



Web Templates em PHP

Como criar templates para páginas Web

IN MEDIO VIRTUS

Autor: António Rodrigues
Data: 20 Abril 2010

Este documento está protegido pelas convenções e legislação sobre direitos de autor e de cópia. O autor não autoriza a cópia total ou parcial do documento ou de qualquer conteúdo textual ou gráfico sem prévia autorização. O seu uso está limitado pelas condições abaixo discriminadas.

O presente guia pode ser utilizado como referência no desenvolvimento de páginas Web, quer de carácter privado, quer de carácter comercial. Mas não pode ser copiado, distribuído, comercializado ou utilizado na criação de outros guias.

O autor autoriza a utilização de referências e de excertos em trabalhos de investigação académica. Não estão contudo abrangidos trabalhos de investigação com fins comerciais ou sobre os quais incidam contrapartidas monetárias.

Qualquer referência a este documento deve incluir o nome do seu autor, assim como a fonte de onde foi acedido.

Este documento foi escrito ao abrigo do novo acordo ortográfico.

Introdução

O presente guia tem como finalidade ajudar a desenvolver páginas Web baseadas em *Templates* ou Modelos com recurso à linguagem PHP. A longo deste documento utiliza-se o termo *Template* em detrimento do termo Modelo por se tratar de uma expressão mais comum no âmbito da engenharia de software.

O que é um *template*? Trata-se de um modelo que define uma estrutura comum a determinado conjunto de objetos.

O termo *Web Template* aplica-se a dois tipos de modelo distintos. O primeiro, o modelo de apresentação gráfica/visual também conhecido como *Web Design Template*, ou seja, o conjunto de componentes gráficos, cores, fontes de texto, etc. O segundo, o modelo da estrutura gráfica que consiste num conjunto de secções ou áreas com determinado conteúdo. É portanto necessário fazer a desambiguação. Este guia aborda o segundo caso.

A utilização de *templates* apresenta diversas vantagens, entre as quais:

- Consistência tanto na apresentação visual como na codificação de cada uma das páginas.
- Reutilização do modelo sem necessidade de duplicação de código.
- Maior organização da estrutura de código.
- Permite reduzir o tempo de desenvolvimento de um sítio Web devido à reutilização de módulos e à organização da estrutura de código.
- Facilita as alterações das páginas devido ao facto de se basear na reutilização de módulos.

O *template* por si só não é suficiente para tirar partido de todas estas vantagens. Aliado ao *template* deve existir uma codificação bem organizada e modular de forma a que cada fragmento de código funcione de forma independente. Só desta forma é possível uma verdadeira reutilização e escalabilidade. Mas a definição de padrões de código está fora do âmbito deste guia.

Nos próximos capítulos são apresentados dois métodos de criação de *templates* fazendo uso da linguagem PHP.

O Modelo

Neste capítulo é explicado como desenvolver um *template* através da apresentação de exemplos práticos. Consequentemente não são abordados conceitos teóricos e avança-se de imediato para os primeiros passos da sua implementação.

Nos exemplos que se seguem utiliza-se a estrutura mais comum de uma página Web representada pela figura seguinte. Tradicionalmente uma página Web é composta por um cabeçalho, um rodapé, um menu e a área principal onde são exibidos os conteúdos. Nesta simples estrutura o menu fica localizado do lado esquerdo do conteúdo principal. O cabeçalho fica localizado no topo da página e o rodapé no fundo.

