

# CRUD avec CodeIgniter 2.0

version 1.0: 2011-09-12

author: Maxime Keltsma (mailto: [smaltek62@gmail.com](mailto:smaltek62@gmail.com))

## Index

Pré requis.....	2
Ce que j'ai utilisé.....	3
Premier test d'URL.....	4
Base de données.....	4
Configuration de l'application .....	6
Dans construction_sites/application/config/config.php:.....	6
Dans construction_sites/application/config/database.php:.....	6
Dans construction_sites/application/config/routes.php:.....	7
Spécifications de project.....	8
Diagramme de classes.....	9
Modèles.....	10
Règles de validation du formulaire.....	10
Code source.....	13
Modèles.....	13
Vues.....	16
Controleur "Site".....	20
CSS.....	32
Divers.....	37
Paramètres.....	37
Commentaires du code source.....	37
Le grand secret.....	37
Conclusion.....	46

Dans ce tutoriel, nous allons construire un CRUD basique (Create Read Update Delete) en utilisant le framework CodeIgniter 2.0

Pour cela nous créerons une petite table contenant des chantiers de construction.

Le résultat final devrait ressembler à ceci:

### Construction sites (CRUD with CodeIgniter 2.0)

LINE	ID	NAME	COST €	PROGRESS %	TYPE	IS STARTED	IS SUSPENDED	ACTIONS
19	318	Royal hotel	4200000	10	PRIVATE	1	0	
20	320	278 republic av	1300000	100	PRIVATE	1	0	
21	322	1206 sanctuary blv	2150000	100	PRIVATE	1	0	
22	324	1998 Michel Jourdan av	3150000	90	PRIVATE	1	0	

< FIRST < 2 3 4

add new record

Les champs qui composent un chantier de construction, vont nous permettre de manipuler:

- Une chaîne de caractères pour le nom.
- Une valeur entière pour le cout du chantier.
- Une liste de choix pour le pourcentage d'avancement du chantier.
- Un bouton radion pour le type du chantier (privé ou public).
- Deux cases à cocher pour les attributs `isStarted` et `isSuspended`.

Le formulaire permettant de créer/modifier un chantier, ressemblera à cela:

## Update a record

ID	324
Name*	1998 Michel Jourdan av
Cost € *	3150000
Progress %	90 ▾
Type*	<input type="radio"/> Public <input checked="" type="radio"/> Private
Is started	<input checked="" type="checkbox"/>
Is suspended	<input type="checkbox"/>
<b>Save</b>	

[◀ Back](#)

Notes:

J'ai rédigé ce tuto en anglais puis je l'ai traduit en français.

J'ai laissé les commentaires dans les codes sources, en anglais.

## Pré requis

Je suppose que:

- Vous disposez d'un serveur de base de données et d'un serveur http (j'ai utilisé Wamp).
- Vous avez installé CodeIgniter 2.0.3 ou mieux, pour un projet nommé "construction\_sites".
- Vous avez au moins des notions élémentaires en PHP 5 et en SQL.
- Vous êtes assez fou pour développer en PHP le week end ;-))

## Ce que j'ai utilisé

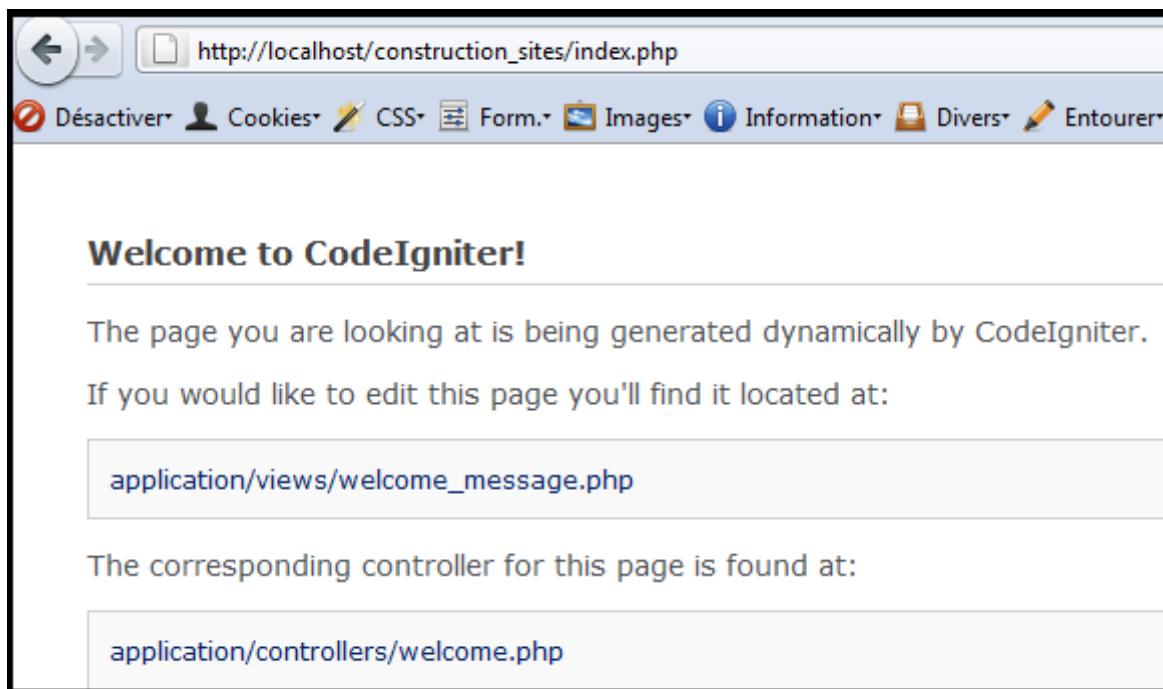
- Windows 7
- Wamp 2.0: <http://www.wampserver.com/> (Mysql 5.1; PHP 5.3.0, Apache 2.2.11)
- CodeIgniter 2.0.3: <http://codeigniter.com/>
- Pour le travail en SQL, j'ai utilisé mon vieux Mysql Query Browser, mais vous pouvez utiliser phpMyAdmin fourni avec Wamp, ou le nouvel outil Mysql Workbench.
- Bouml pour le diagramme de classes: <http://bouml.free.fr>
- Netbeans 6.9: <http://netbeans.org/>
- Open Office: <http://www.openoffice.org/> pour taper ce tutoriel.

## Premier test d'URL

Si ce n'est pas encore fait, créez une nouvelle installation de CodeIgniter pour un projet nommé "construction\_sites".

En utilisant l'URL ci-dessous, vous devriez obtenir l'écran d'accueil de CodeIgniter.

[http://localhost/construction\\_sites/index.php](http://localhost/construction_sites/index.php)



## Base de données

Avec votre outil SQL, exécutez les scripts suivants pour créer et peupler notre petite base de données.

```
CREATE DATABASE `construction_site`;
```

```
CREATE TABLE IF NOT EXISTS `construction_site`.`progress` (
  `id` smallint(5) unsigned NOT NULL DEFAULT '0',
  `progress` smallint(5) unsigned NOT NULL DEFAULT '0' COMMENT 'progression value (percentage)',
  PRIMARY KEY (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COMMENT='progression values';
```

```
CREATE TABLE IF NOT EXISTS `construction_site`.`site` (
  `id` INT(10) UNSIGNED NOT NULL AUTO_INCREMENT ,
```

```

`name` VARCHAR(45) NOT NULL DEFAULT " ,
`cost` INT(10) UNSIGNED NOT NULL DEFAULT '0' ,
`progress` SMALLINT(5) UNSIGNED NOT NULL DEFAULT '0' ,
`type` VARCHAR(45) NOT NULL DEFAULT 'PUBLIC' ,
`isstarted` TINYINT(3) UNSIGNED NOT NULL DEFAULT '0' ,
`issuspended` TINYINT(3) UNSIGNED NOT NULL DEFAULT '0' ,
PRIMARY KEY (`id`)
) ENGINE = MyISAM
AUTO_INCREMENT = 100
DEFAULT CHARACTER SET = utf8;

