
ColdBox

# ColdFusion Powered

## Mobile Applications

ColdBox

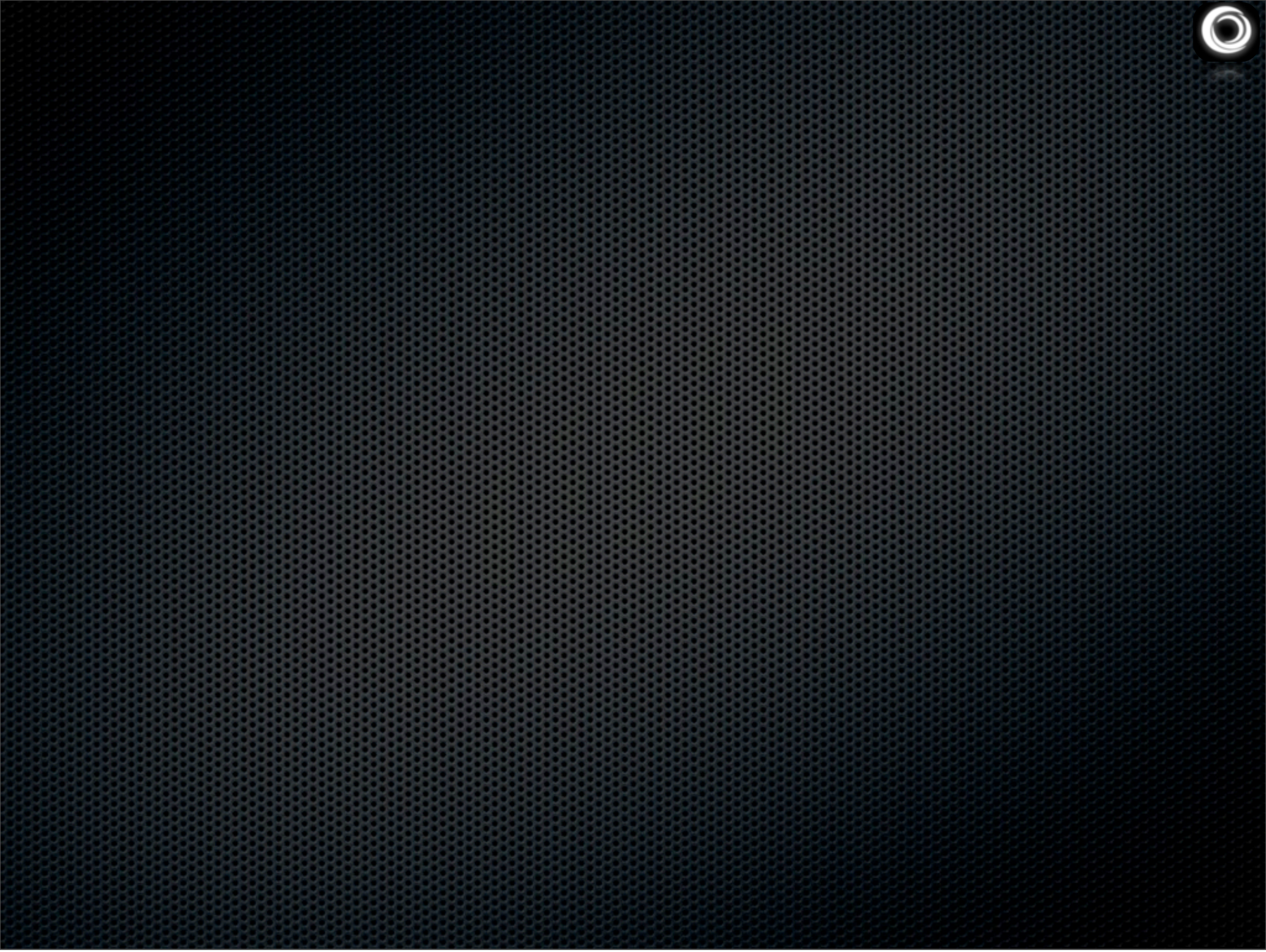
# Who am I?

# Who am I?

- Luis Majano - Computer Engineer
- Born in San Salvador, El Salvador -->
- President of Ortus Solutions
- Manager of the IECFUG (www.iecfug.com)

- Creator of ColdBox, MockBox, LogBox, CacheBox, WireBox, BlogBox, or anything Box!
- Documentation Weirdo!

Ortus Solutions
A Software Revolution



# What we will cover?

- Communication protocols & data formats
- Why power a mobile app via ColdFusion->ColdBox
- ColdBox RESTful web services
- Resources identification & domain analysis
- RESTful Modeling via Relax
- Application development basics
- Documentation, yes Documentation!
- Cool Demo

# Getting Started

# Getting Started

# Getting Started


# Getting Started

# Getting Started

# Getting Started

# Getting Started

# Communication Protocols

# Communication Protocols

- HTTP/S is our main protocol of communication
  - **RESTful web services or**
  - SOAP
- Data Formats
  - XML
  - **JSON**

