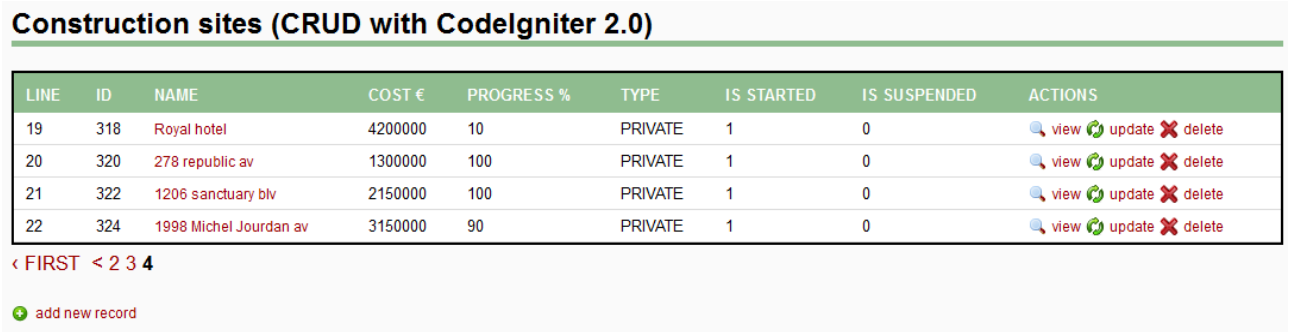
# CRUD with CodeIgniter 2.0

version 1.0: 2011-09-12
author: Maxime Keltsma (mailto: m.keltsma@voila.fr)

## Index



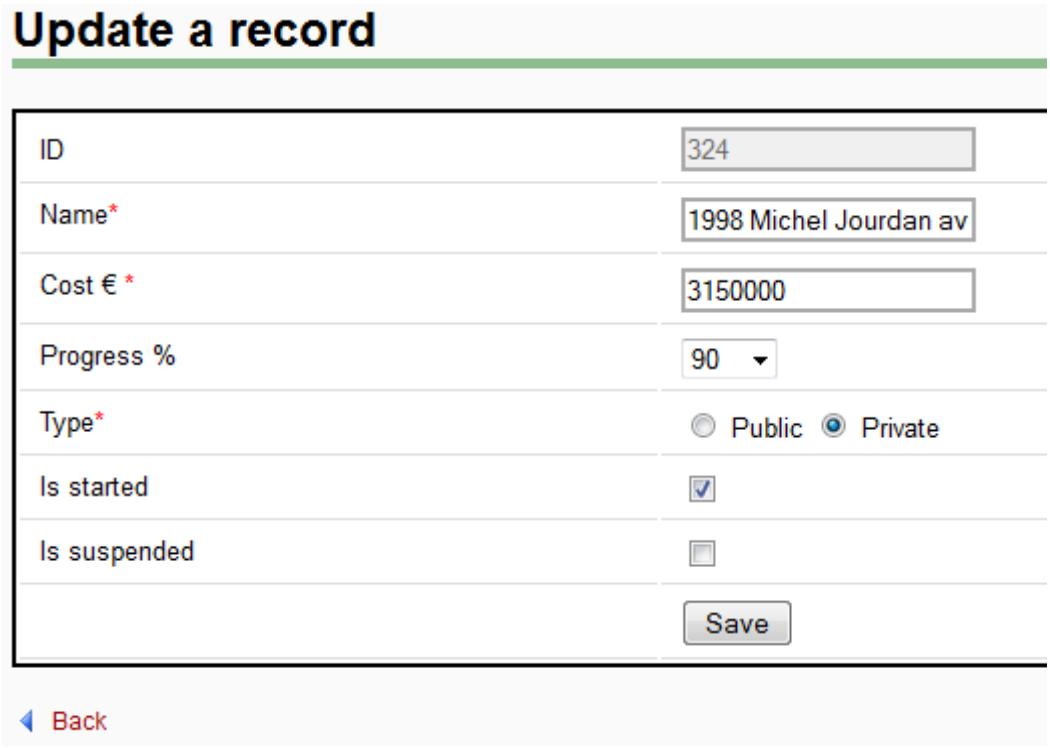
In this tutorial, we are going to make basic CRUD (Create Read Update Delete) application using CodeIgniter 2.0 framework.
We'll handle a simple table containing construction sites.
The CRUD would look like this:



The data that compose a construction site will allow us to manipulate:

- A caracter string for site name.
- An integer value for the cost.
- A select list for progress value.
- A radio button for site type (private or public).
- Two checkboxes for isStarted and isSuspended.

