

Phalcon-jquery : JQuery

Principales fonctionnalités de la librairie Phalcon-JQuery (hors JQuery UI et Twitter Bootstrap)

Pour Bootsrap, voir [Bootstrap](#)

-- Association d'évènements

-- Événements connus

-- Exemple

Click sur un bouton d'id **#btn** :

```
public function clickAction(){
    $this->click("#btn","console.log('click sur #btn');");
    $this->query->compile($this->view);
}
```

Vue associée :

```
<input type="button" value="Test click" id="btn">
{{script_foot}}
```

-- Utilisation du paramètre event du callback

Le paramètre **callback** ne doit pas passer une fonction javascript, mais une suite d'instructions js.

Le code passé au callback peut utiliser le paramètre **event** passé par JQuery ou **\$(this)**, cible de l'évènement (target) :

Exemple : appui sur une touche, récupération du code de la touche frappée et de l'id de l'élément ayant généré l'évènement.

```
public function clickAction(){
    $this->jquery->keydown("#btn","console.log('code de touche appuyée : ' +
event.which + ' sur #' + $(this).attr('id'));");
    $this->jquery->compile($this->view);
}
```

Exemple : affichage/masquage d'un élément sur changement de la valeur d'une checkbox :

```

    public function checkedAction(){
$this->jquery->change("#ck",$this->jquery->toggle("#fs","$(this).checked()"));
    $this->jquery->compile($this->view);
}

```

Vue associée :

```

<input type="checkbox" id="ck" checked>
<fieldset id="fs">
<legend>Elément à afficher/masquer</legend>
</fieldset>
{{script_foot}}

```

-- Liste des méthodes JQuery liées aux évènements

Méthode	Evènement
blur(selector,callback)	Sur perte du focus par l'élément désigné par selector
change(selector,callback)	Sur changement de valeur d'un input , select ou textarea (déclenché généralement sur perte focus)
click(selector,callback)	Sur click de l'élément désigné par selector
dblclick(selector,callback)	Sur double-click de l'élément désigné par selector
error(selector,callback)	Sur erreur de chargement sur selector
focus(selector,callback)	Sur réception du focus par l'élément désigné par selector
hover(selector,callback)	Sur déplacement de la souris au dessus de l'élément désigné par selector
keydown(selector,callback)	Sur touche clavier enfoncée lorsque l'élément désigné par selector a le focus
keypress(selector,callback)	Sur touche clavier enfoncée lorsque l'élément désigné par selector a le focus
keyup(selector,callback)	Sur touche clavier relachée lorsque l'élément désigné par selector a le focus
load(selector,callback)	Sur chargement de l'élément désigné par selector
mousedown(selector,callback)	Sur bouton souris enfoncé lorsque l'élément désigné par selector
mouseout(selector,callback)	Sur sortie de la souris de l"élément désigné par selector
mouseover(selector,callback)	Sur passage souris au dessus de l'élément désigné par selector
mouseup(selector,callback)	Sur bouton souris relaché lorsque l'élément désigné par selector
ready(callback)	Sur chargement du document terminé
scroll(selector,callback)	Sur défilement dans/sur l'élément désigné par selector
trigger(selector,event)	Déclenche l'évènement event sur l'élément désigné par selector
unload(selector,callback)	Sur déchargement de l'élément désigné par selector

-- Evènements non listés

Exécuter du Javascript

Pour les évènements inconnus de JQuery ou ne disposant pas de méthode associée dans la librairie phalcon-jquery, on utilise les méthodes :

- execOn(event,selector,callback) permettant d'exécuter le **callback** sur le déclenchement de **event** sur l'élément désigné par **selector**.

Exemple

Sur **dropdown (event bootstrap)** du bouton, un message est affiché dans les logs de la console

```

public function dropdownAction(){
    ...
$this->jquery->execOn("show.bs.dropdown", "#btnDropdown", "console.log('Dropdown is visible');");
    ...
}

```

Exécuter du JQuery

- dojQueryOn(\$event,\$element,\$elementToModify,\$jqueryCall,\$param="", \$function="") permettant d'exécuter la méthode jquery **jqueryCall** sur **elementToModify**, en lui passant **param** sur le déclenchement de **event** sur l'élément désigné par **element**.

