

Adianti Framework API 8.0

AdiantiApplicationConfig

Application config

Methods

public static function **start**()

Start and apply all configuration

public static function **load**(\$config)

Load configuration from array

public static function **apply**()

Apply some configurations that change env vars

public static function **get**()

Export configuration

AdiantiApplicationLoader

Application loader

Methods

public static function **getLoadedClasses**()

Return application loaded classes

public static function **isLoadedClass**(\$class)

Return if the class was already loaded by this loader

public static function **autoload**(\$class)

Application autoloder

AdiantiAutocompleteService

Autocomplete backend

Methods

public static function **onSearch**(\$param = null)

Search by the given word inside a model

AdiantiClassMap

Class map

Methods

public static function **getMap**()

public static function **getAllowedClasses**()

Return classes allowed to be directly executed

public static function **getInternalClasses**()

Return internal classes

public static function **getAliases**()

AdiantiCoreApplication

Basic structure to run a web application

Methods

public static function **run**(\$debug = FALSE)

Execute class/method based on request

\$debug : Activate Exception debug

public static function **execute**(\$class, \$method, \$request, \$endpoint = null)

Execute internal method

public static function **filterInput**()

Filter specific framework commands

public static function **setRouter**(Callable \$callback)

Set router callback

public static function **getRouter**()

Get router callback

public static function **setActionVerification**(Callable \$callback)

Set action_verification callback

public static function **getActionVerification**()

Get action_verification callback

public static function **executeMethod**(\$class, \$method = NULL, \$parameters = NULL)

Execute a specific method of a class with parameters

\$class : class name

\$method : method name

\$parameters : array of parameters

public static function **processRequest**(\$template)

Process request and insert the result it into template

public static function **gotoPage**(\$class, \$method = NULL, \$parameters = NULL, \$callback = NULL)

Goto a page

\$class : class name

\$method : method name

\$parameters : array of parameters

public static function **loadPage**(\$class, \$method = NULL, \$parameters = NULL)

Load a page

\$class : class name

\$method : method name

\$parameters : array of parameters

public static function **loadPageSimple**(\$class, \$method = NULL, \$parameters = NULL)

Load a page simplified mode

\$class : class name

\$method : method name

\$parameters : array of parameters

public static function **loadPageURL**(\$query)

Load a page url

\$class : class name

\$method : method name

\$parameters : array of parameters

public static function postData(\$formName, \$class, \$method = NULL, \$parameters = NULL)

Post data

\$class : class name

\$method : method name

\$parameters : array of parameters

public static function buildHttpQuery(\$class, \$method = NULL, \$parameters = NULL)

Build HTTP Query

\$class : class name

\$method : method name

\$parameters : array of parameters

public static function reload()

Reload application

public static function registerPage(\$page)

Register URL

\$page : URL to be registered

public static function errorHandler(\$errno, \$errstr, \$errfile, \$errline)

Handle Catchable Errors

public static function getHeaders()

Get request headers

public static function getRequestId()

Returns the execution id

public static function getDebugMode()

Returns the debug mode

AdiantiCoreLoader

Framework class autoloader

Methods

public static function loadClassMap()

Load the class map

public static function setClassPath(\$class, \$path)

Define the class path

\$class : Class name

\$path : Class path

public static function autoload(\$className)

Core autoloader

\$className : Class name

public static function legacyAutoload(\$class)

autoloader

\$class : classname

public static function globalScope(\$class)

make a class global

AdiantiCoreTranslator

Framework translation class for internal messages

Methods

public static function **getInstance()**

Returns the singleton instance

public static function **setLanguage(\$lang)**

Define the target language

\$lang : Target language index

public static function **getLanguage()**

Returns the target language

public static function **translate(\$word, \$param1 = NULL, \$param2 = NULL, \$param3 = NULL, \$param4 = NULL)**

Translate a word to the target language

\$word : Word to be translated

AdiantiDatabaseWidgetTrait

Methods

public static function **getItemsFromModel(\$database, \$model, \$key, \$value, \$ordercolumn = NULL, ?TCriteria \$criteria = NULL)**

Get items (key/value) from database to populate widget

public static function **getObjectsFromModel(\$database, \$model, \$key, \$ordercolumn = NULL, ?TCriteria \$criteria = NULL)**

Get objects from database to populate widget

AdiantiFileSaveTrait

Methods

public function **saveFile(\$object, \$data, \$input_name, \$target_path)**

Save file

\$object : Active Record

\$data : Form data

\$input_name : Input field name

\$target_path : Target file path

public function **saveFiles(\$object, \$data, \$input_name, \$target_path, \$model_files, \$file_field, \$foreign_key)**

Save files

\$object : Active Record

\$data : Form data

\$input_name : Input field name

\$target_path : Target file path

\$model_files : Files Active Record

\$file_field : File field in model_files

\$foreign_key : Foreign key to \$object

public function **saveFilesByComma(\$object, \$data, \$input_name, \$target_path)**

Save files comma separated

\$object : Active Record

\$data : Form data

\$input_name : Input field name

\$target_path : Target file path

public function **saveBinaryFile(\$object, \$data, \$attr_file, \$attr_file_name)**

Save binary file Database and column types supported: ORACLE => BLOB MYSQL => LONGBLOB MSSQL => VARBINARY(MAX) POSTGRES => BYTEA

\$object : Active Record

*\$data : Form data**\$attr_file : Input field name**\$attr_file_name : Active field name for name file*

```
public function saveBinaryFiles($object, $data, $attr_file_data, $model_files, $attr_file_name, $file_field, $foreign_key)
    Save binary file Database and column types supported: ORACLE => BLOB MYSQL => LONGBLOB MSSQL
=> VARBINARY(MAX) POSTGRES => BYTEA
```

*\$object : Active Record**\$data : Form data**\$attr_file_data : Input field name**\$model_files : Files Active Record**\$attr_file_name : Active field name for name file**\$file_field : File field in model_files**\$foreign_key : Foreign key to \$object*

```
public function loadBinaryFile($object, $attr_file, $attr_file_name)
```

*Save binary file on tmp and return path**\$object : Active Record**\$attr_file : Input field name**\$attr_file_name : Active field name for name file*

```
public function loadBinaryFiles($object, $model_files, $attr_file_name, $file_field, $foreign_key)
```

*Save binary files on tmp and return paths**\$object : Active Record**\$model_files : Files Active Record**\$attr_file_name : Active field name for name file**\$file_field : File field in model_files**\$foreign_key : Foreign key to \$object*

AdiantiHttpClient

Basic HTTP Client request

Methods

```
public static function request($url, $method = 'POST', $params = [], $authorization = null, $headers = [], $assoc = false)
```

*Execute a HTTP request**\$url : URL**\$method : method type (GET,PUT,DELETE,POST)**\$params : request body*

AdiantiMasterDetailTrait

Methods

```
public function storeItems($model, $foreign_key, $master_object, $detail_id, ?Callable $transformer = null)
```

*Store an item from details session into database**\$model : Model class name**\$foreign_key : Detail foreign key name**\$master_object : Master object**\$detail_id : Detail key in session**\$transformer : Function to be applied over the objects*

```
public function loadItems($model, $foreign_key, $master_object, $detail_id, ?Callable $transformer = null)
```

*Load items for detail into session**\$model : Model class name**\$foreign_key : Detail foreign key name**\$master_object : Master object**\$detail_id : Detail key in session**\$transformer : Function to be applied over the objects*

AdiantiMultiSearchService

MultiSearch backend

Methods

public static function **onSearch**(\$param = null)
Search by the given word inside a model

AdiantiPageControlTrait

Methods

public function **run**()
Interprets an action based at the URL parameters

public function **setPageName**(\$name)
Set page name

public function **getClassName**()
Return the Page name

public function **setIsWrapped**(\$bool)
Define if the element is wrapped inside another one
@bool : Boolean TRUE if is wrapped

AdiantiRecordService

Record rest service

Methods

public function **load**(\$param)
Find a Active Record and returns it
\$param : HTTP parameter

public function **delete**(\$param)
Delete an Active Record object from the database
[\$id] : HTTP parameter

public function **store**(\$param)
Store the objects into the database
\$param : HTTP parameter

public function **loadAll**(\$param)
List the Active Records by the filter
\$param : HTTP parameter

public function **deleteAll**(\$param)
Delete the Active Records by the filter
\$param : HTTP parameter

public function **countAll**(\$param)
Find the count Records by the filter
\$param : HTTP parameter

public function **handle**(\$param)
Handle HTTP Request and dispatch
\$param : HTTP POST and php input vars

AdiantiStandardCollectionTrait

Methods

protected function **setLimit**(\$limit)

method **setLimit()** Define the record limit

protected function **setCollectionObject**(\$object)

Set list widget

protected function **enableTransactionDebug**()

Enable Debug

protected function **setOrderCommand**(\$order_column, \$order_command)

Set order command

protected function **setDefaultOrder**(\$order, \$direction = 'asc')

Define the default order

\$order : The order field

\$direction : the order direction (asc, desc)

protected function **setFilterField**(\$filterField)

method **setFilterField()** Define wich field will be used for filtering PS: Just for Backwards compatibility

protected function **setOperator**(\$operator)

method **setOperator()** Define the filtering operator PS: Just for Backwards compatibility

protected function **addFilterField**(\$filterField, \$operator = 'like', \$formFilter = NULL, \$filterTransformer = NULL, \$logic_operator = TExpression::AND_OPERATOR)

method **addFilterField()** Add a field that will be used for filtering

\$filterField : Field name

\$operator : Comparison operator

protected function **setCriteria**(\$criteria)

method **setCriteria()** Define the criteria

protected function **setTransformer**(\$callback)

Define a callback method to transform objects before load them into datagrid

protected function **setAfterLoadCallback**(\$callback)

Define a callback method to transform objects before load them into datagrid

protected function **setAfterSearchCallback**(\$callback)

Define a callback method to transform objects after search action

public function **onSearch**(\$param = null)

Register the filter in the session

protected function **clearFilters**()

clear Filters

public function **onReload**(\$param = NULL)

Load the datagrid with the database objects

public function **show**()

method **show()** Shows the page

protected function **setDatabase**(\$database)

method **setDatabase()** Define the database

protected function **setActiveRecord**(\$activeRecord)

method **setActiveRecord()** Define wich Active Record class will be used

AdiantiStandardControlTrait

Methods

protected function **setDatabase**(\$database)
method setDatabase() Define the database

protected function **setActiveRecord**(\$activeRecord)
method setActiveRecord() Define wich Active Record class will be used

AdiantiStandardFormListTrait

Methods

public function **setAfterSaveAction**(\$action)
method setAfterSaveAction() Define after save action

public function **setUseMessages**(\$bool)
Define if will use messages after operations

public function **setLimit**(\$limit)
method setLimit() Define the record limit

public function **setDefaultOrder**(\$order, \$direction = 'asc')
Define the default order
\$order : The order field
\$direction : the order direction (asc, desc)

public function **setCriteria**(\$criteria)
method setCriteria() Define the criteria

public function **setTransformer**(\$callback)
Define a callback method to transform objects before load them into datagrid

public function **onReload**(\$param = NULL)
method onReload() Load the datagrid with the database objects

public function **onSave**()
method onSave() Executed whenever the user clicks at the save button

public function **onDelete**(\$param)
method onDelete() executed whenever the user clicks at the delete button Ask if the user really wants to delete the record

public function **Delete**(\$param)
method Delete() Delete a record

public function **onClear**(\$param)
Clear form

public function **onEdit**(\$param)
method onEdit() Executed whenever the user clicks at the edit button da datagrid

public function **show**()
Shows the page

protected function **setDatabase**(\$database)
method setDatabase() Define the database

protected function **setActiveRecord**(\$activeRecord)
method setActiveRecord() Define wich Active Record class will be used

AdiantiStandardFormTrait

Methods

- public function** `setAfterSaveAction($action)`
method setAfterSaveAction() Define after save action
- public function** `setAfterSaveCallback($callback)`
method setAfterSaveCallback() Define after save callback
- public function** `setUseMessages($bool)`
Define if will use messages after operations
- public function** `onSave()`
method onSave() Executed whenever the user clicks at the save button
- public function** `onClear($param)`
Clear form
- public function** `onEdit($param)`
method onEdit() Executed whenever the user clicks at the edit button da datagrid
\$param : An array containing the GET (\$_GET) parameters
- protected function** `setDatabase($database)`
method setDatabase() Define the database
- protected function** `setActiveRecord($activeRecord)`
method setActiveRecord() Define wich Active Record class will be used

AdiantiStandardListExportTrait

Methods

- public function** `onExportCSV($param)`
Export to CSV
- public function** `onExportXLS($param)`
Export to XLS
- public function** `onExportXML($param)`
Export to XML
- public function** `onExportPDF($param)`
Export datagrid as PDF
- protected function** `exportToCSV($output)`
Export to CSV
\$output : Output file
- protected function** `exportToXLS($output)`
Export to CSV
\$output : Output file
- protected function** `exportToXML($output)`
Export to XML
\$output : Output file
- protected function** `exportToPDF($output)`
Export to PDF
\$output : Output file

AdiantiStandardListTrait

Methods

public function enableTotalRow()
Enable total row

public function onInlineEdit(\$param)
Inline record editing

\$param : Array containing: key: object ID value field name: object attribute to be updated value: new attribute content

public function onDelete(\$param)
Ask before deletion

public function Delete(\$param)
Delete a record

public function onDeleteCollection(\$param)
Ask before delete record collection

public function deleteCollection(\$param)
method deleteCollection() Delete many records

protected function setLimit(\$limit)
method setLimit() Define the record limit

protected function setCollectionObject(\$object)
Set list widget

protected function enableTransactionDebug()
Enable Debug

protected function setOrderCommand(\$order_column, \$order_command)
Set order command

protected function setDefaultOrder(\$order, \$direction = 'asc')
Define the default order

\$order : The order field

\$direction : the order direction (asc, desc)

protected function setFilterField(\$filterField)
method setFilterField() Define wich field will be used for filtering PS: Just for Backwards compatibility

protected function setOperator(\$operator)
method setOperator() Define the filtering operator PS: Just for Backwards compatibility

protected function addFilterField(\$filterField, \$operator = 'like', \$formFilter = NULL, \$filterTransformer = NULL, \$logic_operator = TExpression::AND_OPERATOR)
method addFilterField() Add a field that will be used for filtering

\$filterField : Field name

\$operator : Comparison operator

protected function setCriteria(\$criteria)
method setCriteria() Define the criteria

protected function setTransformer(\$callback)
Define a callback method to transform objects before load them into datagrid

protected function setAfterLoadCallback(\$callback)
Define a callback method to transform objects before load them into datagrid

protected function setAfterSearchCallback(\$callback)

Define a callback method to transform objects after search action

public function **onSearch**(\$param = null)
Register the filter in the session

protected function **clearFilters**()
clear Filters

public function **onReload**(\$param = NULL)
Load the datagrid with the database objects

public function **show**()
method show() Shows the page

protected function **setDatabase**(\$database)
method setDatabase() Define the database

protected function **setActiveRecord**(\$activeRecord)
method setActiveRecord() Define wich Active Record class will be used

public function **onExportCSV**(\$param)
Export to CSV

public function **onExportXLS**(\$param)
Export to XLS

public function **onExportXML**(\$param)
Export to XML

public function **onExportPDF**(\$param)
Export datagrid as PDF

protected function **exportToCSV**(\$output)
Export to CSV
\$output : Output file

protected function **exportToXLS**(\$output)
Export to CSV
\$output : Output file

protected function **exportToXML**(\$output)
Export to XML
\$output : Output file

protected function **exportToPDF**(\$output)
Export to PDF
\$output : Output file

AdiantiStringConversion

String manipulation

Methods

public static function **utf8Encode**(\$string)
Polyfill for utf8_encode()

public static function **utf8Decode**(\$string)
Polyfill for utf8_decode()

public static function **camelCaseFromUnderscore**(\$string, \$spaces = FALSE)
Returns camel case string from underscore string

public static function **underscoreFromCamelCase**(\$string, \$spaces = FALSE)
Returns underscore string from camel case string

public static function **removeAccent**(\$str)
Remove accents from string

public static function **assureUnicode**(\$content)
Returns the string as Unicode when needed

public static function **assureISO**(\$content)
Assure ISO8859-1

public static function **slug**(\$content, \$separator = '-', \$tolower = true)
Returns the slug from string

public static function **replaceBetween**(\$str, \$needle_start, \$needle_end, \$replacement, \$include_limits = true)
Replace text between
\$str : Text to be replaced
\$needle_start : Start mark
\$needle_end : End mark
\$replacement : Text to be inserted
\$include_limits : if the mark limits will be replaced

public static function **getBetween**(\$str, \$needle_start, \$needle_end, \$include_limits = true)
Replace text between
\$str : Text to be replaced
\$needle_start : Start mark
\$needle_end : End mark
\$replacement : Text to be inserted
\$include_limits : if the mark limits will be replaced

AdiantiTemplateHandler

Template manipulation

Methods

public static function **replace**(\$content, \$object, \$cast = null, \$replace_methods = false)
Replace a string with object properties within {pattern}
\$content : String with pattern
\$object : Any object

public static function **evaluateExpression**(\$expression)
Evaluate math expression

public static function **replaceFunctions**(\$content)
replace some php functions

public static function **processAttribution**(\$content, &\$replacements)
Process variable attribution
\$content : Template content
\$replacements : Template variable replacements

AdiantiTemplateParser

Template parser

Methods

```
public static function parse($content)
    Parse template and replace basic system variables
    $content : raw template
```

AdiantiUploaderService

File uploader listener

Methods

```
function show($param)

public static function getMaximumFileUploadSizeFormatted()

public static function getMaximumFileUploadSize()

public static function convertSizeToBytes($size)
```

BootstrapDatagridWrapper

Bootstrap datagrid decorator for Adianti Framework

Methods

```
public function __construct(TDataGrid $datagrid)
    Constructor method

public function show()
    Shows the DataGrid

public function setId($id)
    Set id

public function setMutationAction(TAction $action)
    Define mutation action

public function setActionSide($side)
    Set actions side

public function generateHiddenFields()
    Generate hidden fields

public function disableHtmlConversion()
    Disable htmlspecialchars on output

public function getOutputData()
    Get raw processed output data

public function enablePopover($title, $content, $popside = null, $popcondition = null)
    Enable popover
    $title : Title
    $content : Content

public function makeScrollable()
    Make the datagrid scrollable

public function isScrollable()
    Returns if datagrid is scrollable

public function setActionWidth($width)
    Set the column action width
```

public function disableDefaultClick()
disable the default click action

public function setHeight(\$height)
Define the Height
\$height : An integer containing the height

public function getHeight()
Return datagrid height

public function addColumn(TDataGridColumn \$object, ?TAction \$action = null)
Add a Column to the DataGrid
\$object : A TDataGridColumn object

public function getColumns()
Returns an array of TDataGridColumn

public function addAction(TDataGridAction \$action, \$label = null, \$image = null)
Add an Action to the DataGrid
\$object : A TDataGridAction object

public function setActions(\$actions)
Set actions to the DataGrid
\$actions : TDataGridAction objects

public function prepareForPrinting()
Prepare for printing

public function setPageSize(\$page_size)
Set page Size
\$page_size : (a3,a4,a5,letter,legal)

public function getPageSize()
Return the page size

public function setPageOrientation(\$page_orientation)
Set page orientation
\$page_orientation : (portrait, landscape)

public function getPageOrientation()
Return the page orientation

public function addActionGroup(TDataGridActionGroup \$object)
Add an Action Group to the DataGrid
\$object : A TDataGridActionGroup object

public function getTotalColumns()
Returns the total columns

public function setGroupColumn(\$column, \$mask, \$transformer = null)
Set the group column for break

public function useGroupTotal(\$groupTotal = null)
Set the group column for break

public function clear(\$preserveHeader = TRUE, \$rows = 0)
Clear the DataGrid contents

public function createModel(\$create_header = true)
Creates the DataGrid Structure

public function **getHead()**
Return thead

public function **getBody()**
Return tbody

public function **prependRow(\$row)**
Prepend row

public function **insert(\$position, \$content)**
insert content

public function **addItems(\$objects)**
Add objects to the DataGrid
\$objects : An array of Objects

public function **addItem(\$object)**
Add an object to the DataGrid
\$object : An Active Record Object

public static function **appendRow(\$table_id, \$row)**
Append table row via Javascript

public static function **removeRowById(\$table_id, \$id)**
Remove row by id

public static function **replaceRowById(\$table_id, \$id, \$row)**
Replace row by id

public function **getItems()**
Return datagrid items

public function **getRowIndex(\$attribute, \$value)**
Find the row index by object attribute
\$attribute : Object attribute
\$value : Object value

public function **getRow(\$position)**
Return the row by position
\$position : Row position

public function **getWidth()**
Returns the DataGrid's width

public function **setPageNavigation(\$pageNavigation)**
Assign a PageNavigation object
\$pageNavigation : object

public function **getPageNavigation()**
Return the assigned PageNavigation object

public function **setSearchForm(\$searchForm)**
Assign a TForm object
\$searchForm : object

public function **getSearchForm()**
Return the assigned Search form object

public function **setSearchAttributes(\$attributes)**
Set search attributes

```
public function enableSearch(TField $input, $attributes)
```

