

## **MANUAL DE FORMAÇÃO E APOIO A NOVOS PROGRAMADORES**

**A plataforma SM5 é um CMS (Content Management System)  
desenvolvido com base em Zend Framework (versão < 2).**

**14-09-2012**

## Sumário

Plataforma SM5 .....	2
Introdução .....	2
Estrutura SM5 .....	3
Application .....	3
config.ini .....	4
Pasta Model .....	5
Setup.php .....	9
Admin.php .....	11
Component.php .....	13
GenericController.php .....	15
Controller.php .....	16
Pasta Form .....	16
Application_Admin .....	21
Pasta Form .....	21
Pasta views .....	22
GenericController.php .....	24
Controller.php .....	24
Módulos Backoffice: Setup, Painel, PainelGoweb, Dashboard .....	24
Módulo Setup .....	24
Módulo Painel .....	25
Módulo PainelGoweb .....	25
Módulo Dashboard .....	26
Views: exemplos de utilização, conceito de canal .....	29
Bases de dados: codificação, alterações à estrutura, site offline .....	33
Codificação (Charset) .....	33
Alterações à estrutura .....	33
Site offline para público (parâmetro 'ver') .....	33

## Plataforma SM5

### *Introdução*

A plataforma SM5 é um CMS (Content Management System) desenvolvido com base em Zend Framework (versão < 2). O acesso às funcionalidades disponibilizadas pela Zend Framework é feito essencialmente via classes wrapper Goweb\_

Respeitando o modelo MVC (model-view-controller) da plataforma Zend, SM5 tem um conjunto de características típicas de sistemas CMS:

- armazenamento de páginas web em base de dados (este armazenamento permite por sua vez a estruturação e caracterização das páginas do site: permissões, estrutura de menus, etc.)
- uma mais fácil definição de estrutura de base de dados
  - estrutura de tabelas (campos, PK, integridade referencial, tabelas associativas)
  - esta definição de tabelas permite uma mais fácil criação de formulários para operações CRUD)
  - definições 1:n (campos a considerar...)
- criação automática de operações CRUD (backoffice)
- definição, com possibilidade de parametrização, de componentes a incluir em páginas CMS (p.ex: bloco login, histórico de encomendas, alertas, mostra de loja, mini carrinho de compras, etc.)
- etc.

Este documento não pretende descrever de forma exaustiva SM5, mas sim servir como documento de apoio a uma formação inicial sobre esta framework, nomeadamente numa pequena sessão de apresentação.

**Não é objectivo deste documento servir de documento de apoio ao desenvolvimento.**

Goweb mantém um repositório (mediawiki) de informação em que, apesar de não disponibilizar informação consistente sobre todas as frameworks SMx, nomeadamente SM5, contém informação que pode ser relevante na resolução de questões pontuais, uma vez que descreve soluções encontradas por diversos programadores na implementação de projectos específicos e no desenvolvimento das frameworks SMx. Esta base de dados de conhecimento pode ser acedida em:

<http://php5.go/gowiki>

## Estrutura SM5

Sempre que um novo projecto SM5 entra em produção é criado um novo projecto base, ponto de partida para o desenvolvimento, com a seguinte estrutura de pastas e conteúdos:

```
<nome projecto>
  application
  application_admin
  library
  views
  public_html (document root deste projecto no servidor web)
  config.ini   (ficheiro de configuração “global”)
```

há outras pastas essenciais ao correcto funcionamento da framework (nomeadamente a pasta /cache ou /public\_html/uploads...). Apenas serão referidas as necessárias a este documento (pode ser consultado o projecto idsm5 para uma visão mais completa desta estrutura).

## Application

Cada módulo disponibilizado em Frontoffice numa aplicação SM5 terá uma pasta com o respectivo nome dentro da pasta Application. O conteúdo da pasta de um módulo será semelhante ao seguinte:

```
<módulo>
  Model   (pasta)
  Form    (esta pasta apenas existirá se necessário ao módulo!)
  Controller.php
  GenericController.php
  config.ini
  Admin.php
  Componente.php
  Setup.php
  Soap.php (este ficheiro apenas existirá se necessário à aplicação)
```

## *config.ini*

Este ficheiro de configuração, de que se mostra um possível conteúdo para o módulo catálogo, é utilizado para armazenar parametrizações, configurações específicas ao módulo:

```
[version]
number = 5.0.0

[base]

comentario.email_alert    = xxxx@goweb.pt
comentario.paginas        = 5
paginator.rows_per_page   = 5
paginator.links_per_page  = 10

paginatorlista.rows_per_page = 96
paginatorlista.links_per_page = 1

catálogo.loja            = 0
catálogo.relacao         = 1 ;; Mostrar no detalhe produtos relacionados

images.target             = /uploads/catalogo/imagens
images.prefix             = catálogo
images.sizes = small,medium,big

images.categoria.width    = 80
images.categoria.height   = 80
images.categoria.type_resize = resizeMaintainRatio;; resize;; crop

images.original.save       = 0
images.original.max_size   = 3072000 ;; em bytes
images.original.max_width  = 1040
images.original.max_height = 900
images.original.type_resize = resizeMaintainRatio;; resize;; crop

images.small.width         = 100
images.small.height        = 100
;; images.small.watermark   = /watermark.png
images.small.type_resize   = resizeMaintainRatio;; resize;; crop

images.medium.width        = 200
images.medium.height       = 200
;; images.medium.watermark  = /watermark.png
images.medium.type_resize  = resizeMaintainRatio;; resize;; crop

images.big.width           = 350
images.big.height          = 350
;; images.big.watermark     = /watermark.png
images.big.type_resize     = resizeMaintainRatio;; resize;; crop

files.target               = /uploads/catalogo/ficheiros
files.prefix               =
files.max_size             = 33333333
files.types                = "pdf,txt,text,plain,xls,doc"

;; configuracao para o modulo de Loja
loja.activa                = 1

;; cambios
cambios.activo = 1
cambios.moedas = Euro:euro:Left::Libra:pound:left::USD:#36:right
cambios.id_moeda_base = 6

;; RSS
rss.activo = 0 ;; colocar a 1 para activar os rss
```

O acesso ao conteúdo deste ficheiro a partir da plataforma é feito com recurso à classe `Goweb_Config_Ini`. O código seguinte mostra como instanciar um objecto para manipular este

ficheiro (neste exemplo pretendemos aceder à secção [base] que tem início na 3ª linha do ficheiro config.ini):

```
$this->_moduleConfig = new Gowebe_Config(Gowebe_Config_Ini::load(dirname(__FILE__) .  
DIRECTORY_SEPARATOR . 'config.ini', 'base'));
```

O exemplo seguinte mostra como aceder ao objecto instanciado, no caso ao conteúdo da linha

definição no **ficheiro config.ini**:

```
;; configuracao para o modulo de Loja (esta linha é um comentário!)  
loja.activa = 1
```

utilização (obtenção do parâmetro) na **aplicação**:

```
$loja_activa = $this->_moduleConfig->loja->activa;
```

## ***Pasta Model***

Esta pasta contém as classes com a definição de todas as tabelas da BD, específicas do módulo, ou geridas pelo módulo.

Tomando como exemplo o módulo Catálogo disponibilizado quando é criado um novo projecto, o código seguinte mostra a definição da tabela “Categorias de Artigos”:

```
<?php  
  
Gowebe::loadClass('Gowebe_Db_Model_Category');  
  
/**  
 * Gowebe Sitemanager 5 Framework  
 * @version 5.0.19  
 * @category Frontoffice  
 * @package Catalogo  
 * @subpackage Model  
 * @author I&D <dev@gowebe.pt>, Pedro Pinto <ppinto@gowebe.pt>  
 * @copyright 2009 (c) Gowebe (http://www.gowebe.pt)  
 */  
class Catalogo_Model_Category extends Gowebe_Db_Model_Category {  
  
    /**  
     * Nome da Tabela  
     *  
     * @var string  
     */  
    protected $_table = 'catalogo_categoria';  
  
    /**  
     * Chave Primaria  
     *  
     * @var string  
     */  
    protected $_primary = 'id';  
  
    /**
```

```

* Campo que sera retornado se um
* echo do objecto for chamado
*
* @var string
*/
protected $_toString = 'titulo';

/**
* Metodo obrigatorio. Informa a definicao da tabela
* referente ao Model em questao.
*/
protected function _setup() {
    $config = new Goweb_Config(Goweb_Config_Ini::load(APPLICATION_PATH . DIRECTORY_SEPARATOR . 'Catalogo' .
    DIRECTORY_SEPARATOR . 'config.ini', 'base'));
    $this->initLang();
    $this->setSyncCols(array('ordem', 'imagem'));
    $this->pai_id = $this->SelfKey(array('alias' => 'p', 'label' => translate('Categoria Pai'), 'default' => '0'));
    $this->ficha_id = $this->ForeignKey(array('model' => 'Catalogo_Model_Ficha', 'alias' => 'ficha', 'label' =>
    translate('Ficha dos Produto'), 'default' => '0'));
    $this->iva_id = $this->ForeignKey(array('model' => 'Catalogo_Model_Iva', 'alias' => 'iva', 'label' => translate('I.V.A dos
    Produto'), 'default' => '0'));
    $this->ordem = $this->Integer(array('default' => 0, 'label' => translate('Ordem')));
    $this->titulo = $this->Char(array('maxlength' => 200, 'notnull' => true, 'label' => translate('T&iacute;tulo')));
    $this->url = $this->Char(array('maxlength' => 200, 'label' => translate('Url Amig&aacute;vel'), 'on_update' =>
    '_urlFromTitulo', 'show' => true));
    $this->imagem = $this->Image(array('target' => $config->images->target,
        'prefix' => $config->images->prefix,
        'original' => $config->images->original->save,
        'max_size' => $config->images->original->max_size,
        'max_width' => $config->images->original->max_width,
        'max_height' => $config->images->original->max_height,
        'type_resize' => $config->images->original->type_resize,
        'small_max_width' => $config->images->categoria->width,
        'small_max_height' => $config->images->categoria->height,
        'small_type_resize' => $config->images->categoria->type_resize));
    $this->activo = $this->Boolean(array('default' => '1', 'label' => translate('Activo')));
    $this->actualizado = $this->Datetime(array('on_update' => '_currentDatetime', 'label' => translate('Actualizado em')));
    $this->criado = $this->Datetime(array('on_insert' => '_currentDatetime', 'label' => translate('Criado em')));
    $this->ip = $this->IpAddress(array('on_update' => '_remoteAddress', 'label' => translate('Endere&ccedil;o IP')));

    $this->hasMany('Catalogo_Model_Produto', 'categoria_id', 'produto');
}

/**
* Obter um url amig&acirc;vel com base no titulo
*
* @return String com um url amig&acirc;vel
*/
protected function _urlFromTitulo() {
    if ($this->url == '') {
        $this->url = $this->_clearString($this->titulo);
        $this->url = $this->_getParents($this->pai_id, $this->url);
    } else {
        $this->url = $this->_clearString($this->url);
    }
    $tmpModel = new Catalogo_Model_Categoria('db', false);
    $url = $this->url;
    $count = 0;
    while ($tmpModel->findBy('url', $url) && $tmpModel->id != $this->id){

