

# Bootstrap

Le module **Bootstrap** permet de générer les composants Bootstrap et leur fonctionnement par défaut à partir des contrôleurs Phalcon, pour ensuite les afficher dans les vues.

## -- Pré-requis

### -- Injection du service

Il est nécessaire d'injecter le service **JQuery** au démarrage de l'application, et d'instancier **Bootstrap** :

```
$di->set("jquery", function(){
    $jquery= new JsUtils(array("driver"=>"jQuery"));
    $jquery->bootstrap(new Bootstrap()); //for Twitter Bootstrap
    return $jquery;
});
```

### -- Inclusion des fichiers js et css

#### -- Inclusion par défaut

Par défaut, les scripts insérés sont les dernières versions de Bootstrap disponibles sur MaxCDN

#### Génération des CDNs dans le contrôleur :

```
class IndexController extends ControllerBase{
    public function indexAction(){
        $this->view->setVar("cdn", $this->jquery->genCDNs());
        ...
    }
    ...
}
```

#### Intégration dans les vues des inclusions générées :

```
<html>
  <head>
    {{cdn}}
    ...
  </head>
  ...
</html>
```

## -- Inclusion locale

Si les fichiers sont situés localement (dans **public/css et public/js**), il est nécessaire d'instancier un objet CDN, initialisé avec les chemins locaux :

```
$di->set("jquery",function(){
    $jquery= new JsUtils(array("driver"=>"Jquery"));
    $jquery->bootstrap(new Bootstrap());//for Twitter Bootstrap
    $cdn=new CDNBootstrap("");
    $jquery->setCDNs($cdn->setJsUrl("js/bootstrap.min.js")->setcssUrl("css/bootstrap.min.css"));
    return $jquery;
});
```

## -- Changement de version

Il est également possible de changer de version Bootstrap :

```
$di->set("jquery",function(){
    $jquery= new JsUtils(array("driver"=>"Jquery"));
    $jquery->bootstrap(new Bootstrap());//for Twitter Bootstrap
    $jquery->setCDNs(new CDNBootstrap("3.3.2"));
    return $jquery;
});
```

## -- Composants

### -- Button

Classe	<a href="#">HtmlButton</a>
⇒Hérite de	<a href="#">HtmlDoubleElement</a>



### -- Exemple complet

#### Contrôleur

Comportement attendu : Sur click du bouton **Masquer**, le bouton **Okay** est caché

```
class IndexController extends Phalcon\Mvc\Controller{
    public function buttonsAction(){
        $bootstrap=$this->jquery->bootstrap();
        $bootstrap->htmlButton("btOkay", "Okay");
    }
}
```

```

        $bootstrap->htmlButton("btMasquer", "Masquer Bouton Okay", "btn-
primary", $this->jquery->hide("#btOkay"));

        $this->jquery->compile($this->view);
    }
    ...
}

```

La compilation génère :

1. les éléments HTML et les passe en tant que variables à la vue dans un tableau associatif sous la forme **q["identifiant"]**
2. le script javascript correspondant à la dynamique et le passe dans la variable **script\_foot** à la vue

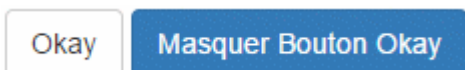
**Vue**

```

{{q['btOkay']}} {{q["btMasquer"]}}

{{script_foot}}

```



## -- Variations

### Affichage direct dans le contrôleur

```

class IndexController extends Phalcon\Mvc\Controller{

    public function buttons2Action(){
        $bt1=new HtmlButton("btOkay", "Okay");
        $bt2=new HtmlButton("btMasquer", "Masquer Bouton Okay", "btn-
primary", $this->jquery->hide("#btOkay"));
        echo $bt1->compile($this->jquery);
        echo $bt2->compile($this->jquery);
        echo $this->jquery->compile();
        $this->view->disableLevel(View::LEVEL_ACTION_VIEW);
    }
    ...
}

```

### Passage explicite à la vue

Contrôleur :

```

class IndexController extends Phalcon\Mvc\Controller{

```

```

public function buttons3Action(){
    $bt1=new HtmlButton("btOkay", "Okay");
    $bt2=new HtmlButton("btMasquer", "Masquer Bouton Okay", "btn-
primary", $this->jquery->hide("#btOkay"));
$this->view->setVars(array("bt1"=>$bt1->compile($this->jquery), "bt2"=>$bt2->compile
($this->jquery));
    $this->jquery->compile($this->view, "script");
}
...

```

Vue :

```

{{bt1}} {{bt2}}

{{script}}

```

### -- Modification des propriétés des composants

#### Une à une, par valeur :

```

$bt1=new HtmlButton("btOkay", "Okay");
$bt1->setStyle("btn-primary");

```

#### Une à une, par index :

```

$bt1=new HtmlButton("btOkay", "Okay");
$bt1->setStyle(0);

```

0 ⇒ "btn-default"

#### A la chaîne :

```

$bt1=new HtmlButton("btOkay", "Okay");
$bt1->setSize(0) ->setStyle(2) ->addBadge("A masquer...");

```

#### A partir d'un array :

```

$bt1=new HtmlButton("btOkay", "Okay");
$bt1->fromArray(array("size"=>"btn-sm", "style"=>4));