```

```
INSERT INTO `construction_site`.`site` (`id`, `name`, `cost`, `progress`, `type`, `isstarted`, `issuspended`) VALUES (NULL, 'building A1', 3000000, 40, 'PUBLIC', 1, 0);
```

```
INSERT INTO `construction_site`.`site` (`id`, `name`, `cost`, `progress`, `type`, `isstarted`, `issuspended`) VALUES (NULL, 'building A2', 3000000, 10, 'PUBLIC', 1, 0);
```

```
INSERT INTO `construction_site`.`site` (`id`, `name`, `cost`, `progress`, `type`, `isstarted`, `issuspended`) VALUES (NULL, 'building F1', 2800000, 0, 'PRIVATE', 0, 0);
```

```
INSERT INTO `construction_site`.`site` (`id`, `name`, `cost`, `progress`, `type`, `isstarted`, `issuspended`) VALUES (NULL, 'building F2', 2800000, 0, 'PRIVATE', 0, 0);
```

```
INSERT INTO `construction_site`.`site` (`id`, `name`, `cost`, `progress`, `type`, `isstarted`, `issuspended`) VALUES (NULL, 'central museum', 3250000, 60, 'PUBLIC', 1, 0);
```

```
INSERT INTO `construction_site`.`site` (`id`, `name`, `cost`, `progress`, `type`, `isstarted`, `issuspended`) VALUES (NULL, 'Victory bridge', 4500000, 50, 'PUBLIC', 0, 0);
```

```
INSERT INTO `construction_site`.`site` (`id`, `name`, `cost`, `progress`, `type`, `isstarted`, `issuspended`) VALUES (NULL, 'state academy', 5300000, 80, 'PUBLIC', 1, 1);
```

```
INSERT INTO `construction_site`.`site` (`id`, `name`, `cost`, `progress`, `type`, `isstarted`, `issuspended`) VALUES (NULL, '12 st georges blv.', 1230000, 30, 'PRIVATE', 1, 0);
```

```
INSERT INTO `construction_site`.`site` (`id`, `name`, `cost`, `progress`, `type`, `isstarted`, `issuspended`) VALUES (NULL, 'Components Factory', 2800000, 0, 'PRIVATE', 0, 0);
```

```
INSERT INTO `construction_site`.`site` (`id`, `name`, `cost`, `progress`, `type`, `isstarted`, `issuspended`) VALUES (NULL, 'Eiffel tour', 2980000, 100, 'PUBLIC', 1, 0);
```

```
INSERT INTO `construction_site`.`site` (`id`, `name`, `cost`, `progress`, `type`, `isstarted`, `issuspended`) VALUES (NULL, 'Redemption church', 970000, 30, 'PRIVATE', 1, 0);
```

```
INSERT INTO `construction_site`.`site` (`id`, `name`, `cost`, `progress`, `type`, `isstarted`, `issuspended`) VALUES (NULL, 'Liberty Hospital', 4150000, 0, 'PUBLIC', 0, 0);
```

```
INSERT INTO `construction_site`.`site` (`id`, `name`, `cost`, `progress`, `type`, `isstarted`, `issuspended`) VALUES (NULL, 'Health care center', 2740000, 20, 'PUBLIC', 1, 0);
```

```
INSERT INTO `construction_site`.`site` (`id`, `name`, `cost`, `progress`, `type`, `isstarted`, `issuspended`) VALUES (NULL, '345 Trinity av', 2100000, 30, 'PRIVATE', 1, 0);
```

```
INSERT INTO `construction_site`.`site` (`id`, `name`, `cost`, `progress`, `type`, `isstarted`, `issuspended`) VALUES (NULL, 'Surcouf Residence', 2300000, 50, 'PRIVATE', 1, 1);
```

```
INSERT INTO `construction_site`.`site` (`id`, `name`, `cost`, `progress`, `type`, `isstarted`, `issuspended`) VALUES (NULL, 'The Legend residence', 5580000, 60, 'PRIVATE', 1, 0);
```

```
INSERT INTO `construction_site`.`site` (`id`, `name`, `cost`, `progress`, `type`, `isstarted`, `issuspended`) VALUES (NULL, 'St Barth theater', 2100000, 70, 'PRIVATE', 1, 0);
```

```
INSERT INTO `construction_site`.`site` (`id`, `name`, `cost`, `progress`, `type`, `isstarted`, `issuspended`) VALUES (NULL, 'Regency hotel', 3145000, 50, 'PRIVATE', 1, 0);
```

```
INSERT INTO `construction_site`.`site` (`id`, `name`, `cost`, `progress`, `type`, `isstarted`, `issuspended`) VALUES (NULL, 'Royal hotel', 4200000, 10, 'PRIVATE', 1, 0);
```

```
INSERT INTO `construction_site`.`site` (`id`, `name`, `cost`, `progress`, `type`, `isstarted`, `issuspended`) VALUES (NULL, '278 republic av', 1300000, 100, 'PRIVATE', 1, 0);
```

```
INSERT INTO `construction_site`.`site` (`id`, `name`, `cost`, `progress`, `type`, `isstarted`, `issuspended`) VALUES (NULL,
```

```
'1206 sanctuary blv', 2150000, 100, 'PRIVATE', 1, 0);
INSERT INTO `construction_site`.`site` (`id`, `name`, `cost`, `progress`, `type`, `isstarted`, `issuspended`) VALUES (NULL,
'1998 Michel Jourdan av', 3150000, 90, 'PRIVATE', 1, 0);

INSERT INTO `construction_site`.`progress` (`id`, `progress`) VALUES (0, 0);
INSERT INTO `construction_site`.`progress` (`id`, `progress`) VALUES (10, 10);
INSERT INTO `construction_site`.`progress` (`id`, `progress`) VALUES (20, 20);
INSERT INTO `construction_site`.`progress` (`id`, `progress`) VALUES (30, 30);
INSERT INTO `construction_site`.`progress` (`id`, `progress`) VALUES (40, 40);
INSERT INTO `construction_site`.`progress` (`id`, `progress`) VALUES (50, 50);
INSERT INTO `construction_site`.`progress` (`id`, `progress`) VALUES (60, 60);
INSERT INTO `construction_site`.`progress` (`id`, `progress`) VALUES (70, 70);
INSERT INTO `construction_site`.`progress` (`id`, `progress`) VALUES (80, 80);
INSERT INTO `construction_site`.`progress` (`id`, `progress`) VALUES (90, 90);
INSERT INTO `construction_site`.`progress` (`id`, `progress`) VALUES (100, 100);
```

## Configuration de l'application

Modifier les fichiers de configuration de CodeIgniter, selon les indications ci-dessous:

### **Dans construction\_sites/application/config/config.php:**

```
$config['base_url'] = 'http://localhost/construction_sites/';
```

### **Dans construction\_sites/application/config/database.php:**

```
$active_group = 'default';
$active_record = TRUE;

$db['default']['hostname'] = 'localhost';
$db['default']['username'] = 'maxime';
$db['default']['password'] = 'abracadabra';
$db['default']['database'] = 'construction_site';
$db['default']['dbdriver'] = 'mysql';
$db['default']['dbprefix'] = "";
$db['default']['pconnect'] = TRUE;
$db['default']['db_debug'] = TRUE;
$db['default']['cache_on'] = FALSE;
```

```
$db['default']['cachedir'] = "";
$db['default']['char_set'] = 'utf8';
$db['default']['dbcollat'] = 'utf8_general_ci';
$db['default']['swap_pre'] = "";
$db['default']['autoinit'] = TRUE;
$db['default']['stricton'] = FALSE;
```

***Dans construction\_sites/application/config/routes.php:***

```
$route['default_controller'] = "site";
$route['404_override'] = "";
```

# Spécifications de project

Voici les actions et les méthodes que nous allons implémenter dans notre contrôleur "site.php":

User action	Controller method	View
Get the records list	index()	siteList_tpl.php
Get the record detail	getDetail()	siteDetail_tpl.php
Update a record	getUpdate()	siteEdit_frm.php
Update a record	submitUpdate()	siteEdit_frm.php
Create a record	getAdd()	siteEdit_frm.php
Create a record	submitAdd()	siteEdit_frm.php
Delete a record	delete()	-

Note:

- Le même formulaire sera utilisé pour les actions de création et de modifications. Ceci nous complique un peu les choses, mais il vaut mieux avoir un seul formulaire par entité (par table).
- J'ai nommé les vues avec \_tpl pour les templates html simples, et \_frm pour les formulaires.

