

Architecting Applications for Multiple UI Technologies

Matt Woodward

4th-6th June 2008
Edinburgh, Scotland

SCOTCH ON THE ROCKS

About Me

- Principal IT Specialist, Office of the Sergeant At Arms, United States Senate
- CFML developer since 1997
- On-again, off-again Flex and Java developer
- Mach-II Release Coordinator
- Member, Open BlueDragon Steering Committee

What We'll Cover

- Architecting applications to support multiple user interface (UI) technologies
- Layered approach to application architecture
- Separation of concerns in application layers
- Cohesion and coupling in objects
- The importance of the service layer

What We Won't Cover

- The basics of OO development
 - I'm happy to answer specific questions as we go, but I won't be covering any OO-101 materials

Agenda

- Overview of layered architecture in CFML applications
- Discussion of domain modeling and the importance of the business logic layer (model)
- Importance of detaching your model from any particular UI technology or framework
- Examine a very simple contact manager

Layered Application Architecture

- “There’s pasta in my software!”



4th-6th June 2008
Edinburgh, Scotland

SCOTCH ON THE ROCKS

Spaghetti

- No matter how you try to dress it up, it's just plain nasty



4th-6th June 2008
Edinburgh, Scotland

SCOTCH ON THE ROCKS

Spaghetti With Meatballs

- Better, but still a little scary



Spaghetti:
Now with CFCs!

Lasagne

- Layered, refined, the ultimate goal



4th-6th June 2008
Edinburgh, Scotland

SCOTCH ON THE ROCKS

The Layers

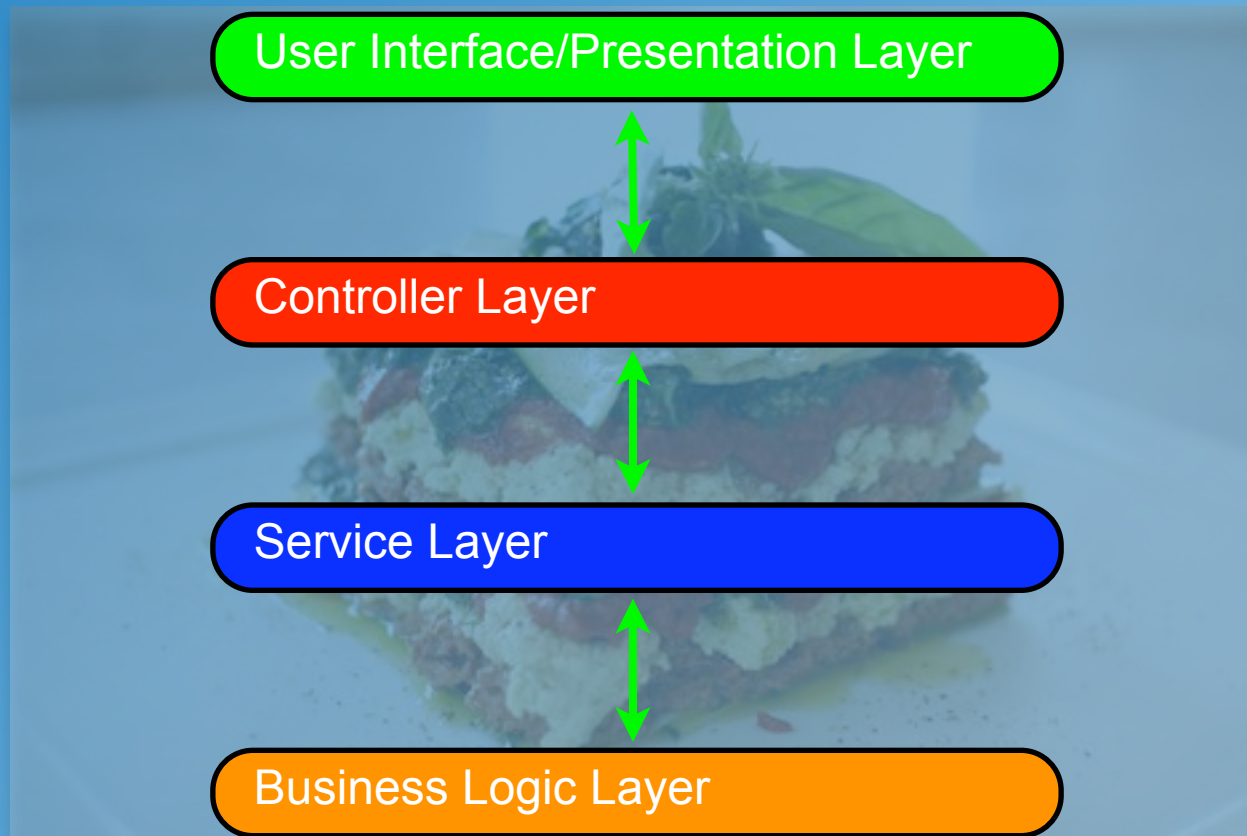
(mmm ... lasagne ...)