Enable fuse search

\$input : Field input for search

\$attribute : Attribute name

```
public function addQuickColumn($label, $name, $align = 'left', $size = 200, ?TAction $action = NULL, $param = NULL)
```

Add a column

\$label : Field Label

\$object : Field Object

\$size : Field Size

```
public function addQuickAction($label, TDataGridAction $action, $field, $icon = NULL)
```

Add action to the datagrid

\$label : Action Label

\$action : TAction Object

\$icon : Action Icon

BootstrapFormBuilder

Bootstrap form builder for Adianti Framework

Methods

```
public function __construct($name = 'my_form')
```

Constructor method

\$name : form name

```
public function setClientValidation($bool)
```

Turn on/off client validation

```
public function enableClientValidation()
```

Enable client validation

```
public function enableCSRFProtection()
```

Enable CSRF Protection

```
public function addExpandButton( $label = null, $icon = null, $start_hidden = true)
```

Add expand button

```
public function hide()
```

Hide

```
public function generateAria()
```

Generate automatic aria-labels

```
public function getId()
```

Returns form id

```
public function setFieldSizes($size)
```

Set field sizes

```
public function setFormTitle($title)
```

Add a form title

\$title : Form title

```
public function getFormTitle()
```

Return the form title

```
public function setPadding($padding)
```

Set padding

\$padding :

public function setCurrentPage(\$i)
Define the current page to be shown
\$i : An integer representing the page number (start at 0)

public function setProperty(\$name, \$value)
Define a style property
\$name : Property Name
\$value : Property Value

public function setHeaderProperty(\$name, \$value)
Define a header style property
\$name : Property Name
\$value : Property Value

public function setName(\$name)
Set form name
\$name : Form name

public function getName()
Get form name

public function addField(AdiantiWidgetInterface \$field)
Add form field
\$field : Form field

public function delField(AdiantiWidgetInterface \$field)
Del form field
\$field : Form field

public function setFields(\$fields)
Set form fields
\$fields : Array of Form fields

public function getField(\$name)
Return form field
\$name : Field name

public function getFields()
Return form fields

public function clear(\$keepDefaults = FALSE)
Clear form

public function setData(\$object)
Set form data
\$object : Data object

public function getData(\$class = 'StdClass')
Get form data
\$class : Object type of return data

public function getActions()
Return form actions

public function validate()
Validate form data

public function appendPage(\$title)
Append a notebook page
\$title : Tab title

public function setTabFunction(\$function)

Set tab click function

public function **setTabAction**(TAction \$action)
Define the action for the Notebook tab

\$action : Action taken when the user clicks over Notebook tab (A TAction object)

public function **addFields**()
Add form fields

mixed : \$fields,... Form fields

public function **addContent**()
Add a form content

mixed : \$content,... Form content

public function **validateInlineArguments**(\$args, \$method)
Validate argument type

\$args : Array of arguments

\$method : Generator method

public function **addAction**(\$label, TAction \$action, \$icon = 'fa:save', \$name = null)
Add a form action

\$label : Button label

\$action : Button action

\$icon : Button icon

public function **addActionLink**(\$label, TAction \$action, \$icon = 'fa:save')
Add a form action link

\$label : Button label

\$action : Button action

\$icon : Button icon

public function **addHeaderAction**(\$label, TAction \$action, \$icon = 'fa:save')
Add a form header action

\$label : Button label

\$action : Button action

\$icon : Button icon

public function **addHeaderWidget**(\$widget)
Add a form header widget

\$widget : Widget

public function **addFooterWidget**(\$widget)
Add a form footer widget

\$widget : Widget

public function **addHeaderActionLink**(\$label, TAction \$action, \$icon = 'fa:save')
Add a form header action

\$label : Button label

\$action : Button action

\$icon : Button icon

public function **addButton**(\$label, \$action, \$icon = 'fa:save')
Add a form button

\$label : Button label

\$action : JS Button action

\$icon : Button icon

public function **delActions**()
Clear actions row

public function **getActionButtons**()
Return an array with action buttons

public function **setColumnClasses**(\$key, \$classes)

public function **render**()
Render form

public function **show**()
Show form

public static function **wrapField**(\$field, \$display, \$default_field_size = null)
Create a field wrapper

public static function **showField**(\$form, \$field, \$speed = 0)

public static function **hideField**(\$form, \$field, \$speed = 0)

public function **getContents**()
Returns the element content as a string

BootstrapFormWrapper

Bootstrap form decorator for Adianti Framework

Methods

public function **__construct**(TQuickForm \$form, \$class = 'form-horizontal')
Constructor method

public function **setClientValidation**(\$bool)
Turn on/off client validation

public function **getElement**()
Return render element

public function **setName**(\$name)
Set form name

public function **getName**()
Get form name

public function **addField**(AdiantiWidgetInterface \$field)
Add form field

public function **delField**(AdiantiWidgetInterface \$field)
Del form field

public function **setFields**(\$fields)
Set form fields

public function **getField**(\$name)
Return form field

public function **getFields**()
Return form fields

public function **clear**()
Clear form

public function **setData**(\$object)
Set form data

public function **getData**(\$class = 'StdClass')
Get form data

public function validate()
Validate form data

public function show()
Shows the decorated form

public function getActionsContainer()
Returns the actions container

public function getTable()
Returns the inner table

public function setFieldsByRow(\$count)
Define the field quantity per row
\$count : Field count

public function getFieldsByRow()
Return the fields by row count

public function getContainer()
Returns the form container

public function setFormTitle(\$title)
Add a form title
\$title : Form title

public function getInputRows()
Returns the input groups

public function addQuickField(\$label, AdiantiWidgetInterface \$object, \$size = 200, ?TFieldValidator \$validator = NULL, \$label_size = NULL)
Add a form field
\$label : Field Label
\$object : Field Object
\$size : Field Size
\$validator : Field Validator

public function addQuickFields(\$label, \$objects, \$required = FALSE)
Add a form field
\$label : Field Label
\$objects : Array of Objects
\$required : Boolean TRUE if required

public function addQuickAction(\$label, TAction \$action, \$icon = 'fa:save')
Add a form action
\$label : Action Label
\$action : TAction Object
\$icon : Action Icon

public function addQuickButton(\$label, \$action, \$icon = 'fa:save')
Add a form button
\$label : Action Label
\$action : Javascript action
\$icon : Action Icon

public function delActions()
Clear actions row

public function getActionButtons()
Return an array with action buttons

public function detachActionButtons()

Detach action buttons

public function **addRow()**
Add a row

public static function **showField**(\$form, \$field, \$speed = 0)

public static function **hideField**(\$form, \$field, \$speed = 0)

BootstrapNotebookWrapper

Bootstrap datagrid decorator for Adianti Framework

Methods

public function **__construct**(TNotebook \$notebook, \$wrapper_class = "")
Constructor method

public function **setTabsDirection**(\$direction, \$divisions = null)
Set tabs direction
\$direction : Tabs direction (left right)

public function **show()**
Show the notebook

public function **setTabsVisibility**(\$visible)
Define if the tabs will be visible or not
\$visible : If the tabs will be visible

public function **setTabsSensibility**(\$sensibility)
Define the tabs click sensibility
\$sensibility : If the tabs will be sensible to click

public function **getId()**
Returns the element ID

public function **setSize**(\$width, \$height)
Set the notebook size
\$width : Notebook's width
\$height : Notebook's height

public function **getSize()**
Returns the frame size

public function **setCurrentPage**(\$i)
Define the current page to be shown
\$i : An integer representing the page number (start at 0)

public function **getCurrentPage()**
Returns the current page

public function **appendPage**(\$title, \$object)
Add a tab to the notebook
\$title : tab's title
\$object : tab's content

public function **getPageCount()**
Return the Page count

public function **setTabAction**(TAction \$action)
Define the action for the Notebook tab
\$action : Action taken when the user clicks over Notebook tab (A TAction object)

```
public function render()
    Render the notebook
```

TAPCache

Adianti APC Record Cache

Methods

```
public static function enabled()
    Returns if the service is active
```

```
public static function setValue($key, $value)
    Store a variable in cache
    $key : Key
    $value : Value
```

```
public static function getValue($key)
    Get a variable from cache
    $key : Key
```

```
public static function delValue($key)
    Delete a variable from cache
    $key : Key
```

```
public static function clear()
    Clear cache
```

TAction

Structure to encapsulate an action

Methods

```
public function __construct($action, $parameters = null)
    Class Constructor
    $action : Callback to be executed
    $parameters : = array of parameters
```

```
public function disableState()
    Disable URL history state register
```

```
public function cloneWithParameters($parameters = [])
```

```
public function getFieldParameters()
    Return fields used in parameters
```

```
public function toString()
    Returns the action as a string
```

```
public function setParameter($param, $value)
    Adds a parameter to the action
    $param : = parameter name
    $value : = parameter value
```

```
public function setParameters($parameters)
    Set the parameters for the action
    $parameters : = array of parameters
```

```
public function getParameter($param)
    Returns a parameter
```

\$param : = parameter name

public function `getParameters()`
Return the Action Parameters

public function `getAction()`
Returns the current callback

public function `setProperty($property, $value)`
Set property

public function `getProperty($property)`
Get property

public function `prepare($object)`
Prepare action for use over an object
\$object : Data Object

public function `preserveRequestParameters($request_parameters = [])`
Keep REQUEST parameters

public function `serialize($format_action = TRUE, $check_permission = FALSE)`
Converts the action into an URL
\$format_action : = format action with document or javascript (ajax=no)

public function `validate()`
Validate action

public function `isStatic()`
Returns if the action is static

public function `usePopover()`
Enable popover

public function `isPopover()`
Return if is using popover

TActionLink extends TTextDisplay

Action Link

Methods

public function `__construct($value, ?TAction $action = null, $color = null, $size = null, $decoration = null, $icon = null)`
Class Constructor
\$value : text content
\$action : TAction Object
\$color : text color
\$size : text size
\$decoration : text decorations (*b=bold*, *i=italic*, *u=underline*)

public function `setAction(TAction $action)`
Set the action

public function `getAction()`
Returns the current callback

public function `setImage($icon)`

public function `setLabel($label)`
Replace current label

```
public function addStyleClass($class)
    Add CSS class
```

TAdiantiCoreTranslator

Framework translation class for internal messages

Methods

```
public static function getInstance()
    Returns the singleton instance
```

```
public static function setLanguage($lang)
    Define the target language
    $lang : Target language index
```

```
public static function getLanguage()
    Returns the target language
```

```
public static function translate($word, $param1 = NULL, $param2 = NULL, $param3 = NULL, $param4 = NULL)
    Translate a word to the target language
    $word : Word to be translated
```

TAlert extends TElement

Alert

Methods

```
public function __construct($type, $message)
    Class Constructor
    $type : Type of the alert (success, info, warning, danger)
    $message : Message to be shown
```

TArrowStep extends TField

Arrow Step

Methods

```
public function __construct($name)
    Constructor
```

```
public static function disableField($formName, $name)
    Disable field
    $name : name of arrow steps
```

```
public static function enableField($formName, $name)
    Enable field
    $name : name of arrow steps
```

```
public static function clearField($formName, $name)
    Clear currents item on steps
    $name : name of arrow steps
```

```
public static function defineCurrent($name, $value)
    Define current item on steps
    $name : name of arrow steps
    $value : value current
```

```
public function setEditable($editable)
```


Define if the field is editable

\$editable : A boolean

public function **getEditable()**
Returns if the field is editable

public function **getPostData()**
Return the post data

public function **setFormName(\$name)**
Set form name

public function **setName(\$name)**
Set name

public function **getName()**
Get name

public function **setValue(\$value)**
Set value current step

public function **getValue()**
Get value current step

public function **setHideText(bool \$hide = true)**
Set hide text
\$hide : bool

public function **setFontSize(\$fontSize)**
Set font size
\$size : string to color

public function **setFilledColor(string \$color, \$fontColor = null)**
Set color arrows
\$color : string to color
\$fontColor : string to color font

public function **setFilledFontColor(string \$fontColor)**
Set font color arrows
\$color : string to color

public function **setUnfilledColor(string \$color, \$fontColor = null)**
Set color arrows
\$color : string to color
\$fontColor : string to color font

public function **setUnfilledFontColor(string \$color)**
Set color arrows
\$fontColor : string to color font

public function **setWidth(\$width)**
Set width
\$width : int|float to width

public function **getWidth()**
Get width

public function **setSize(\$width, \$height = null)**
Set sizes
\$width :
\$height :

```
public function setHeight($height)
```

Set height arrows

\$height : int|float to height

```
public function getHeight()
```

Get heigth

```
public function getSize()
```

Get size

```
public function addItem($title, $id = null, $color = null)
```

Add an item

\$title : Item title

\$id : Item id

\$color : Item color

```
public function setColorItems($colorItems)
```

Set color items

\$colorItems : Items

```
public function setItems($items)
```

Set items

\$item : Items

```
public function addItem($items)
```

Add items

\$item : Items

```
public function getItems()
```

Get items

```
public function getItem($key)
```

Get item

```
public function setAction(TAction $action)
```

Set action

\$action : Action

```
public function getAction()
```

Get action

```
public function setCurrentKey($key)
```

Select current item

```
public function getCurrent()
```

Get current item

```
public function setCurrent($title)
```

Select current item

```
public function show()
```

Show component

TBarCodeInputReader extends TEntry

BarCode Widget

Methods

```
public function __construct($name)
```

Class Constructor

***\$name** : Name of the widget*

public function **setChangeFunction**(\$function)
Set change function

public function **setChangeAction**(TAction \$action)
Define the action to be executed when the user changes the content
***\$action** : TAction object*

public function **show**()
Shows the widget at the screen

TBarcodeDisplay extends TElement

Barcode Display

Methods

public function **__construct**(\$value)
Class Constructor
***\$value** : barcode content*

public function **setValue**(\$value)
Replace current value

public function **setHeight**(\$height)
Define the bar code height
***\$width** : Barcode height in pixels*

public function **setType**(\$type)
Define the bar code type
***\$type** : Barcode type ('ean13', 'code128', 'code39', 'inter25', 'codabar', 'upca', 'pharma')*

public function **getValue**()
Returns the field value

public function **show**()

TBreadcrumb extends TElement

BreadCrumb

Methods

public function **__construct**()
Handle paths from a XML file
***\$xml_file** : path for the file*

public static function **create**(\$options)
Static constructor

public function **addItem**(\$path, \$last = FALSE)
Add an item
***\$path** : Path to be shown*
***\$last** : If the item is the last one*

public function **select**(\$path)
Mark one breadcrumb item as selected

TButton extends TField

*Button Widget***Methods**

public static function **create**(\$name, \$callback, \$label, \$image)
Create a button with icon and action

public function **addStyleClass**(\$class)
Add CSS class

public function **setAction**(TAction \$action, \$label = NULL)
Define the action of the button
\$action : TAction object
\$label : Button's label

public function **getAction**()
Returns the button's action

public function **setTagName**(\$name)
Define the tag name
\$name : tag name

public function **setImage**(\$image)
Define the icon of the button
\$image : image path

public function **setLabel**(\$label)
Define the label of the button
\$label : button label

public function **getLabel**()
Returns the button label

public function **setProperty**(\$name, \$value, \$replace = TRUE)
Define a field property
\$name : Property Name
\$value : Property Value

public function **getProperty**(\$name)
Return field property

public static function **enableField**(\$form_name, \$field)
Enable the field
\$form_name : Form name
\$field : Field name

public static function **disableField**(\$form_name, \$field)
Disable the field
\$form_name : Form name
\$field : Field name

public function **show**()
Show the widget at the screen

TCNPJValidator extends TFieldValidator

CNPJ validation (Valid only in Brazil)

Methods

public function **validate**(\$label, \$value, \$parameters = NULL)

Validate a given value

\$label : Identifies the value to be validated in case of exception

\$value : Value to be validated

\$parameters : additional parameters for validation

TCPFValidator extends TFieldValidator

CPF validation (Valid only in Brazil)

Methods

public function **validate**(\$label, \$value, \$parameters = NULL)

Validate a given value

\$label : Identifies the value to be validated in case of exception

\$value : Value to be validated

\$parameters : additional parameters for validation

TCalendar extends TElement

Calendar Widget

Methods

public function **__construct**()

Class Constructor

public function **highlightWeekend**()

highlight weekend

public function **setSize**(\$width, \$height)

Define the calendar's size

\$width : Window's width

\$height : Window's height

public function **setMonth**(\$month)

Define the current month to display

\$month : Month to display

public function **setYear**(\$year)

Define the current year to display

\$year : Year to display

public function **getMonth**()

Return the current month

public function **getYear**()

Return the current year

public function **setAction**(TAction \$action)

Define the action when click at some day

\$action : TAction object

public function **selectDays**(array \$days)

Select a collection of days

\$days : Collection of days

public function **show**()

Show the calendar

TCardView extends TElement

Card

Methods

public function `__construct()`
Class Constructor

public function `enableDefaultClick($head = TRUE, $body = TRUE)`
Enable default click in title and body

public function `setItemHeight($height)`
Set item min height
\$height : min height

public function `setItemWidth($width)`
Set item min width
\$width : min width

public function `setItemDatabase($database)`
Set item min database
\$database : min database

public function `setContentHeight($height)`
Set content min height
\$height : min height

public function `setTitleAttribute($field)`
Set title attribute
\$field : attribute name

public function `setContentAttribute($field)`
Set content attribute
\$field : attribute name

public function `setColorAttribute($field)`
Set color attribute
\$field : attribute name

public function `setItemClass($class)`
Set custom class card
\$class : class name

public function `clear()`
Clear items

public function `addItem($object)`
Add item
\$object : Item data object

public function `setUseButton()`
Display icons as buttons

public function `setTemplatePath($path)`
Set card item template for rendering
\$path : Template path

public function `setItemTemplate($template)`
Set card item template for rendering
\$template : Template content

public function `setItemTemplateCallback(Callable $callback)`

Set card item callback for rendering

\$callback : Callback

public function setTitleTemplate(\$template)

Set card title template for rendering

\$template : Template content

public function addAction(TAction \$action, \$label, \$icon = NULL, \$display_condition = NULL, \$title = NULL)

Add item action

\$label : Action label

\$action : Action callback (TAction)

\$icon : Action icon

\$display_condition : Display condition

public function addActionGroup(\$title, \$actions, \$icon)

Add action group

\$title : Action group title

\$actions : Array of actions

\$icon : Action group icon

public function enableSearch(TField \$input, \$attribute)

Enable fuse search

\$input : Field input for search

\$attribute : Attribute name

public function setPageNavigation(\$pageNavigation)

Assign a PageNavigation object

\$pageNavigation : object

public function getPageNavigation()

Return the assigned PageNavigation object

public function setSearchForm(\$searchForm)

Assign a TForm object

\$searchForm : object

public function getSearchForm()

Return the assigned Search form object

public function prepareForPrinting()

Prepare for printing

public function setPageSize(\$page_size)

Set page Size

\$page_size : (a3,a4,a5,letter,legal)

public function getPageSize()

Return the page size

public function setPageOrientation(\$page_orientation)

Set page orientation

\$page_orientation : (portrait, landscape)

public function getPageOrientation()

Return the page orientation

public function show()

Show cards

TCheckButton extends TField

*CheckBox widget***Methods**

public function `__construct`(\$name)
Class Constructor

\$name : Name of the widget

public function `setUseSwitch`(\$useSwitch = TRUE, \$labelClass = 'blue')
Show as switch

public function `setIndexValue`(\$index)
Define the index value for check button

public function `after`(\$element)
Insert element after

public function `show`()
Shows the widget at the screen

TCheckGroup extends TField

*A group of CheckBox's***Methods**

public function `__construct`(\$name)
Class Constructor

\$name : name of the field

public function `checkAll`()
Check all options

public function `setLayout`(\$dir)
Define the direction of the options
\$direction : String (vertical, horizontal)

public function `getLayout`()
Get the direction (vertical or horizontal)

public function `setBreakItems`(\$breakItems)
Define after how much items, it will break

public function `setUseButton`()
Show as button

public function `setUseSwitch`(\$useSwitch = TRUE, \$labelClass = 'blue')
Show as switch

public function `addItem`(\$items)
Add items to the check group
\$items : An indexed array containing the options

public function `getItems`()
Return the items

public function `getButtons`()
Return the option buttons

public function `getLabels`()
Return the option labels

public function setValueSeparator(\$sep)

Define the field's separator

\$sep : A string containing the field's separator

public function setValue(\$value)

Define the field's value

\$value : A string containing the field's value

public function getPostData()

Return the post data

public function setChangeAction(TAction \$action)

Define the action to be executed when the user changes the combo

\$action : TAction object

public function setChangeFunction(\$function)

Set change function

public static function reload(\$formname, \$name, \$items, \$options)

Reload checkbox items after it is already shown

\$formname : form name (used in gtk version)

\$name : field name

\$items : array with items

\$options : array of options [layout, size, breakItems, useButton, valueSeparator, value, changeAction, changeFunction, checkAll]

public static function enableField(\$form_name, \$field)

Enable the field

\$form_name : Form name

\$field : Field name

public static function disableField(\$form_name, \$field)

