ClickBlocks\Web\POM\Control

General information

Inheritance	no
Subclasses	All classes of web controls.
Interfaces	ArrayAccess
Source	Framework/web/pom/controls/control.php

Control is the base abstract class for all web controls on the server side. It contains a lot of methods providing the general functionality:

- getting and setting view state of the control;
- managing control properties and attributes;
- rendering HTML of the control;
- getting and setting parent of the control;
- removing and creating copy of the control;
- setting CSS classes, styles for the control;
- adding JS events to the control;
- and much more.

All web controls have the following main properties and attributes:

Attributes

Name	Type	Default value	Description
id	string	automatically generated	the unique identifier of the control. It is readonly attribute.

Properties

Name	Type	Default value	Description
id	string	user defined	the logic control identifier.
visible	boolean	TRUE	determines whether or not the control is visible on the client side.

Public non-static methods

__construct()

```
public void __construct(string $id)
```

$id	string	the logic identifier of the control.

Constructor. Creates the unique control identifier that based on its logic identifier.

> Logic identifier of a control should contains only numeric or alphabetic symbols, or symbol "_".

isAttached()

```
public boolean isAttached()
```

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

getFullID()

```
public string getFullID()
```

Returns full logic identifier of the control. The full logic identifier is a dot-separated string containing logic identifiers of all parents of the given control along with its logic identifier.

getVS()

```
public array getVS()
```

Returns view state of the control.

setVS()

```
public self setVS(array $vs)
```

$vs	array	the view state of the control.

Sets view state of the control.

__set()

```
public void __set(string $attribute, mixed $value)
```

$attribute	string	the attribute name.
$value	mixed	the attribute value.

Sets value of the control attribute.

__get()

```
public mixed &__get(string $attribute)
```

$attribute	string	the attribute name.

Returns value of the control attribute.

__isset()

```
public boolean __isset(string $attribute)
```

$attribute	string	the attribute name.

Returns TRUE if value of the given attribute is defined and FALSE otherwise.

__unset()

```
public void __unset(string $attribute)
```

$attribute	string	the attribute name.

Removes the control attributes.

offsetSet()

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

$property	string	the property name.
$value	mixed	the property value.

Sets value of the control property.

offsetGet()

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

$property	string	the property name.

Returns value of the control property.

offsetExists()

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

$property	string	the property name.

Returns TRUE if the control has the given property and FALSE otherwise.

offsetUnset()

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

$property	string	the property name.

Removes value (sets it to NULL) of the control property.

callback()

```
public string callback(string $callback, boolean $isStatic = false)
```

$callback	string	the control method.
$isStatic	boolean	determines whether the control method is static.

Returns delegate string for the given control method. This string can be used on the client side for creating Ajax requests.

method()

```
public string method(string $callback, array $params = null, boolean $isStatic = false)
```

$callback	string	the control method.
$params	array	the method parameters.

$isStatic	boolean	determines whether the control method is static.

Returns JS code for invoking of the given control method through Ajax request. Example:

```
$ctrl = new Any('myctrl');
echo $ctrl->method('offsetSet', ['text', 'js::this.value']);
```

The above example will output:

```
$ajax.doit('ClickBlocks\\Web\\POM\\Any@myctrl53bb962aaac52542135877->offsetSet', 'text', this.value)
```

getEvents()

```
public array getEvents()
```

Returns the control events.

addEVent()

```
public self addEvent(string $id, string $type, string $callback, array $options = null)
```

$id	string	the unique identifier of the event.
$type	string	the DOM event type, such as "click" or "change", or custom event type.
$callback	string	the control method or marked JS code string of the event handler.
$options	array	the additional event parameters.

Adds new JS event to the control. Parameter **$options** can have the following elements:

```
[
 'params'      => [string], // parameters of the control method if $callback represents the control method.
 'check'       => [string], // JS function that will determine whether or not to trigger the given event handler.
 'toContainer' => [boolean] // determines whether or not the given event should be applied to the container tag of the control.
]
```

removeEvent()

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

$id	string	the unique identifier of the event.

Removes the control event.

hasContainer()

```
public boolean hasContainer()
```

Returns TRUE if the control has the container tag and FALSE otherwise. This method need to be overridden in child classes if your control has the container tag.

hasClass()

```
public boolean hasClass(string $class, boolean $applyToContainer = false)
```

$class	string	the CSS class.

$applyToContainer	boolean	determines whether the container tag of the control is considered.