Exemple

Sur **dropdown (event bootstrap)** du bouton, un message est affiché dans l'élément **#divMessage**

```

public function dropdownAction(){
    ...
$this->jquery->dojQueryOn("show.bs.dropdown", "#btnDropdown", "#divMessage", "html", "Dropdown is visible");
    ...
}

```

-- Requêtes Ajax

-- Requêtes immédiates

Il s'agit des requêtes dont l'exécution n'est pas différée (non associée à un évènement par exemple). Les requêtes immédiates peuvent permettre le chargement d'une page en cascade : élément par élément.

--GET

Exemple : Chargement de **exemple/getResponse** dans la div **#divResponse**

```

public function getAction(){
    $this->jquery->get("exemple/getResponse", "#divResponse");
    echo "<div id='divResponse'>Chargement...</div>";
    echo $this->jquery->compile();
}
public function getResponseAction(){
    echo "Chargement terminé";
    $this->view->disable();
}

```

-- POST

Exemple : Post de valeurs vers **exemple/postResponse** dans la div **#divPostResponse**

```
public function postAction(){
    $this->jquery->post("exemple/postResponse","#divPostResponse","{nom:'Doe',prenom:'John'}");
        echo "<div id='divPostResponse'>Changement...</div>";
        echo $this->jquery->compile();
}
public function postResponseAction(){
    var_dump($_POST);
    $this->view->disable();
}
```

-- JSON

Les requêtes JSON permettent d'alimenter en données une vue, à partir d'un retour d'objet JSON par le serveur.

Exemple : Le get JSON vers la page **exemple/jsonResponse** alimente tous les éléments DOM **nom** et **prenom** avec l'élément JSON retourné.

```
public function jsonAction(){
    $this->jquery->json("exemple/jsonResponse");
    echo "<div id='nom'>Changement...</div>";
    echo "<input type='text' id='prenom' value='chargement...'">";
    echo $this->jquery->compile();
}
public function jsonResponseAction(){
    echo '{"nom":"Doe","prenom":"John"}';
    $this->view->disable();
}
```

-- JSON Array

A partir de la version : [commit du 23-03-2015](#)

Il peut également être intéressant de réceptionner des données JSON sous forme de tableau, et d'afficher ces données dans une vue, en utilisant un mask (présent dans la page) :

```
public function jsonArrayAction(){
    $this->jquery->jsonArray("#mask","exemple/jsonArrayResponse");
    echo $this->jquery->compile();
}
```

La vue contenant le mask :

Les éléments à remplacer dans le mask doivent comporter l'attribut **data-id**, dont le contenu doit correspondre

aux données du tableau JSON.

```
<div id='mask' style='display: none'><button data-id='nom' class='btn btn-primary'>Chargement...</button>&nbsp;
<input type='text' data-id='prenom' value='chargement...' class='form-control'></div>
```

Une réponse JSON pour tester :

```
public function jsonArrayResponseAction(){
    echo
' [{"nom":"DOE","prenom":"John"}, {"nom":"SMITH","prenom":"Robert"}, {"nom":"GATES","prenom":"Bill"}]';
    $this->view->disable();
}
```

Le résultat :

The screenshot shows a user interface with a dropdown menu and three corresponding input fields. The dropdown menu contains three items: 'DOE', 'SMITH', and 'GATES'. Below each item is a text input field containing the names 'John', 'Robert', and 'Bill' respectively. The 'DOE' item is highlighted with a blue background.

Le même exemple, en mettant le chargement JSON sur le click d'un bouton :

on utilise **jsonArrayDeferred**, puisque la requête doit attendre le click du bouton

```
public function jsonArray2Action(){
    $btn=$this->jquery->bootstrap()->htmlButton("bt1","Get
JsonArray",CssRef::CSS_PRIMARY,$this->jquery->jsonArrayDeferred("#mask", "exemple/js
onArrayResponse"));
    echo $btn->compile($this->jquery);
    echo "<div id='mask' style='display:none'><button data-id='nom' class='btn
btn-primary'>Chargement...</button>&nbsp;";
    echo "<input type='text' data-id='prenom' value='chargement...'
class='form-control'></div>";
    echo $this->jquery->compile();
}
```

Même exemple, mais avec séparation contrôleur/vue :

