

Introduction à CodeIgniter

CodeIgniter est un framework php respectant MVC. Il est assez populaire, et relativement facile à prendre en main.

Il a été développé par l'entreprise EllisLab en 2006 pour ses propres besoins, puis mis à disposition du public par la suite.

- [Site officiel](#)
- [Téléchargement](#)
- [Documentation](#)

Téléchargement

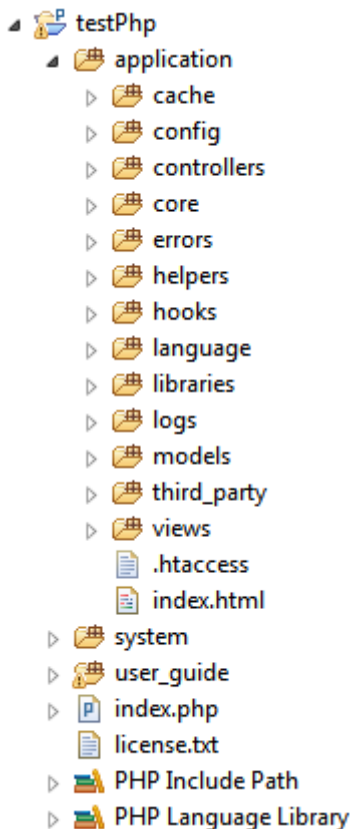
Télécharger la dernière version: [CodeIgniter 2.1.3](#)

Installation

Dans Eclipse, installer le plugin PDT (PHP Development Tool), pour développer plus confortablement en PHP.

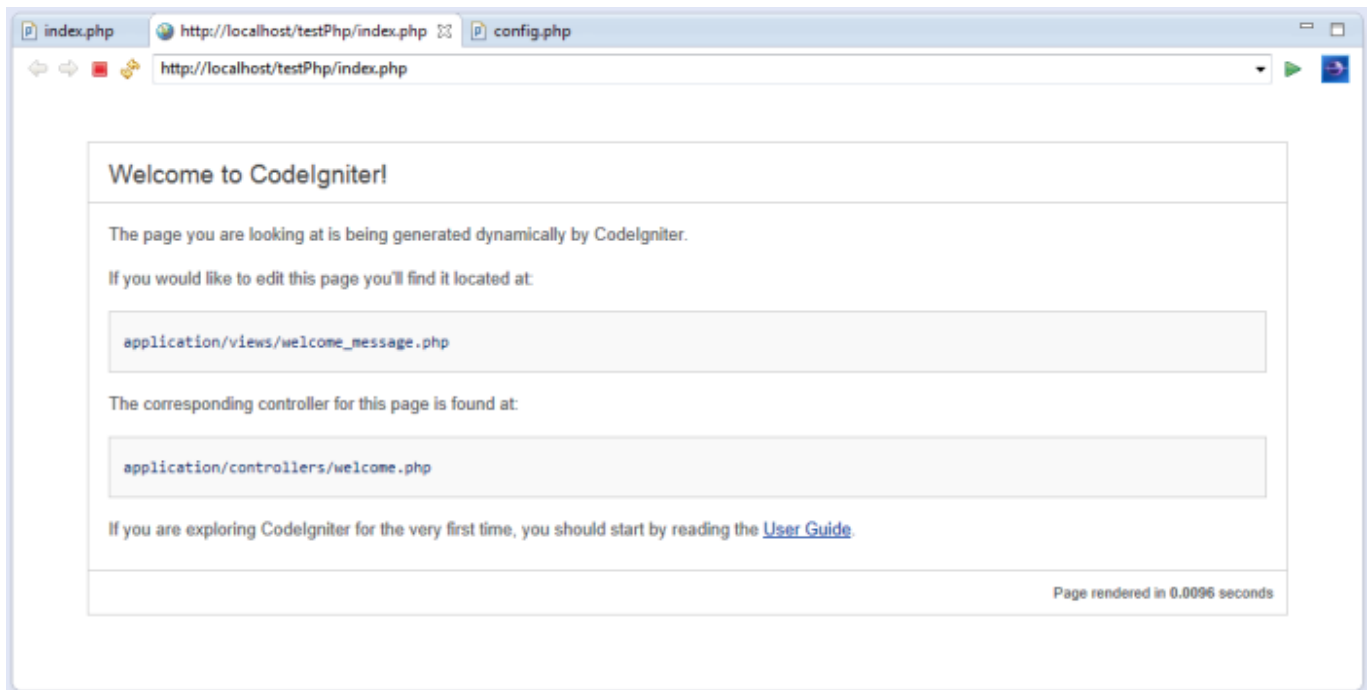
- Créer un Projet PHP
- Dézipper l'archive de CodeIgniter à la racine du projet

