

FRAMEWORKS

2007 CONFERENCE



Anatomy of a Blog: Mach-II in Action

Matt Woodward
Principal IT Specialist, U.S. Senate

About Me

- Principal IT Specialist, Office of the Sergeant at Arms, United States Senate
- ColdFusion developer since 1996
- Also develop in Java, Flex, C#
- Release Coordinator and contributing developer for Mach-II
- Co-host of the ColdFusion Weekly Podcast (<http://www.coldfusionweekly.com>)

Agenda

- Examine the architecture of a “medium-large” Mach-II application
 - Layered approach to architecture
 - Importance of the service layer in creating a framework-agnostic application
 - Scalable approach—this architecture could be used on much larger applications
- Discuss approaches to handling things such as ...
 - Security
 - i18n
 - Application flow

Prerequisites

- Knowledge of basic OO principles
 - I'm happy to answer questions, but due to time constraints I can't do too much "OO 101" discussion
- Some knowledge of Mach-II or another event-driven, MVC framework
- Some knowledge of ColdSpring, or at least knowledge of the problem ColdSpring solves

The Problem

- Numerous sample applications for Mach-II and other frameworks exist
- These are great for basic stuff, but there haven't been too many full-scale applications available for study
 - LOTS of development done in Mach-II, but not a lot of source code can be shared
- MachBlog was built in part to bridge the gap between basic sample applications and more real-world, full-scale apps

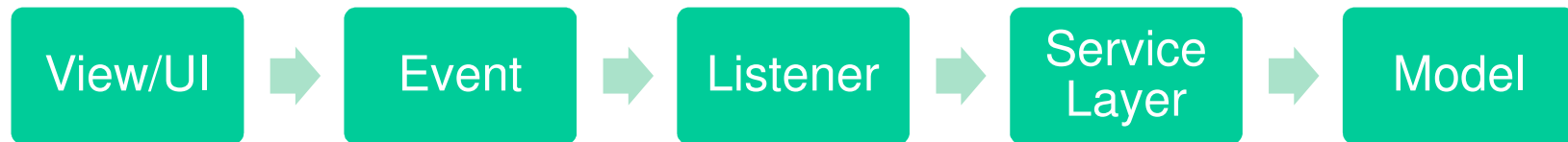
MachBlog Features

- Full-featured blog application
 - Still in a 1.0 release, so we have a laundry list of features to add, but it's pretty full featured
- Built with Mach-II using what the authors feel is best practices
 - Also uses ColdSpring to handle dependency injection
- Supports SQL Server 2000/2005, MySQL 4/5, and PostgreSQL 7.3.15 or higher
 - Relatively easy to add support for additional databases
 - We have no plans to support Access, but someone could easily add it
- Easily skinnable
 - Looking at ways to make this even easier in future releases
- Fully i18n ready
 - Ships with English, very easy to add new languages
- Runs on CF 7.0.2 (Standard and Enterprise) and BlueDragon 7 Server/JX/J2EE (*not* .NET) on Linux, CentOS, Windows, and OS X
 - This wasn't as easy as maybe it sounds, particularly with the i18n bits

MachBlog Architecture

- Like most Mach-II applications, MachBlog is
 - Event-driven
 - MVC
 - Built using OO principles
- Emphasis on service layer
 - Makes model totally reusable and unaware of Mach-II
 - Strict (well, relatively strict) dividing line between Mach-II and the model at the Mach-II listeners

High-Level Architecture Layers



Service Layer

- a.k.a “Service Layer Strategy”
- Can be thought of as an API to the model
 - Remember, the model should be completely unaware of any particular framework
- Extremely good way to encapsulate all the business logic of an application into an easy-to-use (and reuse!) package
 - If you remove Mach-II from the equation, all your business logic is intact and accessible from another framework or front-end

Service Layer/Listener Interaction

- Listeners should be as dumb as possible
 - An ideal listener method simply returns the result of a call to a service object
- Service objects are completely unaware of Mach-II
 - Don't pass the Mach-II event object to a service method!
 - Makes for more work, but makes the business logic completely reusable in other contexts