The form to update/create a site would look like this:




## Prerequisites

I assume:

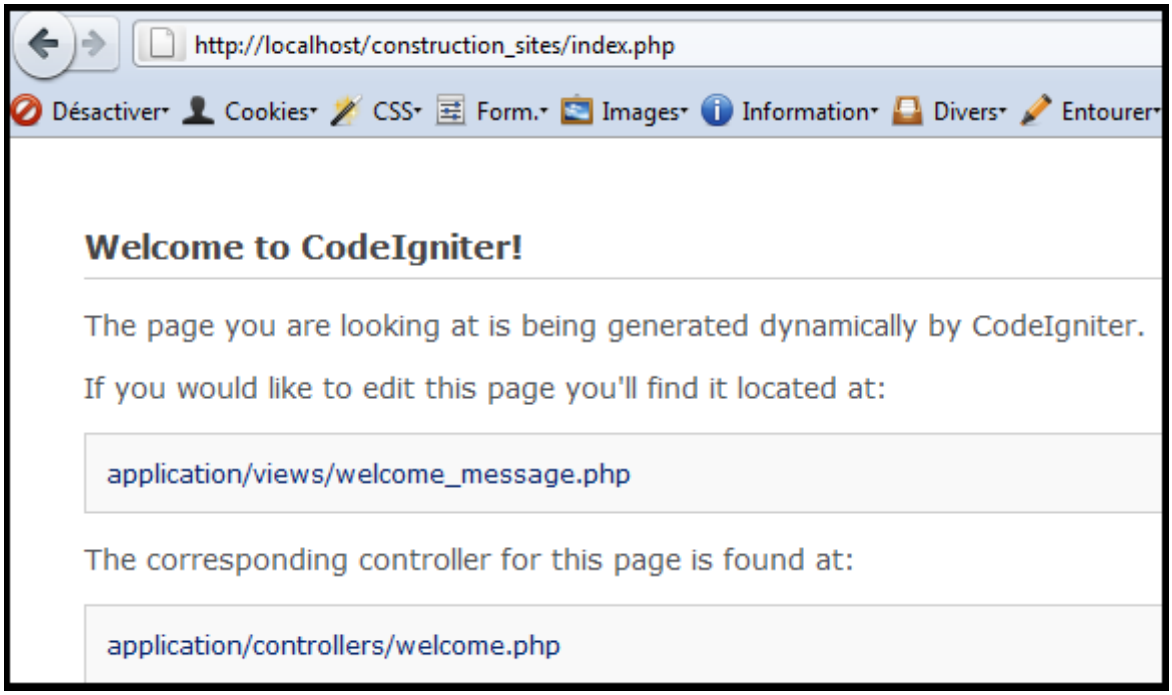
- You have a database server and http server. (I used Wamp)
- You have installed CodeIgniter 2.0.3 or better, for a project called "construction_sites".
- You have at least basic notions in PHP 5 and SQL.
- You are crazy enough to code PHP on week end ;-))

## What i used

- 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/
- For SQL work, i used old Mysql Query Browser, but you can use phpMyAdmin provided with Wamp, or the new tool Mysql Workbench.
- Bouml for UML diagrams: http://bouml.free.fr
- Netbeans 6.9: http://netbeans.org/
- Open Office: http://www.openoffice.org/ to type this tutorial.

## First URL test

If not done, create a new CodeIgniter installation for a project called "construction_sites".
Using this URL, you should see the CodeIgniter welcome screen.
http://localhost/construction_sites/index.php




## Database

With your SQL tool, execute those scripts to create and populate our small database.

```
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);
```

# Application configuration

Set the configuration in the following CodeIgniter config files:

## *In construction_sites/application/config/config.php:*

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

## *In 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;
```

## *In construction_sites/application/config/routes.php:*

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

## Project specifications

Here are actions and methods we are going to implement into our controler site.php.

| User action | Controler method | View |
|---|---|---|
| Get the records list | index() | siteList_tpl.php |
| Get the record details | 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:

- The same form will be used for create and update actions. This give us a little more complicated form, but it's better to have a unique form.
- I named the views with _tpl for simple html template, and _frm for forms.

We'll use:

CodeIgniter Table library, to generate html table code.

CodeIgniter Form_Validation library, to control data coming from the user.

CodeIgniter Pagination library to control pagination and generate navigation bar for the records list.

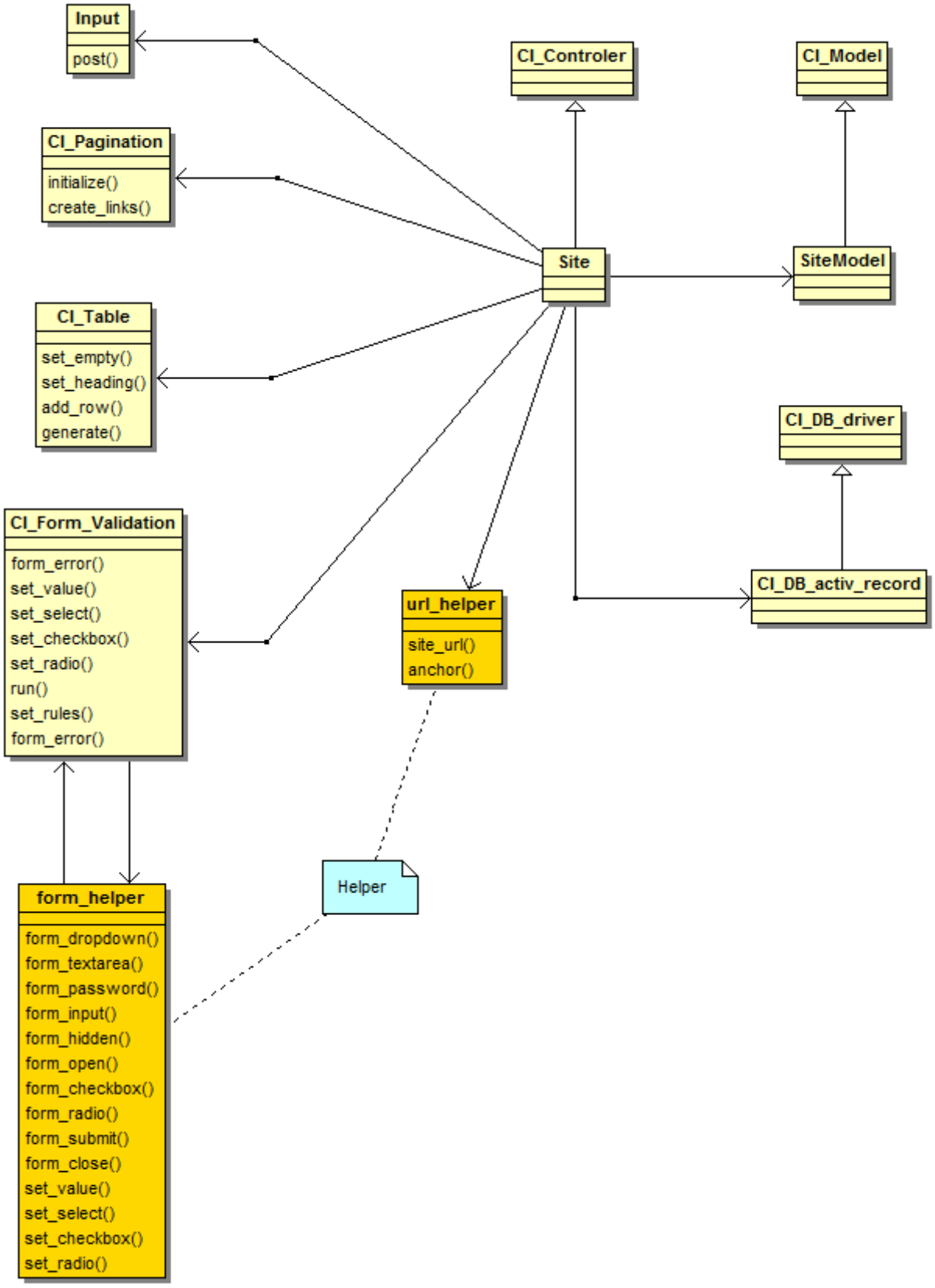
CodeIgniter URL helper.

Here are the files we have to make:

| Type | File name |
|---|---|
| Controler | site.php |
| Model | progressModel.php |
| Model | siteModel.php |
| View | siteDetail_tpl.php |
| View | siteList_tpl.php |
| View | siteEdit_frm.php |

## *Class diagram*

Here is a certainly incomplete class view, but that's what i so far understood from CodeIgniter.



We don't handle:

- Authentication.
- Logs.

## Models

For Site table  model class, we'll use Activ Record technique (see Activ Record in CodeIgniter manuel) because:

- The work is easier.
- We can change database server almost without having to touch the code.

For Progress table, witch is a small referential table, and to see another technique, we'll type classic SQL request  (Not use Activ Record allow to save ressources).

## Form validation rules

Validations rules allow to:

- Control, securise and prepare data typed in by the user.
- Display specific error message when a rule is not satisfied.

CodeIgniter provides several ways to handle validation rules as you can see in CodeIgniter user guide.

For this tutorial, i choose to define the rules in an external file, and group the rules for each action.

Create the file: construction_sites/application/config/form_validation.php, and copy the code below in it:

```
<?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' => ''
        )
)
);
?>
```

# Source code

## *Models*

Create the file: construction_sites/application/models/progressModel.php

And paste the following php source code in it:

```
<?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
```

Create the file: construction_sites/application/models/siteModel.php
And paste the following php source code in it:

```
<?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
```

## *Views*

Create the file: construction_sites/application/views/siteDetail_tpl.php

And paste the following source code in it:

```
<!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 $curency; ?></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>
```

Create the file: construction_sites/application/views/siteEdit_frm.php

And paste the following source code in it:

```
<!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="disable" 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 $curency; ?> <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>
```

```
                <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> </td>
                    <td><input type="submit" value="Save"/></td>
                </tr>
            </table>
        </div>
    </form>
    <br />
    <?php echo $link_back; ?>
  </div>
 </body>
</html>
```

Create the file: construction_sites/application/views/siteList_tpl.php

And paste the following source code in it:

```
<!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>
```

## *Site controler*

Create the file: construction_sites/application/controlers/site.php
And paste the following php source code in it:

```
<?php

class Site extends CI_Controller {

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

    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("&amp;nbsp;");
        $this->table->set_heading('Line', 'ID', 'Name', 'Cost ' . $this->curency, 'Progress %', 'Type', 'Is started', 'Is
Suspended', 'Actions');
```

```
        $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['curency'] = $this->curency;


        $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 submited 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 trnsformed 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['curency'] = $this->curency;


// 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['curency'] = $this->curency;

    // 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['curency'] = $this->curency;

// 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 submited 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['curency'] = $this->curency;


// 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

?>
```

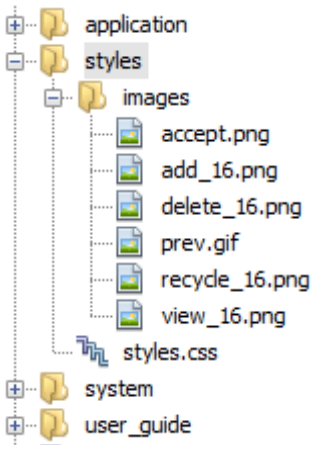
# CSS

Create a directory "styles" at the same level as "application" and "system".

Into "styles" directory:

- Create a file "styles.css".
- Create a "images" sub directory.

You should obtain this:

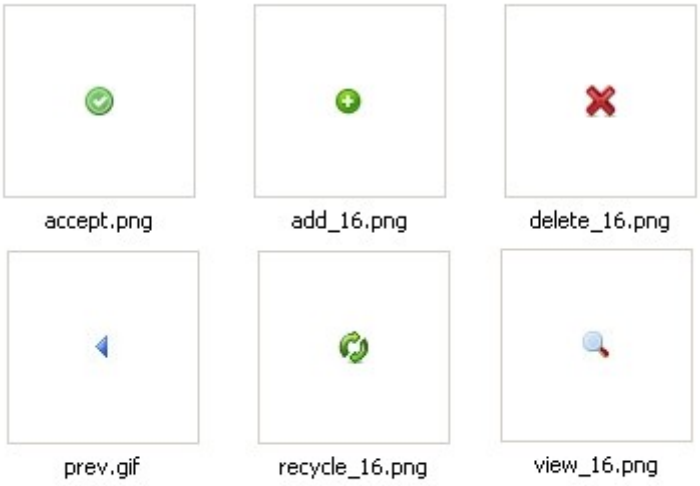



Depending on kind of distribution of this tutorial, i could possibly not be abble to provide icone pictures. In this case, you can find equivalent icones on internet.

I used 16x16 pixels icones.

The following table shows what icones should look like. Respect files names for they are referenced in CSS file.

You just have to put pictures in "images" directory and they should appear on the CRUD.




At last, paste the following css code in styles.css

