MIOLO 2.0

Guia de Referência

Vilson Cristiano Gärtner
vilson@miolo.org.br

Ely Edison Matos
ely.matos@ufjf.edu.br

versão do documento: 1.1

16/05/2006

Sumário

1. MIOLO

class MIOLO

É a classe central do framework, com métodos que são utilizados pelas demais classes e pelos módulos.

function MIOLO

Método construtor da classe. Inicializa as variáveis de configuração definidas no arquivo miolo.conf, inclui outros arquivos (classes) necessários ao framework e instancia as principais classes do kernel.

function Assert($cond,$msg='',$goto='')

Manipulador de asserções. Se a condição $cond de uma asserção falha (é false), uma mensagem de erro é apresentada e a execução do script é suspensa. O parâmetro $goto é uma URL; se informado é adicionado um botão com ação para o respectivo endereço.

function CheckAccess($trans, $Access, $deny)

Verifica se o usuário corrente (objeto Login) tem o direito $access na transação $trans. $deny determina se a execução do script deve ser interrompida ou não.

function Confirmation($msg,$gotoOK='',$gotoCancel='')

Cria uma tela apresentando a mensagem $msg. Os parâmetros $gotoOK e $gotoCancel são URLs; se forem informados são adicionados dois botões (Ok e Cancel) com a ação para os respectivos endereços.

function Dump($var,$file=false,$line=false,$info=false)

Gera um dump de "var" na tela. __FILE__ e __LINE__ são valores globais que indicam o arquivo e linha dessa execução. Info é alguma informação adicional.

function Error($msg='',$goto='', $caption='')

Cria uma tela apresentando a mensagem de erro $msg. O parâmetro $goto é uma URL; se for informado é adicionado um botão com a ação para esse endereço.

function GetAbsolutePath($rel=null)

Compõe um path para um arquivo a partir do diretório do MIOLO.
$arquivo = $MIOLO->GetAbsolutePath('/dir/arquivo.txt');

function GetAbsoluteURL($rel,$module=null)

Compõe uma URL, a partir do home-url definido no miolo.conf, e retorna o endereço html do parâmetro $rel. Se $module for informado, compõe a URL com o nome do módulo, caso contrário, compõe utilizando o diretório de instalação do miolo.

function GetActionURL($module='', $action='NONE', $item='', $args=null , $dispatch=null)

Utilizado para gerar uma URL no formato do MIOLO (v. "Arquitetura"), para execução de um handler no módulo especificado por $module.
$acao=$MIOLO->GetActionURL('modulo', 'arquivo')

function & GetAuth()

Retorna uma referência para o objeto Auth.

function GetBusiness($module,$name='main')

Utilizado para instanciar a classe do arquivo $name no diretório classes do módulo $module. Para evitar conflitos de nomes, a classe derivada de Business deve ter o nome no formato <Business><Módulo><Classe>.
$business = $MIOLO->GetBusiness('módulo, 'arquivo')

function GetDatabase($conf=null,$user=null,$pass=null)

Este método é utilizado para criar uma conexão com a base de dados especificada no parâmetro $conf. A configuração da base deve ter sido previamente criada no arquivo de configuração do MIOLO (miolo.conf). Retorna uma referência para um objeto Database.
$MIOLO->GetDatabase('NomeBase')

function GetModulePath($module,$file)

Retorna um path indicando a localização do arquivo $file do módulo $module.

function & GetPage()

Retorna uma referência para o objeto Page.

function & GetPerms()

Retorna uma referência para o objeto Perms.

function & GetTheme($themeid='',$layout='')

Retorna uma referencia para o tema (um objeto Theme) a ser usado, indicando o nome do tema ($themeid) e o layout a ser usado ($layout)

$theme =& $MIOLO->GetTheme();

function GetThemeURL($rel,$name=null,$default=null)

Retorna o endereço web do tema atualmente utilizado.

function GetUI()

Instancia a classe UI, permitindo acesso às funções de Interface de Usuário

function Information($msg,$goto='')

Cria uma tela apresentando a mensagem $msg. O parâmetro $goto é uma URL; se for informado é adicionado um botão com a ação para esse endereço.

function InvokeHandler($module,$name)

Cria dinamicamente uma classe interna ao script, e coloca na definição desta classe o conteúdo do arquivo $name que está no diretório handlers do módulo indicado por $module. Instancia um objeto desta classe, causando a execução dos comandos presentes no arquivo $name.

function IsLogging()

Determina se o processo de loggin está ativado.

function ListFiles($dir,$type='d')

Retorna uma lista dos arquivos do diretório $dir. O parâmetro $type indica se nomes de diretórios deverão ser incluídos ou não na lista.

function Prompt($prompt,$halt=true)

Método chamado pelos demais métodos de mensagens e avisos. O parâmetro $prompt é um objeto da classe Prompt e $halt indica se o processamento do script deve ou não ser interrompido.

function Question($msg,$gotoYes='',$gotoNo='')

Cria uma tela apresentando a mensagem $msg. Os parâmetros $gotoYes e $gotoNo são URLs; se forem informados são adicionados dois botões (Yes e No) com a ação para os respectivos endereços.

function _Request($vars, $from='ALL')

Obtém o valor de uma variável (ou de várias, em um array) a partir dos arrays globals do PHP ($_REQUEST, $_POST, $_GET, $_SESSION, global).

function UnScramble($text)

Criptografadecriptografa um $texto, usando como chave a sessionid corrente.

function SetLog($logname)

Determina o path para o arquivo de log.

function SetTheme(&$theme)

Utilizado para definir o tema, "theme" é um objeto da classe Theme. Não deve ser confundido com o método SetIdTheme da classe Theme.

function Uses($name,$module=null)

Similar a função include do PHP. Esse método inclui no script o arquivo $name do módulo $module.

function UsesBusiness($module, $name='main')

Inclui o código referente a uma classe Business, sem instanciá-la.

function UsesExtensions()

Esse método inclui no script todos os arquivos do diretório <home>/extensions (que possuem classes que adicionam funcionalidade ao framework).

class Context

Esta classe é usada para ober os dados passados na url do handler.php ($module, $action, $item). Também pode ser usada para compor uma url.

function ComposeURL()

Retorna uma url com os valores atuais dos atributos $module, $action e $item.

function Create($module,$action,$item)
Atribui os valores dos parâmetros aos atributos do objeto.

function GetAction($index=0)
Obtém o valor da string $action na posição indicada por $index.

function PushAction($a)
Acrescenta a string $a no atributo $action.

function ShiftAction()
Obtém o próximo valor na seqüência da string $action.

