Modèle-Vue-Contrôleur (introduction)

Un exemple web avec PHP

monnerat@u-pec.fr

4 mai 2026

IUT de Fontainebleau

Sommaire

Introduction

Introduction

Le principe

Modèle MVC

Principe de conception d'applications (WEB ou pas) qui sépare les fonctions nécessaires en trois composants distincts :

Modèle	$\rightsquigarrow$	Gère les données
Vue	$\rightsquigarrow$	Gère la présentation (UI)
Contrôleur	$\rightsquigarrow$	Agit

Il existe de nombreux framework PHP basés sur cette architecture.

Introduction

Le modèle

Le modèle

Gestion des données

- Gére les échanges avec la base de données.
- Abstrait sous forme de classes les entités (les tables) de la bd.
- On peut utiliser un ORM (object-relational mapping) : doctrine, pdoMap, Propel, etc...

Introduction

La Vue

La vue

Gestion de la présentation des données

- Affichage `HTML` essentiellement (mais pas uniquement).
- Avec des parties utilisant des variables php, calculées par le contrôleur à l'aide du Modèle (détail d'un film, etc...).
- Utilisation possible de templates.

Introduction

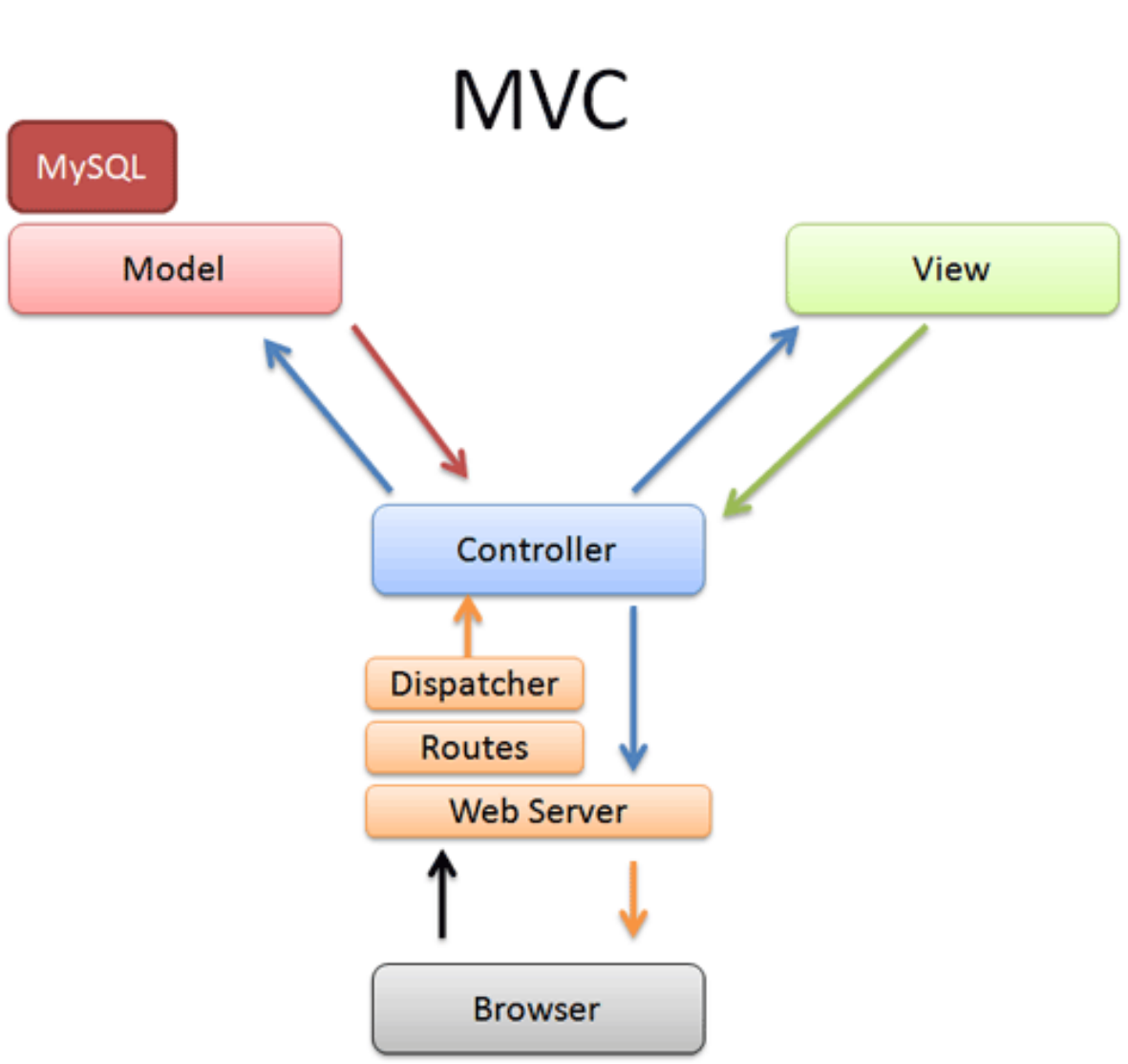
Le contrôleur

Le contrôleur

Gére les requêtes des utilisateurs, retourne une réponse avec l'aide mutuelle des couches Modèle et Vue.

- Le "cerveau" de l'application.
- En fonction des requêtes du client, il fournit la bonne réponse en piochant dans le modèle et la vue pour afficher le bon résultat à l'utilisateur.
- Autant de classes que nécessaires, regroupés en entités logiques (Films, Uilisateurs, etc...)
- Chaque contrôleur posséde des actions : lister, insérer, supprimer, etc...
- Chaque contrôleur a ses vues associées.

MVC WEB


MVC
MySQL
Model
View
Controller
Dispatcher
Routes
Web Server
Browser

Avantages

Modèle et Vue sont séparés

- ⇝ L'application est plus flexible, plus claire, facilement évolutive.
- ⇝ On peut les changer, remplacer, tester séparément.
- ⇝ Le "design" peut-être confié à un désigner.
- ⇝ Abstraction de l'application par rapport aux données.

CodeIgniter

CodeIgniter

CodeIgniter est un framework PHP MVC, très simple de prise en main et d'utilisation.