Returns TRUE if the control has the given CSS class.

addClass()

```
public self addClass(string $class, boolean $applyToContainer = false)
```

$class	string	the CSS class.
$applyToContainer	boolean	determines whether the container tag of the control is considered.

Adds CSS class to the control.

removeClass()

```
public self removeClass(string $class, boolean $applyToContainer = false)
```

$class	string	the CSS class.
$applyToContainer	boolean	determines whether the container tag of the control is considered.

Removes CSS class from the control.

replaceClass()

```
public self replaceClass(string $class1, string $class2, boolean $applyToContainer = false)
```

$class1	string	CSS class to be replaced.
$class2	string	CSS class to replace.
$applyToContainer	boolean	determines whether the container tag of the control is considered.

Replaces CSS class with the other CSS class.

toggleClass()

```
public self toggleClass(string $class1, string $class2 = null, boolean $applyToContainer = false)
```

$class1	string	CSS class to toggle.
$class2	string	CSS class to replace.
$applyToContainer	boolean	determines whether the container tag of the control is considered.

Toggles CSS class.

hasStyle()

```
public boolean hasStyle(string $style, boolean $applyToContainer = false)
```

$style	string	the CSS property name.
$applyToContainer	boolean	determines whether the container tag of the control is considered.

Returns TRUE if the control has the given CSS property and FALSE if it hasn't.

addStyle()

```
public self addStyle(string $style, mixed $value, boolean $applyToContainer = false)
```

$style	string	the CSS property name.
$value	mixed	the CSS property value.
$applyToContainer	boolean	determines whether the container tag of the control is considered.

Adds new CSS property value to the control attribute "style". If the control hasn't the given property it will be added.

setStyle()

```
public self setStyle(string $style, mixed $value, boolean $applyToContainer = false)
```

$style	string	the CSS property name.
$value	mixed	the CSS property value.
$applyToContainer	boolean	determines whether the container tag of the control is considered.

Sets value of the CSS property of the control attribute "style". If CSS property with the given name does not exists it won't be added to the control.

getStyle()

```
public mixed getStyle(string $style, boolean $applyToContainer = false)
```

$style	string	the CSS property name.
$applyToContainer	boolean	determines whether the container tag of the control is considered.

Returns value of the CSS property of the control attribute "style".

removeStyle()

```
public self removeStyle(string $style, boolean $applyToContainer = false)
```

$style	string	the CSS property name.
$applyToContainer	boolean	determines whether the container tag of the control is considered.

Removes CSS property from the control attribute "style".

toggleStyle()

```
public self toggleStyle(string $style, mixed $value, boolean $applyToContainer = false)
```

$style	string	the CSS property name.
$value	mixed	the CSS property value to toggle.
$applyToContainer	boolean	determines whether the container tag of the control is considered.

Toggles CSS property value from the control attribute "style".

isCreated()

```
public boolean isCreated()
```

Returns TRUE if the control has just created and not attached to the current view. Otherwise it returns FALSE.

isRefreshed()

```
public boolean isRefreshed()
```

Returns TRUE if the control has an attribute or a property changed, or if method **refresh()** was invoked with parameter TRUE. Otherwise it returns FALSE.

isRemoved()

```
public boolean isRemoved()
```

Returns TRUE if the control was removed and FALSE otherwise.

getCreationInfo()

```
public array getCreationInfo()
```

Returns information about placement of the newly created control.

refresh()

```
public self refresh(boolean $flag = true)
```

$flag	boolean	determines whether need to refresh the control on the client side.

This method can be used to force the control to refresh on the client side even though no its attributes or properties were changed.

remove()

```
public self remove()
```

Removes the control from the current view.

compare()

```
public string|array compare(array $vs)
```

$vs	array	the old view state of the control.

Returns array of the changed or removed control attributes, or the rendered control HTML. This method is automatically invoked by the current view to refresh DOM of the web page.

getParent()

```
public ClickBlocks\Web\POM\Panel getParent()
```

Returns the parent control. If the control does not have the parent, it returns FALSE.

setParent()

```
public self setParent(ClickBlocks\Web\POM\Control $parent, string $mode = null, string $id = null)
```

$parent	ClickBlocks\Web\POM\Control	the control parent.
$mode	string	determines the placement of the control in the parent template.
$id	string	the unique or logic identifier of one of the parent control or value of the attribute "id" of some element in the parent template.

