Bhupal Sapkota

PyroCMS Version: 1.3.2 Community Tutorial (Update 2011)

What's here:

PyroCMS Module – Folder Location?
PyroCMS Module – Files Required & Folder Structure?
PyroCMS Module – Frontend/Backend Snapshot (of what we are going to build)
PyroCMS Module – Coding the Hello World module with full MVC implementation
Details.php – This file contains the module name, description, version, whether the module is available to the front-end and/or backend, database install and uninstall methods help methods etc.
Hello world module Code

Summary

With this text, I assume you are familiar with the PyroCMS folder structure and theming. I am describing a PyroCMS Module with very basic concepts. Please double check what I have highlighted here, you may get errors when installing/running the module we create, otherwise.

PyroCMS is good, but its documentation is not enough I mean, once you know stuff, its very easy to understand, but this is rarely the case with beginners, that's why I thought I would write this example.

Don't copy-paste the code from the example below rather download source code from github, in copy pasting you might face quotes errors. mysql query might not work.

PyroCMS Module – Folder Location?

PyroCMS Core Modules

e.g htdocs\pyrocms\system\cms\modules\pages

For example **"pages"** is a pyrocms core module that handles the pyroCMS page element

PyroCMS Custom Modules

Custom Modules are placed under projectname\addons\default\modules folder

e.g htdocs\pyrocms\addons\default\modules\helloworld

pyrocms – our project name
helloworld – the module we are going to create.

Remember the location of both the Core and Addons modules and their respective directories.

PyroCMS Module – Files Required?

PyroCMS requires at least a details.php and one controller inside the module-name folder. I will explain file requirements for different types of module creation. Filenames and folders, are understood easily if you know Codeigniter MVC conventions. If you are not familiar with CI and are trying to build a PyroCMS module, it is still fairly easy to grasp the folder structure . You can have the following folders inside a custom module folder - config, controllers, helpers, libraries,

models, views, js, css, img,language,widgets etc (See screenshot showing folder structure)

1. helloworld\details.php

This file contains some useful configuration options that help PyroCMS understand our new module. Introduce your module here and Say "Hello".

2. helloworld\controllers\helloworld.php

The name of the controller (file name & class name) keep them same as module name

If you need a module just for the front end and nothing is required for the backend then, you'd need at least one controller, and extend the controller class from Public_Controller.

If you need to access a module from the backend (like CRUD interface in backend), you'll need a controller class helloworld\controllers\admin.php which would extend Admin_Controller

When I write a file name like helloworld\controllers\helloworld.php I expect, you have created a helloworld.php controller inside the controllers folder, inside our helloworld module.

3. helloworld\views\helloworld.php

View file which would be used for displaying data from the module controller

If you don't need to access a database for your module, then the 3 files above will suffice for creating the module, otherwise you will need a model class. Take your time in understanding the PyroCMS folder structure!

If you need backend admin controllers in step 2 above, then think about placing its views too.
Let's place some view files for the admin side in the views\admin folder.

4. helloworld\views\admin\index.php

Pass data to this view from the admin controller

5. helloworld\views\admin\sidebar.php

Sidebar partial that's shown in left sidebar of the default backend theme.
Partial is just a few html tags and php code together. think of it as general "view file" . For eg. the box in the sidebar of other modules you see while browsing in admin.

6. helloworld\models\helloworld_m.php

If we require database access,then we need to implement the business logic inside one or more models.

helloworld_m.php <— note the_m, this is a convention used for a models. The class name would be Helloworld_m .
You can create more model files according to your needs.

Don't forget to place these files in theor respective Models/Views/Controllers folder according the folder structure in the figure (at end of this section below) .

1. helloworld\details.php
2. helloworld\controllers\helloworld.php
 -----> helloworld\controllers\admin.php [controller for admin interface]

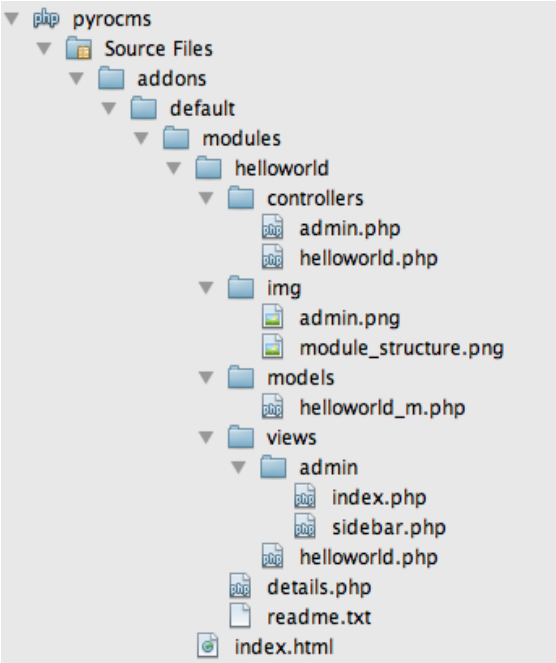
3. helloworld\views\view_file.php
 -------> helloworld\views\admin\index.php –> admin main view
 ------> helloworld\views\admin\sidebar.php –> admin sidebar partial

4. helloworld\models\helloworld_m.php

If your module is complex, you will end up with many more files, that's okay.