Disable the field

\$form_name : Form name

\$field : Field name

public static function clearField(\$form_name, \$field)

clear the field

\$form_name : Form name

\$field : Field name

public function show()

Shows the widget at the screen

TCheckList

Checklist

Methods

public function __construct(\$name)

Construct method

public function setSelectAction(TAction \$action)

Define the action to be executed when the user selects a row

\$action : TAction object

public function disableHtmlConversion()

Disable htmlspecialchars on output

public function setSize(\$size)

Set checklist size

```

function getSize()
    Returns checklist size

public function setId($id)
    Change Id

public static function checkAll($name, $fire_events = true)

public static function checkNone($name, $fire_events = true)

public function disableCheckAll()
    Disable check all button

public function setName($name)
    Define the field's name
    $name : A string containing the field's name

public function getName()
    Returns the field's name

public function setValue($value)
    Define the checklist selected ids
    $value :

public function getValue()
    Returns the selected ids

public function setIdColumn($name)
    Define the Identification column

public function addColumn($name, $label, $align, $width)
    Add list column
    $name : = Name of the column in the database
    $label : = Text label that will be shown in the header
    $align : = Column align (left, center, right)
    $width : = Column Width (pixels)

public function addItem($object)
    Add item

public function addItems($objects)
    add Items

public function fillWith($database, $model, $key, $ordercolumn = NULL, ?TCriteria $criteria = NULL)
    Fill with model objects

public function clear()
    Clear datagrid

public function getFields()
    Get fields

public function getFormName()
    Return the name of the form to wich the field is attached

public function setValueSeparator($sep)
    Define the field's separator
    $sep : A string containing the field's separator

public function getPostData()
    Get post data

```

```
public function addValidation($label, TFieldValidator $validator, $parameters = NULL)
```

Add a field validator

\$label : Field name

\$validator : TFieldValidator object

\$parameters : Additional parameters

```
public function getValidations()
```

Returns field validations

```
public function validate()
```

Validate a field

```
public static function enableField($field)
```

Enable the field

\$form_name : Form name

\$field : Field name

```
public static function disableField($field)
```

Disable the field

\$form_name : Form name

\$field : Field name

```
public function show()
```

Show checklist

```
public static function getItemsFromModel($database, $model, $key, $value, $ordercolumn = NULL, ?TCriteria $criteria = NULL)
```

Get items (key/value) from database to populate widget

```
public static function getObjectsFromModel($database, $model, $key, $ordercolumn = NULL, ?TCriteria $criteria = NULL)
```

Get objects from database to populate widget

TColor extends TEntry

Color Widget

Methods

```
public function __construct($name)
```

Class Constructor

\$name : Name of the widget

```
public function setOption($option, $value)
```

Set extra option TColor

\$option : Key name option

\$value : Option value

```
public function getOptions()
```

Get options TColor

```
public function getOption($option)
```

Get option TColor

```
public function setTheme($theme)
```

Set theme

```
public static function enableField($form_name, $field)
```

Enable the field

\$form_name : Form name

\$field : Field name

public static function **disableField**(\$form_name, \$field)

Disable the field

\$form_name : Form name

\$field : Field name

public function **setChangeFunction**(\$function)

Set change function

public function **setChangeAction**(TAction \$action)

Define the action to be executed when the user changes the content

\$action : TAction object

public function **show**()

Shows the widget at the screen

TCombo extends TField

ComboBox Widget

Methods

public function **__construct**(\$name)

Class Constructor

\$name : widget's name

public function **setBooleanMode**()

Enable/disable boolean mode

public function **setValue**(\$value)

Define the field's value

\$value : A string containing the field's value

public function **getValue**()

Returns the field's value

public function **clear**()

Clear combo

public function **addItems**(\$items)

Add items to the combo box

\$items : An indexed array containing the combo options

public function **getItems**()

Return the combo items

public function **enableSearch**()

Enable search

public function **getPostData**()

Return the post data

public function **setChangeAction**(TAction \$action)

Define the action to be executed when the user changes the combo

\$action : TAction object

public function **setChangeFunction**(\$function)

Set change function

public static function **reload**(\$formname, \$name, \$items, \$startEmpty = FALSE, \$fire_events = TRUE)

Reload combobox items after it is already shown

\$formname : form name (used in gtk version)

\$name : field name
\$items : array with items
\$startEmpty : if the combo will have an empty first item
\$fire_events : If change action will be fired

public static function **enableField**(\$form_name, \$field)
 Enable the field

\$form_name : Form name
\$field : Field name

public static function **disableField**(\$form_name, \$field)
 Disable the field

\$form_name : Form name
\$field : Field name

public static function **clearField**(\$form_name, \$field, \$fire_events = TRUE)
 Clear the field

\$form_name : Form name
\$field : Field name
\$fire_events : If change action will be fired

public function **setDefaultOption**(\$option)
 Define the combo default option value

\$option : option value

public function **renderItems**()
 Render items

public function **show**()
 Shows the widget

TConnection

Singleton manager for database connections

Methods

public static function **open**(\$database)
 Opens a database connection
\$database : Name of the database (an INI file).

public static function **setConfigPath**(\$path)
 Change database configuration Path
\$path : Config path

public static function **openArray**(\$db)
 Opens a database connection from array with db info
\$db : Array with database info

public static function **getDatabaseInfo**(\$database)
 Returns the database information as an array
\$database : INI File

public static function **setDatabaseInfo**(\$database, \$info)
 Set database info
\$database : Database name
\$info : Database connection information

TCriteria extends TExpression

Provides an interface for filtering criteria definition

Methods

public function `__construct()`
Constructor Method

public static function `create($simple_filters, $properties = null)`
create criteria from array of filters

public function `add(TExpression $expression, $operator = self::AND_OPERATOR)`
Adds a new Expression to the Criteria
\$expression : TExpression object
\$operator : Logic Operator Constant

public function `isEmpty()`
Return if criteria is empty

public function `getPreparedVars()`
Return the prepared vars

public function `dump($prepared = FALSE)`
Returns the final expression
\$prepared : Return a prepared expression

public function `setProperty($property, $value)`
Define a Criteria property
\$property : Name of the property (limit, offset, order, direction)
\$value : Value for the property

public function `resetProperties()`
reset criteria properties

public function `setProperties($properties)`
Set properties form array
\$properties : array of properties

public function `getProperty($property)`
Return a Criteria property
\$property : Name of the property (LIMIT, OFFSET, ORDER)

public function `setCaseInsensitive(bool $value) : void`
Force case insensitive searches

public function `getCaseInsensitive() : bool`
Return if case insensitive is turned on

TDBArrowStep extends TArrowStep

Database Arrow Step

Methods

public function `__construct($name, $database, $model, $key, $value, $ordercolumn = NULL, ?TCriteria $criteria = NULL)`
Class Constructor
\$name : widget's name
\$database : database name
\$model : model class name
\$key : table field to be used as key in the combo
\$value : table field to be listed in the combo
\$ordercolumn : column to order the fields (optional)
\$criteria : criteria (TCriteria object) to filter the model (optional)

```
public function setColorColumn($colorcolumn)
```

```
public function show()
```

```
public static function getItemsFromModel($database, $model, $key, $value, $ordercolumn = NULL, ?TCriteria $criteria = NULL)
```

Get items (key/value) from database to populate widget

```
public static function getObjectsFromModel($database, $model, $key, $ordercolumn = NULL, ?TCriteria $criteria = NULL)
```

Get objects from database to populate widget

TDBCheckGroup extends TCheckGroup

Database CheckBox Widget

Methods

```
public function __construct($name, $database, $model, $key, $value, $ordercolumn = NULL, ?TCriteria $criteria = NULL)
```

Class Constructor

\$name : widget's name

\$database : database name

\$model : model class name

\$key : table field to be used as key in the combo

\$value : table field to be listed in the combo

\$ordercolumn : column to order the fields (optional)

\$criteria : criteria (TCriteria object) to filter the model (optional)

```
public static function reloadFromModel($formname, $field, $database, $model, $key, $value, $ordercolumn = NULL, $criteria = NULL, $options = [])
```

Reload checkbox from model data

\$formname : form name

\$field : field name

\$database : database name

\$model : model class name

\$key : table field to be used as key in the checkbox

\$value : table field to be listed in the checkbox

\$ordercolumn : column to order the fields (optional)

\$criteria : criteria (TCriteria object) to filter the model (optional)

\$options : array of options [layout, breakItems, useButton, value, valueSeparator, changeAction, changeFunction]

```
public static function getItemsFromModel($database, $model, $key, $value, $ordercolumn = NULL, ?TCriteria $criteria = NULL)
```

Get items (key/value) from database to populate widget

```
public static function getObjectsFromModel($database, $model, $key, $ordercolumn = NULL, ?TCriteria $criteria = NULL)
```

Get objects from database to populate widget

TDBCheckList extends TCheckList

Database Checklist

Methods

```
public function __construct($name, $database, $model, $key, $value, $ordercolumn = NULL, ?TCriteria $criteria = NULL)
```

Class Constructor

\$name : widget's name

\$database : database name

\$model : model class name

\$key : table field to be used as key in the combo

\$value : table field to be listed in the combo

\$ordercolumn : column to order the fields (optional)

\$criteria : criteria (TCriteria object) to filter the model (optional)

```
public function show()
    show
```

```
public static function getItemsFromModel($database, $model, $key, $value, $ordercolumn = NULL, ?TCriteria $criteria = NULL)
    Get items (key/value) from database to populate widget
```

```
public static function getObjectsFromModel($database, $model, $key, $ordercolumn = NULL, ?TCriteria $criteria = NULL)
    Get objects from database to populate widget
```

TDBCombo extends TCombo

Database ComboBox Widget

Methods

```
public function __construct($name, $database, $model, $key, $value, $ordercolumn = NULL, ?TCriteria $criteria = NULL)
```

Class Constructor

\$name : widget's name

\$database : database name

\$model : model class name

\$key : table field to be used as key in the combo

\$value : table field to be listed in the combo

\$ordercolumn : column to order the fields (optional)

\$criteria : criteria (TCriteria object) to filter the model (optional)

```
public static function reloadFromModel($formname, $field, $database, $model, $key, $value, $ordercolumn = NULL, $criteria = NULL, $startEmpty = FALSE, $fire_events = TRUE)
```

Reload combo from model data

\$formname : form name

\$field : field name

\$database : database name

\$model : model class name

\$key : table field to be used as key in the combo

\$value : table field to be listed in the combo

\$ordercolumn : column to order the fields (optional)

\$criteria : criteria (TCriteria object) to filter the model (optional)

\$startEmpty : if the combo will have an empty first item

\$fire_events : if change action will be fired

```
public static function getItemsFromModel($database, $model, $key, $value, $ordercolumn = NULL, ?TCriteria $criteria = NULL)
```

Get items (key/value) from database to populate widget

```
public static function getObjectsFromModel($database, $model, $key, $ordercolumn = NULL, ?TCriteria $criteria = NULL)
```

Get objects from database to populate widget

TDBEntry extends TEntry

Database Entry Widget

Methods

```
public function __construct($name, $database, $model, $value, $orderColumn = NULL, ?TCriteria $criteria = NULL)
```

Class Constructor

\$name : widget's name

\$database : database name

\$model : model class name

\$value : table field to be listed in the combo
\$ordercolumn : column to order the fields (optional)
\$criteria : criteria (TCriteria object) to filter the model (optional)

public function **setDisplayMask**(\$mask)
 Define the display mask
\$mask : Show mask

public function **setService**(\$service)
 Define the search service
\$service : Search service

public function **setMinLength**(\$length)
 Define the minimum length for search

public function **setOperator**(\$operator)
 Define the search operator
\$operator : Search operator

public function **show**()
 Shows the widget

TDBMultiCombo extends TDBSelect

DBMultiCombo Widget

Methods

public function **__construct**(\$name, \$database, \$model, \$key, \$value, \$ordercolumn = NULL, ?TCriteria \$criteria = NULL)
 Class Constructor
\$name : widget's name

public function **show**()
 Shows the widget

TDBMultiSearch extends TMultiSearch

Database Multisearch Widget

Methods

public function **__construct**(\$name, \$database, \$model, \$key, \$value, \$orderColumn = NULL, ?TCriteria \$criteria = NULL)
 Class Constructor
\$name : widget's name
\$database : database name
\$model : model class name
\$key : table field to be used as key in the combo
\$value : table field to be listed in the combo
\$ordercolumn : column to order the fields (optional)
\$criteria : criteria (TCriteria object) to filter the model (optional)

public function **setService**(\$service)
 Define the search service
\$service : Search service

public function **disableIdSearch**()
 Disable search by id

public function **enableIdTextualSearch**()
 Enable Id textual search

public function setOperator(\$operator)

Define the search operator

\$operator : Search operator

public function setMask(\$mask)

Define the display mask

\$mask : Show mask

public function setFilterColumns(\$columns)

Define the filter columns

\$columns :

public function setValue(\$values)

Define the field's value

\$values : An array the field's values

public function getPostData()

Return the post data

public function show()

Shows the widget

TDBRadioGroup extends TRadioGroup

Database Radio Widget

Methods

public function __construct(\$name, \$database, \$model, \$key, \$value, \$ordercolumn = NULL, ?TCriteria \$criteria = NULL)

Class Constructor

\$name : widget's name

\$database : database name

\$model : model class name

\$key : table field to be used as key in the combo

\$value : table field to be listed in the combo

\$ordercolumn : column to order the fields (optional)

\$criteria : criteria (TCriteria object) to filter the model (optional)

public static function reloadFromModel(\$formname, \$field, \$database, \$model, \$key, \$value, \$ordercolumn = NULL, \$criteria = NULL, \$options = [])

Reload radio from model data

\$formname : form name

\$field : field name

\$database : database name

\$model : model class name

\$key : table field to be used as key in the radio

\$value : table field to be listed in the radio

\$ordercolumn : column to order the fields (optional)

\$criteria : criteria (TCriteria object) to filter the model (optional)

\$options : array of options [layout, breakItems, useButton, value, changeAction, changeFunction]

public static function getItemsFromModel(\$database, \$model, \$key, \$value, \$ordercolumn = NULL, ?TCriteria \$criteria = NULL)

Get items (key/value) from database to populate widget

public static function getObjectsFromModel(\$database, \$model, \$key, \$ordercolumn = NULL, ?TCriteria \$criteria = NULL)

Get objects from database to populate widget

TDBSeekButton extends TSeekButton

Abstract Record Lookup Widget: Creates a lookup field used to search values from associated entities

Methods

```
public function __construct($name, $database, $form, $model, $display_field, $receive_key = null,
    $receive_display_field = null, ?TCriteria $criteria = NULL, $operator = 'like')
```

Class Constructor

\$name : name of the form field

\$database : name of the database connection

\$form : name of the parent form

\$model : name of the Active Record to be searched

\$display_field : name of the field to be searched and shown

\$receive_key : name of the form field to receive the primary key

\$receive_display_field : name of the form field to receive the "display field"

```
public function setCriteria(TCriteria $criteria)
```

Set search criteria

```
public function setOperator($operator)
```

Set operator

```
public function setDisplayMask($mask)
```

Set display mask

\$mask : Display mask

```
public function setDisplayLabel($label)
```

Set display label

\$mask : Display label

```
public function setValue($value)
```

Define the field's value

\$value : Current value

TDBSelect extends TSelect

Database Select Widget

Methods

```
public function __construct($name, $database, $model, $key, $value, $ordercolumn = NULL, ?TCriteria $criteria =
    NULL)
```

Class Constructor

\$name : widget's name

\$database : database name

\$model : model class name

\$key : table field to be used as key in the combo

\$value : table field to be listed in the combo

\$ordercolumn : column to order the fields (optional)

\$criteria : criteria (TCriteria object) to filter the model (optional)

```
public static function getItemsFromModel($database, $model, $key, $value, $ordercolumn = NULL, ?TCriteria $criteria =
    NULL)
```

Get items (key/value) from database to populate widget

```
public static function getObjectsFromModel($database, $model, $key, $ordercolumn = NULL, ?TCriteria $criteria =
    NULL)
```

Get objects from database to populate widget

TDBSortList extends TSortList

Database Sortlist Widget

Methods

```
public function __construct($name, $database, $model, $key, $value, $ordercolumn = NULL, ?TCriteria $criteria = NULL)
```

Class Constructor

\$name : widget's name

\$database : database name

\$model : model class name

\$key : table field to be used as key in the combo

\$value : table field to be listed in the combo

\$ordercolumn : column to order the fields (optional)

\$criteria : criteria (TCriteria object) to filter the model (optional)

```
public static function getItemsFromModel($database, $model, $key, $value, $ordercolumn = NULL, ?TCriteria $criteria = NULL)
```

Get items (key/value) from database to populate widget

```
public static function getObjectsFromModel($database, $model, $key, $ordercolumn = NULL, ?TCriteria $criteria = NULL)
```

Get objects from database to populate widget

TDBUniqueSearch extends TDBMultiSearch

DBUnique Search Widget

Methods

```
public function __construct($name, $database, $model, $key, $value, $orderColumn = NULL, ?TCriteria $criteria = NULL)
```

Class Constructor

\$name : Widget's name

```
public function setValue($value)
```

Define the field's value

\$value : Current value

```
public function getPostData()
```

Return the post data

```
public function getSize()
```

Returns the size

```
public function show()
```

Show the component

TDataGrid extends TTable

DataGrid Widget: Allows to create datagrids with rows, columns and actions

Methods

```
public function __construct()
```

Class Constructor

```
public function setId($id)
```

Set id

```
public function setMutationAction(TAction $action)
```

Define mutation action

```
public function setActionSide($side)
```

Set actions side

```
public function generateHiddenFields()
```

Generate hidden fields

public function **disableHtmlConversion()**
Disable htmlspecialchars on output

public function **getOutputData()**
Get raw processed output data

public function **enablePopover**(\$title, \$content, \$popside = null, \$popcondition = null)
Enable popover
\$title : Title
\$content : Content

public function **makeScrollable()**
Make the datagrid scrollable

public function **isScrollable()**
Returns if datagrid is scrollable

public function **setActionWidth**(\$width)
Set the column action width

public function **disableDefaultClick()**
disable the default click action

public function **setHeight**(\$height)
Define the Height
\$height : An integer containing the height

public function **getHeight()**
Return datagrid height

public function **addColumn**(TDataGridColumn \$object, ?TAction \$action = null)
Add a Column to the DataGrid
\$object : A TDataGridColumn object

public function **getColumns()**
Returns an array of TDataGridColumn

public function **addAction**(TDataGridAction \$action, \$label = null, \$image = null)
Add an Action to the DataGrid
\$object : A TDataGridAction object

public function **setActions**(\$actions)
Set actions to the DataGrid
\$actions : TDataGridAction objects

public function **prepareForPrinting()**
Prepare for printing

public function **setPageSize**(\$page_size)
Set page Size
\$page_size : (a3,a4,a5,letter,legal)

public function **getPageSize()**
Return the page size

public function **setPageOrientation**(\$page_orientation)
Set page orientation
\$page_orientation : (portrait, landscape)

public function **getPageOrientation()**
Return the page orientation

public function **addActionGroup**(TDataGridActionGroup \$object)
Add an Action Group to the DataGrid
\$object : A TDataGridActionGroup object

public function **getTotalColumns()**
Returns the total columns

public function **setGroupColumn**(\$column, \$mask, \$transformer = null)
Set the group column for break

public function **useGroupTotal**(\$groupTotal = null)
Set the group column for break

public function **clear**(\$preserveHeader = TRUE, \$rows = 0)
Clear the DataGrid contents

public function **createModel**(\$create_header = true)
Creates the DataGrid Structure

public function **getHead()**
Return thead

public function **getBody()**
Return tbody

public function **prependRow**(\$row)
Prepend row

public function **insert**(\$position, \$content)
insert content

public function **addItems**(\$objects)
Add objects to the DataGrid
\$objects : An array of Objects

public function **addItem**(\$object)
Add an object to the DataGrid
\$object : An Active Record Object

public static function **appendRow**(\$table_id, \$row)
Append table row via Javascript

public static function **removeRowById**(\$table_id, \$id)
Remove row by id

public static function **replaceRowById**(\$table_id, \$id, \$row)
Replace row by id

public function **getItems()**
Return datagrid items

public function **getRowIndex**(\$attribute, \$value)
Find the row index by object attribute
\$attribute : Object attribute
\$value : Object value

public function **getRow**(\$position)
Return the row by position

\$position : Row position

public function getWidth()
Returns the DataGrid's width

public function setPageNavigation(\$pageNavigation)
Assign a PageNavigation object
\$pageNavigation : object

public function getPageNavigation()
Return the assigned PageNavigation object

public function setSearchForm(\$searchForm)
Assign a TForm object
\$searchForm : object

public function getSearchForm()
Return the assigned Search form object

public function setSearchAttributes(\$attributes)
Set serach attributes

public function enableSearch(TField \$input, \$attributes)
Enable fuse search
\$input : Field input for search
\$attribute : Attribute name

public function show()
Shows the DataGrid

TDataGridAction extends TAction

Represents an action inside a datagrid

Methods

public function __construct(\$action, \$parameters = null)
Class Constructor
\$action : Callback to be executed
\$parameters : = array of parameters

public function enablePost()
Change the request method

public function setField(\$field)
Define wich Active Record's property will be passed along with the action
\$field : Active Record's property

public function setFields(\$fields)
Define wich Active Record's properties will be passed along with the action
\$field : Active Record's property

public function getField()
Returns the Active Record's property that will be passed along with the action

public function getFields()
Returns the Active Record's properties that will be passed along with the action

public function fieldDefined()
Return if there at least one field defined

public function setImage(\$image)