```

#### En spécifiant des propriétés inexistantes dans la classe HtmlButton :

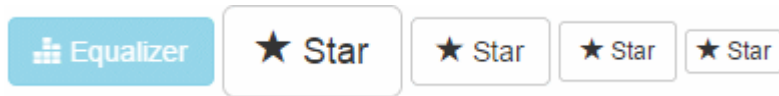
```

$bt1=new HtmlButton("btOkay", "Okay");
$bt1->setProperty("style", "font-size:30px;font-family: Harabara");

```

## -- GlyphButton

Classe	<a href="#">HtmlGlyphButton</a>
⇒ Hérite de	<a href="#">HtmlButton</a>



Le **GlyphButton** possède les mêmes propriétés et le même comportement que le bouton. Il possède juste la propriété supplémentaire **glyph**, permettant d'afficher un [glyphicon bootstrap](#)

### -- Création

```
class IndexController extends Phalcon\Mvc\Controller{
    public function glyphButtonAction(){
        $bt=new HtmlGlyphButton("glButton1","glyphicon-equalizer","Equalizer");
        echo $bt->compile();
        $this->view->disableLevel(View::LEVEL_ACTION_VIEW);
    }
    ...
}
```

### -- Affectations possibles de la propriété glyph

#### A l'instanciation

```
$bt=new HtmlGlyphButton("glButton1","glyphicon-equalizer","Equalizer");
```

#### Après instanciation

```
$bt=new HtmlGlyphButton("glButton1");
$bt->setGlyph("glyphicon-equalizer");
```

#### En utilisant la version courte

```
$bt=new HtmlGlyphButton("glButton1");
$bt->setGlyph("equalizer");
```

#### A partir de l'index du glyphicon

```
$bt=new HtmlGlyphButton("glButton1");
$bt->setGlyph(208);
```

Le tableau **CssRef::glyphIcons()** donne la liste complète des icônes utilisables

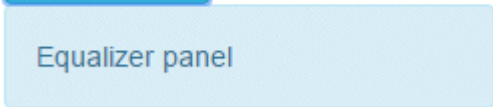
## -- Association d'un comportement

Sur click... Affichage du panel

```
class IndexController extends Phalcon\Mvc\Controller{
    public function glyphButtonAction(){
        $bt=new HtmlGlyphButton("glButton1","glyphicon-equalizer","Equalizer");
        $bt->onClick($this->jquery->show("#panel",2000));
        echo $bt->compile($this->jquery);
        echo "<div id='panel' class='alert alert-info'
style='display:none'>Equalizer panel</div>";
        echo $this->jquery->compile();
        $this->view->disableLevel(View::LEVEL_ACTION_VIEW);
    }
    ...
}
```

Autre évènements : sur entrée de la souris, affichage du panel ; masquage sur sortie...

```
class IndexController extends Phalcon\Mvc\Controller{
    public function glyphButtonAction(){
        $bt=new HtmlGlyphButton("glButton1","glyphicon-equalizer","Equalizer");
        $bt->on("mouseenter",$this->jquery->show("#panel"));
        $bt->on("mouseleave",$this->jquery->hide("#panel"));
        echo $bt->compile($this->jquery);
        echo "<div id='panel' class='alert alert-info'
style='display:none'>Equalizer panel</div>";
        echo $this->jquery->compile();
        $this->view->disableLevel(View::LEVEL_ACTION_VIEW);
    }
    ...
}
```

## -- Badges

Comme sur la plupart des composants Bootstrap, il est possible d'ajouter un Badge à un bouton :

```
$bt->addBadge("Off");
```



Modification des événements : Le texte du badge est modifié sur entrée et sortie de la souris...

```

    $bt->addBadge("Off");
    $bt->on("mouseenter", $this->jquery->show("#panel").$this->jquery->html("#gl1
    .badge", "On"));
    $bt->on("mouseleave", $this->jquery->hide("#panel").$this->jquery->html("#gl1
    .badge", "Off"));

```

### -- Labels

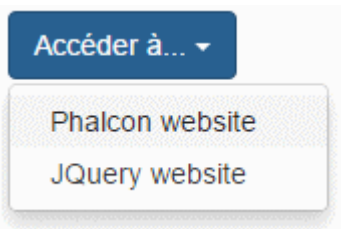
Comme sur la plupart des composants Bootstrap, il est possible d'ajouter un label à un bouton :

```
$bt->addLabel("New");
```



### -- Dropdown

<b>Classe</b>	<a href="#">HtmlDropdownbutton</a>
<b>⇒ Hérite de</b>	<a href="#">HtmlButton</a>



Classe [HtmlDropdown](#)

### --Création

```

class IndexController extends Phalcon\Mvc\Controller{
    ...
    public function dropdownAction(){
        $bootstrap=$this->jquery->bootstrap();
        $dropdown=$bootstrap->htmlDropdown("dd1", "Accéder à...", array("Phalcon
website", "jQuery website"));
        $dropdown->asButton();
        echo $dropdown->compile($this->jquery);
        echo $this->jquery->compile();
        $this->view->disableLevel(View::LEVEL_ACTION_VIEW);
    }
}

```

```
}  
...
```

## -- Attribution de href sur les liens

```
class IndexController extends Phalcon\Mvc\Controller{  
    ...  
    public function dropdownAction(){  
        $bootstrap=$this->jquery->bootstrap();  
        $dropdown=$bootstrap->htmlDropdown("dd1", "Accéder  
à...", array(array("caption"=>"Phalcon  
website", "href"=>"http://www.phalconphp.com/", "target"=>"_new"), "jQuery website"));  
        $dropdown->asButton();  
        echo $dropdown->compile($this->jquery);  
        echo $this->jquery->compile();  
        $this->view->disableLevel(View::LEVEL_ACTION_VIEW);  
    }  
    ...  
}
```