# Our Recommendations

# Our Recommendations

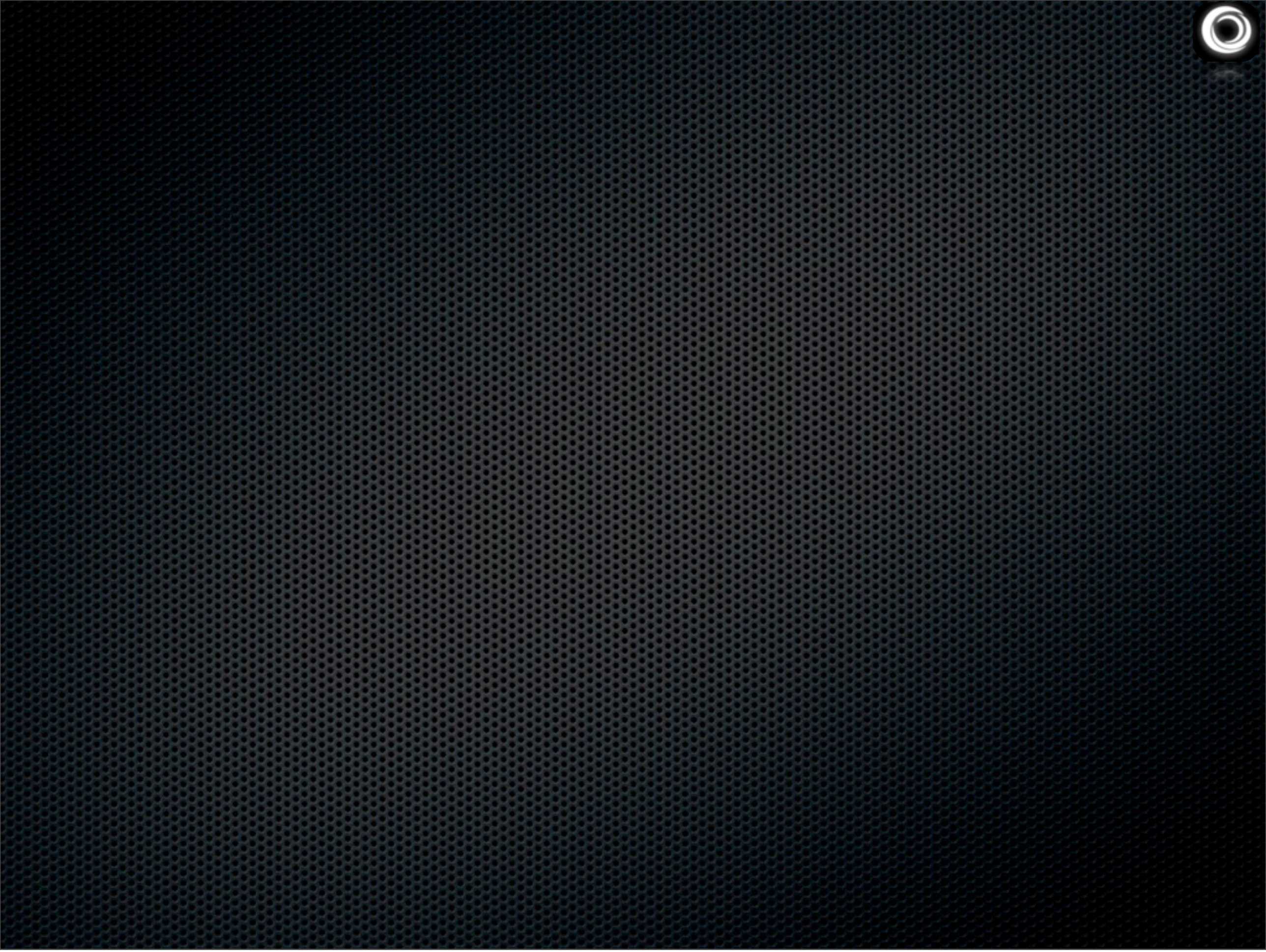
- **RESTful** web services over SOAP
  - low ceremony services
  - resource oriented and not RPC method oriented
  - readability and good abstraction layer
  - better bandwidth
  - stateless
- **JSON** over XML
  - less verbose
  - lightweight communication
  - better iPhone/Android support

# RESTful Architecture

# RESTful Architecture

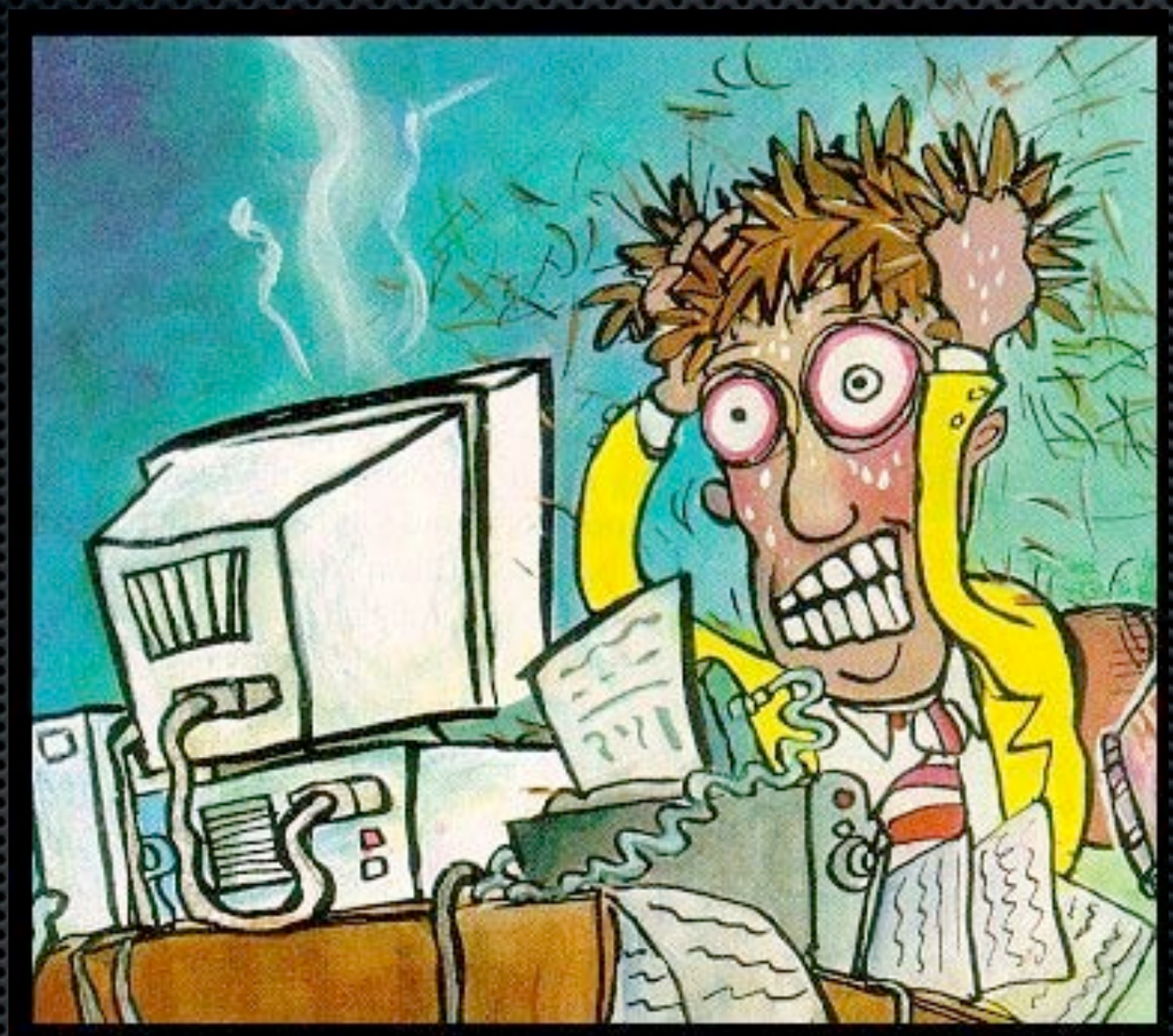
- Representational State Transfer (REST)
  - Identify Resources to an URI
  - The resource responds to different HTTP verbs
  - The deeper you go in the resource -> More detail
  - GET - Represent resource(s)
  - PUT - Update
  - POST - Save
  - DELETE - Delete