```
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-wight: bold;
	padding-left: 20px;
	text-decoration: none;
}

a.simple_link {
	font-size: 9pt;
	color: #900;
	font-wight: 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;
}
```

# Misc

## *Parameters*

Two parameters can be changed  at the begining of site controler:

- Curency symbol. It appears at different places in the views.
- Number of lines in CRUD pages.

## *Source code comment*

The php source code is widely commented.

I commented several lines like "print_r" function that you can uncomment to see data on the screen at interesting phases.

# The great secret

As you see in the controler methods, 80% of the code is dedicated to prepare data for the view.

Sometimes data come from database, sometimes it's provided by URL, and sometimes it must be converted for the view.

The following table explain what the controler do for each data/action, to handle our unique form. That's the heart of the CRUD.

(I had to cut my big table in parts to make it readable)

| Zone name | Is controled by Form_validation ? | PHP code to display the data from the view |
|---|---|---|
| id | yes, but empty rule | echo set_value('id', $site->id); |

| Where does the data comes from for getAdd() ? | Where does the data comes from for SubmitAdd() ? |
|---|---|
| For getAdd() action, the form is not submitted, so form fields are unknown from Form_Validation class, so the default value $site->id is used by the view.<br>That's why controler must prepare an empty site object. | 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('id'). But an empty site object is needed by the view even if it's not used. |

| Where does the data comes from for getUpdate() ? | Where does the data comes from for SubmitUpdate() ? |
|---|---|
| For getUpdate() action, the form is not submitted, so form fields are unknown from Form_Validation class, so the default value $site->id is used by the view.<br>That's why controler must prepare a site object from database. | The form is submitted and there's a rule for that field. Despite the field stay unknown from Form_Validation class, because of the tag disabled="disabled". That's why we add a hidden field for the ID in the form.<br>The hidden field is rendered by set_value('id'). But an empty site object is needed by the view even if it's not used. |

| Zone name | Is controled by Form_validation ? | PHP code to display the data from the view |
|---|---|---|
| name | yes | echo set_value('name', $site->name); |

| Where does the data comes from for getAdd() ? | Where does the data comes from for SubmitAdd() ? |
|---|---|
| For getAdd() action, the form is not submitted, so form fields are unknown from Form_Validation class, so the default value $site->name is used by the view.<br>That's why controler must prepare an empty site object. | 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('name'). But an empty site object is needed by the view even if it's not used. |

| Where does the data comes from for getUpdate() ? | Where does the data comes from for SubmitUpdate() ? |
|---|---|
| For getUpdate() action, the form is not submitted, so form fields are unknown from Form_Validation class, so the default value $site->name is used by the view.<br>That's why controler must prepare a site object from database. | 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('name'). |

| Zone name | Is controled by Form_validation ? | PHP code to display the data from the view |
|---|---|---|
| cost | yes | echo set_value('cost', $site->cost); |

| Where does the data comes from for getAdd() ? | Where does the data comes from for SubmitAdd() ? |
|---|---|
| For getAdd() action, the form is not submitted, so form fields are unknown from Form_Validation class, so the default value $site->cost is used by the view.<br>That's why controler must prepare an empty site object. | 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. |

| Where does the data comes from for getUpdate() ? | Where does the data comes from for SubmitUpdate() ? |
|---|---|
| For getUpdate() action, the form is not submitted, so form fields are unknown from Form_Validation class, so the default value $site->cost is used by the view.<br>That's why controler must prepare a site object from database. | 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. |

| Zone name | Is controled by Form_validation ? | PHP code to display the data from the view |
|---|---|---|
| progress | yes | echo $selectList; |

| Where does the data comes from for getAdd() ? | Where does the data comes from for SubmitAdd() ? |
|---|---|
| This html element is a drop down list that have to be constructed by the controler. Here the controler will set index to zero corresponding to the first list item. | The controler have to construct the drop down list with an index corresponding to what the user selected. |

| Where does the data comes from for getUpdate() ? | Where does the data comes from for SubmitUpdate() ? |
|---|---|
| The controler have to construct the drop down list with an index corresponding to the database field. | The controler have to construct the drop down list with an index corresponding to what the user selected. |