V

C

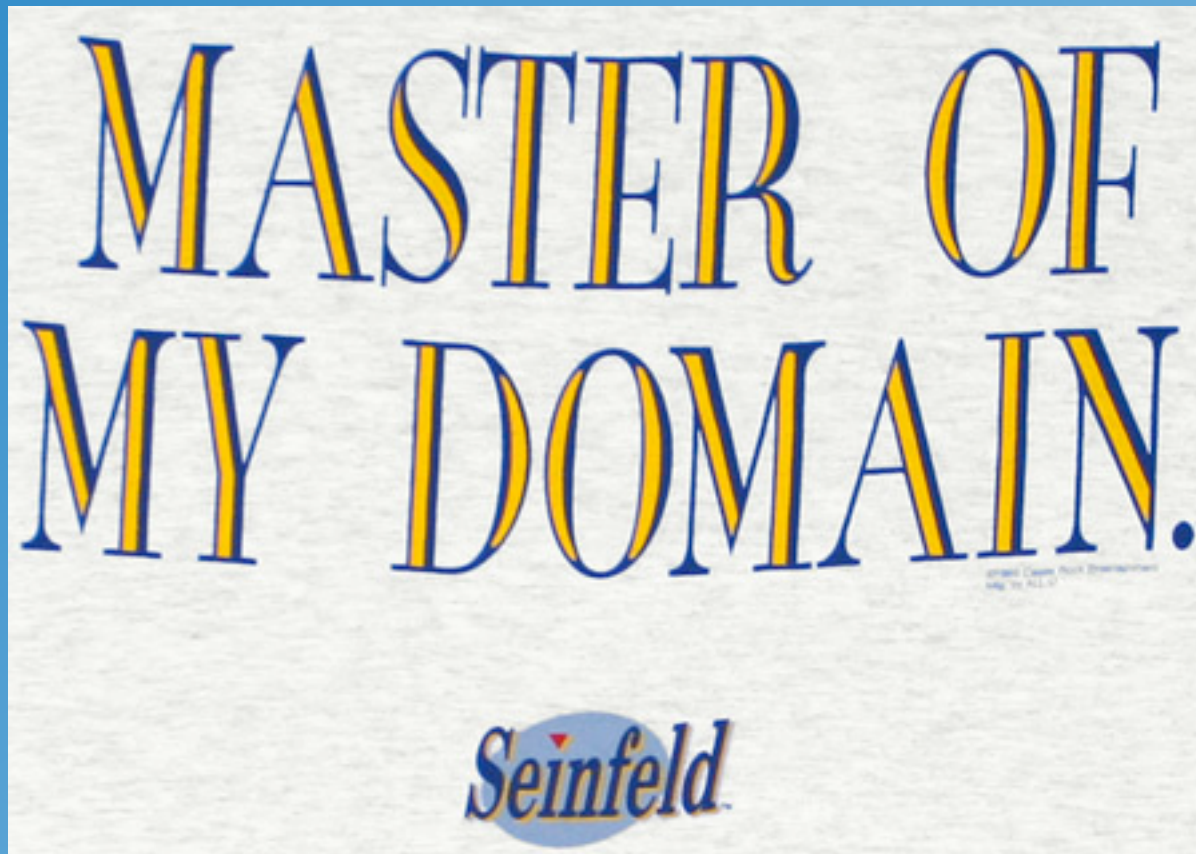
M

M



Mastering Your Domain

- No, that's not what I meant ...



