

# FRAMEWORKS

## 2007 CONFERENCE



### Advanced Fusebox 5: Extending the language of Fusebox

Sean A Corfield  
Team Lead, Hosted Services  
Adobe Systems, Inc.

February 1st- 2nd 2007

[www.frameworksconference.com](http://www.frameworksconference.com)

## What is this about?

- XML is the lingua franca of modern frameworks but for each framework it is normally a fixed language
- Fusebox provides a standard way to extend the XML language, to allow a more expressive way to solve problems

## Who am I?

- Lead Developer, Fusebox 5.0 & 5.1
- Fuseboxer since late 2002
  - Grudgingly at first... 😊
- ColdFusion developer since late 2001
  - Grudgingly at first... 😊
- Software Architect at Adobe /  
Macromedia since mid-2000

# Hands up!

- Who uses Fusebox?
- Who uses Fusebox 5?
- Who uses Fusebox 4?
- Who uses Fusebox 3?
- Who uses Mach II or Model-Glue?

# Agenda

- Very quick, very high level overview of Fusebox structure and how it works
- Code generation – how & why?
- Some examples – code & demo
- Wrap up

# Agenda

- Very quick, very high level overview of Fusebox structure and how it works
- Code generation – how & why?
- Some examples – code & demo
- Wrap up

# Fusebox Grammar 101

- fusebox.xml declares circuits
- circuit.xml declares fuseactions
- fuseactions contain sequences of verbs
  - <do>
  - <if>
  - <include>
  - ...
- verbs are compiled to CFML

## Other Frameworks

- The XML control file(s) are converted to a data structure that is inspected at runtime to determine behavior of the application
- Fusebox 4 and later converts the XML to CFML code and simply executes it

## Fusebox Grammar 102

- The built-in verbs are the “Fusebox lexicon” (in fusebox5/verbs/)
- Each verb (a .cfm file) is executed at “compile-time” and generates CFML
- `<set name=“foo” value=“42”/>` will cause the set.cfm verb to generate this CFML:
- `<cfset foo = “42” />`

# Anatomy of a verb (I)

- A verb is like a custom tag:
  - Executes twice:
    - executionMode = “start” / “end”
  - Validates its attributes
  - Can have nested child tags (and can communicate with them in limited ways)
- Generally written as `<cfscript>`
- Generates code with `fb_appendLine()`

## Anatomy of a verb (II)

```
if (fb_.verbInfo.executionMode is "start") {  
    // validate the attributes  
    // perform code generation for start tag  
    fb_appendLine("...some CFML...");  
} else {  
    // perform code generation for end tag  
    fb_appendLine("...some CFML...");  
}
```

## Anatomy of a verb (III)

- Let's look at the built-in <set> verb

# Agenda

- Very quick, very high level overview of Fusebox structure and how it works
- **Code generation – how & why?**
- Some examples – code & demo
- Wrap up

## Code Generation – How?

- Normally you write code that just runs
- Here we write code that writes code
  - And then we run the result
- We don't generate code every time
  - Only once in Fusebox production mode

# Code Generation – Why?

- Saves time – writes code for us!
- More expressive – small amount of language can generate a lot of code
- More abstract – program with concepts instead of nuts and bolts
- Better performance – the abstraction is “compiled out” of the solution leaving only (low level) code

# Fusebox Grammar 201

- A custom lexicon is a collection of user-defined verbs (a directory containing .cfm files)
- A custom lexicon can be used in a circuit.xml file by introducing it as an XML namespace:
  - `<circuit xmlns:prefix="path/to/verbs/">`
    - ...
    - `<prefix:myverb />`

# Fusebox Grammar 202

- path/to/verbs/myverb.cfm
  - Executed twice – start / end
  - Generates code
- Verbs can nest:
  - <mg:views>
    - <mg:include template="dspStaticPage.cfm" />
  - </mg:views>

# Agenda

- Very quick, very high level overview of Fusebox structure and how it works
- Code generation – how & why?
- **Some examples – code & demo**
- Wrap up

# Examples (I)

- ORM – Abstraction
  - High-level language – list, read, save
  - Hides implementation – Reactor or Transfer
  - Efficient – abstraction is compiled away

## Examples (II)

- Model-Glue – Expressiveness
  - Program with concepts – messages
  - Uses language of Model-Glue
  - Reuses controller and view code completely

## Examples (III)

- Domain Specific Language?

# Agenda

- Very quick, very high level overview of Fusebox structure and how it works
- Code generation – how & why?
- Some examples – code & demo
- Wrap up

## Summary

- Fusebox 5 allows you to extend the language the framework understands
- Fusebox 5 brings code generation to the masses – in a standardized way
- You can program using the language of your problem space – a higher level
- The compilation model eliminates the performance overhead of abstraction

# Resources

- Fusebox
  - <http://fuseboxframework.org/>
- An Architect's View
  - <http://corfield.org/>
- Application Generation
  - <http://pbell.com/>

# FRAMEWORKS

## 2007 CONFERENCE



### Q&A

Sean Corfield  
<http://corfield.org/>  
[sean@corfield.org](mailto:sean@corfield.org)

February 1st- 2nd 2007

[www.frameworksconference.com](http://www.frameworksconference.com)