| Zone name | Is controled by Form_validation ? | PHP code to display the data from the view |
|---|---|---|
| type (radio button) | yes | echo set_radio('type', 'public', $typePublic); |

| Where does the data comes from for getAdd() ? | Where does the data comes from for SubmitAdd() ? |
|---|---|
| For getAdd() action, the form is not submitted, so form fields are unknown from Form_Validation class, so the default value $typePublic is used by the view.<br>That's why controler must set $typePublic variable. | 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_radio('type', 'public'). |

| Where does the data comes from for getUpdate() ? | Where does the data comes from for SubmitUpdate() ? |
|---|---|
| For getUpdate() action, the form is not submitted, so form fields are unknown form Form_Validation, so the default value $typePublic is used by the view.<br>That's why controler must set $typePublic variable. | 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_radio('type', 'public'). |

| Zone name | Is controled by Form_validation ? | PHP code to display the data from the view |
|---|---|---|
| type (radio button) | yes | echo set_radio('type', 'private', $typePrivate); |

| Where does the data comes from for getAdd() ? | Where does the data comes from for SubmitAdd() ? |
|---|---|
| For getAdd() action, the form is not submitted, so form fields are unknown form Form_Validation class, so the default value $typePrivate is used by the view.<br>That's why controler must set $typePrivate variable. | 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_radio('type', 'private'). |

| Where does the data comes from for getUpdate() ? | Where does the data comes from for SubmitUpdate() ? |
|---|---|
| For getUpdate() action, the form is not submitted, so form fields are unknown form Form_Validation, so the default value $typePrivate is used by the view.<br>That's why controler must set $typePrivate variable. | 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_radio('type', 'private'). |

| Zone name | Is controled by Form_validation ? | PHP code to display the data from the view |
|---|---|---|
| isstarted (checkbox) | yes, but empty rule | echo set_checkbox('isstarted', '1', $isStartedCheck); |

| Where does the data comes from for getAdd() ? | Where does the data comes from for SubmitAdd() ? |
|---|---|
| For getAdd() action, the form is not submitted, so form fields are unknown form Form_Validation class, so the default value $isStartedCheck is used by the view.<br>That's why controler must set $isStartedCheck variable. | 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_checkbox('isstarted', '1'). |

| Where does the data comes from for getUpdate() ? | Where does the data comes from for SubmitUpdate() ? |
|---|---|
| For getUpadte() action, the form is not submitted, so form fields are unknown form Form_Validation class, so the default value $isStartedCheck is used by the view.<br>That's why controler must set $isStartedCheck variable. | 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_checkbox('isstarted', '1'). |

| Zone name | Is controled by Form_validation ? | PHP code to display the data from the view |
|---|---|---|
| issuspended (checkbox) | yes, but empty rule | echo set_checkbox('issuspended', '1', $isSuspendedCheck); |

| Where does the data comes from for getAdd() ? | Where does the data comes from for SubmitAdd() ? |
|---|---|
| For getAdd() action, the form is not submitted, so form fields are unknown form Form_Validation class, so the default value $isSuspendedCheck is used by the view. That's why controler must set $isSuspendedCheck variable. | 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_checkbox('isSuspended', '1'). |

| Where does the data comes from for getUpdate() ? | Where does the data comes from for SubmitUpdate() ? |
|---|---|
| For getUpdate() action, the form is not submitted, so form fields are unknown form Form_Validation class, so the default value $isSuspendedCheck is used by the view. That's why controler must set $isSuspendedCheck variable. | 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_checkbox('isSuspended', '1'). |

# Conclusion

Your CRUD should now work properly.

If you create or update a record and a rule is not satisfied (for example you don't type a site name), you should see an error message appear under the relevant field.

Tools today can automatically construct a CRUD for you, but it's interesting to be abble to do it yourself.

This tutorial show a technique with a unique form. Once understood, this technique will be the same for all your CRUD.

Some aspects can be improved. For example:

In the models, you see that some requests are the same for each model like get_by_id(). Those requests could be factorised in an intermediate class between CI_Model and your model class.

<Doctype> and <head> html tags, are  globally the same in each view. We could factorise them in a unique view that generate html headers using CodeIgniter html_helper functions like doctype() and head().

Thank you for reading.

**End of document**