4th-6th June 2008
Edinburgh, Scotland

SCOTCH ON THE ROCKS

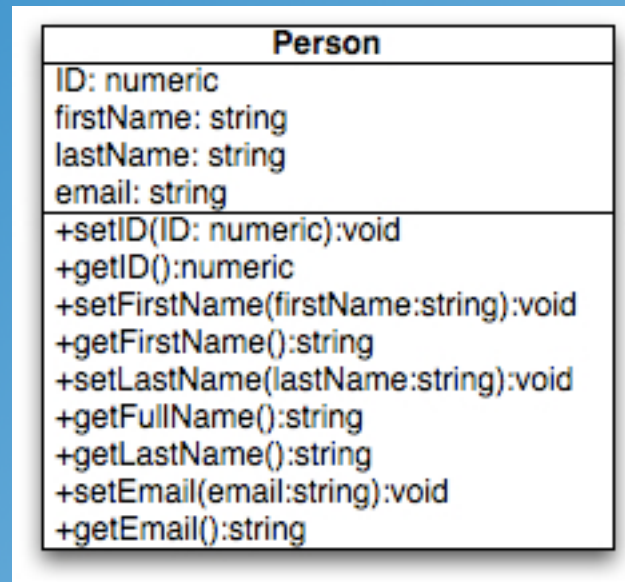
The Domain Model

- Wikipedia: “... conceptual model of a system which describes the various entities involved in that system and their relationships”
- Objects, attributes, and methods
 - objects = nouns
 - methods = verbs
- Spend a LOT of time here
 - think before you code!

Sample Domain Model

- Contact management system
- Think about what the objects (nouns) and methods (verbs) are
- Think about how objects relate to one another
- Use UML to think things through
 - easy to change diagrams, harder to change code

Person.cfc



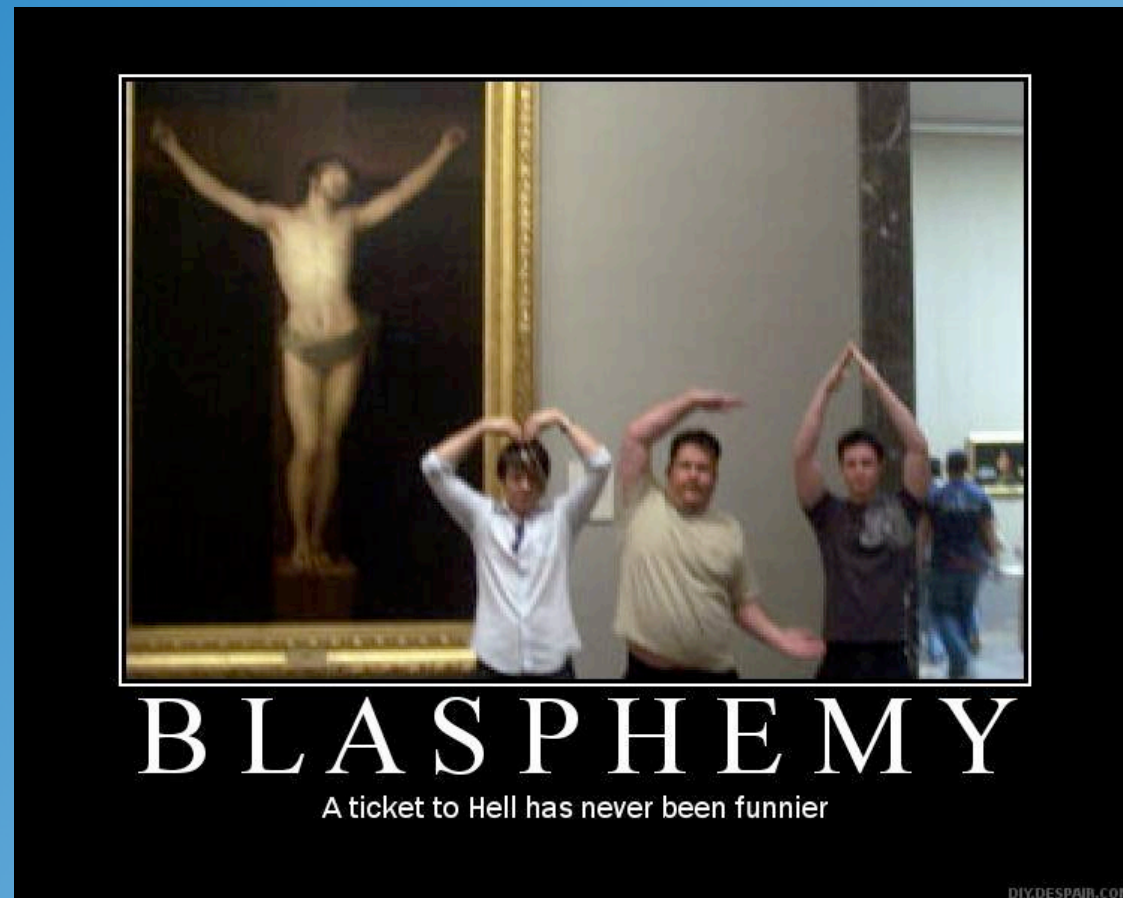
Where's the Database?