```

```

        $count++;
        if ($count == 1) $url .= '-';
        $url=$this->url . $count;
    }
    $this->url = $url;
    return $this->url;
}

/**
 * Obter o url dos pais da categoria
 *
 * @param int $pai_id
 * @param string $parents
 * @return string
 */
protected function _getParents($pai_id = 0, $parents = ''){
    $categoriaModel = new Catalogo_Model_Categoria('db', false);

    if ($categoriaModel->find($pai_id)){
        if (strlen($parents) > 0 && strlen($categoriaModel->url) > 0) $parents = "/" . $parents;
        $parents = $categoriaModel->url . $parents;
        while ($categoriaModel->find($categoriaModel->pai_id)){
            if ($categoriaModel->url && strpos($parents, $categoriaModel->url) === false){
                if (strlen($parents) > 0 && strlen($categoriaModel->url) > 0) $parents = "/" . $parents;
                $parents = $categoriaModel->url . $parents;
            }
        }
    }
    return $parents;
}
}

```

É importante reparar na definição da classe

```
class Catalogo_Model_Categoria extends Goweb_Db_Model_Category {
```

O nome da classe obedece à estrutura <Modulo>\_Model\_<Nome>.

Sendo o nome do módulo **Catalogo** e **Categoria** o nome da classe que descreve a tabela (com nome **catalogo\_categoria** na BD) a regra acima determina que o nome desta classe é **Catalogo\_Model\_Categoria**. Esta classe é uma especialização de Goweb\_Db\_Model\_Category, mas o habitual é ser uma especialização de Goweb\_Db\_Model.

A análise desta classe, nomeadamente do seu método Setup, é importante pela diversidade de aspectos a abordar:

```
$config = new Goweb_Config(Goweb_Config_Ini::load(APPLICATION_PATH . DIRECTORY_SEPARATOR . 'Catalogo' .
DIRECTORY_SEPARATOR . 'config.ini', 'base'));
```

Instancia o objecto \$config para acesso a ficheiro config.ini, secção **base**, na pasta Application/Catalogo. O ficheiro config.ini contém os parâmetros para armazenamento de imagem, como se pode observar na definição do campo **Imagem**.



## Internacionalização

A plataforma SM5 gere as traduções criando réplica das tabelas para cada idioma (atribuindo a estas nomes no formato <nome\_tabela>\_<codigo\_idioma>.

```
$this->initLang();
```

Indica à plataforma que se trata de uma tabela que deve ser replicada nos vários idiomas disponibilizados no projecto em concreto (os idiomas disponibilizados estão definidos no config.ini “global”). Note-se que no caso particular da internacionalização o campo “chave primária” definido em:

```
/**
 * Chave Primaria
 *
 * @var string
 */
protected $_primary = 'id';
```

não é explicitamente instanciado (definição da chave primária fica a cargo do método initLang()). Por norma a chave primária é instanciada da seguinte forma:

```
$this->id = $this->Auto();
```

Por se tratar de uma tabela em que os conteúdos existem em mais de um idioma importa definir os campos cujo conteúdo não depende do idioma. No exemplo abaixo os campos ordem e imagem não dependem do idioma, por isso qualquer alteração que ocorra num destes campos, num qualquer idioma, deve ser replicada em todos os outros idiomas:

```
$this->setSyncCols(array('ordem', 'imagem'));
```

Nota: a definição das colunas a sincronizar é evidentemente característica de cada projecto! Esta avaliação terá de ser feita caso a caso!

### **`$this->selfKey()`, `$this->ForeignKey`, `$this->ForeignKeyMultiple`**

Propriedades do model que definem chave estrangeira para (respectivamente): a própria tabela, tabela distinta e tabela associativa.

**Selfkey** indica uma referência para a própria tabela. No caso concreto do model Catalogo, esta é a forma de implementar hierarquia simples.

**ForeignKey** indica chave estrangeira.

**ForeignKeyMultiple** é a forma da framework SM5 implementar tabelas associativas. Neste último caso esta definição é suficiente para a criação dessas mesmas tabelas, sendo obrigatória a criação da classe (ficheiro model) da tabela associativa.

A utilização de **hasMany** permite a implementação de integridade referencial. No exemplo apresentado, `"$this->hasMany('Catalogo_Model_Produto', 'categoria_id', 'produto');"` indica que a tabela produtos tem chave estrangeira para esta tabela, **Categorias**, o que impede a eliminação de um registo da tabela **Categorias** caso algum registo da tabela **Produtos** referencie, no seu campo **categoria\_id**, o **id** da tabela **Categorias** a remover.

#### **onupdate, oninsert**

Os campos **criado**, **actualizado** e **ip** têm como propriedade **oninsert** e **onupdate** que permitem a execução de acções quando um registo é criado ou alterado.

#### **Definição dos campos de uma tabela**

O método **setup** mostra diversos tipos de campos. Esta definição fornece à framework não só a informação necessária à criação das tabelas na base de dados, fornecendo a definição (e restrições) para cada um dos seus campos, como também informação para a automatização na criação de formulários.

O método **setup** apresentado mostra diversos tipos de campos: Char, Integer, Imagem, Boolean, Datetime, IpAddress (aconselha-se a consulta de todos tipos de campo em **library/Goweb/Db/Col**).

A forma genérica de definição de um campo de uma tabela obedece ao formato:

```
$this-><nome> = $this-><tipo>( array parâmetros );
```

**label**, **default**, **notnull**, **unique**, **maxlength** são alguns dos parâmetros mais utilizados (consultar **library** para lista exhaustiva de atributos).

Há tipos de campos que merecem especial atenção, como p.ex. **Image**, **File**, **Email**, **Auto...**

Muito importante: **'label' => translate('Url Amig&acute;vel')** é a forma correcta de utilizar constantes texto em qualquer ponto da aplicação (no caso específico do backoffice deve ser utilizado o método **translateAdmin**). Esta prática deve ser adoptada mesmo quando à partida a aplicação a desenvolver utiliza apenas um idioma, porque caso o cliente final decida adquirir um ou mais idiomas não será necessário proceder a qualquer alteração ao código fonte.

Para caracteres acentuados utilizar **sempre** html entities, isto é, **Amig&acute;vel** e não **Amig&cedil;vel**.

**urlFromTitulo** e **getParents** são acções definidas para este model (**Categorias**). Embora a sua utilização esteja de acordo (e seja a correcta de acordo com) o modelo MVC, esta abordagem deve ser ponderada em cada caso concreto por poder levantar problemas em actualizações da

framework (este aspecto será melhor entendido quando forem referidos os ficheiros **Controller.php** e **GenericController.php**)

## ***Setup.php***

O ficheiro **setup.php** indica à aplicação quais as tabelas necessárias ao funcionamento do módulo em causa. Todos os models correspondentes às tabelas da BD necessárias ao módulo devem ser indicados no método `initModels()`.

```
<?php

Goweb::loadClass('Goweb_Setup');

/**
 * Goweb Sitemanager 5 Framework
 * @version 5.0.0
 * @category Frontoffice
 * @package Catalogo
 * @subpackage Setup
 * @author I&D <dev@goweb.pt>, Pedro Pinto <ppinto@goweb.pt>
 * @copyright 2009 (c) Goweb (http://www.goweb.pt)
 *
 * As classes _Setup sao utilizadas para a criacao das tabelas
 * relativas aos Models dos Modulos.
 * Para cada Modulo existente no sistema existem ficheiros (Models)
 * que modelam a base de dados.
 * A partir destes Models que o Setup cria as tabelas referentes
 * aos modulos do sistema a ser instalado.
 */
class Catalogo_Setup extends Goweb_Setup {

    /**
     * Nome do modulo
     *
     * @var string
     */
    protected $_nome = 'Catalogo';

    /**
     * Titulo do modulo
     *
     * @var string
     */
    protected $_titulo;

    /**
     * Modulo pertence ao menu do backoffice
     *
     * @var boolean
     */
    protected $_menu = true;

    /**
     * Permitir inclusao de componentes
     *
     * @var boolean
     */
    protected $_componente = true;

    /**
     * Permitir pesquisa no modulo
     *
     * @var boolean
     */
}
```

```

*/
protected $_pesquisa = true;

/**
 * Modulo sujeito a permissoes de utilizadores
 *
 * @var boolean
 */
protected $_permissao = true;

/**
 * Constructor da classe
 *
 */
public function __construct(){
    $this->_titulo = translate('Catalogo');
}

/**
 * Este metodo adiciona os Models necessarios ao
 * setup do modulo.
 */
public function initModels() {
    $this->addModel('Catalogo_Model_Categoria');
    $this->addModel('Catalogo_Model_Produto');
    $this->addModel('Catalogo_Model_Imagem');
    $this->addModel('Catalogo_Model_Ficheiro');
    $this->addModel('Catalogo_Model_Relacao');
    $this->addModel('Catalogo_Model_Ficha');
    $this->addModel('Catalogo_Model_Element');
    $this->addModel('Catalogo_Model_Iva');
    $this->addModel('Catalogo_Model_Cambio');
    $this->addModel('Catalogo_Model_ProdutoCategoria');
}
}

```

Este método é por vezes utilizado para iniciar “conteúdo por default” para algumas tabelas como p.ex., países, moedas, etc.

