* 1 [Classical (Unix/Linux-like)](http://framework.zend.com/wiki/display/ZFDEV/Choosing+Your+Application%27s+Directory+Layout#ChoosingYourApplication%27sDirectoryLayout-Classical%28Unix%2FLinuxlike%29)
* 2 [Classical Extended](http://framework.zend.com/wiki/display/ZFDEV/Choosing+Your+Application%27s+Directory+Layout#ChoosingYourApplication%27sDirectoryLayout-ClassicalExtended)
* 3 [Classical Extended Embedded](http://framework.zend.com/wiki/display/ZFDEV/Choosing+Your+Application%27s+Directory+Layout#ChoosingYourApplication%27sDirectoryLayout-ClassicalExtendedEmbedded)
* 4 [Conventional](http://framework.zend.com/wiki/display/ZFDEV/Choosing+Your+Application%27s+Directory+Layout#ChoosingYourApplication%27sDirectoryLayout-Conventional)
* 5 [Conventional Modular](http://framework.zend.com/wiki/display/ZFDEV/Choosing+Your+Application%27s+Directory+Layout#ChoosingYourApplication%27sDirectoryLayout-ConventionalModular)

**Classical (Unix/Linux-like)**

Initially broken down into two main directories, web-readable and application files. This style mimics Unix/Linux OS directory layouts, and would be preferred by someone who favors Unix/Linux-style naming conventions.

/application

/etc

/lib

/Zend

/(other libraries)

/usr

/controllers

/models

/views

/var

/sessions

/cache

/view\_compiles

/htdocs

/images

/scripts

/styles

**Pros**

* Self-contained; application-related files that are not publicly-accessible are in one place
* Library files located locally (not server-wide)
* Strong layout when models, views, and controllers have overlap, or are interrelated

**Classical Extended**

/application

/config

/library

/Zend

/(other libraries)

/user

/controller

/model

/view

/variable

/sessions

/cache

/view\_compiles

/htdocs

/images

/scripts

/styles

index.php

**Pros**

* Same as Classical
* Shorter Unix/Linux-style names lib, usr, and var are replaced with full length names for clarity

**Cons**

* Same as Classical, except for the cryptic naming scheme

**Classical Extended Embedded**

In certain environments, moving the application-related files outside the web-readable directory might not be an option. In this case, the files are in the web readable directory and should be protected via an .htaccess file if at all possible, or some other related protection mechanism.

|  |  |
| --- | --- |
| http://framework.zend.com/wiki/images/icons/emoticons/forbidden.gif | **Security concern** Directly exposing application files to the outside world can potentially result in sensitive information about the application or server being revealed to attackers. |

/htdocs

/application

/config

/library

/Zend

/(other libraries)

/user

/controller

/model

/view

/variable

/sessions

/cache

/view\_compiles

.htaccess (deny from all)

/images

/scripts

/styles

index.php

**Pros**

* Useful in environments where htdoc directory is all a developer has access to

**Cons**

* This layout could present security issues if not properly configured

**Conventional**

/application

/config

/controllers

/models

/views

/htdocs

/images

/scripts

/styles

index.php

/library

/Zend

/(other libraries)

/tmp

/sessions

/cache

/view\_compiles

**Pros**

* With the exception of user created library code, application related code (as it relates to the domain model, controllers and views) is in one specific location.
* Can take advantage of a library installed globally on a system (like PEAR)
* As opposed to a Classical layout, focus on code containment is at the main directory level, thus creating a wider directory structure at the top most level.
  + Application directory in this layout means 'the code responsible for the direct duties of carrying out application specific actions'.
  + Library directory in this layout are librarys of code that carry out a specific and common pattern of duties, usually not related to the specifics of an application, but related to the carrying out of common tasks.
  + Tmp, or 'variable' directory, exists to aid the application in its duties that may require the use of a filesystem to create tmp files like session files, cache files or view compiles.
* When an available option, this is the suggested format of the Zend Framework.
  + Using the most common or most encouraged format aids future developers (that might inherit or use your software) by reducing thier time getting up to speed.

**Cons**

* Wider top level of directories. If your perception of *application directory* is not limited to the site specific files and includes libraries and system (tmp) files.
* Harder to move top files into an **Embedded** structure (within the htdocs directory) as each concrete directory must be secured against web readability, and routes by the same name as concrete directories cannot exist (easily).
* Certain applications might require custom (or extended) libraries, in that case, it is important to note that you not distinguish between 3rd Party libraries and user generated libraries. Library code is simply, regardless of source, library code.

**Conventional Modular**

In this style, models, views, and controllers for a specific application module are grouped into logical "packages", or directories.

/application

/config (optional)

/(module 1)

/config (optional as needed)

/controllers

/models

/views

/(module 2)

/controllers

/models

/views

/(module n)

/controllers

/models

/views

/htdocs

/images

/scripts

/styles

index.php

/library

/Zend

/(other libraries)

/tmp

/sessions

/cache

/view\_compiles

**Pros**

* same as Conventional
* Allows for packaging of non-interrelated application components (think plugin architecture of a web site with a blog, polls, quote generator).

**Cons**

* same as Conventional