4th-6th June 2008
Edinburgh, Scotland

SCOTCH ON THE ROCKS

The Database Isn't Important



4th-6th June 2008
Edinburgh, Scotland

SCOTCH ON THE ROCKS

OK, The Database Is Important

- It's just not as important as a lot of people think it is
- The database is where you permanently store your data
- The database does NOT drive application design
- On the application side, think in objects
- Once the domain model is done, worry about the database design

Cohesion and Coupling

- VERY. IMPORTANT. CONCEPTS.
- Cohesion: the degree to which something does one thing and does it well
 - tight cohesion is the goal
- Coupling: the degree to which objects rely upon one another
 - loose coupling is the goal
- Strive for tight/high cohesion and loose coupling!

Peeling Back the Layers



4th-6th June 2008
Edinburgh, Scotland

SCOTCH ON THE ROCKS

Business Logic Layer

- Also known as the model
- The most important part of your application
 - this is where you spend most of your development time
- Separate and protect it from the outside world
- Make your model as reusable as possible

Business Logic Layer

(cont.)

- The model shouldn't know or care who's talking to it (security concerns aside)
- Your model should not expose any details about HOW it does what it does
- Your model should only expose the bare minimum to the outside world
 - Only how to talk to it

Business Objects

- Tight cohesion and loose coupling mean you will likely have ...
 - beans (the “nouns” in your system)
 - data access objects (DAOs/Gateways)
 - other business objects, UDFs, etc.
- Make sure each object does one thing and does it well
- Make sure each object doesn't have to know much about any other object

