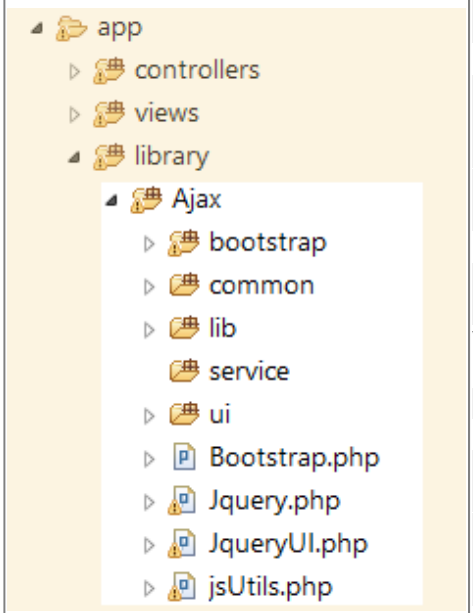
Phalcon-JQuery



Phalcon-JQuery est une librairie PHP compatible avec le Framework Phalcon. Elle permet d'intégrer facilement JQuery, JQuery UI ou Twitter Bootstrap dans les projets Web Phalcon, en respectant la séparation des couches MVC.

-- Installation

1. Télécharger Phalcon-JQuery sur gitHub
2. Dé-zipper dans un dossier local, et copier le dossier Ajax dans le dossier correspondant à ma variable **libraryDir** définie dans la configuration de votre projet Phalcon :




```
    'application' => array(
        'controllersDir' => __DIR__ .
'/../../app/controllers/',
        'modelsDir'      => __DIR__ .
'/../../app/models/',
        'viewsDir'       => __DIR__ .
'/../../app/views/',
        'pluginsDir'     => __DIR__ .
'/../../app/plugins/',
        'libraryDir'     => __DIR__ .
'/../../app/library/',
        'cacheDir'       => __DIR__ .
'/../../app/cache/',
        'baseUri'        => '/siteBaseURL/',
    );
```

Vérifier également la présence de **libraryDir** dans le fichier **loader.php** :

```
$loader = new \Phalcon\Loader();

/**
 * We're a registering a set of directories taken from
the configuration file
 */
$loader->registerDirs(
    array(
        $config->application->controllersDir,
        $config->application->modelsDir,
        $config->application->libraryDir
    )
)->register();
```

--Intégration à l'iDE

Dézipper l'archive 1.3.4.zip dans le dossier référencé par le **Path variable PHP** défini pour Phalcon dans votre IDE. Le dossier 1.3.4 contenu dans l'archive doit remplacer l'ancien **pahlcon-devtools/ide/1.3.4**

Vérifiez que la complétion fonctionne sur la variable **$this→jquery** dans un contrôleur phalcon.

-- Configuration

-- Injection du service

Pour ajouter Phalcon-JQuery à un projet Phalcon, il est nécessaire d'injecter le service JQuery au démarrage de l'application :

```
$di->set("jquery",function(){
    $jquery= new Ajax\JsUtils(array("driver"=>"Jquery"));
    $jquery->ui(new Ajax\JqueryUI());//optional for Phalcon JQuery UI
    $jquery->bootstrap(new Ajax\Bootstrap());//Optional for Phalcon Twitter
Bootstrap
    return $jquery;
});
```

-- Intégration des librairies JS

Pour fonctionner correctement, les librairies JS suivantes doivent être insérées dans les pages HTML principales (non sollicitées via ajax).

Librairie activée	Fichier JS	Fichier css
JQuery	jquery-min.js	
JQuery UI*	jquery-ui-min.js	jquery-ui.css
Twitter Bootstrap*	bootstrap.min.js	bootstrap.min.css

* optionnel

-- Intégration manuelle des fichiers

Attention de veiller à inclure **JQuery** avant les librairies qui en dépendent : Jquery UI ou Bootstrap.

