Présentation du framework Code Igniter

par Jean-Marie RENOUARD

Date de publication : 15 décembre 2010

Dernière mise à jour : 13 mai 2011

I - Introduction

CodeIgniter est un environnement cadre de développement d'application, un ensemble d'outils permettant de structurer et de construire des sites Web en utilisant PHP.

Son objectif est de vous permettre de développer des projets beaucoup plus rapidement que si vous partiez de zéro, en fournissant un ensemble fourni de bibliothèques pour les tâches habituellement nécessaires, ainsi que d'une interface simple et une structuration logique d'accès à ces bibliothèques. CodeIgniter vous permet de vous concentrer sur votre créativité en minimisant la quantité de code nécessaire pour réaliser une tâche donnée.

I-A - But du tutorial

Le but du tutorial est de vous présenter une framework simple et léger capable de vous permettre de réaliser un mini site tel qu'un site ecommerce rapidement en vous fournissant le cadre de développement favorable à une progression rapide allié à une structuration idéale de votre projet.

I-B - Architecture logiciel MVC

L'architecture de CodeIgniter est basée sur le patron de conception MVC.

Il s'agit d'un patron de conception permettant de séparer le code d'accès aux données, la présentation et le contrôle de l'ensemble des actions.

L'article suivant est une très bonne introduction à ce patron de conception.

http://julien-pauli.developpez.com/tutoriels/php/mvc-controleur/

I-C - Utilisation de CodeIgniter

L'utilisation principale reste la mise en place d'un environnement structurant et simplifiant pour vos réalisations techniques.

D'après la documentation officielle CodeIgniter est fait pour vous si:

- Vous voulez un cadre de développement avec un faible impact de performance.
- Vous avez besoin de performances élevées.
- Vous avez besoin d'une large compatibilité de versions de PHP et de configurations.
- Vous voulez un cadre qui ne nécessitant quasiment aucune configuration pour démarrer.
- Vous voulez un cadre qui ne vous oblige pas à utiliser la ligne de commande.
- Vous voulez un cadre qui ne vous oblige pas à adhérer à des règles restrictives de codage.
- Vous ne voulez pas être obligés d'apprendre une langue de template supplémentaires.
- Vous éviter la complexité, en favorisant des solutions simples.
- Vous avez besoin d'une documentation claire et complète.

I-D - Pré requis techniques

Il faut aussi un serveur Apache Http 2.0 ou supérieure afin de pouvoir utiliser PHP en mode Web.

Le framework CodeIgniter est prévu pour fonctionner sur une large variété de version de PHP. Le minimum syndical ici et le PHP 5.1. L'ensemble des versions supérieures étant bien sur supportées. Les versions précédentes ne sont plus supportés depuis la version 2.0 de code Igniter.

Bien qu'il ne soit pas nécessaire, il est préférable d'appuyer son application sur une base de données permettant de garantir la pérennité des données. Le langage PHP supporte les bases de données suivantes : MySQL (4.1+), MySQL, MS SQL, PostgreSQL, Oracle, SQLite et ODBC. Le framework CodeIgniter offre ainsi de base le support d'accès à ces serveurs de base de données.

I-E - Récupération de la dernière version

Pour cet article de présentation de CodeIgniter 2.0.2, nous avons utilisé la dernière version de Code Igniter.

http://codeigniter.com/download.php

Il est possible de récupérer EasyPHP pour réaliser la mise en place des exemples.

EasyPHP5.3.3 est disponible à l'adresse suivante:

https://sourceforge.net/projects/quickeasyphp/files/EasyPHP/5.3.3/EasyPHP-5.3.3-setup.exe/download

Pour les environnements Linux, les utilitaires Yum, Apt-get et autres utilitaires d'installation sont disponibles afin de vous mettre à disposition l'ensemble des applications nécessaires.

II - Démarrer d'un projet CodeIgniter

II-A - Organisation générale de CodeIgniter

L'archive du projet est organisé en 4 parties distincts :

- Le script principal : index.php.
- Le fichier de licence de Code Igniter.
- Le répertoire contenant le manuel d'utilisateur.
- Le répertoire du moteur et de votre application.

II-B - Point d'entrée: index.php

Ce script est le point d'entrée de votre application et tous les appels passent systématiquement par ce script.

Il a pour rôle de coordonner les actions spécifiques de votre projet en les associant au moteur MVC décrits plus loin dans cet article.

Il faut noter 3 points importants dans votre script :

- Le niveau des affichages d'erreurs:

```
error_reporting(E_ALL);
```

- Le répertoire du moteur Code Igniter:

```
$system_folder = "system";
```

- Le répertoire de votre application :

```
$application_folder = "application";
```

Cela signifie que par défaut, il est possible avec Code Igniter de séparer le répertoire d'installation du moteur ($system_folder) du répertoire de votre application ($application_folder).

II-C - Structure des répertoires de votre application

II-C-1 - Index.html : le fichier de sécurité

Pour commencer chaque répertoire de votre application contient un fichier index.html afin d'éviter que l'on puisse lister le contenu de vos répertoires et en déduire la technologie utilisée.

II-C-2 - Structuration en répertoire

Le répertoire contient 9 sous répertoires permettant d'organiser chaque partie de votre application :

- Config: le répertoire contenant les fichiers de paramétrage.
- Controllers: le répertoire contenant le code de l'ensemble des contrôleurs.
- Error: le répertoire des pages d'erreurs par défaut.
- Helpers: le répertoire contenant vos classes utilitaires.
- Hooks: le répertoire contenant vos propres extensions noyau.

- Language: le répertoire contenant les fichiers de langage.
- Librairies: le répertoire contenant vos propres librairies.
- Models: le répertoire contenant vos classes d'utilisation de vos modèles de données.
- Views: le répertoire contenant vos classes d'affichage.

II-D - Sécurisation de l'installation de Code Igniter

II-D-1 - Paramétrage Apache pour interdire de listage des répertoires

Il est possible de paramétrer le serveur Apache par exemple afin de lui interdire explicitement le lister le répertoire.

Ajouter un *.htaccess* à la racine de votre répertoire d'application avec la directive suivante à la base de votre projet.

```
Options -indexes
```

II-D-2 - Retrait des informations inutiles

Afin de sécuriser votre installation, il est impératif de retirer l'inutile de votre serveur de production.

Dans le cadre d'un projet tel que Code Igniter, il est préférable de retirer les répertoires et fichiers suivants :

- Licence.txt : fichier de licence du produit capable de trahit votre framework en un clic.
- User_guide : il est inutile de laisser le manuel permettant à une personne mal intentionnée de disposer de la documentation directement sur le serveur.

II-E - Paramétrage des Urls

II-E-1 - Format des Urls par défaut

Le format des url Code Igniter par défaut est le suivant :

http://www.monsite.com/index.com/site/consultation

Le principale défaut est qu'il ne cache pas bien la technologie utilisée et il est aussi impératif actuellement d'avoir des sites avec des Urls génériques permettant d'évoluer ou de changer simplement de technologie sans casser la structure hiérarchique du site.

Cette structuration est importante dans le cadre de référencement de site Web car les moteurs de recherche n'apprécie pas vraiment le changement d'Urls d'un contenu quelconque et freine la progression du référencement de celui-ci.

II-E-2 - Format d'Urls plus génériques

L'idée est mettre en place le mécanisme de réécriture d'url d'Apache permettant de faire disparaître le nom du script index.php et de rendre le site portable à souhait vers d'autres technologies si le besoin apparaît.

L'url précédente doit devenir : **http://www.monsite.com/site/consultation**

II-E-3 - Paramétrage Apache pour la réécriture

Le fichier .htaccess suivant reprend la préconisation interdisant le listage des répertoires et y ajoute la règle de réécriture simple permettant le joli formatage des url de votre site en excluant la réécriture pour les fichiers suivants :

- Fichier index.php en lui-même.
- Les fichiers du répertoire images.
- Les fichiers du répertoire css.
- Les fichiers du répertoire js.
- Le fichier robots.txt.
- La racine du site.