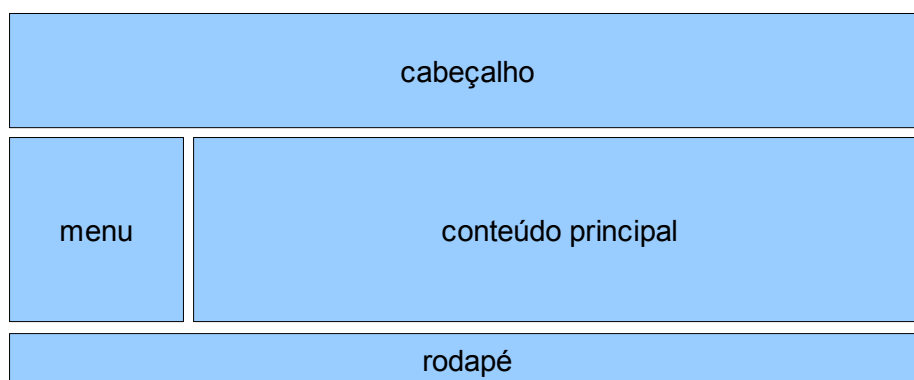


Figura 1: Esboço da estrutura da página

Numa codificação feita à medida esta estrutura seria replicada em todas as páginas que compõem o sítio Web. Como resultado haveria uma duplicação de código. A alteração da estrutura resultaria na necessidade de alterar todas as páginas do sítio Web.

A utilização de um *template* vem facilitar este trabalho pois permite definir a estrutura gráfica num único ficheiro. Este ficheiro será então reutilizado pelas páginas do sítio Web. Em caso de alteração da estrutura apenas o ficheiro *template* é afetado sendo as alterações imediatamente visíveis em todas as páginas.

No exemplo definido na figura, o modelo é implementado num ficheiro com o nome 'template.php' com recurso a HTML padrão. Tipicamente este modelo é construído com elementos DIV ou alternativamente com o elemento TABLE.

O uso de tabelas é frequentemente desaconselhado para definição da estrutura gráfica em detrimento do uso do elemento DIV. Contudo as tabelas são mais flexíveis e permitem resultados que apenas são conseguidos em DIV com recurso a alternativas por vezes complexas em CSS e/ou Javascript.

A escolha fica ao critério do programador. A título de exemplo apresentam-se as duas abordagens. De forma a simplificar os exemplos apresenta-se apenas o código relativo ao corpo do ficheiro HTML omitindo metadados.

Ficheiro **template.php** com recurso a DIV.

```
<body>
  <div id="container">
    <div id="header"> </div>
    <div id="menu"> </div>
    <div id="main"> </div>
    <div id="footer"> </div>
  </div>
</body>
```

O código CSS que define a posição e dimensão dos elementos DIV é o seguinte.

```
div#container {
  width: 800px;
}
div#header {
  height: 100px;
}
div#menu {
  width: 200px;
  height: 450px;
  float: left;
  clear: left;
}
div#main {
  width: 600px;
  height: 450px;
  float: right;
  clear: right;
}
div#footer {
  height: 50px;
  clear: both;
}
```

Ficheiro **template.php** com recurso a TABLE.

```
<body>
  <div id="container">
    <div id="header"> </div>
    <table>
      <tr>
        <td class="menu"> </td>
        <td class="main"> </td>
      </tr>
    </table>
    <div id="footer"> </div>
  </div>
</body>
```

O código CSS é agora mais simples.

```
div#container {  
    width: 800px;  
}  
div#header {  
    height: 100px;  
}  
div#footer {  
    height: 50px;  
}  
td.menu {  
    width: 200px;  
    height: 450px;  
}  
td.main {  
    width: 600px;  
    height: 450px;  
}
```

Estratégias de *Templating*

Neste capítulo apresentam-se duas estratégias de *templating*. Ambas possuem vantagens e desvantagens e são diametralmente opostas entre si. Fica ao critério do programador selecionar a que melhor se adequa às necessidades do sítio Web que está a desenvolver.

Em PHP, e de forma geral em quase todas as linguagens de programação para a Web, os *templates* são construídos com base na inclusão de ficheiros que compõem pequenos módulos ou partes de uma página Web. Em PHP utilizam-se as funções '*include*' e '*require*' para esse efeito.

Existem dois métodos de incorporar um *template* nas páginas Web. O primeiro consiste na inclusão do *template* nas páginas Web. O segundo consiste na inclusão das páginas Web no *template*. No âmbito deste guia os métodos são denominados **N-1** e **1-N** respetivamente. As figuras seguintes representam as duas estratégias.

Estratégia N-1: n páginas Web incluem 1 ficheiro *template*.