Site de CodeIgniter

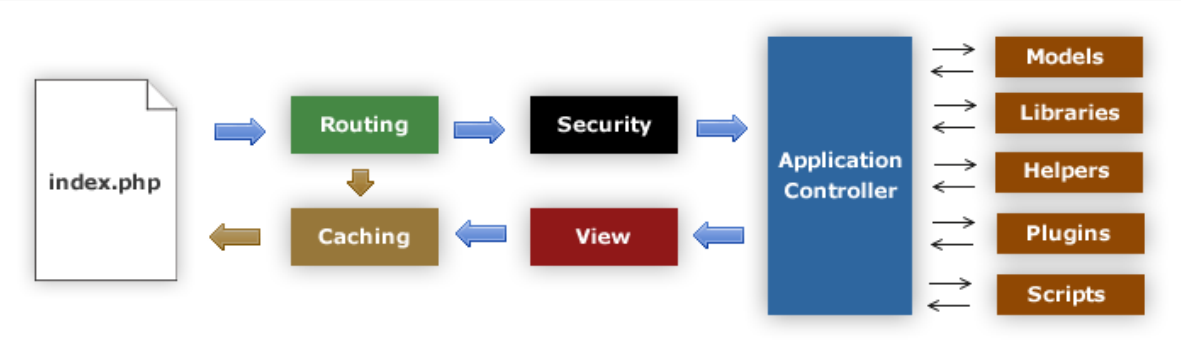
Dernière version stable : version 4.x.x

`http://www.codeigniter.com`

- Instanciation dynamique des ressources.
- Indépendance des composants

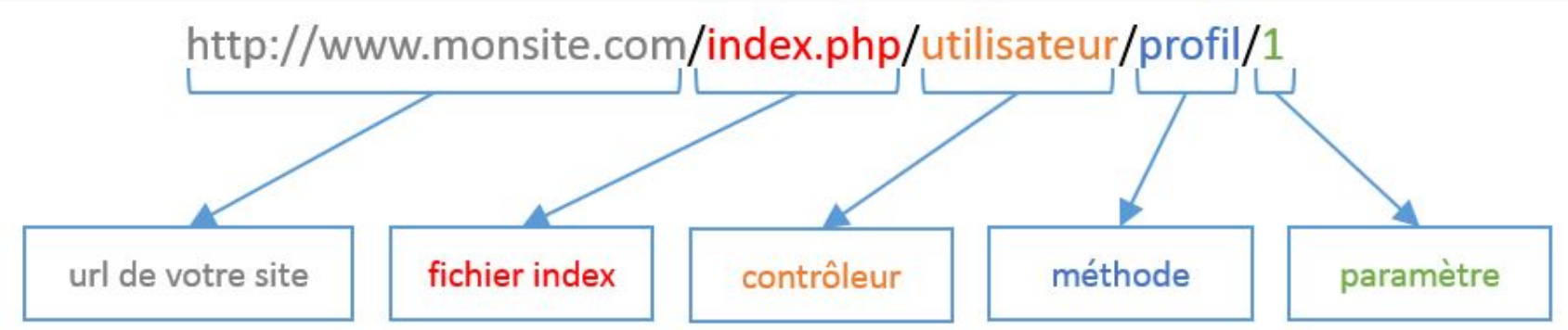
La suite concerne la version 3 de CI

Le fonctionnement général



1. index.php est le seul point d'entrée, initialise les ressources de bases utilisées par CI.
2. Le Routeur examine la requête HTTP pour savoir quoi faire.
3. Si un cache existe, il est renvoyé sans passer le système d'éxecution normal.
4. Securité. Avant l'instanciation du bon contrôleur, la requête HTTP et toutes les informations soumises par l'utilisateur sont filtrés.
5. le contrôleur charge le modèle, les librairies, les helpers, et toutes les ressources nécessaires pour la réponse.
6. La vue finale est calculée et envoyée. Si le cache est activé, la vue est mise en cache pour des requêtes futures.

Les urls



On peut supprimer `index.php` en utilisant un mécanisme de réécriture d'url du serveur `http`.