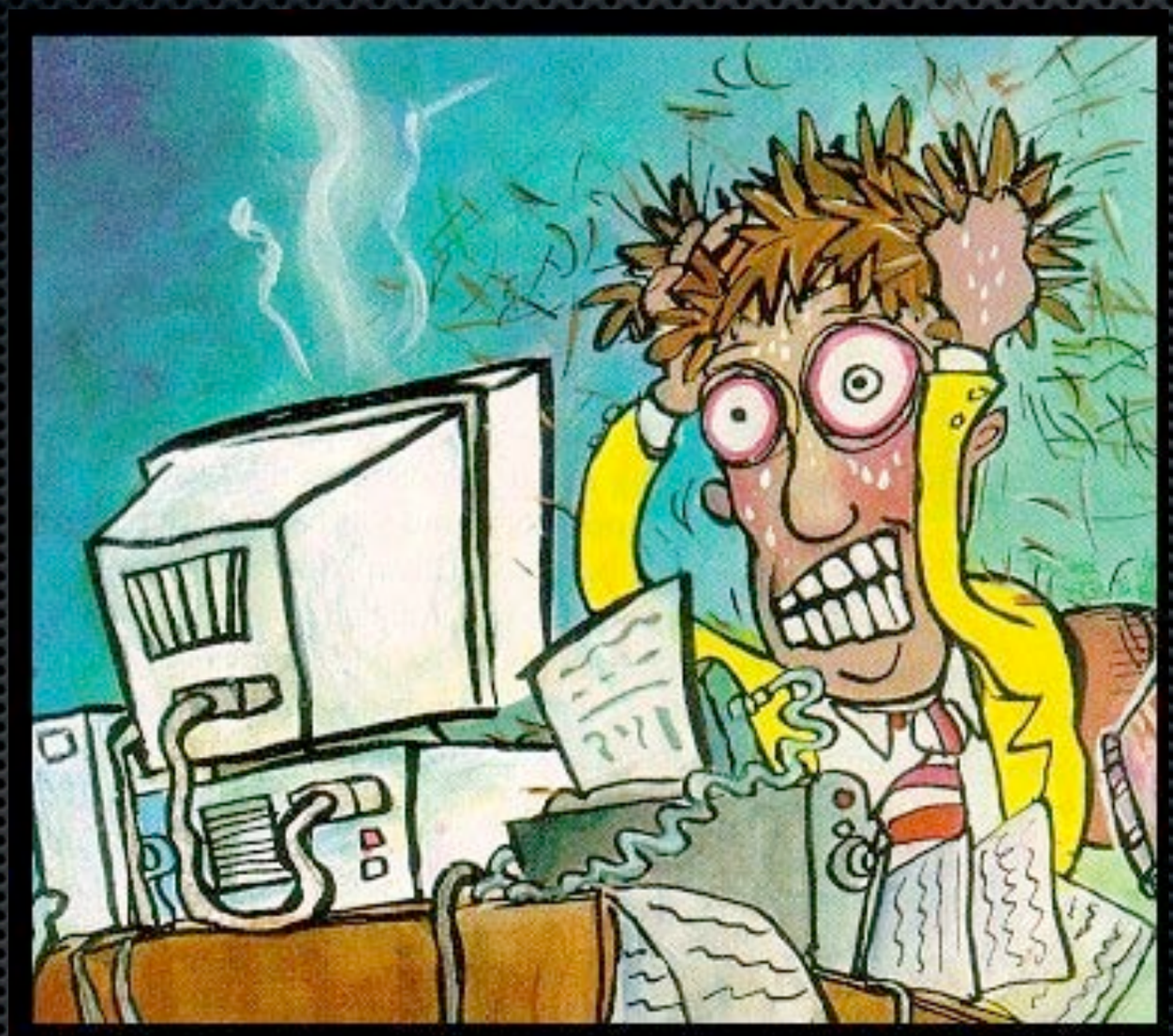
```
/resource
/resource/detail
/resource/detail/moredetail?param1=value
```



# What powers our Mobile Devices

What do I do?