2. Controles

Class UI

A classe UI atua como um ponto central para se obter as instâncias dos objetos referentes a interface com o usuário (forms, menus, imagens etc)

function Alert($msg,$info,$href='')
Emite uma mensagem de alerta.

function CreateForm($title='')
Instancia um formulário (class Form).

function GetForm($module,$name,$data=null, $dir=null)
Instancia um formulário definido pelo usuário. O arquivo com a classe do formulário deve estar localizado em <home_miolo>/$module/forms/<$dir>/$name.class.

function GetListing($module,$name,$data=null, $dir=null)
Instancia um grid (listing) definido pelo usuário. O arquivo com a classe do grid deve estar localizado em <home_miolo>/$module/listings/<$dir>/$name.class.

function GetMenu($module='main',$name='UIMainMenu')
Instancia um menu definido pelo usuário. O arquivo com a classe do menu deve estar localizado em <home_miolo>/$module/menus/$name.class.

function GetReport($module,$name,$data=null, $dir=null)
Instancia um relatório definido pelo usuário. O arquivo com a classe do relatório deve estar localizado em <home_miolo>/$module/reports/<$dir>/$name.class.

function GetImage($module,$nome)
Retorna a URL de uma imagem.

class Theme

O objeto Theme encapsula o layout da página HTML a ser exibida. Os principais elementos HTML são encapsulados na classe e o "desenho" da página é feito através de seus métodos. Além disso a classe Theme permite instanciar os objetos que compõe o tema (Barra de navegação – navbar, Menus, área de conteúdo – content, Barra de status – statusbar).

function & GetNavigationBar()
Retorna uma referência para a barra de navegação (navbar).

function SetNavigationBar(&$bar)
Atribui a barra de navegação.

function ClearMenus()
Elimina os menus da página.

function & GetMenus()
Retorna o array de menus da página.

function & GetMainMenu()
Retorna uma referência para o menu principal.

function & GetMenu($name)
Retorna uma referência para o menu $name, ou o cria se ele não existir.

function & GetStatusBar()
Retorna uma referência para a barra de status.

function SetStatusBar(&$bar)
Atribui a barra de status.

function ClearContent()
Inicializa a área de conteúdo.

function & GetContent()
Retorna uma referência para a área de conteúdo.

function SetContent(&$content)
Atribui o elemento para a area de conteúdo.

function InsertContent(&$element)
Insere o elemento no início da área de conteúdo.

function AppendContent(&$element)
Insere o elemento no final da área de conteúdo.

function GetId()
Retorna o Id do tema em uso.

function SetId($idtheme)
Atribui o Id do tema.

function GetLayout()
Retorna o layout do tema em uso.

function SetLayout($layout)
Atribui o layout do tema.

function AddElement(&$element, $location='content')
Adiciona o elemento a uma das áreas do tema (‘menu’ ou ‘content’).

function & UseComponent($name)
Instancia um componente externo ao Miolo.

class Page

O objeto Page encapsula a funcionalidade do Webform, representando a página HTML que é criada quando uma URL é chamada. Cada página possui um único formulário HTML, ainda que possa possuir vários objetos Form.

function Page()
Instancia um objeto Page, inicializando os atributos da página , a funcionalidade dos temas e o objeto State (que possui o conteúdo das variáveis que mantém estado entre round-trips).

function IsPostBack()
Determina se a página está sendo chamada a primeira vez ou se é um postback (devido a submissão de um formulário).

function SetPostBack($postback)
Atribui um valor ao atributo postback.

function SetAction($action)
Indica a URL que será o action do Webform.

function SetEnctype($enctype)
Indica o tipo de codificação do Webform.

function AddStyle($url)
Adiciona um link para uma folha de estilo CSS.

function AddScript($url)
Adiciona um link para um arquivo de script.

function AddMeta($name,$content)
Adiciona uma tag META NAME do HTML.

function AddHttpEquiv($name,$content)
Adiciona uma tag META HTPP-EQUIV do HTML.

function & GetStyles()
Retorna o array com as URL de folhas de estilo CSS.

function & GetScripts()
Retorna o array com as URL de scripts.

function & GetMetas()
Retorna o array com as tags META.

function & GetTitle()
Retorna o título da página.

function SetTitle(&$title)
Atribui o título da página.

function Request($variable)
Retorna o valor de uma variável submetida, consultando $MIOLO->_Request e array de variáveis de estado.

function LoadViewState()
Carrega as variáveis de estado (que mantém os valores através de round-trips).

function SaveViewState()
Salva o estado das variáveis de estado.

function Transfer($module,$action,$item='')
Transfere a execução para outro handler, através da chamada a $MIOLO->InvokeHandler.

function GoTo($url)

function Redirect($url)
Transfere a execução para outro script, indicado através de $url, usando o campo Location do protocolo HTTP.

function EventHandler(&$subject)
Executa o evento associado à classe $subject.

function Generate()
Dispara o processo de renderização da página HTML.

function Handler()
É o método principal da classe, responsável por carregar as definições do tema, executar o handler necessário e gerar a página HTML.

class Form

A classe Form é o principal mecanismo de interface com o usuário. Os formulários criados pelos desenvolvedores vão, via de regra, estender a class Form para implementar um Webform. Embora não corresponda diretamente a um formulário HTML, é através da classe Form que o controles da página são definidos.
A classe mantém um atributo $fields, que é um array de objetos representando os controles do Form. A classe também acrescenta um prefixo "frm_" ao nome dos campos, antes da renderização.

function Form($title='',$action='')
Instancia um Form. Executa o método CreateFields (geralmente definido em uma classe-filha) e faz a carga inicial do array $fields, chamando o método GetFormFields. É também executado, caso exista, o método OnLoad.

function AddValidator($validator)
Adiciona um controle Validator ao Form.

function SetValidators($validators)
Adiciona um conjunto de contrioles Validator ao Form.

function GetName()
Retorna o nome do Form.

function & GetTitle()
Retorna o título do Form.

function SetTitle($title)
Indica o título do Form.

function & GetFooter()
Retorna o controle usado com rodapé do Form.

function SetFooter($footer)
Indica o controle a ser usado como rodapé do Form.

function GetFormFields()
Atribui valores ao array $fields, com base nas variáveis submetidas via POST ou GET.

function /* static */ GetFormValue($name,$value=null)
Esta função estática retorna o valor do campo $name. Busca no array $fields, $_POST, $GET.

function SetFormValue($name,$value)
Indica um valor para o campo $name.

function OnSubmit($jscode)
Indica um código Javascript para ser executado na submissão do Form.

function AddJsCode($jscode)
Indica um código Javascript a ser acrescentado no Form.

function SetAction($action)
Indica um a URL como action do Form.

function SetHelp($href)
Indica uma URL como página de help.

function SetFields(&$fields)
Indica os valores para o array $fields. No registro de cada controle, é acrescentado o prefixo "frm_" ao nome do controle e criado um atributo de mesmo nome, que representará o controle, para a classe.

function AddField(&$field,$hint=false)
Acrescenta um controle $field no array $fields.

function AddButton(&$btn)
Acrescenta um botão no array $buttons.

function SetButtons(&$btn)
Indica os valores para o array $buttons. Os botões não possuem o prefixo "frm_".