Installation

Téléchargez CI (version 3.x), et désarchivez dans un répertoire servi par un serveur http. Vous obtenez alors l'arborescence suivante :

- `application`
  Dossier où est codé l'application elle-même.
- `system`
  C'est le dossier qui contient le cœur de CodeIgniter. Ne pas modifier son contenu !
- `user_guide`
  la même doc que celle disponible en ligne. Vous pouvez supprimer ce dossier.
- `index.php`
  c'est le fichier que l'internaute appellera lorsqu'il se connectera à votre site. Il est très important, à ne pas toucher.

Dossier application

C'est le dossier où est codé l'application. Voici son découpage avec les plus importants (pour le moment) :

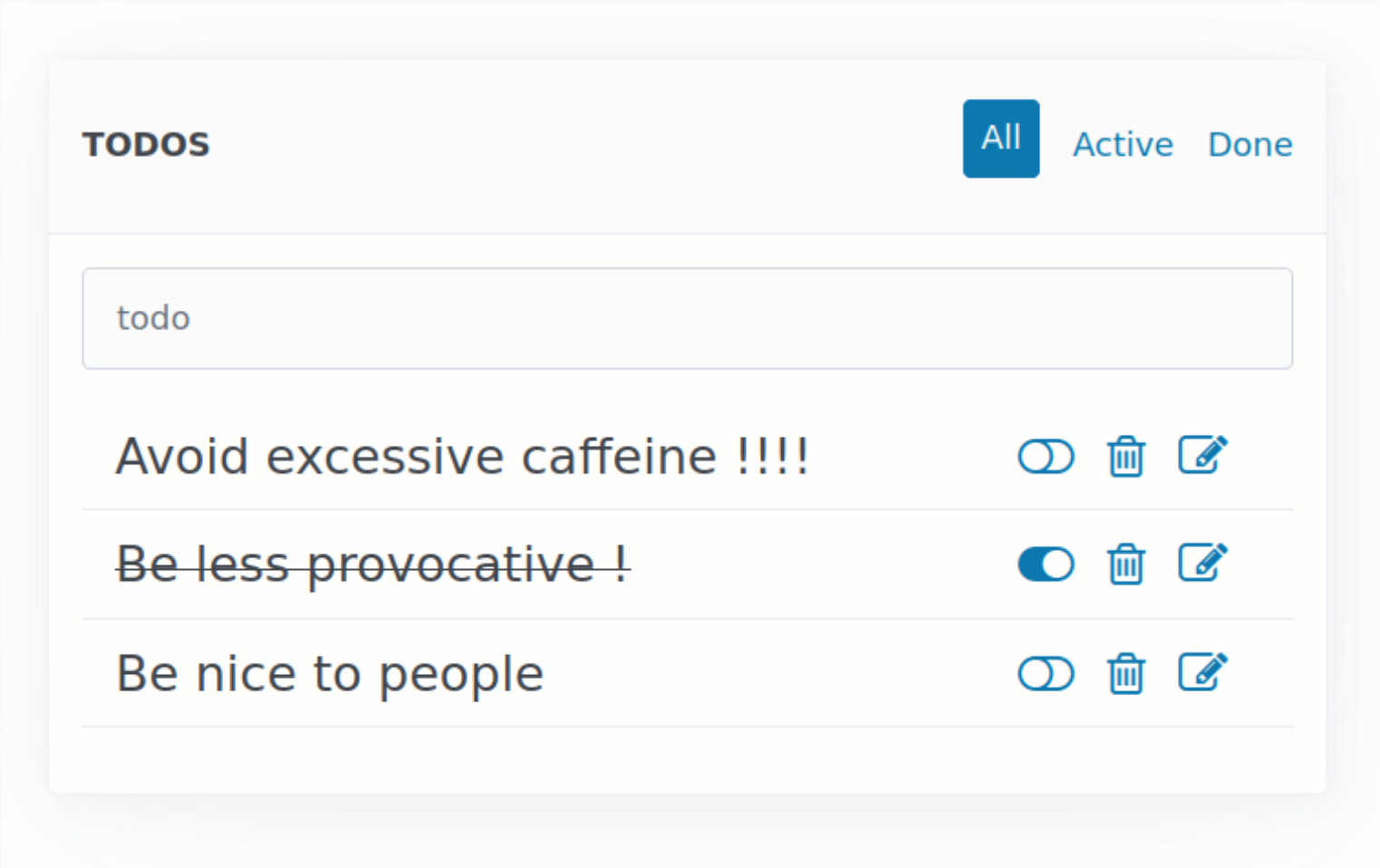
- `config`
  ce dossier contient la configuration du site.
- `controllers`
  les contrôleurs.
- `helpers`
  les helpers (fonction d'aides)
- `librairies`
  bibliothèques utilisables par votre application.
- `models`
  les modèles.
- `views`
  les vues.

