Прост шаблон за сайт с PHP

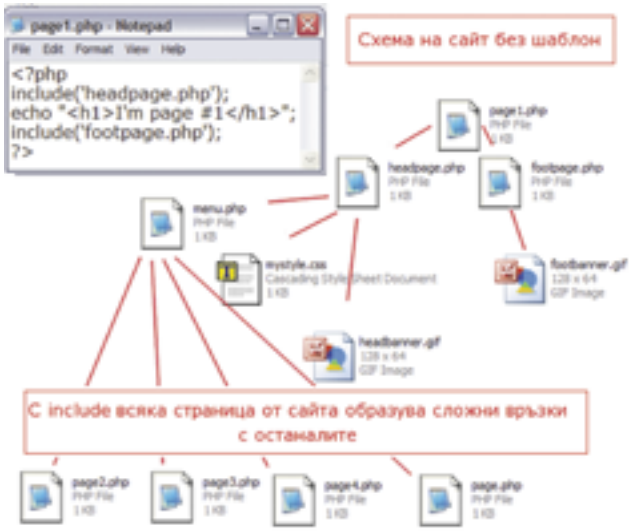
Евгения Христова (София)
info@shevitza.com

В брой 5 от 2005 разгледахме как от статичен сайт с фреймове и прост HTML код можем да преминем към използване на PHP. Ще припомним, че PHP е скриптов език, който се използва свободно и в момента се предлага към повечето хостинг пакети. Хубаво би било човек да има време за подробен курс по PHP, особено ако е от по-старата генерация и обектно-ориентираното програмиране му е непознато. За онези, които нямат време за подобни неща и се учат "в движение", ще покажем как може да се създават сайтове с шаблони, които използват така наречените обекти, описвани с класове.

Всичко, изложено по-долу, се основава на публикации в Интернет и може да бъде намерено на цитираните адреси свободно, напълно в духа на любимото ни списание DOWNLOAD.BG!

Да припомним накратко какво направихме в брой 5:

- Създадохме сайт, при който всяка страница се генерираше от три включени php файла съответно за горната, средната и долната част.
- Променяхме средната част и получавахме съдържанието на новата страница.
- Променяхме горната част, за да добавим позиции в менюто.



Сега сме просто на една крачка от така наречените шаблони.

Да подредим къщата

Досега в папката на сайта имахме изображения, css и php файлове. Работата с шаблони ще изисква да работим с още няколко типа файлове и е добре те да не бъдат в "насипно състояние" в основната папка на сайта. Ще направим папки за:

1. изображенията – img;
2. стиловите формати – css;
3. данните – dat;
4. библиотека за шаблоните lib.

Да, като истинска къща – имаме колекция с картини в хола (img), шкаф с "вкусотии" в кухнята - (css), гардероб (dat) и библиотека (lib) за новите интелигентни обитатели на нашият сайт – шаблоните. Но за да няма писъци от рода на "Къде е кафето?" или "Къде са ми чорапите?", ще трябва не само например да окачим картините в хола, но и да пренасочим пътищата до тях. Към банера вече ще се обръщаме с

```
<img src="img/headbanner.gif">
```

А до шкафа с "подправките" ще отиваме така:

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

Шаблонът е свързан с класове или обекти и сега ни предстои да се докоснем до така наречената обектно ориентирано програмиране (ООП). Която и книга за ООП да отворите, например от поредицата на Брус Екел "Да мислим на.." Java, C++ или в самия Help на PHP – "manual_php", като стане дума за класове, започват да ви редят кулинарни, кухненски или киноложки примери, които на пръв поглед не подхождат на представата за компютърната програма като последователност от команди. Усмихнете се, "настанете се удобно и не затягайте коланите", настройте се наистина за абстрактно мислене.

Как изглежда шаблонът?

Като обикновена интернет страница или файл с разширение html. В тялото му има няколко думи, заградени с фигурни скоби и точно това са местата, в които ще се намести променливото съдържание. Ако искате, слагате си и таблица с невидими рамки, за да пишете свободно във всички части на екрана.

В табл. 1 можете да сравните досегашното с търсеното ново решение със шаблон.