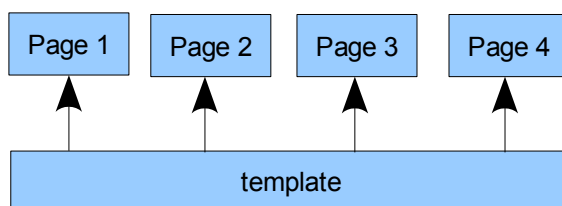


Figura 2: Estratégia de *templating* N-1

Estratégia 1-N: 1 ficheiro *template* inclui n páginas Web.

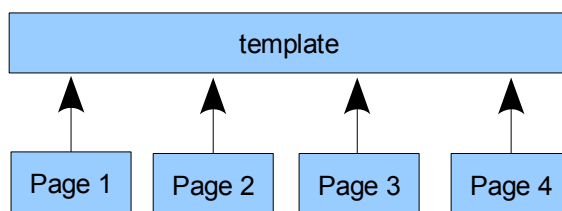


Figura 3: Estratégia de *templating* 1-N

Método N-1

No método N-1 cada página de um sítio Web é implementada através de ficheiros independentes, tal como o seria numa codificação à medida. No entanto a estrutura da página é definida pelo ficheiro *template* que é incluído por cada uma das páginas evitando a duplicação de código.

A principal desvantagem deste método é a necessidade de manter um ficheiro por cada página. Acresce ainda a desvantagem de se utilizarem li-

gações estáticas para cada um dos ficheiros dentro do sítio Web, nomeadamente nos menus.

As vantagens deste método são uma clara separação entre páginas e a possibilidade de disponibilizar ligações diretas a cada uma das páginas. Esta última vantagem é interessante quando se pretende que determinada página seja indexada pelos motores de busca em vez de indexar apenas o sítio Web. A título de exemplo, um sítio Web de uma empresa poderá ter interesse em indexar nos motores de busca a página onde são publicadas as ofertas de emprego.

Para exemplificar a implementação deste método considera-se um sítio Web com 4 páginas: a página principal (main.php), uma página de notícias (news.php), uma página com a descrição dos serviços disponíveis (services.php) e uma página com contactos (contacts.php).

Cada uma das páginas inclui o ficheiro *template*. Mas o ficheiro *template* contém apenas a estrutura da página e não qualquer conteúdo. Portanto há que definir igualmente o conteúdo a disponibilizar na página. O conteúdo pode ser definido através de variáveis PHP ou através de ficheiros que são incluídos pelo *template*. Neste exemplo utiliza-se a última opção por ser a mais simples. O conteúdo de uma página pode ser dinâmico ou estático, de acordo com as necessidades do sítio Web, mas a sua implementação está fora do âmbito deste guia. Ambos os métodos, variáveis ou ficheiros, suportam conteúdos dinâmicos e conteúdos estáticos.

O código seguinte demonstra como definir o conteúdo por inclusão de ficheiros na página de notícias 'news.php'. As restantes páginas serão semelhantes variando apenas o nome dos ficheiros.

Ficheiro news.php.

```
<?php
$fileMenu = 'inc/news_menu.php';
$fileMain = 'inc/news_main.php';

$include('template.php');
?>
```

No ficheiro 'news.php' é indicado que o menu está definido no ficheiro 'news_menu.php' e o conteúdo principal está definido no ficheiro 'news_main/php'. Mas estes ficheiros não foram ainda incluídos. Os mesmos serão incluídos no *template* que passa a ter as seguintes modificações (a vermelho).

Considera-se ainda que o cabeçalho da página é definido no ficheiro 'header.php' e o rodapé no ficheiro 'footer.php'. Estes dois ficheiros serão sempre incluídos em todas as páginas através do *template*, não sendo necessário defini-los no ficheiro de cada página.