Quick Look at Business Objects

- Code samples of beans and data access objects



The Service Layer

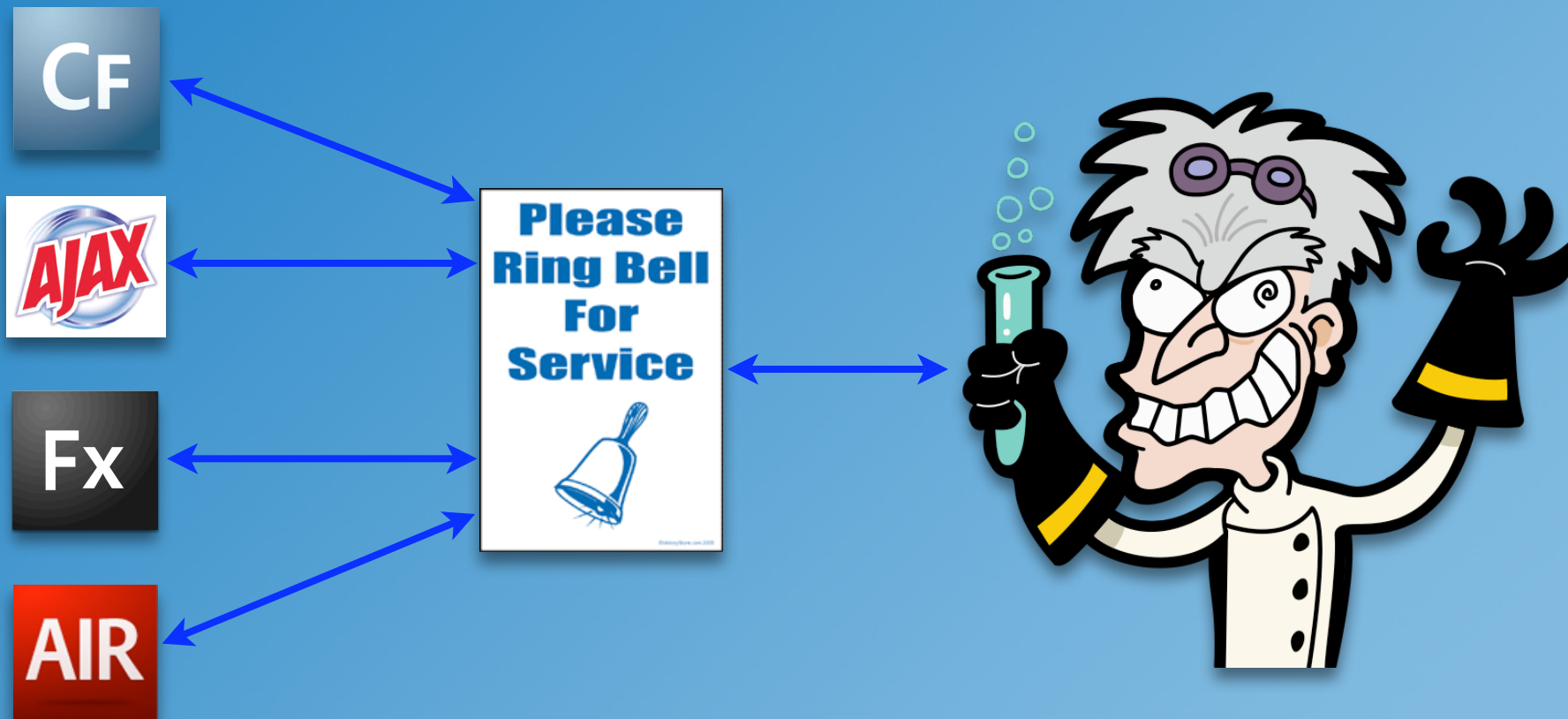
- Your business logic's receptionist



Why Have a Service Layer?

- No one should have a direct line of communication to your model
 - remember loose coupling
- The service layer is absolutely critical in creating a reusable business logic layer
- Think of the service layer as a public API to your business logic

Service Layer in Action



4th-6th June 2008
Edinburgh, Scotland

SCOTCH ON THE ROCKS

The Controller Layer

- We won't spend much time on this, but realize there is one more layer involved
- Controller layer can be ...
 - roll your own
 - framework like Mach-II
 - Flex is more or less its own controller, but there are an increasing number of frameworks for Flex

The User Interface Layer

- Using a layered architecture lets you pick your UI poison
- All UI layers talk through their controllers to the service layer
- In the case of remote calls (i.e. Flex and AJAX) it makes sense to have a remote proxy on top of the service layer

Remote Facade/Proxy

- Used to expose methods from the service layer as remote calls
- Can better handle the specific needs of asynchronous UI technologies
- Keeps your service layer nice and clean
 - service layer still doesn't know or care who's talking to it