function SetButtonLabel($index,$label)
Modifica o label do botão indicado por $index, no array $buttons.

function ShowReturn($state)
Indica se deve ser exibido um botão Return.

function ShowReset($state)
Indica se deve ser exibido um botão Reset.

function GetShowHints()
Retorna se os hints dos campos serão exibidos.

function ShowHints($state)
Indica se os hints dos campos serão exibidos ou não.

function GetFieldList()
Retorna uma lista dos campos do Form que podem possuir valor, omitindo os campos puramente decorativos.

function ValidateAll($assert=true)
Verifica se todos os campos possuem um valor, podendo interromper ou não a execução, dependendo de $assert.

function Validate($required,$assert=true)
Verifica se os campos em $required possuem um valor, podendo interromper ou não a execução, dependnedo de $assert.

function /* static */ AddError($err)
Adiciona um erro à lista de erros do Form.

function HasErrors()

Retorna se o Form possui algum erro.

function CollectInput($data)

Recebe o objeto $data com parâmetro e cria como nome de atributos de $data os nomes dos campos do Form e como valores destes atributos os valores dos campos. Geralmente chamado por GetData().

function GetData()

Executa CollectInput para obter um objeto da class FormData. Pode ser sobreposto pelas classes-filhas, para tratamentos específicos nos dados de entrada.

function SetData($data)

Percorre os atributos do objeto $data e, para os campos do Form que têm o mesmo nome, atribui o valor daquele atributo.

function GetFieldValue($name,$value=false)

Usado depois que os campos já estão registrados. Executa o método GetValue() do controle correspondente ao campo.

function SetFieldValue($name,$value)

Usado depois que os campos já estão registrados. Executa o método SetValue() do controle correspondente ao campo.

function SetFieldValidator($name,$value)

Atribui o controle Validator $value ao campo indicado por $name.

function & GetField($name)

Retorna o controle (objeto) referente ao campo $name.

function & GetButton($name)

Retorna o botão (objeto) referente ao botão $name.

function SetFieldAttr($name,$attr,$value)

Atribui o valor $value ao atributo $attr do controle $name.

function GetFieldAttr($name,$attr, $index=null)

Retorna o valor do atributo $attr do controle $name. Se o atributo for um array usa o índice $index.

function SetButtonAttr($name,$attr,$value)

Atribui o valor $value ao atributo $attr do botão $name.

Server Controls

As diversas classes de controles encapsulam a criação e renderização de controles html, usados principalmente em formulários, bem como servem de base para criação de controles mais complexos, formados através de composição.

function Separator($text=null)

$text (string)

Exibe uma linha e opcionalmente o texto $text.

Exemplo: new Separator();

function Text($name, $text='')

$name (string)

$text (string)

Exibe o texto $text.

Exemplo: $exemplo = new Text('txtNome',$nome);

function TextLabel($name, $text=null,$label='')

$name (string)

$text (string)

$label (string)

Exibe o texto $text com o label $label.

Exemplo: $exemplo = new TextLabel('txtNome',$nome,'Nome');

function HyperLink($name=null,$label=null,$href=null,$text=null)
$name (string)
$label (string)
$href (string)
$text (string)

Exibe um hyperlink para $href com o texto $text ou $label.
Exemplo: $link = new Hyperlink('lnkVoltar',null,$MIOLO->GetActionURL('common','main'), 'Principal');

function ImageForm($name=null,$label=null,$location=null, $attrs=null)
$name (string)
$label (string)
$location (string)
$attrs (array)

Exibe uma imagem. $location indica a URL do arquivo de imagem, $label é usado como atributo "alt" e $attrs é um array com atributos a serem usados na renderização (no formato $attr[atributo] = valor).
Exemplo: $img = new ImageForm('imgDelete','Delete','images/delete.gif',array('border'=>'0'));

function FormButton($name='', $label='',$action=null)
$name (string)
$label (string)
$action (string)

Exibe um botão de ação. $action pode ter um dos seguintes formatos:

- null ou 'submit': gera um evento <$name>:click no formulário
- 'reset': funciona como um botão 'reset' do html
- 'print': abre uma janela para impressão da página corrente
- 'report': gera um evento <$name>:click no formulário e abre uma janela para exibição do relatório em pdf
- 'http://...': desvia imediatamente para a página indicada pela URL

Exemplos:
$btnPost = new FormButton('btnPost','Enviar'); // submit
$btnPrint = new FormButton('btnPrint','Imprimir','print');
$btnMain = new FormButton('btnMain','Principal',$MIOLO->GetActionURL('common','main')); // http://…

function LinkButton($name='', $label='',$action='')
$name (string)
$label (string)
$action (string)

Exibe um hyperlink que aponta para a página indicada por $action e gera um evento <$name>:click
Exemplos:
$btnPost = new LinkButton('btnPost','Enviar'); // submit na mesma página
$btnGoto = new LinkButton('btnGoto','Ir para Principal', $MIOLO->GetActionURL('common','main')); // submit em outra página

function ImageButton($name='', $label='',$action='',$location='')
$name (string)
$label (string)
$action (string)
$location (string)

Exibe uma imagem como um hyperlink que aponta para a página indicada por $action e gera um evento <$name>:click. $location indica a URL da imagem a ser renderizada.
Exemplos:
$btnPost = new ImageButton('btnPost','Enviar','submit','images/post.gif');

function TextField($name,$value='',$label='', $size=10,$hint='',$validator=null)
$name (string)
$value (string)
$label (string)
$size (string)
$hint (string)

$validator (objeto validator)

Exibe um campo para entrada de texto.

Exemplo:

$exemplo = new TextField('edtNome',$nome,'Nome Completo',40);

function PasswordField($name,$value='',$label='', $size=20, $hint='', $validator=null)

$name (string)
$value (string)
$label (string)
$size (string)
$hint (string)
$validator (objeto validator)

Exibe um campo para entrada de senha (mostra '*' no lugar dos caracteres).

Exemplo:

$pass = new PasswordField('edtPass',$password,'Senha',20);

function HiddenField($name='',$value='')

$name (string)
$value (string)

Renderiza um campo oculto no formulário.

Exemplo:

$idCliente = new HiddenField('hidId',$id);

function MultiLineField($name='',$value='',$label='', $size=20, $rows=1, $cols=20, $hint='',$validator=null)

$name (string)
$value (string)
$label (string)
$size (string)
$rows (integer)
$cols (integer)
$hint (string)
$validator (objeto validator)

Exibe um campo para entrada de texto com múltiplas linhas. $rows indica o número de linhas e $cols o número de colunas.

