

Zend Framework Default Project Structure - Wil Sinclair

<ac:macro ac:name="unmigrated-inline-wiki-markup"><ac:plain-text-body><![CDATA[

<ac:macro ac:name="unmigrated-inline-wiki-markup"><ac:plain-text-body><![CDATA[

Zend Framework: Zend Framework Default Project Structure Component Proposal

Proposed Component Name	Zend Framework Default Project Structure
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend Framework Default Project Structure
Proposers	Wil Sinclair Liaison: Ralph Schindler
Revision	0.1 - January 21, 2008: First draft. 1.0 - January 30, 2008: Final draft for review. (wiki revision: 23)

Table of Contents

1. Overview
2. References
3. Component Requirements, Constraints, and Acceptance Criteria
4. Dependencies on Other Framework Components
5. Theory of Operation
6. Milestones / Tasks
7. Class Index
8. Use Cases
9. Class Skeletons

1. Overview

This proposal covers the Zend Framework default project structure. This will be the project structure generated by default when using Zend_Tool. It will also serve as our recommended project structure for basic projects that do not call for more specialized project organization and in environments where the developer has appropriate file system/web server admin privileges.

2. References

- [Ruby on Rails Cheat Sheet](#)
- [CakePHP Cheat Sheet Draft](#)
- [Rails and Django Project Structure Comparison](#)

3. Component Requirements, Constraints, and Acceptance Criteria

- This project structure **must** support all project requirements for core components in an environment where the developer has appropriate file system and web server admin privileges.
- This project structure **will not** support all project requirements or constrained environments where, for example, apache directives can not be overridden.
- This project structure **must** reflect all PHP application security best practices, including minimizing code under the web server document root.
- This project structure **will** require that URL rewrite is enabled on the web server to work properly and properly secure the application.
- This project structure **must** be extensible/expandable to handle new requirements.
- This project structure **should** minimize the complexity of the project directory for the given project. For example, if the project does not use Zend_Search_Lucene, an index folder should not be recommended.
- This project structure **should** allow modules to be added to a non-modular application without having to manually update the project directory structure.
- This project structure **should** be consistent in best-practices recommendations with the project structures that have been documented in the reference guide.

4. Dependencies on Other Framework Components

5. Theory of Operation

Many developers seek guidance on the best project structure for a Zend Framework project in a relatively flexible environment. A 'flexible' environment is one in which the developer can manipulate their file systems and web server configurations as needed to achieve the most ideal project structure to run and secure their application. The default project structure will assume that the developer has such flexibility at their disposal.

The following file/directory structure is designed to be maximally extensible for complex projects, while providing a simple subset of folder and files for project with simpler requirements. This structure also works without alteration for both modular and non-modular ZF applications. The .htaccess files require url-rewrite functionality in the web server as described in this document: [Rewrite Configuration Guide](#).

It is not the intention that this project structure will support all possible ZF project requirements. The default project profile used by Zend_Tool will reflect this project structure, but applications with requirements not supported by this structure should use a custom project profile.

6. Milestones / Tasks

- Milestone 1: Gather community feedback and finalize project structure.
- Milestone 4: Document the project structure in reference guide.

7. Class Index

8. Use Cases

These use cases will necessarily be very high level.

UC-01

The user creates a project without specifying a project profile.

The system finds the default project profile and creates the components of the project structure below necessary for a basic MVC ZF application.

UC-02

The user creates an index for Zend_Search_Lucene in his/her existing project.
The system creates the required index directory for the Lucene indexes and creates necessary files to facilitate index building.

9. Class Skeletons

The following reflects the file system structure of the proposed default project structure:

```
<project name>/
application/
  services/
  configs/
  controllers/
  helpers/
  layouts/
  filters/
  helpers/
  scripts/
models/
modules/
views/
  filters/
  helpers/
  scripts/
Bootstrap.php
data/
  cache/
  indexes/
  locales/
  logs/
  sessions/
  uploads/
docs/
library/
public/
  css/
  js/
  images/
  .htaccess
  index.php
scripts/
  jobs/
  build/
  temp/
  tests/
```

Please note that the directory layout described here can be duplicated under modules with the exception of the Bootstrap.php.
Justifications for the above.

application/ - This directory contains your application. It will house the MVC system, as well as configurations, services used, and your bootstrap file.

services/ - This directory is for your application specific web-service files that are provided by your application.

configs/ - The application-wide configuration directory

controllers/models/views/ - These directories serve as the default controller/model/view directories. Having these three directories inside the application directory provides the best layout for starting a simple project as well as starting a modular project that has global controllers/models/views.

controllers/helpers/ - These directories will contain action helpers. Action helpers will be namespaced either as "Controller_Helper_" for the default module or "<Module>Controller_Helper" in other modules.

data/ - This directory provides a place to store application data that is volatile and possibly temporary. The disturbance of data in this directory might cause the application to fail. Also, the information in this directory may or may not be committed to a subversion repository. Examples of

things in this directory are session files, cache files, sqlite databases, logs and indexes.

layouts/ - This layout directory is for MCV-based layouts. Since Zend_Layout is capable of MVC- and non-MVC-based layouts, the location of this directory reflects that layouts are not on a 1-to-1 relationship with controllers and are independent of templates within views/.

Bootstrap.php - This file is the entry point for your application. The purpose for this file is to bootstrap the application and make components available to the application by initializing them. This file should not call dispatch() on the front controller.

modules/ - Modules allow a developer to group a set of related controllers into a logically organized group. The structure under the modules directory would resemble the structure under the application directory minus the Bootstrap.php file.

docs/ - This directory contains documentation, either generated or directly authored.

library/ - This directory houses the library/Zend/* folder which contains Zend Framework. Developers should also place their application's library code under this directory in a separate namespace- ie, a namespace that does not begin with 'Zend'.

scripts/ - This directory contains maintenance and/or build scripts. Such scripts might include command line, cron, or phing build scripts that are not executed at runtime but are part of the correct functioning of the application.

temp/ - The temp folder is set aside for transient application data. This information would not typically be committed to the applications svn repository. If data under the temp/ directory were deleted, the application should be able to continue running with a possible decrease in performance until data is once again restored/recached.

tests/ - This directory contains application tests. These could be hand written, phpunit tests, selenium based tests or based on some other testing framework. By default, library code can be tested by mimicing the directory structure of your library/ directory. Additionally, functional tests for your application could be written mimicing the application/ directory structure (including the application subdirectory).

public/ - This directory contains all public files for your application. index.php sets up the PHP environment, include the Bootstrap.php file from your application/ directory and finally dispatch() the front controller. The web root of your web server would typically be set to this directory.

]]></ac:plain-text-body></ac:macro>

]]></ac:plain-text-body></ac:macro>