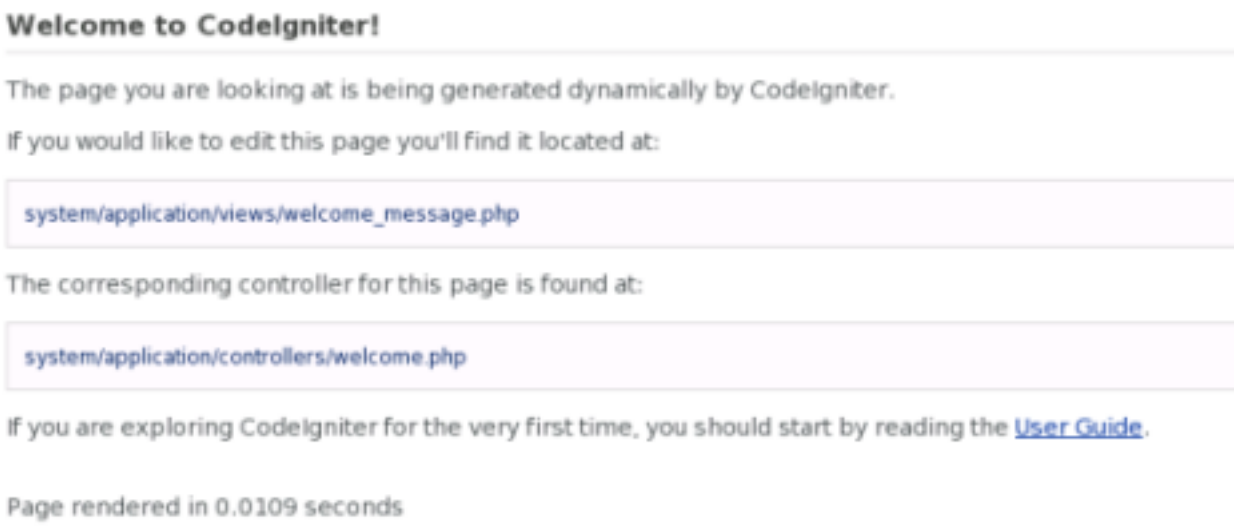
```
Options -indexes
RewriteEngine on
RewriteCond $1 !^(index\.php|css|js|images|robots\.txt|favicon\.ico|$)
RewriteRule ^(.*)$ index.php/$1 [L]
```

II-E-4 - Accès à l'installation initiale

La racine du site est accessible depuis les trois Urls suivantes :

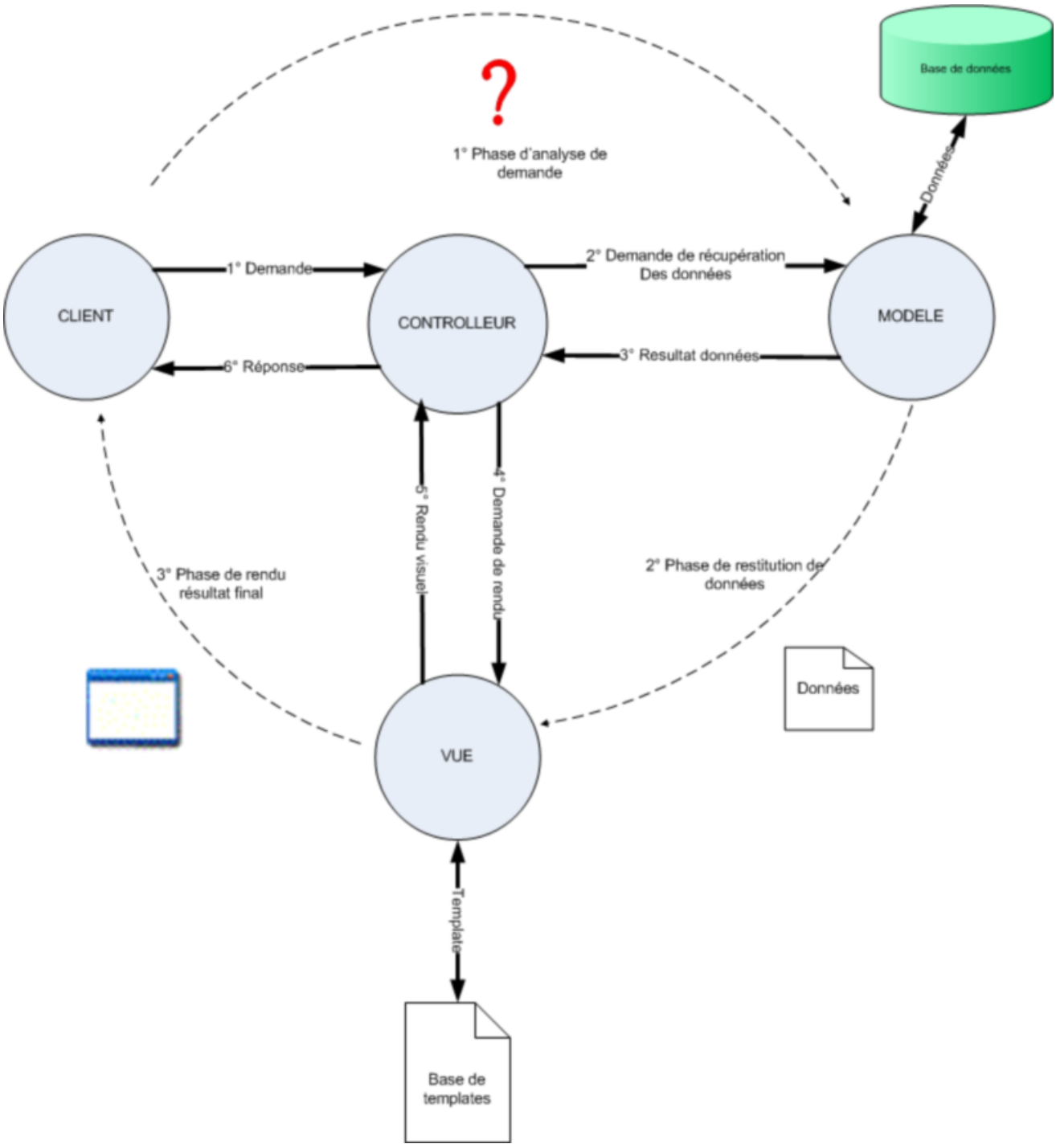
- http://www.monsite.com/
- http://www.monsite.com/index.php
- http://www.monsite.com/index

II-E-5 - Résultat de la page principale



III - Concepts fondamentaux de CodeIgniter

III-A - Architecture globale d'une application MVC



III-B - Présentation d'un Contrôleur

Le contrôleur en l'organe de contrôle du système. Il a en charge d'analyse des demandes clientes et l'orchestration des appels aux modèles et aux vues nécessaires à la fourniture de la réponse attendues. Il doit aussi effectuer le choix des appels et l'ordre de ceci. Dans Code Igniter, un contrôleur est une classe PHP héritant des propriétés de la clase Controler.

III-C - Présentation d'un modèle

Le Modèle est l'organe de récupération et de sélection des données pertinentes pour répondre à la demande. Son rôle consiste à récupérer, filtrer, modifier les données afin de fournir un sous ensemble de donnée pertinent pour la réponse.

Dans Code Igniter, un contrôleur est une classe PHP héritant des propriétés de la clase Model.

III-D - Présentation d'une vue

La Vue est l'organe en charge de produire la présentation des résultats en fonction de données qui lui sont fournies. Pour cela la vue s'appuie généralement sur des « Template » ou modèle de réponse auquel elle intègre les données afin de composer le résultat final.

Dans le cas de Code Igniter, les templates sont des fichiers PHP contenant du code de génération d'affichage uniquement.

III-E - Premier cas pratique: Bonjour le Monde !

III-E-1 - Ecrire son premier contrôleur

Le premier contrôleur peut être déposé dans le répertoire system/application/controllers/ et être nommé : Hello.php

```
<?php
class Hello extends Controller {
 function Hello()
 {
  parent::Controller();
 }

 function index()
 {
  $this->load->view('hello_message');
 }
}
/* End of file hello.php */
/* Location: ./system/application/controllers/hello.php */
?>
```

Il est conseillé fortement de respecter le formalisme du contrôleur et des commentaires de pied de page afin de garder votre code homogène et lisible.

Le moyen utilisé par le contrôleur pour associer une vue de votre traitement est réalisé par la l'appel suivant :

```
$this->load->view('hello_message');
```

III-E-2 - Ecrire sa première vue

Le premier contrôleur peut être déposé dans le répertoire system/application/views/ et être nommé : hello_message.php

```
<html>
 <head>
  <title>HELLO World</title>
 </head>
 <body>
  <h1>Hello World</h1>
```

```
  </body>
</html>
```

III-E-3 - Appel au nouveau contrôleur

L'appel peut-être effectuer sur par 2 URLS distinctes :

- **http://www.monsite.com/hello**
- **http://www.monsite.com/hello/**
- **http://www.monsite.com/hello/hello**
- **http://www.monsite.com/hello/index**

Le résultat est assez simple et va nous permettre de construire rapidement de nouvelle page par simple ajout de contrôleur et de vue.