Define an icon for the action

\$image : The Image path

public function getImage()
Returns the icon of the action

public function setLabel(\$label)
define the label for the action
\$label : A string containing a text label

public function setTitle(\$title)
define the title for the action
\$label : A string containing a text label

public function getLabel()
Returns the text label for the action

public function getTitle()
Returns the tooltip title for the action

public function setButtonClass(\$buttonClass)
define the buttonClass for the action
\$buttonClass : A string containing the button css class

public function getButtonClass()
Returns the buttonClass

public function setUseButton(\$useButton)
define if the action will use a regular button
\$useButton : A boolean

public function getUseButton()
Returns if the action will use a regular button

public function setDisplayCondition(/*Callable*/ \$displayCondition)
Define a callback that must be valid to show the action
Callback : \$displayCondition Action display condition

public function getDisplayCondition()
Returns the action display condition

public function prepare(\$object)
Prepare action for use over an object
\$object : Data Object

public function serialize(\$format_action = TRUE, \$check_permission = FALSE)
Converts the action into an URL
\$format_action : = format action with document or javascript (ajax=no)

TDataGridActionGroup

Represents a group of Actions for datagrids

Methods

public function __construct(\$label, \$icon = NULL)
Constructor
\$label : Action Group label
\$icon : Action Group icon

public function getLabel()
Returns the Action Group label

public function `getIcon()`
Returns the Action Group icon

public function `addAction(TAction $action)`
Add an action to the actions group
***\$action** : TAction object*

public function `addSeparator()`
Add a separator

public function `addHeader($header)`
Add a header
***\$header** : Options header*

public function `getActions()`
Returns the actions

public function `getHeaders()`
Returns the headers

public function `getSeparators()`
Returns the separators

TDataGridColumn

Representes a DataGrid column

Methods

public function `__construct($name, $label, $align, $width = NULL)`
Class Constructor
***\$name** : = Name of the column in the database*
***\$label** : = Text label that will be shown in the header*
***\$align** : = Column align (left, center, right)*
***\$width** : = Column Width (pixels)*

public function `setVisibility($bool)`
Define column visibility

public function `enableAutoHide($width)`
Enable column auto hide

public function `disablePrinting()`
Disable in exporting formats

public function `enableSearch()`
Enable column search

public function `enableHtmlConversion()`
Enable htmlspecialchars on output

public function `disableHtmlConversion()`
Disable htmlspecialchars on output

public function `hasHtmlConversionEnabled()`
return if has html conversion

public function `getInputSearch()`
Get input search

public function `isSearchable()`

Returns if column is searchable

public function isPrintable()

Returns if column is printable

public function setProperty(\$name, \$value)

Define a column header property

\$name : Property Name

\$value : Property Value

public function setDataProperty(\$name, \$value)

Define a data property

\$name : Property Name

\$value : Property Value

public function getProperty(\$name)

Return a column property

\$name : Property Name

public function getDataProperty(\$name)

Return a data property

\$name : Property Name

public function getProperties()

Return column properties

public function getDataProperties()

Return data properties

public function getName()

Returns the database column's name

public function getLabel()

Returns the column's label

public function setLabel(\$label)

Set the column's label

\$label : column label

public function getAlign()

Returns the column's align

public function getWidth()

Returns the column's width

public function setAction(TAction \$action, \$parameters = null)

Define the action to be executed when the user clicks over the column header

\$action : TAction object

\$parameters : Action parameters

public function getAction()

Returns the action defined by set_action() method

public function removeAction()

Remove action

public function setEditAction(TDataGridAction \$editaction)

Define the action to be executed when the user clicks do edit the column

\$action : A TDataGridAction object

public function getEditAction()

Returns the action defined by setEditAction() method

public function **setTransformer**(Callable \$callback)

Define a callback function to be applied over the column's data

\$callback : A function name of a method of an object

public function **getTransformer**()

Returns the callback defined by the setTransformer()

public function **enableTotal**(\$function, \$prefix = null, \$decimals = 2, \$decimal_separator = ',', \$thousand_separator = '.')

Enable total

public function **setTotalFunction**(Callable \$callback, \$apply_transformer = true)

Define a callback function to totalize column

\$callback : A function name of a method of an object

\$apply_transformer : Apply transform function also in total

public function **getTotalCallback**()

Returns the callback defined by the setTotalFunction()

public function **getTotalFunction**()

Returns total function

public function **getTotalMask**()

Returns total mask

public function **totalTransformed**()

Is total transformed

TDatabase

Database Task manager

Methods

public static function **dropTable**(\$conn, \$table, \$ifexists = false)

Drop table

\$conn : Connection

\$table : Table name

\$ifexists : Drop only if exists

public static function **createTable**(\$conn, \$table, \$columns)

Create table

\$conn : Connection

\$table : Table name

\$columns : Array of columns

public static function **dropColumn**(\$conn, \$table, \$column)

Drop column

\$conn : Connection

\$table : Table name

\$column : Column name

public static function **addColumn**(\$conn, \$table, \$column, \$type, \$options)

Add column

\$conn : Connection

\$table : Table name

\$column : Column name

\$type : Column type

\$options : Column options

public static function **insertData**(\$conn, \$table, \$values, \$avoid_criteria = null)

Insert data

\$conn : Connection
\$table : Table name
\$values : Array of values
\$avoid_criteria : Criteria to avoid insertion

public static function **updateData**(\$conn, \$table, \$values, \$criteria = null)
 Update data

\$conn : Connection
\$table : Table name
\$values : Array of values
\$avoid_criteria : Criteria to avoid insertion

public static function **clearData**(\$conn, \$table, \$criteria = null)
 Clear table data

\$conn : Connection
\$table : Table name
\$criteria : Filter criteria

public static function **execute**(\$conn, \$query)
 Execute SQL

\$conn : Connection
\$query : SQL

public static function **getData**(\$conn, \$query, \$mapping = null, \$prepared_values = null, ?Closure \$action = null)
 Get RAW Data

\$conn : Connection
\$query : SQL
\$mapping : Mapping between fields
\$prepared_values : Parameters for SQL Query

public static function **getRowData**(PDO \$conn, \$table, \$criteria = null)
 Get a row from the table

\$conn : PDO source connection
\$table : Source table
\$criteria : Filter criteria

public static function **countData**(PDO \$conn, \$table, \$criteria = null)
 Count data from table

\$conn : PDO source connection
\$table : Source table
\$criteria : Filter criteria

public static function **copyData**(PDO \$source_conn, PDO \$target_conn, \$source_table, \$target_table, \$mapping, \$criteria = null, \$bulk_inserts = 1, \$auto_commit = false)
 Copy data from table to table

\$source_conn : PDO source connection
\$target_conn : PDO target connection
\$source_table : Source table
\$target_table : Target table
\$mapping : Mapping between fields
\$criteria : Filter criteria
\$bulk_inserts : Inserts per time
\$auto_commit : Auto commit after x inserts

public static function **copyQuery**(PDO \$source_conn, PDO \$target_conn, \$query, \$target_table, \$mapping, \$prepared_values = null, \$bulk_inserts = 1, \$auto_commit = false)
 Copy data from query to table

\$source_conn : PDO source connection
\$target_conn : PDO target connection
\$query : SQL Query
\$target_table : Target table
\$mapping : Mapping between fields
\$prepared_values : Parameters for SQL Query
\$bulk_inserts : Inserts per time

\$auto_commit : Auto commit after x inserts

```
public static function importFromFile($filename, $target_conn, $target_table, $mapping, $separator = ',', $bulk_inserts = 1)
```

Import data from CSV file

\$filename : CSV File to import

\$target_conn : Target connection

\$target_table : Target table

\$mapping : Mapping between fields

\$separator : Columns separator [,]

```
public static function exportToFile($source_conn, $source_table, $filename, $mapping, $criteria = null, $separator = ',')
```

Export data to CSV file

\$source_conn : Source connection

\$source_table : Target table

\$filename : CSV File to import

\$mapping : Mapping between fields

\$criteria : Select criteria

\$separator : Columns separator [,]

TDate extends TEntry

DatePicker Widget

Methods

```
public function __construct($name)
```

Class Constructor

\$name : Name of the widget

```
public function setValue($value)
```

Store the value inside the object

```
public function getPostData()
```

Return the post data

```
public static function convertToMask($value, $fromMask, $toMask)
```

Convert from one mask to another

\$value : original date

\$fromMask : source mask

\$toMask : target mask

```
public function setMask($mask, $replaceOnPost = FALSE)
```

Define the field's mask

\$mask : Mask for the field (dd-mm-yyyy)

```
public function getMask()
```

Return mask

```
public function setDatabaseMask($mask)
```

Set the mask to be used to collect the data

```
public function getDatabaseMask()
```

Return database mask

```
public function setOption($option, $value)
```

Set extra datepicker options

```
public function setExitAction(TAction $action)
```

Define the action to be executed when the user changes the field

\$action : TAction object

```
public function setChangeAction(TAction $action)
```

Define the action to be executed when the user changes the field

\$action : TAction object

public function **setChangeFunction**(\$function)
Set change function

public static function **date2us**(\$date)
Shortcut to convert a date to format yyyy-mm-dd
\$date := date in format dd/mm/yyyy

public static function **date2br**(\$date)
Shortcut to convert a date to format dd/mm/yyyy
\$date := date in format yyyy-mm-dd

public static function **enableField**(\$form_name, \$field)
Enable the field
\$form_name : Form name
\$field : Field name

public static function **disableField**(\$form_name, \$field)
Disable the field
\$form_name : Form name
\$field : Field name

public function **show**()
Shows the widget at the screen

TDateTime extends TEntry

DateTimePicker Widget

Methods

public function **__construct**(\$name)
Class Constructor
\$name : Name of the widget

public function **setValue**(\$value)
Store the value inside the object

public function **getPostData**()
Return the post data

public static function **convertToMask**(\$value, \$fromMask, \$toMask)
Convert from one mask to another
\$value : original date
\$fromMask : source mask
\$toMask : target mask

public function **setChangeFunction**(\$function)
Set change function

public function **setExitAction**(TAction \$action)
Define the action to be executed when the user changes the field
\$action : TAction object

public function **setChangeAction**(TAction \$action)
Define the action to be executed when the user changes the field
\$action : TAction object

public function **setMask**(\$mask, \$replaceOnPost = FALSE)
Define the field's mask

\$mask : Mask for the field (dd-mm-yyyy)

public function setDatabaseMask(\$mask)
Set the mask to be used to collect the data

public function setOption(\$option, \$value)
Set extra datepicker options

public static function enableField(\$form_name, \$field)
Enable the field
\$form_name : Form name
\$field : Field name

public static function disableField(\$form_name, \$field)
Disable the field
\$form_name : Form name
\$field : Field name

public function show()
Shows the widget at the screen

TDropDown extends TElement

TDropDown Widget

Methods

public function __construct(\$label, \$icon = NULL, \$use_caret = FALSE, \$title = "", \$height = null)
Class Constructor
\$title : Dropdown title
\$icon : Dropdown icon

public function setPullSide(\$side)
Define the pull side

public function setButtonSize(\$size)
Define the button size

public function setButtonClass(\$class)
Define the button class

public function getButton()
Returns the dropdown button

public function addAction(\$title, \$action, \$icon = NULL, \$popover = "", \$add = true)
Add an action
\$title : Title
\$action : Action (TAction or string Javascript action)
\$icon : Icon

public function addPostAction(\$title, \$action, \$form, \$icon = NULL, \$popover = "", \$add = true)
Add an action
\$title : Title
\$action : Action (TAction or string Javascript action)
\$icon : Icon

public function addActionGroup(\$title, \$actions, \$icon)
Add an action group

public function addHeader(\$header)
Add a header
\$header : Options Header

public function **addSeparator()**
Add a separator

public function **clearItems()**
Clear child elements

public function **getItems()**
Get dropdown items

TElement

Base class for all HTML Elements

Methods

public function **__construct(\$tagname)**
Class Constructor
\$tagname : tag name

public function **enableVoidElement()**
Turn element into a void element

public static function **tag(\$tagname, \$value, \$attributes = NULL)**
Create an element
\$tagname : Element name
\$value : Element value
\$attributes : Element attributes

public function **hide()**
hide object

public function **after(\$element)**
Insert element after

public function **getAfterElement()**
Return the after element

public function **setName(\$tagname)**
Change the element name
\$tagname : Element name

public function **getName()**
Returns tag name

public function **setIsWrapped(\$bool)**
Define if the element is wrapped inside another one
@bool : Boolean TRUE if is wrapped

public function **getIsWrapped()**
Return if the element is wrapped inside another one

public function **setProperty(\$name, \$value)**
Set tag property
\$name : Property Name
\$value : Property Value

public function **setProperties(\$properties)**
Set element properties

public function **getProperty(\$name)**
Return a property

\$name : property name

public function **getProperties()**
Return element properties

public function **add(\$child)**
Add an child element
\$child : Any object that implements the show() method

public function **addMany(\$children)**
Add many children
\$children : Array of child

public function **insert(\$position, \$child)**
Insert an child element
\$position : Element position
\$child : Any object that implements the show() method

public function **setUseLineBreaks(\$linebreaks)**
Set the use of linebreaks
\$linebreaks : boolean

public function **setUseSingleQuotes(\$singlequotes)**
Set the use of single quotes
\$singlequotes : boolean

public function **del(\$object)**
Del an child element
\$child : Any object that implements the show() method

public function **getChildren()**
get children

public function **find(\$element, \$properties = null)**
Find child element
\$element : tag name
\$properties : match properties

public function **get(\$position)**
Get an child element
\$position : Element position

public function **openTag()**
Opens the tag

public function **open()**
BC only

public function **show()**
Shows the tag

public function **closeTag()**
Closes the tag

public function **close()**
BC only

public function **getContents()**
Returns the element content as a string

public function **clearChildren()**
Clear element children

TEmailValidator extends TFieldValidator

Email validation

Methods

public function validate(\$label, \$value, \$parameters = NULL)
Validate a given value
\$label : Identifies the value to be validated in case of exception
\$value : Value to be validated
\$parameters : additional parameters for validation

TEntry extends TField

Entry Widget

Methods

public function __construct(\$name)
Class Constructor
\$name : name of the field

public function enableToggleVisibility(\$toggleVisibility = TRUE)
Enable toggle visible

public function setInputType(\$type)
Define input type

public function setInnerIcon(TImage \$image, \$side = 'right')
Define the Inner icon

public function exitOnEnter()
Turn on exit on enter

public function setMask(\$mask, \$replaceOnPost = FALSE)
Define the field's mask
\$mask : A mask for input data

public function setNumericMask(\$decimals, \$decimalsSeparator, \$thousandSeparator, \$replaceOnPost = FALSE, \$reverse = FALSE, \$allowNegative = TRUE)
Define the field's numeric mask (available just in web)
\$decimals : Sets the number of decimal points.
\$decimalsSeparator : Sets the separator for the decimal point.
\$thousandSeparator : Sets the thousands separator.
\$allowNegative : Sets negative allowed.

public function setValue(\$value)
Define the field's value
\$value : A string containing the field's value

public function getPostData()
Return the post data

public function setMaxLength(\$length)
Define max length
\$length : Max length

function setCompletion(\$options)
Define options for completion
\$options : array of options for completion

function setExitAction(TAction \$action)
Define the action to be executed when the user leaves the form field

\$action : TAction object

public function **setExitFunction**(\$function)

Define the javascript function to be executed when the user leaves the form field

\$function : Javascript function

public function **disableAutoComplete**()

Disable auto complete

public function **forceLowerCase**()

Force lower case

public function **forceUpperCase**()

Force upper case

public function **setDelimiter**(\$delimiter)

Set autocomplete delimiter

\$delimiter : autocomplete delimiter

public function **setMinLength**(\$length)

Define the minimum length for search

public static function **reloadCompletion**(\$field, \$list, \$options = null)

Reload completion

\$field : Field name or id

\$options : array of options for autocomplete

protected function **formatMask**(\$mask, \$value)

Apply mask

\$mask : Mask

\$value : Value

public static function **changeMask**(\$formName, \$name, \$mask)

Change mask dynamically

public function **show**()

Shows the widget at the screen

TExceptionView

Exception visualizer

Methods

function **__construct**(Throwable \$e)

Constructor method

TExpander extends TElement

Expander Widget

Methods

public function **__construct**(\$label = "")

Class Constructor

\$value : text label

public function **setCaretSide**(\$caret_side)

Set caret side

public function **setPullSide**(\$side)

Define the pull side

public function **setButtonProperty**(\$property, \$value)

Define a button property

\$property : Property name (Ex: style)

\$value : Property value

public function **setProperty**(\$property, \$value)

Define a container property

\$property : Property name (Ex: style)

\$value : Property value

public function **add**(\$content)

Add content to the expander

\$content : Any Object that implements show() method

public function **show**()

Shows the expander

TExpression

Base class for TCriteria and TFilter (composite pattern implementation)

Methods

abstract public function **dump**();

TField

Base class to construct all the widgets

Methods

public function **__construct**(\$name)

Class Constructor

\$name : name of the field

public function **setTagName**(\$name)

Change input tag name

public function **setValueCallback**(\$callback)

Set callback for setValue method

public function **setLabel**(\$label)

Define the field's label

\$label : A string containing the field's label

public function **getLabel**()

Returns the field's label

public function **setName**(\$name)

Define the field's name

\$name : A string containing the field's name

public function **getName**()

Returns the field's name

public function **setId**(\$id)

Define the field's id

\$id : A string containing the field's id

public function **getId**()

Returns the field's id

```

public function setValue($value)
    Define the field's value
    $value : A string containing the field's value

public function getValue()
    Returns the field's value

public function getFormName()
    Return the name of the form to wich the field is attached

public function setTip($tip)
    Define the field's tooltip
    $name : A string containing the field's tooltip

public function getPostData()
    Return the post data

public function setEditable($editable)
    Define if the field is editable
    $editable : A boolean

public function getEditable()
    Returns if the field is editable

public function setProperty($name, $value, $replace = TRUE)
    Define a field property
    $name : Property Name
    $value : Property Value

public function getPropertiesAsString($filter = null)
    Get properties as string

public function getProperty($name)
    Return a field property
    $name : Property Name
    $value : Property Value

public function setSize($width, $height = NULL)
    Define the Field's width
    $width : Field's width in pixels

public function getSize()
    Returns the field size

public function addValidation($label, TFieldValidator $validator, $parameters = NULL)
    Add a field validator
    $label : Field name
    $validator : TFieldValidator object
    $parameters : Additional parameters

public function getValidations()
    Returns field validations

public function isRequired()
    Returns if the field is required

public function validate()
    Validate a field

public function getContents()
    Returns the element content as a string

```

public static function enableField(\$form_name, \$field)

Enable the field

\$form_name : Form name

\$field : Field name

public static function disableField(\$form_name, \$field)

Disable the field

\$form_name : Form name

\$field : Field name

public static function clearField(\$form_name, \$field)

Clear the field

\$form_name : Form name

\$field : Field name

TFieldList extends TTable

Create a field list

Methods

public function __construct()

Class Constructor

public function disablePostEmptyRow()

public function getPostData()

Get post data as object list

public function getRowCount(\$field_name = null)

Get post row count

public function disableRemoveButton()

Disable remove button

public function enableSorting()

Enable sorting

public function generateAria()

Generate automatic aria-labels

public function setSortAction(TAction \$action)

Define the action to be executed when the user sort rows

\$action : TAction object

public function setRemoveFunction(\$action, \$icon = null, \$title = null)

Set the remove javascript action

public function setRemoveAction(?TAction \$action = null, \$icon = null, \$title = null)

Set the remove action

public function setCloneFunction(\$action)

Set the clone javascript action

public function addButtonFunction(\$function, \$icon, \$title)

Add function

public function addButtonAction(TAction \$action, \$icon, \$title)

Add action

public function setTotalUpdateAction(TAction \$action)

Define total update action

```
public function setFieldPrefix($prefix)
    Set field prefix
```

```
public function getFieldPrefix()
    Get field prefix
```

```
public function addField($label, AdiantiWidgetInterface $field, $properties = null)
    Add a field
    $label : Field Label
    $object : Field Object
```

```
public function addHeader()
    Add table header
```

```
public function addDetail( $item, ?Callable $callback = null )
    Add detail row
    $item : Data object
```

```
public function addCloneAction(?TAction $clone_action = null, $icon = null, $title = null)
    Add clone action
```

```
public function addTotalFields($row)
```

```
public static function clear($name)
    Clear field list
    $name : field list name
```

```
public static function clearRows($name, $start = 0, $length = 0)
    Clear some field list rows
    $name : field list name
    $index : field list name
    $quantity : field list name
```

```
public static function addRows($name, $rows, $timeout = 50)
    Add rows on field list
    $name : field list name
    $rows : quantity rows
    $timeout : timeout
```

```
public function makeScrollable($height)
    Enable scrolling
```

```
public function getHead()
    Get head
```

```
public function getFoot()
    Get foot
```

```
public function getBody()
    Get body
```

```
public static function enableField($field)
    Enable the field
    $form_name : Form name
    $field : Field name
```

```
public static function disableField($field)
    Disable the field
    $form_name : Form name
    $field : Field name
```

```
public function show()
```