Nous utiliserons:

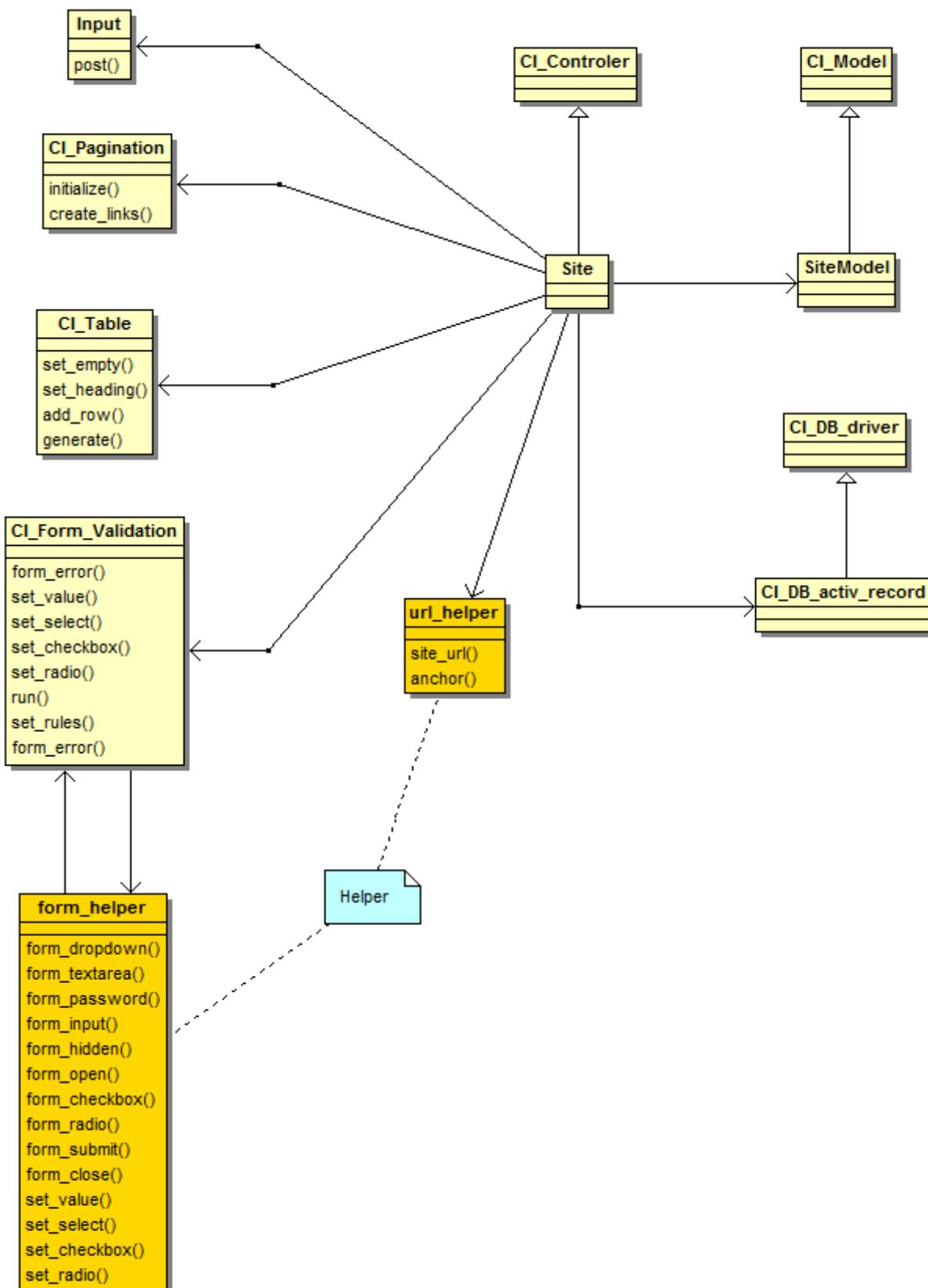
- La librairie Table de CodeIgniter, pour générer le code html du tableau de la liste du CRUD.
- La librairie Form\_Validation de CodeIgniter, pour contrôler les données saisies par l'utilisateur.
- La librairie Pagination de CodeIgniter, pour gérer les différentes pages du CRUD, et générer la barre de navigation.
- Le helper URL (assistant) de CodeIgniter.

Voici les fichiers que nous devons implémenter:

Type	File name
Controller	site.php
Model	progressModel.php
Model	siteModel.php
View	siteDetail_tpl.php
View	siteList_tpl.php
View	siteEdit_frm.php

## Diagramme de classes

Ce diagramme est certainement incomplet mais il correspond à ce que j'ai constaté jusqu'à présent dans CodeIgniter.



Nous ne gèrerons pas:

- l'authentification.
- Les logs.

## Modèles

Pour la classe du modèle de la table "site", nous utiliserons la technique Activ Record (voir Activ Record dans la documentation de CodeIgniter) pour les raisons suivantes:

- La construction des requêtes est plus facile.
- On peut changer de serveur de base de données, pratiquement sans toucher au code.

Pour la table Progress, qui est une petite table référentielle, et pour voir une autre technique, nous utiliserons les requêtes SQL classiques. Il faut aussi savoir que le fait de ne pas utiliser Activ Record permet d'économiser des ressources en mémoire centrale.

## Règles de validation du formulaire

Les règles de validation permettent:

- Contrôler sécuriser et préparer les données saisies par l'utilisateur.
- Afficher un message d'erreur spécifique lorsqu'une règle n'est pas définie.

CodeIgniter autorise plusieurs approches pour gérer les règles de validation, comme vous pouvez le voir dans la documentation de CodeIgniter.

Pour ce tutoriel, j'ai choisi de définir les règles dans un fichier externe, et de les regrouper par action.

Créez le fichier: [construction\\_sites/application/config/form\\_validation.php](#) et copiez-y le code ci-dessous:

```
<?php  
$config = array(  
    'site/submitUpdate' => array(  
        array(  
            'field' => 'id',  
            'label' => 'ID',  
            'rules' => 'trim|required|integer'
```

```

),
array(
    'field' => 'name',
    'label' => 'Nom',
    'rules' => 'trim|required|strToUpper'
),
array(
    'field' => 'cost',
    'label' => 'Cost',
    'rules' => 'trim|required|integer'
),
array(
    'field' => 'progress',
    'label' => 'Progress',
    'rules' => 'trim|required|integer'
),
array(
    'field' => 'type',
    'label' => 'Type',
    'rules' => 'trim|required'
),
array(
    'field' => 'isstarted',
    'label' => 'Is Started',
    'rules' => "
),
array(
    'field' => 'issuspended',
    'label' => 'Is Suspended',
    'rules' => "
)
),
),

'site/submitAdd' => array(
array(
    'field' => 'name',
    'label' => 'Nom',
    'rules' => 'trim|required|strToUpper'
),
array(
    'field' => 'cost',
    'label' => 'Cost',
    'rules' => 'trim|required|integer'