Hello World

IV - Concepts fondamentaux - suite

IV-A - Présentation de l'utilisation des modèles

Un modèle est un organe permettant de récupérer, filtrer et organiser des données au sein d'une source de données persistantes.

Dans le cas de votre application, il s'agit d'une classe PHP héritant de la classe Model. Il s'agit d'une classe contenant des méthodes facilitant la manipulation des données applicatives en général et offrant un pont idéal entre les données et vos objets métier. Votre contrôleur se servira de votre modèle afin de récupérer les objets utiles et réalisera un appel à votre Vue afin de produire le rendu final associé à vos données.

IV-B - Présentation des classes utilitaires

IV-B-1 - Présentation des classes utilitaires

Un classe utilitaire est une classe contenant des fonctions utilitaires pour un domaine particulier de la programmation en d'application en PHP et avec Code Igniter.

IV-B-2 - Tableau de synthèse des classes utilitaires de Code Igniter

Nom de la classe utilitaire	Description
Array Helper	Cette classe utilitaire permet de manipuler des tableaux PHP. Fonctions : element() et ramdom_element() Chargement : $this->load->helper('array') ;
Compatibility Helper	Cette classe utilitaire permet de fournir des méthodes de compatibility entre PHP 4 et PHP5. Fonctions : file_put_contents(), fputcsv(), http_build_query(), str_ireplace() et stripos() Chargement : $this->load->helper('compatibility') ;
Cookie Helper	Cette classe utilitaire permet de manipuler des cookies HTTP. Fonctions : set_cookie(), get_cookie() et delete_cookie () Chargement : $this->load->helper('cookie') ;
Date Helper	Cette classe utilitaire permet de manipuler des dates en PHP. Fonctions : now(), mdate(), standard_date(), local_to_gmt(),mysql_to_unix(), unix_to_human(), human_to_unix(), timespan(), days_in_month(), timezones() et timezone_menu() Chargement : $this->load->helper('date') ;
Directory Helper	Cette classe utilitaire permet de manipuler des répertoires. Fonction : directory_map () Chargement : $this->load->helper('directory') ;
Download Helper	Cette classe utilitaire permet de manipuler les entêtes http pour forcer le téléchargement plutôt que l'affichage de données. Fonction : force_download ()

	Chargement : $this->load->helper('download) ;
Email Helper	Cette classe utilitaire permet de gérer la validité et l'envoi d'emails. Fonction : email () Chargement : $this->load->helper('email) ;
File Helper	Cette classe utilitaire permet de manipuler des fichiers. Fonctions : read_file(), write_file(), delete_files(), get_filenames(), get_dir_file_info(), get_file_info(), get_mime_by_extension(), symbolic_permissions() et octal_permissions() Chargement : $this->load->helper('file') ;
Form Helper	Cette classe utilitaire permet de générer par programmation des formulaires HTML. Fonctions :form_open(), form_open_multipart(), form_hidden(), form_input(), form_password(), form_upload(), form_textarea(), form_dropdown(), form_multiselect(),form_fieldset(), form_fieldset_close(), form_checkbox(), form_radio(), form_submit(), form_label(), form_reset(), form_button(), form_close(), form_prep(), set_value(), set_select() et set_radio() Fonctions : Chargement : $this->load->helper('form') ;
HTML Helper	Cette classe utilitaire permet de gérer des éléments HTML. Fonctions : br(), heading(), img(), link_tag(), nbs(), ol(), ul(), meta() et doctype() Chargement : $this->load->helper('html') ;
Inflector Helper	Cette classe utilitaire permet de modifier des chaînes de caractères. Fonctions : singular(), plural(), camelize(), underscore() et humanize() Chargement : $this->load->helper('inflector') ;
Language Helper	Cette classe utilitaire permet de gérer les tranductions. Fonction : lang() Chargement : $this->load->helper('language') ;
Number Helper	Cette classe utilitaire permet de traduire des valeurs en octets dans un format lisibles avec unité. Fonction : byte_format() Chargement : $this->load->helper('number') ;
Path Helper	Cette classe utilitaire permet de récupérer le chemin absolu d'une ressource. Fonctions : set_realpath() Chargement : $this->load->helper('path') ;
Security Helper	Cette classe utilitaire permet de faciliter la gestion de la sécurité au sein de votre application. Fonctions : xss_clean(), dohash(), strip_image_tags() et encode_php_tags() Chargement : $this->load->helper('security') ;
Smiley Helper	Cette classe utilitaire permet de traduire des smileys en image à partir de ressources fournies par le projet Code Igniter http://codeigniter.com/download_files/smileys.zip Fonctions : get_clickable_smileys, smiley_js() et parse_smileys() Chargement : $this->load->helper('smiley') ;
String Helper	Cette classe utilitaire permet de manipuler et générer des chaînes de caractères.

	Fonctions : random_string(), alternator(), repeater(), reduce_double_slashes(), trim_slashes(), reduce_multiples() et quotes_to_entities() Chargement : $this->load->helper('string') ;
Text Helper	Cette classe utilitaire permet de manipuler du texte. Fonctions : word_limiter(), character_limiter(), ascii_to_entities(), entities_to_ascii(), word_censor(), highlight_code(), highlight_phrase() et word_wrap() Chargement : $this->load->helper('text') ;
Typography Helper	Cette classe utilitaire permet de gérer la typographie d'un texte dans la page Web. Fonctions : auto_typography () et nl2br_except_pre () Chargement : $this->load->helper('typography') ;
URL Helper	Cette classe utilitaire permet de manipuler et gérer des URLs. Fonctions : site_url(), base_url(), current_url(), uri_string(), index_page(), anchor(), anchor_popup(), mailto(), safe_mailto(), auto_link(), url_title(), prep_url() et redirect() Chargement : $this->load->helper('url') ;
XML Helper	Cette classe utilitaire permet de gérer les caractères réservés en XML en les convertissant. Fonction : xml_convert () Chargement : $this->load->helper('xml) ;

IV-C - Présentation des classes librairie de Code Igniter

IV-C-1 - Présentation des librairies Code Igniter

Une librairie est une classe spécifique de Code Igniter permettant de réaliser des tâches complexes dans un domaine particulier de la programmation en d'application en PHP et avec Code Igniter. Une librairie est un organe plus complexe qu'une simple classe utilitaire nécessitant une gestion plus avancée des états.

IV-C-2 - Tableau de synthèse des librairies de Code Igniter

Nom de la classe utilitaire	Description
Benchmarking Class	Cette librairie permet de réaliser des mesures de temps de traitements de votre application. Chargement :$this->output->enable_profiler(TRUE);
Calendar Class	Cette librairie permet de fournir des méthodes de gestion de calendrier. Chargement : $this->load->library('calendar') ;
Cart Class	Cette librairie permet de gérer un caddy pour site de ecommerce. Chargement : $this->load->library('cart') ;
Config Class	Cette librairie permet de gérer des fichiers de configuration. Chargement : Automatique
Database Class	Cette librairie permet de manipuler des données dans une base de données. Chargement : $this->load->library('database') ;
Email Class	Cette librairie permet de manipuler et envoyer des emails avec des fonctionnalités avancées tel que les pièces jointes.