Layered Architecture

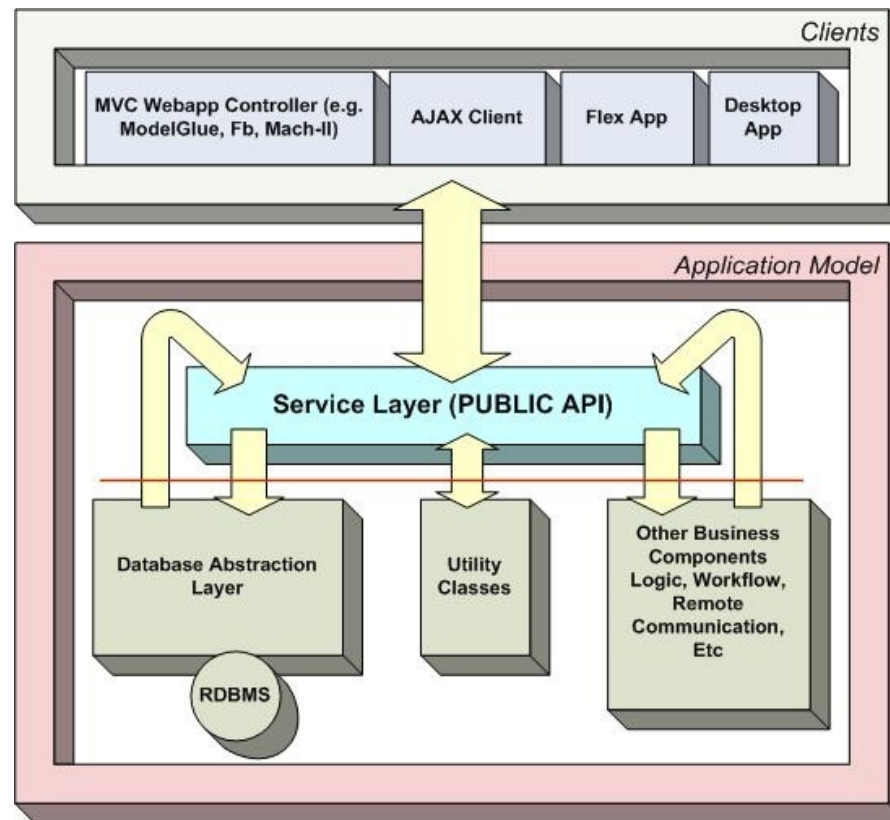


Diagram by Dave Ross

Other Architectural Interests

- Factory pattern used for data objects
 - Application asks factory for DAOs and gateways, and factory returns the object appropriate to the database being used
- Some use of facades
 - Saved my bacon when I moved where the resource bundle (used for i18n) was stored

Code Organization

- In order to avoid conflicts and forcing people to create mappings, we decided to use a package notation starting with “machblog” (e.g. machblog.org.machblog.)
- URL then becomes <http://server/machblog>
 - Can use mappings to support different URLs, e.g. <http://server/blog> or just http://server

Directory Structure

- ▼ machblog [https://svn1.cvsdude.com/mpwoodward/machblog]
 - ▶ config
 - ▶ database
 - ▶ docs
 - ▶ FCKeditor
 - ▶ i18n
 - ▶ images
 - ▶ js
 - ▼ org
 - ▶ hastings
 - ▼ machblog
 - ▼ beans
 - Aggregator.cfc 939 8/23/06 9:09 PM mpwoodward
 - Audit.cfc 939 8/23/06 9:09 PM mpwoodward
 - Category.cfc 561 5/16/06 12:36 PM mpwoodward
 - Comment.cfc 888 7/8/06 9:30 AM mpwoodward
 - Datasource.cfc 171 1/30/06 9:35 AM mpwoodward
 - Email.cfc 791 6/13/06 4:34 PM mpwoodward
 - Entry.cfc 939 8/23/06 9:09 PM mpwoodward
 - NotificationConfig.cfc 794 6/13/06 5:21 PM mpwoodward
 - Settings.cfc 939 8/23/06 9:09 PM mpwoodward
 - Subscriber.cfc 1010 12/20/06 10:58 PM mpwoodward
 - Trackback.cfc 484 5/14/06 12:31 PM mpwoodward
 - User.cfc 939 8/23/06 9:09 PM mpwoodward
 - ▶ dataobjects
 - ▶ facades
 - ▶ factories
 - ▶ filters
 - ▶ listeners
 - ▶ plugins
 - ▶ services
 - ▶ utils
 - ▶ skins
 - ▶ uploads
 - ▶ views
 - ▶ .project
 - Application.cfm 804 6/14/06 10:37 PM pfarrell
 - index.cfm 810 6/17/06 1:50 PM pfarrell
 - LICENSE 167 1/30/06 9:25 AM mpwoodward
 - NOTICE 167 1/30/06 9:25 AM mpwoodward