En utilisant des CDN

```
<html>
    <head>
    <?php
    // JQuery
    echo
```

```
Phalcon\Tag::javascriptInclude("http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/j
query.min.js", false);
    //jquery ui
    echo
Phalcon\Tag::javascriptInclude("https://ajax.googleapis.com/ajax/libs/jqueryui/1.11
.2/jquery-ui.min.js", false);
    //Thème Humanity for UI
    echo
Phalcon\Tag::stylesheetLink("https://ajax.googleapis.com/ajax/libs/jqueryui/1.11.2/
themes/humanity/jquery-ui.css", false);

?>
</head>
```

Même chose si vous utilisez **volt** :

```
<html>
    <head>
    <!--JQuery -->
{{javascript_include("http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.
js", false)}}
    <!--jquery ui -->
{{javascript_include("https://ajax.googleapis.com/ajax/libs/jqueryui/1.11.2/jquery-
ui.min.js", false)}}
    <!--Thème Humanity for UI -->
{{stylesheet_link("https://ajax.googleapis.com/ajax/libs/jqueryui/1.11.2/themes/hum
anity/jquery-ui.css", false)}}

</head>
```

En copiant les librairies JS et Css localement sur le serveur

Pour une installation locale :

- Copier les fichiers css dans **public/css**
- Copier les fichiers js dans **public/js**

```
<html>
    <head>
    <?php
    // JQuery
    echo Phalcon\Tag::javascriptInclude("js/jquery.min.js");
    //jquery ui
    echo Phalcon\Tag::javascriptInclude("js/jquery-ui.min.js");
    //Thème Humanity for UI
    echo Phalcon\Tag::stylesheetLink("css/jquery-ui.css");

?>
</head>
```

-- Intégration automatisée

Il est possible de récupérer directement via les classes CDN* les adresses des librairies et de générer automatiquement leur inclusion en suivant les étapes ci-dessous :

-- Paramétrage d'un contrôleur

Pour chaque action sollicitée par le contrôleur, on passe à la vue une variable **jqueryCDN** contenant les liens CDN à générer : Sans autre précisions, la méthode **genCDNs** génère l'insertion des liens en prenant les versions les plus récentes des librairies nécessaires : Si le service **jquery** injecté au démarrage de l'application déclare **JQuery UI**, les fichiers JS et CSS de UI seront insérés, de même pour **Bootstrap**.

```
class ExempleController extends ControllerBase{
    public function initialize(){
        $this->view->setVar("jqueryCDN", $this->jquery->genDCNs("humanity"));
    }
...
```

-- Insertion dans la/es vue/s

```
<html>
    <head>
    <?php
    echo $jqueryCDN;
    ?>
...
```

Résultat produit avec JQuery UI (Humanity template)+ Bootstrap activés :

```
<head>
...
    <script type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>
    <script type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/jqueryui/1.11.2/jquery-ui.min.js"></scri
pt>
    <link rel="stylesheet" type="text/css"
href="https://ajax.googleapis.com/ajax/libs/jqueryui/1.11.2/themes/humanity/jquery-
ui.css">
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.1/css/bootstrap.min.css">
    <script type="text/javascript"
src="http://maxcdn.bootstrapcdn.com/bootstrap/3.3.2/js/bootstrap.min.js"></script>
...
</head>
```

-- Utilisation

Le principe d'utilisation est toujours le même, que ce soit avec **JQuery**, **JQuery UI** ou **Twitter Bootstrap** :

- Les composants et leur logique applicative (association de code aux événements) sont créés dans le contrôleur
- Le contrôleur les transmet à la vue
- La vue affiche et/ou génère le script côté client

-- Exemple JQuery

-- Pré-requis

Il est nécessaire d'injecter le service JQuery au démarrage de l'application :

```
$di->set("jquery",function(){
    $jquery= new Ajax\JsUtils(array("driver"=>"Jquery"));
    return $jquery;
});
```

Il s'agit par exemple d'effectuer une requête Ajax vers la page **exemple/reponse** de notre site, sur le click du bouton d'id **btn**, et d'afficher la réponse dans la zone d'id **panelReponse** :