	Chargement : $this->load->library('email') ;
Encryption Class	Cette librairie permet de manipuler le chiffrage et de déchiffrage de données. Chargement : $this->load->library('encrypt') ;
File Uploading Class	Cette librairie permet de gérer l'upload de fichier depuis le navigateur client. Chargement : $this->load->library('upload');
Form Validation Class	Cette librairie permet de gérer la validation des formulaires HTML. Chargement : $this->load->library('form_validation') ;
FTP Class	Cette librairie permet de gérer des transferts de fichiers via FTP. Chargement : $this->load->library('ftp') ;
HTML Table Class	Cette librairie permet de générer des tableaux HTML. Chargement : $this->load->library('table') ;
Image Manipulation Class	Cette librairie permet de manipuler des images, créer des vignettes, retailler, effectuer des rotations et des fusions. Chargement : $this->load->library('image_lib') ;
Input and Security Class	Cette librairie permet de gérer le filtrage et la manipulation sécurisée des données. Chargement : Automatique
Loader Class	Cette librairie permet de gérer le chargement des différents éléments dans Code Igniter. Chargement : Automatique
Language Class	Obsolète et remplacé par la classe utilitaire Language.
Output Class	Cette librairie permet de gérer le résultat final envoyé au client. Chargement : Automatique
Pagination Class	Cette librairie permet de gérer la pagination et la navigation dans vos pages. Chargement : $this->load->library('pagination') ;
Session Class	Cette librairie permet de manipuler et de gérer les sessions. Chargement : $this->load->library('session') ;
Trackback Class	Cette librairie permet de gérer et manipuler des retroliens ou trackback. Chargement : $this->load->library('trackback') ;
Template Parser Class	Cette librairie permet de supporter un langage simple de template pour vos pages Web. Chargement : $this->load->library('parser') ;
Typography Class	Cette librairie permet de gérer la typographie de vos textes. Chargement : $this->load->library('typography') ;
Unit Testing Class	Cette librairie permet de réaliser et exécuter des tests unitaires. Chargement : $this->load->library('unit_test') ;
URI Class	Cette librairie permet de manipuler des URI. Chargement : Automatique
User Agent Class	Cette librairie permet de manipuler l'entête User-Agent. Chargement : $this->load->library('user_agent') ;
XML-RPC Class	Cette librairie permet de fournir un service XML au format XML-RPC. Chargement : $this->load->library('xmlrpc') ;
Zip Encoding Class	Cette librairie permet de manipuler des données à au format ZIP.

	Chargement : $this->load->library('zip') ;

IV-D - Chargement automatique de composant

Il se peut que vous ayez sans arrêt les mêmes éléments à charger dans votre contrôleur.

```
$this->load->model('...');
$this->load->librairie('...');
$this->load->helper('...');
$this->load->plugin('...');
$this->load->config('...');
$this->load->lang ('...');
```

Et afin de ne pas surcharger votre contrôleur de ses lignes de code, il est possible de procéder à leur chargement automatique à chaque appel.

Pour cela il faut modifier le fichier application/config/autoload.php contenant l'ensemble des éléments à pré charger automatiquement.

Attention car à chaque appel, l'ensemble du codes des éléments à pré charger sera lu, interprété et exécuté avant de passer à votre application.

```
$autoload['libraries'] = array();
$autoload['helper'] = array();
$autoload['plugin'] = array();
$autoload['config'] = array();
$autoload['language'] = array();
$autoload['model'] = array();
```

Il suffit d'ajouter le nom des éléments à charger afin de les voir pris en compte automatiquement dans votre application.

Par exemple, si votre application utilise massivement des bases de données et la validation de formulaire ainsi que l'envoi d'email, il est probable que la ligne suivante puisse correspondre à votre besoin.

```
$autoload['libraries'] = array('database', 'email', 'form_validation');
```

De même, si votre application nécessite un fichier de configuration supplémentaire.

Il suffit d'ajout un fichier de configuration dans le répertoire application/config/mesconfigs.hp et de le charger automatiquement avec;:

```
$autoload['config'] = array('mesconfigs');
```

V - Présentation de quelques librairies utiles

V-A - Librairie de gestion des bases de données

V-A-1 - Paramétrage de l'accès à la base de données

La configuration de l'accès à la base de données est réalisée dans le fichier de configuration application/configuration/database.php.

Il existe une configuration par défaut:

```
$db['default']['hostname'] = "localhost";
$db['default']['username'] = "admin";
$db['default']['password'] = "";
$db['default']['database'] = "basededonnees";
$db['default']['dbdriver'] = "mysql";
$db['default']['dbprefix'] = "";
$db['default']['pconnect'] = TRUE;
$db['default']['db_debug'] = FALSE;
$db['default']['cache_on'] = FALSE;
$db['default']['cachedir'] = "";
$db['default']['char_set'] = "utf8";
$db['default']['dbcollat'] = "utf8_general_ci";
```

Afin de charger la configuration par défaut, il suffit d'ajouter l'appel suivant dans votre code.

```
$this->load->database();
```

Le chargement sera effectif et vous pourrez immédiatement utiliser le composant d'accès aux bases de données.

Il est cependant possible de créer de multiples configurations en ajoutant un nouveau sous-ensemble.

```
$db['autreconfig']['hostname'] = "localhost";
$db['autreconfig']['username'] = "admin";
$db['autreconfig']['password'] = "";
$db['autreconfig']['database'] = "autrebasededonnees";
$db['autreconfig']['dbdriver'] = "mysql";
$db['autreconfig']['dbprefix'] = "";
$db['autreconfig']['pconnect'] = TRUE;
$db['autreconfig']['db_debug'] = FALSE;
$db['autreconfig']['cache_on'] = FALSE;
$db['autreconfig']['cachedir'] = "";
$db['autreconfig']['char_set'] = "utf8";
$db['autreconfig']['dbcollat'] = "utf8_general_ci";
```

Afin de charger la nouvelle configuration, il suffit d'ajouter l'appel suivant dans votre code en y précisant le nom de la configuration à charger.

```
$this->load->database('autreconfig');
```

Le chargement sera effectif et vous pourrez immédiatement utiliser le composant d'accès aux bases de données.

V-A-2 - Exécution et validation une requête SQL

La solution pour exécuter une requête SQL sans récupération de donnée en dehors du résultat du bon fonctionnement de la requête SQL est d'utiliser la méthode $this->db->simple_query).

Exemple:

```
$this->load->database('autreconfig');
$res=$this->db->simple_query("INSERT INTO ARTICLE VALUES ( ... )");
if ( $res== TRUE ) echo "Requete [OK]";
else echo " Requete [ECHEC]";
```

V-A-3 - Exécution et récupération des résultats au format objet

L'utilisation la plus courante de votre base de données est la consultation et l'affichage pour cela il est possible d'utiliser les méthodes $this->db->query() et $query->result(). Les résultats sont retournés sous forme d'objet.

```
$query = $this->db->query("SELECT * FROM ARTICLE");
if ($query->num_rows() > 0) {
 foreach ($query->result() as $row) {
  echo $row->ID;
  echo $row->LABEL;
  echo $row->PRIXHT;
  }
} else { echo "Pas d'article."; }

$query->free_result();
```

Il est possible de récupérer élément par élément plutôt qu'en masse, les données résultat.

Voici donc une version permettant de ne pas allouer massivement de la mémoire au début de la boucle for.

```
$query = $this->db->query("SELECT * FROM ARTICLE");
if ($query->num_rows() > 0) {
 for ($i=0; $i<$query->num_rows(); $i++) {
  $row= $query->row($i);
  echo $row->ID;
  echo $row->LABEL;
  echo $row->PRIXHT;
 }
} else { echo "Pas d'article."; }

$query->free_result();
```

V-A-4 - Exécution et récupération des résultats au format tableau

Parfois, il est plus pratique pour des raisons de compatibilité de manipuler des tableaux PHP que des objets PHP. Code Igniter fournit l'ensemble des outils pour manipuler des tableaux au lieu des objets.

```
$query = $this->db->query("SELECT * FROM ARTICLE");

if ($query->num_rows() > 0) {
  foreach ($query->result_array() as $row) {
   echo $row['ID'];
   echo $row['LABEL'];
   echo $row['PRIXHT'];
  }
} else { echo "Pas d'article."; }
$query->free_result();
```

Il est possible de récupérer élément par élément plutôt qu'en masse, les données résultat.

Voici donc une version permettant de ne pas allouer massivement de la mémoire au début de la boucle for.

```
$query = $this->db->query("SELECT * FROM ARTICLE");
if ($query->num_rows() > 0) {
 for ($i=0; $i< $query->num_rows(); $i++)> {
  $row= $query->row_array($i);
  echo $row['ID'];
  echo $row['LABEL'];
  echo $row['PRIXHT'];
 }
} else { echo "Pas d'article."; }
$query->free_result();
```

V-A-5 - Utilisation d'Active Record pour générer les échanges

Active Record est un patron de conception permettant de faciliter l'écriture de requêtes SQL par programmation.

V-A-6 - Méthodes disponibles pour la lecture

L'implémentation d'Active Record dans Code Igniter est riche et permet de plus de chaîner les appels de méthodes.