PyroCMS module file/folder structure is MVC, just bundled within a module folder. That's what makes PyroCMS HMVC (Hierarchical Model View Controller). PyroCMS is codeigniter based, if you know codeigniter fairly well, the you have the option to understand HMVC and build your own CMS ;) Good Luck.

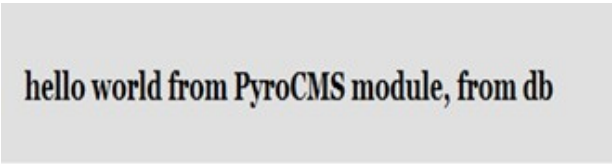
Here's the PyroCMS helloworld module folder structure:

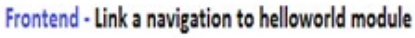


PyroCMS Module – Frontend/Backend Snapshot

Let's have a look at what our module will look like in both the front-end and the backend

Frontend: <h1><?php echo $msg; ?></h1> that's it



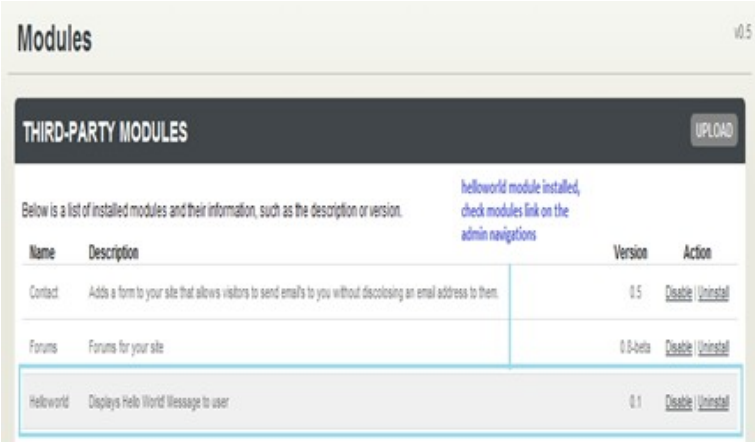


(to see a module in action, after installing the helloworld module, create a link in the header navigation to our module)

Section of what you will see once you have added the navigation link:

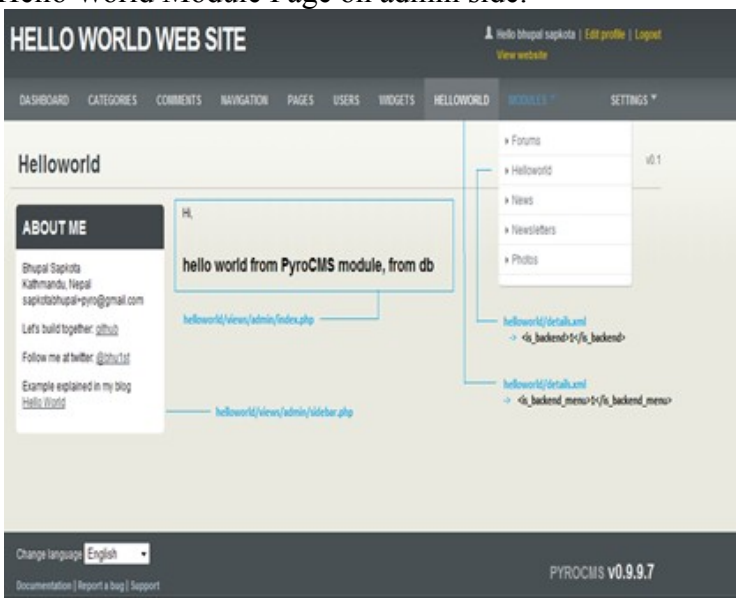
Backend:

Module installed and listed in the Admin –> Modules-> Third party Section (we don't create views for this listing, PyroCMS automatically lists our new module there if it was successfully installed, the information is taken from details.php)



We have – index.php and sidebar.php for admin side view,
when we build template from these two views that will look as below on the admin side.

Hello World Module Page on admin side:



Did you notice the – left sidebar? main content? helloworld link on the admin navigation, link to hello world module in the drop-down modules list ?

PyroCMS Module – Coding the "helloworld" module

The code is just for illustration purposes. My idea is to help you create and install your own custom module.