Configuration

Le répertoire `config` contient :

- `autoload.php`
  helpers et librairies chargés par défaut
- `config.php`
  configuration de bases du serveur (url, protocoles, etc.)
- `database.php`
  Accès bases de données.

Exemple avec la todolist (cf TP)



Configuration

URL de base dans `config.php`

```php
/*
 * |--------------------------------------------------------------------------
 * | Base Site URL
 * |--------------------------------------------------------------------------
 * |
 * | URL to your CodeIgniter root. Typically this will be your base URL,
 * | WITH a trailing slash:
 * |
 * |slashhttp://example.com/
 * |
 * | WARNING: You MUST set this value!
 * |
 * | If it is not set, then CodeIgniter will try guess the protocol and path
 * | your installation, but due to security concerns the hostname will be set
 * | to $_SERVER['SERVER_ADDR'] if available, or localhost otherwise.
 * | The auto-detection mechanism exists only for convenience during
 * | development and MUST NOT be used in production!
 * |
 * | If you need to allow multiple domains, remember that this file is still
 * | a PHP script and you can easily do that on your own.
 * |
 * */
$config['base_url'] = '/~denis/todo/';
```

Configuration

Accès bases de données dans `database.php`

```php
$db['default'] = array(
  'dsn'=> '',
  'hostname' => 'localhost',
  'username' => 'test',
  'password' => 'test',
  'database' => 'todos',
  'dbdriver' => 'mysqli',
  'dbprefix' => '',
  'pconnect' => FALSE,
  'db_debug' => (ENVIRONMENT !== 'production'),
  'cache_on' => FALSE,
  'cachedir' => '',
  'char_set' => 'utf8',
  'dbcollat' => 'utf8_general_ci',
  'swap_pre' => '',
  'encrypt' => FALSE,
  'compress' => FALSE,
  'stricton' => FALSE,
  'failover' => array(),
  'save_queries' => TRUE
);
```

Configuration

Chargement automatique dans `autoload.php`

```php
/*
| -------------------------------------------------------------------
|  Auto-load Helper Files
| -------------------------------------------------------------------
| Prototype:
|
|  $autoload['helper'] = array('url', 'file');
*/
$autoload['helper'] = array('url','html','download');
```