Exemplo:

$edtObs = new MultiLineField('edtObs',$obs,'Observação',20,5,20);

function LookupTextField($name='',$value='',$label='', $size=10, $hint='', $validator=null, $related='', module='',$item='', $event='', $autocomplete=false)

$name (string)
$value (string)
$label (string)
$hint (string)
$validator (objeto validator)
$related (string)
$module (string)
$item(string)
$event (string)
$autocomplete (boolean)

Exibe um campo para entrada de texto, que permite fazer pesquisas no banco de dados. A descrição do funcionamento é feita no capítulo Lookup.

function MultiTextField2($name='',$value=null,$label='',$fields='', $width=200, $buttons=false, $layout='vertical',$hint='')

$name (string)
$value (string)
$label (string)

$fields (array)
$width (integer)
$buttons (Boolean)
$layout (string)
$hint (string)

MultiTextField é um controle composto, que permite criar uma lista de dados a partir de campos TextField ou Selection. O controle provê botões para adicionar, modificar e remover dados da lista, e opcionalmente botões para percorrer a lista. Os campos são indicados através do array $fields. Cada elemento de $fields é um array com o seguinte formato:

array($id,$label,$name,[$options_array]), onde

$id: identificador do campo
$label: texto a ser exibido como label do campo
$name: texto opcional a ser exibido junto ao label
$options_array: array de opções, no campo de um campo tipo Selection; se não for fornecido, o campo é renderizado como um TextField

A propriedade $info permite informar um label a ser exibido para a lista de dados.

São fornecidos dois métodos para facilitar o tratamento da lista de dados: GetCodeValue e SetCodeValue.

$value = $obj->GetCodeValue()
O método retorna em $value um array onde cada elemento é uma linha da lista de dados. Cada linha, por sua vez é também representada como um array, onde cada elemento é o valor do campo correspondente.

$obj->SetCodeValue()
O método define o conteúdo da lista de dados, baseado no array $value (onde cada elemento é uma linha da lista de dados e cada linha, por sua vez é um array, onde cada elemento é o valor do campo correspondente).

function FileField($name='',$value='',$label='',$size=40,$hint='')
$name (string)
$value (string)
$label (string)
$hint (string)

Exibe um controle para upload de arquivos.

function CalendarField($name='',$value='',$label='',$size=10,$hint='')
$name (string)
$value (string)
$label (string)
$hint (string)

Exibe um entrada de texto para receber uma data. Permite a execução de um calendário em javascript.

function CheckBox($name='',$value='',$label='', $checked=false, $text=null, $hint='')
$name (string)
$value (string)
$label (string)
$hint (string)

Exibe um controle do tipo checkbox. $checked indica se o controle está inicialmente marcado ou não e $text define o texto a ser exibido. $value é o valor a ser enviado no post.

function RadionButton($name='',$value='',$label='', $checked=false, $text=null, $hint='')
$name (string)
$value (string)
$label (string)
$hint (string)

Exibe um controle do tipo radiobutton. $checked indica se o controle está inicialmente marcado ou não e $text define o texto a ser exibido. $value é o valor a ser enviado no post.

function Option($name,$value=null,$label,$checked=false)
$name (string)
$value (string)
$label (string)
$checked (boolean)

Option é uma classe auxiliar, que permite a definição de itens para os controles CheckBoxGroup e RadioButtonGroup.

function CheckBoxGroup($name='',$label='',$options='',$hint='')
$name (string)
$label (string)
$options (array)
$hint (string)

Exibe um grupo de controles do tipo checkbox. $options é um array, onde cada elemento é um objeto do tipo Option.

function RadioButtonGroup($name='',$label='',$options='', $default=false, $layout='horizontal', $hint='')
$name (string)
$label (string)
$options (array)
$default (boolen)
$layout (string) : 'horizontal' ou ' vertical'
$hint (string)

Exibe um grupo de controles do tipo radiobutton. $options é um array, onde cada elemento é um objeto do tipo Option. $default indica qual opção vai estar selecionada por default (deve ser o valor de um dos elementos de $options). $layout indica como os itens serão dispostos no formulário.

function LinkButtonGroup($name='',$label='',$options='', $layout='horizontal', $break=' ')
$name (string)
$label (string)
$options (array)
$layout (string) : 'horizontal' ou ' vertical'
$break (string)

Exibe um grupo de controles do tipo linkbutton. $options é um array, onde cada elemento é um objeto do tipo Option. $layout indica como os botões serão dispostos no formulário. $break indica como os itens deverão ser separados.

function Selection($name='',$value,$label='', $options=Array('Não','Sim'), $showValues=false, $hint='')
$name (string)
$value (string)
$label (string)
$options (array)
$showValues (boolean)
$hint (string)

Exibe uma caixa de seleção. $options é um array onde cada elemento tem o formato
$options[value] = text
onde 'value' indica o valor a ser enviado no post, quando o item for selecionado, e 'text' indica o texto a ser apresentado na caixa de seleção para aquele item. $showValues indica se o valor do item deve ser exibido ou não.

function MultiSelection($name='', $values=Array('1','2','3'), $label=' ', $options=Array('Option1','Option2','Option3'), $showValues=false, $hint='', $size='')
$name (string)
$values (array)
$label (string)

$options (array)
$showValues (boolean)
$hint (string)
$size (integer)

Exibe uma caixa de seleção múltipla. $values é um array onde cada elemento é um valor de um item da caixa de seleção.
$options é um array com uma ou duas dimensões. Se for unidimensional, onde cada elemento tem o formato
$options[value] = text
onde 'value' indica o valor a ser enviado no post, quando o item for selecionado, e 'text' indica o texto a ser apresentado na caixa de seleção para aquele item. Se $options for bidimensional, cada elemento tem o formato
$options[] = array(value, text)
$showValues indica se o valor do item deve ser exibido ou não. No post, o controle retorna um array onde cada elemento é um item que foi selecionado no formulário.

function ComboBox($name='',$value='',$label='',$options='',$hint='')
$name (string)
$value (string)
$label (string)
$options (array)
$hint (string)

Exibe uma caixa de texto associada a uma caixa de seleção. $options é um array onde cada elemento tem o formato
$options[value] = text
onde 'value' indica o valor a ser enviado no post, quando o item for selecionado, e 'text' indica o texto a ser apresentado na caixa de seleção para aquele item. Quando um item é selecionado, seu valor é colocado automaticamente na caixa de texto para ser enviado no post.

Lookup

Objetivo: Abrir uma janela no browser, onde o usuário pode informar argumentos de pesquisa no banco de dados.

Processamento:

- Definição de um campo LookupTextField no formulário.
  Ex: new LookupTextField('lkpNome','','Nome',80)

- Definição das propriedades do campo LookupTextField:

  Module e **Item**: define o módulo onde está a função que implementa a pesquisa. Todo módulo que ofereça pesquisa, deve possuir um arquivo <miolo>/<module>/classes/lookup.class, implementando a classe "Business<module>Lookup". Nesta classe deve haver um método para cada tipo de pesquisa que pode ser feita. Cada método recebe o nome "Lookup<item>" (ex. LookupPessoa)
  Event: que evento será disparado após a seleção do item pesquisado. Pode ser usado o valor "filler", se o objetivo é apenas preencher os campos do formulário.
  Related: lista dos campos que deverão ser preenchidos, caso o evento seja "filler".

