

[<< Back to Dashboard](#)

ColdBox Directory Structure & Conventions

Contents

- [ColdBox Directory Structure & Conventions](#)
 - [Core Conventions](#)
 - [Directory Structure](#)
 - [Application Templates](#)
 - [Modifying Application Conventions](#)
 - [Implicit Execution Conventions](#)

Covers up to version 3.5.0

In order to create a ColdBox application you must adhere to some naming conventions and a directory structure. This is needed in order for ColdBox to find what it needs and to help you find modules more easily. Also, ColdBox provides you with the ability to change these conventions for your applications.

Core Conventions

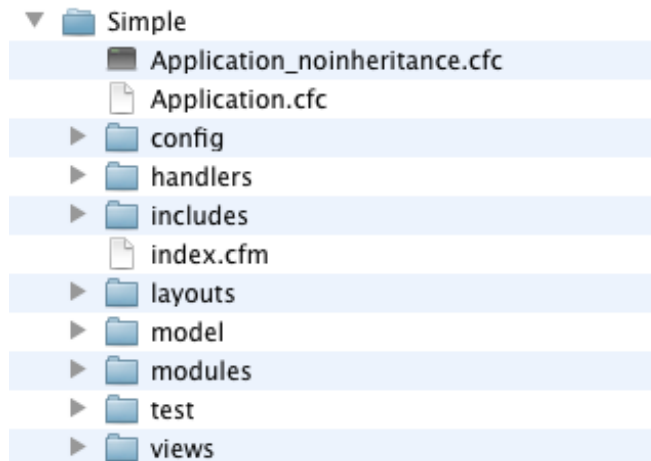
The core conventions delineate the contract between ColdBox and you for file/directory locations and more.

Convention	Default Value	Required	Description
config file location	<i>/config/Coldbox.cfc</i>	true	The location of the application configuration file
handlers location	<i>/handlers</i>	true	Where all event handlers are located
layouts location	<i>/layouts</i>	false	Where all layouts are located
views location	<i>/views</i>	false	Where all views are located
plugins location	<i>/plugins</i>	false	Where all plugins are located
models location	<i>/model</i>	false	Where all model objects are located

modules location	<i>/modules</i>	false	Where all modules are located
default event action	<i>index()</i>	false	The name of the default event action (method) to use when an event handler is called with no method
default event	main.index	false	The default event to execute when not defined in the configuration file and no incoming event is found

Directory Structure

Please note that the directory structure of your application relies on the conventions that you setup. So the structure changes according to the settings described above. Below is the simplest form of a ColdBox application directory layout:



Above you will see the **core conventions** plus some extra files and folders:

Folder/File	Required	Description
Application_noinheritance.cfc	false	The bootstrap approach that uses non-inheritance (See Bootstrapper)
Application.cfc	true	Used to bootstrap the framework application using inheritance (See Bootstrapper)

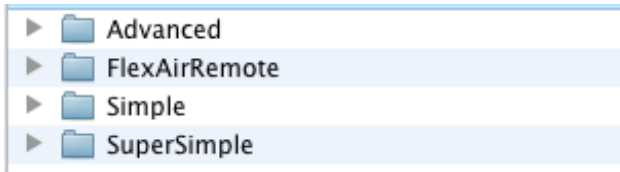
index.cfm	true	A placeholder file the framework needs for front controller operation. The file is usually empty.
config	true	Where your ColdBox configuration file goes, routing, and more.
handlers	true	Where your controller layer resides in the form of CFC event handlers
includes	false	Where you can place static assets like (css,images,i18n,js,etc)
layouts	false	Where your layout skins or themes go
model	false	Where your model layer resides
modules	false	Where all your ColdBox modules are placed.
test	false	Where all your unit, mocks and integration tests go (Yes, you need to test!)
views	false	Where your view layer resides

Read more about our Application.cfc [Bootstrapper](#)

Please read about our Bootstrapper to understand the difference between loading the framework using the standard inheritance Application.cfc or the non-inheritance approach. The major difference is that the non-inheritance approach allows you to locate coldbox via per application mappings.

Application Templates

The ColdBox **bundle** download includes several application templates that can help you get started with ColdBox. Also, the best way to create or generate applications is via our [ColdBox Platform Utilities extension](#) for Adobe ColdFusion Builder. Our bundle download includes the following application templates:



Modifying Application Conventions

The only convention you can not directly modify in the configuration file is well, the location of the configuration file. However, you can still do this by setting up the location via the [Application.cfc directives](#) (See [Loading a custom configuration file](#)). These per-app conventions increase the portability of the ColdBox application, as now you can override any setting a framework-wide installation might have implemented and you are guaranteed that they will work. All you need to do is add the following to the configuration file:

```
//Conventions
conventions = {
  handlersLocation = "handlers" ,
  pluginsLocation  = "plugins" ,
  viewsLocation    = "views" ,
  layoutsLocation  = "layouts" ,
  modelsLocation   = "model" ,
  modulesExternalLocation = "" ,
  eventAction      = "index"
};
```

We also encourage you to review the full [Configuration CFC guide](#) for an in-depth review.

Important All conventions that are locations require a relative location to the Root of the running application. They cannot be set outside the root.

Implicit Execution Conventions

ColdBox also provides you with several implicit execution conventions that are mostly used in your [event handlers](#). Basically by creating a method with a certain signature or name, ColdBox will call it for you. Below is just a listing of these method conventions.

- **preHandler(event,action,eventArguments)** - Occurs before any action event in a handler CFC
- **postHandler(event,action,eventArguments)** - Occurs after any action event in a handler CFC
- **aroundHandler(event,targetAction,eventArguments)** - Occurs around any action event in a handler CFC
- **onMissingAction(event,missingAction,eventArguments)** - Occurs when a requested action is missing in the handler CFC
- **onError(event,action,exception,eventArguments)** - Occurs when an

exception occurs within any action in the handler CFC

- **onDIComplete()** - Occurs after handler creation and dependency injection has been performed