Ficheiro template.php.

```

<body>
  <div id="container">
    <div id="header"><?php include('inc/header.php'); ?></div>
    <div id="menu"><?php include($fileMenu); ?></div>
    <div id="main"><?php include($fileMain); ?></div>
    <div id="footer"><?php include('inc/footer.php'); ?></div>
  </div>
</body>

```

Assim ao aceder ao endereço <http://mycompany.com/news.php> é disponibilizado o conteúdo definido no ficheiro 'news_main.php' e exibido o menu definido no ficheiro 'news_menu.php'.

Nota: como o cabeçalho e o rodapé são comuns a todas as páginas poderiam ser definidos no ficheiro 'template.php'. Mas a sua separação por ficheiros torna o código muito mais organizado e legível.

Método 1-N

No método 1-N a inclusão dos ficheiros é oposta. Existe um ficheiro principal com o *template* que inclui os ficheiros de página.

A vantagem deste método é o de concentrar num único ficheiro todos os acessos reduzindo substancialmente o número total de ficheiros. Esta vantagem é de grande importância quando, por razões de segurança, se pretende esconder o acesso direto a determinadas páginas, como por exemplo formulários que de outra forma poderiam ser diretamente acedidos por processos automáticos.

A desvantagem de concentrar todos os acessos num único ficheiro é a impossibilidade de indexar as páginas separadamente nos motores de busca. Como todas as páginas são acedidas pelo ficheiro principal apenas este ficheiro pode ser indexado nos motores de busca.

O ficheiro principal é o *template* propriamente dito, mas por se tratar do único acesso ao sítio Web será denominado 'index.php'.

Para definir a página que se quer visualizar é agora necessário passar um identificador. Neste exemplo vamos utilizar a forma mais simples através da identificação por nome. Por uma questão de consistência utilizam-se os mesmos nomes que no exemplo N-1. Poderia ser utilizado um identificador estático numérico ou alfanumérico, ou um identificador gerado dinamicamente. A escolha depende mais uma vez das necessidades de cada sítio Web.

O identificador é passado num parâmetro "pid", acrónimo de *Page ID*. O *template* anteriormente utilizado vai ser modificado (a vermelho) para detetar o parâmetro. No caso de não ser passado nenhum parâmetro, ou este ser inválido, será exibida a página principal.

O conteúdo de cada uma das páginas é definido da mesma forma que no exemplo N-1. Os menus são definidos nos ficheiros 'news_menu.php',

'main_menu.php', etc. O conteúdo principal nos ficheiros 'news_main.php', 'main_main.php', etc.

Ficheiro index.php.

```
<?php
    $validPages = new array('main', 'news', 'services', 'contacts');

    $pid = 'main';

    if ((array_key_exists('pid', $_GET)
        && (in_array($_GET['pid'], $validPages))) {
        $pid = $_GET['pid'];
    }

    $fileMenu = "inc/$pid_menu.php";
    $fileMain = "inc/$pid_main.php";
?>
<body>
    <div id="container">
        <div id="header"><?php include('inc/header.php'); ?></div>
        <div id="menu"><?php include($fileMenu); ?></div>
        <div id="main"><?php include($fileMain); ?></div>
        <div id="footer"><?php include('inc/footer.php'); ?></div>
    </div>
</body>
```

No corpo PHP são definidos os parâmetros que são válidos. O parâmetro por omissão é 'main' que se refere à página principal. Na expressão *if* é verificado se existe o parâmetro 'pid' e se este é um parâmetro válido. Os nome dos ficheiros a incluir é então definido com base no parâmetro que foi passado.

Ao aceder ao endereço <http://mycompany.com/index.php?pid=news> será exibida a página de notícias. Caso nenhum parâmetro seja passado ou o valor passado seja inválido é exibida a página principal.

Conclusão

Como se pode ver pelo código apresentado nos exemplos a implementação de *templates* é bastante simples. A organização do código através de ficheiros separados que são incluídos no *template* torna a manutenção do sítio Web mais fácil e rápida.

Os exemplos são apenas um ponto de partida, pois um sítio Web é normalmente mais complexo do que o que foi apresentado. Mas são exemplos práticos que podem ser reutilizados de imediato e enriquecidos gradualmente à medida que o programador avança no seu trabalho.