- A renderização do campo LookupTextField gera um campo texto com a imagem de uma lupa ao lado. O usuário pode preencher o campo texto com o valor inicial a ser pesquisado e então clicar na imagem.

- No evento OnClick da imagem, é criado um objeto javascript do tipo LookupContext, que reúne as propriedades do campo LookupTextField. É então executada a função MIOLO_Lookup (do arquivo common.js), sendo passado o objeto LookupContext.

- A função javascript monta uma URL para uma página lookup.php e abre a página em uma nova janela do browser.

- A página lookup.php recebe os parâmetros relativos à pesquisa e instancia um objeto da classe Lookup. O objetivo desta classe é simplesmente agregar todas propriedades do formulário de pesquisa. A página então cria um novo formulário (classe Form), que é composto por caixas de texto para os argumentos de pesquisa (filtros) e um datagrid que exibe os resultados.

- A página lookup.php também é responsável por instanciar o objeto "Business<module>Lookup" e executar o método "Lookup<item>", passando como parâmetro o objeto Lookup criado anteriormente. O método "Lookup<item>" é que efetivamente define quais os campos para filtrar a pesquisa, qual a coluna chave do datagrid, e qual a query será executada no banco de dados. O argumento inicial da pesquisa pode ser acessado através de $lookup->filterValue.

- Através da coluna de ação do datagrid é possível selecionar um registro específico. Quando um registro é selecionado, é executada a função javascript MIOLO_Deliver (do arquivo common.js). Esta função recebe como parâmetros o nome do campo, o número da coluna chave (a coluna do datagrid que contém o valor da chave do registro exibido) e uma lista de valores (que são os valores exibidos na linha do datagrid). Caso o evento seja "filler", é feito o preenchimento dos campos referenciados na lista "related" com os valores das colunas do datagrid (a atribuição é feita de acordo com a posição na lista). Caso seja um outro evento, este é disparado através da função _doPostBack do formulário, passando o valor do campo chave como parâmetro.

Datagrid

Objetivo: Apresentar um grid para exibição do resultado de consultas ao banco de dados, em objetos Query, permitindo paginação, ordenação e filtragem.

Criação

$datagrid = new DataGrid($query, $columns, $href_datagrid,$pagelength);

$query: objeto Query, representando uma consulta feita ao banco de dados
$columns: array com objetos representando as colunas a serem exibidas
$href_datagrig: URL representando a página onde o datagrid está inserido
$pagelength: quantidade de linhas a serem exibidas por página (0, se não for feita paginação)

Colunas

Os dados podem ser exibidos em 3 tipos de colunas: dados (DataGridColumn), links (DataGridHyperlink) e controles (DataGridControl).

Class DataGridColumn

DataGridColumn($field, $title='', $align='left', $nowrap=false, $width=0, $visible=true, $options=null, $order=false, $filter=false)

$field: nome do campo a ser exibido nesta coluna. Este nome deve aparecer na lista de campos do objeto Query. Pode estar no formato "field" ou "table.field".
$title: título da coluna
$align: define o alinhamento do campo (rigth, center, left). Por default, números são alinhados à direira.
$nowrap: define se o dado pode ser quebrado em mais de uma linha ou não (true/false)
$width: define a largura da coluna
$visible: define a coluna deve ser exibida ou não (true/false)
$options: array de opções no formato ("opção" => "texto"), "texto" é exibido quando o valor do campo for igual a uma das opções do array.
$order: expressão usada para ordenação do datagrid (string).
$filter: define se esta coluna pode ser usada para filtrar a exibição de dados no datagrid (true/false).

function SetAttr($attr, $value)
$attr (string)
$value (mixed)

Define o valor do atributo $attr para $value.

Class DataGridHyperlink

DataGridHyperlink($field, $title='', $href, $width=0,$visible=true,$options=null, $order=false, $filter=false)

$field: nome do campo a ser exibido nesta coluna. Este nome deve aparecer na lista de campos do objeto Query. Pode estar no formato "field" ou "table.field".
$title: título da coluna
$href: URL do hyperlink. Nesta URL o token #?# é substituído pelo valor do campo corrente e o token #n# é substituído pelo valor do campo "n" da query.
$width: define a largura da coluna
$visible: define a coluna deve ser exibida ou não (true/false)
$options: array de opções no formato ("opção" => "texto"), "texto" é exibido quando o valor do campo for igual a uma das opções do array.
$order: expressão usada para ordenação do datagrid (string).
$filter: define se esta coluna pode ser usada para filtrar a exibição de dados no datagrid (true/false).

Class DataGridControl

DataGridControl(&$control, $name, $title='',$align=null,$nowrap=false,$width=0,$visible=true)

$control: referência ao objeto base representando o controle a ser exibido. É criado um novo objeto para cada linha exibida.
$name: nome dos objetos a serem criados em cada linha. Neste nome pode ser usado o token %n%, onde "n" é o número do campo da query.
$title: título da coluna
$align: define o alinhamento do campo (rigth, center, left).
$nowrap: define se o dado pode ser quebrado em mais de uma linha ou não (true/false)
$width: define a largura da coluna
$visible: define a coluna deve ser exibida ou não (true/false)

Ações

A primeira coluna do datagrid pode ser usada para ações a serem aplicadas às linhas. DataGridAction é a classe base para as ações, sendo acessada através dos métodos AddActionSelect, AddActionIcon e AddActionText, do datagrid. As funções Enable() e Disable(), da classe DataGridAction, são usadapara para habilitar ou desabilitar uma ação.

Métodos

function SetTitle($title)
$title (string)
Define o título do datagrid.

function SetFilter($filter)
$filter (string)
Define uma expressão do tipo sql-where para filtragem do datagrid.

function SetLinkType($linktype)
$linktype (string): 'hyperlink' ou 'linkbutton'
Define a forma de uso dos links do datagrid. 'Linkbutton' força a utilização do método Post, permitindo que sejam enviados os conteúdos dos controles apresentados na página.

function SetButtons($aButtons)
$aButtons (array): array de botões de controle (Button)
Define botões a serem apresentados no rodapé.

function SetControls($aControls)
$aControls (array): array de controles

Define controles a serem apresentados no rodapé.

function SetWidth($width)

$width (string) – ex: '95%'

Define a largura da tabela onde o datagrid será apresentado.

function SetRowMethod($class, $method)

$class (string): nome de uma classe

$method (string): nome de um método da classe $class

Define o método a ser chamado antes da criação de cada linha do datagrid, permitindo que sejam feitos teste e alterações no conteúdo das colunas. O método a ser chamada tem a seguinte assinatura:

<method_name>($row,$actions,$columns, $fieldpos);

$row (array): array com o conteúdo da linha a ser renderizada. Cada elemento de $row é um campo da consulta feita no objeto Query.

