поставим в "главата" на нашата горна постоянна част - headpage.php.

```
<link rel="stylesheet" type="text/css"  href="mystyle.css">
```

По този повод нека да кажем малко и за програмите, генериращи CSS код. Почти няма free или open source средства, които да дават наистина ефективен и красиво изглеждащ код. Или поне не би могло да се мине без стабилни познания по CSS и без "дописване" на кода. Човек трябва да бъде внимателен и критичен, даже и когато използва известни и "платени" програми за дизайн и по-специално за форматиране. Браузерите са толкова различни във възможностите си да интерпретират правилно код, написан според изискванията на стандартите.
Четейки задълбочено документите на W3, може да попаднете на Web Standards Group - http://webstandardsgroup.org/ Това е едно сдружение от дизайнери и разработчици, които се интересуват от web стандартите за HTML, XHTML, XML, CSS, XSLT и от добрита практики, свързани с тях. От там може да намерите чудесен туториал за използването на списъци - http://css.maxdesign.com.au/
Там именно е показано превръщането на списъка в хоризонтално меню с rollover ефекти. Ако разгледате текстовете подробно, ще попаднете на много ефектни и приложими за всички браузери примери. Свойството

```
#navlist li

{

display: inline;}

дава възможност да се списъкът да се появи в един ред. Липсата на номер
или bullet се осигурява с:

list-style-type: none;
```

Така стиловия формат, след корекцията в цветовете ще придобие следния вид:

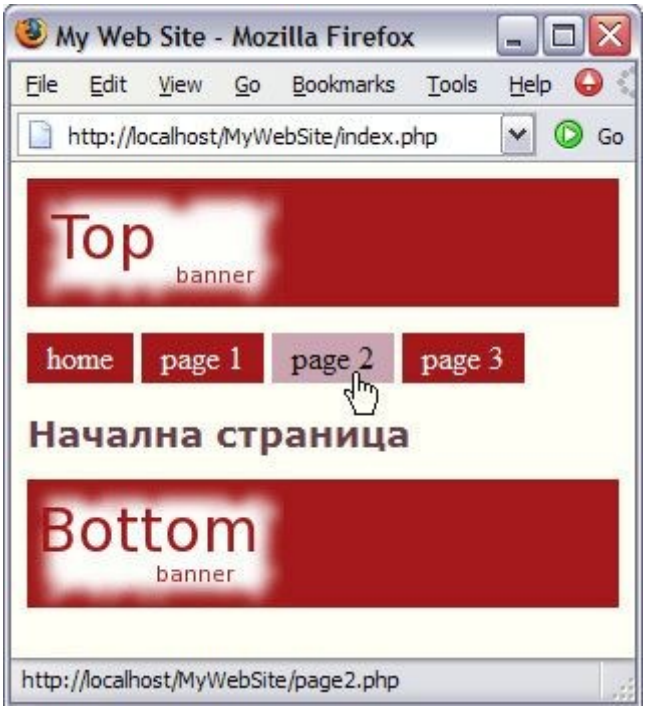
```
ul#navlist

{

margin-left: 0;

padding-left: 0;

white-space: nowrap;

}

#navlist li
```

```
{
display: inline;
list-style-type: none;
}
#navlist a { padding: 3px 10px; }
#navlist a:link, #navlist a:visited
{
color: #fff;
background-color: #A4181B;
text-decoration: none;
}
#navlist a:hover
{
color: #000000;
background-color: #CBA4B2;
text-decoration: none;
}
```

А менюто ще има вида:

```
<?php
echo "<div id=\"navcontainer\"> <ul id=\"navlist\">";
echo "<li> <a href=\"index.php\">home</a> </li> ";
echo "<li><a href=\"page1.php\">page 1</a>  </li>";
echo "<li><a href=\"page2.php\">page 2</a>  </li>";
echo "<li><a href=\"page3.php\">page 3</a>  </li>";
echo "</ul></div>";
?>
```

Ето и резултатът, който вече е по-приличен.



Използването на карта, Map, също е доста добро решение.
То обаче има недостатъка, че при промяна на броя позиции в менюто, трябва да се рисува отново, а и кодът трябва да се напише за новата картинка.

Ако искаме вертикално меню?

Пак на http://css.maxdesign.com.au/selectutorial/ ще намерите чудесен пример за изграждане на сайт без използване на таблици, като текстът стои в три колони и има вертикално разположено меню в ляво. Авторът тук е майстор, който внимателно е заобиколил всички опасни места по пътя. Примерите изглеждат перфектно и през Explorer, и през Mozilla.
Това е красиво, но е сложно. През през последните месеци често попадах на красиво изработени сайтове, които гледани през Mozilla обаче, имаха доста припокрити и нечитаеми текстове, така че на човек му се свива сърцето, като си помисли за труда на дизайнерите. Пък и изобщо няма гаранция, че вашият потенциален клиент няма да разгледа “чудесиите” с които мислите да го спечелите, с IE 4.0 например 
Така че, след въздишка на съжаление, си казваме:
По-добре таблици
В лявата колона - менюто, в дясната - съдържанието на конкретната страница. И от горе, и отдолу - все таблици. Защото, когато задавате свойствата от групата box и искате ширина width: 640 например, Mozilla не приема това.
Ще трябва да преместим менюто в средната променливата част.
Ето едно решение за вертикално меню от същото място. Отново срещаме:

```
list-style-type: none;
```