# ColdFusion-ColdBox To The Rescue

Testing

Modularity

Extensibility

Utilities

SES URLs


Logging

Scalability

Caching

Debuggers

Data Formats

HTTP Security

HTTP Verb Recognition

# Why ColdBox?

# Why ColdBox?

- Built in SES and RESTFul resource management

# Why ColdBox?

- Built in SES and RESTFul resource management
- HTTP verb recognitions & Routing (POST,GET,PUT,DELETE,HEAD,OPTIONS)

# Why ColdBox?

- Built in SES and RESTFul resource management
- HTTP verb recognitions & Routing (POST,GET,PUT,DELETE,HEAD,OPTIONS)
- Data/ORM transformations to XML, JSON, WDDX, Custom

# Why ColdBox?

- Built in SES and RESTFul resource management
- HTTP verb recognitions & Routing (POST,GET,PUT,DELETE,HEAD,OPTIONS)
- Data/ORM transformations to XML, JSON, WDDX, Custom
- RESTful resource security and Basic Authentication

# Why ColdBox?

- Built in SES and RESTFul resource management
- HTTP verb recognitions & Routing (POST,GET,PUT,DELETE,HEAD,OPTIONS)
- Data/ORM transformations to XML, JSON, WDDX, Custom
- RESTful resource security and Basic Authentication
- Integration testing and mocking facilities

# Why ColdBox?

- Built in SES and RESTFul resource management
- HTTP verb recognitions & Routing (POST,GET,PUT,DELETE,HEAD,OPTIONS)
- Data/ORM transformations to XML, JSON, WDDX, Custom
- RESTful resource security and Basic Authentication
- Integration testing and mocking facilities
- Logging, profiling and tuning

# Why ColdBox?

- Built in SES and RESTFul resource management
- HTTP verb recognitions & Routing (POST,GET,PUT,DELETE,HEAD,OPTIONS)
- Data/ORM transformations to XML, JSON, WDDX, Custom
- RESTful resource security and Basic Authentication
- Integration testing and mocking facilities
- Logging, profiling and tuning
- Resource caching and scalability

# Why ColdBox?

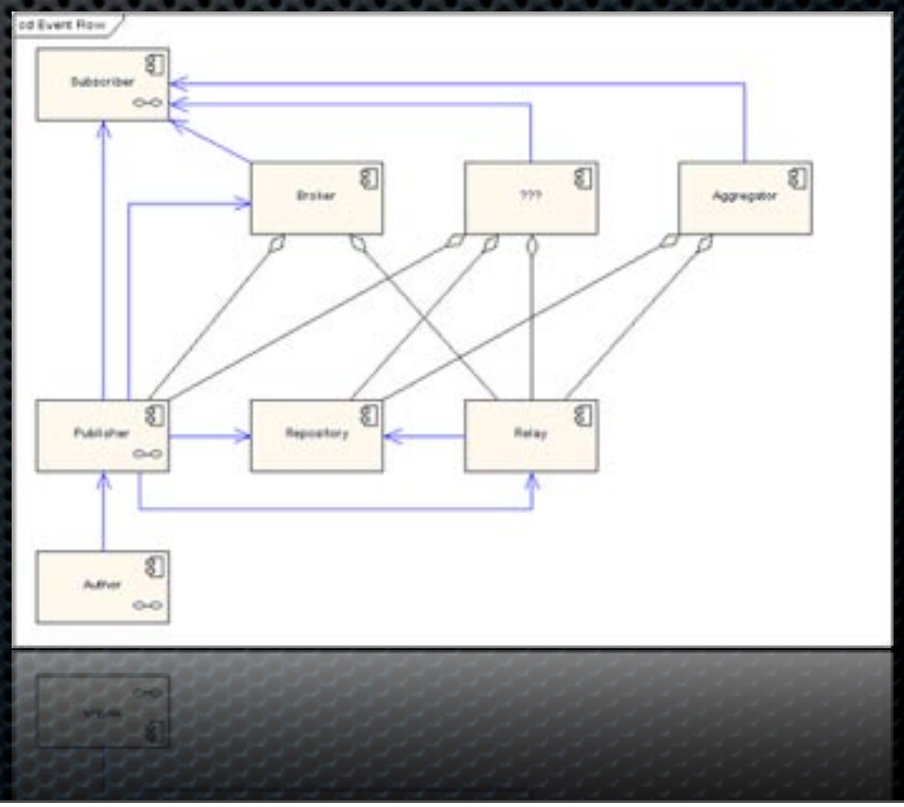
- Built in SES and RESTFul resource management
- HTTP verb recognitions & Routing (POST,GET,PUT,DELETE,HEAD,OPTIONS)
- Data/ORM transformations to XML, JSON, WDDX, Custom
- RESTful resource security and Basic Authentication
- Integration testing and mocking facilities
- Logging, profiling and tuning
- Resource caching and scalability