Le contrôleur :

```
public function jsonArray3Action(){
    $btn=$this->jquery->bootstrap()->htmlButton("bt1","Get
JsonArray",CssRef::CSS_PRIMARY);
$btn->onClick($this->jquery->jsonArrayDeferred("#my","exemple/jsonArrayResponse"));
    $this->jquery->compile($this->view);
}
```

La vue associée :

```
{{q["bt1"]}}
<fieldset>
<legend>Résultat du JsonArray</legend>
<div id='my' style='display:none'><button data-id='nom' class='btn btn-primary'>Chargement...</button>&nbsp;
<input type='text' data-id='prenom' value='chargement...' class='form-control'></div>
</fieldset>
{{script_foot}}
```

Exemple avec affectation plus complexe

On souhaite maintenant que le retour JSON puisse modifier de multiples valeurs présentes dans le mask html :

Soit le masque suivant, défini dans la vue :

Nous souhaitons modifier les valeurs des progressbars Bootstrap avec les valeurs **pb** et **style** de notre JSON : il faut dans ce cas préciser inclure dans le masque les membres **[[pb]]** et **[[style]]** aux endroits souhaités.

```
{{q["bt1"]}}
{{q["bt1"]}}
<fieldset>
<legend>Résultat du JsonArray</legend>
<div id='my' style='display:none'><button data-id='nom' class='btn btn-primary'>Chargement...</button>&nbsp;
<input type='text' data-id='prenom' value='chargement...' class='form-control'>
<div class="progress">
    <div class="progress-bar progress-bar[[style]]" role="progressbar" aria-
valuenow="[[pb]]" aria-valuemin="0" aria-valuemax="100" style="width: [[pb]]%;">
        [[pb]]%
    </div>
</div>
</div>
</fieldset>
{{script_foot}}
```

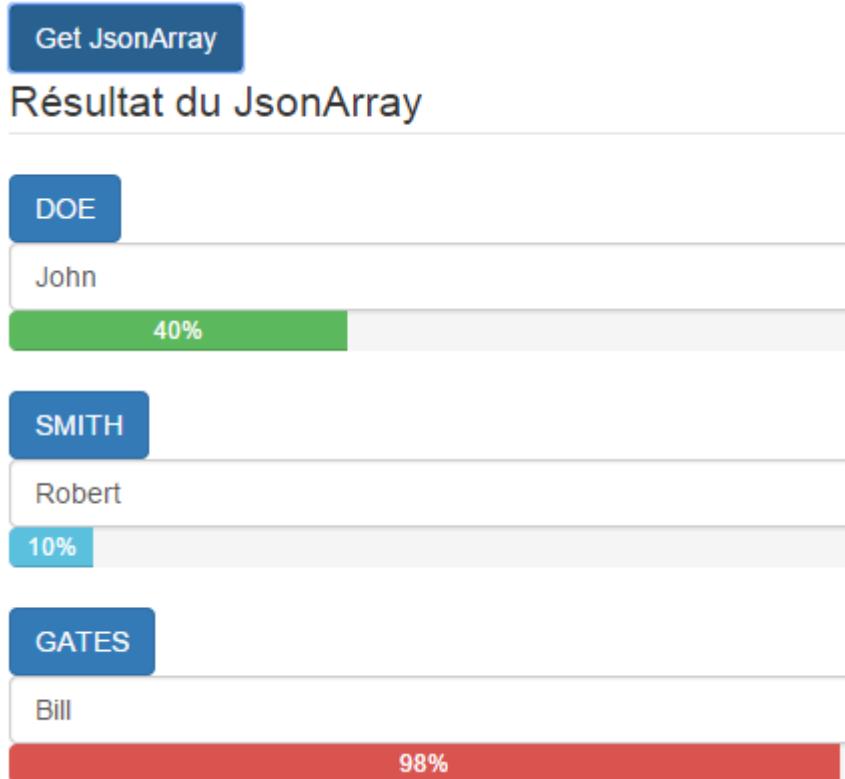
Retour JSON :

```
public function jsonArrayResponseAction(){
    echo '[{"nom":"DOE","prenom":"John","pb":40,"style":"success"}, {"nom":"SMITH","prenom":"Robert","pb":10,"style":"info"}, {"nom":"GATES","prenom":"Bill","pb":98,"style":"danger"}]';
    $this->view->disable();
}
```

Le Contrôleur ne change pas :

```
public function jsonArray2Action(){
    $btn=$this->jquery->bootstrap()->htmlButton("bt1","Get
JsonArray",CssRef::CSS_PRIMARY);
$btn->onClick($this->jquery->jsonArrayDeferred("#my","exemple/jsonArrayResponse"));
    $this->jquery->compile($this->view);
}
```

Résultat



-- Association de requêtes à des évènements

Pour faire plus court, les exemples suivants ne comportent pas de vues, et les éléments HTML sont directement produits dans le contrôleur (à éviter...)

