Modern frameworks for robust Adobe Coldfusion development

Современные фреймворки для качественной разработки на Adobe Coldfusion

Родион Быков

16-я встреча Microsoft .NET User Group Sevastopol

О себе

- Web-разработчик (Adobe Coldfusion, Adobe Flex/AIR, jQuery, PHP/MySQL)
- 2002г. — Алвион, Севастополь
- 2005г. — Фрилансер

О чем вообще речь ?

- Фреймворк — набор повторно используемого кода, шаблон проекта
- Фреймворк ограничивает фантазию разработчика, но предлагает хорошие решения
- Adobe Coldfusion – сервер приложений, созданный по спецификации J2EE
- Интерпретирует код на языке CFML

- Основное использование — создание динамических web-страниц
- Прямые аналоги — ASP, JSP, PHP, другие скриптовые языки
- Также используется для создания и манипуляции документами в других форматах (PDF, XLS)
- Работа с messaging gateway, flash remoting
- Легкость - достоинство и недостаток

История вопроса

- Coldfusion – 16 лет в эфире
- Статичные сайты
- CGI/Perl
- Нет легких средств разработки
- ASP, PHP – 1998, JSP – 1999

Почти HTML !

```
<CFQUERY NAME=”userdata”
  DATASOURCE=”datasource”>
  SELECT firstname, lastname, phone
  FROM users
  WHERE lastname LIKE '#search#'
</CFQUERY>
<HTML>
  <BODY>
    <CFOUTPUT QUERY=”userdata”>
    <P>#lastname#,#firstname#: #phone#</P>
    </CFOUTPUT>
  </BODY>
</HTML>
```

Однако...

- Неструктурированный (“spaghetti”) код
- Отсутствие архитектуры проекта

Как следствие

- Тяжело чинить ошибки (“вечные” баги)
- Тяжело расширять функциональность
- Тяжело передать проект другой команде

Общая проблема

- Отсутствие общей структуры (архитектуры кода)
- Низкая связность (low cohesion – бизнес-логика “размазана” по модулям случайным образом)
- Высокая связанность (tight coupling — модули кода могут выполнять действия только в связке с другими модулями или зависят от внешних данных)

Решение

- **Архитектура** — паттерн MVC (Model-View-Controller)
- **Повышение связности** — организация кода в модули, реализующие одну, хорошо определенную бизнес-функцию
- **Понижение связанности** — делать модули как можно менее зависимыми друг от друга, при этом связь осуществляется передачей параметров или посылкой сообщений

Хороший фреймворк

- Написан умными людьми
- Хорошо документирован
- Развитие кода активным сообществом
- Не требует изменений
- Подходит любому проекту
- Оптимизирован
- Легко изучить

Что дает использование фреймворка

- Является стандартом, разнопланновые проекты будут одинаково устроены внутри
- Заставляет следовать хорошим принципам организации кода
- Улучшает взаимодействие в команде (код более понятен, можно совместно редактировать код проекта, не мешая друг другу)
- Новый разработчик легко подхватит работу

Популярные фреймворки

- Структурные: Fusebox, Mach-II, Model-Glue, FW/1
- Логические: Coldspring, LightWire
- ORM: TransferORM, Reactor, CF9 ORM
- Комплексные: ColdBox, cfWheels
- index.cfm?action=module.procedure
- index.cfm?event=module.procedure

Fusebox

- Суперстар, не поддерживается сообществом
- Прекрасно реализует MVC-паттерн
- Код разбивается на circuits и fuseactions
- В версии 4 использует XML для описания действий
- Процедурный подход
- Поддерживает плагины и лексиконы

