Enterprise Class MVC with ColdFusion & Java

Andrew Powell

Sr. Consultant

Universal Mind

UNIVERSAL MIND

t umAndy

SA3A

Goals

- Leverage Java & Common Java Frameworks to Provide A Modular Business Model.

- Use ColdFusion 8 & Mach-II to Put a Face On a Spring/ Hibernate Powered Java Applicaiton

Introductions

- Sr. Consultant, Universal Mind

- UM is a 2007 Inc. 500 Honoree

- 10+ Years Experience In Software Development

- ASP --> Java --> ColdFusion --> Flex

- Multiple Publications on Flex & ColdFusion

- http://www.infoaccelerator.net

Topics

- Introduction To Enterprise Software and MVC Concept

- Building The Java Model

  - Implementing Spring

  - Implementing Hibernate

- Building The Controller and View

  - Implementing Spring with Mach-II

  - Implementing The Controller & View

Enterprise Class MVC

* **Enterprise Software**

  * Software which provides business logic support functionality for an organization, typically in commercial organizations, which aims to improve the organization's productivity and efficiency.

* **MVC**

  * Decouples data access and business logic from data presentation and user interaction by introducing an intermediate component: the controller.

Why Java?

* Web-Enable Existing Legacy Java Applications

    * Add a Web 2.0 Interface

* Portable Applications - Disparate UI Implementations

    * External: Web - ColdFusion / AJAX

    * Internal: Swing

* Right Hammer For The Job

* It's Faster!

    * Going "native" to the Java level
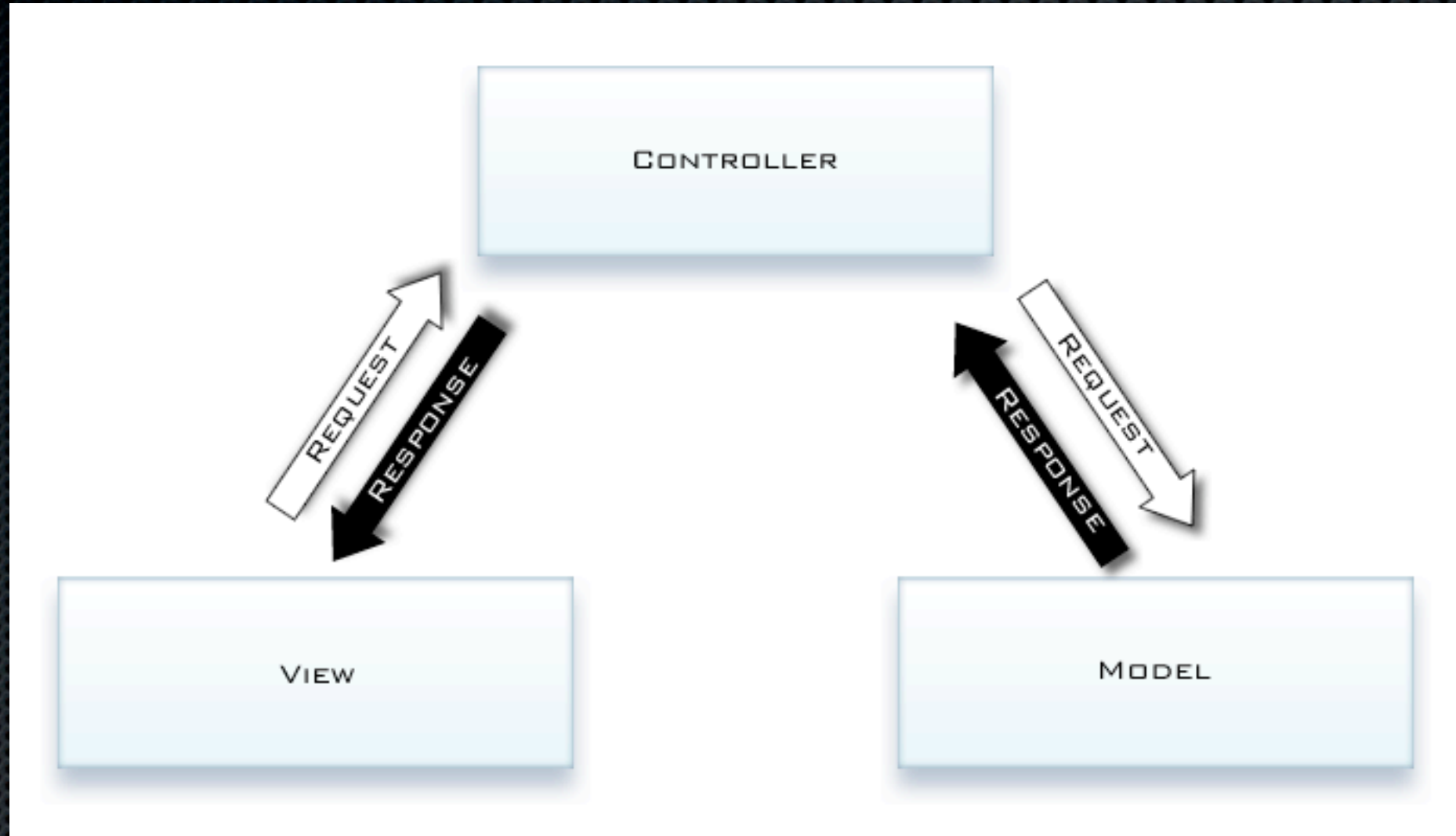
The Concept
Model-View-Controller