**“Everything you need to power a mobile app!”**

# Planning our Service

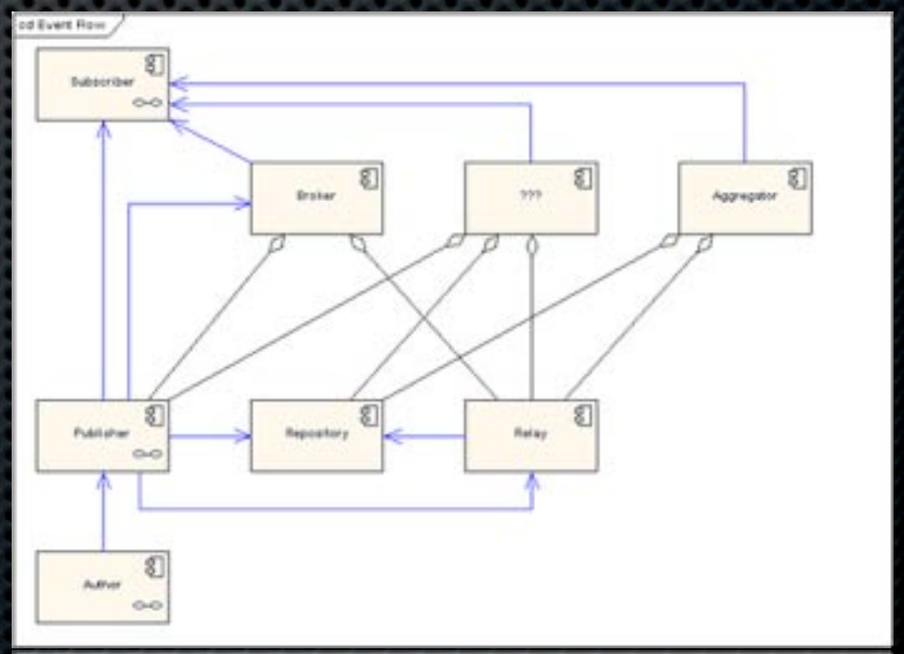
# Planning our Service

- Domain Analysis & Design (UML)



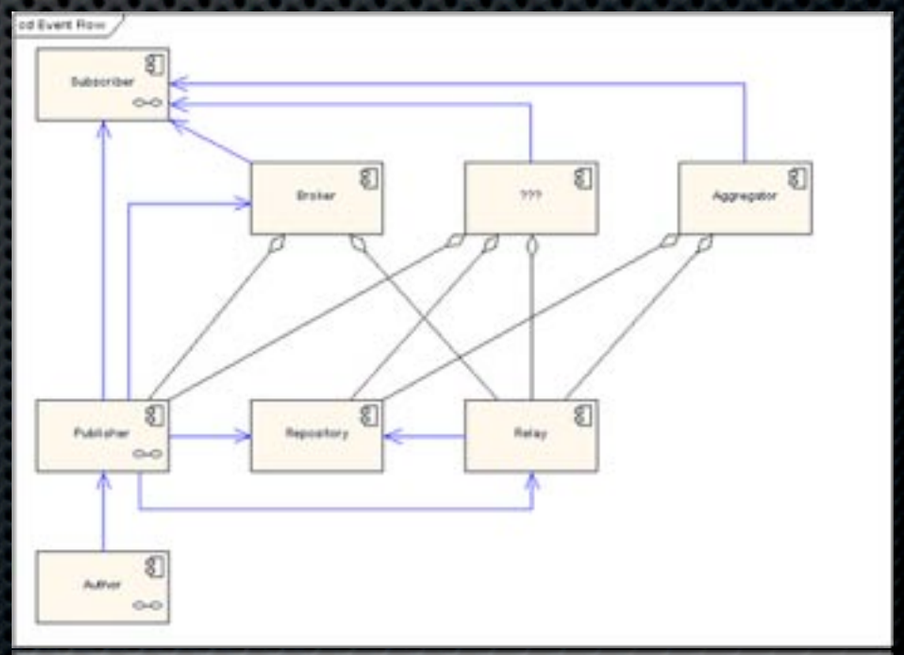
# Planning our Service

- Domain Analysis & Design (UML)
- Plan our model layer, build it and test it, yes TEST IT!



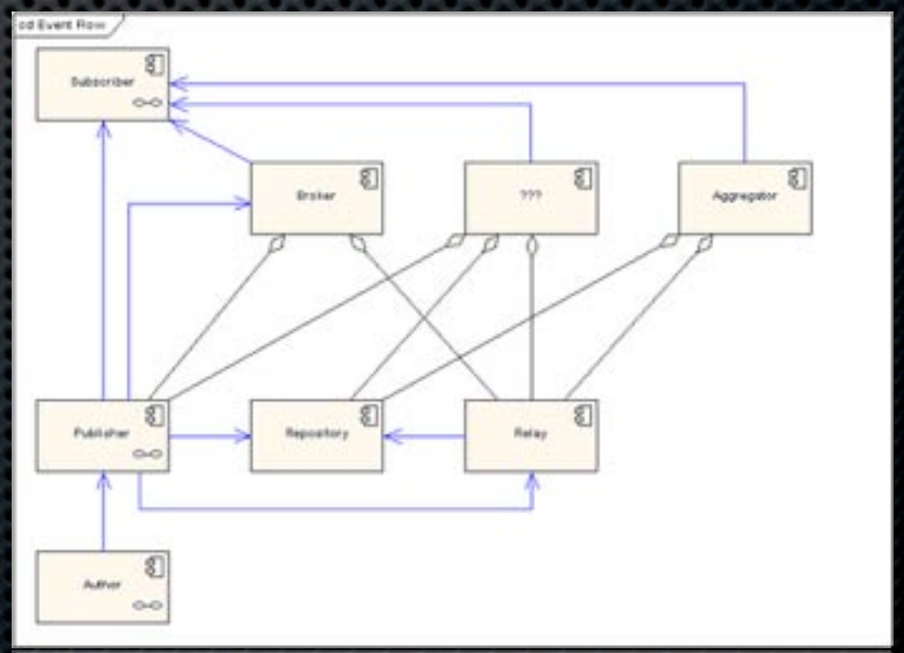
# Planning our Service

- Domain Analysis & Design (UML)
- Plan our model layer, build it and test it, yes TEST IT!
- Think about API Security (If Any)