## -- Accès aux items

Un item est une instance de la classe [HtmlMenuItem](#).

```
$dropdown->getItem(1)->setHref("http://jquery.com");
```

## -- Ajout d'items supplémentaires

1 seul :

```
$dropdown->addItem("Javascript opération");
```

Ou plusieurs :

```
$dropdown->addItems(array("item1", "item2"));
```

## -- Association d'événements

Sur click...

```
$dropdown->addItem("Javascript opération")->onClick($this->jquery->show("#panel"));
```

**Pour prévenir le comportement par défaut** et faire en sorte que le menu ne disparaisse pas sur click...

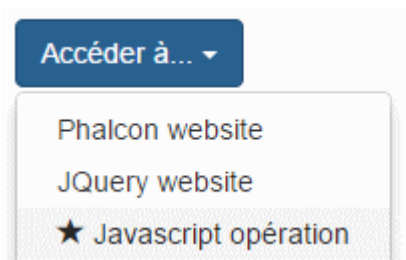
On utilise les derniers paramètres **stopPropagation** et **preventDefault** :

```
$dropdown->addItem("Javascript  
opération")->onClick($this->jquery->show("#panel"),true,true);
```

### -- Ajout de Glyphicons, labels, badges ou autres éléments

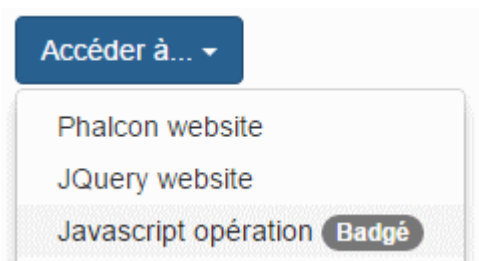
#### Glyphicon

```
$item=dropdown->getItem(2)->addGlyph("star");
```



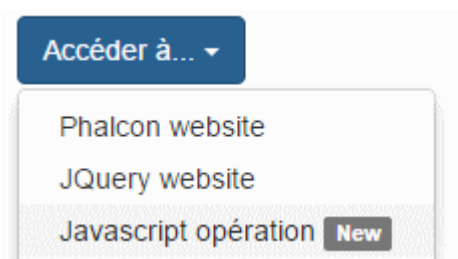
#### Badge

```
$item=dropdown->getItem(2)->addBadge("Badgé");
```



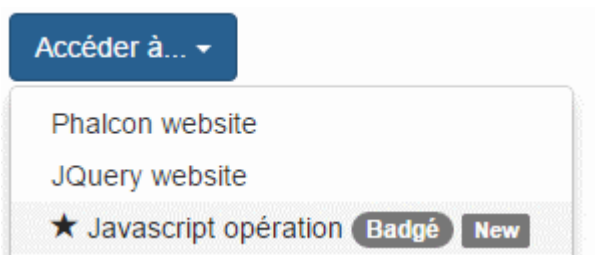
#### Label

```
$item=dropdown->getItem(2)->addLabel("New");
```



## Glyphicon, badge et label

```
$item=$dropdown->getItem(2) ->addGlyph("star") ->addBadge("Badgé") ->addLabel("New");
```



## -- Diviseurs

### dividers

Les 2 syntaxes sont équivalentes :

```
$dropdown->addItem("-");
```

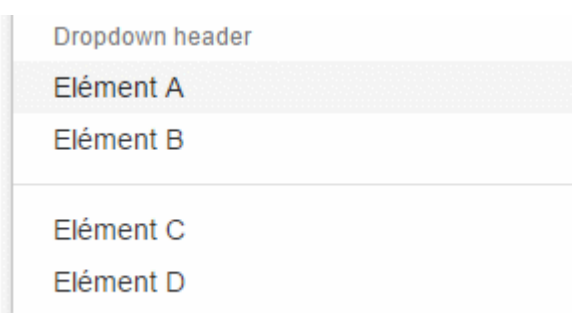
```
$dropdown->addDivider();
```

### headers

Les 2 syntaxes sont équivalentes :

```
$dropdown->addItem("-Dropdown Header");
```

```
$dropdown->addHeader(Dropdown Header);
```



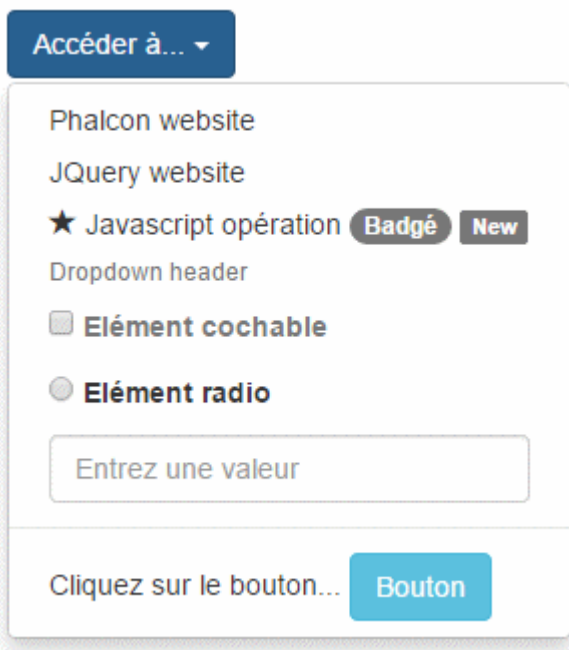
### -- Désactivation d'un élément

```
$dropdown->getItem(2)->disable();
```

