ColdFusion MVC Framework Analysis

Authored by Jon Messer
Senior RIA Architect
June 9th, 2008

Background:

- In preparation for a large scale insurance focused Enterprise CRM application project, Amcom Technology has been evaluating options in order to finalize and propose the recommended technology stack that would best meet our customer's requirements.
- One component to that analysis was the determination of the technologies and methodologies that are to be used within the ColdFusion space – in this case CF MVC Frameworks.
- Key considerations factored in the aggressive timeline of the project (i.e. learning curve needed to be low as possible), the varying degrees of skills within the team (from maintenance developers to architects), and scalability.
- A quick one week high level analysis was done to evaluate the CF MVC Framework options to determine which Framework best matches the needs of the project. As part of that research, a simple application was built where the same Model was used in each Framework, including a Flex front-end.
- Additional technologies used in the evaluation include ColdSpring and Transfer.
- Design patterns (DP) utilized include the Service Layer DP, and Data Access Object/Gateway DP.
- This document reflects the results of this research.

Goals:

- The goal of this document is to evaluate the major ColdFusion MVC Frameworks.
- Determine their relative strengths and weaknesses.
- Guide how and where to integrate them into our technology stack.

Problem definition:

- The need to decouple business logic from presentation logic.
- Mixing business logic and presentation logic adds unnecessary complexity both in terms of understanding the code and in terms of modification and maintenance.
- Maintenance of monolithic CFM templates becomes increasingly difficult with time and change.

Summary of Proposed Technology:

- The recommended option is the ***ColdBox*** MVC Framework.
- By isolating business logic from user interface considerations in an application, it becomes easier to modify either the visual appearance of the application or the underlying business rules without affecting the other. This allows for more modular and maintainable code.
- **Costs:**
  - All of the frameworks evaluated are open source with no licensing costs.
  - Support and updates are provided by the community and Framework authors.
  - There is also commercial support and training available for each of the frameworks evaluated.
- **Evaluation Notes:**
  - All of the Frameworks evaluated demonstrated high degrees of performance and scalability through advanced caching capabilities.
  - The learning curve is relatively the same for each; however ColdBox presents a distinct advantage with its extensive documentation set which thereby reduces the learning curve.
  - The ColdBox Framework further distinguishes itself with an abundant feature set, the ability to hook in your own Framework extensions at a very granular level.

Benefits/Advantages/ROI of MVC Frameworks:

- Development Velocity ;
  - A standardized environment which facilitates team based development (and thus natural knowledge transfer).
  - Reduced dependency burden on high degrees of specialty application knowledge.
  - External resources knowledgeable in a Framework can be brought in with little or no need for Domain specific knowledge.
  - Reduced coding time spent on the Framework.
- Strategic allocation of Development resources:
  - Allows for most development effort to be spent on the 2 areas that matter: the interface and the business logic.
  - Reduced developer training costs and learning time.
- Risk management:
  - These Frameworks are widely used, thus ensuring a thoroughly tested code base vs. a homegrown Framework.
  - The Frameworks help the standardization of the application, which can reduce the chances of a defect, especially if Unit testing and Regression testing is implemented.
  - Frameworks with active development are viewed as more favorable.
- Business Value:
  - More maintainable code, substantially reduced maintenance costs.
  - By separating the business logic from the presentation, the level of technical expertise for the presentation tier is essentially reduced to that of design skills.
  - Knowledge of the Framework becomes an off the shelf commodity
  - Ability to bring in outside contract support.

Drawbacks/Limitations/Risks:

- Any CFC heavy application in ColdFusion has a large up front load time from cold server restart.
  - Mitigated by:
    - Application start up should only happen on server restart.
    - Sun working on fixing Class Loader bug in Java 6, beta fix available.
- Performance: there is added overhead when using a Framework (even your own).
  - Mitigated by:
    - All of these Frameworks when properly designed and implemented have proven themselves capable of handling extremely high volume sites due to their caching capabilities.
    - For example BroadChoice handles 20,000 concurrent sessions.
    - By using production settings along with caching, these Frameworks perform better than non-framework code.
- By Relying on third party solution problems could arise that are not of our making.
  - Mitigated by:
    - large community support.
    - Active development by Frameworks authors.
    - Open source, so we can fix it ourselves.
    - Commercial support also available.

How well does it fit into the existing technology stack?

- All of the Frameworks are 100% compatible with Coldfusion 8.
- All can function as a backend service to Flex.
- They can be deployed alongside existing code without requiring any change to the existing code (but the existing code would get no benefit until it was re-engineered)[

How well does it leverage the existing skills of the team?

- Required Skills :
  - Object Oriented Programming concepts
  - Understanding of Event driven programming (similar to Flex and JavaScript).
- Existing Resources:
  - 25% of the staff has extensive knowledge in Model-Glue.
  - This experience is easily transferred from one Framework to another.

Timeframe Considerations:

- Basic understanding of these technologies can be had in a matter of days.