Show component

TFieldValidator

TFieldValidator abstract validation class

Methods

abstract public function **validate**(\$label, \$value, \$parameters = NULL);
Validate a given value
\$label : Identifies the value to be validated in case of exception
\$value : Value to be validated
\$parameters : additional parameters for validation

TFile extends TField

FileChooser widget

Methods

public function **__construct**(\$name)
Constructor method
\$name : input name

public function **enableImageGallery**(\$width = null, \$height = 100)
Enable image gallery view

public function **enablePopover**(\$title = null, \$content = "")
Enable popover
\$title : Title
\$content : Content

public function **setLimitUploadSize**(\$limit)
Define upload size limit
\$limit : Size limit MBs

public function **enablePHPFileUploadLimit**()
Define upload size limit

public function **setDisplayMode**(\$mode)
Define the display mode {file}

public function **setService**(\$service)
Define the service class for response

public function **setAllowedExtensions**(\$extensions)
Define the allowed extensions

public function **enableFileHandling**()
Define to file handling

public function **disableFileHandling**()
Disable file handling

public function **setPlaceholder**(TElement \$widget)
Set place holder

public function **setSize**(\$width, \$height = NULL)
Set field size

public function **setHeight**(\$height)

Set field height

```
public function getPostData()
    Return the post data
```

```
public function setValue($value)
    Set field value
```

```
public function show()
    Show the widget at the screen
```

```
public function setCompleteAction(TAction $action)
    Define the action to be executed when upload is finished
    $action : TAction object
```

```
public function setErrorAction(TAction $action)
    Define the action to be executed when some error occurs
    $action : TAction object
```

```
public static function enableField($form_name, $field)
    Enable the field
    $form_name : Form name
    $field : Field name
```

```
public static function disableField($form_name, $field)
    Disable the field
    $form_name : Form name
    $field : Field name
```

```
public static function clearField($form_name, $field)
    Clear the field
    $form_name : Form name
    $field : Field name
```

TFilter extends TExpression

Provides an interface to define filters to be used inside a criteria

Methods

```
public function __construct($variable, $operator, $value, $value2 = NULL)
    Class Constructor
    $variable : = variable
    $operator : = operator (>,
    $value : = value to be compared
    $value2 : = second value to be compared (between)
```

```
public function getPreparedVars()
    Return the prepared vars
```

```
public function dump( $prepared = FALSE )
    Return the filter as a string expression
```

```
public function setCaseInsensitive(bool $value) : void
    Force case insensitive searches
```

```
public function getCaseInsensitive() : bool
    Return if case insensitive is turned ON
```

TForm

Wrapper class to deal with forms

Methods

public function **__construct**(\$name = 'my_form')

Class Constructor

\$name : Form Name

public function **setTagName**(\$name)

Change tag name

public function **silentField**(\$name)

Silent field

public function **setProperty**(\$name, \$value, \$replace = TRUE)

Define a form property

\$name : Property Name

\$value : Property Value

public function **setVirtualProperty**(\$name, \$value)

Set Virtual Property

public function **getVirtualProperty**(\$name)

Get Virtual Property

public function **unsetProperty**(\$name)

Unset form property

public static function **getFormByName**(\$name)

Returns the form object by its name

public function **setName**(\$name)

Define the form name

\$name : A string containing the form name

public function **getName**()

Returns the form name

public static function **sendData**(\$form_name, \$object, \$aggregate = FALSE, \$fireEvents = TRUE, \$timeout = 0)

Send data for a form located in the parent window

\$form_name : Form Name

\$object : An Object containing the form data

public function **setEditable**(\$bool)

Define if the form will be editable

\$bool : A Boolean

public function **addField**(AdiantiWidgetInterface \$field)

Add a Form Field

\$field : Object

public function **delField**(AdiantiWidgetInterface \$field)

Remove a form field

\$field : Object

public function **delFields**()

Remove all form fields

public function **setFields**(\$fields)

Define wich are the form fields

\$fields : An array containing a collection of TField objects

```

public function getField($name)
    Returns a form field by its name
    $name : A string containing the field's name

public function getFields()
    Returns an array with the form fields

public function clear($keepDefaults = FALSE)
    clear the form Data

public function setData($object)
    Define the data of the form
    $object : An Active Record object

public function getData($class = 'StdClass')
    Returns the form POST data as an object
    $class : A string containing the class for the returning object

public function getValues($class = 'StdClass', $withOptions = false)
    Returns the form start values as an object
    $class : A string containing the class for the returning object

public function validate()
    Validate form

public function add($object, $is_field = false)
    Add a container to the form (usually a table or panel)
    $object : Any Object that implements the show() method

public function pack()
    Pack a container to the form (usually a table or panel)
    mixed : $object, ...Any Object that implements the show() method

public function getChild()
    Returns the child object

public static function enableField($form_name, $field, $timeout = null)
    Enable the field
    $form_name : Form name
    $field : Field name

public static function disableField($form_name, $field, $timeout = null)
    Disable the field
    $form_name : Form name
    $field : Field name

public function show()
    Shows the form at the screen

```

TFormSeparator extends TElement

Form separator

Methods

```

public function __construct($text, $fontColor = 'var(--bs-body-color)', $fontSize = '16', $separatorColor = '#eeeeee')
    Class Constructor
    $text : Separator title

public function setFontSize($size)
    Set font size
    $size : font size

```

public function **setFontColor**(\$color)

Set font color

\$color : font color

public function **setSeparatorColor**(\$color)

Set separator color

\$color : separator color

TFrame extends TElement

Frame Widget: creates a bordered area with a title located at its top-left corner

Methods

public function **__construct**(\$width = NULL, \$height = NULL)

Class Constructor

\$value : text label

public function **getSize**()

Returns the frame size

public function **setLegend**(\$legend)

Set Legend

\$legend : frame legend

public function **getLegend**()

Returns the inner legend

TFullCalendar extends TElement

FullCalendar Widget

Methods

public function **__construct**(\$current_date = NULL, \$default_view = 'month')

Class Constructor

\$current_date : Current date of calendar

\$default_view : Default view (month, agendaWeek, agendaDay, listWeeky)

public function **setOption**(\$option, \$value)

Set extra datepicker options (ex: autoclose, startDate, daysOfWeekDisabled, datesDisabled)

public function **enableFullHeight**(\$full_height = TRUE)

Define use full height

public function **setHeight**(\$height)

Define height

public function **setTimeRange**(\$min_time, \$max_time)

Define the time range

public function **enableDays**(\$days)

Enable these days

public function **setCurrentDate**(\$date)

Set the current date of calendar

\$date : Current date of calendar

public function **setCurrentView**(\$view)

Set the current view of calendar

\$view : Current view of calendar (month, agendaWeek, agendaDay, listWeek)

public function **setReloadAction**(TAction \$action)

Define the reload action

\$action : reload action

public function **setEventClickAction**(TAction \$action)

Define the event click action

\$action : event click action

public function **setDayClickAction**(TAction \$action)

Define the day click action

\$action : day click action

public function **setEventUpdateAction**(TAction \$action)

Define the event update action

\$action : event updaet action

public function **enablePopover**(\$title, \$content)

Enable popover

\$title : Title

\$content : Content

public function **disableResizing**()

Disable event resize

public function **disableDragging**()

Disable event dragging

public function **disableWeekend**()

Set disable view weekend

public function **addEvent**(\$id, \$title, \$start, \$end = NULL, \$url = NULL, \$color = NULL, \$object = NULL)

Add an event

\$id : Event id

\$title : Event title

\$start : Event start time

\$end : Event end time

\$url : Event url

\$color : Event color

public function **setSearchForm**(\$searchForm)

Assign a TForm object

\$searchForm : object

public function **setEditForm**(\$editForm)

Assign a TForm object

\$editForm : object

public function **getSearchForm**()

Return the assigned Search form object

public function **getEditForm**()

Return the assigned Edit form object

public function **setMetadata**(\$metadata, \$value)

Set metadata

public function **getMetadata**(\$metadata)

Get metadata

public static function **renderPopover**(\$title, \$poptitle, \$popcontent)

Render title popover

\$title : Event title

\$poptitle : Popover Title
\$popcontent : Popover Content

public function show()
 Show the callendar and execute required scripts

TGantt extends TElement

TGantt

Methods

public function __construct(\$view_mode, \$size = 'md')
 Constructor method

public function setViewMode(\$view_mode)
 Change view mode

public function getViewMode()
 Get view mode

public static function renderPopover(\$title, \$poptitle, \$popcontent)
 Render title popover
\$title : Event title
\$poptitle : Popover Title
\$popcontent : Popover Content

public function setTitle(\$title)
 Define title of header gantt
\$title : String title header

public function setSizeMode(\$size)
 Define view mode
\$mode : String mode

public function getSizeMode()
 Get size mode

public function setTransformerTimeTitle(callable \$transformer)
 Define transformer time title \$transformer callable (start, end, events)

public function setTransformerEventLabel(callable \$transformEventLabel)
 Define transformer event label \$transformer callable (object event, events, times)

public function setStartDate(\$date)
 Define date start gantt
\$date : String date

public function getStartDate()
 Return date start gantt

public function getEndDate()
 Return date end gantt

public function setInterval(\$interval = '1 month')
 Define interval between dates
\$interval : String 1|2|4 number concat with type month|day|year e.g: 1 month, 10 days

public function removeSpaceBetweenEvents()
 Remove spaces between events

public function enableStripedMonths()

Add background striped on columns

public function **enableStripedRows()**
Add background striped on rows

public function **enableFullHours()**
Enable hour when MODE_DAYS_WITH_HOUR 24h

public function **setReloadAction**(TAction \$reloadAction)
Define reload events action
\$reloadAction : TAction

public function **setDayClickAction**(TAction \$action)
Define the day click action
\$action : day click action

public function **addHeaderAction**(TAction \$action, \$label = "", ?TImage \$icon = null)
Define reload events action
\$reloadAction : TAction Button action click
\$label : String Button label
\$icon : TImage Button icon

public function **addHeaderWidget**(\$widget)
Add a form header widget
\$widget : Widget

public function **addRow**(\$id, \$label)
Add new row
\$id : any Key of row
\$title : any Label of row

public function **clearEvents()**

public function **addEvent**(\$id, \$rowId, \$title, \$start_time, \$end_time, \$color = NULL, \$percent = null)
Add new event on Gantt
\$id : any Key of event
\$rowId : any Key of row
\$title : String title
\$start_time : String date start
\$end_time : String date end
\$color : String color background
\$percent : float percent color

public function **setEventClickAction**(\$action)
Define click event action
\$action : TAction

public function **enableDragEvent**(TAction \$updateAction, \$minutesStep = 1440)
Define drag event action
\$action : TAction

function **enableViewModeButton**(\$with_label = TRUE, \$with_icon = TRUE, \$label = NULL, \$icon = NULL)
Enable view_mode button

function **enableSizeModeButton**(\$with_label = TRUE, \$with_icon = TRUE, \$label = NULL, \$icon = NULL)
Enable size_mode button

public function **show()**

THBox extends TElement

Horizontal Box

Methods

public function `__construct()`
Class Constructor

public function `add($child, $style = 'display:inline-table;')`
Add an child element
\$child : Any object that implements the show() method

public function `addRowSet()`
Add a new row with many cells
\$cells : Each argument is a row cell

public static function `pack()`
Static method for pack content
\$cells : Each argument is a cell

THidden extends TField*Hidden field***Methods**

public function `__construct($name)`
Class Constructor
\$name : name of the field

public function `getPostData()`
Return the post data

public function `show()`
Show the widget at the screen

THtmlEditor extends TField*Html Editor***Methods**

public function `__construct($name)`
Class Constructor
\$name : Widget's name

public function `setMaxLength($length)`
Define max length
\$length : Max length

public function `setOption($option, $value)`
Set extra calendar options

public function `addCustomButton($name, $function, $title, TImage $icon, $showLabel = false)`
Add custom button
\$name : String name action
\$function : String function(context){ }
\$title : String title icon
\$icon : TImage toolbar icon

public function `setSize($width, $height = NULL)`
Define the widget's size
\$width : Widget's width
\$height : Widget's height

public function `getSize()`
Returns the size

public function `disableToolbar()`
Disable toolbar

function `setCompletion($options)`
Define options for completion
\$options : array of options for completion

public static function `enableField($form_name, $field)`
Enable the field
\$form_name : Form name
\$field : Field name

public static function `disableField($form_name, $field)`
Disable the field
\$form_name : Form name
\$field : Field name

public static function `clearField($form_name, $field)`
Clear the field
\$form_name : Form name
\$field : Field name

public static function `reloadCompletion($field, $options)`
Reload completion
\$field : Field name or id
\$options : array of options for autocomplete

public static function `insertText($form_name, $field, $content)`
Insert text
\$form_name : Form name
\$field : Field name
\$content : Text content

public static function `insertHTML($form_name, $field, $content)`
Insert html
\$form_name : Form name
\$field : Field name
\$content : Text content

public function `show()`
Show the widget

THtmlRenderer

Html Renderer

Methods

public function `__construct($path)`
Constructor method
\$path : HTML resource path

public static function `create($path, $replaces)`
Creates HTML Renderer

public function `disableHtmlConversion()`
Disable htmlspecialchars on output

public function `enableTranslation()`

Enable translation inside template

```
public function enableSection($sectionName, $replacements = NULL, $repeat = FALSE)
```

Enable a HTML section to show

\$sectionName : Section name

\$replacements : Array of replacements for this section

\$repeat : Define if the section is repeatable

```
public function disableSection($sectionName)
```

Disable section

```
public function show()
```

Show the HTML and the enabled sections

```
public static function recursiveKeyArraySearch($needle,$haystack)
```

Static search in memory structure

```
public function getContents()
```

Returns the HTML content as a string

THyperLink extends TTextDisplay

File Link

Methods

```
public function __construct($value, $location, $color = null, $size = null, $decoration = null, $icon = null)
```

Class Constructor

\$value : text content

\$location : link location

\$color : text color

\$size : text size

\$decoration : text decorations (b=bold, i=italic, u=underline)

TIcon extends TCombo

Color Widget

Methods

```
public function __construct($name)
```

Class Constructor

\$name : Name of the widget

TIconView extends TElement

TIconView Widget

Methods

```
public function __construct()
```

Constructor Method

```
public function enableDoubleClick()
```

Enable double click

```
public function setItemTemplate($template)
```

Set item template for rendering

\$template : Template content

```
public function setTemplateAttribute($attribute)
```

Set template attribute for rendering

\$attribute : Template attribute

public function enablePopover(\$title, \$content, \$popside = null, \$popcondition = null)

Enable popover

\$title : Title

\$content : Content

public function enableMoveAction(\$moveAction, \$source_selector, \$target_selector)

Enable move action

\$source_selector : Source item selector

\$target_selector : Target item selector

public function setInfoAttributes(\$fields)

Set info fields

public function addItem(\$object)

Add object item with data

public function setIconAttribute(\$iconField)

Define the field that will identify the icon

public function setLabelAttribute(\$labelField)

Define the field that will identify the label

public function addContextMenuOption(\$label, \$action = null, \$icon = null, /*Callable*/ \$displayCondition = null)

Add a context menu option

public function show()

Show iconview items

TImage extends TElement

Image Widget

Methods

public function __construct(\$source = "")

Class Constructor

\$source : Image path, of bs:bs-glyphicon, fab:font-awesome

TImageCapture extends TImageCropper

Image capture

Methods

public function __construct(\$name)

TImageCropper extends TField

Image uploader with cropper

Methods

public function __construct(\$name)

public function setImagePlaceholder(TImage \$image)

Set image placeholder

TImage : \$image image placeholder

public function setWindowTitle(\$title)

Set window title

String : \$title Window title

public function **setButtonLabel**(\$text)

Set text button crop

String : \$text Text button confirm

public function **setAspectRatio**(\$aspectRatio)

Define initial aspect ratio

double : \$aspectRatio Aspect ratio crop image

public function **enableBase64**()

Define usage base64

public function **enableWebCam**()

public function **enableFileHandling**()

Define to file handling

public function **disableButtonsDrag**()

Disable buttons drag move | resize

public function **disableButtonsZoom**()

Disable buttons zoom

public function **disableButtonsScale**()

Disable buttons scale

public function **disableButtonReset**()

Disable button reset

public function **disableButtonsRotate**()

Disable buttons rotates

public function **setValue**(\$value)

Define image initial

String : \$data Image url or image base64

public function **setAllowedExtensions**(\$extensions)

Define the allowed extensions

public function **getAllowedExtensions**()

Get the allowed extensions

public function **setService**(\$service)

Define the service class for response

public function **setSize**(\$width, \$height = NULL)

Define the Field's width

double : \$width Field's width in pixels

double : \$height Field's height in pixels

public function **getSize**()

Returns the field sizes

public function **setCropSize**(\$width, \$height)

Set image size after crop

px : \$width

px : \$height

public function **getOptions**()

Return component specific options

```
public function show()  
    Show
```

TInputDialog

Input Dialog

Methods

```
public function __construct($title_msg, AdiantiFormInterface $form, ?TAction $action = NULL, $caption = "")  
    Class Constructor  
    $title_msg : Dialog Title  
    $form : Dialog form body  
    $action : Action to be processed when closing the dialog  
    $caption : Button caption
```

TJQueryDialog extends TElement

JQuery dialog container

Methods

```
public function __construct()  
    Class Constructor  
    $name : Name of the widget  
  
public function disableEscape()  
    Disable close on escape  
  
public function disableScrolling()  
    Disable scrolling  
  
public function enableScrolling()  
    Enable scrolling  
  
public function setDialogClass($class)  
    Set Dialog class  
    $class : Class name  
  
public function setCloseAction(TAction $action)  
    Set close action  
  
public function setUseOKButton($bool)  
    Define if will use OK Button  
    $bool : boolean  
  
public function setTitle($title)  
    Define the dialog title  
    $title : title  
  
public function setModal($bool)  
    Turn on/off modal  
    $modal : Boolean  
  
public function setResizable($bool)  
    Turn on/off resizable  
    $bool : Boolean  
  
public function setDraggable($bool)  
    Turn on/off draggable  
    $bool : Boolean
```

```

public function getId()
    Returns the element ID

public function setSize($width, $height)
    Define the dialog size
    $width : width
    $height : height

public function setMinWidth($percent, $absolute)
    Define the window's min width between percent and absolute
    $percent : width
    $absolute : width

public function setPosition($left, $top)
    Define the dialog position
    $left : left
    $top : top

public function addAction($label, $action)
    Add a JS button to the dialog
    $label : button label
    $action : JS action

public function setStackOrder($order)
    Define the stack order (zIndex)
    $order : Stack order

public function show()
    Shows the widget at the screen

public function close()
    Closes the dialog

public static function closeById($id)
    Close window by id

public static function closeAll()
    Close all TjQueryDialog

public static function closeLatest()
    Close all TjQueryDialog

```

TKanban extends TElement

Kanban

Methods

```

public function __construct()
    Class Constructor

public function setStageHeight($height)

public function enableTopScrollbar()
    Enable Top Scrollbar

public function addStage($id, $title, $object = null, $color = null)
    Add stage to kanban board
    $id : Stage id
    $title : Stage title
    $color : Stage color
    $object : Stage data object

```

```

public function addItem($id, $stage_id, $title, $content, $color = null, $object = null)
    Add item to stage
    $id : Item id
    $stage_id : Stage id
    $title : Item title
    $content : Item content
    $color : Item color
    $object : Item data object

public function addObject(TRecord $object)

public function clear()
    Clear items

public function setTemplatePath($path)
    Set kanban item template for rendering
    $path : Template path

public function setItemTemplate($template)
    Set card item template for rendering
    $template : Template content

public function setItemTemplateCallback(Callable $callback)
    Set card item callback for rendering
    $callback : Callback

public function setItemDatabase($database)
    Set item min database
    $database : min database

public function setItemDropAction(TAction $action)
    Set item drop action
    $action : TAction object

public function setStageDropAction(TAction $action)
    Set stage drop action
    $action : TAction object

public function addItemAction($label, TAction $action, $icon = NULL, $display_condition = NULL, $useButton = FALSE)
    Add item action
    $label : Action label
    $action : Action callback (TAction)
    $icon : Action icon
    $display_condition : Display condition
    $useButton : Action display button

public function addStageAction($label, TAction $action, $icon = NULL, $display_condition = NULL)
    Add stage action
    $label : Action label
    $action : Action callback (TAction)
    $icon : Action icon

public function addStageShortcut($label, TAction $action, $icon = NULL)
    Add stage shortcut
    $label : Action label
    $action : Action callback (TAction)
    $icon : Action icon

public function setSearchForm($searchForm)
    Assign a TForm object
    $searchForm : object

public function setEditForm($editForm)