### -- Insertion d'autres éléments

Le composant HtmlDropdown accepte d'autres éléments en tant que DropdownItem : checkbox, radio, input, button...

```
//Case à cocher
    $ck=new HtmlInputCheckbox("ck","Elément cochable");
    $dropdown->addItem("")->setContent($ck);
//Bouton radio
    $op=new HtmlInputRadio("op","Elément radio");
    $dropdown->addItem("")->setContent($op);
//Zone de texte
    $input=new HtmlInput("text1");
    $input->setProperty("placeholder", "Entrez une valeur");
    $item=$dropdown->addItem("")->setContent($input);
    $item->onClick("",true,true);//trick pour faire en sorte que le menu ne
disparaisse pas sur click
//Bouton
    $bt=new HtmlButton("bt","Bouton","btn-info");
    $item=$dropdown->addItem("Cliquez sur le bouton... ");
    $item->addContent($bt);
```



### -- Gestion des événements

TODO

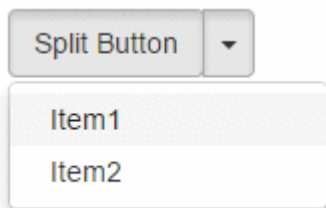
## -- SplitButton

<b>Classe</b>	<a href="#">HtmlSplitbutton</a>
<b>Hérite de</b>	<a href="#">HtmlDropdown</a>

### Exemple

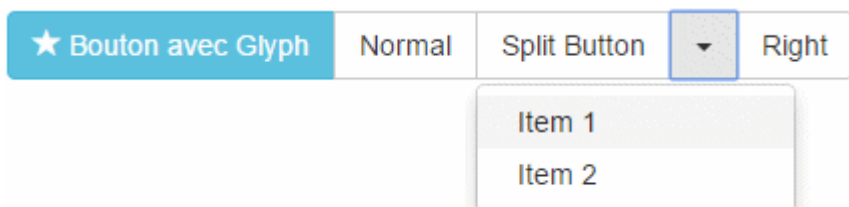
L'évènement **onButtonClick** permet d'agir sur le bouton associé.

```
class IndexController extends Phalcon\Mvc\Controller{
    public function splitAction(){
        $split=new HtmlSplitbutton("split1","Split Button",array("Item1","Item2"));
        $split->onButtonClick($this->jquery->show("#panel"));
        $split->onClick($this->jquery->hide("#panel"));
        echo $split->compile($this->jquery);
        echo "<div id='panel' class='alert alert-info' style='display:none'>Split
panel</div>";
        echo $this->jquery->compile();
        $this->view->disableLevel(View::LEVEL_ACTION_VIEW);
    }
    ...
}
```



## -- Button groups

<b>Classe</b>	<a href="#">HtmlButtongroups</a>
<b>⇒ Hérite de</b>	<a href="#">HtmlDoubleElement</a>



## -- Affichage direct dans le contrôleur

```
class IndexController extends Phalcon\Mvc\Controller{
    public function buttonGroupsAction(){
        $gr1=new HtmlButtongroups("gr1",array("Bouton1","Bouton2","Bouton3"));
    }
}
```

```

    echo $gr1->compile();
    $this->view->disableLevel(View::LEVEL_ACTION_VIEW);
}
...

```

Bouton1 Bouton2 Bouton3

## -- Création dans le contrôleur et passage à la vue

```

class IndexController extends Phalcon\Mvc\Controller{
    public function buttonGroupsAction(){
        $bootstrap=$this->jquery->bootstrap();
        $gr1=$bootstrap->htmlButtongroups("gr1",array("Bouton1","Bouton2","Bouton3"));
        $this->jquery->compile($this->view);
    }
    ...
}

```

```
{{q['gr1']}}
```

Bouton1 Bouton2 Bouton3

## -- Accès aux éléments

Le premier élément est à l'index 0, le second à l'index 1...

```
$gr1->getElement(1)->addLabel("New");
```

## -- Association d'événements

Contrôleur :

- la méthode **getDeferred** permet d'effectuer une requête ajax, différée sur le click du bouton à la position 0, et affichée ensuite dans la zone **#panel**

```

class IndexController extends Phalcon\Mvc\Controller{
    public function buttonGroupsAction(){
        $bootstrap=$this->jquery->bootstrap();
        $gr1=$bootstrap->htmlButtongroups("gr1",array("Bouton1","Bouton2","Bouton3"));
        $elm1=$gr1->getElement(0)->addLabel("New");
        $elm1->onClick($this->jquery->getDeferred("index/get","#panel"));
        $this->jquery->compile($this->view);
    }

    public function getAction($id){
        echo "Réponse obtenue Sur click du bouton <b>#".$id."</b>";
    }
}

```

```

    $this->view->disable();
}
...