Voici la liste des méthodes permettant par programmation de générer une requête SQL équivalente;:

- $this->db->get()
- $this->db->get_where()
- $this->db->select()
- $this->db->select_max()
- $this->db->select_min()
- $this->db->select_avg()
- $this->db->select_sum()
- $this->db->from()
- $this->db->join()
- $this->db->where()
- $this->db->or_where()
- $this->db->where_in()
- $this->db->or_where_in()
- $this->db->where_not_in()
- $this->db->or_where_not_in()
- $this->db->like()
- $this->db->or_like()
- $this->db->not_like()
- $this->db->or_not_like()
- $this->db->group_by()
- $this->db->distinct()
- $this->db->having()
- $this->db->or_having()
- $this->db->order_by()
- $this->db->limit()
- $this->db->count_all_results()
- $this->db->count_all()

V-A-7 - Exemples de lectures avec Active Record

Récupération tous les articles

```
$query=$this->db->get('ARTICLE');
```

Récupération du premiers articles de la base

```
$this->db->select('LABEL')->from('ARTICLE')->limit(1);
```

```
$query = $this->db->get();
```

- Récupération des 10 premiers labels des articles de moins de 100 euros hors taxes en donnant les moins chers.

```
$this->db-
>select('LABEL')->from('ARTICLE')->where('PRIXHT <', 100)->order_by('PRIXHT', 'ASC')->limit(10);
$query = $this->db->get();
```

V-A-8 - Méthodes disponibles pour l'insertion

Les méthodes suivantes permettent d'insérer des données dans vos bases de données.

- $this->db->insert()
- $this->db->set()

V-A-9 - Exemples d'insertions avec Active Record

V-A-10 - Méthodes disponibles pour la mise à jour

La méthode suivante permet de mettre à jour simplement vos données.

- $this->db->update()

V-A-11 - Exemples de mises à jour avec Active Record

V-A-12 - Méthodes disponibles pour la mise à jour

Les méthodes suivantes permettent de mettre à jour simplement vos données.

- $this->db->delete()
- $this->db->empty_table()
- $this->db->truncate()

V-A-13 - Exemples de mises à jour avec Active Record

V-A-14 - Méthodes de mise en cache avec Active Record

- $this->db->start_cache()
- $this->db->stop_cache()
- $this->db->flush_cache()

V-A-15 - Exemples de mises en cache avec Active Record

V-B - Librairie de gestion des emails

La librairie de gestion de mail permet à votre application de gérer l'envoi d'email.

Le chargement de la librairie est simple.

$this->load->library('email');

Cette librairie offre le support de plusieurs protocoles Mail (Mail, Sendmail, et SMTP) afin de mieux s'adapter à votre environnement.

La librairie offre l'ensemble des fonctionnalités tels que le support des destinataires en copie (CC) et en copie cachée (BCC).

Elle permet la découpage des textes et l'envoi au format teste ou HTML.

Les pièces jointes peuvent être ajoutés dans vos messages.

V-B-1 - Méthodes disponibles pour l'envoi de messages email

Les méthodes suivantes permettent de gérer la création, envoi et l'analyse de vos message email dans votre application.

- $this->email->from()
- $this->email->reply_to()
- $this->email->to()
- $this->email->cc()
- $this->email->bcc()
- $this->email->subject()
- $this->email->message()
- $this->email->set_alt_message()
- $this->email->clear()
- $this->email->send()
- $this->email->attach()
- $this->email->print_debugger()

V-B-2 - Exemple simple d'envoi d'email

```
$list= array(
"Jean-Marie" => "jmrenouard@gmail.com",
"Sophie" => "sophie@gmail.com",
"Clement" => '<link href="mailto:clement@gmail.com">clement@gmail.com</link>',
);

foreach ($list as $prenom => $email) {
 $this->email->clear();
 $this->email->to($email);
 $this->email->from('codeigniter@gmail.com');
 $this->email->subject('Message à l'attention de '.$prenom);
 $this->email->message("Bonjour $prenom,\nVous avez gagnez une voiture
\n après avoir répondu à nos 100 questions...");

 $this->email->send();
}
```

V-B-3 - Exemple d'envoi d'email avec pièces jointes

```
$list= array(
"Jean-Marie" => "jmrenouard@gmail.com",
"Sophie" => "sophie@gmail.com",
"Clement" => '<link href="mailto:clement@gmail.com">clement@gmail.com</link>',
);

foreach ($list as $prenom => $email) {
 $this->email->clear();
 $this->email->to($email);
 $this->email->from('codeigniter@gmail.com');
 $this->email->subject('Message à l'attention de '.$prenom);
 $this->email->message("Bonjour $prenom,\nVous avez gagnez une voiture
\n après avoir répondu à nos 100 questions...");
 $this->email->attach('/doc/logo.jpg');
 $this->email->attach('/doc/questionnaire.doc');
 $this->email->attach('/doc/image_billets.jpg');
 $this->email->send();>
}
```

V-C - Librairie de création de Web Services

Le framework Code Igniter permet de gérer vos propres Web Services via le support du protocole XML-RPC. Il s'agit d'un protocole simple donc la spécification se trouve **http://www.xmlrpc.com/**.

XML-RPC est un protocole léger de Web services basée sur XML.

Il offre l'avantage de s'abstraire des environnements techniques spécifiques (Windows/Linux) et des problématiques de transcodages de résultat et des demandes.

Il s'agit donc d'un protocole idéal pour lier plusieurs technologies hétérogènes au sein d'un système d'information sans provoquer un risque d'incompatibilité entre application.

V-C-1 - Méthodes disponibles pour gérer vos échanges XML-RPC

- $this->xmlrpc->server()
- $this->xmlrpc->timeout()
- $this->xmlrpc->method()
- $this->xmlrpc->request()
- $this->xmlrpc->send_request()
- $this->xmlrpc->set_debug(TRUE)
- $this->xmlrpc->display_error()
- $this->xmlrpc->display_response()
- $this->xmlrpc->send_error_message()
- $this->xmlrpc->send_response()

V-C-2 - Gestion des appels à un service XML-RPC

Il est possible de gérer des appels à des services distants par XML-RPC.

Idéal, pour structurer des services au sein d'une entreprise et permettre de mettre à disponibilité des informations et des outils de traitement de l'information.

Voici donc un exemple de contrôleur permettant d'interroger un serveur XML-RPC.

```php
<?php
class Xmlrpc_client extends Controller {
 function index()
 {
  $this->load->helper('url');
  $server_url = site_url('xmlrpc_server');
  $this->load->library('xmlrpc');
  $this->xmlrpc->server(http://www.monsitecom/xml_server, 80);
  $this->xmlrpc->method('ping');
  $request = array('Comment vas-u ?');
  $this->xmlrpc->request($request);

  if ( ! $this->xmlrpc->send_request()) {
   echo $this->xmlrpc->display_error();
  } else {
   echo '<pre>';
   print_r($this->xmlrpc->display_response());
   echo '</pre>';
  }
 }
}
?>
```

V-C-3 - Gestion des appels entrant au format XML-RPC

La librairie Code Igniter permet de créer un web service rapidement en complément de votre site Web.

Il permet de lier vos partenaires à votre site et d'automatiser certains échanges.

```php
<?php
class Xmlrpc_server extends Controller {
 function index()
 {
  $this->load->library('xmlrpc');
  $this->load->library('xmlrpcs');
  $config['functions']['Ping'] = array('function' => 'Xmlrpc_server.process');
  $this->xmlrpcs->initialize($config);
  $this->xmlrpcs->serve();
 }
 function process($request)
 {
  $parameters = $request->output_parameters();
  $response = array(
    array(
     'Tu dis : ' => $parameters['0'],
     'Je reponds' => 'Cc est plutôt bon signe :).'),
    'struct');
  return $this->xmlrpc->send_response($response);
 }
}
?>
```

V-D - Librairie de gestion d'accès FTP

Le protocole FTP est idéal pour gérer les transferts de fichier.

Ce protocole est largement utilise pour les transfert de site. Il est donc possible d'imaginer un site qui vous permette de déployer vos configurations et applications chez un hébergeur directement depuis un site d'administration.