```

```
),
array(
    'field' => 'progress',
    'label' => 'Progress',
    'rules' => 'trim|required|integer'
),
array(
    'field' => 'type',
    'label' => 'Type',
    'rules' => 'trim|required'
),
array(
    'field' => 'isstarted',
    'label' => 'Is Started',
    'rules' => ''
),
array(
    'field' => 'issuspended',
    'label' => 'Is Suspended',
    'rules' => ''
)
);
?>
```

# Code source

## Modèles

Créez le fichier: [construction\\_sites/application/models/progressModel.php](#)  
Et collez-y le code source suivant:

```
<?php

class ProgressModel extends CI_Model {

    /* Progress table is a small referential table.
       So we don't use Activ Record technique to save ressources
       and we type classic SQL requests.
    */

    private $tableName = 'progress';

    function __construct(){
        parent::__construct();
    }

    // Get all, as an object:
    function get_all(){
        $sql = 'SELECT id, progress ';
        $sql .= ' FROM ' . $this->tableName;
        $sql .= ' order by id ';
        return $this->db->query($sql, array());
    }

    // Get all, as a clean array:
    function get_all_clean_array(){
        $recordSet = $this->get_all()->result();
        $array = array();

        foreach($recordSet as $row) {
            $array[$row->id] = $row->progress;
        }
    }
}
```

```

    return $array;
}

// Get by ID, as an object:
function get_by_id($id){
    $sql = 'SELECT * ';
    $sql .= ' FROM ' . $this->tableName;
    $sql .= ' WHERE id = ' . $id;
    return $this->db->query($sql, array());
}

} // end class

```

Créez le fichier: [construction\\_sites/application/models/siteModel.php](#)  
Et collez-y le code source suivant:

```

<?php

class SiteModel extends CI_Model {

    private $tableName = 'site';

    function __construct(){
        parent::__construct();
    }

    // Get number of records in table:
    function count_all(){
        return $this->db->count_all($this->tableName);
    }

    // Get records according paging:
    function get_paged_records($limit = 6, $offset = 0){
        $this->db->order_by('id','asc');
        return $this->db->get($this->tableName, $limit, $offset);
    }
}

```

```

// Add new record:
function save($arrNewRecord){
    $this->db->insert($this->tableName, $arrNewRecord);
    return $this->db->insert_id();
}

// Get record by id as an object:
function get_by_id($id){
    $this->db->where('id', $id);
    return $this->db->get($this->tableName);
}

// Insert new record.
// Returns last ID inserted:
function insertRecord($tblFields){
    $this->db->insert($this->tableName, $tblFields);
    return $this->db->insert_id();
}

// Update record by id:
function updateRecord($id, $arrFields){
    $this->db->where('id', $id);
    $this->db->update($this->tableName, $arrFields);
}

// Delete record by id:
function delete($id){
    $this->db->where('id', $id);
    $this->db->delete($this->tableName);
}

function getEmptyRecordAsObject() {
    $tblRecord = array(
        'id' => '',
        'name' => '',
        'cost' => '',
        'progress' => '',
        'isstarted' => ''
    );
}

```

```

'issuspended' => "
);
return (object)$tblRecord;
}

} // end class

```

## Vues

Créez le fichier: [construction\\_sites/application/views/siteDetail\\_tpl.php](#)

Et collez-y le code source suivant:

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>Construction sites (CRUD with CodeIgniter 2.0)</title>
<link href=<?php echo base_url(); ?>styles/styles.css" rel="stylesheet" type="text/css" />
</head>
<body>
<div class="content">
<h1><?php echo $title; ?></h1>
<div class="data">
<table>
<tr>
<td width="20%">ID</td>
<td><?php echo $site->id; ?></td>
</tr>
<tr>
<td valign="top">Name</td>
<td><?php echo $site->name; ?></td>
</tr>
<tr>
<td valign="top">Cost <?php echo $scurrency; ?></td>
<td><?php echo $site->cost; ?></td>

```

```

</tr>
<tr>
    <td valign="top">Progress %</td>
    <td><?php echo $site->progress; ?></td>
</tr>
<tr>
    <td valign="top">Type</td>
    <td><?php echo $site->type; ?></td>
</tr>
<tr>
    <td valign="top">Is Started</td>
    <td><?php echo $site->isstarted; ?></td>
</tr>
<tr>
    <td valign="top">Is Suspended</td>
    <td><?php echo $site->issuspended; ?></td>
</tr>
</table>
</div>
<br />
<?php echo $link_back; ?>
</div>
</body>
</html>

```

Créez le fichier: [construction\\_sites/application/views/siteEdit\\_frm.php](#)

Et collez-y le code source suivant:

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>Construction sites (CRUD with CodeIgniter 2.0)</title>
    <link href="<?php echo base_url(); ?>styles/styles.css" rel="stylesheet" type="text/css" />
</head>

<body>

```

```

<div class="content">
    <h1><?php echo $title; ?></h1>
    <?php echo $okMessage; ?>
    <form method="post" action="<?php echo $action; ?>">
        <div class="data">
            <table>
                <tr>
                    <td width="30%">ID</td>
                    <td>
                        <input type="text" name="id" disabled="disabled" class="text" value="<?php echo set_value('id', $site->id); ?>"/>
                        <?php echo form_error('id'); ?>
                        <input type="hidden" name="id" value="<?php echo set_value('id', $site->id); ?>"/>
                    </td>
                </tr>
                <tr>
                    <td valign="top">Name<span style="color:red;">*</span></td>
                    <td><input type="text" name="name" class="text" value="<?php echo set_value('name', $site->name); ?>"/>
                        <?php echo form_error('name'); ?></td>
                </tr>
                <tr>
                    <td valign="top">Cost <?php echo $currency; ?> <span style="color:red;">*</span></td>
                    <td><input type="text" name="cost" class="text" value="<?php echo set_value('cost', $site->cost); ?>"/>
                        <?php echo form_error('cost'); ?></td>
                </tr>
                <tr>
                    <td valign="top">Progress %<span style="color:red;">*</span></td>
                    <td>
                        <?php echo $selectList; ?>
                    </td>
                </tr>
                <tr>
                    <td valign="top">Type<span style="color:red;">*</span></td>
                    <td><input type="radio" name="type" value="public" <?php echo set_radio('type', 'public', $typePublic); ?>/> Public
                        <input type="radio" name="type" value="private" <?php echo set_radio('type', 'private', $typePrivate); ?>/> Private
                        <?php echo form_error('type'); ?>
                    </td>
                </tr>
        </div>
    </form>
</div>

```

```

<tr>
    <td valign="top">Is started<span style="color:red;">*</span></td>
    <td>
        <input type="checkbox" name="isstarted" value="1" <?php echo set_checkbox('isstarted', '1', $isStartedCheck); ?></input>
    </td>
</tr>
<tr>
    <td valign="top">Is suspended<span style="color:red;">*</span></td>
    <td><input type="checkbox" name="issuspended" value="1" <?php echo set_checkbox('issuspended', '1', $isSuspendedCheck); ?></input>
    </td>
</tr>
<tr>
    <td>&nbsp;</td>
    <td><input type="submit" value="Save"/></td>
</tr>
</table>
</div>
</form>
<br />
<?php echo $link_back; ?>
</div>
</body>
</html>

```

Créez le fichier: [construction\\_sites/application/views/siteList\\_tpl.php](#)

Et collez-y le code source suivant:

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
        <title>Construction sites (CRUD with CodeIgniter 2.0)</title>
        <link href="<?php echo base_url(); ?>styles/styles.css" rel="stylesheet" type="text/css" />
    </head>
    <body>

```

```

<div class="content">
    <h1>Construction sites (CRUD with CodeIgniter 2.0)</h1>
    <div class="data"><?php echo $table; ?></div>
    <div class="paging"><?php echo $pagination; ?></div>
    <br />
    <?php echo anchor('site/getAdd/', 'add new record', array('class' => 'add')); ?>
</div>
</body>
</html>

```

## **Controleur "Site"**

Créez le fichier: [construction\\_sites/application/controllers/site.php](#)

Et collez-y le code source suivant:

```

<?php

class Site extends CI_Controller {

    // Number of records per page:
    private $nbrOfLines = 6;
    private $currency = '€';

    function __construct(){
        parent::__construct();

        // Load libraries:
        $this->load->library(array('table', 'form_validation', 'pagination'));

        // Load helpers:
        $this->load->helper('url');

        // Load Site model with Activ Record activated:
        $this->load->model('siteModel', TRUE);

        // Load Progress model with Activ Record NOT activated.
        // For little referential tables, Activ Record is useless
    }
}

```

```

// but then we need to write classic SQL queries in the model.
$this->load->model('progressModel', FALSE);

// Redefine certain error messages:
$this->form_validation->set_message('required', '* required');

// We define a delimiter and a css class for error messages:
$this->form_validation->set_error_delimiters('<p class="error">', '</p>');

} // end constructor

function index($offset = 0){

/*
 * @Goal: Display the records list, according to offset.
 * Offset represent the number of the first line to be displayed.
 * @param: offset.
 */

// Offset provided by URI:
$uriSegment = 3;
$offset = $this->uri->segment($uriSegment);

// Load data:
$sites = $this->siteModel->get_paged_records($this->nbrOfLines, $offset)->result();

// Configure and generate pagination:
$config['base_url'] = site_url('site/index/');
$config['total_rows'] = $this->siteModel->count_all();
$config['per_page'] = $this->nbrOfLines;
$config['uri_segment'] = $uriSegment;
$this->pagination->initialize($config);
$data['pagination'] = $this->pagination->create_links();

// Generate data table:
$this->table->set_empty(" ");
$this->table->set_heading('Line', 'ID', 'Name', 'Cost ' . $this->currency, 'Progress %', 'Type', 'Is started', 'Is Suspended', 'Actions');


```

```

$offset = 0;
$i = 0 + $offset;

foreach ($sites as $site){
    $deleteMessage = 'Do you really want to delete this record: ' . $site->name . '?';
    $this->table->add_row(
        ++$i,
        $site->id,
        anchor('site/getDetail/' . $site->id, $site->name, array('class'=>'simple_link')),
        $site->cost,
        $site->progress,
        $site->type,
        $site->isstarted,
        $site->issuspended,
        anchor('site/getDetail/' . $site->id, 'view', array('class'=>'view')) . '',
        anchor('site/getUpdate/' . $site->id, 'update', array('class'=>'update')) . '',
        anchor('site/delete/' . $site->id, 'delete', array('class'=>'delete', 'onclick'=>"return confirm('" . $deleteMessage . "')"))
    ); // end add_row
} // end foreach

$data['table'] = $this->table->generate();

// Load view:
$this->load->view('siteList_tpl', $data);

} // end function

function getAdd(){
    /**
     * @Goal: Provide the form to create a new record.
     */
}

//*****
// BEGIN data preparation for the view.
//*****

// We have to complete $data[] to reach the same perimeter as Update action.
// This is necessary for we use the same form for both actions.
// To do so we fetch an empty record object. This object will not be used

```

```

// by the view, but must be present:
$objSite = $this->siteModel->getEmptyRecordAsObject();
$data['site'] = $objSite;

// Idem for site type:
$data['typePublic'] = FALSE;
$data['typePrivate'] = FALSE;

// Get progress table as an array:
$progress_array = $this->progressModel->get_all_clean_array();

// Construct the progress drop down list:
$indice = 0;
$data['selectList'] = form_dropdown('progress', $progress_array, $indice);

// Set some basic properties for the view:
$data['title'] = 'Add new record';
$data['okMessage'] = '';
$data['action'] = site_url('site/submitAdd');
$data['link_back'] = anchor('site/index/','Back',array('class'=>'back'));
$data['currency'] = $this->currency;

$data['isStartedCheck'] = FALSE;
$data['isSuspendedCheck'] = FALSE;

//*****
// END data preparation for the view.
//*****


// Load view:
$this->load->view('siteEdit_frm', $data);

} // end function

function submitAdd(){
/*
 * @Goal: Handle the submitted form to create a new record.

```

```

* @param: several params provided by URI.
*/
/* print_r($_REQUEST);
echo '</br>';
*/
// Run validation.
// form_validation object automatically fetch data provided by URI,
// and can transform it directly according your rules.
// The rules are in form_validation.php
if ($this->form_validation->run() == FALSE){
    $data['okMessage'] = "";
} else{
    // Prepare new record for the INSERT.
    // $this->input->post('xxxxx') return the data that has been
    // controled and/or transformed by form_validation:
    $arrNewRecord = array('name' => $this->input->post('name'),
        'cost'=> $this->input->post('cost'),
        'progress'=> $this->input->post('progress'),
        'type'=> $this->input->post('type'),
        'isstarted' => $this->input->post('isstarted'),
        'issuspended' => $this->input->post('issuspended'),
    );
}

/*
echo '</br>';
print_r($arrNewRecord);
echo '</br>';
*/
// Trigger INSERT:
$IdNewRecord = $this->siteModel->save($arrNewRecord);

// Set user message:
$data['okMessage'] = '<div class="success">Add new record: success</div>';

} // end if

```

```

//*****
// BEGIN data preparation for the view.
//*****


// set common properties for the view:
$data['title'] = 'Add new record';
$data['action'] = site_url('site/submitAdd');
$data['link_back'] = anchor('site/index/','Back',array('class'=>'back'));
$data['currency'] = $this->currency;

// We have to complete $data[] to reach the same perimeter as Update action.
// This is necessary for we use the same form for both actions.
// To do so we fetch an empty record object. This object will not be used
// by the view, but must be present:
$objSite = $this->siteModel->getEmptyRecordAsObject();
$data['site'] = $objSite;

// Set 'type' field for the view:
if (strtoupper($this->input->post('type')) == 'PUBLIC') {
    $data['typePublic'] = TRUE;
    $data['typePrivate'] = FALSE;
} else {
    $data['typePublic'] = FALSE;
    $data['typePrivate'] = TRUE;
}

// Get progress table as an array:
$progress_array = $this->progressModel->get_all_clean_array();

// Construct the progress drop down list:
$indice = $this->input->post('progress');
$data['selectList'] = form_dropdown('progress', $progress_array, $indice);

// Set default boolean values for the check boxes (necessary for the view):
$data['isStartedCheck'] = FALSE;

```

```

$data['isSuspendedCheck'] = FALSE;

//*****
// END data preparation for the view.
//*****


// Load view.
// Success or not: we re-display the form:
$this->load->view('siteEdit_frm', $data);

} // end function

function getDetail($id){

/*
 * @Goal: display a detail view of the record.
 * @param: site ID: provided by URI.
 */

/*
print_r($_REQUEST);
echo '</br>';
*/

//*****
// BEGIN data preparation for the view.
//*****


// set common properties
$data['title'] = 'Site detail';
$data['link_back'] = anchor('site/index/','Back',array('class'=>'back'));
$data['currency'] = $this->currency;

// Get site record:
$data['site'] = $this->siteModel->get_by_id($id)->row();

//*****
// END data preparation for the view.

```

```

//*****  

// Load view:  

$this->load->view('siteDetail_tpl', $data);  

} // end function  

function getUpdate($id){  

/**  

 * @Goal: Provide the form to update a record.  

 * @param: site id: provided by URI.  

 * @comment: For this getUpdate() action, the form fields can't be setted  

 * with form_validation methods like set_value(), because here we don't  

 * submit the form, we get it, and at this time form_validation doesn't  

 * know a thing about our fields.  

 * Fortunately we can use default values with form_validation methods.  

 * With set_value() for example, the second parameter (optional) can  

 * define a default value.  

 * That's why each form field is defined, in the view, with a form_validation  

 * method witch includes something for a default value.  