```

Vue :

- **q['gr1']** permet d'afficher le contrôle
- **script\_foot** insère le script JQuery correspondant à la dynamique

```

{{q['gr1']}}
<div id='panel' class='alert alert-info'></div>

{{script_foot}}

```

## -- Association d'événements au groupe

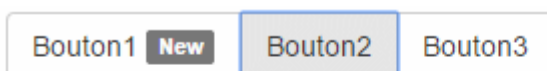
Chaque bouton du groupe est cette fois associé au même événement sur click. La requête ajax effectuée récupère comme paramètre l'id de l'élément HTML cliqué :

```

class IndexController extends Phalcon\Mvc\Controller{
    public function buttonGroupsAction(){
        $bootstrap=$this->jquery->bootstrap();
        $gr1=$bootstrap->htmlButtongroups("gr1",array("Bouton1","Bouton2","Bouton3"));
        $elm1=$gr1->getElement(0)->addLabel("New");
        $gr1->onClick($this->jquery->getDeferred("index/get","#panel"));
        $this->jquery->compile($this->view);
    }

    public function getAction($id){
        echo "Réponse obtenue Sur click du bouton <b>#".$id."</b>";
        $this->view->disable();
    }
}
...

```



Réponse obtenue Sur click du bouton #gr1-button-2

## -- Adaptation de la réponse à l'élément cliqué

- Chaque bouton du groupe est cette fois défini à partir d'un array, lui affectant les attributs **value** (texte du bouton), **title** (sera utilisé pour modifier la classe css de la cible sur mouseover) et **data-target** (cible du get ⇒ #panel1, #panel2 ou #panel3)
- Les appels jquery utilisent le paramètre **event** pour récupérer les valeurs de **title** ou de **data-target** des boutons ayant émis l'événement
- Sur mouseover du bouton **#1**, le panel **#panel1** prend la classe css stockée dans l'attribut **title** de bouton #1
- Sur click du bouton **#1**, le résultat du get est envoyé vers le panel **#panel1**

```

class IndexController extends Phalcon\Mvc\Controller{
    public function buttonGroupsAction(){
        $bootstrap=$this->jquery->bootstrap();
        $gr1=$bootstrap->htmlButtongroups("gr1",array(array("value"=>"Bouton1","title"=>"su
        ccess","data-target"=>1),
            array("value"=>"Bouton2","title"=>"warning","data-target"=>2),
            array("value"=>"Bouton3","title"=>"danger","data-target"=>3)));
        $gr1->getElement(0)->addLabel("New");
        $gr1->onClick($this->jquery->getDeferred("testsbs/get","'#panel'+$(event.target).at
        tr('data-target')"));
        $gr1->on("mouseover",$this->jquery->addClass("#panel'+$(event.target).attr('data-
        target')","'alert-'+event.target.title"));
        $this->jquery->compile($this->view);
    }

    public function getAction($id){
        echo "Réponse obtenue Sur click du bouton <b>#".$id."</b>";
        $this->view->disable();
    }
}
...

```

```

{{q['gr1']}}
<div id='panell1' class='alert'>--</div>
<div id='panel2' class='alert'>--</div>
<div id='panel3' class='alert'>--</div>
{{script_foot}}

```



### -- Ajout d'éléments supplémentaire

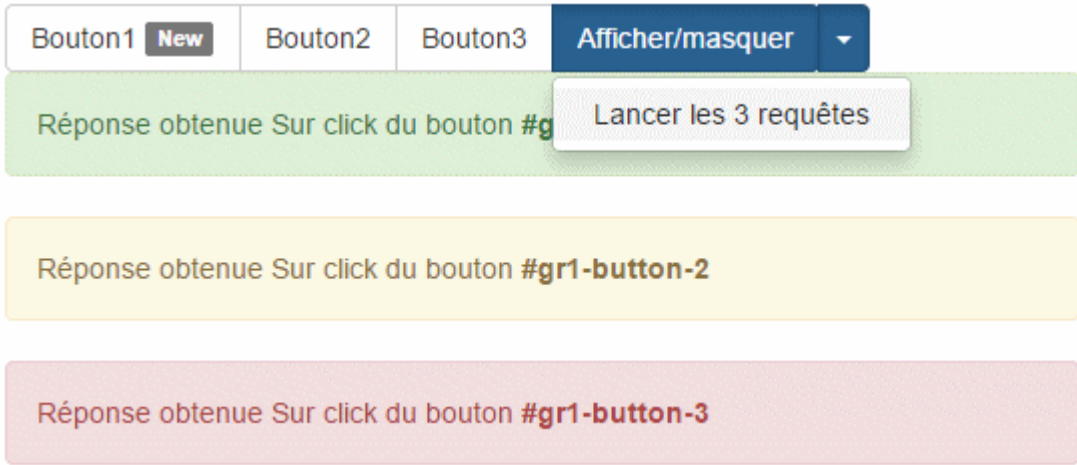
On ajoute un SplitButton au Buttongroups :

- Le click sur le bouton doit permettre d'afficher/masquer (méthode **toggle**) les 3 panels
- Le click sur l'item présent dans le menu du splitButton doit provoquer l'événement clic (méthode **trigger**) sur les 3 premiers boutons du groupe (donc les 3 requêtes) :

```

...
    $split=$gr1->addElement(new
HtmlSplitbutton("split","Afficher/masquer",array("Lancer les 3 requêtes"),"btn-
primary"));
    $split->getItem(0)->onClick($this->jquery->trigger(".btn-
default","click"));
    $split->onButtonClick($this->jquery->toggle(".alert"));
...

```



### -- Button toolbar

<b>Classe</b>	<a href="#">HtmlButtontoolbar</a>
<b>⇒ Hérite de</b>	<a href="#">HtmlButtongroups</a>



Le composant **Buttontoolbar** permet d'afficher plusieurs **Buttongroups**.