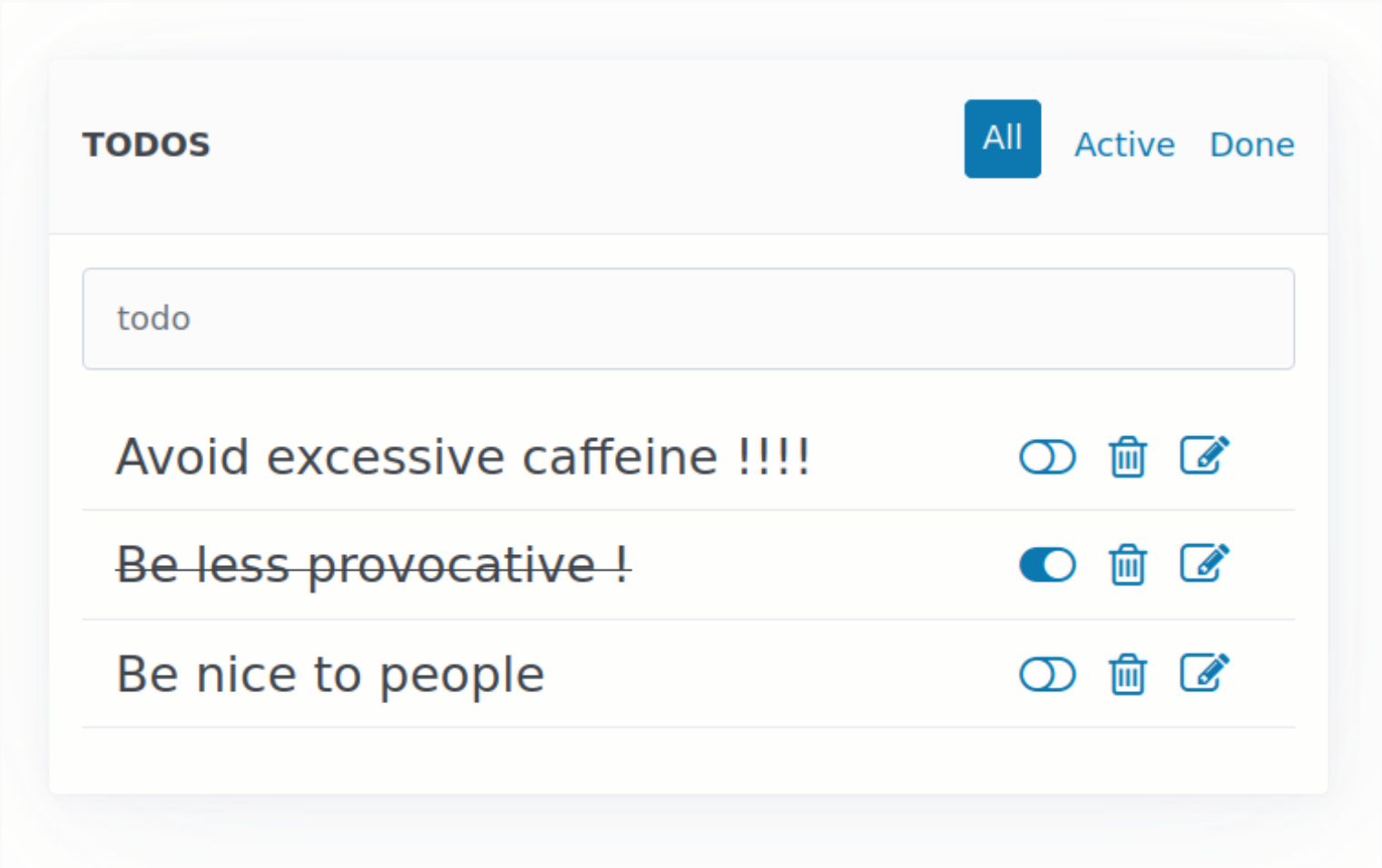
Configuration

Routes dans `routes.php`

```php
$route['default_controller'] = 'todo';
```

Le routage est implicite (mapping url-contrôleurs), mais on peut ajouter des routes manuellement ici.

Le code pour obtenir cette vue ?



Le modèle

```php
<?php
class Model_todo extends CI_Model {
  public function __construct()
  {
    $this->load->database();
  }
  public function getTodos($filter='all')
  {
    $where_filter = ["done" => 1, "active" => 0, "all" => "%"];
    return $this->db->query(
      "SELECT * FROM todo WHERE done LIKE ?",
      [$where_filter[$filter]]
    )->result();
  }

}
```

Le contrôleur

- Chargement du modèle.
- Récupération du tableau de tous les todos.
- Chargement des vues, en passant le tableau précedent.