Veremos mais adiante que a informação contida neste módulo é utilizada na criação da estrutura da base de dados, bem como na avaliação de alterações à estrutura de tabelas após a sua inicialização.

## Soap.php

Este ficheiro supostamente deve conter informação necessária à utilização de Web Services (não encontrei qualquer implementação nos projectos em que participei nem qualquer informação útil na documentação que pesquisei).

## Admin.php

A classe Admin permite a criação automática, em backoffice, de módulos (e acções para esses módulos), através da definição das secções a disponibilizar. (É possível definir novas acções em backoffice recorrendo à sua definição em Application\_Admin/<módulo>/Controller.php ou Application\_Admin/<módulo>/GenericController.php).

Exemplo de Admin.php para o módulo Catálogo com a definição de duas secções:

<?php

```
Goweb::loadClass('Goweb_Admin');
Goweb::loadClass('Goweb_Admin_Section');
Goweb::loadModel('Catalogo_Model_Iva');

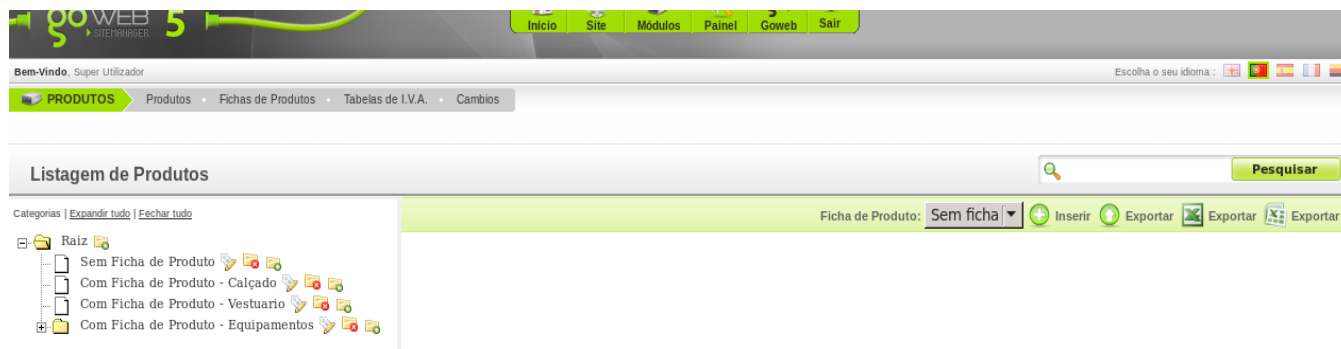
/**
 * Goweb Sitemanager 5 Framework
 * @version 5.0.0
 * @category Frontoffice
 * @package Catalogo
 * @subpackage Admin
 * @author I&D <dev@goweb.pt>, Pedro Pinto <ppinto@goweb.pt>
 * @copyright 2009 (c) Goweb (http://www.goweb.pt)
 *
 * As classes _Admin sao responsaveis pela criacao automatica do backoffice.
 */
class Catalogo_Admin extends Goweb_Admin {

    /**
     * Definicao para a montagem dinamica do back-office.
     */
    public function _setup() {
        $this->_title = translate('Catalogo');

        $categoriasSection = new Goweb_Admin_Section();
        $categoriasSection->setModel(Goweb::getModelInstance('Catalogo_Model_Categoria', 'db'));
        $categoriasSection->setTitle(translate('Categorias'));
        $categoriasSection->setTitleSingular(translate('Categoria'));
        $categoriasSection->setListCols(array('id', 'titulo'));
        $categoriasSection->setOrdering('criado');
        $categoriasSection->setSearchCols('titulo');
        $categoriasSection->setListAction('listProduto');
        $categoriasSection->setParentIdCol('pai_id');
        $this->_addSection('categorias', $categoriasSection);

        $ivaSection = new Goweb_Admin_Section();
        $ivaSection->setModel(Goweb::getModelInstance('Catalogo_Model_Iva', 'db'));
        $ivaSection->setTitle(translate('Tabelas de I.V.A.'));
        $ivaSection->setTitleSingular(translate('Tabelas de I.V.A.'));
        $ivaSection->setListCols(array('titulo', 'iva', 'criado'));
        $ivaSection->setOrdering('titulo');
        $ivaSection->setSearchCols('titulo');
        $this->_addSection('iva', $ivaSection);
    }
}
```

A imagem seguinte mostra o backoffice, módulo catálogo:



Na porção de código seguinte, extraída de Application\_Admin/Catalogo/GenericController, é definida a árvore de categorias visível na imagem, bem como a adição de duas novas secções: produtos e fichas.

```
/**
 * Parametros para a criacao da arvore
 *
 * @var array
 */
protected $_categoriasTreeParams = array('target' => null,
    'folderLinks' => true,
    'useSelection' => true,
    'useCookies' => false,
    'useLines' => true,
    'useIcons' => true,
    'useStatusText' => false,
    'closeSameLevel' => false,
    'inOrder' => false);

/**
 * Parametros para a criacao dos nos da arvore
 *
 * @var array
 */
protected $_categoriasNodesParams = array('rootName' => 'Raiz',
    'id' => 'id',
    'pid' => 'pai_id',
    'name' => 'titulo',
    'url' => '/catalogo/list_produto/categoria_id/',
    'urlId' => 'id',
    'target' => null,
    'title' => null,
    'icon' => null,
    'iconOpen' => null,
    'open' => null,
    'url_1' => '/catalogo/edit/section/categorias/id/',
    'url_1_icon' => '/media/images/icon_edit.png',
    'url_2' => '/catalogo/delete/section/categorias/id/',
    'url_2_icon' => '/media/images/icon_delete.gif',
    'url_3' => '/catalogo/add/section/categorias/pai_id/',
    'url_3_icon' => '/media/images/icon_add.png');

/**
 * @var array
 */
protected $_rootCategoriesNodeParams = null;

/**
 * Mensagem de Erro
 * @var string
 */
protected $_error = null;

/**
 * Mensagem de Sucesso
```

```

* @var string
*/
protected $_message = null;

/**
* Construtor da classe
*/
public function init() {
    if ($this->_getParam('section') && strcmp($this->_getParam('section'), 'catalogo') === 0) $this->_forward('listProduto');

    $this->_title = translateAdmin('Gest&atilde;o de Cat&aacute;logo');

    $this->_sections[] = array('title' => translateAdmin('Produtos'),
        'section' => 'produtos',
        'link' => '/catalogo/list_produto/');
    $this->_sections[] = array('title' => translateAdmin('Fichas de Produtos'),
        'section' => 'fichas',
        'link' => '/catalogo/list_fichas/');

    parent::init();
}

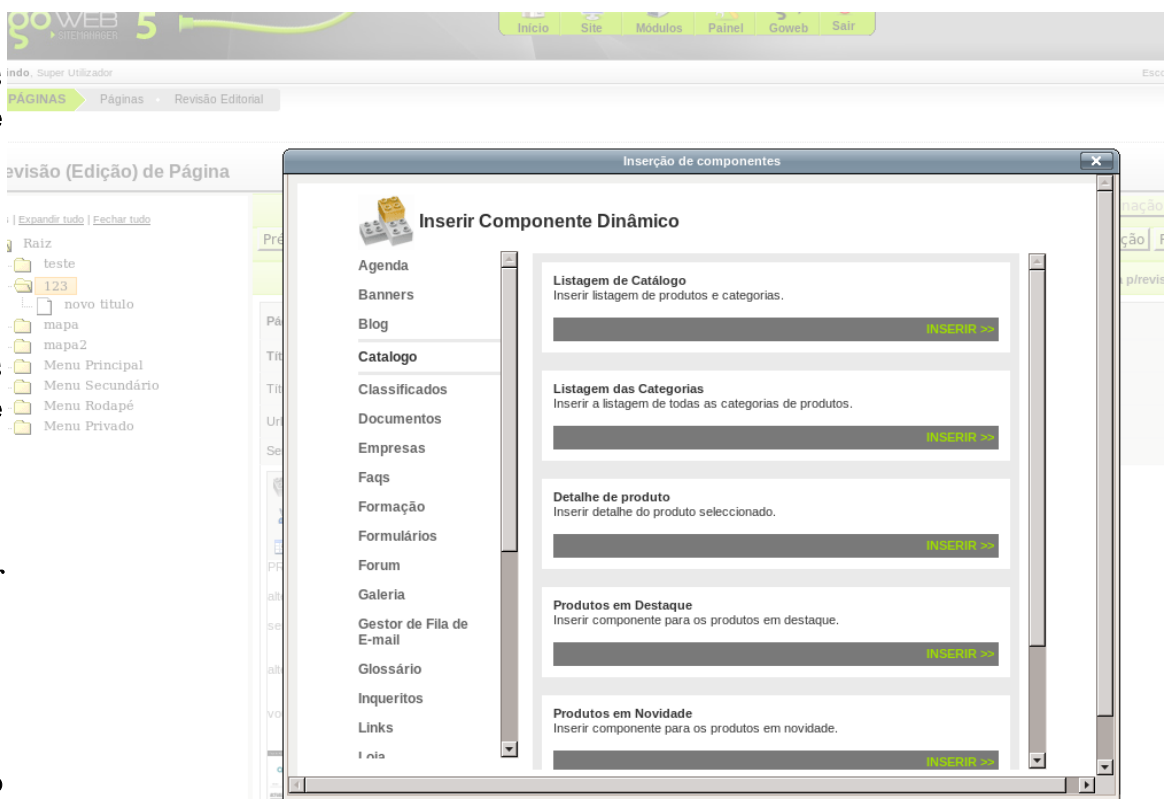
```

## Component.php

Esta classe permite a introdução dos componentes aqui definidos em páginas Cms, via backoffice. Essencial para disponibilizar esta funcionalidade a cliente final, não necessário a nível de desenvolvimento (qualquer componente pode ser chamado directamente num determinado tpl, como veremos adiante).