 */
  

//*****  

// BEGIN data preparation for the view.  

//*****  

// We fetch the site object from the model, and put it in the data[]  

// table for the view:  

$objSite = $this->siteModel->get_by_id($id)->row();  

$data['site'] = $objSite;  

/*  

echo '<br>';  

print_r($objSite);  

echo '<br>';  

*/

```

```

// Set some basic properties for the view:
$data['title'] = 'Update a record';
$data['okMessage'] = "";
$data['action'] = site_url('site/submitUpdate');
$data['link_back'] = anchor('site/index/', 'Back', array('class'=>'back'));
$data['currency'] = $this->currency;

// Set 'type' field for the view:
if (strtoupper($objSite->type) == 'PUBLIC') {
    $data['typePublic'] = TRUE;
    $data['typePrivate'] = FALSE;
} else {
    $data['typePublic'] = FALSE;
    $data['typePrivate'] = TRUE;
}

// Get progress table as an array:
$arrProgress = $this->progressModel->get_all_clean_array();

// Construct the progress drop down list for the view:
$indice = $objSite->progress;
$data['selectList'] = form_dropdown('progress', $arrProgress, $indice);

// Set default boolean values for the check boxes (necessary for the view):
if( $objSite->isstarted == 1 ){
    $data['isStartedCheck'] = TRUE;
} else {
    $data['isStartedCheck'] = FALSE;
}
if( $objSite->issuspended == 1 ){
    $data['isSuspendedCheck'] = TRUE;
} else {
    $data['isSuspendedCheck'] = FALSE;
}

//*****
// END data preparation for the view.

```

```

//*****



// load view:
$this->load->view('siteEdit_frm', $data);

} // end function


function submitUpdate(){

/*
 * @Goal: Handle the submitted form.
 * @param: several params provided with URI.
 */

/*
print_r($_REQUEST);
echo '</br>';
*/


// Run validation.
// form_validation object automatically fetch data provided by URI,
// and can transform it directly according your rules.
// The rules are in form_validation.php
if ($this->form_validation->run() == FALSE){
    // Error messages will be provided by form_validation.
    // We just set a general user message:
    $data['okMessage'] = '<div class="nosuccess">Something is wrong</div>';
} else{
    // All is well: we can update the record.
    // We take data that have been
    // securised and/or preped by form_validation, according your rules.
    // Note: each field must have a rule defined (even an empty
    // rule), otherwise the field will stay unknown from form_validation.
    $id = set_value('id');
    $arrSite = array('name' => $this->input->post('name'),
                    'cost' => $this->input->post('cost'),
                    'progress' => $this->input->post('progress'),
}

```

```

'type'=> $this->input->post('type'),
'isstarted' => $this->input->post('isstarted'),
'issuspended' => $this->input->post('issuspended'),
);

/*
echo '<br>';
print_r($arrSite);
echo '</br>';
*/
// We trigger UPDATE:
$this->siteModel->updateRecord($id, $arrSite);

// Set user message:
$data['okMessage'] = '<div class="success">Update record: success</div>';

} // end if

//*****
// BEGIN data preparation for the view.
//*****

// Set some basic properties for the view:
$data['title'] = 'Update record';
$data['action'] = site_url('site/submitUpdate'); // we will re-submit the form.
$data['link_back'] = anchor('site/index', 'Back', array('class'=>'back'));
$data['currency'] = $this->currency;

// We have to complete $data[] to reach the same perimeter as Add action.
// This is necessary for we use the same form for both actions.
// To do so we fetch an empty record object. This object will not be used
// by the view, but must be present:
$objSite = $this->siteModel->getEmptyRecordAsObject();
$data['site'] = $objSite;

// Set 'type' field for the view, even if it's not used by the view.
$data['typePublic'] = FALSE;

```

```

$data['typePrivate'] = FALSE;

// Get progress table as an array:
$progress_array = $this->progressModel->get_all_clean_array();

// Construct the progress drop down list.
// We use the form_dropdown helper:
$indice = $this->input->post('progress');
$data['selectList'] = form_dropdown('progress', $progress_array, $indice);

// Set default boolean values for the check boxes (necessary for the view):
$data['isStartedCheck'] = FALSE;
$data['isSuspendedCheck'] = FALSE;

*****  

// END data preparation for the view.  

*****  

// Load view:  

$this->load->view('siteEdit_frm', $data);  

} // end function

```

```

function delete($id){  

    /**  

     * @Goal: Delete the record according ID.  

     * @param: record ID.  

     */  


```

```

// Delete record:  

$this->siteModel->delete($id);  

// Redirect to list page:  

redirect('site/index/','refresh');

```

```
 } // end function
```

```
} // end class
```

```
?>
```

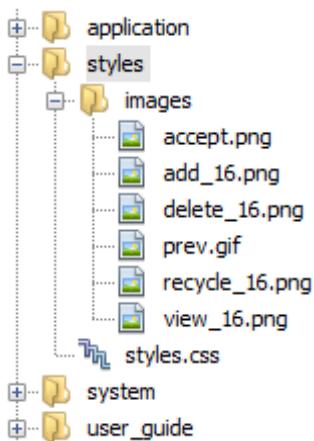
## CSS

Créez un repertoire "styles" au même niveau que "application" et "system".

Dans le repertoire "styles":

- Créez un fichier styles.css
- Créez un sous repertoire "images".

Vous devez obtenir ceci:

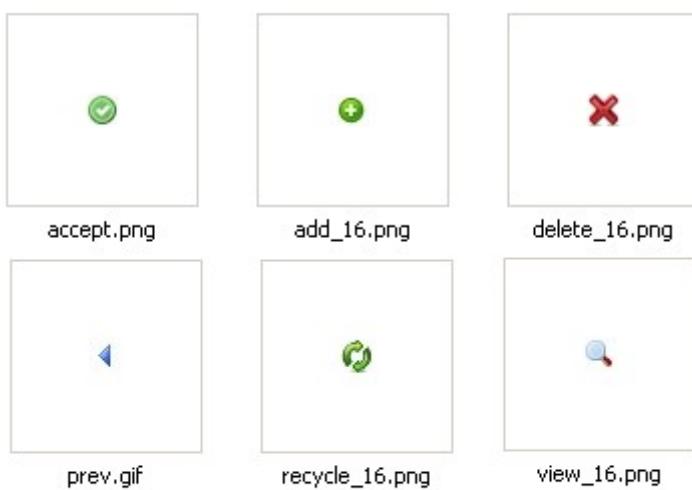


Selon le mode de diffusion de ce tutoriel, il se peut que je ne puisse pas l'accompagner des fichiers images. Dans ce cas vous pouvez trouver des images équivalentes sur internet (tapez "icônes" dans google).

Les images que j'ai utilisées font 16 pixels sur 16.

Le tableau ci-dessous montre à quoi doivent ressembler les images. Respectez le nom des fichiers images car ils sont référencés dans le fichier css.

Il suffit de poser les images dans le repertoire "images" et elles devraient apparaître dans le CRUD.



Enfin, dans le fichier styles.css créé plus haut, collez le code source suivant:

```

body {
    font-family: Gill, Helvetica, sans-serif;
    padding: 5px;
    margin: 5px;
    background: #fafafa;
}

div.content {
    padding: 5px 5px;
}
div.content h1 {
    font-size: 18pt;
    border-bottom: 5px solid darkseagreen;
    padding: 0px;
    margin: 10px 0px 20px;
    width: 80%;
}

div.content div.data table {
    border: 2px solid #000;
    background: #fff;
    width: 80%;
}
div.content div.data table td {

```