L'arborescence du projet est la suivante :



Le dossier **system** contient les fichiers propres à codeigniter, le dossier **application** les sources de notre application, déjà structurées.

Tester le bon fonctionnement en allant à l'adresse : <http://localhost/nomProjet/>



Structure d'une application

Il est indispensable par la suite de respecter la structure initiale d'une application codeigniter :

| Dossier | Rôle |
|-------------|--|
| assets | contient toutes les ressources qui vont être téléchargées par les visiteurs. Par exemple, les feuilles de style externes (CSS), les fichiers JavaScript et les images. |
| system | contient les sources de CodeIgniter, ne pas y toucher ! |
| application | contient tous nos scripts, qu'il faut placer dans les bons dossiers. |

Sous dossier application

| Dossier | Rôle |
|-------------|---|
| config | Fichiers permettant de configurer CodeIgniter ou une bibliothèque. Certains sont inclus automatiquement, d'autres à la demande. |
| controllers | Contient les contrôleurs. |
| errors | Contient les pages d'erreurs. |
| helpers | Dossier pour les helpers. Les helpers fournis avec CodeIgniter sont situés dans le dossier system. |
| hooks | Dossier contenant des fichiers permettant d'exécuter des scripts à différents moments du processus d'exécution de CodeIgniter. |
| language | Dossier contenant les fichiers de langue. |
| libraries | Dossier des bibliothèques. Comme les helpers, les bibliothèques fournies par CodeIgniter sont situées dans un autre dossier. |
| models | Dossier des models (classes métier). |
| views | Dossier des vues. |

Configuration de base

Vérifier que le mod reWrite d'Apache est activé. Ajouter le fichier htaccess suivant dans la racine de l'application Web, de façon à rediriger toutes les requêtes vers le fichier contrôleur **index.php** :

[.htaccess](#)

```
# Empêche la visualisation de l'arborescence, n'a rien à voir avec le
# masquage du « index.php ».
Options -Indexes

# Active le module de réécriture d'URL.
RewriteEngine on

#
# Fixe les règles de réécriture d'URL. Ici, nous utilisons une liste
# blanche.
#

# Toutes les URL qui ne correspondent pas à ces masques sont réécrites.
RewriteCond $1 !^(index\.php|assets/|robots\.txt)

# Toutes les autres URL vont être redirigées vers le fichier index.php.
RewriteRule ^(.*)$ index.php/$1 [L]
```

config.php

Modifier le fichier application/config/config.php :

Utiliser <http://randomkeygen.com/> pour générer l'**encryption_key**.

[|h config.php](#)

```
$config['base_url'] = 'http://localhost/testPhp/';
$config['index_page'] = '';
$config['url_suffix'] = '';
$config['encryption_key'] = 'GZD39H6F95i3jb34PA4sw1LccPI807uu';
```

database.php

Exécuter le script de création de la base de données dans phpmyadmin :

Définir les paramètres de connexion à la base dans database.php :

[|h database.php](#)

```
$db['default']['hostname'] = 'localhost';
$db['default']['username'] = 'root';
$db['default']['password'] = '';
$db['default']['database'] = 'tests';
$db['default']['dbdriver'] = 'mysql';
```

autoload.php

Chargement de bibliothèques et de helpers :

[|h autoload.php](#)

```
$autoload['libraries'] = array('database','session');
$autoload['helper'] = array('url');
```

