<< Back to Dashboard

ColdBox Application.cfc Bootstrapper

Contents

- ColdBox Application.cfc Bootstrapper
  - Application.cfc Directives
  - Application.cfc Methods
    - Using Inheritance
    - No Inheritance

Covers up to version 3.5.0

The *Application.cfc* is one of the most important files in your application as it is where you define all the implicit ColdFusion engine events, session, client scopes, ORM, etc. It is also how you tell ColdFusion to bootstrap the ColdBox Platform for your application. There are two ways to bootstrap your application:

1. Making your *Application.cfc* extend the core bootstrapper *coldbox.system.Coldbox*
2. Adapting your *Application.cfc* to use composition and bootstrap ColdBox

The first two approaches are what we recommend you do to bootstrap your applications and most of the time you don't really have to worry about these files since they are either generated for you from our IDE tools or you can use the ones that ship with the platform's application templates. If you look in your download archive you should see the following:

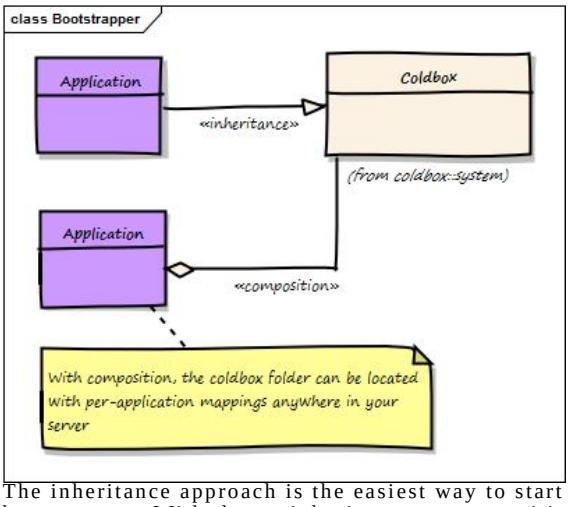
```
/ApplicationTemplates
  /Advanced
    + Application.cfc
    + Application_noinheritance.cfc
  /Flex
  /Simple
```

Those application templates are all ready configured to just drop, customize as you see fit and work. Obviously the inheritance approach is the easiest to use but it has its limitations that you cannot use per-application mappings to map ColdBox. Therefore, you can use the no inheritance approach which has a little bit of more boiler plate code provided for you. Next, lets see how we can control some aspects of our application via the ColdBox directives.

Application.cfc Directives

The provided *Application.cfc* object in the application templates can come in two flavors:

1. Using inheritance
2. Using Composition (**Our Preference**)



The inheritance approach is the easiest way to start but it is bound to extending the *Application.cfc* object to the ColdBox bootstrapper. With the no-inheritance or composition approach it is much more flexible and we can use per-application mappings for choosing the location of the **coldbox** distribution package. Anyways, once you make this decision there are some directives or variables you can set that will alter some behavior.

Variable	Type	Required	Default	Description
COLDBOX_APP_ROOT_PATH	string	true	getDirectoryFromPath(getCurrentTemplatePath())	Automatically set for you. This path tells the framework what is the base root location of your application and where it should start looking for all the agreed upon conventions. You usualy will never change this, but you can.

COLDBOX_APP_MAPPING	string	DEPENDS	---	The application mapping is ESSENTIAL when dealing with Flex or Remote applications. This is the location of the application from the root of the web root. So if your app is at the root, leave this setting blank. If your application is embedded in a sub-folder like *MyApp* then this setting will be */MyApp*.
COLDBOX_CONFIG_FILE	string	false	*config/Coldbox.cfc*	The absolute or relative path to the configuration CFC file to load. This bypasses the conventions and uses the configuration file of your choice.
COLDBOX_APP_KEY	string	false	*cbController*	The name of the key the framework will store the application controller under in the *application* scope.

Most of the time you will never alter these settings if your application is the root application and it will look like the following:

```
<---  COLDBOX STATIC PROPERTY, DO NOT CHANGE UNLESS THIS IS NOT THE ROOT OF YOUR COLDBOX APP --->
<cfset COLDBOX_APP_ROOT_PATH = getDirectoryFromPath(getCurrentTemplatePath())   >
<---  The web server mapping to this application. Used for remote purposes or static purposes --->
<cfset COLDBOX_APP_MAPPING   =  "">
<---  COLDBOX PROPERTIES --->
<cfset COLDBOX_CONFIG_FILE   =  "">
<---  COLDBOX APPLICATION KEY OVERRIDE --->
<cfset COLDBOX_APP_KEY       =  "">
```

Or you can modify it, like the following example in order to load a custom configuration file and a custom application mapping.

```
<---  COLDBOX STATIC PROPERTY, DO NOT CHANGE UNLESS THIS IS NOT THE ROOT OF YOUR COLDBOX APP --->
<cfset COLDBOX_APP_ROOT_PATH = getDirectoryFromPath(getCurrentTemplatePath())   >
<---  The web server mapping to this application. Used for remote purposes or static purposes --->
<cfset COLDBOX_APP_MAPPING   =  "apps.myApp" >
<---  COLDBOX PROPERTIES --->
<cfset COLDBOX_CONFIG_FILE   =  "shared.myApp.coldbox" >
<---  COLDBOX APPLICATION KEY OVERRIDE --->
<cfset COLDBOX_APP_KEY       =  "">
```

All of the directives have their appropriate getter and setter methods that you can use by calling it from the ColdBox bootstrapper object or via your *Application.cfc* using inheritance.

Application.cfc Methods

Apart from having some directives we also have several methods that we can interact with to control some other aspects of the bootstrapper.

- *setLockTimeout(seconds)* : You can override the default locking of the templates
- *getLockTimeout()* : You can get the default locking of the templates

Using Inheritance

If we are using inheritance then we can just simply call the methods directly as they are inherited in our *Application.cfc*

```
setLockTimeout(30);
```

No Inheritance

If we are using the composition approach then we need to talk to the bootstrapper via the object that gets created:

```
application.cbBootstrap.setLockTimeout(30);
```