### -- Création dans le contrôleur et passage à la vue

L'ajout de boutons crée par défaut un nouveau **buttongroups** dans le composant :

```

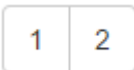
class IndexController extends Phalcon\Mvc\Controller{
    public function buttonToolbarAction(){
        $bootstrap=$this->jquery->bootstrap();
        $toolbar=$bootstrap->htmlButtontoolbar("toolbar");
        $toolbar->addElements(array("1","2"));
        $this->jquery->compile($this->view);
    }
...

```

```

{{q['toolbar']}}

```



## -- Ajout d'éléments

Ajout de boutons dans un nouveau Buttongroups :

```
$toolbar->addElement(new HtmlButtongroups("gr2",array("3","4","5"),"btn-success"));
```

D'une autre façon :

```
$toolbar->addGroup()->addElement(array("6",array("glyph"=>"star")));
```

## -- accès aux éléments

Accès aux **buttonsgroup** :

```
$buttonGroup1=$toolbar->getElement(0);
```

Accès aux **buttonsgroup** :

```
$button1=$toolbar->getElement(0)->getElement(0);
```

## -- Association d'événements

Sur click de tous les boutons de la toolbar :

```
$toolbar->onClick("console.log('click général')");
```

Sur click de tous les boutons d'un buttonsgroup :

```
$toolbar->getElement(0)->onClick("console.log('click buttonsgroup')");
```

Sur click de l'un des boutons d'un buttonsgroup :

```
$toolbar->getElement(0)->getElement(0)->onClick("console.log('click button')");
```

## -- Navbar

Classe	<a href="#">HtmlNavbar</a>
⇒ Hérite de	<a href="#">BaseHtml</a>

Phalcon-JQuery   Link 1   Link 2   Dropdown ▾         Link   Dropdown ▾

## -- Exemple

### Contrôleur

```
class IndexController extends Phalcon\Mvc\Controller{
    public function navbarAction(){
        $bootstrap=$this->jquery->bootstrap();
        $nav=$bootstrap->htmlNavbar("navbar-1", "Phalcon-
jQuery", "https://github.com/jcheron/phalcon-jquery");
        $this->jquery->compile($this->view);
    }
    ...
}
```

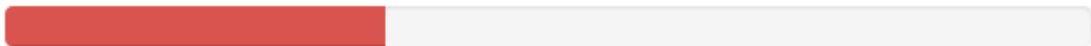
### Affichage dans la vue

```
{{q["navbar-1"]}}
{{script_foot}}
```

### Ajout d'éléments

```
//Liens avec href à #
    $nav->addElements(array("Link 1", "Link 2"));
//Liens avec href renseigné
    $nav->addElement(array("http://www.google.com", "Google"));
//Plusieurs liens
$nav->addElements(array(array("http://www.google.com", "Google"), array("http://www.m
icrosoft.com", "Microsoft")));
//Dropdown
    $nav->addElements(array("Dropdown"=>array("Action", "Another action", "SomeThing
else here", "-", "Separated link")));
```

## -- Progressbar



## -- Création

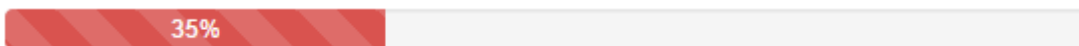
### Contrôleur

```
...
public function progressBarAction(){
    $pb=$this->jquery->bootstrap()->htmlProgressbar("pb","danger",35);
    $this->jquery->compile($this->view);
}
...
```

### Vue associée

```
{{q["pb"]}}
```

## -- Modification des attributs (stripped, active, caption...)



```
...
public function progressBarAction(){
    $pb=$this->jquery->bootstrap()->htmlProgressbar("pb","danger",35);
    $this->jquery->compile($this->view);
    $pb->setStriped(true);
    $pb->setActive(true);
    $pb->showCaption(true);
}
...
```

## -- Stacked progressbars



```
...
public function progressBarAction(){
    $pb=$this->jquery->bootstrap()->htmlProgressbar("pb");
    $pb->showCaption(true);
    $pb->setActive(true);
    $pb->setStriped(true);
    $pb->stack(new HtmlProgressbar("pb-1","info",10));
    $pb->stack(new HtmlProgressbar("pb-2","success",20));
    $pb->stack(new HtmlProgressbar("pb-2","warning",15));
    $pb->stack(new HtmlProgressbar("pb-2","danger",35));
}
```

```
$this->jquery->compile($this->view);  
}  
...
```

## -- Panel

### -- Création

#### Contrôleur

```
public function panelAction(){  
    $pan=$this->jquery->bootstrap()->htmlPanel("pan1");  
    $pan->setContent("Lorem ipsum...");  
    $pan->addContent(new HtmlAlert("alert"));  
    $pan->setStyle(CssRef::CSS_WARNING);  
    $pan->addHeaderH("Titre","3");  
    $pan->addFooter("Panel exemple...");  
    $pan->setCollapsable(true);  
    $pan->show(true);  
    $this->jquery->compile($this->view);  
}
```