V-D-1 - Méthodes disponibles pour gérer vos échanges FTP

- $this->ftp->connect()
- $this->ftp->upload()
- $this->ftp->rename()
- $this->ftp->move()
- $this->ftp->delete_file()
- $this->ftp->delete_dir()
- $this->ftp->list_files()
- $this->ftp->mirror()
- $this->ftp->mkdir()
- $this->ftp->chmod()
- $this->ftp->close()

V-D-2 - Exemples d'utilisation de FTP

Uploader un fichier

```
$this->load->library('ftp');

$config['hostname'] = 'ftp.free.fr';
$config['username']= 'jmrenouard';
$config['password'] = 'XXXXXX';
$this->ftp->connect($config);
$this->ftp->upload('/site/monfichier.html', '/public_html/monfichier.html');
$this->ftp->close();
<u>Effectuer une copie intégrale de repertoire par FTP</u>

$this->load->library('ftp');
$config['hostname'] = 'ftp.free.fr';
$config['username'] = 'jmrenouard';
$config['password'] = 'XXXXXX';
$this->ftp->connect($config);
$this->ftp->mirror(('/site/','/public_html');
$this->ftp->close();
```

Lister un repertoire par FTP

```
$this->load->library('ftp');
$config['hostname'] = 'ftp.free.fr';
$config['username'] = 'jmrenouard';
$config['password'] = 'XXXXXX';

$this->ftp->connect($config);
$list=$this->ftp->list('/public_html');
print_r($list);
$this->ftp->close();
```

V-E - Librairies de gestion de formulaire

La librairie de validation de formulaire est un outil idéal pour vous garantir une gestion simple des formulaires HTML en garantissant la sécurité et la validation des paramètres passés dans le formulaire.

V-E-1 - Méthodes disponibles pour gérer vos formulaires

- $this->form_validation->set_rules()
- $this->form_validation->run();

- $this->form_validation->set_message()

V-E-2 - Fonctions de la classe utilitaire

- form_error()
- validation_errors()
- set_value()
- set_select()
- set_checkbox()
- set_radio()

V-F - Exemple simple de gestion de la validation de formulaire

Voici un exemple simple de formulaire de création de compte.

V-F-1 - La vue du formulaire : view/formulaire.php

La vue formulaire permet de fabriquer une vue prenant en compte et l'affichage du formulaire et des erreurs éventuelles. Ces erreurs seront affichées en haut du formulaire et devant le champ ne respectant pas les règles de validation.

```
<html>
 <head>
 <title>MON FORMULAIRE</title>
 </head>

 <body>
  <?php echo validation_errors(); ?>
  <?php echo form_open('form'); ?>
  <h5>Login</h5>

  <?php echo form_error('login'); ?>
  <input type="text" name="login" value="<?php echo set_value('login'); ?>" size="50" />
  <h5>Password</h5>
  <?php echo form_error('password'); ?>
  <input type="text" name="password" value="<?php echo set_value('password'); ?>" size="50" />
  <h5>Validation du Password </h5>
  <?php echo form_error('passconf'); ?>
  <input type="text" name="passconf" value="<?php echo set_value('passconf'); ?>" size="50" />
  <h5>Adresse Email</h5>
  <?php echo form_error('email'); ?>
  <input type="text" name="email" value="<?php echo set_value('email'); ?>" size="50" />
  <div><input type="submit" value="Submit" /></div>

  </form>
 </body>
</html>
```

V-F-2 - La vue de confirmation de création de compte : view/succes.php

Il s'agit d'une vue simple permettant de construire une page de résultat indiquant que les règles ont été respectés et le traitement du formulaire effectués.

```
<html>
 <head>
 <title>Mon formulaire</title>
 </head>
 <body>
```

```
  <h3>Le formulaire de création de compte a bien été transmis !</h3>
  <p><?php echo anchor('formulaire', 'Essayez à nouveau'); ?></p>
 </body>
</html>
```

V-F-3 - Le contrôleur permettant de gérer la validation du formulaire : controllers/form.php

Le contrôleur est l'élément central de la procédure de validation.

Les règles de validation sont positionnées par la méthode;: $this->form_validation->set_rules() et la validation est effectué par la méthode $this->form_validation->run()

```
<?php
class Form extends Controller {
 function index()
 {
  $this->load->helper(array('form', 'url'));
  $this->load->library('form_validation');
  $this->form_validation->set_rules('login', 'Username', 'required');
  $this->form_validation->set_rules('password', 'Password', 'required');
  $this->form_validation->set_rules('passconf', 'Password Confirmation', 'required');
  $this->form_validation->set_rules('email', 'Email', 'required');

  if ($this->form_validation->run() == FALSE) {
   $this->load->view('formulaire');
  } else { $this->load->view('succes'); }
 }
}
?>
```

V-F-4 - Modification avancée des paramètres de filtrage

Il est possible de définir les règles de paramétrage de la validation dans un tableau PHP. Cette version utilise ce modèle. De même, les règles de validation sont enrichies en utilisant la cascade de règles.

La règle suivante;: trim|required|min_length[5]|max_length[12]|xss_clean'

Indique que;:

- trim: la chaîne va etre réduite des espaces au début et à la fin.
- required: indique que la chaine ne peut être vides après réduction.
- min_length[5]: indique que la chaîne doit faire au moins 5 caractères.
- max_length[12]: indique que la chaîne doit faire au plus 121 caractères.
- xss_clean: indique que la chaîne doit être sécurisé contre une attaque type XSS.

```
$config = array(
 array('field'=> 'login','label'=> 'Login','rules'=> 'trim|required|min_length[5]|max_length[12]|
xss_clean'),
 array('field'=> 'password','label'=>'Password', 'rules'=> 'trim|required|matches[passconf]|md5'),
 array('field'=> 'passconf', 'label'=> 'Validation du Password','rules'=> 'trim|required|
matches[passconf]|md5'),
 array('field'=> 'email', 'label'=> 'Email', 'rules'=> 'trim|required|valid_email')
);
$this->form_validation->set_rules($config);
```

VI - Concepts avancée de URLS

VI-A - Gestion avancée des URLs

VI-A-1 - Redéfinition d'URL par configuration

Le fichier de configuration application/config/routes.php permet de définir les contrôleurs associés à une URI.

```
$route['journals'] = "blogs";
$route['product/(:num)'] = "catalog/product_lookup_by_id/$1";
```

Tous les appels à une URI commençant par /journals seront dirigés vers le contrôleur blogs.

Dans le second cas, tous les appels à une URI commençant par /products/{un nombre quelconque} seront dirigés vers le contrôleur catalog et la méthode product_lookup_by_id avec comme paramètre le nombre.

- **http://www.monsite.com/journals**
- **http://www.monsite.com/journals/**
- **http://www.monsite.com/journals/index**
- **http://www.monsite.com/journals/toto**
- **http://www.monsite.com/products/34**
- **http://www.monsite.com/products/5**

VI-A-2 - Redéfinition d'URL par programmation

Dans chaque contrôleur, il est possible de définir la distribution dans les méthodes par la méthode _remap

```
function _remap($method) {
 if ($method == 'methode_inconnue') {
  $this->$index();
 } else {
  $this->default_method();
 }
}
```

VI-B - Gestion des erreurs

La gestion des erreurs peuvent être réalisées par programmation et sont en général géré par le framework.

VI-B-1 - Les vues associées aux pages d'erreur

Il est possible de spécifier 4 pages spécifiques d'erreur;:

- error_404.php: pour les pages non trouvées.
- error_db.php: pour les pages d'erreur associés aux bases de données.
- error_php.php: pour les pages d'erreur associés aux problèmes de code PHP.
- error_general.php: pour les pages d'erreur associés aux problèmes généraux.

VI-B-2 - Méthodes de gestion des erreurs

- show_error('*message*' [, int *$status_code*= 500])

- show_404('*page*')
- log_message('*level*', 'message')

VI-C - Appel en ligne de commande

Il est possible d'écrire un client pour vos applications Code Igniter afin d'appeler vos méthodes directement sans passer par un navigateur ou un client http.

VI-C-1 - Code du client Code Igniter

```
#!/usr/bin/php
<?php
/* Eviter les coupures de timeout */
set_time_limit(0);

/* Definir une limite de buffer mémoire suffissante */
ini_set('memory_limit', '256M');

/* Eviter d'utiliser le script depuis un navigateur */
if (isset($_SERVER['REMOTE_ADDR'])) die('Permission denied.');

/* Possitionnement de quelques variables */
define('CMD', 1);
$ROOT_DIR="/var/www/html/site/";

/* retrait du premier paramétre : le nom du script */
unset($argv[0]);

/* Positionnement des parameters pour retrouver le bon controlleur et la benne méthde */
$_SERVER['REQUEST_URI'] = '/' . implode('/', $argv) . '/';
$_SERVER['QUERY_STRING'] = $_SERVER['PATH_INFO'] = $_SERVER['REQUEST_URI'] ;
/* Appel au framework */
include("$ROOT_DIR/index.php");
?>
```

VI-C-2 - Exemple d'utilisation

```
$ php ci_call.php ecommerce index
```

...

```
$ php ci_call.php xmlrpc_client ping
```

...