```

Assign a TForm object

\$editForm : object

public function **getSearchForm()**
Return the assigned Search form object

public function **getEditForm()**
Return the assigned Edit form object

public function **setMetadata(\$metadata, \$value)**
Set metadata

public function **getMetadata(\$metadata)**
Get metadata

public function **show()**
Show kanban

TLabel extends TField

Label Widget

Methods

public function **__construct(\$value, \$color = null, \$fontsize = null, \$decoration = null, \$size = null)**
Class Constructor
\$value : text label

public function **enableToggleVisibility(\$toggleVisibility = TRUE)**
Enable toggle visible

public function **setFontSize(\$size)**
Define the font size
\$size : Font size in pixels

public function **setFontStyle(\$decoration)**
Define the style
\$decoration : text decorations (b=bold, i=italic, u=underline)

public function **setFontFace(\$font)**
Define the font face
\$font : Font Family Name

public function **setFontColor(\$color)**
Define the font color
\$color : Font Color

public function **setTextAlign(\$align)**
Define the text align
\$align : Text Align

function **add(\$content)**
Add a content inside the label
\$content :

public function **getValue()**
Get value

public function **show()**
Shows the widget at the screen

TLikertScale extends TField

Likert Scale

Methods

public function **__construct**(\$name)

Class Constructor

\$name : name of the field

public function **enableFullWidth**()

public function **addItem**(\$items)

Add items to the radio group

\$items : An indexed array containing the options

public function **getItems**()

Return the items

public function **getButtons**()

Return the option buttons

public function **getLabels**()

Return the option labels

public function **setChangeAction**(TAction \$action)

Define the action to be executed when the user changes the combo

\$action : TAction object

public function **setChangeFunction**(\$function)

Set change function

public static function **reload**(\$formname, \$name, \$items, \$options = [])

Reload radio items after it is already shown

\$formname : form name (used in gtk version)

\$name : field name

\$items : array with items

\$options : array of options [value, changeAction, changeFunction]

public static function **enableField**(\$form_name, \$field)

Enable the field

\$form_name : Form name

\$field : Field name

public static function **disableField**(\$form_name, \$field)

Disable the field

\$form_name : Form name

\$field : Field name

public static function **clearField**(\$form_name, \$field)

clear the field

\$form_name : Form name

\$field : Field name

public function **show**()

Show the widget at the screen

TLogger

Provides an abstract interface to register LOG files

Methods

public function **__construct**(\$filename = NULL)

Class Constructor

\$filename : path for LOG file

abstract function **write**(\$message);

Write abstract method Must be declared in child classes

TLoggerHTML extends TLogger

Register LOG in HTML files

Methods

public function **write**(\$message)

Writes an message in the LOG file

\$message : Message to be written

TLoggerSTD extends TLogger

Register LOG in Standard Output

Methods

public function **write**(\$message)

Writes an message in the LOG file

\$message : Message to be written

TLoggerTXT extends TLogger

Register LOG in TXT files

Methods

public function **write**(\$message)

Writes an message in the LOG file

\$message : Message to be written

TLoggerXML extends TLogger

Register LOG in HTML files

Methods

public function **write**(\$message)

Writes an message in the LOG file

\$message : Message to be written

TMaxLengthValidator extends TFieldValidator

Maximum length validation

Methods

public function **validate**(\$label, \$value, \$parameters = NULL)

Validate a given value

\$label : Identifies the value to be validated in case of exception

\$value : Value to be validated

\$parameters : additional parameters for validation (length)

TMaxValueValidator extends TFieldValidator

Maximum value validation

Methods

```
public function validate($label, $value, $parameters = NULL)
    Validate a given value

    $label : Identifies the value to be validated in case of exception
    $value : Value to be validated
    $parameters : additional parameters for validation (max value)
```

TMenu extends TElement

Menu Widget

Methods

```
public function __construct($xml, $permission_callback = NULL, $menu_level = 1, $menu_class = 'dropdown-menu',
    $item_class = "", $link_class = 'dropdown-toggle', $item_transformer = null, $menu_transformer = null)
    Class Constructor
    $xml : SimpleXMLElement parsed from XML Menu

public function addMenuItem(TMenuItem $menuItem)
    Add a MenuItem
    $menuItem : A TMenuItem Object

public function getMenuItems()
    Return the menu items

public function parse($xml, $permission_callback = NULL)
    Parse a XMLElement reading menu entries
    $xml : A SimpleXMLElement Object
    $permission_callback : check permission callback

public function show()
    Shows the widget at the screen
```

TMenuBar extends TElement

Menubar Widget

Methods

```
public function __construct()

public static function newFromXML($xml_file, $permission_callback = NULL, $bar_class = 'nav navbar-nav',
    $menu_class = 'dropdown-menu', $item_class = "")
    Build a MenuBar from a XML file
    $xml_file : path for the file
    $permission_callback : check permission callback

public function show()
    Show
```

TMenuItem extends TElement

MenuItem Widget

Methods

```
public function __construct($label, $action, $image = NULL, $level = 0, $menu_transformer = null)
    Class constructor
```

\$label : The menu label
\$action : The menu action
\$image : The menu image

public function **getAction**()
Returns the action

public function **setAction**(\$action)
Set the action

public function **getLabel**()
Returns the label

public function **setLabel**(\$label)
Set the label

public function **getImage**()
Returns the image

public function **setImage**(\$image)
Set the image

public function **getMenu**()
Returns the menu

public function **getLevel**()
Returns the level

public function **getLink**()
Returns the link

public function **setLinkClass**(\$class)
Set link class

public function **setMenu**(TMenu \$menu)
Define the submenu for the item
\$menu : A TMenu object

public function **setItemClass**(\$class)
Set link class Item

public function **setClassIcon**(\$class)
Set icon class Item

public function **setTagLabel**(\$tag)
Set tag label

public function **show**()
Shows the widget at the screen

TMenuParser

Menu Parser

Methods

public function **__construct**(\$xml_file)
Parse a menu XML file
\$xml_file : file path

public function **getIndexedPrograms**()

Return an indexed array of programs

```
public function getPath($controller)
    Return the controller path
```

```
public function moduleExists($module)
    Check if a module exists
```

```
public function getModules()
    Get Modules
```

TMessage

Message Dialog

Methods

```
public function __construct($type, $message, ?TAction $action = NULL, $title_msg = "")
    Class Constructor

    $type : Type of the message (info, warning, error)
    $message : Message to be shown
    $action : Action to be processed when closing the dialog
    $title_msg : Dialog Title
```

TMinLengthValidator extends TFieldValidator

Minimum length validation

Methods

```
public function validate($label, $value, $parameters = NULL)
    Validate a given value

    $label : Identifies the value to be validated in case of exception
    $value : Value to be validated
    $parameters : additional parameters for validation (length)
```

TMinValueValidator extends TFieldValidator

Minimum value validation

Methods

```
public function validate($label, $value, $parameters = NULL)
    Validate a given value

    $label : Identifies the value to be validated in case of exception
    $value : Value to be validated
    $parameters : additional parameters for validation (min value)
```

TModalForm

Modal form constructor for Adianti Framework

Methods

```
public function __construct($name = 'my_form')
    Constructor method

    $name : form name

public function getId()
    Returns form id
```

```

public function setFormTitle($title)
    Add a form title
    $title : Form title

public function getFormTitle()
    Return the form title

public function setName($name)
    Set form name
    $name : Form name

public function getName()
    Get form name

public function addField(AdiantiWidgetInterface $field)
    Add form field
    $field : Form field

public function delField(AdiantiWidgetInterface $field)
    Del form field
    $field : Form field

public function setFields($fields)
    Set form fields
    $fields : Array of Form fields

public function getField($name)
    Return form field
    $name : Field name

public function getFields()
    Return form fields

public function clear( $keepDefaults = FALSE )
    Clear form

public function setData($object)
    Set form data
    $object : Data object

public function getData($class = 'StdClass')
    Get form data
    $class : Object type of return data

public function getActions()
    Return form actions

public function validate()
    Validate form data

public function addRowField($label_string, AdiantiWidgetInterface $field, $floating = false)
    Add row
    $label_string : Field label
    $field : Field object
    $floating : turn on/off floating labels

public function addRowContent($content)
    Add a row content

public function addAction($label, TAction $action, $icon = 'fa:save', $name = null)
    Add a form action

```

\$label : Button label
\$action : Button action
\$icon : Button icon

public function **addFooterAction**(\$label, TAction \$action, \$icon = 'fa:save', \$name = null)
Add a form action

\$label : Button label
\$action : Button action
\$icon : Button icon

public function **addFooterActionLink**(\$label, TAction \$action, \$icon = 'fa:save')
Add a form action

\$label : Button label
\$action : Button action
\$icon : Button icon

public function **setModalClass**(\$class)
Set modal class

public function **render**()
Render DOM

public function **show**()
Show form

TMultiCombo extends TSelect

MultiCombo Widget

Methods

public function **__construct**(\$name)
Class Constructor
\$name : widget's name

public static function **reload**(\$formname, \$name, \$items, \$startEmpty = FALSE)
Reload combobox items after it is already shown
\$formname : form name (used in gtk version)
\$name : field name
\$items : array with items
\$startEmpty : ...

public function **show**()
Shows the widget

TMultiEntry extends TSelect

Multi Entry Widget

Methods

public function **__construct**(\$name)
Class Constructor
\$name : Widget's name

public function **setSize**(\$width, \$height = NULL)
Define the widget's size
\$width : Widget's width
\$height : Widget's height

public function **setMaxSize**(\$maxsize)
Define the maximum number of items that can be selected

public static function **enableField**(\$form_name, \$field)

Enable the field

\$form_name : Form name

\$field : Field name

public static function **disableField**(\$form_name, \$field)

Disable the field

\$form_name : Form name

\$field : Field name

public static function **clearField**(\$form_name, \$field)

Clear the field

\$form_name : Form name

\$field : Field name

protected function **renderItems**(\$with_titles = true)

Render items

public function **setValueSeparator**(\$sep)

Define the field's separator

\$sep : A string containing the field's separator

public function **setValue**(\$value)

Define the field's value

\$value : A string containing the field's value

public function **getPostData**()

Return the post data

public function **show**()

Shows the widget

TMultiFile extends TField

FileChooser widget

Methods

public function **__construct**(\$name)

Constructor method

\$name : input name

public function **enableImageGallery**(\$width = null, \$height = 100)

Enable image gallery view

public function **enablePopover**(\$title = null, \$content = "")

Enable popover

\$title : Title

\$content : Content

public function **setService**(\$service)

Define the service class for response

public function **setAllowedExtensions**(\$extensions)

Define the allowed extensions

public function **enableFileHandling**()

Define to file handling

public function **disableFileHandling**()

Disable file handling

public function setSize(\$width, \$height = NULL)
Set field size

public function setHeight(\$height)
Set field height

public function getPostData()
Return the post data

public function setLimitUploadSize(\$limit)
Define upload size limit
\$limit : Size limit MBs

public function enablePHPFileUploadLimit()
Define upload size limit

public function setValue(\$value)
Set field value

public function show()
Show the widget at the screen

function setCompleteAction(TAction \$action)
Define the action to be executed when the user leaves the form field
\$action : TAction object

public static function enableField(\$form_name, \$field)
Enable the field
\$form_name : Form name
\$field : Field name

public static function disableField(\$form_name, \$field)
Disable the field
\$form_name : Form name
\$field : Field name

public static function clearField(\$form_name, \$field)
Clear the field
\$form_name : Form name
\$field : Field name

TMultiSearch extends TSelect

Multi Search Widget

Methods

public function __construct(\$name)
Class Constructor
\$name : Widget's name

public function disableMultiple()
Disable multiple selection

public function disableClear()
Disable clear

public function disableSearch()
Disable search

public function setSize(\$width, \$height = NULL)
Define the widget's size

\$width : Widget's width
\$height : Widget's height

public function **getSize()**
Returns the size

public function **setMinLength(\$length)**
Define the minimum length for search

public function **setMaxSize(\$maxsize)**
Define the maximum number of items that can be selected

public function **setValueSeparator(\$sep)**
Define the field's separator
\$sep : A string containing the field's separator

public function **setValue(\$value)**
Define the field's value
\$value : A string containing the field's value

public function **getPostData()**
Return the post data

public static function **enableField(\$form_name, \$field)**
Enable the field
\$form_name : Form name
\$field : Field name

public static function **disableField(\$form_name, \$field)**
Disable the field
\$form_name : Form name
\$field : Field name

public static function **clearField(\$form_name, \$field)**
Clear the field
\$form_name : Form name
\$field : Field name

public function **show()**
Shows the widget

TNotebook extends TElement

Notebook

Methods

public function **__construct(\$width = null, \$height = null)**
Class Constructor
\$width : Notebook's width
\$height : Notebook's height

public function **setTabsVisibility(\$visible)**
Define if the tabs will be visible or not
\$visible : If the tabs will be visible

public function **setTabsSensibility(\$sensibility)**
Define the tabs click sensibility
\$sensibility : If the tabs will be sensible to click

public function **getId()**
Returns the element ID

public function setSize(\$width, \$height)

Set the notebook size

\$width : Notebook's width
\$height : Notebook's height

public function getSize()

Returns the frame size

public function setCurrentPage(\$i)

Define the current page to be shown

\$i : An integer representing the page number (start at 0)

public function getCurrentPage()

Returns the current page

public function appendPage(\$title, \$object)

Add a tab to the notebook

\$title : tab's title
\$object : tab's content

public function getPageCount()

Return the Page count

public function setTabAction(TAction \$action)

Define the action for the Notebook tab

\$action : Action taken when the user clicks over Notebook tab (A TAction object)

public function render()

Render the notebook

public function show()

Show the notebook

TNumeric extends TEntry

Numeric Widget

Methods

public function __construct(\$name, \$decimals, \$decimalsSeparator, \$thousandSeparator, \$replaceOnPost = true, \$reverse = FALSE, \$allowNegative = TRUE)

public function setAllowNegative(\$allowNegative)

Define input allow negative

TNumericValidator extends TFieldValidator

Numeric validation

Methods

public function validate(\$label, \$value, \$parameters = NULL)

Validate a given value

\$label : Identifies the value to be validated in case of exception
\$value : Value to be validated
\$parameters : additional parameters for validation (min value)

TPage extends TElement

Page Controller Pattern: used as container for all elements inside a page and also as a page controller

Methods

public function `__construct()`
Class Constructor

public static function `create()`
Static creation

public static function `setPageTitle($title)`
Change page title

public function `setTargetContainer($container)`
Set target container for page content

public function `getTargetContainer()`
Return target container

public static function `include_js($js)`
Include a specific JavaScript function to this page
\$js : JavaScript location

public static function `include_css($css)`
Include a specific Cascading Stylesheet to this page
\$css : Cascading Stylesheet

public static function `register_css($cssname, $csscode)`
Register a specific Cascading Stylesheet to this page
\$cssname : Cascading Stylesheet Name
\$csscode : Cascading Stylesheet Code

public static function `openFile($file, $basename = null)`
Open a File Dialog
\$file : File Name

public static function `openPage($page)`
Open a page in new tab

public static function `isMobile()`
Discover if the browser is mobile device

public function `setCurtainWidth($width)`
Set last curtain width

public function `show()`
Decide wich action to take and show the page

public function `run()`
Interprets an action based at the URL parameters

public function `setPageName($name)`
Set page name

public function `getClassName()`
Return the Page name

public function `setIsWrapped($bool)`
Define if the element is wrapped inside another one
@bool : Boolean TRUE if is wrapped

TPageNavigation

Page Navigation provides navigation for a datagrid

Methods

public function `__construct()`
Constructor method

public function `hide()`
Hide

public function `enableCounters()`
Enable counters

public function `setLimit($limit)`
Set the Amount of displayed records
\$limit : An integer

public function `getLimit()`
Returns the limit of records

public function `setWidth($width)`
Define the PageNavigation's width
\$width : PageNavigation's width

public function `setCount($count)`
Define the total count of records
\$count : An integer (the total count of records)

public function `getCount()`
Return the total count of records

public function `setPage($page)`
Define the current page
\$page : An integer (the current page)

public function `getPage()`
Returns the current page

public function `setFirstPage($first_page)`
Define the first page
\$page : An integer (the first page)

public function `setOrder($order)`
Define the ordering
\$order : A string containint the column name

public function `setDirection($direction)`
Define the ordering
\$direction : asc, desc

public function `setProperties($properties)`
Set the page navigation properties
\$properties : array of properties

public function `setAction($action)`
Define the PageNavigation action
\$action : TAction object (fired when the user navigates)

public function `show()`
Show the PageNavigation widget

TPageStep extends TElement

Page Step

Methods

public function **__construct**()
Constructor

public function **addItem**(\$title, \$action = null)
Add an item
\$title : Item title
\$completed : Item is completed
\$action : Item action

public function **select**(\$title)
Select current item

TPanel extends TElement

Panel Container: Allows to organize the widgets using fixed (absolute) positions

Methods

public function **__construct**(\$width, \$height)
Class Constructor
\$width : Panel's width
\$height : Panel's height

public function **setSize**(\$width, \$height)
Set the panel's size
\$width : Panel width
\$height : Panel height

public function **getSize**()
Returns the frame size

public function **put**(\$widget, \$col, \$row)
Put a widget inside the panel
\$widget : = widget to be shown
\$col : = column in pixels.
\$row : = row in pixels.

public function **show**()
Show the widget

TPanelGroup extends TElement

Bootstrap native panel for Adianti Framework

Methods

public function **__construct**(\$title = NULL, \$background = NULL)
Constructor method
\$title : Panel Title
\$footer : Panel Footer

public static function **pack**(\$title, \$element, \$footer = null)
Static creator for panels
\$title : Panel title
\$element : Panel content

public function **collapse**()
Collapse panel group by default

```
public function setTitle($title)
    Set title
```

```
public function addHeaderActionLink($label, TAction $action, $icon = 'fa:save')
    Add a form header action
    $label : Button label
    $action : Button action
    $icon : Button icon
```

```
public function addHeaderWidget($widget)
    Add a form header widget
    $widget : Widget
```

```
public function add($content)
    Add the panel content
```

```
public function getHeader()
    Return panel header
```

```
public function getBody()
    Return panel body
```

```
public function getFooter()
    Return panel footer
```

```
public function addFooter($footer)
    Add footer
```

TPassword extends TField

Password Widget

Methods

```
public function __construct($name)
    Class Constructor
    $name : Name of the widget
```

```
public function enableToggleVisibility($toggleVisibility = TRUE)
```

```
public function disableToggleVisibility()
```

```
function setExitAction(TAction $action)
    Define the action to be executed when the user leaves the form field
    $action : TAction object
```

```
public function setMaxLength($length)
    Define max length
    $length : Max length
```

```
public function setInnerIcon(TImage $image, $side = 'right')
    Define the Inner icon
```

```
public function setExitFunction($function)
    Define the javascript function to be executed when the user leaves the form field
    $function : Javascript function
```

```
public function disableAutoComplete()
    Disable auto complete
```

```
public function show()
    Show the widget at the screen
```

TPillBar extends TElement

Pillbar

Methods

public function `__construct()`
Constructor

public function `addItem($title, $action = null)`
Add an item
\$title : Item title
\$completed : Item is completed
\$action : Item action

public function `selectIndex($index)`

public function `select($title)`
Select current item

TProgressBar extends TElement

TProgressBar

Methods

public function `__construct()`

public function `setMask($mask)`
set mask for progress bar value Ex: "{value}%"

public function `setClass($class)`
set style class

public function `setValue($value)`
Set the value of progress bar

public function `show()`
Shows the widget at the screen

TQRCodeDisplay extends TElement

QR CodeDisplay

Methods

public function `__construct($value)`
Class Constructor
\$value : barcode content

public function `setValue($value)`
Replace current value

public function `setHeight($height)`
Define the bar code height
\$width : Barcode height in pixels

public function `getValue()`
Returns the field value

public function `show()`

TQRCodeInputReader extends TEntry

QR Code Input Reader

Methods

public function **__construct**(\$name)

Class Constructor

\$name : Name of the widget

public function **setChangeFunction**(\$function)

Set change function

public function **setChangeAction**(TAction \$action)

Define the action to be executed when the user changes the content

\$action : TAction object

public function **show**()

Shows the widget at the screen

TQuestion

Question Dialog

Methods

public function **__construct**(\$message, ?TAction \$action_yes = NULL, ?TAction \$action_no = NULL, \$title_msg = "", \$label_yes = "", \$label_no = "")

Class Constructor

\$message : A string containint the question

\$action_yes : Action taken for YES response

\$action_no : Action taken for NO response

\$title_msg : Dialog Title

TQuickForm extends TForm

Create quick forms for input data with a standard container for elements

Methods

public function **__construct**(\$name = 'my_form')

Class Constructor

\$name : Form Name

public function **setClientValidation**(\$bool)

Turn on/off client validation

public function **getActionsContainer**()

Returns the actions container

public function **getTable**()

Returns the inner table

public function **setFieldsByRow**(\$count)

Define the field quantity per row

\$count : Field count

public function **getFieldsByRow**()

Return the fields by row count

public function **getContainer**()