As imagens seguintes mostram a funcionalidade disponibilizada pelas classes Component:

“Listagem de Catálogo”, “Listagem de Categorias”, “Detalhe do produto”, “Produtos em Destaque”, etc., estão definidas no método init() da classe



## Catalogo\_Component:

```
class Catalogo_Component extends Cms_BaseComponent {

    /**
     * Este metodo informa ao sistema quais as accoes permitidas para este
     * modulo. Todas as accoes aqui definidas devem ser implementadas no
     * _Controller do modulo.
     */
    public function init() {
        parent::init();
        $this->_actions = array('list' => array('title' => translateAdmin('Listagem de Cat&aacute;logo'),
            'helper' => translateAdmin('Inserir listagem de produtos e categorias.')),
            'listCategorias' => array('title' => translateAdmin('Listagem das Categorias'),
            'helper' => translateAdmin('Inserir a listagem de todas as categorias de produtos.')),
            'detail' => array('title' => translateAdmin('Detalhe de produto'),
            'helper' => translateAdmin('Inserir detalhe do produto seleccionado.')),
            'destaques' => array('title' => translateAdmin('Produtos em Destaque'),
            'helper' => translateAdmin('Inserir componente para os produtos em destaque.')),
            'novidades' => array('title' => translateAdmin('Produtos em Novidade'),
            'helper' => translateAdmin('Inserir componente para os produtos em novidade.')),
            'menu' => array('title' => translateAdmin('Cat&aacute;logo Menu'),
            'helper' => translateAdmin('Inserir o menu das categorias de produtos.')));
    }
}
```

A complexidade do código fonte associado a estas actions é directamente proporcional à complexidade do componente associado (nomeadamente necessidade de obter parâmetros...):

```
/**
 * Listagem das Categorias
 */
public function listCategoriasAction() {
    $form = new GoweWeb_Form('list_categorias');
    return $form;
}

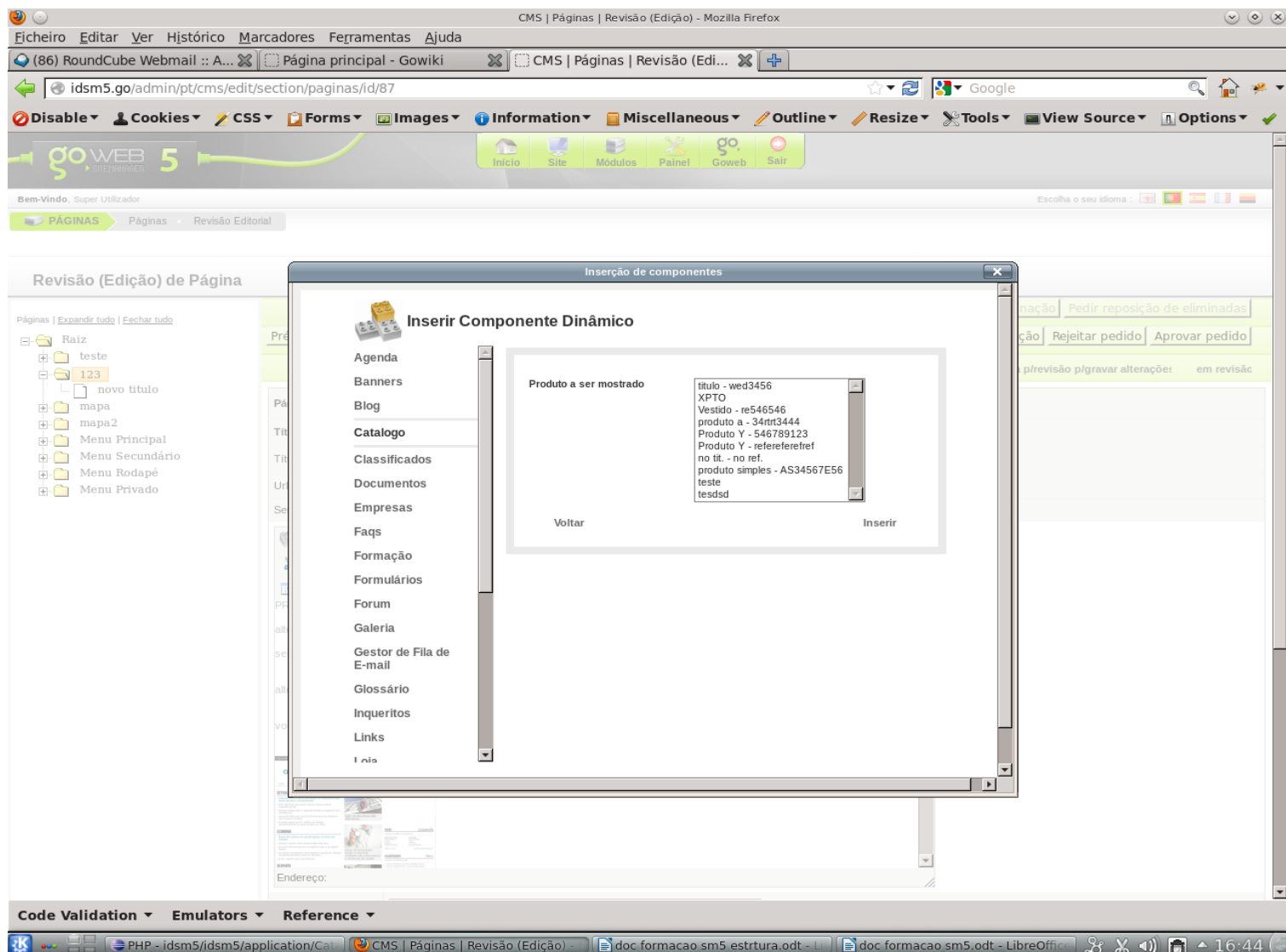
/**
 * Inserir detalhe de um produto
 */
public function detailAction() {
    // "id" - ID do produto
    $form = new GoweWeb_Form('produtos_list');
    $form->id = $form->Select();
    $form->id->label = translateAdmin('Produto a ser mostrado');
    $form->id->size = 10;

    $produtoModel = GoweWeb::getModelInstance('Catalogo_Model_Produto', 'db');
    $produtoModel->loadForeigns(false);
    $produtos = $produtoModel->fetchAll(array('id', 'titulo', 'referencia'));
    $options = array();
    foreach ($produtos as $produto) {
        $options[$produto->id] = $produto->titulo . (strlen($produto->referencia) ? ' - ' . $produto->referencia : "");
    }
    $form->id->options = $options;
    $form->id->validators->Required(translateAdmin('O campo <b>{{campo}}</b> &eacute; de preenchimento obrigat&oacute;rio.', $form->id->label));

    return $form;
}
```

Enquanto **listCategoriasAction()** representa apenas um “place holder” (ficando a implementação deste método a cargo do Controller ou GenericController), já **detailAction()** obriga a implementação mais complexa por solicitar ao utilizador parâmetros para a utilização deste componente (qual o produto cujo detail pretende...):





## ***GenericController.php***

Um Controller, no modelo MVC, é responsável por processar todos os pedidos recebidos pelo servidor e, em última instância, é responsável por delegar esses pedidos nos ActionControllers.

A plataforma SM5 foi desenhada por forma a permitir a comercialização de determinados módulos: Catálogo, Classificados, Documentos, Portefólio, Loja, Newsletter, etc... Estes módulos implementam o comportamento e funcionalidades característicos de determinada versão (funcionalidades por defeito). Este comportamento “por defeito” é implementado em GenericController.php.

Este é o ponto de partida para o desenvolvimento e personalização da aplicação para um determinado cliente.

Como este código está intimamente ligado à versão da framework a que pertence (no presente documento SM5) está sujeito a alterações, quer por melhoramentos introduzidos quer por correcção de bugs, estes ficheiros não devem, em circunstância alguma, sofrer qualquer alteração para implementar qualquer funcionalidade específica a um projecto: a actualização da plataforma faz-se actualizando quer a pasta library quer os GenericController de todos os módulos (frontoffice e backoffice). Qualquer personalização destes módulos é perdida quando a plataforma SMx (versão >= 5) é actualizada.

Como única forma de garantir a possibilidade de actualizar SMx, mantendo o desenvolvimento específico a um determinado projecto, GenericController deve ser extendido em Controller.php, ficando (apenas) nesta classe qualquer alteração ao código fonte.

### ***Controller.php***

Permite sobrescrever (adaptar/personalizar) quaisquer métodos de qualquer módulo, e adicionar novos, em um qualquer projecto.

É nesta Classe que todo o código alterado e todo o novo código desenvolvido deve ser colocado, sob pena de um qualquer projecto perder toda e qualquer personalização/desenvolvimento específico durante uma actualização da plataforma SMx!

Este princípio é válido tanto para frontoffice como para backoffice.

### ***Pasta Form***

A pasta Form contém a definição das classes correspondentes aos formulários utilizados pelo módulo (se o módulo não utiliza qualquer formulário esta pasta não é necessária).

