

# Suindara

Guia para desenvolvimento dos  
templates



## Criando um template

Para criar um template é necessário estar logado no sistema na área da Administração. Selecione "Templates" no menu lateral esquerdo e clique no botão "Gerar novo template". Caso a operação seja executada com sucesso uma mensagem aparecerá no topo informando o ocorrido e identificando a localização do template no disco. Na listagem de templates será adicionado um novo template com o nome "Template\_x" onde x é o identificador do template. Por padrão todos os templates são criados no diretório

***<Local de instalação do CMS3>/app/webroot/templates/***

e a pasta do template em desenvolvimento terá o mesmo nome do template ("Template\_x").

## Estrutura

O template é composto pela seguinte estrutura de pastas:

***\_init*** : Contêm o script init.sql utilizado para gerar tabelas e inserções no banco de dados na instalação do template.

***\_end*** : Contêm o script end.sql utilizado para apagar as tabelas do banco de dados na remoção do template.

***css*** : contêm os arquivos de css.

***js*** : Contêm os arquivos de script.

***images*** : Contêm as imagens utilizadas no template.

***noticias*** : Contêm os arquivos de visualização e listagem das notícias.

***paginas*** : Contêm o arquivo de visualização das páginas.

**Elements** : Arquivos parciais do template. Normalmente arquivos de visualização que podem ser compartilhados, como o cabeçalho e rodapé das páginas.

**Layouts** : Arquivos para alterar o layout de todas as páginas do template.

Elements e Layouts são recursos emprestados do CakePHP, para mais informações verificar a documentação online (<http://book.cakephp.org>).

## Resgatar dados no template

Os exemplos apresentados abaixo foram aplicados no Site Modelo que acompanha o CMS 3, seu código pode ser encontrado em:

**<Local de instalação do CMS3>/app/webroot/templates/default**

É importante destacar que todos os arquivos de visualização utilizados possuem a extensão ".ctp".

Para resgatar os dados são utilizados subclasses dos *ViewHelpers* (<http://book.cakephp.org/2.0/en/views/helpers.html>). Cada *ViewHelper* é responsável por tratar e retornar os dados de uma área do sistema (notícias, páginas, banners ...).

Nas próximas seções serão apresentados alguns trechos de código para resgatar os dados de cada área do sistema. Maiores informações podem ser encontradas no fonte do Site Modelo e na documentação que acompanha o Suindara CMS 3.

## Notícias

O trecho de código abaixo carrega uma notícia e exibe algumas informações dela:

```

1  <?php
2      $noticia_id = $this->CmsTemplate->getParams(0);
3      $noticia = $this->CmsNoticias->getNoticia($noticia_id);
4  ?>
5  <div>
6      <?php echo $noticia->htmlTitulo() ?>
7      <p>Publicado em
8          <?php echo $noticia->htmlDataPublicacao() ?>
9      </p>
10     <p> <?php echo $noticia->htmlTexto() ?> </p>
11 </div>

```

Na linha 2 o valor retornado pelo método *getParams* é o primeiro parâmetro recebido pela url que, neste caso, é o identificador da notícia. Na linha 3 é carregada a notícia, além do *getNoticias* existem outros métodos que podem simplificar a busca por notícias. É importante observar que, por padrão, somente as notícias que possuem o status de "publicada" serão retornadas pelo método.

Na linha 6, 8 e 10 é gerado o html para a apresentação das informações. Também é possível acessar o campo sem gerar o html correspondente, para isso basta chamar pelo nome do campo registrado no banco de dados. Por exemplo, para acessar o título da notícia sem gerar o html pode-se utilizar:

```

1      <p> <?php echo $noticia->titulo ?> </p>

```

Para acessar as imagens, vídeos e arquivos associados a notícia existem os métodos *getImagens*, *getVideos* e *getArquivos*. Por exemplo, para acessar as imagens da notícia carregada anteriormente:

```

1      <?php
2          $imgs = $noticia->getImagens();
3          if ($imgs) {
4              foreach ($imgs as $img) {
5                  echo $img->htmlImagem(TIMG_ORIGINAL,
                                          array('width' => '165',
                                                'height' => '120'));
6              }
7          }
8      ?>

```

Na linha 2 todas as imagens relacionadas a notícia são carregadas, e na linha 5 é gerado o html para mostrar cada imagem. Na função *htmlImagem* é possível informar se a imagem utilizada será a original ou uma versão menor. O acesso aos vídeos e arquivos é feito de forma similar. Para mais informações sobre os métodos utilizados e outros disponíveis verifique a documentação.

## Páginas

O acesso aos dados das páginas é feito de forma similar ao de notícias, segue um exemplo onde os dados de um página são carregados e exibidos.

```

1      <?php
2          $pagina_id = $this->CmsTemplate->getParams(0);
3          $pagina = $this->CmsPaginas->getPagina($pagina_id);
4
5          echo $pagina->htmlTitulo()
6          echo $pagina->htmlTexto()
7      ?>

```

Nas linhas 2 e 3 é carregado a página com o identificador informado pelo primeiro parâmetro passado pela url. Nas linhas 5 e 6 o título e o texto da página são gerados e exibidos. O método *getImagens* também pode ser utilizado em páginas para carregar as imagens relacionadas.