Options:

Option 1 - Coldbox (RECOMMENDED OPTION):

- **Summary**
  - Relatively new (3 years old) explicit invocation Framework, favors convention over configuration, many built-in tools, and extensive documentation.
- **Risks/Limitations:**
  - While this Framework is growing in popularity and getting a lot of new users, it's primarily supported by 4 developers (potentially it could stop being supported).
- **Mitigating Risks:**
  - It is open source so one could maintain it yourself.
  - There is an ever growing community using and supporting it.
  - ***NOTE:** A well designed Model decouples its dependency on any particular MVC framework, so if needed you could switch to another Framework (which is true for any of the options).
- **Plugins**
  - AOP Error Logging.
  - SES Interceptor for Pretty URLs, Environment Detection Interceptor.
  - IoC Framework Integration: ColdSpring & LightWire.
  - WebService Declarations & Abstraction.
  - File & Java Utilities.
  - Custom Exception Handling.
  - Environment Control Interceptor.
  - ColdBox Logging Facilities with auto-archive rotation.
- **Pros:**
  - *Excellent Documentation*, the most extensive and detailed of all these MVCs.
  - Very passionate community, the author responds to questions/bug reports within hours
  - A lot of built in functionality that goes beyond vanilla MVC Frameworks (logging, i18n, cache, SES, etc…).
  - Eclipse extensions for code completion and documentation.
  - Integrated unit testing.
  - The most amount of extension points and Framework hooks. Virtually every aspect of the Framework was designed to be custom extended without modify the core framework.
  - Published and actively maintained roadmap.
- **Cons:**
  - Explicit invocation, this isn't necessarily a con but some situations are easier to solve with implicit invocation.
  - Not as large a community as ModelGlue or Mach II yet.
- **Cost & Effort:**
  - There are no hard costs.
  - Estimated Development training time for basic knowledge at 16 hours.
  - Due to the larger API and extensive built in features, mastery of all the optional features will take longer.
- **Resources:**
  - Home Page: http://www.coldboxframework.com/
  - Sites that use it
    - http://www.jpl.nasa.gov/index.cfm
    - http://www.esri.com/
    - http://www.ortussolutions.com/
    - http://docs.transfer-orm.com/
  - Discussion Group: http://groups.google.com/group/coldbox
  - Paid support http://www.coldboxframework.com/index.cfm/support/paid

- **Option 2 - Model-Glue (MG):**
  - **Summary**
    - Popular implicit invocation MVC framework, large use of XML for configuration.
    - Requires ColdSpring
    - Easy integration with Transfer & Reactor ORMs.
  - **Risks/Limitations:**
    - No additional risks (other than above) with this Framework vs. others.
  - **Pros:**
    - Built in code generation via scaffolding (useful for getting started or prototypes).
    - Large active user community.
    - Commercial training available.
    - Implicit invocation allowing multiple events handlers to fire.
  - **Cons:**
    - Weak documentation. Developer must rely on blogs and mail lists.
    - XML (for configuration) can be very verbose.
  - **Cost & Effort:**
    - There are no hard costs.
    - Estimated Development training time for basic knowledge at 24 hours.
    - Mastery of the Framework would take longer.
  - **Resources:**
    - Home Page: http://www.model-glue.com/
    - Sites that use it: http://www.model-glue.com/sites.cfm
    - Discussion Group: http://groups.google.com/group/model-glue

- **Option 3 - Mach II:**
  - **Summary**
    - The oldest and most mature ColdFusion MVC.
    - Light weight, and uses implicit invocation.
  - **Risks/Limitations:**
    - There has been some recent activity, however this Framework didn't isn't as actively developed as MG and CB.
  - **Mitigating Risks:**
    - Same as the others.
  - **Pros:**
    - Well tested and stable.
    - Large community
  - **Cons:**
    - Weak and out of date documentation, samples are based on older versions.
    - Not as many extras.
    - Core development at a slower pace than the others.
  - **Cost & Effort:**
    - There are no hard costs.
    - Estimated Development training time for basic knowledge at 32 hours.
    - Mastery of the Framework would take longer.
  - **Resources:**
    - Home Page: http://www.mach-ii.com/
    - Sites that use it :http://www.mach-ii.info/index.cfm?event=resources
    - Discussion Group: http://groups.google.com/group/mach-ii-for-coldfusior

- **Option 4 - FuseBox:**
  - **Summary**
    - The oldest of all ColdFusion Frameworks, not explicitly MVC.
    - * NOTE: This evaluation did not heavily evaluate FuseBox.
  - **Risks/Limitations:**
    - No risks other than using third party software**.**
  - **Pros:**
    - Very well tested and stable.
    - The Largest community.
  - **Cons:**
    - Difficult to configure (relative to others).
    - Allows for very procedural programming. Good choice for hybrid environments.
  - **Cost & Effort:**
    - There are no hard costs.
    - Estimated Development training time for basic knowledge at 24 hours.
    - Mastery of the Framework would take longer.
  - **Resources:**
    - Home Page: http://www.fusebox.org
    - Site that use it: http://www.fusebox.org/go/fusebox-community/community-resources/web-sites
    - Discussion Group: http://www.fusebox.org/go/fusebox-community

Options Summary:

- **Score:** 0 – 10. 10 = Better.
- **Items not evaluated against each other:**
  - Relative Performance: All of these Frameworks have proven track records with supporting extremely large applications with high traffic volumes and concurrent connections. However the relative performance from one Framework vs. another was not part of this evaluation – the requirement was that there needed to be evidence that any particular Framework could support our needs.

Option	Learning Ease	Price	Docs	Features	Extensibility	Activity	Support
ModelGlue	5	$0	3	6	7	7	Yes
Coldbox	8	$0	9	9	10	10	Yes
Mach II	5	$0	2	0	4	3	Yes
FuseBox	6	$0	3	3	3	5	Yes