ClickBlocks\Web\POM\View

General information

Inheritance	no
Subclasses	no
Interfaces	ArrayAccess
Source	Framework/web/view.php

This class represents View component in MVC design pattern. It contains methods that allow you to add CSS or JS files on your web page, parse the XHTML template of the page, synchronize control view state information between server and clients sides, and, of course, find web control objects in POM (Page Object Mode).
The instance of **View** is a value of property **view** of the active page class. And some class methods such as **parse()**, **assign()**, **process()** and **invoke()** are automatically called by the front controller.

Besides of class **View** there is global function **ID()** in file **view.php**. This function is intended for getting unique identifier of a web control through its logic identifier.

ID()

```
string|boolean function ID(string $id)
```

$id	string	the logic control identifier.

Returns unique control identifier or FALSE if the control with such logic ID dose not exist.

> This global function is useful in page templates when you need to get unique ID of some control.

Public non-static properties

tpl

```
public ClickBlocks\Core\Template $tpl
```

Via this property you can manage variables of the XHTML page template.

Public static methods

inParsing()

```
public static boolean inParsing()
```

Returns TRUE if the view of the active page is in state of parsing and FALSE otherwise.

encodePHPTags()

```
public static mixed encodePHPTags(string $xhtml, mixed &$marks)
```

$xhtml	string	the template string of the view or path to the template file.
$marks	mixed	some variable which will contain encoded PHP code fragments from **$xhtml**.

It processes HTML template of the page so that the parsing is made possible. It returns HTML templates of the page in which all PHP code fragments are encoded.

decodePHPTags()

```
public static mixed decodePHPTags(mixed $obj, array $marks)
```

$obj	mixed	an object to decode. It can be a control object, an array or the page template.
$marks	array	array of encoded PHP code fragments.

Decodes previously encoded fragments of PHP code in HTML template or in control properties and attributes. Returns the given object with decoded fragments of PHP code.

evaluate()

```
public static mixed evaluate(string $value, array $marks)
```

$value	string	some string containing PHP code or being a string of PHP code with PHP code marker in the beginning.
$marks	array	the previously stored encoded fragments of PHP code contained in the page template.

Executes PHP code in the given string. The result is a new string with executed PHP code or a PHP object if the given string is a marked string of PHP code. For example,

```
$v1 = 'Hello <?php echo 'world'; ?>';
echo View::evaluate($v1);
// The result is string "Hello World"
...
$v2 = "php::[1 => 'one', 'two', 'three']";
print_r(View::evaluate($v2));
// The result is array
//
// Array
// (
//     [1] => one
//     [2] => two
//     [3] => three
// )
```

analyze()

```
public static array analyze(string $template, array $vars = null)
```

$template	string	the page template or path to the template file.
$vars	array	template variables for the template preprocessing.

Parses arbitrary template and returns information about this template and its controls. The method returns an array that has the following structure:

```
[
 'dtd'      => [string] // DTD of the XHTML document.
 'title'    => [string] // Title (content of tag <title>) of the page.
 'meta'     => [array]  // Meta information of the XHTML document.
 'js'       => [array]  // Information about all JS scripts included in the web page.
 'css'      => [array]  // Information about all CSS scripts included in the web page.
 'html'     => [string] // Parsed XHTML templated of the page.
```

```
 'controls' => [array]  // Page controls that constitute the PAge Object Model (POM).
]
```

getControlPlaceholder()

```
public static string getControlPlaceholder(string $uniqueID)
```

$uniqueID	string	The control unique identifier.

Returns template placeholder for the given control.

Public non-static methods

__construct()

```
public void __construct(string $template = null)
```

$template	string	the page template or path to the template file.

The class constructor. Initializes property **$tpl** and handles Ajax file downloading.

offsetSet()

```
public void offsetSet(string $name, mixed $value)
```

$name	string	the variable name.
$value	mixed	the variable value.

Sets template variable for preprocessing.

offsetExists()

```
public boolean offsetExists(string $name)
```

$name	string	the variable name.

Checks existence of the template variable for preprocessing.

offsetUnset()

```
public void offsetUnset(string $name)
```

$name	string	the variable name.

Removes the template variable for preprocessing.

offsetGet()

```
public mixed &offsetGet(string $name)
```

$name	string	the variable name.