VI-D - Paramétrage pour la production

Il est important de paramétrer votre php.ini en production afin qu'il garantisse un minimum de protection.

Il est conseiller de désactiver l'envoi de l'entête http avec le numéro de version PHP et de désactiver l'affichage des erreurs PHP dans la fenêtre de résultat même si les pages peuvent être modifier pour l'affichage.

VI-D-1 - Paramètres à valider en production

```
Expose_php =Off
Display_errors=Off
```

VII - Amélioration des performances

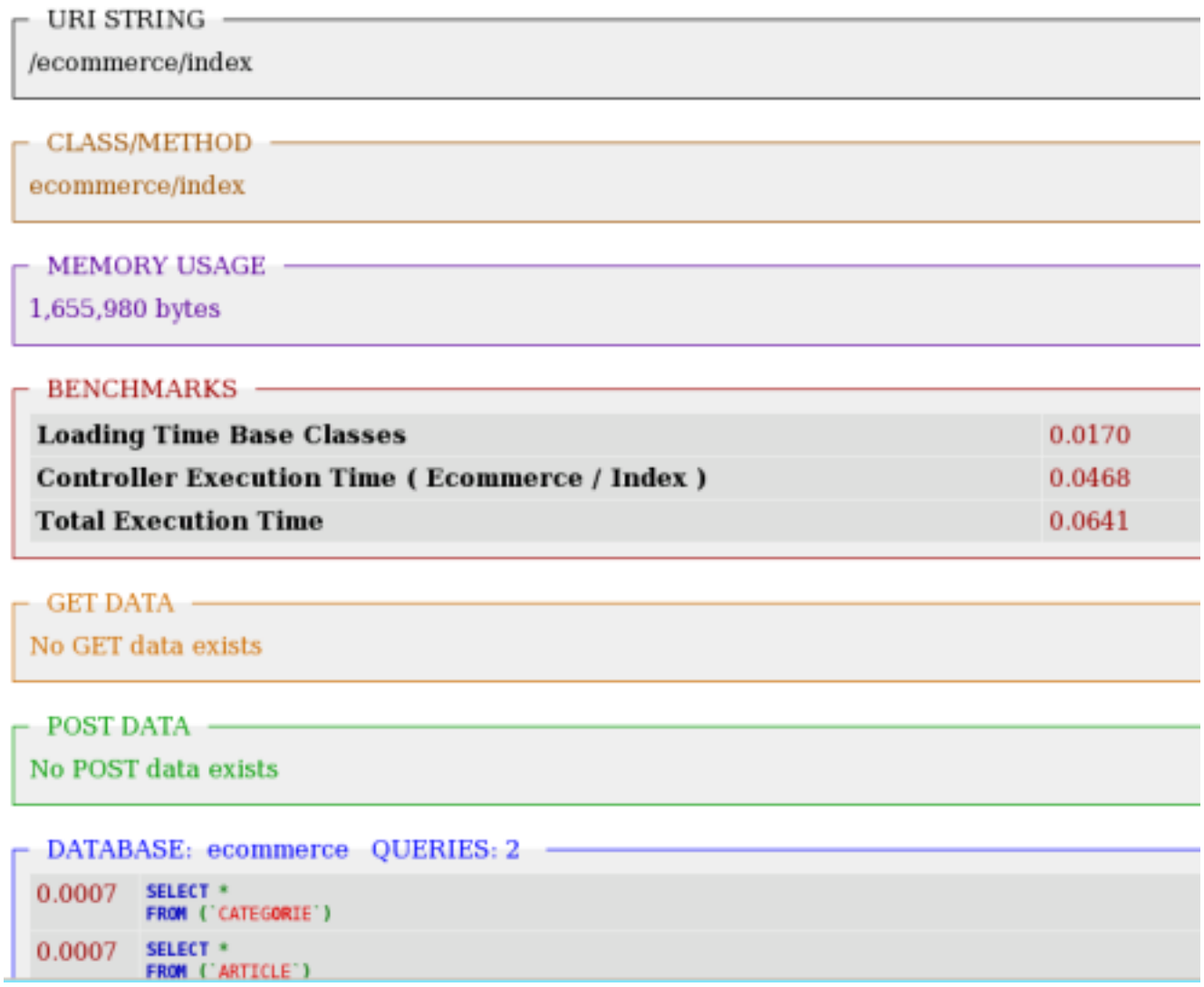
VII-A - Mesure des performances

Mesurer les performances grâce à Code Igniter est très simple.

Il suffit d'ajouter cette ligne dans votre contrôleur pour l'activer et la retirer pour la désactiver.

```
$this->output->enable_profiler(TRUE);
```

Dés que la ligne est ajoutée, des informations supplémentaires apparaissent en bas de page.



Les informations suivantes sont affichées afin de débugger facilement et trouver les source de contention de votre application.

- URL appelée.
- Méthode et classe PHP du contrôleur.
- Utilisation mémoire totale.
- Durée respective de chacune des étapes triées par ordre de parution dans le code.
- Les données GET passées en paramètre.

- Les données POST passées en paramètre.
- Les requêtes SQL envoyés vers les bases de données.

VII-B - Présentation de la librairie Benchmarking

La librairie Benchmark est chargé automatique par Code Igniter et il est donc inutile d'invoquer $this->load->library pour charger la librairie.

Pour ajouter une ligne de mesure spécifique pour un bloc de code donnée, il faut;:

Placer au début du bloc, $this->benchmark->mark('xxx_start') et $this->benchmark->mark('xxx_stop') à la fin du bloc.

Cela permettra d'ajouter une nesure de performance labélisé 'xxx'.

VII-B-1 - Exemple de code

```
$this->benchmark->mark('tag_start');> sleep(2);> $this->benchmark->mark('tag_end');>
$this->benchmark->mark('tag2_start');> sleep(1);> $this->benchmark->mark('tag2_end');
```

Grâce à ces ajouts de code, 2 nouvelles lignes apparaissent au niveau du tableau benchmarks indiquant les temps exécution pour le tag et tag2.

VII-C - Mise en place de cache

Afin d'améliorer les performances, il est possible pour certaines pages de mettre le résultat dans un cache. Dans ce cas, le résultat ne sera pas recalculé mais directement pris depuis un fichier dans le cache contenant le résultat de la page.

La librairie output nécessaire est chargée par défaut.

Pour activer cette fonctionnalité par page ou méthode de contrôleur, il faut 3 choses;:

- Le nombre de minute de persistance du cache.
- Les droits d'écriture sur le répertoire system/cache.
- L'appel explicite dans le contrôleur de la demande de mise en cache.

Si vous souhaitez mettre en place le cache de page pour une durée de $n secondes, voici la ligne a ajouté pour cela.

```
$this->output->cache($n);
```

VII-D - Réalisation des ses propres librairies

Il est possible d'écrire vos propres librairies.

Le framework Code Igniter vous permets trois choses;:

- Créer vos propres librairies.
- Etendre des librairies existantes.
- Remplacer des librairies existantes.

Vos librairies doivent être stockées dans le répertoire application/librairies

VII-D-1 - Création d'une librairie

Le fichier de la librairie doit commencé par une majuscule;: Soap.php

```
<?php
if ( ! defined('BASEPATH')) exit('No direct script access allowed');

class Soap {
 function recevoir_une_requete_soap() {...}
 function envoyer_une_requete_soap () {...}
}

?>
```

Rien de complexe et afin d'utiliser votre nouvelle librairie, le chargement s'effectue par programmation.

```
$this->load->library('soap');
$this->soap->envoyer_une_requete_soap();
```

VII-D-2 - Extension de librairie existante

Pour ceci, il suffit de trouver la thématique générale de votre librairie de surcharger la classe.

Pour connaître le préfixe de surcharge des librairies fondamentales, il faut consulter le paramètre suivant dans le fichier de configuration application/config/config.php

```
$config['subclass_prefix'] = 'MY_';
```

toutes classes avec un prefixe en MY_ sera considerer comme une extension des librairies fondamentales portent le même nom.

ici, il s'agit d'une extension de la classe CI_Mail de Code Igniter avec l'ajout d'une fonction annuler_dernier_email.

```
class MY_Email extends CI_Email {
 function My_Email() {
  parent::CI_Email();
 }
 function annuler_dernier_email(){
 }
}
```

Le chargement de la nouvelle librairie surchargé se fait de ma même manière sans supplément.

```
$this->load->library('email');
$this->email->annuler_dernier_email();
```

VIII - La sécurité

VIII-A - Fonctionnalités de sécurité de CodeIgniter

La fonction de filtrage de sécurité de Code Igniter est invoquée automatiquement à l'appel d'un contrôleur.