$actions (array): array com as ações que foram definidas para este datagrid.

$columns (array): array com os objetos que representam as colunas do datagrid.

$fieldpos (array): array com a posição em $row de cada campo da consulta feita no objeto Query.

function HeaderLink($id, $label, $href)

$id (integer | string) : índice do array de links

$label ($string)

$href (string)

Define um controle LinkButton a ser colocado no cabeçalho do datagrid.

function SetColumnAttr($field,$attr,$value)

$field (string)

$attr (string)

$value (mixed)

Define ou altera um atributo da coluna referente ao campo $field.

function AddActionSelect($index=0)

$index (integer)

Adiciona uma ação do tipo 'select' (um controle do tipo CheckBox) na coluna de ações. O controle recebe o nome "selectRow<n>", onde "n" é o número da linha do controle, e o valor $row[$index].

function AddActionIcon($alt,$icon,$href,$index=0)

$alt (string) : texto alternativo para a imagem

$icon (string | array)

$href (string)

$index (integer)

Adiciona uma ação do tipo "image" na coluna de ações. $icon indica o nome do arquivo de imagem, ou um array com dois arquivos (quando a ação pode estar habilitada ou desabilitada). $href indica a URL a ser executada (como um hyperlink ou um linkbutton, dependendo do valor da propriedade $linktype do datagrid). Nesta URL o token "$id" é substituído por $row[$index], o token "%n%" é substituído por urlencode($row[$n]) e o token "#n#" é substituído por $row[$n].

function AddActionText($alt,$text,$href,$index=0)

$alt (string) : texto alternativo

$text (string)

$href (string)

$index (integer)

Adiciona uma ação do tipo "text" na coluna de ações. $text indica a string a ser exibida na coluna. Em $text o token "%n%" é substituído por $row[$n]. $href indica a URL a ser executada (como um hyperlink ou um linkbutton, dependendo do valor da propriedade $linktype do datagrid). Nesta URL o token "$id" é substituído por $row[$index], o token "%n%" é substituído por urlencode($row[$n]) e o token "#n#" é substituído por $row[$n].

function AddActionUpdate($href)

$href (string)

Adiciona uma ação do tipo 'image', com $alt = 'Editar' e $icon = 'button_edit.png'. $href indica a URL a ser executada (como um hyperlink ou um linkbutton, dependendo do valor da propriedade $linktype do datagrid). Nesta URL o token "$id" é substituído por $row[$index], o token "%n%" é substituído por urlencode($row[$n]) e o token "#n#" é substituído por $row[$n].

function AddActionDelete($href)

$href (string)

Adiciona uma ação do tipo 'image', com $alt = 'Excluirr' e $icon = 'button_drop.png'. $href indica a URL a ser executada (como um hyperlink ou um linkbutton, dependendo do valor da propriedade $linktype do datagrid). Nesta URL o token "$id" é substituído por $row[$index], o token "%n%" é substituído por urlencode($row[$n]) e o token "#n#" é substituído por $row[$n].

function ShowID($state)

$state (Boolean)

Usado com $options. Indica se o índice do array $options deve ser exibido na coluna ou não.

function HasErrors()

Indica se há erros no datagrid.

Tabelas

class TableRaw

function TableRaw($title='', $array, $colTitle=null)

Renderiza um array como uma tabela HTML. $title é o título da tabela, $array contém o conteúdo a ser exibido e $colTitle é um array com o título de cada coluna.

HTML Controls

-- incluir --

Theme Controls

Menus

Os menus do sistemas estão baseados na classe ThemeMenu, que por sua vez estende a classe ThemeControl.

class ThemeMenu

function ThemeMenu($title,$base='',$home='main')

Instancia um objeto Menu.

function Clear()

Inicializa o menu, removendo todas as entradas.

function SetTitle($title)

Atribui o título do menu.

function SetStyle($style)

Atribui o estilo do menu.

function SetBase($base)

Atribui a base do menu.

function AddLink($label,$link='#',$target='_self')

Adiciona um link ao menu.

function AddHyperLink($hyperlink)

Adiciona um link ao menu, com base em um controle Hyperlink.

function AddOption($label,$module='main',$action='',$item='')

Adiciona uma entrada ao menu.

function AddUserOption($transaction, $access,$label,$module='main',$action='',$item='')

Adiciona uma entrada ao menu, dependendo dos direitos do usuário.

function AddSeparator($name=null)

Adiciona um separador ao menu.

function AddMenu($label,$options)

Adiciona um sub-menu.

Validators

Validators são objetos que fazem a validação da entrada de dados do usuário. São geralmente renderizados como rotinas Javascript executadas no lado do cliente. A maioria dos Validator recebe três parâmetros:

- $field: nome do controle HTML a ser validado
- $label: nome "amigável" do campo, para aparecer na mensagem ao usuário
- $type: tipo de validação; pode ser 'required' (o usuário deve fornecer um valor) ou 'optional' (o valor é opcional) ou 'ignore' (se a validação puder ser ignorada, em certas situações).

class RequiredValidator: function RequiredValidator($field,$label='')
O usuário deve fornecer um valor para o campo.

class MASKValidator: function MASKValidator($field,$label='',$mask,$type = 'ignore')
Valida a entrada do usuário segundo uma máscara. A máscara pode conter os caracteres '9', para forçar um digito, ou 'a', para forçar uma letra. Um caracter diferente destes, força a existência daquele caracter específico. O tamanho da máscara delimita o número máximo de caracteres que podem ser digitados.

class EmailValidator: function EMAILValidator($field,$label='',$type = 'optional')
Valida a entrada do usuário com um email.

class CEPValidator: function CEPValidator($field,$label='',$type = 'optional')
Valida a entrada do usuário com um CEP.

class PHONEValidator: function PHONEValidator($field,$label='',$type = 'optional')
Valida a entrada do usuário com um telefone – (xx)xxx-xxxx

class TIMEValidator: function TIMEValidator($field,$label='',$type = 'optional')
Valida a entrada do usuário com uma hora – xx:xx

class CPFValidator: function CPFValidator($field,$label='',$type = 'optional')
Valida a entrada do usuário com um CPF.

class CNPJValidator: function CNPJValidator($field,$label='',$type = 'optional')
Valida a entrada do usuário com um CNPJ.

class DATEDMYValidator: function DATEDMYValidator($field,$label='',$type = 'optional')
Valida a entrada do usuário com uma data no formato dd/mm/aaaa.

class DATEYMDValidator: function DATEYMDValidator($field,$label='',$type = 'optional')
Valida a entrada do usuário com uma data no formato aaaa/mm/dd.

class CompareValidator: function CompareValidator($field,$label='', $operator, $value, $datatype='s', $type = 'optional')
Valida a entrada do usuário comparando com o valor fornecido em $value. $operator indica qual a operação de comparação ('==','>','<','>=','<=','!='). $datatype indica se o valor é string ('s') ou inteiro ('i').

class RangeValidator: function RangeValidator($field,$label='', $min, $max, $datatype='s', $type = 'optional')
Valida a entrada do usuário comparando-a com a faixa de valores estabelecida por $min e $max. $datatype indica se o valor é string ('s') ou inteiro ('i')

class RegExpValidator: function RegExpValidator($field,$label='', $regexp='', $type = 'optional')
Valida a entrada do usuário de acordo com uma expressão regular.