# Planning our Service

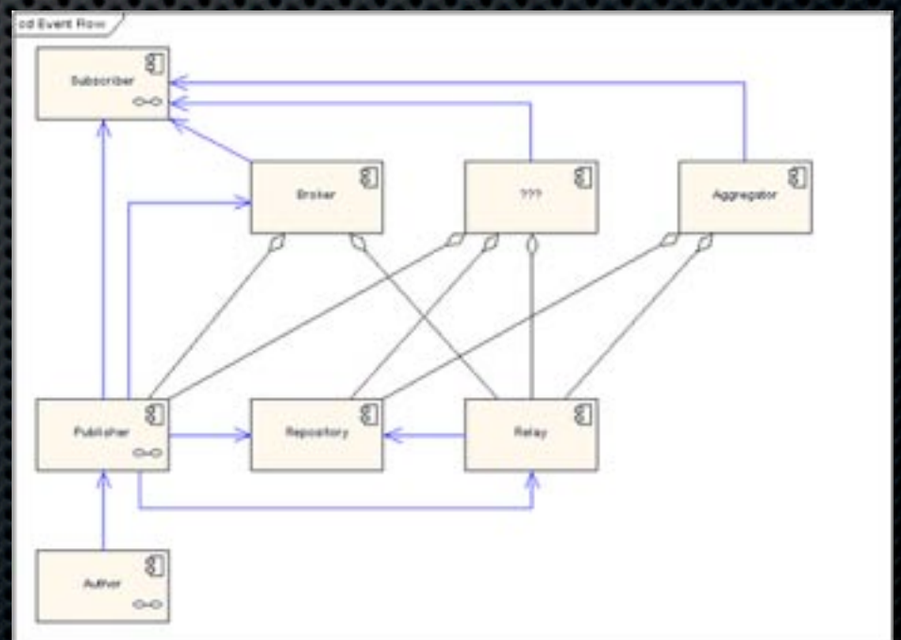
- Domain Analysis & Design (UML)
- Plan our model layer, build it and test it, yes TEST IT!
- Think about API Security (If Any)
- UI Mockups (Balsamiq, Paper, HTML, if any)



# Planning our Service

- Domain Analysis & Design (UML)
- Plan our model layer, build it and test it, yes TEST IT!
- Think about API Security (If Any)
- UI Mockups (Balsamiq, Paper, HTML, if any)
- **Resource Identification & Planning**

```
/api/user/:username?
 - GET -> index
 - POST -> create
 - PUT -> update
 - DELETE -> remove
```



# API Security

# API Security

- Approaches:
  - Registered Client API Keys or Tokens
  - API Key + Secret Encryption Keys (Like Amazon)
  - Basic Authentication (At least its something!)
  - IP Based Filtering/Tagging (Programmatic/Firewall/Etc)
  - Headers
- HTTPS is a must!
- HTTP Method Security
- Request Throttling
- How secure do you need to be?

# Let's build our service!

now?

# Right Tools

# Right Tools

- CFBuilder + ColdBox Platform Utilities
  - Code Generation
  - Templating
  - Syntax Highlighting
  - Code Introspection
  - API Quick Docs
  - Much more..

# Right Tools

# Right Tools

- HTTP Monitors/Proxies
  - Charles, FireBug, HTTPFox
- RESTFul Consoles
  - Relax
  - Eclipse
  - REST Client for FireFox

# Right Tools

# Right Tools


- RESTFul Tools For Lazy Experts
- ColdBox Module
- Resource, Headers, Parameter and Results Modeling
- Documentation Generation
  - CodexWiki, Mediawiki, Trac, PDF, HTML
- Service Logging & Monitoring
- RelaxURL Console (Pronounced “Relaxer”)
  - Test your resources
  - Save your requests for later testing

# Service Setup

# Service Setup


- Generate application
- Configure Logging (LogBox)
  - DB Appender 4 Relax Monitoring
- Configure 4 Full URL Rewrites
  - mod_rewrite, ISAPI, etc
- Install Relax Module

# Relax Configuration

```
// Relax Configuration Settings
settings = {
    // Relax Version: DO NOT ALTER
    version = this.version,

    // Relax DSL component that has the resource definitions, this is an instanatiation path
    configCFC = "resources.myapi.Relax",
    // History stack size, the number of history items to keep on requests
    maxHistory = 10,
    // logbox integration information needed for log viewer to work
    // this means that it can read tables that are written using the logbox's DB Appender.
    relaxLogs = {
        // THE CF DATASOURCE NAME
        datasource = "relax",
        // THE DB TO USE FOR LOGS, AVAILABLE ADAPTERS ARE: mysql, mssql, postgres, oracle
        adapter = "mysql",
        // THE TABLE WHERE THE LOGS ARE
        table   = "api_logs",
        // PAGING MAX ROWS
        maxRows = 50,
        // PAGING CARROUSEL BANDGAP
        bandGap = 3
    }
};
```

**relax/ModuleConfig.cfc**

# Relax Configuration

```
// Relax Configuration Settings
settings = {
    // Relax Version: DO NOT ALTER
    version = this.version,

    // Relax DSL component that has the resource definitions, this is an instanatiation path
    configCFC = "resources.myapi.Relax",
    // History stack size, the number of history items to keep on requests
    maxHistory = 10,
    // logbox integration information needed for log viewer to work
    // this means that it can read tables that are written using the logbox's DB Appender.
    relaxLogs = {
        // THE CF DATASOURCE NAME
        datasource = "relax",
        // THE DB TO USE FOR LOGS, AVAILABLE ADAPTERS ARE: mysql, mssql, postgres, oracle
        adapter = "mysql",
        // THE TABLE WHERE THE LOGS ARE
        table   = "api_logs",
        // PAGING MAX ROWS
        maxRows = 50,
        // PAGING CARROUSEL BANDGAP
        bandGap = 3
    }
};
```

**relax/ModuleConfig.cfc**

# Relax Configuration

```
// Relax Configuration Settings
settings = {
    // Relax Version: DO NOT ALTER
    version = this.version,

    // Relax DSL component that has the resource definitions, this is an instanatiation path
    configCFC = "resources.myapi.Relax",
    // History stack size, the number of history items to keep on requests
    maxHistory = 10,
    // logbox integration information needed for log viewer to work
    // this means that it can read tables that are written using the logbox's DB Appender.
    relaxLogs = {
        // THE CF DATASOURCE NAME
        datasource = "relax",
        // THE DB TO USE FOR LOGS, AVAILABLE ADAPTERS ARE: mysql, mssql, postgres, oracle
        adapter = "mysql",
        // THE TABLE WHERE THE LOGS ARE
        table   = "api_logs",
        // PAGING MAX ROWS
        maxRows = 50,
        // PAGING CARROUSEL BANDGAP
        bandGap = 3
    }
};
```