1. helloworld\details.php
2. helloworld\controllers\helloworld.php
    a. ----> helloworld\controllers\admin.php [controller for admin interface]
3. helloworld\views\view_file.php
    a. -------> helloworld\views\admin\index.php –> admin main view
    b. ------> helloworld\views\admin\sidebar.php –> admin sidebar partial
4. helloworld\models\helloworld_m.php

We must create the following files: 1 (details.php) + 2 (controllers) + 3 (views) + 1 (model file) = 7 files in total and respective folders Models\Views\Controllers and place them inside the MVC folder structure.

1. helloworld\details.php

A. basic file / when no backend and MySQL required:

-> When we don't need custom table for our module in database or when we are just working on PyroCMS tables to access data, we will have a minimal information in details.php.

B. Advanced file / when backend of module exists and our module will have its own database table to operate:

Found it to only work with CI Active Record Syntax ??

```
   public function install()
  {
     $fields = array
            (
                    'id' => array(
                    'type'=> 'int',
                    'constraint'=> 11,
                    'autoincrement'=> TRUE,
                    ),
                'msg' => array(
                    'type'=> 'VARCHAR',
                    'constraint'=> 255,
                     )
          );

       $this->dbforge->add_key('id', TRUE);
     //gives PRIMARY KEY `id` (`id`)

/*After the fields have been defined, they can be added using:
 $this->dbforge->add_field($fields); followed by a call to the create_table() function.*/
          $this->dbforge->add_field($fields);
          $this->dbforge->create_table('hello_world', TRUE);
          // gives CREATE TABLE IF NOT EXISTS table_name
              $data = array(
              'msg' => 'Hello World from PyroCMS module, from database',
             'NULL' =>' Greetings from @bhu1st '
            );

       $this->db->insert('hello_world', $data);
```

Name of the admin controller and its methods

We add code below when we create module that have backend interface.
controller name=”admin”> tells we are going to create a

helloworld\controllers\admin.php - controller

The controller that will extend Admin_Controller and we would have one method named index in it.

MySQL Queries required at module install time

Before explaining this I want you to read and remember the basic contents of the details.php file!

PyroCMS module install routine.

View the contents of details.php

```
<?php defined('BASEPATH') or exit('No direct script access allowed');
class Module_HelloWorld extends Module
{
 public $version ='1.0';
  public function info()
 {
   return array(
        'name'      => array(
           'en'=>'Hello World',
        ),
        'description' => array(
           'en'=> 'This is a Hello World Module'
        ),
        'frontend' => TRUE,
        'backend'  => TRUE,    //CRUD for database being used!
        'menu'     => 'content'// will only appear in admin menu
        );
 }
  public function install()
 {
    $fields = array
            (
                'id' => array(
                 'type'=> 'int',
                 'constraint'=> 11,
                 'autoincrement'=> TRUE,
                 ),
             'msg' => array(
                 'type'=> 'VARCHAR',
```

```
                    'constraint'=> 255,
                     )
          );
      $this->dbforge->add_key('id', TRUE);
    // gives PRIMARY KEY `id` (`id`)

/*After the fields have been defined, they can be added using $this->dbforge->add_field($fields); followed by a call to the create_table() function.*/
            $this->dbforge->add_field($fields);
            $this->dbforge->create_table('hello_world', TRUE);
            // gives CREATE TABLE IF NOT EXISTS table_name
            $data = array(
               'msg' => 'Hello World from PyroCMS module, from database'
             );
      $this->db->insert('hello_world', $data);
    }
      public function uninstall()
  {
     //Drop the table whn uninstalling the module
     $this->dbforge->drop_table('hello_world');
    }
  public function upgrade($old_version)
  {
     // Your Upgrade Logic
     return TRUE;
  }

public function help()
  {
     // Return a string containing help info
     // You could include a file and return it here.
     return "No help specified yet!";
  }
  }/*end of class file details.php */
```

Remember:
We first drop the table “hello_world” if it exists in the database, then we create and insert some data in the hello_world table. That’s it. Our table is created and populated with the supplied data, if module installation went fine, you should be able to see the default_hello_world table in PyroCMS database you setup before.

C. When your module is just accessed in backend, like Newsletter module etc..

In that case it is not necessary to have a model with database queries and the backend is set to FALSE in the details.php file

2. helloworld\controllers\helloworld.php

Code/comment explains itself, you may refer above sections too.

```
<?php  if (!defined('BASEPATH')) exit('No direct script access allowed');
/*
|-------------------------------------
| Publich Controller of our module
|-------------------------------------
| accessed from front end
|
| extends Public_Controller
*/

class Helloworld extends Public_Controller
{
   function __construct()
   {
      parent::Public_Controller();
   }
   function index()
   {
      //load model
      $this->load->model('helloworld_m');
      //get message from model
      $message = $this->helloworld_m->getHelloMsg();
      //pass message and build template
      $this->data->msg = $message;
      $this->template->build('helloworld', $this->data);
   }

}
```