Controles do usuário

--incluir--

Styles

--incluir--

3. Regras de negócio

classe Business

Classe que serve de base para os objetos relativos às regras de negócios (diretório "classes" de cada módulo).

function CheckError($db)
Verifica ocorrência de erros no objeto Database $db e registra estes erros no objeto Business. Não apresenta nenhuma mensagem, apenas insere a informação que pode ser usada posteriormente.

function GetErrors()
Retorna os erros associados ao objeto.

function AddError($err)
Adiciona um erro à lista de erros do objeto.

function GetDatabase($database=null)
Associa um objeto Database ao objeto Business.

function GetQuery($sql, $parameters=null, $maxrows=0)

function objQuery($sql, $parameters=null, $maxrows=0)
Obtém um objeto Query, como resultado da execução da consulta em $sql.

function Execute($sql, $parameters=null)
Executa o comando $sql.

function GetAffectedRows()
Retorna o número de registro afetados pela execução do último comando.

function ExecuteBatch($cmds)
Executa uma seqüência de comando SQL, dentro de uma transação.

function objQueryRange($sql,&$range)
Obtém um objeto Query, como resultado da execução da consulta em $sql.

function ExecuteSP($sql, $parameters=null)
Execura uma StoredProcedure (não implementado).

function Log($operacao,$descricao)
Registra uma operação na tabela de Log do sistema.

function BeginTransaction()
Coloca a conexão do banco de dados em estado de transação.

function EndTransaction()
Encerra uma transação (com comit ou rollback).

4. Acesso a Dados

No framework Miolo, o acesso às base de dados relacionais é encapsulado em uma camada de acesso a dados, o que permite uma grande independência do banco de dados usado. No framework são distintos os conceitos de:

- **ResultSet:** corresponde a um array de linhas, retornado como resultado de uma query SQL. Cada linha corresponde a um registro e cada coluna corresponde a um campo. As colunas têm índice número a partir de 0, e são ordenadas de acordo com o comando SQL.
- **DataSet:** corresponde a uma instância da classe DataSet, usada internamente pela camada de acesso a dados, para encapsular o acesso a um ResultSet.

classe Database

function Database($conf,$system,$host,$db,$user,$pass)
Abre uma conexão com o banco de dados, criando uma instancia do objeto Connection correspondente àquele banco. Os parâmetros para a conexão são os descritos no arquivo de configuração miolo.conf.

function Close()
Fecha a conexão com o banco de dados.

function GetError()
Obtém o primeiro erro da lista de erros da conexão, se existir.

function GetErrors()
Obtém a lista de erros relativa à conexão.

function StartTransaction()
Inicia uma transação.

function CompleteTransaction()
Encerra uma transação com sucesso (comit).

function FailTransaction()
Encerra uma transação com falha (rollback).

function Execute($sql)
Executa o comando $sql.

function ExecuteBatch($sql_array)
Executa os comandos SQL do array $sql_array (simula a execução de um script). Os comandos são executados dentro de uma transação.

function Count($sql)
Retorna o número de registros do resultado da execução de $sql.

function Query($sql,$maxrows=0)
Retorna o ResultSet correspondente à execução de $sql, com no máximo $maxrows linhas.

function GetQuery($sql,$maxrows=0)

function objQuery($sql,$maxrows=0)
Retorna um objeto Query, correspondente à execução do comando $sql.

function GetTable($tablename)
Retorna um objeto Query, com todos os registros e campos da tabela $tablename. Corresponde à execução de "select * from $tablename".

function QueryRange($sql,&$range)
Retorna o ResultSet correspondente à execução do comando $sql. $range é uma instância da classe QueryRange e delimita a faixa de registros retornados.

function GetQueryRange($sql,&$range)

function objQueryRange($sql,&$range)
Retorna o objeto Query correspondente à execução do comando $sql. $range é uma instância da classe QueryRange e delimita a faixa de registros retornados.

function QueryChunk($sql, $maxrows, $offset, &$total)
Retorna o objeto Query correspondente à execução do comando $sql. $maxrows indica o máximo de registros a serem retornados, $offset indica o deslocamento a partir do início da tabela e $total retorna o efetivo número de registros do resultado.

function ExecProc($sql, $aParams=null)
Executa uma StoredProcedure (não implementado).

function Assert($info=false)
Interrompe a execução do script caso haja um erro no acesso ao banco de dados.

function GetAffectedRows()
Retorna o número de registros afetados pela execução de um comando SQL.

function CharToTimestamp($timestamp)
Faz a conversão do tipo char para timestamp (dependente do banco de dados).

function CharToDate($date)
Faz a conversão do tipo char para date (dependente do banco de dados).

function TimestampToChar($timestamp)
Faz a conversão do tipo timestamp para char (dependente do banco de dados).

function DateToChar($date)
Faz a conversão do tipo date para char (dependente do banco de dados).

classe Connection

A classe Connection é usada internamente pela camada de acesso a dados, definindo métodos abstratos que são implementados pelas classes *_connection (de cada banco de dados específico).

classe Dataset

A classe Dataset é usada internamente pela camada de acesso a dados, definindo métodos para acesso aos ResultSet originados pela execução de consultas SQL. As consultas são realizadas através de objetos da classe Query (que estende a classe Dataset). Os métodos descritos aqui são acessados através de um objeto Query, retornado por um método da classe Database. É usado o conceito de "ponteiro" para indicar a posição do ResultSet está sendo acessada em determinado momento e de "registro corrente" a linha indica pelo "ponteiro".

function MovePrev()
Move o ponteiro para o registro anterior ao corrente (ou BOF).

function MoveNext()
Move o ponteiro para o registro seguinte ao corrente (ou EOF).

function MoveFirst()
Move o ponteiro para o primeiro registro do ResultSet.

function MoveLast()
Move o ponteiro para o último registro do ResultSet.

function MoveTo($row)
Move o ponteiro para o registro $row do ResultSet.

function GetRowCount()
Retorna quantas linhas o ResultSet possui.

function GetColumnCount()
Retorna quantas colunas o ResultSet possui.

function GetColumnName($col)
Retorna o nome do campo corresponte à coluna $col.

function GetColumnPos($colname)
Retorna a posição da coluna correspondente ao campo $colname.

function GetValue($col)
Retorna o conteúdo do campo $col no registro corrente.

function Fields($field)
Retorna o conteúdo do campo $field no registro corrente.

