

CFFrameworks Interview Notes

INTRO

My name is Luis Majano, I am a computer engineer, I am currently working for ESRI here in Redlands, California. I have been developing coldfusion applications for about 7 years now, since version 4.0. Most of you know me because of my open source projects like the ColdBox framework and my blog at www.luismajano.com.

My Definition of a Framework

My idea of a framework is that it is a solid software foundation on which I can build my applications on. A good framework, provides me with a set of reusable code and tools that I can use to increase my productivity, and it provides me with a standard that comes in very handy when working in a team environment.

Do you use frameworks for everything you build regardless of size and Scope? If not when would you not use one.

I don't use frameworks for everything I build. For some scenarios a framework would be more of a deterrent than a helping hand. Its like killing a fly with a canon. There are some cases where you only need two or three simple forms and good organization.

For projects where I can see them grow, where I am working with a team of developers and I need flexibility, I will turn to the framework.

How ColdBox was Born?

Coldbox started as a very specific framework for an enterprise application I was working for about 2 years ago. I never really meant to create an open source framework, but it all just started coming together as development continued for that application.

So I did my research for about 3-4 months, looking other coldfusion frameworks at the time, struts, and immersed myself in enterprise design patterns and started creating proof of concepts for all the features that I wanted to see on a foundation that could be used to build any web application.

I saw how every application needed certain programming aspects apart from the core functionality and basically started to come up with the idea of creating plugins or sets of reusable code that could be loaded on the fly in an application or even just declared in a configuration file.

I also had a lot of influence from Oscar Arevalo, a very good friend of mine. Actually, Oscar came up with the name ColdBox.

How long have you been working on it?

I started working on ColdBox around December 2005 and had a first release by July 2006

Where does my framework sit?

ColdBox is a Presentation layer framework. However, it is also a development toolkit of some sorts. So it crosses the border of not being just a framework.

What problem does it address particularly well?

First of All, **Documentation**. I believe that documentation should be one of the most important aspects of a community software application. If you can document really well a foundation like coldbox and give the developers all they need to quickly start with it, your ease of use and adaptation will improve. I am a huge fan of documentation and try to document as much as I can in order to really make the user understand how the framework works, what it can do for you, its limitations and its extensibility. Another big part of this is creating Sample applications and I try to create as much as I can in order for users to see real implementations out of the box.

Ease of Use

In ColdBox there is no need to write XML declarative logic for events, it makes it very simple to code and sometimes you don't even realize you are using a framework because you are basically working with CFC's. That is why some people don't even know anything about MVC and can dig right in to it, they just need the CFC exposure. ColdBox has a very basic core API that you interact with which makes it very unobtrusive to the developer.

Novel Features of ColdBox:

Aspect programming:

ColdBox comes bundled with a set of plugins that will help on every day software application tasks like bug reports and notifications, AOP file logging with auto-archiving, a way to sense your environment as either dev or pro, storage facilities for cluster environments, object caching, datasource declarations, web services integrations, and internationalization. And this is how ColdBox can be very **EXTENSIBLE** because they are all plugins that can be loaded on the fly and it allows the developers to have the ability to create small libraries which can be reused.

ColdBox introduces Handler packages where I can organize my handlers by application or task and coldbox auto registers them. This means that I can use the same application

skeleton for a blog, a forum, a website, etc, without writing any configuration files for each and trying to mesh them all together. There is no need for that, I can gradually make my application grow and extend its functionality just by dropping in more handlers and views to it. Drop and Play

ColdBox Dashboard application. Its a visual application that helps you configure your framework, it is self-documenting and also helps you upgrade to new versions of the framework.

ColdBox cache is new for 1.2.0 and it just simplifies object and data caching. You have several tuning parameters for the cache and also cache reports in the debugging panel. You can actually see how many objects and what type of objects are in your cache, the efficiency of your cache and the tuning parameters. I really think this will help developers save time and actually not have to worry about persistence.

XML Controller

Coldbox doesn't rely on xml declaritive logic where I have to define the event and what happens next. You basically expose methods on a handler cfc by turning their access to public or remote. The framework then will auto-register the handler cfc's and now you are able to use the methods. So the declaritive logic is now placed within the methods, where I place my method exit points, what business logic I call and what view to render or what event to surrender execution to. This is how ColdBox can help you create multi layered applications with one single skeleton and configuration file. So instead of me working with the configuration file all the time, I mostly am working with cfc's all the time. I just use the config file to setup my project or maybe tweak some settings.