```
font-size: 10pt;
padding: 5px 10px;
border-bottom: 1px solid #ddd;
text-align :left;
}

div.content div.data table th {
    text-align: left;
    font-size: 10pt;
    padding: 10px 10px 7px;
    text-transform: uppercase;
    color: #fff;
    background: darkseagreen;
}

div.paging {
    font-size:13pt;
    margin:5px 0px;
}

div.paging a {
    color:#900;
    text-transform: uppercase;
    text-decoration: none;
}

div.paging a:hover {
    color: blue;
}

div.paging b {
    color:#900;
}

div.success {
    font-size:14pt;
    background:url(images/accept.png) left 5px no-repeat;
    padding:0px;
    padding-left:20px;
    margin:0px 0px 10px;
    color:#060;
    width:80%;
}
```

```
div.nosuccess {  
    font-size:14pt;  
    padding:0px;  
    padding-left:20px;  
    margin:0px 0px 10px;  
    color:#f00;  
    width:80%;  
}  
  
a.update, a.delete, a.add, a.view, a.back, a.simple_link {  
    font-size: 9pt;  
    color: #900;  
    font-weight: bold;  
    padding-left: 20px;  
    text-decoration: none;  
}  
  
a.simple_link {  
    font-size: 9pt;  
    color: #900;  
    font-weight: bold;  
    padding-left: 0px;  
    text-decoration: none;  
}  
  
a.update {  
    background:url(images/recycle_16.png) left center no-repeat;  
}  
  
a.delete {  
    background:url(images/delete_16.png) left center no-repeat;  
}  
  
a.add {  
    background:url(images/add_16.png) left center no-repeat;  
}  
  
a.view {  
    background:url(images/view_16.png) left center no-repeat;  
}
```

```
a.back {  
    background:url(images/prev.gif) left center no-repeat;  
}  
  
a.update:hover, a.delete:hover, a.add:hover, a.view:hover, a.simple_link:hover {  
    color: blue;  
}  
  