Returns value of the given template variable.

getVars()

```
public array getVars()
```

Returns array of all template variables for the template preprocessing.

setVars()

```
public void setVars(array $vars)
```

$vars	array	the template variables.

Sets template variables for the template preprocessing.

getDTD()

```
public string getDTD()
```

Returns DTD string of the XHTML document.

setDTD()

```
public self setDTD(string $dtd)
```

$dtd	string	DTD of the XHTML document.

Sets DTD of the XHTML document.

getTitle()

```
public string|array getTitle(boolean $withAttributes = false)
```

$withAttributes	boolean	determines whether to return the page title together attributes of <title> tag.

Returns the page title if the only method parameter is FALSE and otherwise it returns the two-element array of the following structure:

```
[
  'title'      => [string] // The page title.
  'attributes' => [array]  // Attributes of <title> tag.
]
```

setTitle()

```
public self setTitle(string $title = null, array $attributes = null)
```

$title	string	the page title.
$attributes	array	the attributes of <title> tag.

Sets the page title and/or attributes of <title> tag.

> You can set only the page title or only attributes of the <title> tag if you pass NULL instead of one of the two method parameters. In this case the page title or tag attributes will not be changed.

addMeta()

```
public self addMeta(array $attributes)
```

$attributes	array	attributes of the <meta> tag.

Adds meta information on the web page.

setMeta()

```
public self setMeta(string $id, array $attributes)
```

$id	string	some unique identifier of the given meta information.
$attributes	array	attributes of the <meta> tag.

Adds meta information on the web page and associates this information with some unique identifier.

getMeta()

```
public array getMeta(string $id)
```

$id	string	unique identifier of the required meta information.

Returns meta information associated with the given unique identifier. If meta information with the given ID is not found, it returns FALSE.

removeMeta()

```
public self removeMeta(string $id)
```

$id	string	the unique identifier of the meta information.

Removes meta information that associated with the given unique identifier.

getAllMeta()

```
public array getAllMeta()
```

Returns all meta data of the web page.

addCSS()

```
public self addCSS(array $attributes, string $style = null, integer $order = null)
```

$attributes	array	attributes of <style> or <link> tag.
$style	string	inline CSS (content of <style> tag).
$order	integer	an integer number that specify attaching order of CSS files on the page.

Adds CSS file or css string on the web page. If attributes "href" is defined then it will be used as a unique identifier of this CSS.

setCSS()

```
public self setCSS(string $id, array $attributes, string $style = null, integer $order = null)
```

$id	string	unique identifier of the attaching CSS.
$attributes	array	attributes of <style> or <link> tag.
$style	string	inline CSS (content of <style> tag).
$order	integer	an integer number that specify attaching order of CSS files on the page.

Attaches CSS file (or inline CSS) to the page. The attached CSS is associated with the given unique identifier.

getCSS()

```
public array getCSS(string $id)
```

$id	string	unique identifier of the attached CSS.

Returns information about CSS file (or inline CSS) that included in the page.

removeCSS()

```
public self removeCSS(string $id)
```

$id	string	unique identifier of the attached CSS.

Removes CSS that was previously added on the page.

getAllCSS()

```
public array getAllCSS()
```

Returns the list of all CSS that where added on the page.

addJS()

```
public self addJS(array $attributes, string $script = null, boolean $inHead = true, integer $order = null)
```

$attributes	array	attributes of <script> tag.
$script	string	inline JavaScript (content of <script> tag).
$inHead	boolean	determines the placement of the adding script: in the head section of the HTML or in the bottom of it (before closed <body> tag).
$order	integer	determines the order in which scripts are attached to the page.

Adds JS on the web page. If attribute "src" is defined, it will be used as a unique identifier of the adding JS.

setJS()

```
public self setJS(string $id, array $attributes, string $script = null, boolean $inHead = true, integer $order = null)
```

$id	string	unique identifier of the adding JS.
$attributes	array	attributes of <script> tag.

$script	string	inline JavaScript (content of <script> tag).
$inHead	boolean	determines the placement of the adding script: in the head section of the HTML or in the bottom of it (before closed <body> tag).
$order	integer	determines the order in which scripts are attached to the page.

Adds JS on the web page and associates it with some unique identifier.

getJS()

```
public array getJS(string $id, boolean $inHead = true)
```

$id	string	unique identifier of the adding JS.
$inHead	boolean	determines the placement of the adding script: in the head section of the HTML or in the bottom of it (before closed <body> tag).