#### Vue

```
{{q["pan1"]}}  
{{script_foot}}
```

## -- Tabs

### -- Création

#### Contrôleur

```
public function tabsAction(){  
    $tabs1=new HtmlTabs("tabs1");  
    $tabs1->addTab(array("content"=>"Elément 1","href"=>"#home"));  
    $tabs1->addTabs(array(array("content"=>"Elément  
2","href"=>"#elem2"),array("content"=>"Elément 3","href"=>"#elem3")));  
    $tabs1->setTabstype("pills");  
    $tabs1->run($this->jquery);  
  
    $bt5=new HtmlSplitbutton("id5");  
    $bt5->setValue("Fichier");  
    $bt5->setTagName("button");
```

```

    $bt5->setBtnClass("btn btn-primary dropdown-toggle");
$bt5->setItems(array(array("caption"=>"Joueurs", "href"=>"#elem3"), array("caption"=>
"Accordion", "href"=>"#elem3")));
    $bt5->setRole("nav");
    $bt5->run($this->jquery);

    $tabs1->addTab($bt5);
    echo $tabs1->compile();
    $tabs1->addTabContents();
    $tabs1->setContentToTab(0, "aa");
    $tabs1->setContentToTab(1, "bb");
    $tabs1->fadeEffect();
    echo $tabs1->getContent();
    echo $tabs1->getBsComponent()->show(0);
    $tabs1->getBsComponent()->onShow(1,
$this->jquery->get("exemple/form/1", "#elem2"));
    $tabs1->getBsComponent()->onShow(2,
$this->jquery->get("exemple/index/nom", "#elem3"));
    echo $this->jquery->compile();
    $this->view->disableLevel(View::LEVEL_ACTION_VIEW);
}

```

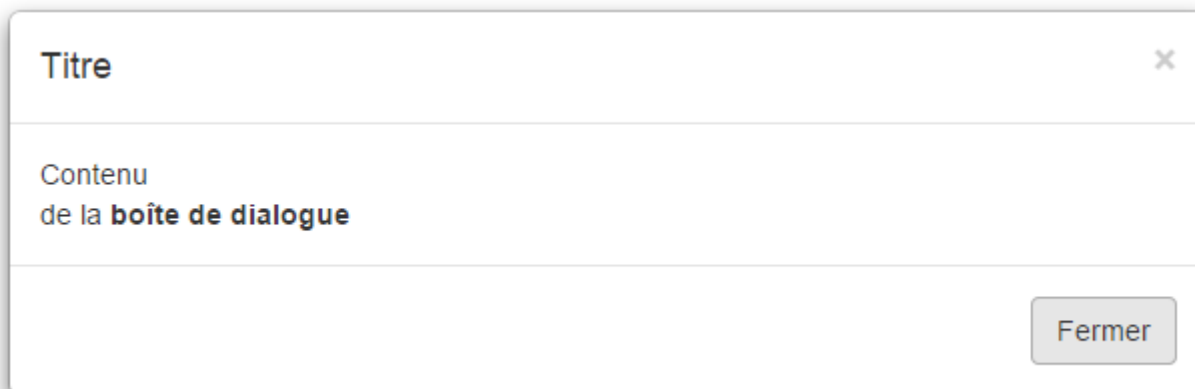
**Vue**

```

{{q["tabs1"]}}
{{script_foot}}

```

**-- Modals**



**-- Création**

Affichage d'une boîte modale sur clic d'un bouton :

**Contrôleur**

```

public function modalAction(){
    $bs=$this->jquery->bootstrap();
    $btn=$bs->htmlButton("btn1","Afficher la boîte modale");
    $modal=$bs->htmlModal("modal1","Titre","Contenu<br>de la <strong>boîte de
dialogue</strong>");
    $modal->addCancelButton("Fermer");
    $btn->onClick($modal->jsShow());
    $this->jquery->compile($this->view);
}

```

## Vue

```

{{q["btn1"]}}
{{q["modal1"]}}
{{script_foot}}

```

## -- Initialisation du contenu (content)

### -- En PHP (initié côté serveur)

```

public function modalAction(){
    $bs=$this->jquery->bootstrap();
    $btn=$bs->htmlButton("btn1","Afficher la boîte modale");
    $modal=$bs->htmlModal("modal1","Titre");
    $modal->renderContent($this->view, "exemple",
"modalForm",array("mail"=>"admin@local.net","password"=>"xxxx"));
    $modal->addCancelButton("Fermer");
    $btn->onClick($modal->jsShow());
    $this->jquery->compile($this->view);
}

```

La méthode **renderContent** charge une vue, mais ne change pas le contrôleur actif.

La vue correspondante :

```

<form>
  <div class="form-group">
    <label for="exampleInputEmail1">Email address</label>
    <input type="email" class="form-control" id="exampleInputEmail1"
placeholder="Enter email" value="{{mail}}">
  </div>
  <div class="form-group">
    <label for="exampleInputPassword1">Password</label>
    <input type="password" class="form-control" id="exampleInputPassword1"
placeholder="Password" value="{{password}}">
  </div>
</form>

```

Boîte chargée en PHP
✕

---

**Email address**

**Password**

-- En ajax (initié côté client)

Ajout d'un bouton **Suivant** :