Estas classes estão para os formulários como as classes model estão para as tabelas da base de dados. O código seguinte mostra a definição de um **formulário** para o módulo **Classificados**:

```
<?php

Goweb::loadClass('Goweb_Form');

/**
 * Goweb Sitemanager 5 Framework
 * @version 5.0.0
 * @category Frontoffice
 * @package Classificados
 * @subpackage Form
 * @author I&D <dev@goweb.pt>, Pedro Pinto <ppinto@goweb.pt>
 * @copyright 2009 (c) Goweb (http://www.goweb.pt)
 */
class Form_Classificado extends Goweb_Form {

    /**
     * Tipo de classificado
     *
     * @var array
```

```

*/
protected $_tipo;

/**
 * Tipo de contactos
 *
 * @var array
 */
protected $_contacto;

/**
 * Objecto Classificados_Model_Categoria
 *
 * @var Classificados_Model_Categoria
 */
protected $_categoriaModel = null;

/**
 * Construtor da classe
 */
public function __construct($categoriaModel) {

    $this->_tipo = array('1' => translate('Venda'),
                        '2' => translate('Compra'),
                        '3' => translate('Aluguer'),
                        '4' => translate('Precisa-se'),
                        '5' => translate('Oferece-se'),
                        '6' => translate('Trespassa'));

    $this->_contacto = array('1' => translate('Empresa'),
                             '2' => translate('Particular'));

    $this->_categoriaModel = $categoriaModel;
    parent::__construct('FormClassificado', '/classificados/save/', 'POST');
}

/**
 * Metodo obrigatorio as classes herdeiras do Goweb_Form
 */
protected function _setup() {

    $result = $this->_categoriaModel->fetchAll();
    if ($result) {
        $result = $result->toArray();
    }

    $this->categoria_id = $this->Select();
    if ($result) {
        foreach ($result as $categoria) {
            $this->categoria_id->addOption($categoria['id'], $categoria['titulo']);
        }
    }
    $this->categoria_id->setLabel('Categoria');

    $this->titulo = $this->Text();
    $this->titulo->setLabel(translate('T&iacute;tulo'));
    $this->titulo->validators->Required(translate('Por favor, preencha o campo T&iacute;tulo.));

    $this->descricao = $this->Textarea(array('cols' => 45, 'rows' => 8));
    $this->descricao->setLabel(translate('Descr&iacute;ç&atilde;o'));
    $this->descricao->validators->Required(translate('Por favor, preencha o campo Descr&iacute;ç&atilde;o.));

    $this->preco = $this->Text();
    $this->preco->setLabel(translate('Pre&ccedil;o'));
    $this->preco->validators->Required(translate('Por favor, preencha o campo Pre&ccedil;o.));
    $this->preco->validators->Float(translate('O campo Pre&ccedil;o deve ser um valor real.));

    $this->localidade = $this->Text();
    $this->localidade->setLabel(translate('Localidade'));

    $this->nome_contacto = $this->Text();
    $this->nome_contacto->validators->Required(translate('Por favor, preencha o campo Nome do Contacto.));
    $this->nome_contacto->setLabel(translate('Nome do Contacto'));

    $this->email = $this->Text();
    $this->email->setLabel(translate('E-mail'));
    $this->email->validators->Email(translate('O campo <b>{ {campo} }</b> n&atilde;o &eacute; um email v&aacute;lido.', translate('Email')));

    $this->telefone = $this->Text();
    $this->telefone->setLabel(translate('Telefone'));

    $this->tipo = $this->Select();
    foreach ($this->_tipo as $key => $value) {
        $this->tipo->addOption($key, $value);
    }
    $this->tipo->setLabel(translate('Sub-Categoria'));

```

```

$this->contacto = $this->Select();
foreach ($this->_contacto as $key => $value) {
    $this->contacto->addOption($key, $value);
}
$this->contacto->setLabel(translate('Tipo de Contacto'));

$this->imagem_1 = $this->File();
$this->imagem_1->setLabel(translate('Imagem 1'));

$this->imagem_2 = $this->File();
$this->imagem_2->setLabel(translate('Imagem 2'));

$this->imagem_3 = $this->File();
$this->imagem_3->setLabel(translate('Imagem 3'));

$this->imagem_4 = $this->File();
$this->imagem_4->setLabel(translate('Imagem 4'));

$this->submit = $this->Submit();
$this->submit->setValue(translate('Enviar'));
}
}

```

Esta classe define um formulário, a submeter para **/classificados/save**, com os campos categoria\_id, do tipo select list, titulo, tipo texto, descrição, tipo text area, etc...(ver pasta /library/Goweb/Form/Element para lista exhaustiva do tipo de elementos permitidos pela plataforma SM5).

A cada campo do formulário é possível atribuir um ou mais Validators, como por exemplo Required, Float, Email, etc. (Quando um formulário é utilizado para obter informação a armazenar numa determinada tabela da base de dados ocorrem ainda validações adicionais, estas relacionadas com restrições dos campos da tabela, como p.ex. campos 'notnull'...)

O método <form>->validate() faz uso dos validators de cada campo (quer tenham sido definidos no próprio formulário, quer tenham sido criados dinamicamente). Este método deve ser sempre chamado antes de proceder ao armazenamento da informação contida no formulário.

A existência destas classes permite a apresentação do formulário e sua validação de forma extremamente simples:

```

/**
 * Acao para o registo de um novo classificado
 */
public function newClassificadosAction() {
    // ===== //
    // instanciar a classe para o formulario //
    // ===== //
    Goweb::loadClass('Form_Classificado', dirname(__FILE__));

    $form = new Form_Classificado($this->_categoriaModel);
    $form->render();
    $this->view->form = $form;

    if (isset($form->categoria_id)) {
        $this->view->flag = true;
    } else {
        $this->view->flag = false;
    }

    // ===== //

    if ($this->_directRequest) { // significa que foi chamado directamente.
        Goweb::loadClass('Cms_Controller', APPLICATION_PATH);
        Cms_Controller::renderOutput($this->view,
            'form.tpl',
            'Classificados',
            'classificados-newClassificados',
            translate('Novo Classificado'));
        $this->_helper->viewRenderer->setRender($this->_canal);
    }
}

```

```

    } else {
        $this->_helper->viewRenderer->setRender('form');
    }
}

```

Tudo o que é necessário para apresentar o formulário é:  
instanciar um objecto do tipo de formulário pretendido, invocar o método render() desse objecto, e passar uma referência ao objecto view (que veremos posteriormente).

O resto do código fonte mostrado será abordado posteriormente.

Na classe Form\_Classificados definimos a action do formulário como sendo /classificados/save, pelo que teremos de ter um método public saveAction() responsável pelo tratamento dos dados submetidos no formulário (método GET ou POST de acordo com a definição na classe formulário):

```

/**
 * Acao para guardar o novo classificado em base de dados
 */
public function saveAction() {
    Goweb::loadClass('Form_Classificado', dirname(__FILE__));
    $form = new Form_Classificado($this->_categoriaModel);
    if ($form->validate()) {
        // inserir registo

        $dataInicio = date('Y-m-d');
        $dataFim = (date('Y') + 1) . '-' . date('m-d');

        $this->_classificadosModel->categoria_id = $this->_request->categoria_id;
        $this->_classificadosModel->titulo = $this->_request->titulo;
        $this->_classificadosModel->descricao = $this->_request->descricao;
        $this->_classificadosModel->preco = $this->_request->preco;
        $this->_classificadosModel->localidade = $this->_request->localidade;
        $this->_classificadosModel->nome_contacto = $this->_request->nome_contacto;
        $this->_classificadosModel->email = $this->_request->email;
        $this->_classificadosModel->telefone = $this->_request->telefone;
        $this->_classificadosModel->tipo = $this->_request->tipo;
        $this->_classificadosModel->contacto = $this->_request->contacto;
        $this->_classificadosModel->data_inicio = $dataInicio;
        $this->_classificadosModel->data_fim = $dataFim;

        $this->_classificadosModel->criado = $dataInicio;
        $this->_classificadosModel->actualizado = $dataInicio;
        $this->_classificadosModel->ip = $_SERVER['REMOTE_ADDR'];
    }
    /**
     * @todo verificar funcionalidades :-S
     */
    $imagem1 = $this->_uploadFile($this->_moduleConfig, $this->_request->imagem_1);
    $this->_classificadosModel->imagem_1 = $imagem1;

    $imagem2 = $this->_uploadFile($this->_moduleConfig, $this->_request->imagem_2);
    $this->_classificadosModel->imagem_2 = $imagem2;

    $imagem3 = $this->_uploadFile($this->_moduleConfig, $this->_request->imagem_3);
    $this->_classificadosModel->imagem_3 = $imagem3;

    $imagem4 = $this->_uploadFile($this->_moduleConfig, $this->_request->imagem_4);
    $this->_classificadosModel->imagem_4 = $imagem4;

    $this->_classificadosModel->save();

    return $this->listCategorias();
} else {
    // enviar para newClassificados
    foreach ($form->getErrorMessage() as $msg) {
        echo $msg . '<br/>';
    }
    return $this->newClassificadosAction();
}
}

```

Este método, simples, invoca o método validate() do formulário. Se o formulário é válido grava

um novo registo na tabela definida no model `$this->classificadosModel` e evoca a listagem de categorias, senão obtém todas as mensagens de erro e volta a apresentar o formulário ao utilizador.

The screenshot shows a web form titled "NOVO CLASSIFICADO" (New Classified). The form is divided into sections: "Dados do Anúncio" (Advertisement Data) and "Colocação de Imagens" (Image Placement). In the "Dados do Anúncio" section, there are dropdown menus for "Categoria" (set to "Vagas") and "Sub-Categoria" (set to "Venda"). Below these are input fields for "Título" (Title) and "Descrição" (Description), both marked with red error icons and the message "Por favor, preencha o campo". Further down are fields for "Preço" (Price), "Localidade" (Location), "Nome do Contacto" (Contact Name), "E-mail", "Telefone" (Phone), and "Tipo de Contacto" (Contact Type), all also marked with red error icons and the same message. There is also an "Empresa" (Company) dropdown. The "Colocação de Imagens" section contains four image upload fields labeled "Imagem 1" through "Imagem 4", each with a "Procurar..." (Search...) button. At the bottom of the form is a "submit" button.

A aplicação disponibiliza tratamento de formulários mais evoluído

dos e complexos, mas o exemplo apresentado serve perfeitamente à ilustração destas classes.

## Application\_Admin

A implementação do Backoffice difere da implementação Frontoffice porque enquanto os módulos disponibilizados em Frontoffice têm de o ser explicitamente na pasta Application, SM5 permite a implementação automática do backoffice através da utilização das classes Admin.

Assim, na pasta Application\_Admin, apenas vamos encontrar pastas correspondentes ao módulos cujo backoffice não é implementado em Application/<nome módulo>/Admin.php ou, sendo implementados pelas classes Admin, carecem de mais funcionalidades e/ou personalização de acordo com as especificações do projecto em implementação.

SM5 disponibiliza em backoffice módulos muitos específicos como **Dashboard**, **Painel**, **PainelGoweb**, **Setup** além de outras funcionalidades e utilitários, que veremos em secção própria.

Por agora faremos uma avaliação dos módulos Backoffice seguindo a mesma abordagem utilizada para a apresentação dos módulos Frontoffice.