**relax/ModuleConfig.cfc**

# Relax Configuration

```
// Relax Configuration Settings
settings = {
    // Relax Version: DO NOT ALTER
    version = this.version,

    // Relax DSL component that has the resource definitions, this is an instanatiation path
    configCFC = "resources.myapi.Relax",
    // History stack size, the number of history items to keep on requests
    maxHistory = 10,
    // logbox integration information needed for log viewer to work
    // this means that it can read tables that are written using the logbox's DB Appender.
    relaxLogs = {
        // THE CF DATASOURCE NAME
        datasource = "relax",
        // THE DB TO USE FOR LOGS, AVAILABLE ADAPTERS ARE: mysql, mssql, postgres, oracle
        adapter = "mysql",
        // THE TABLE WHERE THE LOGS ARE
        table   = "api_logs",
        // PAGING MAX ROWS
        maxRows = 50,
        // PAGING CARROUSEL BANDGAP
        bandGap = 3
    }
};
```

**relax/ModuleConfig.cfc**

# Relax Configuration

```
// Relax Configuration Settings
settings = {
    // Relax Version: DO NOT ALTER
    version = this.version,

    // Relax DSL component that has the resource definitions, this is an instanatiation path
    configCFC = "resources.myapi.Relax",
    // History stack size, the number of history items to keep on requests
    maxHistory = 10,
    // logbox integration information needed for log viewer to work
    // this means that it can read tables that are written using the logbox's DB Appender.
    relaxLogs = {
        // THE CF DATASOURCE NAME
        datasource = "relax",
        // THE DB TO USE FOR LOGS, AVAILABLE ADAPTERS ARE: mysql, mssql, postgres, oracle
        adapter = "mysql",
        // THE TABLE WHERE THE LOGS ARE
        table   = "api_logs",
        // PAGING MAX ROWS
        maxRows = 50,
        // PAGING CARROUSEL BANDGAP
        bandGap = 3
    }
};
```

**relax/ModuleConfig.cfc**

# Relax Configuration

```
// Relax Configuration Settings
settings = {
    // Relax Version: DO NOT ALTER
    version = this.version,

    // Relax DSL component that has the resource definitions, this is an instanatiation path
    configCFC = "resources.myapi.Relax",
    // History stack size, the number of history items to keep on requests
    maxHistory = 10,
    // logbox integration information needed for log viewer to work
    // this means that it can read tables that are written using the logbox's DB Appender.
    relaxLogs = {
        // THE CF DATASOURCE NAME
        datasource = "relax",
        // THE DB TO USE FOR LOGS, AVAILABLE ADAPTERS ARE: mysql, mssql, postgres, oracle
        adapter = "mysql",
        // THE TABLE WHERE THE LOGS ARE
        table   = "api_logs",
        // PAGING MAX ROWS
        maxRows = 50,
        // PAGING CARROUSEL BANDGAP
        bandGap = 3
    }
};
```

**relax/ModuleConfig.cfc**

# Relax Configuration

```
// Relax Configuration Settings
settings = {
    // Relax Version: DO NOT ALTER
    version = this.version,

    // Relax DSL component that has the resource definitions, this is an instanatiation path
    configCFC = "resources.myapi.Relax",
    // History stack size, the number of history items to keep on requests
    maxHistory = 10,
    // logbox integration information needed for log viewer to work
    // this means that it can read tables that are written using the logbox's DB Appender.
    relaxLogs = {
        // THE CF DATASOURCE NAME
        datasource = "relax",
        // THE DB TO USE FOR LOGS, AVAILABLE ADAPTERS ARE: mysql, mssql, postgres, oracle
        adapter = "mysql",
        // THE TABLE WHERE THE LOGS ARE
        table   = "api_logs",
        // PAGING MAX ROWS
        maxRows = 50,
        // PAGING CARROUSEL BANDGAP
        bandGap = 3
    }
};
```

**relax/ModuleConfig.cfc**

# Relax Configuration

```
// Relax Configuration Settings
settings = {
    // Relax Version: DO NOT ALTER
    version = this.version,

    // Relax DSL component that has the resource definitions, this is an instanatiation path
    configCFC = "resources.myapi.Relax",
    // History stack size, the number of history items to keep on requests
    maxHistory = 10,
    // logbox integration information needed for log viewer to work
    // this means that it can read tables that are written using the logbox's DB Appender.
    relaxLogs = {
        // THE CF DATASOURCE NAME
        datasource = "relax",
        // THE DB TO USE FOR LOGS, AVAILABLE ADAPTERS ARE: mysql, mssql, postgres, oracle
        adapter = "mysql",
        // THE TABLE WHERE THE LOGS ARE
        table   = "api_logs",
        // PAGING MAX ROWS
        maxRows = 50,
        // PAGING CARROUSEL BANDGAP
        bandGap = 3
    }
};
```

**relax/ModuleConfig.cfc**

# Relax Configuration

```
// Relax Configuration Settings
settings = {
    // Relax Version: DO NOT ALTER
    version = this.version,

    // Relax DSL component that has the resource definitions, this is an instanatiation path
    configCFC = "resources.myapi.Relax",
    // History stack size, the number of history items to keep on requests
    maxHistory = 10,
    // logbox integration information needed for log viewer to work
    // this means that it can read tables that are written using the logbox's DB Appender.
    relaxLogs = {
        // THE CF DATASOURCE NAME
        datasource = "relax",
        // THE DB TO USE FOR LOGS, AVAILABLE ADAPTERS ARE: mysql, mssql, postgres, oracle
        adapter = "mysql",
        // THE TABLE WHERE THE LOGS ARE
        table   = "api_logs",
        // PAGING MAX ROWS
        maxRows = 50,
        // PAGING CARROUSEL BANDGAP
        bandGap = 3
    }
};
```