The Players

The Environment

- JBoss 4.0.5 GA - Application Server

  - ColdFusion 8

    - Just Another J2EE App

- Java 1.6.0 - SoyLatte (OpenJDK)

  - Hibernate 3.2 - ORM Framework

  - Spring 2.5.0 - IOC/AOP Framework

- Mach-II 1.5

Building The Java Model

Introducing Spring

* What Is Spring?

* Spring Provides:

  * Inversion of Control

  * Aspect Orient Programming Model

* Many Sub-Projects:

  * Spring Security (Acegi Security)

  * Spring Web Services

* Basis of ColdSpring Framework

Spring Basics

* Configured Via XML

  * Can Use Multiple XML Files

* Application Context Loaded At App Startup

  * Config Defined in web.xml

* Does Not Provide Data Access, Logging, Transactions, etc.

  * Provides Interfaces to Existing Libraries

  * JMS, ORMs, etc.

Show Me The Code

Introducing Hibernate

* What Is Hibernate?

* Hibernate Provides:

  * Relational Persistence

  * Automatic Object and Query Caching

* Sub Projects:

  * Shards - Horizontal Data Partitioning

  * Search - Integration with Lucene

  * Validator - Data Integrity Validation API

* Shares A Philosophy with Transfer & Reactor

Hibernate Basics

- Configured Via XML or Properties File

  - hibernate.cfg.xml - config file

  - myPOJO.hbm.xml - mapping file

    - Can Also Use Java Annotations

- Inline, Modular Caching Configuration

- Inline, Modular Transaction Management

- Can Also Be Configured in Spring Configuration

Want To See Some More Code?

Building
The Controller And View

What's ColdFusion's Role?

* Marshaling Data In / Out of Model

  * Validating Data

  * Formatting Data

* Rendering Views

Problems To Solve

* How Do We Get Access To The Spring Application Context?

    * Application Context == Bean Factory

* How Do We Keep The Session Open Through The View's Completion?

Where Oh Where Is My ApplicationContext?

* Loaded at Startup by Application Server

* Accessible Via ColdFusion

  * Manage With Mach-II Plugin?

  * Load Into Session Scope?

  * Load Into ApplicationScope?

Solving One Problem With ColdFusion

* How Do We Get Access To The Spring Application Context?

  * Via Java:

    * WebApplicationContext context = WebApplicationContextUtils.getWebApplicationContext(ServletContext context);

  * Via ColdFusion:

    * getPageContext().getServletContext()

Solving One Problem With Spring

- How Do We Keep The Session Open Through The View, If Needed?

    - Open Session In View Filter (OSIV) in <u>web.xml</u>

Problems Solved, What Next?

* Time to look at some ColdFusion code...

* The model is defined already, what's left?

  * MVC Config - mach-ii.xml

  * Controller

    * Listeners

    * Plugins

    * Filters

  * Views

Thank You Sir, May I Have Another Demo?

Drawbacks To This Approach

- 2 Codebases To Manage

- Need Java Talent On Team

- Difficult To Debug

Advantages To This Approach

* VERY Portable Application

  * Easily Port To Flex/AIR via BlazeDS / LCDS

  * Easily Port To JSP via SpringMVC

* Guaranteed Optimization of Persistence Calls (SQL)

  * Can Tune Fetch Methods

* Easily Scales and Clusters

Want To Learn More?

* http://www.hibernate.org

* http://www.springframework.org

* http://www.mach-ii.com

* http://www.infoaccelerator.net

Q & A