Putting It All Together

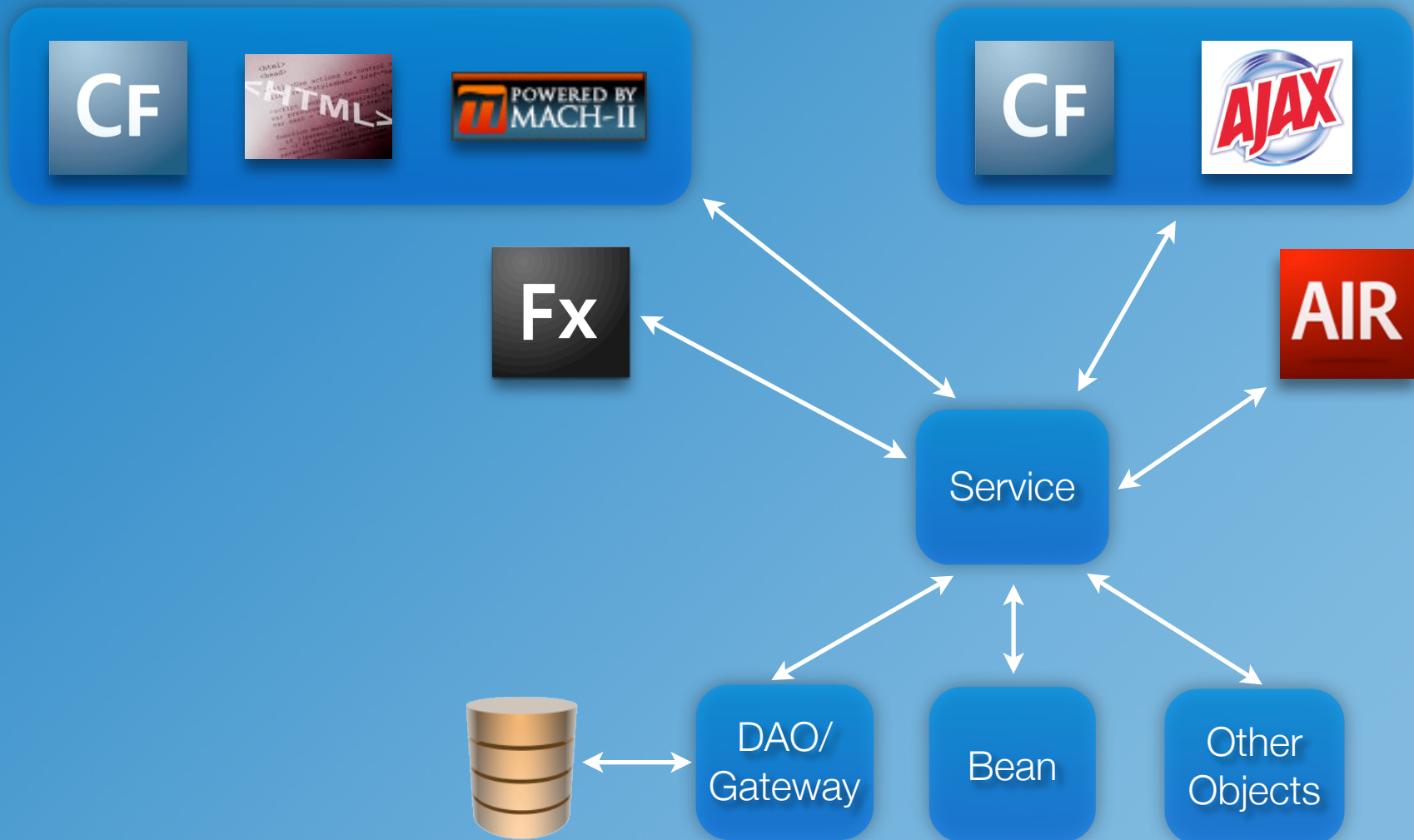
- How the magic happens



4th-6th June 2008
Edinburgh, Scotland

SCOTCH ON THE ROCKS

Poetry in Motion



4th-6th June 2008
Edinburgh, Scotland

SCOTCH ON THE ROCKS

Contact Manager Application

- Demo and code
- HTML, AJAX, Flex, and AIR all talking to the same business logic WITH NO REWORK!
- Example of remote proxy used for Flex, AJAX, and AIR

Recap

- With a little planning and some discipline, architecting for multiple UI technologies is pretty simple
- Makes for good architecture regardless of UI choice
- Keeps your business logic future proof
- Remember:
 - spend lots of time on your domain model
 - keep separation of concerns in mind
 - tight cohesion, loose coupling

Questions?

- Thanks!
- Matt Woodward
matt@mattwoodward.com
<http://www.mattwoodward.com/blog>