A pasta Application\_Admin contém, após criação de novo projecto, diversas pastas, correspondendo cada uma a um módulo backoffice. Cada uma dessas pastas terá um conteúdo semelhante a:

```
<módulo>
  <Form>
  <views>
  Controller.php
  GenericController.php
```

É importante recordar que os módulos backoffice disponibilizados exclusivamente pelas classes Application/<módulo>/Admin.php não terão uma pasta com o seu nome em Application\_Admin.

As pastas **Form** e **views** apenas existirão se necessárias à implementação do respectivo módulo backoffice.

### ***Pasta Form***

A pasta Form, tal como em frontoffice, conterá as classes correspondentes aos formulários que o módulo utiliza. Estes formulários são instanciados e validados tal como em frontoffice.

O código seguinte corresponde ao formulário utilizado pelo módulo PainelGoweb para edição de ficheiros de configuração:

```
<?php
```

```
Goweb::loadClass('Goweb_Form');

/**
 * Goweb Sitemanager 5 Framework
 * @version 5.1.2
 * @category Backoffice
 * @package Index
 * @subpackage Form
 * @author I&D <dev@goweb.pt>, Pedro Pinto <ppinto@goweb.pt>
 * @copyright 2010 (c) Goweb (http://www.goweb.pt)
 */
class Form_Edit extends Goweb_Form {

    /**
     * Formulário de pesquisa generica
     */
    protected function _setup() {
        $this->ficheiros = $this->Select();
        $this->ficheiros->label = translateAdmin('Escolha o Ficheiro a Editar');
        goweb::loadClass('Zend_Cache');
        $cache = Zend_Registry::get('cache');
        if (!$options = $cache->load('getConfigFiles')) {
            $options = $this->_getConfigFiles();
            if (is_writable(ROOT_PATH . '/config.ini')) {
                $options['config.ini'] = 'config.ini';
            }
            asort($options);
        }
        $this->ficheiros->options = $options;
        $this->ficheiros->setAttrib('onchange', 'updateFileContents(this.value)');
        $this->edit = $this->Textarea();
        $this->edit->label = translateAdmin('Ficheiro');
        $this->submit = $this->Submit();
        $this->submit->value = translate('Gravar');
        $this->submit->class = 'form_submit';
    }

    /**
     * Obter os ficheiros de configuração
     *
     * @param String $directory
     * @return Array ini files
     */
    protected function _getConfigFiles($directory = APPLICATION_PATH){
        $array_items = array();
        if ($handle = opendir($directory)) {
            while (false !== ($file = readdir($handle))) {
                if ($file != "." && $file != ".." && substr($file,0,1) != '.') {
                    if (is_dir($directory . "/" . $file)) {
                        $array_items = array_merge($array_items, $this->_getConfigFiles($directory . "/" . $file));
                    } else {
                        if (!strcmp('config.ini', $file)){
                            if (is_writable($directory . "/" . $file)){
                                $file = str_replace(APPLICATION_PATH . DIRECTORY_SEPARATOR, "", $directory) . "/" . $file;
                                $file = preg_replace("/\\//si", "/", $file);
                                $array_items[$file] = $file;
                            }
                        }
                    }
                }
            }
        }
        closedir($handle);
    }
    return $array_items;
}
}
```

Repare-se na utilização de `translateAdmin()` em vez de `translate()`, uma vez que estamos a implementar módulos backoffice.

A instrução seguinte mostra como podemos definir atributos para os elementos do formulário:

```
$this->ficheiros->setAttrib('onchange', 'updateFileContents(this.value)');
```

Neste exemplo a select list **ficheiros** terá um atributo, `onchange`, que invocará a função javascript **updateFileContents()**, que terá de estar definida na página gerada pela aplicação.



Para entendermos a geração de páginas HTML precisamos introduzir o conceito de view.

## *Pasta views*

A classe GoweView é um wrapper para a classe Zend\_View, e é a classe responsável pela “porção” view no modelo MVC. Existe para ajudar a separar a codificação da apresentação, da codificação dos Controllers e Models. A utilização da view acontece em duas etapas: no controller criamos uma instância de GoweView e assignamos variáveis a essa instância; em seguida no controller solicitamos a esta classe o render de uma determinada view gerando assim o output.

O código das views dos módulos backoffice está contido em ficheiros <nome view>.tpl, armazenados na pasta views de cada módulo.

O código seguinte mostra parcialmente o ficheiro cambios.tpl do módulo backoffice PaineiGoweView.

```
<?php $this->loadCss('/admin/media/css/dtree.css'); ?>
<?php $this->loadJavascript('/admin/media/js/dtree.js'); ?>

<div id="accoes_bg" style="z-index:1;">
  <div id="titulo_accao" style="z-index:1;">
    <?php if (isset($section_title)): ?>
    <nobr>Listagem de <?= $section_title; ?></nobr>
    <?php endif; ?>
  </div>
</div>

<script type="text/javascript">
<!--
var selection = false;

function invert() {
  if ( selection == false ) {
    selection = true;
  } else {
    selection = false;
  }

  for (var i=0; i<document.deleteList.elements.length; i++) {
    var e = document.deleteList.elements[i];
    if ( selection == true ) {
      e.checked = true;
    } else {
      e.checked = false;
    }
  }
}

function clean() {
  selection = false;
  document.deleteList.selectAll.checked = false;
}
-->
</script>

<div id="conteudo">
  <table width="100%" border="0" align="center" cellpadding="0" cellspacing="2" class="total">
    <tr>
      <td height="320" valign="top" id="manutencaodisplay">
        <div id="acessibilidade">
          <?php if (array_key_exists('add', $links)): ?>
            <input type="button" id="inserir" name="inserir" class="form_inserir" value="Inserir"
            onClick="location='<?= $links['add']; ?>';">
          <?php endif; ?>
        </div>
      </td>
    </tr>
  </table>
</div>
```

```
<br />  
<table border="0" align="center" cellpadding="0" width="700" class="total">
```

( ... )

Como a análise deste código permite constatar, um ficheiro view (\*.tpl) pode conter código php, html, javascript, ...

Os métodos **loadCss** e **loadJavascript** permitem inclusão de ficheiros css e javascript, respectivamente...

### ***GenericController.php***

Tal como em Application, as classes GenericController.php em Application\_Admin/<módulo>/ implementam o comportamento por defeito dos módulos respectivos (neste caso módulo Backoffice).

Estas classes não devem ser alteradas, pelos motivos já expostos anteriormente. Quaisquer alterações devem ser implementadas nas classes Controller.php.

### ***Controller.php***

Permite sobrescrever (adaptar/personalizar) quaisquer métodos, ou acrescentar novos, a qualquer módulo backoffice em um qualquer projecto.

Tal como já referido anteriormente, é nesta Classe que todo o código alterado e todo o novo código desenvolvido deve ser colocado, sob pena de um qualquer projecto perder toda e qualquer personalização/desenvolvimento específico durante uma actualização da plataforma SMx!

## **Módulos Backoffice: Setup, Painei, PaineiGoweb, Dashboard**

O correcto funcionamento da plataforma SM5 exige o conhecimento de alguns módulos backoffice.

### ***Módulo Setup***

O módulo Setup permite a criação das tabelas definidas nas várias classes Application/<módulo>/Setup.php, na base de dados definida em /config.ini.

O ficheiro /config.ini descreve o tipo de base de dados a utilizar, o seu nome e as credenciais de acesso.



**Nota importante:** os menus backoffice não permitem o acesso a este módulo pelo que, após login em backoffice, é necessário indicar o url no formato:

`http://<url do site>/admin/<idioma>/setup`

Este módulo cria as tabelas que ainda não existam na base de dados e, caso existam, compara a sua estrutura com a definição na respectiva classe Model, reportando, se necessário, os comandos SQL a executar para a criação ou actualização da estrutura das tabelas.

Apenas os comandos SQL para criação de tabelas são executados, os comandos para criação, alteração ou remoção de campos e índices podem ser copiados da janela do browser para posterior execução com o cliente de acesso à base de dados da preferência de cada um .

As opções 'Converter BD em UTF-8' e 'adicionar tradução de URL' só estão disponíveis nesta opção.

### ***Módulo Painel***

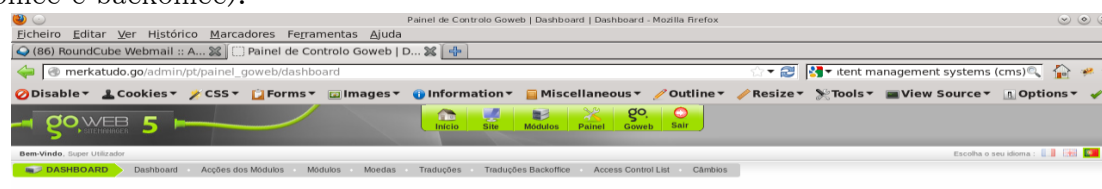
O módulo Painel permite as acções indicadas na imagem seguinte:



É através deste módulo que são definidos os perfis e contas de utilizadores de **sistema**, ou seja, utilizadores com acesso ao **backoffice**.