Caching

For caching, I really wanted to take my time and prepare something useful and smart. It has some timeout tuning parameters such as default object timeout, default last access timeout and a reaping frequency. This is an in-memory cache designed for handlers, plugins and any other objects or data. You can choose to place timeouts on them or use the global default timeout the framework provides.

The cool thing about the last access timeout is that you can tell the cache to purge objects that have not been used for X amount of minutes. So the cache cleans itself out periodically using the reaping frequency. So every X minutes, when a request comes in that needs something from cache, it will run the reaping routing to clean old objects out.

And since I really wanted for users to know what was going on in order for them to tune their cache, I added a caching snapshot report to the debugging panel. It tells you the efficiency of the cache in hits vs miss ratio, it shows you how many objects exist in the cache, the free and total JVM memory and it even charts for you the hits vs misses and the actual types of objects in that are currently in the cache in a pie chart.

There is still a lot of improvements I would like to make for the cache for the upcoming final release.

How Ajax Works

Ajax integration with coldbox is very simple. The framework provides you with the tools to use the same way you code a normal page as to interact with ajax. I include a sample application called ColDbox reader that is a full Ajax application. It uses div replacements to update content. So I basically just post and do get's via Ajax as I would a normal coldbox event, `index.cfm?event=something`.

An important part of debugging ajax apps is debugging, and since coldbox provides you with AOP logging and tracing. You can include your logs and traces and be able to read your log files or get them via email. So AJAX integration is a breeze with coldbox.

Compare to other frameworks?

Well, that coldbox is a newer framework and that as you can see from the novel features it has it is not based on XML declarative logic but it relies entirely in CFC's and ColdBox implements a lot of software aspects as plugins.

How do I extend this framework?

You extend coldbox by using the plugin architecture built into it. This allows the developers to extend the functionality of the framework and make it their own. Not only that, but plugins can be completely reusable from application to application, so this is a great way to create libraries of reusable tasks that can be loaded on the fly if you are using ColdBox. You don't have to worry about persisting them or how to instantiate them, Coldbox does that for you.

How does integrate with RIA, plans to address?

I am not very familiar with FLEX yet and have not done any coldbox-ria integrations as of yet, but it is something that I look forward to doing my research on.

Weakness?

I believe that one of the weaknesses that coldbox has, is that it has not been around for that long. It is not even 1 year old as an open source framework. However, due to its uniqueness and ease of use, the downloads and community support has been very impressive. I believe that the framework has matured in lightning speed and it is proving to be a very reliable software foundation.

EXTRA

The download gives a bad first impression because its so big. Users might get discouraged at the amount of files the download comes with, but it is mostly documentation and sample applications. So give it a shot.

Remove or Re-Design

I think I have done that for this upcoming 1.2.0 release with caching and the new event bus. I really took my time to model the entire framework in UML and go section by section and try to optimize it as much as I could. And the good thing is that I did not do it alone this time, but had the help from the community in doing so. So I am really thankful to Oscar Arevalo, Sana Ullah, Aaron Conran and Rob Gonda for helping me out in the design of 1.2

However, something that still bothers me that I believe can be really improved on and really make it much more compact is the configuration file. I want it to be even easier than what it is already.

Plans for Futures

Well, I want to do so much but the short term goals would be to create a plugins community directory where developers can share and contribute. This would be very beneficial to the community.

Another big area that I want to concentrate on is the new coldbox caching manager. I want to make it even easier for users. Also a way to fine tune it and introduce more tuning parameters that the developer can use. And also introduce event caching. Which will cache your event's output and I believe will really accelerate your applications.

I am also doing research in interceptor patterns and filters, and trying to create a game plan of how to introduce them to ColdBox.

I am also writing a hands on book for ColdBox. I started writing it about 6 months ago and want to have it ready in a few months. I then just have to look on how to publish it.

Finally, I am also planning a commercial version of the dashboard, that will really concentrate on helping developers do what they do best and that is code. It will contain a lot of tools to make developing coldbox applications really easy and fast, but I won't go into detail as of yet.

Why use the framework?

Its hard to point out just a single reason, but one of the major reasons that I have gotten from coldbox users is its ease of use. How easily you can start delving into it, just by learning a few core API methods and learning where to put your stuff really accelerates the learning curve. And since it is based on CFC's, its just a matter of starting to type coldfusion code. The configuration file is basically set up from the start, so you can easily set up a working hello application in less than a minute. And since its based on cfc's it gives you the flexibility to expand your application by just dropping in more handlers and views.

Tag: cfcomponent