Premier Contrôleur

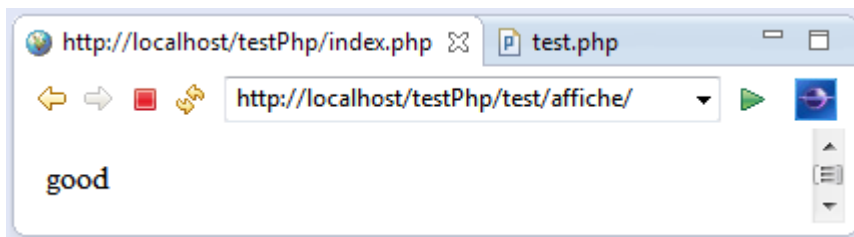
Création d'un contrôleur :

Créer la classe Test dans le dossier **controllers**, et nommez le fichier **test.php** :

[|h controllers/test.php](#)

```
<?php
class Test extends CI_Controller{
    public function affiche(){
        echo ("good");
    }
}
?>
```

Aller ensuite à l'adresse : <http://localhost/testPhp/test/affiche/>



- Un contrôleur est un fichier contenant une classe dont le nom commence par une majuscule.
- Le nom du fichier doit être le même que celui de la classe, mais en minuscule.
- Les méthodes de la classe sont directement accessibles via l'url :
`http://localhos/site/controllerClass/methodeName/`

Contrôleur et variables GET

Un contrôleur peut accepter les variables passées par un GET, par l'intermédiaire des paramètres déclarés dans le prototype de la méthode appelée :

[|h controlGet.php](#)

```
<?php
class Test extends CI_Controller{
    public function affiche(){
        echo ("good");
    }
}
```

```
    }  
    public function testGet($nom,$autre='') {  
        echo ('Le nom passé par GET est : '.$nom);  
        if($autre!='')  
            echo ('Le paramètre autre passé par GET est : '.$autre);  
    }  
}  
?>
```

La méthode testGet accepte 2 paramètres, le premier obligatoire, le second facultatif.

Tester les urls suivantes :

- <http://localhost/testPhp/test/testGet/>
- <http://localhost/testPhp/test/testGet/jcHeron>
- <http://localhost/testPhp/test/testGet/jcHeron/unAutre>

Vues

Création

Créer la vue accueil.php dans le dossier views :

|h views/accueil.php

```
<?php  
echo("<h1>Accueil</h1>")  
?>
```

Appel

Appeler la vue accueil.php dans une méthode du contrôleur Test :

|h controller/test.php

```
class Test extends CI_Controller {  
    function __construct() {  
        parent::__construct();  
    }  
    public function accueil() {  
        $this->load->view('accueil');  
    }  
}
```

Aller à l'adresse : <http://localhost/testPhp/test/accueil/>

Passage de variables du contrôleur à la vue

Modifier la méthode accueil de la façon suivante :

[|h controller/test.php](#)

```
<?php
class Test extends CI_Controller{
    function __construct(){
        parent::__construct();
    }
    public function accueil(){
        $data=array();
        $data['nom']='DUPONT';
        $data['prenom']='Maurice';
        $this->load->view('accueil',$data);
    }
}
?>
```

La vue accueil peut maintenant récupérer les variables passées dans le tableau \$data :

[|h views/accueil.php](#)

```
<?php
echo("<h1>Accueil</h1>");
echo ("Bienvenue ".$nom." ".$prenom);
?>
```

Il est possible d'appeler plusieurs vues dans le contrôleur :

```
$this->load->view('vue1');
$this->load->view('vue2',$data);
```

Il est également possible de mettre le résultat d'une vue dans une variable (en mettant à **true** le troisième paramètre) :

```
$vue1=$this->load->view('vue1',$data,true);
```

Helpers

Les helpers (classes apportant des fonctionnalités) peuvent être chargés automatiquement dans le fichier **autoload.php** :

```
$autoload['helper'] = array('url');
```

ou à la demande, dans un contrôleur par exemple :

```
$this->load->helper('url');
```

Le chargement d'un Helper permet ensuite d'appeler toutes les fonctions déclarées par le Helper.

[Helpers>>](#)

From:

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

Permanent link:

<http://slamwiki2.kobject.net/slam4/php/codeigniter?rev=1354980569>

Last update: **2019/08/31 14:41**