function GetRowValues()
Retorna a linha correspondente ao registro corrente, como um array com índices numéricos.

function GetFieldValues()
Retorna a linha correspondente ao registro corrente, como um array com índices alfabéticos (nome dos campos).

function eof()
Indica se foi alcançado o fim do ResultSet.

function bof()
Indica se foi alcançado o início do ResultSet.

function chunkResult($key=0, $value=1, $showKeyValue=true)
Faz a compactação do ResultSet em um array associativo, cujos índices são determinados pelos valores da coluna $key e cujos valores são determinados pelos valores da coluna $value. $showKeyValue indica de o valor do índice será exibido no conteúdo do array.

function chunkResultMany($key, $values, $type='S', $separator='')
Faz a compactação do ResultSet em um array associativo, cujos índices são determinados pelos valores da coluna $key. $values é um array indicando quais colunas serão agregadas. $type determina se os valores serão agrupados como strings ($type='S'), separados por $separator, ou se serão colocados em um array.

function treeResult($group, $node)
Cria uma estrutura em árvore, baseada em um array associativo. $group indica quais campos devem ser agrupados, e $node indica quais campos formarão o valor do nó.

classe Query

A classe Query encapsula os métodos para a execução de consultas ao banco de dados. Esta classe define alguns métodos abstratos que são implementados pelas classes *_query para cada banco de dados específico. Os métodos de acesso ao ResultSet são herdados da classe Dataset.

function _query()
Método abstrato. Realiza efetivamente o acesso ao banco de dados.

function _error()
Método abstrato. Indica se houve algum erro no acesso ao banco de dados.

function _close()
Método abstrato. Libera os recursos usados na consulta.

function _setmetadata()
Método abstrato. Obtém metadados (p.ex. nome e tipo dos campos).

function GetError()
Retorna o erro encontrado na consulta, se houver.

function Open($maxrows=null, $offset=null)
Executa a consulta, retornando no máximo $maxrows a partir do registro $offset.

function Close()
Libera os recusos usados na consulta.

function SetConnection(&$c)
Indica qual conexão será usada para consulta. $c é um objeto Connection.

function SetSQL($sql)
Indica o comando SQL correspondente à consulta.

function SetOrder($order)
Acrescenta a cláusula "order by" ao comando SQL.

function SetFilter($filter)
Acrescenta a cláusula "where" ao comando SQL.

function GetCSV($filename = '')
Converte o ResultSet para o formato CSV.

classe Sql

A classe Sql é usada para encapsular um comando SQL, com seus respectivos parâmetros. No comando SQL o uso de parâmetros é indicado pelo caracter "?". Antes da execução do comando, os parâmetros devem ser substituídos por seus valores atuais.

function sql($columns='',$tables='',$where='',$orderBy='',$groupBy='',$having='')
Instancia um objeto Sql. Cada um dos parâmetros, quando fornecido, é usado na construção do comando SQL.

function Prepare($parameters)
Faz a substituição dos parâmetros formais pelos valores dos parâmetros atuais.

function Insert($parameters=null)
Gera uma instrução SQL INSERT, fazendo a substituição dos parâmetros formais pelos valores dos parâmetros atuais, se necessário.

function Delete($parameters=null)
Gera uma instrução SQL DELETE, fazendo a substituição dos parâmetros formais pelos valores dos parâmetros atuais, se necessário.

function Update($parameters=null)
Gera uma instrução SQL UPDATE, fazendo a substituição dos parâmetros formais pelos valores dos parâmetros atuais, se necessário.

function Select($parameters=null)
Gera uma instrução SQL SELECT, fazendo a substituição dos parâmetros formais pelos valores dos parâmetros atuais, se necessário.

function Clear()
Remove o conteúdo dos atributos, inicializando o objeto.

function SetParameters($parameters)
Indica o atributo parameters.

function CreateFrom($sqltext)
Estabelece os atributos do objeto com base um comando SQL SELECT.

classe QueryRange

function QueryRange($page,$rows,$total=0)
Instancia um objeto QueryRange, onde $page é o número da página, $rows é a quantidade de linhas por página e $total é quantidade efetiva de linha na página.

5. Segurança

class Login

Mantém as informações sobre o usário logado no sistema:

$id: login (armazenado no banco de dados)
$time: data/hora que o usuário logou
$user: nome completo do usuário
$userData: array com dados associados ao módulo
$idkey: "login id" no banco de dados
$idsetor: setor do usuário
$isAdmin: direitos de Administrador (true/false)
$idsessao: "session id" do PHP
$rights: array com os direitos de acesso
$groups: array com os grupos aos quais o usuário pertence
$idpessoa: id relativo à tabela Pessoa do banco de dados

classe Auth

Implementa os métodos de autenticação do usuário e mantém o objeto Login correspondente.

function CheckLogin()
Determina se há algum usuário logado. Retorna true se houver ou se o sistema não exige login e faz o login automático dependendo da configuração em miolo.conf.

function Authenticate($user,$pass)
Autentica o usuário e cria os objetos Login e Sessão correspondentes. Retorna o objeto Login, em caso de sucesso, ou null caso ocorra falha na autenticação.

function AuthenticateMD5($user,$challenge, $response)
Realiza a autenticação com MD5.

function & GetLogin()
Retorna o objeto Login em uso.

function SetLogin($login)
Atribui o objeto Login.

function IsLogged()
Determina se há um usuário logado.

function Logout($forced='')
Registra o fim da sessão, destrói o objeto session e atribui null para o objeto Login.

classe Perms

Implementa os métodos para verificação de direitos de acesso às transações do sistema. Os seguintes direitos estão definidos:

A_ACCESS/A_QUERY : permite o acesso de leitura/consulta
A_INSERT : permite inserções de registros nas tabelas

A_DELETE : permite remoções de registros
A_UPDATE : permite atualizações de registros
A_EXECUTE : permite inserção/remoção/atualização de regidtros'
A_SYSTEM/A_ADMIN: direitos de Admnistrador

function CheckAccess($trans, $access, $deny=false)

Verifica se o usuário corrente (objeto Login) tem o direito $access na transação $trans. $deny determina se a execução do script deve ser interrompida ou não.

function GetRights($trans, $loginid)

Retorna o direito de acesso do usuário $loginid na transação $trans.

function GetTransactionRights($loginid)

Retorna um array (transação, direito) com os direitos de acesso do usuário $loginid em cada transação.

function GetGroups($loginid)

Retorna um array com os grupos aos quais o usuário $loginid pertence.

function isMemberOf($login, $group)

Determina se o usuário $login pertence ao grupo $group.

function GetUsersAllowed($trans, $action)

Retorna um array com os usuários que têm o direito $action na transação $trans.

function GetGroupsAllowed($trans, $action)

Retorna um array com os grupos que têm o direito $action na transação $trans.

function isAdmin()

Determina se o usuário corrente tem direito de Administrador.