Returns information about attached JS that associated with the given unique identifier.

removeJS()

```
public self removeJS(string $id, boolean $inHead = true)
```

$id	string	unique identifier of the adding JS.
$inHead	boolean	determines the placement of the adding script: in the head section of the HTML or in the bottom of it (before closed <body> tag).

Removes JS attached to the page.

getAllJS()

```
public array getAllJS()
```

Returns the list of all JS that where added on the page.

action()

```
public self action(string $action, mixed $param1, mixed $param2, ..., integer $delay = 0)
```

$action	string	the command name that determines actions that should be executed on the client side.
$param1, **$param2**, ...	mixed	parameters of the particular command.
$delay	integer	the delay in seconds of command execution.

Establishes JS code that should be performed on the client side. This method can be used both under Ajax requests and under usual requests.

All possible commands are listed in below:

- **alert** - shows simple alert message on the client side.

  ```
  action('alert', string $message, integer $delay = 0)
  ```

  - **$message** - the text message that should be shown.
  - **$delay** - the delay in seconds of command execution.

- **redirect** - redirects to the given URL.

```
action('redirect', string $url, integer $delay = 0)
```

  - **$url** - the URL to redirect.
  - **$delay** - the delay in seconds of command execution.

- **reload** - reloads the current web page.

```
action('reload', integer $delay = 0)
```

  - **$delay** - the delay in seconds of command execution.

- **addClass** - added CSS class to the given DOM element.

```
action('addClass', string $id, string $class, integer $delay = 0)
```

  - **$id** - value of attribute "id" of the required DOM element.
  - **$class** - name of CSS class that should be added to the given element.
  - **$delay** - the delay in seconds of command execution.

- **toggleClass** - toggles CSS class of the given element.

```
action('toggleClass', string $id, string $class, integer $delay = 0)
```

  - **$id** - value of attribute "id" of the required DOM element.
  - **$class** - CSS class that should be added to or removed from the required DOM element.
  - **$delay** - the delay in seconds of command execution.

- **removeClass** - removes CSS class from the given DOM element.

```
action('removeClass', string $id, string $class, integer $delay = 0)
```

  - **$id** - value of attribute "id" of the required DOM element.
  - **$class** - CSS class that should be removed from the required DOM element.
  - **$delay** - the delay in seconds of command execution.

- **insert** - sets property **innerHTML** of the given DOM element.

```
action('insert', string $id, string $html, integer $delay = 0)
```

  - **$id** - value of attribute "id" of the required DOM element.
  - **$html** - new value of property **innerHTML** of the given DOM element.
  - **$delay** - the delay in seconds of command execution.

- **replace** - replaces the given DOM element with the provided HTML content.

```
action('replace', string $id, string $html, integer $delay = 0)
```

  - **$id** - value of attribute "id" of the required DOM element.
  - **$html** - the HTML string to replace.
  - **$delay** - the delay in seconds of command execution.

- **inject** - according to the given mode adds the provided HTML content after or before the given DOM element or at the top or bottom of its child elements.

```
action('inject', string $id, string $html, string $mode, integer $delay = 0)
```

  - **$id** - value of attribute "id" of the required DOM element.
  - **$html** - the HTML string to inject.
  - **$mode** - the inject mode. Valid values are "top", "bottom", "before", "after".

- **$delay** - the delay in seconds of command execution.

- **remove** - removes the given DOM element.

```
action('remove', string $id, integer $delay = 0)
```

  - **$id** - value of attribute "id" of the required DOM element.
  - **$delay** - the delay in seconds of command execution.

- **focus** - sets focus to the given DOM element.

```
action('focus', string $id, integer $delay = 0)
```

  - **$id** - value of attribute "id" of the required DOM element.
  - **$delay** - the delay in seconds of command execution.

- **display** - sets, removes or toggles value of CSS property "display" of the given DOM element.

```
action('display', string $id, string $display, integer $expire = 0, integer $delay = 0)
```

  - **$id** - value of attribute "id" of the required DOM element.
  - **$display** - the value of CSS property "display" or NULL if you want to toggle between "display:none;" and default display value.
  - **$expire** - the time in seconds, upon the expiration of which the property "display" will restore its original value. It should be NULL or 0 if you want to toggle value of property "display".
  - **$delay** - the delay in seconds of command execution.