## Categorias

No trecho de código abaixo é carregada uma categoria:

```
1 $categoria = $this->CmsCategorias->getCategoria("Evento");
```

Neste caso a categoria carregada possui o identificador "Evento", sendo que este identificador é uma string registrada no momento em que a categoria é criada no CMS, outra possibilidade é utilizar o id da categoria para carregá-la.

A categoria pode ser utilizada para filtrar notícias que estejam vinculadas a ela. Por exemplo:

```
1 $noticias = $categoria->getNoticias();  
2 foreach ($noticias as $noticia) {  
3     echo '<p>' . $noticia->titulo() . '</p>';  
4 }
```

Na linha 1 todas as notícias vinculadas a categoria "Eventos" são carregadas, nas linhas 2 a 3 o título de cada notícia é mostrado.

## Menus

Para carregar um menu é necessário informar o seu identificador, de forma similar ao acesso as categorias mostrado acima.

```
1 $menu = $this->CmsMenu->getMenu('AcessibilidadeVirtual');
```

O identificador "AcessibilidadeVirtual" é definido quando o usuário criar o menu pelo CMS.

O menu é composto pela raiz e pelos sub-menus, podendo comportar vários níveis de sub-menu. Para mostrar o menu pode-se iterar sobre seus sub-menus e apresentar as informações necessárias:

```
1     foreach ($menu->getItens() as $submenu){  
2         echo "<a href={$submenu->getLink()}>  
           {$submenu->nome}  
           </a>";  
3     }
```

Em alguns casos pode ser necessário criar formas mais complexas para apresentar o menu, ou reutilizar a forma como o menu é desenhado em outros templates. Nesses casos a montagem do menu pode ser encapsulada em um objeto que implemente a interface *CmsMontadorMenu*. Essa interface possui o método `htmlMenu(CmsMenuelement $baseMenu)` que deve retornar uma string com o HTML gerado do menu. Segue o exemplo de um montador de menus utilizando com apenas um nível.

```

1  App::uses('CmsMontadorMenu', 'View/Helper/Util');
2  App::uses('CmsMenuElement', 'View/Helper/Din');
3
4  class MontadorMenuExemplo implements CmsMontadorMenu
5  {
6      public function htmlMenu(CmsMenuElement $baseMenu)
7      {
8          $result = "";
9          foreach ($baseMenu->getItens() as $submenu) {
10             $result .= "<a href={$submenu->getLink()}>
                        {$submenu->nome}
                        </a>";
11          }
12
13          return $result;
14      }
15  }

```

Para apresentar o menu utilizando o *MontadorMenuExemplo* basta passá-lo para o método *htmlMenu*:

```

1  $montador = new MontadorMenuExemplo();
2  echo $menu->htmlMenu($montador);

```

O exemplo de um montador de menus com mais níveis pode ser encontrado em:

**<Local de instalação do CMS3>/app/View/Helper/Util/CmsMontadorMenuPadrao.php**



## Banners

Para carregar um banner é necessário informar o seu identificador:

```
1    $banner = $this->CmsBanners->getBanner('Banner01');
```

Para apresentá-lo:

```
1    $banner->htmlBanner();
```

Como cada banner registrado no CMS pertence a um grupo, é possível filtrá-los:

```
1    $grupo = $this->CmsBanners->getGrupo('Principal');
2    foreach ($grupo->getBanners() as $banner) {
3        echo '<li>' . $banner->htmlBanner() . '</li>';
4    }
```

Neste caso, todos os banners registrados no grupo "Principal" serão mostrados como itens de uma lista.

## Páginas estáticas

Em alguns casos pode ser necessário a criação de uma página diferenciada que não apresenta nenhuma similaridade com os padrões já aplicados no projeto. Nesses casos pode-se criar páginas estáticas que possuem um layout próprio.

Para gerar uma página estática é necessário criar um arquivo .ctp no diretório do template, sua chamada pela url será feita através do caminho lógico

do arquivo, por exemplo se a página 'contato.ctp' for criada no diretório raiz do template seu endereço será:

***/contato***

Caso esteja dentro de uma pasta chamada 'geral':

***/geral/contato***

Exemplos de páginas estáticas podem ser encontrados no template do Site Modelo que acompanha o Suindara CMS.

## Gerar dados no banco

Conforme apresentado na seção Estrutura, o arquivo ***\_init/init.sql*** pode ser executado na instalação do template, gerando as tabelas necessárias para o template. Já o arquivo ***\_end/end.sql*** será executado na remoção do template, podendo remover qualquer dado utilizado pelo template.

## Empacotando o template

Para preparar o pacote a ser distribuído é importante descrevê-lo no arquivo info.json que deve acompanhar todos os templates. A estrutura do arquivo info.json é apresentada abaixo:

```
1  {  
2    "nome": "Nome do Template",  
3    "print": "Imagem.png",  
4    "autor": "Autor",  
5    "descricao": "Exemplo"  
6  }
```

O campo "print" será uma imagem pequena, utilizada para a preview do template na listagem de templates.

O arquivo deve ser colocado no diretório raiz do template. Para finalizar o pacote é necessário compactar os arquivos para um .zip. É importante salientar que os arquivos do template devem estar, obrigatoriamente, na raiz do arquivo .zip.