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

class Todo extends CI_Controller {
  public $filter = 'all';
  public function __construct(){
    parent::__construct();
    $this->filter = $this->input->get('filter') ?? 'all';

  }
  public function index(){
    $this->load->model('model_todo');
    $todos = $this->model_todo->getTodos($this->filter);
    $this->load->view('layout/header');
    $this->load->view('todos',['todos'=>$todos,'filter'=>$this->filter]);
    $this->load->view('layout/footer');
  }
}
```

Les vues

L'entête

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>TODO APP</title>
    <link
      rel="stylesheet"
      href="https://cdn.jsdelivr.net/npm/@picocss/pico@2/css/pico.min
    />
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font
    <?=link_tag('assets/style.css')?>
  </head>
  <body>
    <main class="container">
```

Les vues

Le tableau des contacts est passé à la vue par le contrôleur.

```
<table>
<?php
foreach($todos as $todo){
  if ($todo->done){
    echo "<tr><td><s>{$todo->text}</s></td>";
    echo "<td class='action'>";
    echo anchor("todo/toggle/{$todo->id}", '<i class="fa fa-toggle-on"></i>');
    echo anchor("todo/delete/{$todo->id}", '<i class="fa fa-trash-o"></i>');
    echo anchor("todo/edit/{$todo->id}", '<i class="fa fa-edit"></i>');
  } else {
    echo "<tr><td>{$todo->text}</td>";
    echo "<td class='action'>";
    echo anchor("todo/toggle/{$todo->id}", '<i class="fa fa-toggle-off"></i>');
    echo anchor("todo/delete/{$todo->id}", '<i class="fa fa-trash-o"></i>');
    echo anchor("todo/edit/{$todo->id}", '<i class="fa fa-edit"></i>');
  }
  echo "</td></tr>";
}
?>
</table>
```

Syntaxe alternative

```
<table>
<?php foreach($todos as $todo): ?>
  <?php if ($todo->done): ?>
    <tr>
    <td><s><?=$todo->text?></s></td>
    <td class='action'>
    <?=anchor("todo/toggle/{$todo->id}", '<i class="fa fa-toggle-on"></i>')?>
    <?=anchor("todo/delete/{$todo->id}", '<i class="fa fa-trash-o"></i>')?>
    <?=anchor("todo/edit/{$todo->id}", '<i class="fa fa-edit"></i>')?>

  <?php else: ?>

    <tr><td><?=$todo->text?></td>
    <td class='action'>
    <?=anchor("todo/toggle/{$todo->id}", '<i class="fa fa-toggle-off"></i>')?>
    <?=anchor("todo/delete/{$todo->id}", '<i class="fa fa-trash-o"></i>')?>
    <?=anchor("todo/edit/{$todo->id}", '<i class="fa fa-edit"></i>')?>
  <?php endif; ?>
  </td></tr>
<?php endforeach; ?>
</table>
```

Les helpers

Chargement

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

ou dans le fichier de config `application/config/autoload`

- URL helper : contructution d'urls relatives au site.
- HTML helper : génération de code html.
- SECURITY helper : fonctions liées à la sécurité.
- etc.

```php
<?=anchor('news/local/123', 'My News', 'title="News title"');?>
// Prints:
// <a href="http://example.com/index.php/news/local/123" title="News title">My News</a>
```

Librairies

CI vient avec des librairies stockées dans `systemlibraries`.

```php
$this->load->library('class_name');
```

- Form Validation : validation de formulaires.
- Input Class : entrées (requêtes, cookies, etc.) et filtrage automatique.
- Session Library : gestion des sessions.
- File Uploading Class : upload de fichiers.
- Security Class : sanitization des données, CSRF protection.
- etc.

Remaplacement/Extension de classes systèmes

Le repertoire core de votre application permet de remplacer ou d'étendre les classes du système : `Controller`, `Router`, etc.

- Utilisation de sa propre classe au lieu de celle du système :

```
class CI_Controller {

}
```

  à placer dans le fichier `application/core/Controller.php`

- Extension d'une classe du système :

```
class MY_Controller extends CI_Controller {

}
```

  à placer dans le fichier `application/core/MY_Controller.php`

On peut alors utiliser ses propres classes :

```
class Welcome extends MY_Controller {
  public function index()
  {
    $this->load->view('welcome_message');
  }
}
```

Le prefixe `MY_` est configurable dans le fichier de configuration `application/config/config.php`

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