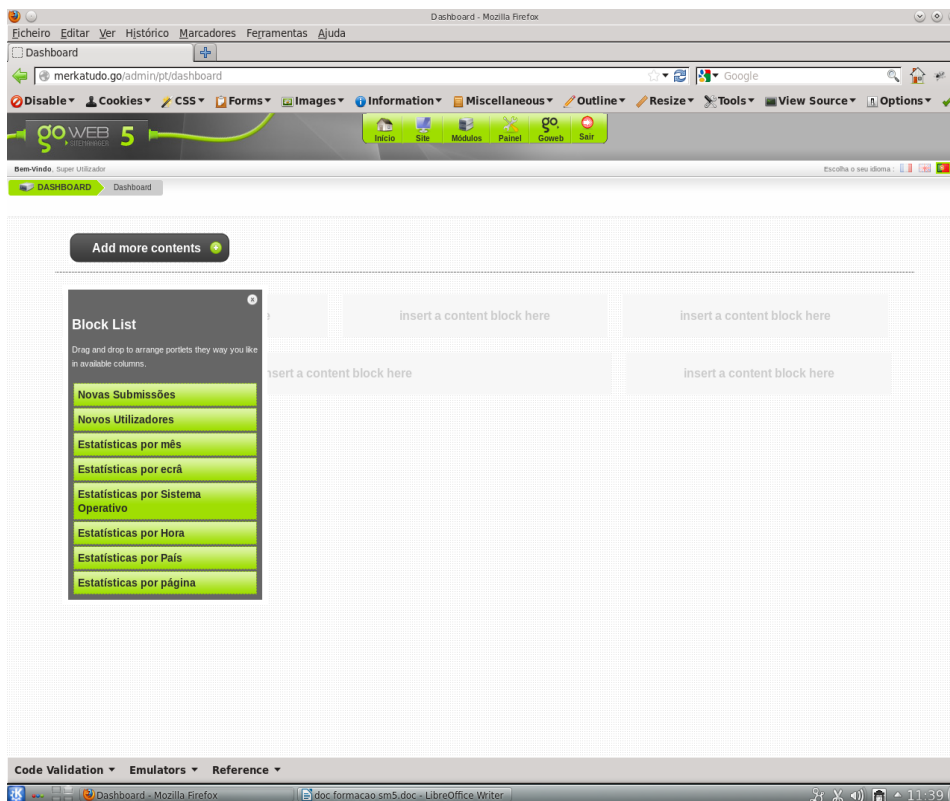
### ***Módulo PainelGoweb***

O módulo PainelGoweb permite, além de funcionalidades como publicação de site, edição de ficheiros de configuração, importação de catálogo, etc. (ver imagem seguinte), parametrização do acesso a módulos (pesquisar, pode ser componente, ...), definição de ACL e traduções (frontoffice e backoffice).



***Módulo  
Dashboard***





O dashboard, página de entrada no backoffice após login válido, disponibiliza alguns “portlets” que cada utilizador pode, fazendo drag&drop para uma das células disponíveis, colocar na página (a página dashboard é personalizada por utilizador backoffice, sendo as preferências de cada um armazenadas recorrendo a cookies).

A sua implementação é simples, resumindo-se à criação dos métodos necessários em `Application_Admin/Dashboard/Controller.php` (os portlets mostrados na imagem são os disponibilizados “por defeito” pelo `GenericController.php`) e à criação das respectivas views.

Comecemos por analisar o ficheiro Dashboard.tpl:

```
<div id="page_wrapper">

  <div class="botao">
    <a href="#" id="portal-block-list-link" title="Click to add portlets" name="portal-block-list-link" class="bt_content"><span>Add more
contents&nbsp;&nbsp;&nbsp;</span></a>
  </div>
  <div id="content_portal"><!-- begin content -->

    <div id="portal">

      <br class="clear" />

      <div class="portal-column" id="portal-column-0">
        <h2>insert a content block here</h2>
      </div>

      <div class="portal-column" id="portal-column-1">
        <h2>insert a content block here</h2>
      </div>

      <div class="portal-column" id="portal-column-2">
        <h2>insert a content block here</h2>
      </div>

    </div>

  </div>
</div>
```

```

<div class="portal-column" id="portal-column-3">
  <h2>insert a content block here</h2>
</div>

<div class="portal-column" id="portal-column-4">
  <h2>insert a content block here</h2>
</div>
<div class="portal-column" id="portal-column-block-list" style="display: none;">
  <div id="portal-column-block-list-bt_close" class="bt_close"><?php echo $this->img('/admin/media/js/portal/block_close.png'); ?></div>
  <h2 class="block-list-handle">Block List</h2>
  <p>Drag and drop to arrange portlets they way you like in available columns.</p>
  <?php echo $this->component('/dashboard/gowebRss');?>
  <?php echo $this->component('/dashboard/twitter');?>
  <?php echo $this->component('/dashboard/form');?>
  <?php echo $this->component('/dashboard/loja');?>
  <?php echo $this->component('/dashboard/usuarios');?>
  <?php echo $this->component('/dashboard/inqueritos');?>
  <?php echo $this->component('/dashboard/newsletter');?>
  <?php echo $this->component('/dashboard/visitas');?>
  <?php echo $this->component('/dashboard/stats/type/month');?>
  <?php echo $this->component('/dashboard/stats/type/screen');?>
  <?php echo $this->component('/dashboard/stats/type/os');?>
  <?php echo $this->component('/dashboard/stats/type/country');?>
  <?php echo $this->component('/dashboard/stats/type/hour');?>
  <?php echo $this->component('/dashboard/stats');?>
</div>
</div>
</div><!-- end content_portal -->
</div>
<script type="text/javascript">
<!--
var settings = {};
var portal;
var portalCookies = new Cookies();
function portalInit() {
  portal = new Portal();
  portal.applySettings(settings);
  portal.readCookie();
}
Event.observe(window, 'load', portalInit, false);
//-->
</script>

```

(ver pasta </admin/media/js/portal>, que contém javascript, css e imagens necessárias ao objecto portal).

Este tpl disponibiliza quer os place holders para drop dos portlets quer a chamada aos componentes utilizados.

A imagem seguintes mostra um portlet não incluído “por default” em SM5:

Add more contents +

### Block List

Drag and drop to arrange portlets they way you like in available columns.

Novas Submissões

Encomendas do site

Novos Utilizadores

Revisão Editorial

Resultados Inqueritos

Novos Inscritos na Newsletter

### Fila de e-mail

E-mails em fila: origens 4, e-mails:199

31 - loja

165 - Newsletter-mais agendamento

1 - Newsletter-teste 2

2 - utilizadores

[ver mais](#)

[enviar agora](#)

E-mails com falha: origens 2, e-mails:5

4 - Newsletter-teste 2

1 - utilizadores

[ver mais](#)

insert a content block here



## Views: exemplos de utilização, conceito de canal

Tomemos como exemplo o seguinte código fonte, extraído de /Application/Banners/GenericController.php:

```
/**
 * Accao para incluir um banner fixo no local especificado sem
 * validacoes.
 */
public function fixedAction() {
    if (($this->_validateId($this->_getParam('banner_id'))) &&
        ($this->_bannerModel->find($this->_getParam('banner_id'))) && $this->_bannerModel->activo) {
        $array_banners = array();
        $array_banners[$this->_bannerModel->id] = array();
        foreach ($this->_bannerModel->getColsNames() as $col) {
            if ($col == 'link_popup') {
                $array_banners[$this->_bannerModel->id]['popup'] = $this->_bannerModel->{$col};
            } else {
                $array_banners[$this->_bannerModel->id][$col] = $this->_bannerModel->{$col};
            }
        }

        $this->view->banner = array();

        $this->_addViews($this->_bannerModel->id);
        $this->view->banner = $array_banners[$this->_bannerModel->id];
        $this->view->banner_id = $this->_bannerModel->id;

        $this->view->upload_path = 'uploads' . DIRECTORY_SEPARATOR . 'banners' . DIRECTORY_SEPARATOR;

        if ($this->_directRequest) { // significa que foi chamado directamente.
            Gowebe::loadClass('Cms_Controller', APPLICATION_PATH);
            Cms_Controller::renderOutput($this->view,
                'banner.tpl',
                'Banners',
                'banners-list',
                translate('Listagem de Banners'));
            $this->_helper->viewRenderer->setRender($this->_canal);
        } else {
            $this->_helper->viewRenderer->setRender('banner');
        }
    } else {
        $this->_helper->viewRenderer->setNoRender();
    }
}
```

Para uma melhor leitura importa acrescentar:

`$this->_canal` está definido no topo da classe:

```
/**
 * canal a ser usado neste controller
 *
 * @var String
 */
protected $_canal = 'base';
```

No método `init()` desta classe está definido o caminho para pesquisa dos ficheiro do tipo view associados a este módulo (ficheiros com extensão tpl):

```
$this->view->setViewsPath(VIEWS_PATH . DIRECTORY_SEPARATOR . 'Banners');
```

`VIEWS_PATH` está definida no ficheiro `index.php` (localizado na root do site):

```
define('VIEWS_PATH', ROOT_PATH . DIRECTORY_SEPARATOR . 'views');
```

Fica assim explicada a existência da pasta /views referida ao abordar a estrutura dos projectos SM5: é nesta pasta que estão armazenados quer os ficheiros tpl que correspondem aos 'canais' quer, na pasta referente a cada módulo, os ficheiros tpl referentes a cada um. Recorde-se que os módulos backoffice têm as suas views localizadas na pasta /Application\_Admin/<módulo>/views.

Interpretando este código fonte verificamos que, se o valor de **banner\_id** (passado como parâmetro e obtido com **\$this->getParam('banner\_id')**), é válido (fazendo uso de **\$this->validateId(\$this->getParam('banner\_id'))**) e existe um registo na tabela banners (referenciada por **\$this->bannersModel**) com id (ou melhor, com chave primária) igual a esse valor (método **find** da classe **Model**) e esse registo tem o campo **activo** com valor igual a 1, atribuímos ao objecto View todos os valores necessários ao render correcto da view a apresentar ao utilizador. No nosso exemplo essa view corresponde ao ficheiro banner.tpl (referenciada como '**banner.tpl**' ou simplesmente '**banner**', de acordo com o método utilizado: método **renderOutput** de **Cms\_Controller** ou método **viewRenderer**).

**\$this->directRequest** é utilizado para determinar se pretendemos **apenas** o render do ficheiro '**banner.tpl**' (incluir na página html a processar) ou o render do ficheiro '**banner.tpl**' **enquadrado** numa página '**master**' (a que chamamos **canal** nas plataformas SMx), sendo necessário, neste último caso, o render de ambas as views.