Returns the form container

```
public function setFormTitle($title)
```

Add a form title

\$title : Form title

```
public function getInputRows()
```

Returns the input groups

```
public function addQuickField($label, AdiantiWidgetInterface $object, $size = 200, ?TFieldValidator $validator = NULL, $label_size = NULL)
```

Add a form field

\$label : Field Label

\$object : Field Object

\$size : Field Size

\$validator : Field Validator

```
public function addQuickFields($label, $objects, $required = FALSE)
```

Add a form field

\$label : Field Label

\$objects : Array of Objects

\$required : Boolean TRUE if required

```
public function addQuickAction($label, TAction $action, $icon = 'fa:save')
```

Add a form action

\$label : Action Label

\$action : TAction Object

\$icon : Action Icon

```
public function addQuickButton($label, $action, $icon = 'fa:save')
```

Add a form button

\$label : Action Label

\$action : Javascript action

\$icon : Action Icon

```
public function delActions()
```

Clear actions row

```
public function getActionButtons()
```

Return an array with action buttons

```
public function detachActionButtons()
```

Detach action buttons

```
public function addRow()
```

Add a row

```
public static function showField($form, $field, $speed = 0)
```

```
public static function hideField($form, $field, $speed = 0)
```

TQuickGrid extends TDataGrid

Create quick datagrids through its simple interface

Methods

```
public function addQuickColumn($label, $name, $align = 'left', $size = 200, ?TAction $action = NULL, $param = NULL)
```

Add a column

\$label : Field Label

\$object : Field Object

\$size : Field Size

```
public function addQuickAction($label, TDataGridAction $action, $field, $icon = NULL)
```

Add action to the datagrid

\$label : Action Label

\$action : TAction Object

\$icon : Action Icon

TQuickNotebookForm extends TQuickForm

Create quick forms with a notebook wrapper

Methods

public function **__construct**(\$name = 'my_form')

Class Constructor

\$name : Form Name

public function **setNotebookWrapper**(\$notebook)

Set the notebook wrapper

\$notebook : Notebook wrapper

public function **setFormTitle**(\$title)

Add a form title

\$title : Form title

public function **appendPage**(\$title, \$container = NULL)

Append a notebook page

\$title : Page title

\$cotnainer : Page container

public function **addQuickAction**(\$label, TAction \$action, \$icon = 'fa:save')

Add a form action

\$label : Action Label

\$action : TAction Object

\$icon : Action Icon

public function **show**()

Show the component

TRadioButton extends TField

RadioButton Widget

Methods

public function **__construct**(\$name)

Class Constructor

\$name : Name of the widget

public function **show**()

Show the widget at the screen

TRadioGroup extends TField

A group of RadioButton's

Methods

public function **__construct**(\$name)

Class Constructor

\$name : name of the field

public function **setBooleanMode**()

Enable/disable boolean mode

public function setValue(\$value)

Define the field's value

\$value : A string containing the field's value

public function getValue()

Returns the field's value

public function getPostData()

Return the post data

public function setLayout(\$dir)

Define the direction of the options

\$direction : String (vertical, horizontal)

public function getLayout()

Get the direction (vertical or horizontal)

public function setBreakItems(\$breakItems)

Define after how much items, it will break

public function setUseButton()

Show as button

public function addItems(\$items)

Add items to the radio group

\$items : An indexed array containing the options

public function getItems()

Return the items

public function getButtons()

Return the option buttons

public function getLabels()

Return the option labels

public function setChangeAction(TAction \$action)

Define the action to be executed when the user changes the combo

\$action : TAction object

public function setChangeFunction(\$function)

Set change function

public static function reload(\$formname, \$name, \$items, \$options = [])

Reload radio items after it is already shown

\$formname : form name (used in gtk version)

\$name : field name

\$items : array with items

\$options : array of options [layout, breakItems, size, useButton, value, changeAction, changeFunction, checkAll]

public static function enableField(\$form_name, \$field)

Enable the field

\$form_name : Form name

\$field : Field name

public static function disableField(\$form_name, \$field)

Disable the field

\$form_name : Form name

\$field : Field name

public static function clearField(\$form_name, \$field)

clear the field

\$form_name : Form name
\$field : Field name

public function **show()**
Show the widget at the screen

TRecord

Base class for Active Records

Methods

public function **__construct**(\$id = NULL, \$callObjectLoad = TRUE)
Class Constructor Instantiates the Active Record
[\$id] : Optional Object ID, if passed, load this object

public function **getIterator** () : Traversable
Returns iterator

public static function **create**(\$data)
Create a new TRecord and returns the instance
\$data : indexed array

public function **getCacheControl**()
Returns the cache control

public function **getEntity**()
Returns the name of database entity

public function **getPrimaryKey**()
Returns the name of the primary key for that Active Record

public function **getPrimaryKeyValue**()
Returns the primary key value for that Active Record

public function **getCreatedAtColumn**()
Returns the name of the created at column

public function **getUpdatedAtColumn**()
Returns the name of the updated at column

public static function **getDeletedAtColumn**()
Returns the name of the deleted at column

public static function **getPrefilters**()
Returns the prefilters

public function **getCreatedByColumn**()
Returns the the name of the created at column

public function **getUpdatedByColumn**()
Returns the the name of the updated at column

public static function **getDeletedByColumn**()
Returns the the name of the deleted at column

public function **getByUserSessionIdentifier**()
Returns the information related to the logged user

public function **mergeObject**(TRecord \$object)
Fill the Active Record properties from another Active Record

\$object : An Active Record

public function **fromArray**(\$data, \$check_attribute_list = true)

Fill the Active Record properties from an indexed array

\$data : An indexed array containing the object properties

public function **toArray**(\$filter_attributes = null)

Return the Active Record properties as an indexed array

\$filter_attributes : Array of attributes to be returned.

public function **getVirtualData**()

Return virtual data (non-persistent properties)

public function **toJson**()

Return the Active Record properties as a json string

public function **render**(\$pattern, \$cast = null)

Render variables inside brackets

public function **evaluate**(\$pattern)

Evaluate variables inside brackets

public function **addAttribute**(\$attribute, \$writer = NULL, \$reader = NULL)

Register an persisted attribute

\$attribute : Attribute name

\$writer : String, session var or callable to write this attribute

\$reader : String, session var or callable to read this attribute

public function **getAttributes**()

Return the persisted attributes

public function **getAttributeList**()

Get attribute list

public function **store**()

Store the objects into the database

public function **runReaders**()

Iterate readers to fill attributes

public function **exists**(\$id)

Tests if an ID exists

\$id : The object ID

public function **reload**()

ReLoad an Active Record Object from the database

public function **load**(\$id)

Load an Active Record Object from the database

\$id : The object ID

public function **loadTrashed**(\$id)

Load trashed records

public function **delete**(\$id = NULL)

Delete an Active Record object from the database

[\$id] : The Object ID

public function **restore**()

Restore soft deleted object

public function **getFirstID**()

Returns the *FIRST* Object ID from database

public function `getLastID()`

Returns the *LAST* Object ID from database

public static function `getObjects($criteria = NULL, $callObjectLoad = TRUE, $withTrashed = FALSE)`

Method `getObjects`

\$criteria : Optional criteria

\$callObjectLoad : If `load()` method from Active Records must be called to load object parts

public static function `countObjects($criteria = NULL, $withTrashed = FALSE)`

Method `countObjects`

\$criteria : Optional criteria

\$withTrashed :

public function `checkDependencies($class, $foreign_key, $id = null, $label = null)`

Check active record dependencies

public function `loadComposite($composite_class, $foreign_key, $id = NULL, $order = NULL)`

Load composite objects (parts in composition relationship)

\$composite_class : Active Record Class for composite objects

\$foreign_key : Foreign key in composite objects

\$id : Primary key of parent object

public function `hasMany($composite_class, $foreign_key = NULL, $primary_key = NULL, $order = NULL)`

Load composite objects. Shortcut for `loadComposite`

\$composite_class : Active Record Class for composite objects

\$foreign_key : Foreign key in composite objects

\$primary_key : Primary key of parent object

public function `filterMany($composite_class, $foreign_key = NULL, $primary_key = NULL, $order = NULL)`

Create a criteria to load composite objects

\$composite_class : Active Record Class for composite objects

\$foreign_key : Foreign key in composite objects

\$primary_key : Primary key of parent object

public function `findAggregationFor($class)`

Returns the aggregation information for some class

\$class : Active Record Class name

public function `findCompositionFor($class)`

Returns the composition information for some class

\$class : Active Record Class name

public function `findAssociationFor($class)`

Returns the association information for some class

\$class : Active Record Class name

public function `deleteComposite($composite_class, $foreign_key, $id, $callObjectLoad = FALSE)`

Delete composite objects (parts in composition relationship)

\$composite_class : Active Record Class for composite objects

\$foreign_key : Foreign key in composite objects

\$id : Primary key of parent object

public function `saveComposite($composite_class, $foreign_key, $id, $objects, $callObjectLoad = FALSE)`

Save composite objects (parts in composition relationship)

\$composite_class : Active Record Class for composite objects

\$foreign_key : Foreign key in composite objects

\$id : Primary key of parent object

\$objects : Array of Active Records to be saved

public function `loadAggregate($aggregate_class, $join_class, $foreign_key_parent, $foreign_key_child, $id = NULL)`

Load aggregated objects (parts in aggregation relationship)

\$aggregate_class : Active Record Class for aggregated objects
\$join_class : Active Record Join Class (Parent / Aggregated)
\$foreign_key_parent : Foreign key in Join Class to parent object
\$foreign_key_child : Foreign key in Join Class to child object
\$id : Primary key of parent object

public function **loadAggregatedClass**(\$aggregate_class)

Load aggregated objects (parts in aggregation relationship)

\$aggregate_class : Active Record Class for aggregated objects
\$join_class : Active Record Join Class (Parent / Aggregated)
\$foreign_key_parent : Foreign key in Join Class to parent object
\$foreign_key_child : Foreign key in Join Class to child object
\$id : Primary key of parent object

public function **loadAggregatelds**(\$aggregate_class, \$join_class, \$foreign_key_parent, \$foreign_key_child, \$id = NULL)

Load aggregated objects (parts in aggregation relationship)

\$aggregate_class : Active Record Class for aggregated objects
\$join_class : Active Record Join Class (Parent / Aggregated)
\$foreign_key_parent : Foreign key in Join Class to parent object
\$foreign_key_child : Foreign key in Join Class to child object
\$id : Primary key of parent object

public function **belongsToMany**(\$aggregate_class, \$join_class = NULL, \$foreign_key_parent = NULL, \$foreign_key_child = NULL)

Load aggregated objects. Shortcut to loadAggregate

\$aggregate_class : Active Record Class for aggregated objects
\$join_class : Active Record Join Class (Parent / Aggregated)
\$foreign_key_parent : Foreign key in Join Class to parent object
\$foreign_key_child : Foreign key in Join Class to child object

public function **saveAggregate**(\$join_class, \$foreign_key_parent, \$foreign_key_child, \$id, \$objects)

Save aggregated objects (parts in aggregation relationship)

\$join_class : Active Record Join Class (Parent / Aggregated)
\$foreign_key_parent : Foreign key in Join Class to parent object
\$foreign_key_child : Foreign key in Join Class to child object
\$id : Primary key of parent object
\$objects : Array of Active Records to be saved

public static function **first**(\$withTrashed = FALSE)

Returns the first object

public static function **firstOrCreate**(\$filters = NULL)

First record or a new one

public static function **firstOrNew**(\$filters = NULL)

First record or persist a new one

public static function **last**(\$withTrashed = FALSE)

Returns the last object

public static function **find**(\$id, \$withTrashed = FALSE)

Find a Active Record and returns it

public static function **all**(\$indexed = false, \$withTrashed = FALSE)

Returns all objects

public function **save**()

Save the object

public static function **getIndexedArray**(\$indexColumn, \$valueColumn, \$criteria = NULL, \$withTrashed = FALSE)

Creates an indexed array

public static function **select**()

\$value := value to be compared

\$logicOperator := logical operator (*TExpression::AND_OPERATOR*, *TExpression::OR_OPERATOR*)

public function **addFilter**(*TFilter* \$filter)
Add a filter

public function **set**(\$column, \$value)
Assign values to the database columns

\$column := column name

\$value := column value

public function **orWhere**(\$variable, \$operator, \$value)
Add a run time OR criteria using fluent interfaces

\$variable := variable

\$operator := comparison operator (>,

\$value := value to be compared

public function **orderBy**(\$order, \$direction = 'asc')
Define the ordering for criteria using fluent interfaces

\$order := Order column

\$direction := Order direction (asc, desc)

public function **groupBy**(\$group)
Define the group for criteria using fluent interfaces

\$group := Group column

public function **take**(\$limit)
Define the LIMIT criteria using fluent interfaces

\$limit := Limit

public function **skip**(\$offset)
Define the OFFSET criteria using fluent interfaces

\$offset := Offset

public function **load**(?TCriteria \$criteria = NULL, \$callObjectLoad = TRUE)
Load a collection of objects from database using a criteria

\$criteria := An TCriteria object, specifying the filters

\$callObjectLoad := If load() method from Active Records must be called to load object parts

public function **loadStatic**()
Load with no aggregates

public function **getIndexedArray**(\$indexColumn, \$valueColumn = NULL, \$criteria = NULL)
Return a indexed array

public function **update**(\$setValues = NULL, ?TCriteria \$criteria = NULL)
Update values in the repository

public function **delete**(?TCriteria \$criteria = NULL, \$callObjectLoad = FALSE)
Delete a collection of Active Records from database

\$criteria := An TCriteria object, specifying the filters

public function **count**(?TCriteria \$criteria = NULL)
Return the amount of objects that satisfy a given criteria

\$criteria := An TCriteria object, specifying the filters

public function **countDistinctBy**(\$column, \$alias = null, ?Callable \$transformation = null)
Count distinct aggregate

\$column := Column to be aggregated

public function **countBy**(\$column, \$alias = null, ?Callable \$transformation = null)
Count aggregate

\$column : Column to be aggregated
\$alias : Column alias

public function **countByAnd**(\$column, \$alias = null, ?Callable \$transformation = null)
 Count aggregate and do another aggregate after
\$column : Column to be aggregated
\$alias : Column alias

public function **sumBy**(\$column, \$alias = null, ?Callable \$transformation = null)
 Sum aggregate
\$column : Column to be aggregated
\$alias : Column alias

public function **sumByAnd**(\$column, \$alias = null, ?Callable \$transformation = null)
 Sum aggregate and do another aggregate after
\$column : Column to be aggregated
\$alias : Column alias

public function **avgBy**(\$column, \$alias = null, ?Callable \$transformation = null)
 Average aggregate
\$column : Column to be aggregated
\$alias : Column alias

public function **avgByAnd**(\$column, \$alias = null, ?Callable \$transformation = null)
 Average aggregate and do another aggregate after
\$column : Column to be aggregated
\$alias : Column alias

public function **minBy**(\$column, \$alias = null, ?Callable \$transformation = null)
 Min aggregate
\$column : Column to be aggregated
\$alias : Column alias

public function **minByAnd**(\$column, \$alias = null, ?Callable \$transformation = null)
 Min aggregate and do another aggregate after
\$column : Column to be aggregated
\$alias : Column alias

public function **maxBy**(\$column, \$alias = null, ?Callable \$transformation = null)
 Max aggregate
\$column : Column to be aggregated
\$alias : Column alias

public function **maxByAnd**(\$column, \$alias = null, ?Callable \$transformation = null)
 Max aggregate and do another aggregate after
\$column : Column to be aggregated
\$alias : Column alias

protected function **aggregate**(\$function, \$column, \$alias = null, ?Callable \$transformation = null)
 Aggregate column
\$function : Aggregate function (count, sum, min, max, avg)

public function **get**(?TCriteria \$criteria = NULL, \$callObjectLoad = TRUE)
 Alias for load()

public function **first**(\$callObjectLoad = TRUE)
 Returns the first collection item

public function **last**(\$callObjectLoad = TRUE)
 Returns the last collection item

public function **transform**(Callable \$callback, \$callObjectLoad = TRUE)
 Returns transformed collection

public function **transformColumn**(\$alias, Callable \$callback)
Set a transformation for a column

public function **filter**(Callable \$callback, \$callObjectLoad = TRUE)
Returns filtered collection

public function **dump**(\$prepared = FALSE)
Dump Criteria

public static function **getCollectionAttribute**(\$collection, \$attribute)
Get an attribute list from a collection

TRequiredListValidator extends TFieldValidator

Required field validation

Methods

public function **validate**(\$label, \$values, \$parameters = NULL)
Validate a given value
\$label : Identifies the value to be validated in case of exception
\$value : Value to be validated
\$parameters : additional parameters for validation

TRequiredValidator extends TFieldValidator

Required field validation

Methods

public function **validate**(\$label, \$value, \$parameters = NULL)
Validate a given value
\$label : Identifies the value to be validated in case of exception
\$value : Value to be validated
\$parameters : additional parameters for validation

TScript

Base class for scripts

Methods

public static function **create**(\$code, \$show = TRUE, \$timeout = null)
Create a script
\$code : source code

public static function **importFromFile**(\$script, \$show = TRUE, \$timeout = null)
Import script
\$script : Script file name

TScroll extends TElement

Scrolled Window: Allows to add another containers inside, creating scrollbars when its content is bigger than its visual area

Methods

public function **__construct**()
Class Constructor

public function setSize(\$width, \$height)

Set the scroll size

\$width : Panel's width

\$height : Panel's height

public function setMargin(\$margin)

Set the scrolling margin

\$margin : Margin

public function setTransparency(\$bool)

compability reasons

public function show()

Shows the tag

TSeekBar extends TEntry

Record Lookup Widget: Creates a lookup field used to search values from associated entities

Methods

public function __construct(\$name, \$icon = NULL)

Class Constructor

\$name : name of the field

public static function createButton(\$name, \$icon)

Create seek button object

public function setSize(\$width, \$height = NULL)

Define the Field's width

\$width : Field's width in pixels

public function setUseOutEvent(\$bool)

Define if the out event will be fired

public function setAction(TAction \$action)

Define the action for the SeekButton

\$action : Action taken when the user clicks over the Seek Button (A TAction object)

public function getAction()

Return the action

public function setAuxiliar(\$object)

Define an auxiliar field

\$object : any TField object

public function hasAuxiliar()

Returns if has auxiliar field

public function setExtraSize(\$extra_size)

Set extra size

public function getExtraSize()

Returns extra size

public static function enableField(\$form_name, \$field)

Enable the field

\$form_name : Form name

\$field : Field name

public static function disableField(\$form_name, \$field)

Disable the field

\$form_name : Form name
\$field : Field name

public function show()
 Show the widget

TSelect extends TField

Select Widget

Methods

public function __construct(\$name)
 Class Constructor
\$name : widget's name

public function enableSearch()
 Enable search

public function disableMultiple()
 Disable multiple selection

public function disableTitles()
 Disable option titles

public function setDefaultOption(\$option)

public function addItem(\$items)
 Add items to the select
\$items : An indexed array containing the combo options

public function getItems()
 Return the items

public function setSize(\$width, \$height = NULL)
 Define the Field's width
\$width : Field's width in pixels
\$height : Field's height in pixels

public function getSize()
 Returns the size

public function setValueSeparator(\$sep)
 Define the field's separator
\$sep : A string containing the field's separator

public function setValue(\$value)
 Define the field's value
\$value : A string containing the field's value

public function getPostData()
 Return the post data

public function setChangeAction(TAction \$action)
 Define the action to be executed when the user changes the combo
\$action : TAction object

public function setChangeFunction(\$function)
 Set change function

public static function reload(\$formname, \$name, \$items, \$startEmpty = FALSE)
 Reload combobox items after it is already shown

\$formname : form name (used in gtk version)
\$name : field name
\$items : array with items
\$startEmpty : ...

public static function **enableField**(\$form_name, \$field)
Enable the field

\$form_name : Form name
\$field : Field name

public static function **disableField**(\$form_name, \$field)
Disable the field

\$form_name : Form name
\$field : Field name

public static function **clearField**(\$form_name, \$field)
Clear the field

\$form_name : Form name
\$field : Field name

protected function **renderItems**(\$with_titles = true)
Render items

public function **show**()
Shows the widget

TSession

Session Data Handler

Methods

public function **__construct**(?SessionHandlerInterface \$handler = NULL, \$path = NULL)
Class Constructor

public static function **enabled**()
Returns if the service is active

public static function **setValue**(\$var, \$value)
Define the value for a variable
\$var : Variable Name
\$value : Variable Value

public static function **getValue**(\$var)
Returns the value for a variable
\$var : Variable Name

public static function **delValue**(\$var)
Clear the value for a variable
\$var : Variable Name

public static function **regenerate**()
Regenerate id

public static function **clear**()
Clear session

public static function **freeSession**()
Destroy the session data Backward compatibility

TSlider extends TField

Slider Widget

Methods

public function `__construct`(\$name)

Class Constructor

\$name : Name of the widget

public function `setRange`(\$min, \$max, \$step)

Define the field's range

\$min : Minimal value

\$max : Maximal value

\$step : Step value

public static function `enableField`(\$form_name, \$field)

Enable the field

\$form_name : Form name

\$field : Field name

public static function `disableField`(\$form_name, \$field)

Disable the field

\$form_name : Form name

\$field : Field name

public function `show`()