Да създадем клас

Примери за класове-шаблони има изключително много. Но прости и разбираеми – твърде малко. Следващият пример е на Тимоти Броншчик, намира се на адрес: **http://codewalkers.com/tutorials/58/8.html** и е забележителен с това, че е само 35 реда. Нищо по-подходящо от него, за да можем да разберем как работи класът на шаблона.

Класът ще се казва Page, ще има една променлива (наречена още поле или свойство) $page, конструктор Page($template) и други две функции - replace_tags($tags) за замяна на таговете ии function output() за извеждане.

```
<?php
class Page
{
var $page;
function Page($template)
{
}
function replace_tags($tags)
{
}
function output()
{
}
}
?>
```

Без клас	Със дефиниране на клас template.html
<html> <head> <title>From Frame to PHP</title> <link rel="stylesheet" type="text/css" href="css/mystyle.css"> </head> <body> <img src="img/headbanner.gif"> <?php include('menu.php'); ?>	<html> <head> <title>{TitleBarText}</title> <link rel="stylesheet" type="text/css" href="css/mystyle.css"> </head> <body> <img src="img/headbanner.gif"> <ul id="navlist">"; home page 2 page 3 {PageTitle} {PageText} <img src="img/footbanner.gif"> </body> </html>
<?php echo "<div id=\"navcontainer\"> <ul id=\"navlist\">"; echo " home "; echo " page 2 "; echo " page 3 "; echo " "; ?>	
echo " I'm page #3 ";	
<?php include('footpage.php'); ?>	

табл.1

Код на класа - lib/template.php

```
<?php
class Page
{
  var $page;
//Началото на конструктора
  function Page($template = "template.html") {
    if (file_exists($template))
      $this->page = join("", file($template));
    else
      die("Template file $template not found.");
  }
//Тук се реализира динамичното съдържание
  function parse($file) {
    ob_start();
    include($file);
    $buffer = ob_get_contents();
    ob_end_clean();
    return $buffer;
  }
//Тук се заменят таговете
  function replace_tags($tags = array()) {
    if (sizeof($tags) > 0)
      foreach ($tags as $tag => $data) {
        $data = (file_exists($data)) ? $this->parse($data) : $data;
        $this->page = eregi_replace("{" . $tag . "}", $data,
                      $this->page);
        }
    else
      die("No tags designated for replacement.");
  }
// Тук е визуализацията
  function output() {
    echo $this->page;
  }
}
?>
```

Как работи конструкторът

Ако файлът съществува "се прави" страницата, а ако не – се показва съобщение за грешка. Той използва функциите:

- file_exists – връща истина, ако файлът съществува;
- file връща прочетения файл в масив
- join присъединява елементите на прочетения масив с ""
- die съобщава за евентуална грешка

"Разбор" на динамичното съдържание parse

Използвани функции:

- ob_start – включва външното буфериране и после вмъква файла;
- ob_get_contents – връща съдържанието на изходния буфер;
- ob_end_clean – изключва буферирането след което функцията връща буфер;

Замяна на таговете използва parse

- Sizeof връща броя на таговете, които са записани в масив
- foreach чете елементите на масива и ги събира в буфер с функцията parse
- eregi_replace премахва фигурните скоби и праща всичко в страницата, която се показва с функцията output() и echo

Първата страница по шаблона

Следва първата страница на шаблона за нашия пример е отново page.php. Тя използва шаблона html, класа page и функцията за замяна на таговете.

```
<?php
require_once("lib/template.php");
$page = new Page("template.html");
$page->replace_tags(array(
  "TitleBarText" => "dat/tb.dat",
  "PageTitle" => "dat/tt.dat",
  "PageText" => "dat/p.dat"
));
$page->output();
?>
```

Използвана е функцията require_once, вмъква файл, който за разлика от require или include е по-особен, в случая това е шаблонът.

Сглобяване

1. С прост тектов редактор като Notepad или подобен създайте файла template.php като в точката "Код на класа" и запишете го в папката lib.