- **message** - inserts some HTML content in the given DOM element and after the provided time removes that content.

```
action('message', string $id, string $html, integer $expire = 0, integer $delay = 0)
```

  - **$id** - value of attribute "id" of the required DOM element.
  - **$html** - the HTML content that will be inserted to the given DOM element.
  - **$expire** - the time in seconds, upon the expiration of which the provided HTML content will be removed from the DOM element.
  - **$delay** - the delay in seconds of command execution.

- **download** - downloads the given file.

```
action('download', string $file, string $filename = null, string $contentType = null, boolean $deleteAfterDownload = false)
```

  - **$file** - the full path to the downloading file.
  - **$filename** - the name of the downloading file.
  - **$contentType** - the mime type of the downloading file.
  - **deleteAfterDownload** - determines whether the file should be deleted after download.

- **script** - executes arbitrary JavaScript on the client. This command exists in two variants of method call.

```
action('script', string $js, integer $time = 0)
action(string $js, integer $time = 0)
```

  - **$js** - some JS code to execute.
  - **$delay** - the delay in seconds of JS code execution.

> Command name is case insensitive.

attach()

```
public self attach(ClickBlocks\Web\POM\Control $ctrl)
```

$ctrl	ClickBlocks\Web\POM\Control	a control that should be attached to the POM.

Attaches the given control object to the view.

isAttached()

```
public boolean isAttached(mixed $ctrl)
```

$ctrl	string, ClickBlocks\Web\POM\Control	a control object or its unique identifier.

Returns TRUE if the given control is attached to the view and FALSE otherwise.

get()

```
public boolean|ClickBlocks\Web\POM\Control get(string $id, boolean $searchRecursively = true,
                                               ClickBlocks\Web\POM\Control $context = null)
```

$id	string	unique or logic identifier of the required control.
$searchRecursively	boolean	determines whether the control search should be recursive.
$context	ClickBlocks\Web\POM\Control	the panel, only inside which the search should be performed.

Searches the required control in the page object model and returns its instance. The method returns FALSE if the required control is not found.

clean()

```
public self clean(string $id, boolean $searchRecursively = true)
```

$id	string	unique or logic identifier of the required control.
$searchRecursively	boolean	determines whether the method should be recursively applied to all child panels of the given panel.

Restores default value of the control property "value" (if it has that property). If the given control is a panel, its children will also restore default value of their properties "value".

check()

```
public self check(string $id, boolean $flag = true, boolean $searchRecursively = true)
```

$id	string	unique or logic identifier of the panel.
$flag	boolean	determines whether each checkbox will be checked or not.
$searchRecursively	boolean	determines whether the method should be recursively applied to all child panels of the given panel.

Checks or unchecks checkboxes that contained in the given panel.

getFormValues()

```
public array|boolean getFormValues(string $id = null, boolean $searchRecursively = true)
```

$id	string	the logic or unique identifier of the panel.
searchRecursively	boolean	determines whether values of controls of nested panels will also be gathered.

Returns associative array of values of form (panel) controls. The keys of this array are logic identifiers of the panel(s) controls and its values are values of property **value** of the panel(s) controls.

The method returns FALSE if the given control is not a panel or if a panel with the given identifier does not exist.

> If the panel identifier is not defined, the identifier of the **body** control will be used.

setFormValues()

```
public self|boolean setFormValues(string $id, array $values, boolean $searchRecursively = true)
```

$id	string	the logic or unique identifier of the panel.
$values	array	the values to be assigned to.
$searchRecursively	boolean	determines whether the given values will be also assigned to the controls of nested panels.

Assigns values to the form (panel) controls. The keys of values array should be logic identifiers of the panel(s) controls and its values should be values of property **value** of the panel(s) controls.
The method returns FALSE if the given control is not a panel or if a panel with the given identifier does not exist.

invoke()

```
public self invoke(string $method, mixed $id = null)
```

$method	string	name of the required method of the control.
$id	mixed	unique (or logic) control identifier or control object.

Invokes the required method of the given control. If the given control is a panel, the required method will be recursively invoked for every its child control.

getValidators()

```
public array getValidators(string $groups = 'default')
```

$groups	string	comma separated validation groups or symbol "*".

Returns array of the validators of the given validation group(s).