Fusebox No-XML

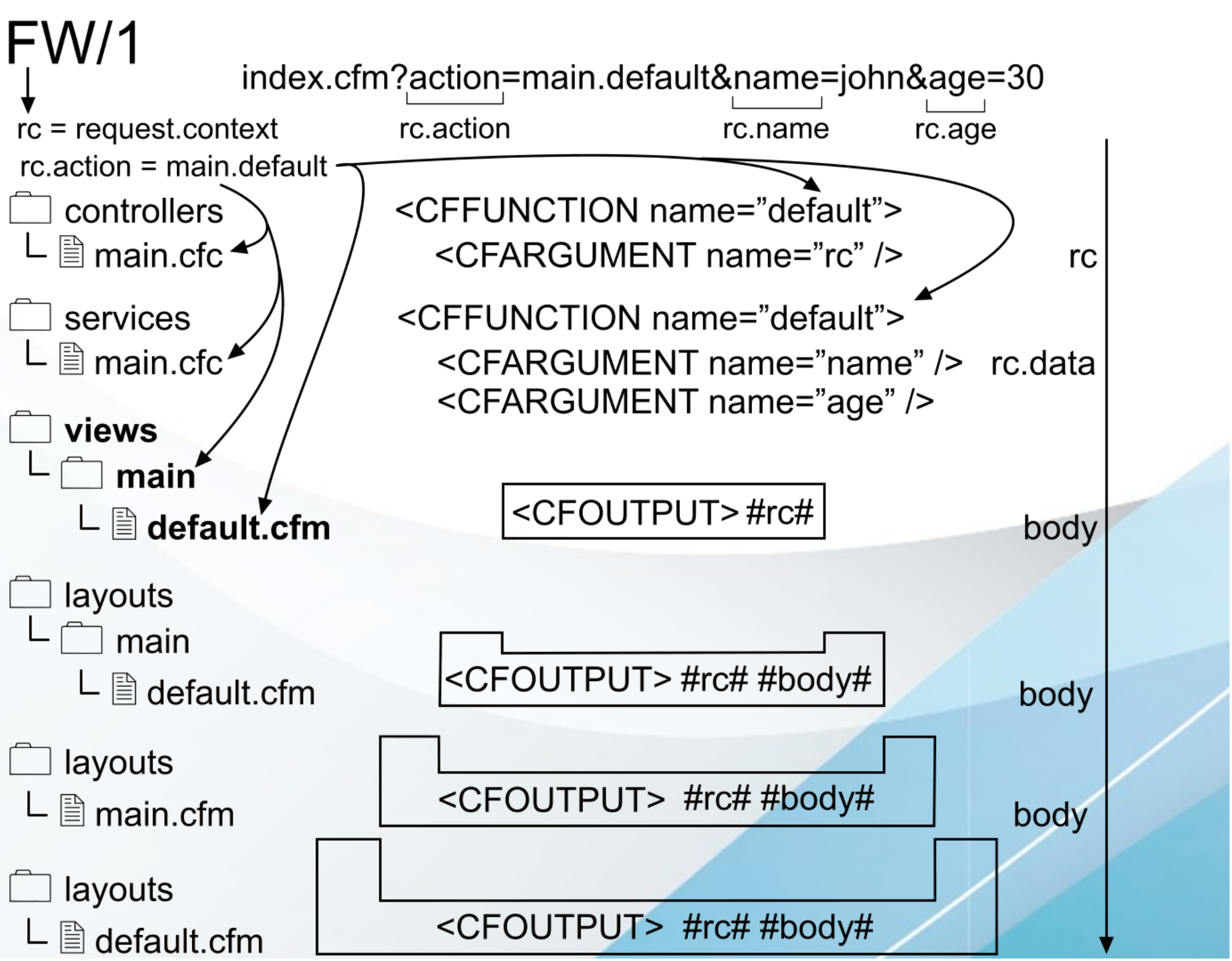
- Convention over configuration
- Использует CFC для описания circuits, методы становятся описанием fuseaction-а.

Mach-II

- Появилась в 2003г., с появлением CFC в Coldfusion MX
- Первый OO-фреймворк
- Использует XML для описания
- Implicit invocation – events, listeners, filters
- Event – приходит с URL или генерируется
- CFC-listener выполняет действия

FW/1 (Framework One)

- Фреймворк помещается в один CFC файл
- Convention over configuration
- CFC-контроллеры: “traffic cop”, бизнес-логика, валидация, управление
- CFM - views, layouts – 3 уровня вложенности
- Приложение разбивается на sections и items
- Реализованы подсистемы


FW/1
index.cfm?action=main.default&name=john&age=30
rc = request.context
rc.action
rc.name
rc.age
rc.action = main.default
controllers
main.cfc
services
main.cfc
views
main
default.cfm
layouts
main
default.cfm
layouts
main.cfm
layouts
default.cfm
<CFFUNCTION name="default">
<CFARGUMENT name="rc" />
<CFFUNCTION name="default">
<CFARGUMENT name="name" />
<CFARGUMENT name="age" />
rc
rc.data
<CFOUTPUT> #rc#
body
<CFOUTPUT> #rc# #body#
body
<CFOUTPUT> #rc# #body#
body
<CFOUTPUT> #rc# #body#

Coldspring

- Вдохновлена Java Spring
- Одна фабрика для производства объектов (в том числе синглтонов)
- Автоматическое разрешение зависимостей между объектами (dependency injection)
- Использует XML для описания объектов и зависимостей

TransferORM

- Использует XML для описания сущностей
- Использует существующие таблицы БД, на которые делается mapping
- Умеет работать с зависимостями (one-to-many, many-to-one, many-to-many)
- Поддерживает запросы (TQL – Transfer Query Language)
- Работает с CF8 и Railo

CF9 ORM

- Базируется на Hibernate (скорость работы и надежность)
- Описывает сущности в существующих таблицах БД
- Для описания использует CFML или CFScript
- Для работы не нужно знания Hibernate

Мне надо все это знать ?

- Нет
- Применять осмысленно
- Имеет смысл для команд, новых проектов или новых частей проектов (субъективно)
- Многие проекты и программисты прекрасно обходятся без использования фреймворков
- Есть недостатки (ORM – повод для дискуссий)

Спасибо

- Coldfusion User Group Ukraine
  http://cfug.org.ua
- @rodionbykov