Shows the widget at the screen

TSortList extends TField

A Sortable list

Methods

public function `__construct`(\$name)

Class Constructor

\$name : widget's name

public function `setOrientation`(\$orientation)

Define orientation

\$orientation : (horizontal, vertical)

public function `setLimit`(\$limit)

Define limit

public function `setItemIcon`(TImage \$icon)

Define the item icon

\$image : Item icon

public function `setSize`(\$width, \$height = NULL)

Define the list size

public function `setValueSeparator`(\$sep)

Define the field's separator

\$sep : A string containing the field's separator

public function `setValue`(\$value)

Define the field's value

\$value : An array the field's values

public function `connectTo`(TSortList \$list)

Connect to another list

\$list : Another TSortList

public function **addItems**(\$items)

Add items to the sort list

\$items : An indexed array containing the options

public function **getItems**()

Return the sort items

public function **getPostData**()

Return the post data

public function **setChangeAction**(TAction \$action)

Define the action to be executed when the user changes the combo

\$action : TAction object

public function **setChangeFunction**(\$function)

Set change function

public static function **enableField**(\$form_name, \$field)

Enable the field

public static function **disableField**(\$form_name, \$field)

Disable the field

public static function **clearField**(\$form_name, \$field)

Clear the field

public static function **reload**(\$form_name, \$field, \$items)

Reload elements

public function **show**()

Shows the widget at the screen

TSourceCode

SourceCode View

Methods

public function **loadFile**(\$file)

Load a PHP file

\$file : Path to the PHP file

public function **loadString**(\$content)

Load from string

public function **stripTags**()

public function **generateRowNumbers**()

Generate row numbers

public function **insertRowNumbers**(\$highlighted_string)

Insert row numbers

public function **show**()

Show the highlighted source code

public function **getContents**()

Returns the element content as a string

TSpinner extends TField

Spinner Widget (also known as spin button)

Methods

public function **__construct**(\$name)

Class Constructor

\$name : Name of the widget

public function **setRange**(\$min, \$max, \$step)

Define the field's range

\$min : Minimal value

\$max : Maximal value

\$step : Step value

public function **getPostData**()

Return the post data

function **setExitAction**(TAction \$action)

Define the action to be executed when the user leaves the form field

\$action : TAction object

public static function **enableField**(\$form_name, \$field)

Enable the field

\$form_name : Form name

\$field : Field name

public static function **disableField**(\$form_name, \$field)

Disable the field

\$form_name : Form name

\$field : Field name

public function **setExitFunction**(\$function)

Set exit function

public function **enableStepper**()

Enable stepper buttons

public function **disableStepper**()

Disable stepper buttons

public function **show**()

Shows the widget at the screen

public function **setValue**(\$value)

Set the value

TSqlDelete extends TSqlStatement

Provides an Interface to create DELETE statements

Methods

public function **getInstruction**(\$prepared = FALSE)

Returns a string containing the DELETE plain statement

\$prepared : Return a prepared Statement

TSqlInsert extends TSqlStatement

Provides an Interface to create an INSERT statement

Methods

public function `__construct()`
Constructor method

public function `setRowData($column, $value)`
Assign values to the database columns
\$column : Name of the database column
\$value : Value for the database column

public function `unsetRowData($column)`
Unset row data
\$column : Name of the database column

public function `setCriteria(TCriteria $criteria)`
this method doesn't exist in this class context
\$criteria : A TCriteria object, specifying the filters

public function `getPreparedVars()`
Return the prepared vars

public function `getInstruction($prepared = FALSE)`
Returns the INSERT plain statement
\$prepared : Return a prepared Statement

TSqlMultiInsert extends TSqlStatement

Provides an Interface to create an MULTI INSERT statement

Methods

public function `__construct()`
Constructor method

public function `addRowValues($row)`
Add a row data
\$row : Row data

public function `setCriteria(TCriteria $criteria)`
this method doesn't exist in this class context
\$criteria : A TCriteria object, specifying the filters

public function `getInstruction($prepared = FALSE)`
Returns the INSERT plain statement
\$prepared : Return a prepared Statement

TSqlSelect extends TSqlStatement

Provides an Interface to create SELECT statements

Methods

public function `addColumn($column)`
Add a column name to be returned
\$column : A string containing a column name

public function `getInstruction($prepared = FALSE)`
Returns the SELECT statement as a string according to the database driver
\$prepared : Return a prepared Statement

public function `getStandardInstruction($prepared)`

Returns the *SELECT* statement as an string for standard open source drivers

\$prepared : Return a prepared Statement

public function **getInterbaseInstruction**(\$prepared)

Returns the *SELECT* statement as an string for standard open source drivers

\$prepared : Return a prepared Statement

public function **getSqlServerInstruction**(\$prepared)

Returns the *SELECT* statement as an string for mssql/dblib drivers

\$prepared : Return a prepared Statement

public function **getOracleInstruction**(\$prepared)

Returns the *SELECT* statement as an string for oci8 drivers

\$prepared : Return a prepared Statement

TSqlStatement

Provides an abstract Interface to create a SQL statement

Methods

final public function **setEntity**(\$entity)

defines the database entity name

\$entity : Name of the database entity

final public function **getEntity**()

Returns the database entity name

public function **setCriteria**(TCriteria \$criteria)

Define a select criteria

\$criteria : An TCriteria object, specifying the filters

protected function **getRandomParameter**()

Returns a random parameter

abstract function **getInstruction**();

TSqlUpdate extends TSqlStatement

Provides an Interface to create UPDATE statements

Methods

public function **setRowData**(\$column, \$value)

Assign values to the database columns

\$column : Name of the database column

\$value : Value for the database column

public function **unsetRowData**(\$column)

Unset row data

\$column : Name of the database column

public function **getPreparedVars**()

Return the prepared vars

public function **getInstruction**(\$prepared = FALSE)

Returns the UPDATE plain statement

\$prepared : Return a prepared Statement

TStandardForm extends TPage

Standard page controller for forms

Methods

- public function** **setAfterSaveAction**(\$action)
method setAfterSaveAction() Define after save action
- public function** **setAfterSaveCallback**(\$callback)
method setAfterSaveCallback() Define after save callback
- public function** **setUseMessages**(\$bool)
Define if will use messages after operations
- public function** **onSave**()
method onSave() Executed whenever the user clicks at the save button
- public function** **onClear**(\$param)
Clear form
- public function** **onEdit**(\$param)
method onEdit() Executed whenever the user clicks at the edit button da datagrid
\$param : An array containing the GET (\$_GET) parameters
- protected function** **setDatabase**(\$database)
method setDatabase() Define the database
- protected function** **setActiveRecord**(\$activeRecord)
method setActiveRecord() Define wich Active Record class will be used

TStandardFormList extends TPage

Standard page controller for form/listings

Methods

- public function** **setAfterSaveAction**(\$action)
method setAfterSaveAction() Define after save action
- public function** **setUseMessages**(\$bool)
Define if will use messages after operations
- public function** **setLimit**(\$limit)
method setLimit() Define the record limit
- public function** **setDefaultOrder**(\$order, \$direction = 'asc')
Define the default order
\$order : The order field
\$direction : the order direction (asc, desc)
- public function** **setCriteria**(\$criteria)
method setCriteria() Define the criteria
- public function** **setTransformer**(\$callback)
Define a callback method to transform objects before load them into datagrid
- public function** **onReload**(\$param = NULL)
method onReload() Load the datagrid with the database objects
- public function** **onSave**()

method onSave() Executed whenever the user clicks at the save button

public function onDelete(\$param)

method onDelete() executed whenever the user clicks at the delete button Ask if the user really wants to delete the record

public function Delete(\$param)

method Delete() Delete a record

public function onClear(\$param)

Clear form

public function onEdit(\$param)

method onEdit() Executed whenever the user clicks at the edit button da datagrid

public function show()

Shows the page

protected function setDatabase(\$database)

method setDatabase() Define the database

protected function setActiveRecord(\$activeRecord)

method setActiveRecord() Define wich Active Record class will be used

TStandardList extends TPage

Standard page controller for listings

Methods

public function enableTotalRow()

Enable total row

public function onInlineEdit(\$param)

Inline record editing

\$param : Array containing: key: object ID value field name: object attribute to be updated value: new attribute content

public function onDelete(\$param)

Ask before deletion

public function Delete(\$param)

Delete a record

public function onDeleteCollection(\$param)

Ask before delete record collection

public function deleteCollection(\$param)

method deleteCollection() Delete many records

protected function setLimit(\$limit)

method setLimit() Define the record limit

protected function setCollectionObject(\$object)

Set list widget

protected function enableTransactionDebug()

Enable Debug

protected function setOrderCommand(\$order_column, \$order_command)

Set order command

protected function **setDefaultOrder**(\$order, \$direction = 'asc')

Define the default order

\$order : The order field

\$direction : the order direction (asc, desc)

protected function **setFilterField**(\$filterField)

method setFilterField() Define wich field will be used for filtering PS: Just for Backwards compatibility

protected function **setOperator**(\$operator)

method setOperator() Define the filtering operator PS: Just for Backwards compatibility

protected function **addFilterField**(\$filterField, \$operator = 'like', \$formFilter = NULL, \$filterTransformer = NULL, \$logic_operator = TExpression::AND_OPERATOR)

method addFilterField() Add a field that will be used for filtering

\$filterField : Field name

\$operator : Comparison operator

protected function **setCriteria**(\$criteria)

method setCriteria() Define the criteria

protected function **setTransformer**(\$callback)

Define a callback method to transform objects before load them into datagrid

protected function **setAfterLoadCallback**(\$callback)

Define a callback method to transform objects before load them into datagrid

protected function **setAfterSearchCallback**(\$callback)

Define a callback method to transform objects after search action

public function **onSearch**(\$param = null)

Register the filter in the session

protected function **clearFilters**()

clear Filters

public function **onReload**(\$param = NULL)

Load the datagrid with the database objects

public function **show**()

method show() Shows the page

protected function **setDatabase**(\$database)

method setDatabase() Define the database

protected function **setActiveRecord**(\$activeRecord)

method setActiveRecord() Define wich Active Record class will be used

public function **onExportCSV**(\$param)

Export to CSV

public function **onExportXLS**(\$param)

Export to XLS

public function **onExportXML**(\$param)

Export to XML

public function **onExportPDF**(\$param)

Export datagrid as PDF

protected function **exportToCSV**(\$output)

Export to CSV

\$output : Output file

protected function **exportToXLS**(\$output)

Export to CSV

\$output : Output file

protected function **exportToXML**(\$output)

Export to XML

\$output : Output file

protected function **exportToPDF**(\$output)

Export to PDF

\$output : Output file

TStandardSeek extends TWindow

Standard Page controller for Seek buttons

Methods

public function **__construct**()

Constructor Method Creates the page, the search form and the listing

public function **render**()

Render datagrid

public function **fill**()

Fill datagrid

public function **onSearch**()

Search datagrid

public function **onReload**(\$param = NULL)

Load the datagrid with objects

public function **onSetup**(\$param=NULL)

Setup seek parameters

public static function **onSelect**(\$param)

Send the selected register to parent form

public function **show**()

Show page

TStrippedTextMaxLengthValidator extends TFieldValidator

Maximum stripped text length validation

Methods

public function **validate**(\$label, \$value, \$parameters = NULL)

Validate a given value

\$label : Identifies the value to be validated in case of exception

\$value : Value to be validated

\$parameters : additional parameters for validation (min value)

TStrippedTextMinLengthValidator extends TFieldValidator

*Minimum stripped text length validation***Methods**

public function **validate**(\$label, \$value, \$parameters = NULL)
Validate a given value
\$label : Identifies the value to be validated in case of exception
\$value : Value to be validated
\$parameters : additional parameters for validation (min value)

TStyle*StyleSheet Manager***Methods**

public function **__construct**(\$name)
Class Constructor
\$name : Name of the style

public static function **importFromFile**(\$filename)
Import style
\$style : Style file name

public function **getName**()
Returns the style name

public static function **findStyle**(\$object)
Find a style by its properties

public function **hasContent**()
Return if the style has any content

public function **getContent**()
Returns the style content

public function **getInline**()
Return the style inline code

public function **getContents**()

public function **show**(\$inline = FALSE)
Show the style

TTable extends TElement*Creates a table layout, with rows and columns***Methods**

public function **__construct**()
Class Constructor

public static function **create**(\$properties)
Create a table

public function **addSection**(\$type)
Add section

public function **addRow**()
Add a new row (TTableRow object) to the table

public function `addRowSet()`

Add a new row (TTableRow object) with many cells

\$cells : Each argument is a row cell

public static function `fromData($array_data, $table_properties = null, $header_properties = null, $body_properties = null)`

Create a table from data array

\$array_data : Array with raw data

\$table_properties : Array of CSS properties for table

\$header_properties : Array of CSS properties for header

\$body_properties : Array of CSS properties for body

TTableCell extends TElement

TableCell: Represents a cell inside a table

Methods

public function `__construct($value, $tag = 'td')`

Class Constructor

\$value : TableCell content

TTableRow extends TElement

TableRow: Represents a row inside a table

Methods

public function `__construct($section = 'tbody')`

Class Constructor

public function `addCell($value)`

Add a new cell (TTableCell) to the Table Row

\$value : Cell Content

public function `addMultiCell()`

Add a multi-cell content to a table cell

\$cells : Each argument is a row cell

public function `clearChildren()`

Clear any child elements

TText extends TField

Text Widget (also known as Memo)

Methods

public function `__construct($name)`

Class Constructor

\$name : Widget's name

public function `setSize($width, $height = NULL)`

Define the widget's size

\$width : Widget's width

\$height : Widget's height

public function `getSize()`

Returns the size

public function `setMaxLength($length)`

Define max length

\$length : Max length

function **setExitAction**(TAction \$action)

Define the action to be executed when the user leaves the form field

\$action : TAction object

public function **setExitFunction**(\$function)

Set exit function

public function **forceLowerCase**()

Force lower case

public function **forceUpperCase**()

Force upper case

public function **getPostData**()

Return the post data

public function **show**()

Show the widget

TTextDisplay extends TElement

Text Display

Methods

public function **__construct**(\$value, \$color = null, \$fontSize = null, \$decoration = null)

Class Constructor

\$value : text content

\$color : text color

\$fontSize : text size

\$decoration : text decorations (b=bold, i=italic, u=underline)

public function **setLabel**(\$label)

Replace current label

public function **setSize**(\$width)

Define the Text's width

\$width : Field's width in pixels

public function **getSize**()

Returns the field size

public function **enableToggleVisibility**(\$toggleVisibility = TRUE)

Enable toggle visible

public function **setTransformer**(Callable \$callback)

Define a callback function to be applied over the column's data

\$callback : A function name of a method of an object

public function **show**()

TTime extends TEntry

TimePicker Widget

Methods

public function **__construct**(\$name)

Class Constructor

\$name : Name of the widget

```

public function setMask($mask, $replaceOnPost = FALSE)
    Define the field's mask
    $mask : Mask for the field (dd-mm-yyyy)

public function setOption($option, $value)
    Set extra datepicker options

public function setExitAction(TAction $action)
    Define the action to be executed when the user changes the field
    $action : TAction object

public function setChangeAction(TAction $action)
    Define the action to be executed when the user changes the field
    $action : TAction object

public function setChangeFunction($function)
    Set change function

public static function enableField($form_name, $field)
    Enable the field
    $form_name : Form name
    $field : Field name

public static function disableField($form_name, $field)
    Disable the field
    $form_name : Form name
    $field : Field name

public function show()
    Shows the widget at the screen

```

TTimeline extends TElement

Timeline

Methods

```

public function __construct()
    Class Constructor

public function setFinalIcon( $icon )
    Define the final timeline icon
    $icon : icon

public function setTimeDisplayMask( $mask )
    Define the labelmask
    $mask : Mask

public function setUseBothSides()
    Define the labelmask
    $mask : Mask

public function setItemTemplate($template)
    Set card item template for rendering
    $template : Template content

public function setTemplatePath( $template_path )
    Define the item template path
    $template_path : Template path

public function setItemDatabase($database)
    Set item min database

```

\$database : min database

public function addItem(\$id, \$title, \$content, \$date, \$icon, \$align = null, \$object = null)

Add Item

\$id : ID

\$title : Title

\$content : Item content

\$date : Item date

\$icon : Item icon

\$align : Item align

\$object : Item data object

public function addAction(TAction \$action, \$label, \$icon, \$display_condition = null)

Add Action

\$label : Action Label

\$action : Action

\$field : Action field

\$icon : Action icon

\$btn_class : Action button class

\$display_condition : Action display condition

public function show()

Show

TToast

Toast

Methods

public static function show(\$type, \$message, \$place = 'bottom center', \$icon = null)

Class Constructor

\$message : Message to be shown

TTransaction

Manage Database transactions

Methods

public static function open(\$database, \$dbinfo = NULL)

Open a connection and Initiates a transaction

\$database : Name of the database (an INI file).

\$dbinfo : Optional array with database information

public static function openFake(\$database)

Open fake transaction

\$database : Name of the database (an INI file).

public static function get()

Returns the current active connection

public static function rollback()

Rollback all pending operations

public static function close()

Commit all the pending operations

public static function closeAll()

close all transactions

public static function **rollbackAll**()
rollback all transactions

public static function **setLoggerFunction**(Closure \$logger)
Assign a Logger closure function
\$logger : A Closure

public static function **setLogger**(?AdiantiLoggerInterface \$logger = null)
Assign a Logger strategy
\$logger : A TLogger child object

public static function **log**(\$message)
Write a message in the LOG file, using the user strategy
\$message : Message to be logged

public static function **getDatabase**()
Return the Database Name

public static function **getDatabaseInfo**()
Returns the Database Information

public static function **getUniqId**()
Returns the Transaction uniqid

public static function **dump**(\$file = null)
Enable transaction log

TTreeView extends TElement

TreeView

Methods

public function **__construct**()
Class Constructor

public function **setTransformer**(\$callback)
Set node transformer

public function **setFolderTransformer**(\$callback)
Set node transformer

public function **setSize**(\$width)
Set size
\$size : width

public function **setItemIcon**(\$icon)
Set item icon
\$icon : icon location

public function **setItemAction**(\$action)
Set item action
\$action : icon action

public function **collapse**()
Collapse the Tree

public function **fromArray**(\$tree)
Fill treeview from an multi-dimensional array
multi-dimensional : array

TUniqueSearch extends TMultiSearch

Unique Search Widget

Methods

public function `__construct`(\$name)

Class Constructor

\$name : Widget's name

public function `setValue`(\$value)

Set value

public function `getPostData`()

Return the post data

public function `getSize`()

Returns the size

public function `show`()

Show the component

TVBox extends TElement

Vertical Box

Methods

public function `__construct`()

Class Constructor

public function `add`(\$child)

Add an child element

\$child : Any object that implements the show() method

public function `addColSet`()

Add a new col with many cells

\$cells : Each argument is a row cell

public static function `pack`()

Static method for pack content

\$cells : Each argument is a cell

TWindow extends TElement

Window Container (jQueryDialog wrapper)

Methods

public function `__construct`()

Constructor method

public function `setTargetContainer`(\$container)

public function `getId`()

Returns ID

public static function `create`(\$title, \$width, \$height, \$params = null)

Create a window

public function `removePadding`()

Remove padding

```

public function removeTitleBar()
    Remove titlebar

public function setDialogClass($class)
    Set Dialog class
    $class : Class name

public function setStackOrder($order)
    Define the stack order (zIndex)
    $order : Stack order

public function setTitle($title)
    Define the window's title
    $title : Window's title

public function setModal($modal)
    Turn on/off modal
    $modal : Boolean

public function disableEscape()
    Disable Escape

public function disableScrolling()
    Disable scrolling

public function enableScrolling()
    Enable scrolling

public function setSize($width, $height)
    Define the window's size
    $width : Window's width
    $height : Window's height

public function setMinWidth($percent, $absolute)
    Define the window's min width between percent and absolute
    $percent : width
    $absolute : width

public function setPosition($x, $y)
    Define the top corner positions
    $x : left coordinate
    $y : top coordinate

public function setProperty($property, $value)
    Define the Property value
    $property : Property name
    $value : Property value

public function add($content)
    Add some content to the window
    $content : Any object that implements the show() method

public function setCloseAction(TAction $action)
    set close action
    $action : close action

public static function blockUI($timeout = null)
    Block UI

public static function unBlockUI($timeout = null)
    Unblock UI

```


public static function **closeWindow**(\$id = null)
Close JQueryDialog's

public static function **closeAll**()
Close all windows

public static function **closeWindowByName**(\$name)
Close window by name of controller

public function **show**()
Decide wich action to take and show the page

public function **run**()
Interprets an action based at the URL parameters

public function **setPageName**(\$name)
Set page name

public function **getClassName**()
Return the Page name

public function **setIsWrapped**(\$bool)
Define if the element is wrapped inside another one
@bool : Boolean TRUE if is wrapped

TXMLBreadCrumb extends **TBreadCrumb**

XMLBreadCrumb

Methods

public function **__construct**(\$xml_file, \$controller)
Handle paths from a XML file
\$xml_file : path for the file

public function **getPath**(\$controller)
Return the controller path