Заедно с това

```
display: block;
```

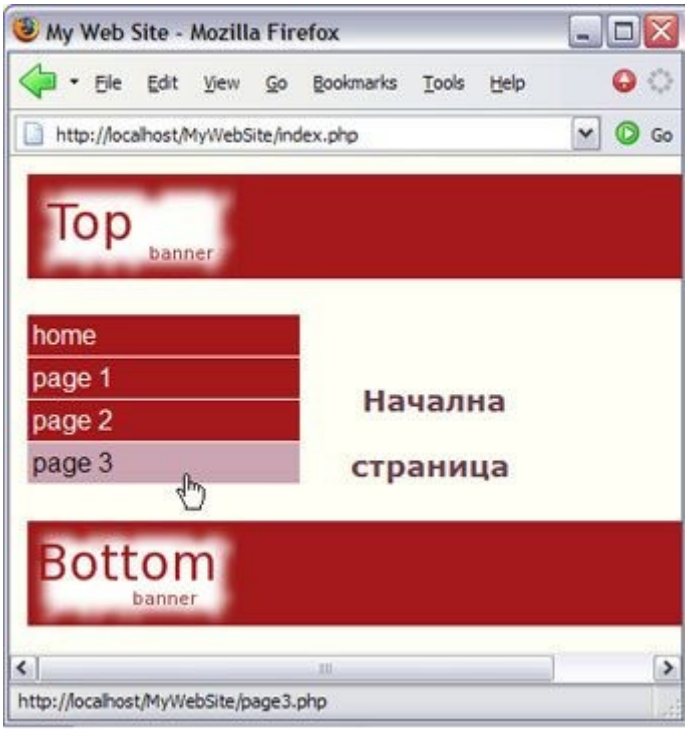
и за да се отделят визуално отделните позиции, една рамка отдолу:

```
border-bottom: 1px solid #ffffff;
```

Така менюто няма да се промени, а форматирането му ще бъде:

```
#navcontainer ul
{
margin-left: 0;
padding-left: 0;
list-style-type: none;
font-family: Arial, Helvetica, sans-serif;
}
#navcontainer a
{
display: block;
padding: 3px;
background-color: #A4181B;
border-bottom: 1px solid #ffffff;
}
#navcontainer a:link, #navlist a:visited
{
color: #ffffff;
text-decoration: none;
}
#navcontainer a:hover
{
background-color: #CBA4B2;
color: #000000;
}
```

Ето и резултатът:



Малко "педантизъм"

Работата не свършва с реализацията на работещ сайт. Ако оставите нещата до тук без да си запишете кое къде е и как се променя, ще има да си блъскате главата само след месец при първия update. Така че, даже и сам да сте си началник, седнете и си запишете:

1. Нова страница се създава като .....
2. Позиция в менюто се добавя като ...
3. Стиловия формат се вика от еди къде си...
4. ...и т.н.

Може да си направите папки и за отделните части, за да е по ясно кои картинки и текстове от къде се вземат. Тези усилия си струват, но това се разбира по-късно.

Require или Include

Require също вмъква файлове в документа, но съобщението, което се появява ако необходимият файл не бъде намерен е "фатална грешка" и абсолютно нищо друго не се вижда.

Require()

Ще ви покаже:

Fatal error: main(): Failed opening required 'headpage.php' (include_path='.;c:\php4\pear') in d:\program files\apache group\apache\htdocs\mywebsite\index.php on line 2

...и нищо от страницата.

Include()

Ако се използва Include обаче, намерената част все пак се визуализира, а съобщението е "предупреждение"

Warning: main(headpage.php): failed to open stream: No such file or directory in...

Но останалата част от страницата ще се види.

Изводите

1. Не “ровичкане” в кода на парче, а стройни правила за форматиране и промяна. Изглежда малко “сковаващо”, но полученото решение е наистина стилно.
2. От старата страница получавате нова, само като замените текста. Бихте могли да направите база данни за страниците. Ако “база данни” звучи сложно, мислете си за таблица, в която има колони за име на страницата, заглавие, под-заглавие, текст, изображение. Всичко това може да се систематизира, ако използвате възможностите на MySQL, които и без това се предлагат към хостинга.
3. Решите ли да използвате базата данни, ще “усвоите” и похвати, свързани с използването на масиви.
4. Следваща крачка от тук са шаблоните, но те изискват използване на класове, което може да ви затрудни, ако сега започвате заниманията си по програмиране.

Ето как идва идеята за така наречения Content Management System (CMS). Но всички разбираме, че е абсурдно да използваме една такава free система, без изобщо да сме наясно какво всъщност се прави. Е, надявам се че изложеното по-горе би ви улеснило.

Статията е публикувана в Download.BG, бр. 05/2005.