O conteúdo do ficheiro banner.tpl é o seguinte:

```
<?php echo $this->loadCss('/media/css/banner.css'); ?>
<?php if (isset($banner) && $banner) : ?>
    <?php if ($banner['flash']) : ?>
        <?php echo $this->htmlFlash($baseUrl . $upload_path . $banner['flash'],
            array('width' => $banner['width'],
                'height' => $banner['height'],
                'id' => 'banner' . $banner['id'],
                'align' => 'middle',
                'class' => 'flash',
                'name' => 'banner' . $banner['id'],
                array('quality' => 'high',
                    'wmode' => 'transparent',
                    'bgcolor' => '#ffffff',
                    'allowsriptaccess' => 'sameDomain')); ?>
    <?php elseif ($banner['imagem']) : ?>
        <a title="<?php echo $this->escapeTitle($banner['titulo'])?>" href="<?php echo $this->link('/banners/click_redirect/id/' . $banner_id); ?>"<?php if
($banner['popup']) : ?> target="blank"<?php endif; ?>escapeTitle($banner['titulo'])?>" title="<?php echo $this->escapeTitle($banner['titulo'])?>" border="0" /></a>
    <?php endif; ?>
<?php else : ?>
<?php echo translate('N&atilde;o existem banners nesta categoria'); ?>
<?php endif; ?>
```

Este tpl, extremamente simples começa por carregar o ficheiro css indicado na primeira linha. Se o parâmetro \$banner foi passado a esta view é gerado o código para apresentação do respectivo banner. Este código, como é evidente, é distinto caso se trate de um banner em flash ou um banner composto por uma imagem. O link associado ao banner é também distinto, conforme este deva abrir uma janela popup ou simplesmente redireccionar para um determinado url.

Se o parâmetro \$banner não está definido (não foi fornecido a esta view) é afixada na página gerada a mensagem 'não existem banners...'

O conteúdo do canal **base.tpl** é bastante mais complexo, como se pode confirmar através deste excerto:

```
<?php
if (!isset($id) || !$id) {
    $id = "0";
}

$this->loadJavascript('/media/js/rollover.js');
$this->loadJavascript('/media/js/utills.js');
$this->loadJavascript('/media/js/prototype.js');
$this->loadJavascript('/admin/media/js/scriptaculous/scriptaculous.js');
$this->loadJavascript('/media/js/slide_menu.js');
$this->loadJavascript('/media/js/pushup/js/pushup.js');
$this->loadJavascript('/media/js/pro_drop_2/stuHover.js');
$this->loadJavascript('/media/js/prototype/product.js');
$this->loadCss('/media/js/pro_drop_2/pro_dropdown_2.css');
$this->loadCss('/media/js/pushup/css/pushup.css');
$this->loadCss('/media/css/base.css');
$this->loadCss('/media/css/cms.css');
$this->loadCss('/media/css/usuarios.css');
$this->loadCss('/media/css/slide_menu.css');
$this->loadCss('/media/css/glossario.css');
$this->loadCss('/media/css/links.css');
$this->loadCss('/media/css/loja.css');
$this->loadCss('/media/css/pullup.css');
$this->headScript()->appendScript("var baseUrl = " . $baseUrl . ";\n");
$this->headScript()->appendScript("var langSM = " . $session->lang . ";\n");
$config = Goweb::registry('config'); ?>
<?php echo $this->doctype() ?>

<html xmlns="http://www.w3.org/1999/xhtml" lang="<?php echo $session->lang; ?>" xml:lang="<?php echo $session->lang; ?>">
<head>
<?php $this->headMeta()->appendHttpEquiv('Content-Type',
    'text/html; charset=' . CHARSET)
    ->appendHttpEquiv('Content-Language', $session->lang);
$this->headMeta()->appendHttpEquiv('Copyright', 'Goweb 2012');
$this->headMeta()->appendHttpEquiv('Author', 'goweb.pt');
$this->headMeta()->appendHttpEquiv('Generator', 'GowebSiteManager.com 5');
$this->headMeta()->appendHttpEquiv('X-UA-Compatible', 'IE=8');
// setting the site in the title; possibly in the layout script:
$this->headTitle($config->painel->title);
if (isset($titulo) && $titulo == strip_tags($titulo)) $this->headTitle(html_entity_decode($titulo, ENT_NOQUOTES, CHARSET));
// setting a separator string for segments:
$this->headTitle()->setSeparator(' / ');
if ($config->painel->meta_copyright)
    $this->headMeta($config->painel->meta_copyright, 'copyright');
if ($config->painel->meta_keywords)
    $this->headMeta($config->painel->meta_keywords, 'keywords');
if (isset($keywords))
    $this->headMeta($keywords, 'keywords');
if ($config->painel->meta_description)
    $this->headMeta($config->painel->meta_description, 'description');
?>
<?php foreach ($this->getCss() as $css): ?>
<?php $this->headLink()->appendStylesheet($baseUrl . $css) ?>
<?php endforeach; ?>
<?php // $this->headLink()->appendStylesheet($baseUrl . $this->GetCssMinified()); ?>
<?php $this->headScript()->appendFile($baseUrl . '/media/js/go_admin.js')?>
<?php foreach ($this->getJavascripts() as $javascript): ?>
<?php $this->headScript()->appendFile($baseUrl . $javascript)?>
<?php endforeach; ?>
<?php // $this->headScript()->appendFile($baseUrl . $this->GetJavascriptsMinified()); ?>
<?php $this->headScript()->appendFile($baseUrl . '/media/js/AC_RunActiveContent.js')?>
<?php echo $this->headMeta() ?>
<?php foreach ($this->getRss() as $rss): ?>
<link rel="alternate" type="application/rss+xml" title="RSS" href="<?php echo $this->link($rss); ?>" />
<?php endforeach; ?>
.
.
.
.
<?php echo $config->painel->google_analytics; ?>
</head>
```

Neste tpl é facilmente perceptível, nomeadamente a nível do header, a preocupação em obter uma página 'bem formada' e de acordo com as boas práticas instituídas.

O excerto seguinte mostra que este canal é responsável pela criação de uma página com header, menus, corpo e footer. Mostra também a forma de adicionar conteúdo providenciado por componentes.

```
<div id="container">
<!--START CONTAINER-->
<div id="header">
<!--START HEADER-->
    <?php echo $this->component('/loja/mini_carrinho/'); ?>
    <?php echo $this->component('/utilizadores/loginBlock/'); ?>
    <?php echo $this->component('/cms/pesquisa/'); ?>
<!--END HEADER-->
</div>
<div id="menu">
    <?php echo $this->component('/cms/menu-list4/pai_id/0/id_selected/'. $id); ?>
</div>
<div id="content">
<!--START CONTENT-->
    <h1><?php echo $titulo; ?></h1>
    <br/>
    <?php if (isset($error) && $error):?>
        <div class="messages">
            <div class="error"><?php echo $error; ?></div>
        </div>
    <?php endif; ?>
    <?php if (isset($message) && $message): ?>
        <div class="messages">
            <div class="message"><?php echo $message; ?></div>
        </div>
    <?php endif; ?>
    <?php echo $output ?>
<!--END CONTENT-->
</div>

    <div id="push"></div>
<!--END CONTAINER-->
</div>
<div id="footer">
<!--START FOOTER-->
    <div class="left">GOWEB &copy; 2010</div>
    <div class="center"><?php echo $this->component('/cms/menu-list-footer/pai_id/21/limit/4'); ?></div>
    <div class="right">
        <a href="http://www.goweb.pt" target="_blank" title="<?php echo translate('Goweb | Web Design, Web Marketing, Solu&ccedil;&otilde;es Web'); ?>"><?php echo
$this->img('/media/images/go.gif', array('alt' => translate('Goweb | Web Design, Web Marketing, Solu&ccedil;&otilde;es Web'), 'border' =>0)); ?></a>
    </div>
<!--END FOOTER-->
</div>
<script type="text/javascript">
function replaceText(text){
    while(text.lastIndexOf("&") > 0){
        text = text.replace('&', '[i-Stats]');
    }
    return text;
}

var web_referrer = replaceText(document.referrer);
<!--
istat = new Image(1,1);
istat.src = "
.
.
.
.
```

Da análise da porção de código apresentado importa destacar os seguintes pontos:

– introdução de conteúdo providenciado por componentes:

```
<?php echo $this->component('/loja/mini_carrinho/'); ?>
<?php echo $this->component('/utilizadores/loginBlock/'); ?>
<?php echo $this->component('/cms/pesquisa/'); ?>
```

– introdução do conteúdo de uma view:

(A variável **\$output** contém o código HTML da view a apresentar. O respectivo render já foi feito).

```
<?php echo $output ?>
```

A partir de SM5 a plataforma assume, se nada for explicitamente dito em contrário, que a cada action de um controller corresponde uma view com o mesmo nome. É por essa razão que no método `fixedAction()` quando o parâmetro `banner_id` não corresponde ao id de um registo válido na tabela `banners` é invocado o método

```
$this->_helper->viewRenderer->setNoRender();
```

indicando à plataforma que não pretendemos esse comportamento por defeito. Neste método se o id do banner não é válido não pretendemos qualquer tipo de output.

## Bases de dados: codificação, alterações à estrutura, site offline

### *Codificação (Charset)*

Importante, após a criação de novo projecto, verificar se o charset da base de dados está de acordo com aquele definido no ficheiro `/config.ini`:

```
(...)  
database.charset = utf8  
(...)
```

A incoerência destes parâmetros com o charset da base de dados é responsável por diversos problemas na apresentação de conteúdos, de difícil resolução quando a base de dados já contém informação.

### *Alterações à estrutura*

Sempre que uma classe Model for alterada, seja por alteração, inserção ou remoção de campos, a forma mais segura de garantir que a estrutura das tabelas está de acordo com a definição contida nestas classes é a execução do método `Setup`, que deve ser acedido em backoffice, após login e tal como anteriormente referido, através do url:

`http://<url do site>/admin/<idioma>/setup`

### *Site offline para público (parâmetro 'ver')*

Sempre que haja necessidade de colocar um site offline (e esta necessidade apenas ocorre em sites em produção) é possível continuar a utilizar a aplicação referenciando no url explicitamente o ficheiro index.php com o parâmetro ver=1. A sua utilização é demonstrada nesta porção de código fonte extraída do ficheiro index.php (situado na raiz do site):

```
(...)  
if (is_file(CACHE_PATH . '/hide_site')) {  
    if ((array_key_exists('ver', $_GET)) && ($_GET['ver'] == '1')) {  
        setcookie('VER', '1', time()+8000, '/');  
    } elseif ($_COOKIE['VER'] != '1') {  
        require_once ROOT_PATH . '/library/Zend/Controller/Front.php';  
        setcookie('VER', '0', time()+8000, '/');  
        echo '<script>location="' . Zend_Controller_Front::getInstance()->getBaseUrl() . 'index.html';</script>';  
        exit;  
    }  
}  
(...)
```