Cette fonction permets de:

- Suppression du tableau global $_GET car les données sont disponibles sous d'autre forme dans le contrôleur.
- Suppression de toutes les variables globales même avec le paramètre php.ini register_globals à ON.
- Filtrage des données des tableaux $_POST et $_COOKIE en ne permettant que les caractères alphanumériques.
- Fourniture de solution de protection XSS activable.
- Normalisation des retours à la ligne en \n.

VIII-B - Activation du filtrage de securité

Il existe 2 moyens d'activer la validation et la protection XSS des données passées au framework.

```
$data = $this->input->xss_clean($data);
```

La seconde technique consiste à ajouter le filtrage automaitque dans le ficheir de configuration principale : application/config/config.php

```
$config['global_xss_filtering'] = TRUE;
```

IX - Qualité du code

La qualité du code peut être attribué à de nombreux facteurs tel que;:

- Architecture du code.
- Règle de nommage et codage.
- Extensibilité des fonctionnalités.
- Capacité de tester unitairement chaque parie du code.
- Capacité à internationaliser l'application rapidement.

IX-A - Mise en place de tests unitaires pour votre site

Il est possible de permettre en place des tests unitaires et ainsi garantir une qualité de code mesurable.

Afin de charger la librairie des tests unitaires, il vous faut ajouter suivante dans votre contrôleur.

$this->load->library('unit_test');

IX-A-1 - Méthode de production de tests unitaires

- $this->unit->run(*"élément testé"*, "résultat attendue" , "nom du test");

La partie résultat attendue peut être substituée par des chaînes réservées à certains tests.

- is_string;: l'élément testé est une chaîne de caractère.
- is_bool: l'élément testé est un booléen.
- is_true: l'élément testé est VRAI.
- is_false: l'élément testé est FAUX.
- is_int: l'élément testé est un entier.
- is_numeric: l'élément testé est une valeur numérique.
- is_float: l'élément testé est une valeur flottante.
- is_double: l'élément testé est une valeur double.
- is_array: l'élément testé est un tableau PHP.
- is_null: l'élément testé est NULL.

IX-A-2 - Génération des rapports de tests

IX-A-3 - Désactivation des tests unitaires

IX-A-4 - Exemples de tests unitaires avec rapport formaté

```
function testTU() {
 $this->load->library('unit_test');
 $this->unit->run("ceci est une chaine", 'is_string', 'Test de chaine');
 $this->unit->run("ceci est une chaine", 'is_null', 'Test de nullite');
 $this->unit->run(5, 'is_int', 'Test de nombre entier');
 echo "<H1>Tests unitaires</H1>";
 echo $this->unit->report();
}
```

IX-A-5 - Rapport résultat

Tests unitaires

Test Name	Test de chaine
Test Datatype	String
Expected Datatype	String
Result	Passed
File Name	C:\wamp\www\ci_ecommerce\system\application\controllers\ecommerce.php
Line Number	36

Test Name	Test de nullite
Test Datatype	String
Expected Datatype	Null
Result	Failed
File Name	C:\wamp\www\ci_ecommerce\system\application\controllers\ecommerce.php
Line Number	37

Test Name	Test de nombre entier
Test Datatype	Integer
Expected Datatype	int
Result	Passed
File Name	C:\wamp\www\ci_ecommerce\system\application\controllers\ecommerce.php
Line Number	38

IX-A-6 - Exemples de tests unitaires avec rapport brut

```
function testTU() {
 $this->load->library('unit_test');
 $this->unit->run("ceci est une chaine", 'is_string', 'Test de chaine');
 $this->unit->run("ceci est une chaine", 'is_null', 'Test de nullite');
 $this->unit->run(5, 'is_int', 'Test de nombre entier');
 echo "<H1>Tests unitaires</H1>";
 echo "<pre>";
 print_r ($this->unit->result());
 echo "</pre>";
}
```

IX-A-7 - Résultat brut

```
Tests unitaires
Array
(
[0] => Array
(
[Test Name] => Test de chaine
[Test Datatype] => String
[Expected Datatype] => String
[Result] => Passed
[File Name] => C:\wamp\www\ci_ecommerce\system\application\controllers\ecommerce.php
[Line Number] => 36
)
[1] => Array
(
```

```
[Test Name] => Test de nullite
[Test Datatype] => String
[Expected Datatype] => Null
[Result] => Failed
[File Name] => C:\wamp\www\ci_ecommerce\system\application\controllers\ecommerce.php
[Line Number] => 37
)
[2] => Array
(
[Test Name] => Test de nombre entier
[Test Datatype] => Integer
[Expected Datatype] => int
[Result] => Passed
[File Name] => C:\wamp\www\ci_ecommerce\system\application\controllers\ecommerce.php
[Line Number] => 38
)
)
```

IX-B - Internationalisation de votre site

L'une des fonctionnalités les plus difficile et fastidieuse à rattraper dans une application est la mise en place de l'internationalisation de votre application.

L'internationalisation permet de définir une langue pour l'ensemble des messages de votre application aussi bien pour vos erreurs que pour vos titres et labels.

Code Igniter offre une classe utilitaire et une librairie permettant de réaliser rapidement cette tâche.

Initialiser dès le commencement d'un projet, il facile sa maintenance en évitant une passe ou 2 complète sur votre code afin de modifier les vues pour coller au principe de l'internationalisation de Code Igniter.

IX-B-1 - Fichier de traduction

Un fichier de traduction doit être placé dans le répertoire application/language/french/message_lang.php.

Dans le sous-répertoire application/language, il doit y avoir un répertoire par langue traduite.

Chaque fichier de traduction doit être nommé par xxxx_lang.php pour être pris en compte par le framework Code Igniter.

IX-B-2 - Exemple de fichier de traduction

Application/language/french/message_lang.php :

```
<?php
 $lang['title']='Titre pour l\'internationnalisation de Code Igniter';
 $lang['application_name']='Le produit Code Igniter';
?>
```

Application/language/english/message_lang.php :

```
<?php
 $lang['title']='Title for Code Igniter internationalisation';
 $lang['application_name']='the Framework Code Igniter';
?>
```

IX-B-3 - Méthode d'utilisation de la classe utilitaire Language

- lang('nom de l element', 'identifiant du formulaire')
- lang('nom de l element')

Elle permette de raccourcir la syntaxe qui ressemble plus à :

$this->lang->line('page_title')

Pour utiliser, ce raccourci de syntaxe, il faut charger soit dans application/config/autoload.php language dans la partie helper ou ajouter la ligne suivante au début du contrôleur.

$this->load->helper('language')

Le choix du langage se fait par appel à la méthode de chargement du bon fichier de traduction;:

$this->lang->load('message', 'french')

IX-B-4 - Utilisation de l'internationalisation en français

```
function testInt() {
 $this->load->helper('language');
 $this->lang->load('message', 'french');
 $this->load->view('entete', array('titre'=> lang('title')));
 $this->load->view('affichage_brut', array( 'titre' => lang('title'), 'contenu'=>
 lang('application_name')) );
 $this->load->view('pied');  $this->load->helper('language');
}
```

IX-B-5 - Utilisation de l'internationalisation en français

```
function testInt() {
 $this->load->helper('language');
 $this->lang->load('message', 'english');
 $this->load->view('entete', array('titre'=> lang('title') ) );
 $this->load->view('affichage_brut', array( 'titre' => lang('title'), 'contenu'=>
 lang('application_name') ) );
 $this->load->view('pied');
}
```

X - Aide et support

Vous trouverez de nombreuse information en français: sur les sites :

- **http://codeigniter.fr/**
- **http://codeigniter.com/**

Vous pouvez trouver de l'aide sur le forum anglophone de Code Igniter à l'adresse suivante;:

- **http://codeigniter.com/forums/**

Le forum francophone se trouve à l'adresse suivante:

- **http://codeigniter.fr/cms/forums**

La documentation de référence **http://codeigniter.com/user_guide** est entièrement traduite à l'adresse suivante:

- **http://www.codeigniter.fr/user_guide/**

L'une des sources les plus complètes sont le wiki: **http://codeigniter.com/wiki** et le répertoire Code Igniter ou vous trouverez de nombreuse référence vers des sites relatifs à Code Igniter.

- **http://codeigniterdirectory.com/**

XI - Liens de l'article

Formation et technologie du Web L'article au format PDF