-- getOn

Permet d'associer l'exécution d'un Get à un évènement déclenché sur un élément :

Exemple : Sur click du bouton **#btnGet**, on veut effectuer un get vers **exemple/getResponse**, et afficher le résultat dans la div **#divResponse**.

```
public function getAction(){
    $this->jquery->getOn("click", "#btn", "testsbs/getResponse", "#divResponse");
    echo "<input type='button' id='btn' value='doGet'>";
    echo "<div id='divResponse'>En attente de click...</div>";
    echo $this->jquery->compile();
}
public function getResponseAction(){
    echo "Chargement terminé";
    $this->view->disable();
}
```

-- postFormOn

Permet d'associer le post d'un formulaire à un évènement déclenché sur un élément :

Exemple : Sur click du bouton **#btnPost**, on veut poster le formulaire **frmLogin** vers **exemple/postResponse**, et afficher le résultat dans la div **#divResponse**.

```
public function postResponseAction(){
    var_dump($_POST);
    $this->view->disable();
}

public function postFormAction(){
$this->jquery->postForOn("click", "#btn", "testsbs/postResponse", "frmLogin", "#divPostResponse");
    echo "<form id='frmLogin' name='frmLogin' onsubmit='return false;'><input type='text' name='login' placeholder='Login...'>
        <input type='password' name='password' placeholder='password...'/></form>";
    echo "<input type='button' id='btnPost' value='doPost'>";
    echo "<div id='divPostResponse'>Avant validation</div>";
    echo $this->jquery->compile();
}
```

Il est également possible d'utiliser :

Méthode	Rôle
postOn(event,element,url[,params,responseElement])	Post les paramètres params vers url sur l' event produit sur element

Ou de combiner les méthodes différées avec les méthodes évènementielles :

Requête Ajax Get sur click d'un bouton :

```

    public function getAction(){
$this->jquery->click("#btn",$this->jquery->getDeferred("testsbs/getResponse","#divResponse"));
    echo "<input type='button' id='btn' value='doGet'>";
    echo "<div id='divResponse'>En attente de click...</div>";
    echo $this->jquery->compile();
}

```

-- Manipulation du DOM

Méthode	Rôle
append(to,element)	ajoute l'élément DOM element à la fin de chaque élément désigné par to
prepend(to,element)	ajoute l'élément DOM element au début de chaque élément désigné par to
attr(selector,attributeName [,value,immediatly])	Lit/modifie l'attribut attributeName des éléments correspondant à selector
html(selector[,value,immediatly])	Lit le contenu html / Affecte value au contenu html des éléments correspondant à selector

-- Utilitaires

Méthode	Rôle
show(selector[,speed,callback])	Affiche le ou les éléments DOM correspondant au selecteur selector
hide(selector[,speed,callback])	Masque le ou les éléments DOM correspondant au selecteur selector
fadeIn(selector[,speed,callback])	Effectue une transition sur le ou les éléments DOM correspondant au selecteur selector pour le(s) rendre opaque
fadeOut(selector[,speed,callback])	Effectue une transition sur le ou les éléments DOM correspondant au selecteur selector pour le(s) rendre transparent
toggle(selector)	Affiche/masque le ou les éléments DOM correspondant au selecteur selector
animate(selector[,params,speed,extra])	Effectue une animation sur le ou les éléments DOM correspondant au selecteur selector
addClass(selector,class[,immediatly])	Ajoute la classe class sur les éléments DOM correspondant au selecteur selector
toggleClass(selector,class[,immediatly])	Ajoute/supprime la classe class sur le ou les éléments DOM correspondant au selecteur selector

From:

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

Permanent link:

<http://slamwiki2.kobject.net/slamp4/php/phalcon/jquery/usage>

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