2. a. helloworld\controllers\admin.php [controller for admin interface]

The Code:

```
<?php if (!defined('BASEPATH')) exit('No direct script access allowed');
/*
|-------------------------------------
| Admin Controller of our module
|-------------------------------------
| accessed from back end
| extends Admin_Controller
|
*/
class Admin extends Admin_Controller
{
   public function __construct()
    {
      parent::Admin_Controller();
      //load model
      $this->load->model('helloworld_m');
      //set views/admin/sidebar as 'sidebar' partial,
      //that is to be shown on Sidebar section of backend
      $this->template->set_partial('sidebar', 'admin/sidebar');
   }
    //Show Helloworld message to admin
```

```
    function index()
    {
        //function triggred when we click on the module name in backend in the menu
        //get message from model
        $message = $this->helloworld_m->getHelloMsg();
        //pass message and build template
        $this->data->msg = $message;
        $this->template->build('admin/index', $this->data);
    }
}
?>
```

3. helloworld\views\view_file.php

```
<h1><?php echo $msg; ?></h1>
```

3.a. helloworld\views\admin\index.php - admin main view

```
Hi,
<br/> <br/>
<h2><?php echo $msg; ?></h2>
```

3.b, helloworld\views\admin\sidebar.php –> admin sidebar partial

see how it is set as sidebar partial in constructor of our Admin controller **2. a. helloworld\controllers\admin.php**

```
  $this->template->set_partial('sidebar', 'admin/sidebar');
```

this view file would contain following html:

```
<div class="box">
    <h3>About Me</h3>
    <div class="box-container">
        Bhupal Sapkota<br/>
        Kathmandu, Nepal<br/>
    </div>
</div>
```

I’ve put a little bit info about me, never mind. Put any <html> or Pyro stuff you like there.

4. helloworld\models\helloworld_m.php – our model class

```
<?php if (!defined('BASEPATH')) exit('No direct script access allowed');
/*
|-------------------------------------
| Model Class of our module
|-------------------------------------
| for database access
|
*/

class Helloworld_m extends CI_Model
{
    function __construct()
    {
        parent::__construct();
    }
```

```
function getHelloMsg()
    {
        //get hello_world table
        $query = $this->db->get('hello_world');
        //if module successfully installed and data exists in table, grab it, return it
        if($query->num_rows() > 0){
            $result = $query->row_array();
            return $result['msg'];
        }else { //otherwise return simple hello message
            return "Hello world from PyroCMS Module!";
        }
    }
}
?>
```

Install Helloworld Module

->Download code from [Helloworld PyroCMS Module in Github]

Zip your module, and upload it from the Modules –> Addon section, Upload link

If your MySQL queries had no problems, you'd see hello_world table added in PyroCMS database. Helloworld module will be listed in admin navigation, modules list, in third party section.

When you uninstall

The module gets deleted from addons\default\modules folder. Remember to think of making a backup before you uninstall the table from the database!

Module In Action

Access module from Admin Navigation menu or type this in browser:

Backend:

http://localhost/**projectname**/admin/**modulename**

http://localhost/pyrocms/admin/helloworld

Frontend:
Assign one frontend navigation link to point our helloworld module

http://localhost/**projectname**/**modulename**

http://localhost/pyrocms//helloworld

Output Check

I have added validation in our model Helloworld_m,

```
function getHelloMsg()
    {
        //get hello_world table
        $query = $this->db->get('hello_world');
        //if module successfully installed and data exists in table, grab it, return it
         if($query->num_rows() > 0){
```

```
            $result = $query->row_array();
            return $result['msg'];
        }else { //otherwise return simple hello message
            return "Hello world from PyroCMS Module!";
        }
    }
```

If SQL executed successfully during module install, the $msg in view renders Addon Module, **“Hello World from database table”** otherwise **"Hello world from PyroCMS Module!"**;

Conclusion

1. follow the file/folder structures and naming conventions.
2. Understand the module creation requirement and accordingly settings in details.php
3. Check your queries in database before saving them.
4. Upload your module to install it. [remember to upload zip file, zip file name same as module name]
[helloworld.zip] or [yourmodule.zip]
5. If upload failed, delete raw upload from third_party modules folder, database column if any, check your settings, check queries again, zip it, and re-upload.
5. If still couldn’t install it:
--> i think you now understand the module creation flow
--> insert few queries in database [details.xml]
-> add your module settings in modules table in database [details.xml]
-> don’t wait for any other documentation ;) the kickass CMS, created over the kickass framework is really easy to grasp in one shot back tracing, 2-3 hours max
-> hack it use it. contribute back if any.
-> if you are familiar with Codeigniter and not with HMVC <------ read