- Chargeant en ajax l'url **exemple/modalForm2** (controller/action) sur le click du bouton **Suivant**
- Modifiant le titre du dialog
- Masquant le bouton suivant

```

public function modalAction(){
    $bs=$this->jquery->bootstrap();
    $btn=$bs->htmlButton("btn1","Afficher la boîte modale");
    $modal=$bs->htmlModal("modall","Titre");
    $modal->renderContent($this->view, "exemple",
"modalForm",array("mail"=>"admin@local.net","password"=>"xxxx"));
    $modal->addCancelButton("Fermer");
    $bt=$modal->addButton("Suivant
>>","success")->onClick($modal->jsGetContent($this->jquery, "exemple/modalForm2"));
    $bt->onClick($modal->jsSetTitle("Contenu chargé en Ajax"));
    $bt->onClick($modal->jsHideButton("Suivant"));
    $btn->onClick($modal->jsShow());
    $this->jquery->compile($this->view);
}

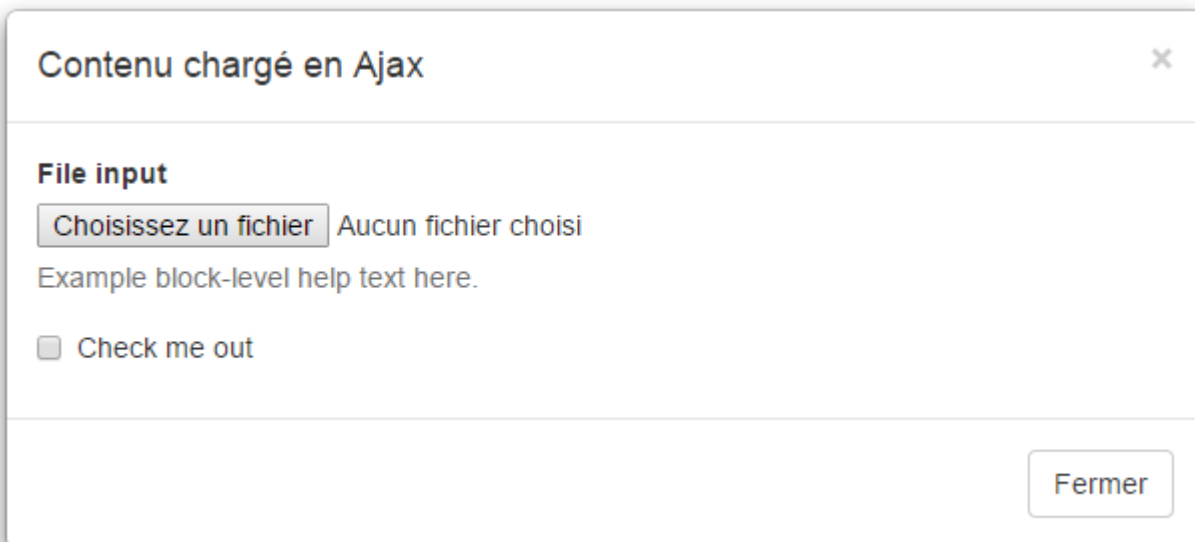
```

Le contrôleur correspondant à la vue sollicitée :

```
public function modalForm2Action(){  
    $this->view->disableLevel(View::LEVEL_MAIN_LAYOUT);  
}
```

La vue correspondante :

```
<form>  
  <div class="form-group">  
    <label for="exampleInputFile">File input</label>  
    <input type="file" id="exampleInputFile">  
    <p class="help-block">Example block-level help text here.</p>  
  </div>  
  <div class="checkbox">  
    <label>  
      <input type="checkbox"> Check me out  
    </label>  
  </div>  
</form>
```



## -- Boutons des boîtes modales

### -- Création

```
public function modalAction(){  
    $bs=$this->jquery->bootstrap();  
    //Ajout de 2 boutons à l'instanciation  
    $modal=$bs->htmlModal("modal1","Titre du  
dialogue", "Contenu",array("Ajouter","Retirer"));  
    //Ajout d'un bouton Annuler  
    $modal->addCancelButton("Fermer");  
    //Ajout d'un bouton Okay  
    $modal->addOkayButton("Valider");
```

```

//Ajout d'un bouton classique
$bt=$modal->addButton("Suivant
>>", "success")->onClick($modal->jsGetContent($this->jquery, "exemple/modalForm2"));
    $this->jquery->compile($this->view);
}

```

### Boutons spécifiques :

- Le bouton **Annuler** permet la fermeture du dialogue
- Le bouton **Okay** ne ferme le dialogue que si la condition javascript **validCondition** est vraie

La condition par défaut est **\$('#identifiant').prop('valid')**, il est possible de la changer à tout moment :

**Exemple :** Vérification de la présence d'un élément d'id #ck dans la page :

```
$modal->setValidCondition("$('#ck').length>0");
```

**Exemple :** en rendant la sortie automatiquement valide :

```
$modal->setValid();
```

### -- Accès aux boutons

Il est possible d'accéder aux boutons déjà créés soit par leur **identifiant** (attribut id du HTML), soit par leur **index** (à partir de 0 dans le tableau des boutons) :

```

public function modalAction(){
    $bs=$this->jquery->bootstrap();
    //Ajout de 2 boutons à l'instanciation
    $modal=$bs->htmlModal("modal1", "Titre du
dialogue", array("Ajouter", "Retirer"));
    //Accès au bouton par son nom (identifiant = nom corrigé)
    $modal->getButton("Ajouter")->setStyle("Success");
    //Accès au bouton par son index
    $modal->getButton(1)->setStyle("Warning");

    $this->jquery->compile($this->view);
}

```




From:

<http://slamwiki2.kobject.net/> - **SlamWiki 2.1**

Permanent link:

<http://slamwiki2.kobject.net/slam4/php/phalcon/jquery/bootstrap>

Last update: **2019/08/31 14:21**