input.text {  
    border:2px solid #aaa;  
}  
  
.error {  
    background: #FBE6F2 none repeat scroll 0 0;  
    border: 1px solid #D893A1;  
    color: #333;  
    margin: 5px 0 0;  
    padding: 5px;  
    font-size: 10px;  
    font-family: Lucida Grande,Verdana,Geneva,Sans-serif;  
}
```

## Divers

### Paramètres

Deux paramètres peuvent être changés au début du contrôleur (site.php):

- Le symbole de la monnaie utilisée (\$currency). Il apparaît à différents endroits dans les vues.
- Le nombre de lignes souhaité dans chaque page du CRUD (\$nbrOfLines).

### Commentaires du code source

Les sources PHP sont généreusement commentés.

J'ai mis en commentaire plusieurs lignes du type "print\_r", que vous pouvez décommenter pour afficher les données à l'écran, à certaines phases intéressantes.

## Le grand secret

Comme vous le constatez dans les méthodes du contrôleur, 80% du code est consacré à la préparation des données pour la vue.

Quelquefois les données viennent de la base de données, quelquefois elles proviennent de l'URL, et quelquefois elles doivent être converties pour la vue.

Le tableau suivant explique ce que le contrôleur doit faire pour chaque couple donnée/action, pour alimenter notre unique formulaire. Cet aspect représente vraiment le cœur du CRUD.

(J'ai du couper le tableau en plusieurs parties pour le rendre lisible)

Nom de zone	Controlée par Form_validation ?	Instruction PHP pour afficher la donnée dans la vue
id	oui, mais règle vide.	echo set_value('id', \$site->id);

D'où vient la donnée pour getAdd() ?	D'où vient la donnée pour SubmitAdd() ?
<p>Avec getAdd(), le formulaire n'est pas soumis, donc les champs du formulaire sont inconnus de la classe Form_validation, c'est donc la valeur par défaut \$site-&gt;id qui est utilisée par la vue.</p> <p>C'est pourquoi le contrôleur doit préparer un objet « site » vide.</p>	<p>Le formulaire est soumis et il y a une règle de validation pour ce champ, il est donc connu de la classe Form_validation, et il est restitué par set_value('id').</p> <p>L'objet "site" vide est tout de même nécessaire à la vue, même s'il n'est pas utilisé.</p>

D'où vient la donnée pour getUpdate() ?	D'où vient la donnée pour SubmitUpdate() ?
<p>Avec getUpdate(), le formulaire n'est pas soumis, donc les champs du formulaire sont inconnus de la classe Form_validation, c'est donc la valeur par défaut \$site-&gt;id qui est utilisée par la vue.</p> <p>C'est pourquoi le contrôleur doit préparer un objet « site » avec les données de la base.</p>	<p>Le formulaire est soumis et il y a une règle de validation pour ce champ. Cependant le champ restera inconnu de la classe Form_validation à cause de l'attribut disabled="disabled".</p> <p>C'est pourquoi on a ajouté un deuxième ID dans le formulaire, sous la forme d'un champ caché (hidden). Le champ caché est restitué par set_value('id'). L'objet "site" vide est toujours requis même si non utilisé.</p>

Nom de zone	Controlée par Form_validation ?	Instruction PHP pour afficher la donnée dans la vue
name	oui	echo set_value('name', \$site->name);

D'où vient la donnée pour getAdd() ?	D'où vient la donnée pour SubmitAdd() ?
<p>Avec getAdd(), le formulaire n'est pas soumis, donc les champs du formulaire sont inconnus de la classe Form_validation, c'est donc la valeur par défaut \$site-&gt;name qui est utilisée par la vue.</p> <p>C'est pourquoi le contrôleur doit préparer un objet « site » vide.</p>	<p>Le formulaire est soumis et il y a une règle de validation pour ce champ, il est donc connu de la classe Form_validation, et il est restitué par set_value('name').</p> <p>L'objet "site" vide est tout de même nécessaire à la vue, même s'il n'est pas utilisé.</p>

D'où vient la donnée pour getUpdate() ?	D'où vient la donnée pour SubmitUpdate() ?
<p>Avec getUpdate(), le formulaire n'est pas soumis, donc les champs du formulaire sont inconnus de la classe Form_validation, c'est donc la valeur par défaut \$site-&gt;name qui est utilisée par la vue.</p> <p>C'est pourquoi le contrôleur doit préparer un objet « site » avec les données de la base.</p>	<p>Le formulaire est soumis et il y a une règle de validation pour ce champ, il est donc connu de la classe Form_validation, et il est restitué par set_value('name').</p> <p>L'objet "site" vide est tout de même nécessaire à la vue, même s'il n'est pas utilisé.</p>

Nom de zone	Controlée par Form_validation ?	Instruction PHP pour afficher la donnée dans la vue
cost	oui	<code>echo set_value('cost', \$site-&gt;cost);</code>

D'où vient la donnée pour getAdd() ?	D'où vient la donnée pour SubmitAdd() ?
<p>Avec getAdd(), le formulaire n'est pas soumis, donc les champs du formulaire sont inconnus de la classe Form_validation, c'est donc la valeur par défaut \$site-&gt;cost qui est utilisée par la vue.</p> <p>C'est pourquoi le contrôleur doit préparer un objet « site » vide.</p>	<p>Le formulaire est soumis et il y a une règle de validation pour ce champ, il est donc connu de la classe Form_validation, et il est restitué par set_value('cost').</p> <p>L'objet "site" vide est tout de même nécessaire à la vue, même s'il n'est pas utilisé.</p>

D'où vient la donnée pour getUpdate() ?	D'où vient la donnée pour SubmitUpdate() ?
<p>Avec getUpdate(), le formulaire n'est pas soumis, donc les champs du formulaire sont inconnus de la classe Form_validation, c'est donc la valeur par défaut \$site-&gt;cost qui est utilisée par la vue.</p> <p>C'est pourquoi le contrôleur doit préparer un objet « site » avec les données de la base.</p>	<p>The form is submitted and there's a rule for that field, so it's known from Form_Validation class, and is rendered by set_value('cost'). But an empty site object is needed by the view even if it's not used.</p>

Nom de zone	Controlée par Form_validation ?	Instruction PHP pour afficher la donnée dans la vue
progress	oui	echo \$selectList;

D'où vient la donnée pour getAdd() ?	D'où vient la donnée pour SubmitAdd() ?
Cet élément HTML est une liste de choix, qui doit être construite par le contrôleur. Ici le contrôleur définira l'index à zéro pour positionner la liste sur le premier item.	Le contrôleur doit constituer la liste de choix en positionnant l'index sur l'item sélectionné par l'utilisateur.

D'où vient la donnée pour getUpdate() ?	D'où vient la donnée pour SubmitUpdate() ?
Le contrôleur doit constituer la liste de choix en positionnant l'index sur l'item provenant de la base de données..	Le contrôleur doit constituer la liste de choix en positionnant l'index sur l'item sélectionné par l'utilisateur.

Nom de zone	Controlée par Form_validation ?	Instruction PHP pour afficher la donnée dans la vue
type (radio button)	oui	echo set_radio('type', 'public', \$typePublic);

D'où vient la donnée pour getAdd() ?	D'où vient la donnée pour SubmitAdd() ?
Avec getAdd(), le formulaire n'est pas soumis, donc les champs du formulaire sont inconnus de la classe Form_validation, c'est donc la valeur par défaut \$typePublic qui est utilisée par la vue. C'est pourquoi le contrôleur doit préparer la variable \$typePublic.	Le formulaire est soumis et il y a une règle de validation pour ce champ, il est donc connu de la classe Form_validation, et il est restitué par set_radio('type', 'public'). La variable \$typePublic est tout de même nécessaire même si non utilisée.

D'où vient la donnée pour getUpdate() ?	D'où vient la donnée pour SubmitUpdate() ?
Avec getUpdate(), le formulaire n'est pas soumis, donc les champs du formulaire sont inconnus de la classe Form_validation, c'est donc la valeur par défaut \$typePublic qui est utilisée par la vue. C'est pourquoi le contrôleur doit préparer la variable \$typePublic.	Le formulaire est soumis et il y a une règle de validation pour ce champ, il est donc connu de la classe Form_validation, et il est restitué par set_radio('type', 'public'). La variable \$typePublic est tout de même nécessaire même si non utilisée.

Nom de zone	Controlée par Form_validation ?	Instruction PHP pour afficher la donnée dans la vue
type (radio button)	oui	<code>echo set_radio('type', 'private', \$typePrivate);</code>

D'où vient la donnée pour getAdd() ?	D'où vient la donnée pour SubmitAdd() ?
<p>Avec getAdd(), le formulaire n'est pas soumis, donc les champs du formulaire sont inconnus de la classe Form_validation, c'est donc la valeur par défaut \$typePrivate qui est utilisée par la vue.</p> <p>C'est pourquoi le contrôleur doit préparer la variable \$typePrivate.</p>	<p>Le formulaire est soumis et il y a une règle de validation pour ce champ, il est donc connu de la classe Form_validation, et il est restitué par set_radio('type', 'private').</p> <p>La variable \$typePrivate est tout de même nécessaire même si non utilisée.</p>

D'où vient la donnée pour getUpdate() ?	D'où vient la donnée pour SubmitUpdate() ?
<p>Avec getUpdate(), le formulaire n'est pas soumis, donc les champs du formulaire sont inconnus de la classe Form_validation, c'est donc la valeur par défaut \$typePrivate qui est utilisée par la vue.</p> <p>C'est pourquoi le contrôleur doit préparer la variable \$typePrivate.</p>	<p>Le formulaire est soumis et il y a une règle de validation pour ce champ, il est donc connu de la classe Form_validation, et il est restitué par set_radio('type', 'private').</p> <p>La variable \$typePrivate est tout de même nécessaire même si non utilisée.</p>

Nom de zone	Controlée par Form_validation ?	Instruction PHP pour afficher la donnée dans la vue
isstarted (checkbox)	oui, mais règle vide.	echo set_checkbox('isstarted', '1', \$isStartedCheck);

D'où vient la donnée pour getAdd() ?	D'où vient la donnée pour SubmitAdd() ?
Avec getAdd(), le formulaire n'est pas soumis, donc les champs du formulaire sont inconnus de la classe Form_validation, c'est donc la valeur par défaut \$isStartedCheck qui est utilisée par la vue. C'est pourquoi le contrôleur doit préparer la variable \$isStartedCheck.	Le formulaire est soumis et il y a une règle de validation pour ce champ, il est donc connu de la classe Form_validation, et il est restitué par set_checkbox('isstarted', '1'). La variable \$isStartedCheck est tout de même nécessaire même si non utilisée.

D'où vient la donnée pour getUpdate() ?	D'où vient la donnée pour SubmitUpdate() ?
Avec getUpdate(), le formulaire n'est pas soumis, donc les champs du formulaire sont inconnus de la classe Form_validation, c'est donc la valeur par défaut \$isStartedCheck qui est utilisée par la vue. C'est pourquoi le contrôleur doit préparer la variable \$isStartedCheck.	Le formulaire est soumis et il y a une règle de validation pour ce champ, il est donc connu de la classe Form_validation, et il est restitué par set_checkbox('isstarted', '1'). La variable \$isStartedCheck est tout de même nécessaire même si non utilisée.

Nom de zone	Controlée par Form_validation ?	Instruction PHP pour afficher la donnée dans la vue
issuspended (checkbox)	yes, but empty rule	echo set_checkbox('issuspended', '1', \$isSuspendedCheck);

D'où vient la donnée pour getAdd() ?	D'où vient la donnée pour SubmitAdd() ?
Avec getAdd(), le formulaire n'est pas soumis, donc les champs du formulaire sont inconnus de la classe Form_validation, c'est donc la valeur par défaut \$isSuspendedCheck qui est utilisée par la vue. C'est pourquoi le contrôleur doit préparer la variable \$isSuspendedCheck.	Le formulaire est soumis et il y a une règle de validation pour ce champ, il est donc connu de la classe Form_validation, et il est restitué par set_checkbox('issuspended', '1'). La variable \$isSuspendedCheck est tout de même nécessaire même si non utilisée.

D'où vient la donnée pour getUpdate() ?	D'où vient la donnée pour SubmitUpdate() ?
Avec getUpdate(), le formulaire n'est pas soumis, donc les champs du formulaire sont inconnus de la classe Form_validation, c'est donc la valeur par défaut \$isSuspendedCheck qui est utilisée par la vue. C'est pourquoi le contrôleur doit préparer la variable \$isSuspendedCheck.	Le formulaire est soumis et il y a une règle de validation pour ce champ, il est donc connu de la classe Form_validation, et il est restitué par set_checkbox('issuspended', '1'). La variable \$isSuspendedCheck est tout de même nécessaire même si non utilisée.

# Conclusion

Votre CRUD devrait maintenant fonctionner correctement.

Si vous créez ou modifiez un enregistrement et qu'une règle n'est pas satisfaite (par exemple vous ne donnez pas de nom au chantier), vous devriez voir un message d'erreur apparaître sous la zone concernée.

Des outils aujourd'hui sont capables de construire des CRUD automatiquement pour vous, mais il est intéressant de pouvoir le faire soi-même.

Ce tutoriel montre une technique avec un formulaire unique. Une fois maîtrisée, cette technique sera identique pour tous vos CRUD.

Quelques aspects peuvent être améliorés. Par exemple:

- Dans les modèles, on constate que certaines requêtes sont les mêmes pour plusieurs modèles (c'est vrai qu'ici on n'en n'a que deux), comme `get_by_id()`. Ces requêtes peuvent être factorisées dans une classe intermédiaire entre `CI_Model` et votre classe modèle.
- Les balises html `<Doctype>` et `<head>` sont globalement les mêmes dans chaque vue. On pourrait les factoriser dans une vue unique qui générera les headers html en utilisant les helpers CodeIgniter comme `doctype()` et `head()`.

Merci d'avoir lu ce tutoriel.

**Fin du document**