-- Logique applicative dans le contrôleur

```
class ExempleController extends \Phalcon\Mvc\Controller

    public function ajaxSampleAction(){
        $this->jquery->getAndBindTo("#btn", "click",
"exemple/reponse","#panelReponse");
        $this->jquery->compile($this->view);
    }
...
```

-- Affichage et Intégration du script généré dans la vue

La vue associée à notre action (**app/views/Exemple/ajaxSample.volt**) récupère la variable $script_foot, générée par l'appel de la méthode **jquery→compile** appelée dans le contrôleur :

```
<button id="btn">Lancer la requête</button>
<div id="panelReponse">Réponse non reçue</div>
{{script_foot}}
```

-- Création de la réponse

Pour l'instant, le click sur le bouton **btn** génère une réponse avec erreur 404, puisque l'action sollicitée n'existe pas : il faut donc créer l'action **reponse** dans le contrôleur **Exemple** :

```
class ExempleController extends \Phalcon\Mvc\Controller
...
    public function reponseAction(){
        echo "Réponse bien reçue";
        $this->view->disable();
    }
...
```

Pour aller plus loin : Phalcon-jquery : JQuery

-- Exemple JQuery UI

-- Pré-requis

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

```
$di->set("jquery",function(){
    $jquery= new Ajax\JsUtils(array("driver"=>"Jquery"));
    $jquery->ui(new Ajax\JqueryUI());//for JQuery UI
    return $jquery;
});
```

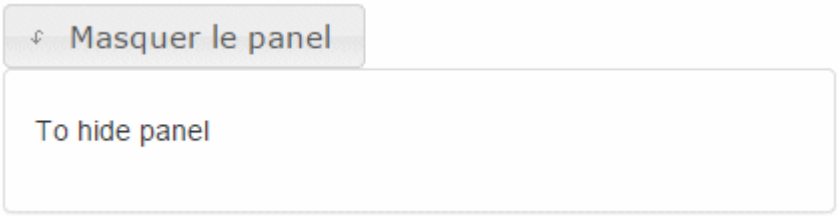
-- Création d'un bouton

Contrairement à Bootsrap, **JQuery UI** ne s'occupe pas de la partie DOM, le bouton doit donc déjà exister dans la page pour invoquer la méthode **button** :

- La méthode **button** permet d'attacher un **button JQuery UI** à un élément DOM, avec paramètres d'initialisation.
- La méthode on permet d'associer une action (masquage du panel) à un événement (click)

```
class ExempleController extends \Phalcon\Mvc\Controller{
...
    public function buttonsAction(){
        $bt1=$this->jquery->ui()->button("#btHide",array("label"=>"Masquer le
panel","icons"=>"{ primary: 'ui-icon-locked' }"));
        $bt1->on("click",$this->jquery->hide(".toHide",3000));
        echo "<button id='btHide'></button>";
        echo "<div class='toHide'><p>To hide panel</p></div>";
        echo $this->jquery->compile();
        $this->view->disableLevel(View::LEVEL_ACTION_VIEW);
```

```
    }
...
```



-- Exemple Twitter Bootstrap

-- Pré-requis

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

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

-- Création d'un bouton

Avec **Bootsrap**, le bouton est créé dans la page :

- La méthode **onClick** permet d'associer du code à l'événement **click**
- la méthode **compile** permet de générer le bouton et son comportement

```
class ExempleController extends \Phalcon\Mvc\Controller{
...
    public function buttonsbsAction(){
        $bt1=new HtmlButton("bt1");
        $bt1->setValue("Masquer le panel");
        $bt1->setStyle(CssRef::CSS_PRIMARY);
        $bt1->onClick($this->jquery->hide(".toHide",3000));
        echo $bt1->compile($this->jquery);
        echo "<div class='panel panel-default container toHide'><br><p>To hide
panel</p><br></div>";
        echo $this->jquery->compile();
        $this->view->disableLevel(View::LEVEL_ACTION_VIEW);
    }
...
```

Masquer le panel

To hide panel

Pour aller plus loin : Bootstrap

From:
http://slamwiki2.kobject.net/ - **SlamWiki 2.1**

Permanent link:
http://slamwiki2.kobject.net/slam4/php/phalcon/jquery

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