2. Създайте файл template.html, като в тялото му запишете например:

```
<body>
<div class="head">
<img src="img/headbanner.gif">
</div>
<div id="navcontainer">
 <ul id="navlist">
<li> <a href="index.php">home</a>
</li>
<li><a href="page2.php">page 2</a>
</li>
<li><a href="page3.php">page 3</a>
</li>
</ul>
</div>

<h1>{PageTitle}</h1>
<p>{PageText}</p>
<div class="foot">
<img src="img/footbanner.gif">
</div>
</body>
```

3. Създайте page.php като първа страница на шаблона.

4. Създайте в папката dat, файловете с данни за показване

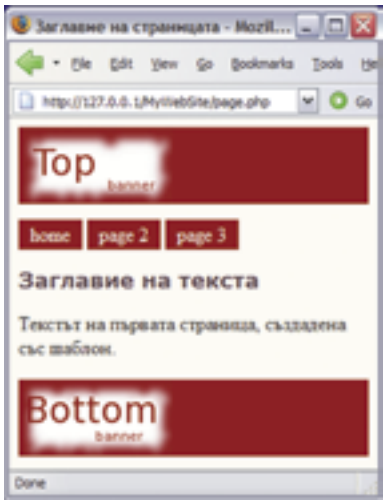
a. tb.dat – съдържащ текста на заглавието на прозореца на page.php;

b. tt.dat – съдържащ заглавието на показвания през page.php текст;

c. p.dat – съдържащ самото текстово съдържание на page.php

5. Надявам се, че папката на вашия малък сайт е разположена като подпапка на www, така че php файловете да могат да се стартират. Ще припомним, че PHP можете да ползвате на домашния си компютър, ако си го свалите или инсталирате или ако използвате EasyPHP или WAMP, (**http://www.wampserver.com**/) или някоя друга подобна програма, имитираща средата, в която работят файловете ви при закупен хостинг.

6. Отворете браузер, напишете в адресна лента **http://127.0.0.1/** . Отворете папката на сайта и стартирайте page.php. Трябва да видите това:



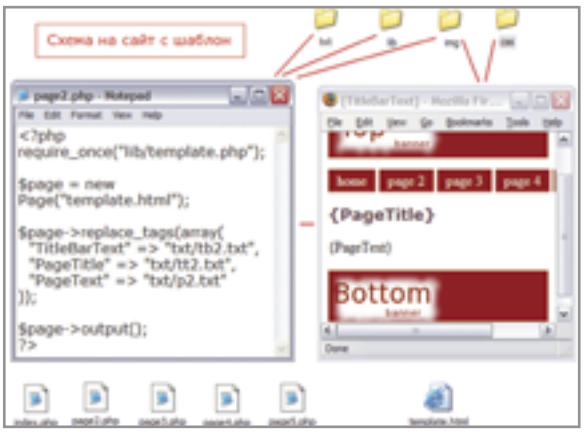
7. Следва създаването на index.php и останалите страници на сайта.

Резултатът

Всичко си работи нормално, добавянето на нови страници е лесно и става почти само с Copy - Paste. Ние сме горди с пускането на почти професионален шаблон. Със задоволство може да отбележите, че ако добавите html код в p.dat файловете, той се визуализира нормално. Ако сложим Java Script? Ура, Java Script работи!

Има някои неудобства с отварянето на dat през бележника. Експериментът показва, че txt файлове вършат същата работа.

Следва схема на сайта, в която dat вече е txt.



Главната папка на сайта има само индекс и странци, там всичко е чисто и подредено. Но когато отворим папката – гардероб dat или сегашната txt, изведнъж се "изсипват" купища вещи. За всяко едно поле, на всяка страница, по един файл. Е, за това си има бази данни, да не забравяме, че разполагаме с MySQL, което трябва да е нашата следваща планирана стъпка.

Такъв един експеримент с шаблон може и да не завърши със създаването на собствен сайт с PHP. Той обаче е полезен за всички, които искат да използват Content Management Sysyem [CMS] – системи за управление на съдържанието. Както е известно, има много CMS за най-различни случаи като блогове, форуми, галерии, екипна работа, обучение, които се основават на използването на PHP, MySQL и на подобни шаблони и се разпространяват под GPL лиценз. Познанието за това гкак са направени" ще ни помогне да се възползваме от тези истински съкровища в мрежата, които само чакат да бъдат открити.