Directory Structure

- Main object directory package organized by type of object
- Another completely valid way to organize things is by “noun”
 - E.g. “user” directory would contain User.cfc, UserService.cfc, UserDAO.cfc, UserGateway.cfc
 - I tend to prefer this organization now

Directory Structure

- machblog.org.hastings contains the i18n CFCs
 - Paul Hastings has been indispensable to the success of the application's i18n functionality
- Views directory contains all the views, but these just do an include of the appropriate view for each skin
 - Adding a new skin is as simple as creating it, then dropping it under the "skins" directory

Database Objects

- Collection of base database objects
- Each database-specific object extends the base object
 - This keeps the type attribute correct regardless of the database being used
 - Yes, could use “any,” but what can I say—I’m a bit anal about typing
- Each database type has its own complete set of objects
 - Some, OK much, of the code is redundant, but this structure gives total freedom and independence for each database platform

Internationalization (i18n)

- Literally none of the text in MachBlog is hard-coded in English
- Adding support for another language is as simple as adding a new properties file for the language and changing a setting in the Mach-II config file
- Leverages the underlying Java i18n functionality
 - Made for some compatibility issues between operating systems and CFML engines, but hey, if this stuff was easy what fun would it be?
 - Means MachBlog will NOT run on BlueDragon .NET
- Again, a MILLION thanks to Paul Hastings for all his help

MachBlog Initialization

- When MachBlog first loads ...
 - Mach-II loads up and starts instantiating listeners, filters, and plugins
 - ColdSpring handles dependency injection—Mach-II plugin fires up ColdSpring
 - Even on smallish applications, ColdSpring makes life MUCH simpler—USE IT!
 - Singletons are instantiated with (listeners, filters, plugins, others)
 - Simple and complex properties are loaded into the property manager
 - Simple: name/value pairs declared in XML
 - Complex: CFCs like the ResourceBundleService
 - Database object factory instantiates all the appropriate database objects
 - Default event is announced

An Event in Detail: Posting an Entry

- Most of the events in MachBlog follow this same basic pattern
- Involves ...
 - Announcing an event
 - Security filter
 - Notifying a listener
 - Handling event beans and event data
 - Service layer
 - Database layer
 - Concept of an “exit event”

Event Flow Through Layered Architecture

UI/View Announces Event



Security filter applied



Event bean created from form data



subscriberListener: getSubscribers()



entryListener: processBlogEntryForm()

Event Flow Through Layered Architecture (continued)

entryListener: processBlogEntryForm()

entryService: createEntry(entry)

entryDAO: create(entry)

announceEvent, either showBlogEntryForm_error
or showBlogEntryAdmin

Posting an Entry: The Code

- Let's look at the code ...

Lots of Objects?

- Particularly if you're new to OO, this may seem like a LOT of objects and a LOT of "this calling that"
- In one sense this is true, but the advantage of this is a huge amount of flexibility

Cohesion and Coupling

- Cohesion: The degree to which an object does only one thing
 - Do one thing and do it well
 - Tight cohesion: good
- Coupling: The degree to which objects rely on one another
 - Particularly knowledge of the inner workings of one another
 - Loose coupling: good

Advantages of Layered Architecture

- Low impact maintenance
 - Can radically change one piece of the system and the rest of the application won't know or care
- Service layer creates public API to the model
 - This is where most of your work is done—best to make it as reusable and flexible as possible

What Do YOU Want to Discuss?

- Purposely left some time in here to look at specific aspects of MachBlog that we didn't cover
- i18n?
- Other architectural aspects?
- “Why didn't you do it this way?”
- General questions?

Future Features

- Mobile blogging and ability to submit from other applications
 - XML-RPC
- Possibly a Flex-based administrator
- Better integration with things like del.icio.us, flickr, digg, etc.
- Perhaps a photo gallery/photo blog capabilities
- Way to easily add content/functionality pods
- Tagging, other Web 2.0 buzzwords ☺
- AJAXifying portions of MachBlog where it makes sense
- Rework to take advantage of new features in Mach-II 1.5
- Even easier skinning—encapsulate view logic in custom tags
- Interested? GET INVOLVED!
<http://www.machblog.org>

Questions?

- THANKS!
- Matt Woodward
mpwoodward@gmail.com
<http://www.mattwoodward.com/blog>
<http://www.machblog.org>
<http://www.coldfusionweekly.com>