Sets parent of the control. The control will be attached to the view if it isn't and its parent is attached to the view.

The valid values of mode are:

- **ClickBlocks\Utils\DOMDocumentEx::DOM_INJECT_TOP** - the control will be added at the top of the parent template.
- **ClickBlocks\Utils\DOMDocumentEx::DOM_INJECT_BOTTOM** - the control will be added at the bottom of the parent template.
- **ClickBlocks\Utils\DOMDocumentEx::DOM_INJECT_BEFORE** - the control will be added before the specified control of the parent.
- **ClickBlocks\Utils\DOMDocumentEx::DOM_INJECT_AFTER** - the control will be added after the specified control of the parent.

The below examples explain usage of method **setParent()**:

Let's say we have the following XHTML template of the parent and the PHP code that adds new controls on the parent panel:

```
<panel id="prnt">
  <div id="cntr"><TextBox id="txt1" /></div>
  <div><TextBox id="txt2" /></div>
</panel>
```

```
...
// Gets parent control.
$parent = $this->get('prnt');

// Creates the first control and adds it at the top of the parent template.
$ctrl = new Any('ctrl1', 'label', 'New Control #1');
$ctrl->setParent($parent, Utils\DOMDocumentEx::DOM_INJECT_TOP);

// Creates the second control and adds it at the bottom of the parent template.
$ctrl = new Any('ctrl2', 'label', 'New Control #2');
$ctrl->setParent($parent, Utils\DOMDocumentEx::DOM_INJECT_BOTTOM);

// Creates the third control and adds it before the control "txt2" of the parent.
$ctrl = new Any('ctrl3', 'label', 'New Control #3');
$ctrl->setParent($parent, Utils\DOMDocumentEx::DOM_INJECT_BEFORE, 'txt2');

// Creates the fourth control and adds it after <div> element "cntr".
$ctrl = new Any('ctrl4', 'label', 'New Control #4');
$ctrl->setParent($parent, Utils\DOMDocumentEx::DOM_INJECT_AFTER, 'cntr');
```

Then, after execution of the above code we will have the following XHTML template of the parent:

```
<panel id="prnt">
  <label id="ctrl1">New Control #1</label>
  <div id="cntr"><TextBox id="txt1" /></div>
  <label id="ctrl4">New Control #4</label>
  <div>
    <label id="ctrl3">New Control #3</label>
    <TextBox id="txt2" />
  </div>
  <label id="ctrl2">New Control #2</label>
</panel>
```

copy()

```
public ClickBlocks\Web\POM\Control copy(string $id = null)
```

$id	string	the logic identifier of the control copy.

Creates full copy of the control. If **$id** is not defined the logic identifier of the original control is used.

render()

```
abstract public string render()
```

Returns rendered HTML of the control.

__toString()

```
public string __toString()
```

Returns rendered HTML of the control.

renderXHTML()

```
public string renderXHTML()
```

Returns XHTML of the control.

Protected non-static properties

isRefreshed

```
protected boolean $isRefreshed = false
```

Determines whether the control properties or attributes were changed and the control should be refreshed on the client side.

isRemoved

```
protected boolean $isRemoved = false
```

Determines whether the control was removed and it should be removed on the client side.

isCreated

```
protected boolean $isCreated = false
```

Determines whether the control has just created and it should be added on the page on the client side.

creationInfo

```
protected array $creationInfo
```

Information about the placement on the web page of the newly created control.

parent

```
protected string|ClickBlocks\Web\POM\Control $parent
```

Unique identifier of the parent control or its object.

ctrl

```
protected string $ctrl;
```

The control type. This type is shown as attribute "data-ctrl" in the control tag on the web page.

attributes

```
protected array $attributes = []
```

HTML global attributes of the control.

dataAttributes

```
protected array $dataAttributes = []
```

Non-standard control attributes, that should be rendered as "data-" attributes on the web page.

properties

```
protected array $properties = []
```

The control properties.

events

```
protected array $events = []
```

The control events.

Protected non-static methods

renderAttributes()

```
protected string renderAttributes(boolean $renderBaseAttributes = true)
```

$renderBaseAttributes	boolean	determines whether the attributes of the main control tag are rendered.

Renders HTML of the control attributes.

renderXHTMLAttributes()

```
protected string renderXHTMLAttributes()
```

Returns the control attributes and properties rendered to XHTML.

invisible()

```
protected string invisible()
```

Returns the standard HTML for an invisible control.