> If **$groups** equals symbol "*" it means that all validators of all groups will be returned.

validate()

```
public boolean validate(string $groups = 'default', string $classInvalid = null, string $classValid = null)
```

$groups	string	comma separated validation groups or symbol "*", which means all validators.
$classInvalid	string	style class that will be applied to invalid controls.
$classValid	string	style class that will be applied to valid controls.

Launches validators of the given group(s). It returns TRUE if all validated controls are valid and FALSE otherwise.

reset()

```
public self reset(string $groups = 'default', string $classInvalid = null, string $classValid = null)
```

$groups	string	comma separated validation groups or symbol "*", which means all validators.

$classInvalid	string	style class that applied to invalid controls.
$classValid	string	style class that applied to valid controls.

Sets validators of the given group(s) to valid state. Validated controls are also set to valid state.

assign()

```
public boolean assign(string $UID, array $data, integer $timestamp)
```

$UID	string	unique identifier of the view.
$data	array	changes of control attributes received from the client side.
$timestamp	integer	the time when the given changes were made.

Merges attribute values of the client side controls with the same attributes of the server side controls. It returns FALSE if the current session is expired and TRUE otherwise.

process()

```
public self process(mixed $data)
```

$data	mixed	result of the Ajax request execution.

Forms HTTP response for the result of the Ajax request execution.

parse()

```
public self parse()
```

Parses the view template.

render()

```
public string render()
```

Renders the view and returns the rendered HTML of the page.

Protected static properties

process

```
protected static integer $process = 0;
```

Contains the number of parsing threads. This property is used for determining that the parsing process takes place in the moment.

emptyTags

```
protected static array $emptyTags = ['br' => 1, 'hr' => 1, 'meta' => 1, 'link' => 1, 'img' => 1, 'embed' => 1, 'param' => 1, 'inpu
```

List of all empty HTML tags.

Protected non-static properties

vars

```
protected array $vars = [];
```

Array of variables for the template preprocessing.

controls

```
protected array $controls = [];
```

Array of control objects that used for quick access to the required control object.

actions

```
protected array $actions = [];
```

Array of JS commands that should be performed on the client side.

dtd

```
protected string $dtd = '<!DOCTYPE html>';
```

DTD of the HTML document.

title

```
protected array $title = ['title' => '', 'attributes' => []];
```

Title of the page and attributes of the HTML <title> tag.

meta

```
protected array $meta = [];
```

Meta information of the web page.

css

```
protected array $css = [];
```

Array of all CSS files which placed on the page.

js

```
protected array $js = ['top' => [], 'bottom' => []];
```

Array of all JS files which placed on the page.

Protected non-static methods

renderAttributes()

```
protected string renderAttributes(array $attributes)
```

$attributes	array	attributes of some HTML tag.

Renders attributes of an HTML tag.

getCacheID()

```
protected string getCacheID(boolean $init)
```

$init	boolean	determines cache type of the controls view state.

Returns unique identifier of the view state cache. if **$init** is TRUE, the cache ID of initial view state is returned and otherwise the cache ID of intermediate view states is returned.

isExpired()

```
protected boolean isExpired(boolean $init = false)
```

$init	boolean	determines cache type of the controls view state.

Returns TRUE if the current session is expired and TRUE otherwise.

pull()

```
protected void pull(boolean $init = false)
```

$init	boolean	determines cache type of the controls view state.

Extracts the control view states from the cache.

push()

```
protected void push(boolean $init = false)
```

$init	boolean	determines cache type of the controls view state.

Puts the control view states into the cache.

commit()

```
protected void commit()
```

Calculates the view states of controls and stores them in the private class property.

merge()

```
protected void merge(array $data, integer $timestamp)
```

$data	array	new attribute values of the controls.
$timestamp	integer	the time when the changes of attribute values occurred.

Merges new values of control attributes with their old values.

compare()

```
protected void compare()
```

Compares the current values of attributes and properties of the controls with their old values and forms JS code for refreshing the changed controls on the client side.

prepareTemplate()

```
protected array prepareTemplate(string $template, array $vars = null)
```

$template	string	template string or path to the template file.
$vars	array	template variables for the template preprocessing.

Prepares template to the parsing. It returns the detailed information about the template.

parseTemplate()

```
protected array parseTemplate(array $ctx)
```

$ctx	array	array of the template detailed information.

Parses the view template and returns the updated detailed information about the template.