**Root directory Structure of the ZEND Framework 2 Project**

Accessing the root directory, in addition to the archives of composer and autoloader, we have the following directories:

–Config  
–data  
–module  
–public  
–vendor

Now we just try to describe the function of each directory in the proposed organization by Zend. See the directories:

**– Config:** This directory is now self-explained. Here are settings in the application as a whole. Within this we have yet another folder, “autoload”, where it is suggested that they are global to local files (development) configurations (production) and. If you use Git, no file by default on this folder that ends with “. Local.php” will be synchronized in your commits.

**– Data:** Here you can store your files that are not part of the application code, but others related to the project, such as diagrams, database dumps, etc …

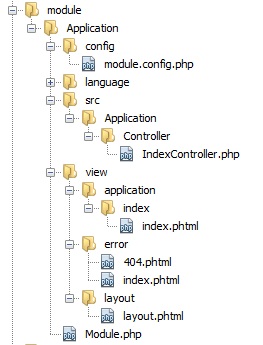
**– Vendor:** dedicated to third-party libraries directory. If you are using the Bootstrap, jQuery, Libraries for e-mail or otherwise, this is the folder where they should stay. Note the existence of the Composer and Zend already over there …

**– Public:** Zend Framework works with FrontController, ie everything that you access, you are accessing via a path only. Here, and only here, that users of your application will access. Your images, CSS and JS files among others should stay here in this directory. The organization and distribution of the files within this directory is up to you.

**– Module:** The directory where are the modules of your application. The Zend Framework, now in its first version, aimed to be modular. However, due to many factors (the Zend itself and also programmers using the framework), this brick structure was never put to good use. Starting with version 2, Zend Framework made ​​it clear that it should be implemented by modules within the application, and this has been respected to the letter. Then you ask yourself: ” *My application is small, use the ZF will create only one module. Worth I use it?* “The short answer would be” YES! “because with this architecture, if your application needs expand, you will have complete freedom within the same structure to implement the growth of it. The ideal question here would be: ” *Do not think I could take advantage of the capabilities of ZF, and I will create a simple application with no intention to expand it. Should I use ZF in this case?* “Then yes, the most correct answer would be negative. No one should use the framework just because it exists. The implementation of advanced codes with it becomes much simpler, but to implement basic things work there is to configure and assemble the structure sometimes does not pay.

**Going deeper into ZEND Framework Project Folder**

Well, by now you already have a preview of the initial structure of **Zend Framework 2** . Let’s go into the folder “module” at this time. This is the main directory of your application. Here you will get the modules you will develop. Note the existing structure here:

[](http://www.w3programmers.com/getting-started-with-zend-framework-2-part-1-project-structure/module-folder-structure/)

config

-autoload

-local.php

-global.php

-application.config.php

module

-Application

-config

- module.config.php

-src

-Application

-Controller

-AlbumController.php

-IndexController.php

-Model

-View

-Helper

-Myviewhelper.php

-Forms

-view

-application

-album

-index.phtml

-index

-index.phtml

-error

-404.php

-layout

-layout.phtml

-module.php

-Album

public

-index.php

vendor

index.php

init\_autoloder.php

The structure is interpreted to be complicated at first, but we will navigate through it step by step and you will see that it is well organized.

 Let us begin with the folder **Application** . Folders at this level are modules of your application.The module name should be assigned to the directory. For each new module, a new directory must be created. Simple as that. Deepen far this base module available from Zend …

Within the Application, we have three main directories: ” *config* “, ” *src* ” and ” *view* “.  Understand each:

**Application / config**

Here are the files independent configuration of this module. Note that Zend worked well this question, since there is a directory above with general settings for the application, and this which are specific settings “Application” module.

**Application / src**

Here are the codes of your application. Note that this directory is inside a new directory named “Application”. This happens because of namespaces. Within one (Application) module can have multiple namespaces (Application / src / Application, Application / src / model, and so on …). Do not consider this sub-modules, but a better way to organize your code.

Within the Application / src / Application directory, note the existence of “Controller” folder.There are controllers (oohhh!!). At the same level of “Controller” directory would be the “Model” directory for files that deal tables your database. On this basis of ZF, for teaching purposes, this was not implemented.

**Application / view**

The ZF works in layers (MVC architecture). Here in this folder are only files of data visualization layer. See that just below “view” we have a “application” directory. She refers to the “Application” namespace within the “src” folder. As soon as the example given above, this level would also have a directory called “model”.

Inside the “application” we have another directory called “index” and within it a “index.phtml” file called. Explaining: The directory “index” refers to the controller “IndexController” (in this case, if we had a contest called “TesteController”, the directory at this level would be called “test”) and “index.phtml” file refers to the action “indexAction “within the Controller. The extension “. Phtml” is standard in Zend Framework.

At the same level of “application” we also have the folders “error” and “layout”. The first is self-explanatory, own files to display errors during execution and also error page not found (404).The second has the layout of your application. The header, footer and everything is defaults on all pages of your application are defined in this file. The Zend Framework is very flexible as to this, as it allows the creation of multiple layouts for each module, and also allows you to disable the layout in any single part of the application.

**The Heart of the Module**

The file “Module.php” then appears to close this structure. It is within the “Application” module directory, and has the responsibility to control the module in question. It works as a “front controller” for the module. Before you do anything in this module, Zend gives just pass by “Module.php”.