Astuces

<
Bibliothèques >>

1. Introduction à CodeIgniter
2. Helpers
3. Bibliothèques
4. Sessions CodeIgniter
5. Validation des formulaires
6. ORM IgnitedRecord
7. Doctrine
8. Javascript et codeIgniter

Contrôleurs

Création d'un contrôleur de base personnalisé par surdéfinition de CI_Controller

Le contrôleur de base de codeIgniter correspond à la classe **CI_Controller** localisée dans system/core.

Classe dérivée

```
class BaseCtrl extends \CI_Controller{
    public function __construct(){
    // TODO Auto-generated method stub
        parent::__construct();
    }
}
```

Modification de config.php

Il est nécessaire de modifier le fichier **config.php** pour que la classe **BaseCtrl** soit automatiquement chargée au lancement de l'application :

Ajouter le code suivant à la fin de config.php, en utilisant la fonction php **_autoload()**

```
//Fin de config.php
function __autoload($class){
    if(strpos($class, 'CI_') !== 0){
        $paths = array(APPPATH . 'core/', APPPATH . 'controllers/');
        foreach($paths as $path){
            if (file_exists($path . $class . EXT)){
                @include_once( $path . $class . EXT );
                break;
```

```
            }
        }
    }
}
```

Toutes les classes (ne commençant pas par le préfixe CI_) et présentes dans application/controller ou application/core seront automatiquement chargées.

Contrôleur personnalisé et complétion de code

L'inconvénient majeur de php est l'absence de typage des variables, combiné dans le cas présent au fonctionnement même de codeIgniter, les 2 réunis font qu'aucune complétion n'est présente par défaut sur les principaux objets accessibles depuis un contrôleur :

- les librairies chargées (doctrine, jsutils…)
- le loader load, l'objet ci…

Pour remédier à ce manque et avoir l'auto-complétion Eclipse sur ces objets utilisés fréquemment, il suffit de surcharger CI_Controller, de définir explicitement les membres de données existants, et de les commenter en style javadoc.
Eclipse utilisera les commentaires de type @var pour proposer l'auto-complétion sur les membres concernés.

```
<?php

class BaseCtrl extends \CI_Controller{

    /**
     * @var CI_JsUtils
     */
    protected $jsutils;

    /**
     * @var CI_Session
     */
    protected $session;
    /**
     * @var CI_Base
     */
    protected $CI;
    /**
     * @var Doctrine
     */
    private $doctrine;
    /**
     * @var CI_Loader
     */
    protected $load;
    public function __construct(){
    // TODO Auto-generated method stub
    parent::__construct();
    }
}
```

Contrôleur personnalisé et contrôle des accès

```
<?php
class BaseCtrl extends \CI_Controller{
...

    public function __construct(){
        parent::__construct();
        if(!$this->_isValid())
            $this->_onInvalidControl();
    }
    /**
     * retourne Vrai si l'accès au contrôleur est autorisé
     * @return boolean
     */
    protected function _isValid(){
        return true;
    }
    protected function _onInvalidControl(){
        header('HTTP/1.1 401 Unauthorized', true, 401);
        exit;
    }
}
```

Helpers

Helper avec quelques fonctions liées à la requête, à charger dans l'autoload ou directement depuis un contrôleur :

```
<?php  if ( ! defined('BASEPATH')) exit('No direct script access allowed');
/**
 * CodeIgniter
 *
 * An open source application development framework for PHP 5.1.6 or newer
 *
 * @package     CodeIgniter
 * @author      jcheron
 * @copyright   Copyright (c) 2008 - 2014, EllisLab, Inc.
 * @license     lgpl
 * @version 1.0
 * @filesource
 */

// ------------------------------------------------------------------------

/**
 * Request Helpers
 *
 * @package     CodeIgniter
 * @subpackage  Helpers
 * @category    Helpers
```

```
 * @author      jcheron
 * @link        http://slamwiki.kobject.net
 */

// ------------------------------------------------------------------------

    /**
     * setValuesToObject
     * Affecte membre à membre les valeurs du tableau associatif $values aux
membres de l'objet $object
     * Utilisé par exemple pour récupérer les variables postées et les affecter aux
membres d'un objet
     * @param Class $object
     * @param associative array $values
     */
    if(!function_exists("setValuesToObject")){
        function setValuesToObject($object,$values=null){
            if(!isset($values))
                $values=$_POST;
            foreach ($values as $key=>$value){
                $accessor="set".ucfirst($key);
                if(method_exists($object, $accessor)){
                    $object->$accessor($value);
                }
            }
        }
    }
    /**
     * getPost
     * Appel d'une fonction de nettoyage sur le post
     * @param string $function
     * @return multitype:
     */
    if(!function_exists("getPost")){
        function getPost($function="htmlentities"){
            return array_map($function, $_POST);
        }
    }
```

doctrine_helper.php