**relax/ModuleConfig.cfc**

# Relax Configuration

```
// Relax Configuration Settings
settings = {
    // Relax Version: DO NOT ALTER
    version = this.version,

    // Relax DSL component that has the resource definitions, this is an instanatiation path
    configCFC = "resources.myapi.Relax",
    // History stack size, the number of history items to keep on requests
    maxHistory = 10,
    // logbox integration information needed for log viewer to work
    // this means that it can read tables that are written using the logbox's DB Appender.
    relaxLogs = {
        // THE CF DATASOURCE NAME
        datasource = "relax",
        // THE DB TO USE FOR LOGS, AVAILABLE ADAPTERS ARE: mysql, mssql, postgres, oracle
        adapter = "mysql",
        // THE TABLE WHERE THE LOGS ARE
        table    = "api_logs",
        // PAGING MAX ROWS
        maxRows = 50,
        // PAGING CARROUSEL BANDGAP
        bandGap = 3
    }
};
```

**relax/ModuleConfig.cfc**

# Relax Configuration

```
// Relax Configuration Settings
settings = {
    // Relax Version: DO NOT ALTER
    version = this.version,

    // Relax DSL component that has the resource definitions, this is an instanatiation path
    configCFC = "resources.myapi.Relax",
    // History stack size, the number of history items to keep on requests
    maxHistory = 10,
    // logbox integration information needed for log viewer to work
    // this means that it can read tables that are written using the logbox's DB Appender.
    relaxLogs = {
        // THE CF DATASOURCE NAME
        datasource = "relax",
        // THE DB TO USE FOR LOGS, AVAILABLE ADAPTERS ARE: mysql, mssql, postgres, oracle
        adapter = "mysql",
        // THE TABLE WHERE THE LOGS ARE
        table    = "api_logs",
        // PAGING MAX ROWS
        maxRows = 50,
        // PAGING CARROUSEL BANDGAP
        bandGap = 3
    }
};
```

**relax/ModuleConfig.cfc**

# Relax Configuration

```
// Relax Configuration Settings
settings = {
    // Relax Version: DO NOT ALTER
    version = this.version,

    // Relax DSL component that has the resource definitions, this is an instanatiation path
    configCFC = "resources.myapi.Relax",
    // History stack size, the number of history items to keep on requests
    maxHistory = 10,
    // logbox integration information needed for log viewer to work
    // this means that it can read tables that are written using the logbox's DB Appender.
    relaxLogs = {
        // THE CF DATASOURCE NAME
        datasource = "relax",
        // THE DB TO USE FOR LOGS, AVAILABLE ADAPTERS ARE: mysql, mssql, postgres, oracle
        adapter = "mysql",
        // THE TABLE WHERE THE LOGS ARE
        table    = "api_logs",
        // PAGING MAX ROWS
        maxRows = 50,
        // PAGING CARROUSEL BANDGAP
        bandGap = 3
    }
};
```

**relax/ModuleConfig.cfc**

# Relax Configuration

```
// Relax Configuration Settings
settings = {
    // Relax Version: DO NOT ALTER
    version = this.version,

    // Relax DSL component that has the resource definitions, this is an instanatiation path
    configCFC = "resources.myapi.Relax",
    // History stack size, the number of history items to keep on requests
    maxHistory = 10,
    // logbox integration information needed for log viewer to work
    // this means that it can read tables that are written using the logbox's DB Appender.
    relaxLogs = {
        // THE CF DATASOURCE NAME
        datasource = "relax",
        // THE DB TO USE FOR LOGS, AVAILABLE ADAPTERS ARE: mysql, mssql, postgres, oracle
        adapter = "mysql",
        // THE TABLE WHERE THE LOGS ARE
        table    = "api_logs",
        // PAGING MAX ROWS
        maxRows = 50,
        // PAGING CARROUSEL BANDGAP
        bandGap = 3
    }
};
```

**relax/ModuleConfig.cfc**

# Relax Configuration

```
// Relax Configuration Settings
settings = {
    // Relax Version: DO NOT ALTER
    version = this.version,

    // Relax DSL component that has the resource definitions, this is an instanatiation path
    configCFC = "resources.myapi.Relax",
    // History stack size, the number of history items to keep on requests
    maxHistory = 10,
    // logbox integration information needed for log viewer to work
    // this means that it can read tables that are written using the logbox's DB Appender.
    relaxLogs = {
        // THE CF DATASOURCE NAME
        datasource = "relax",
        // THE DB TO USE FOR LOGS, AVAILABLE ADAPTERS ARE: mysql, mssql, postgres, oracle
        adapter = "mysql",
        // THE TABLE WHERE THE LOGS ARE
        table   = "api_logs",
        // PAGING MAX ROWS
        maxRows = 50,
        // PAGING CARROUSEL BANDGAP
        bandGap = 3
    }
};
```

**relax/ModuleConfig.cfc**

# Relax Configuration

```
// Relax Configuration Settings
settings = {
    // Relax Version: DO NOT ALTER
    version = this.version,

    // Relax DSL component that has the resource definitions, this is an instanatiation path
    configCFC = "resources.myapi.Relax",
    // History stack size, the number of history items to keep on requests
    maxHistory = 10,
    // logbox integration information needed for log viewer to work
    // this means that it can read tables that are written using the logbox's DB Appender.
    relaxLogs = {
        // THE CF DATASOURCE NAME
        datasource = "relax",
        // THE DB TO USE FOR LOGS, AVAILABLE ADAPTERS ARE: mysql, mssql, postgres, oracle
        adapter = "mysql",
        // THE TABLE WHERE THE LOGS ARE
        table   = "api_logs",
        // PAGING MAX ROWS
        maxRows = 50,
        // PAGING CARROUSEL BANDGAP
        bandGap = 3
    }
};
```

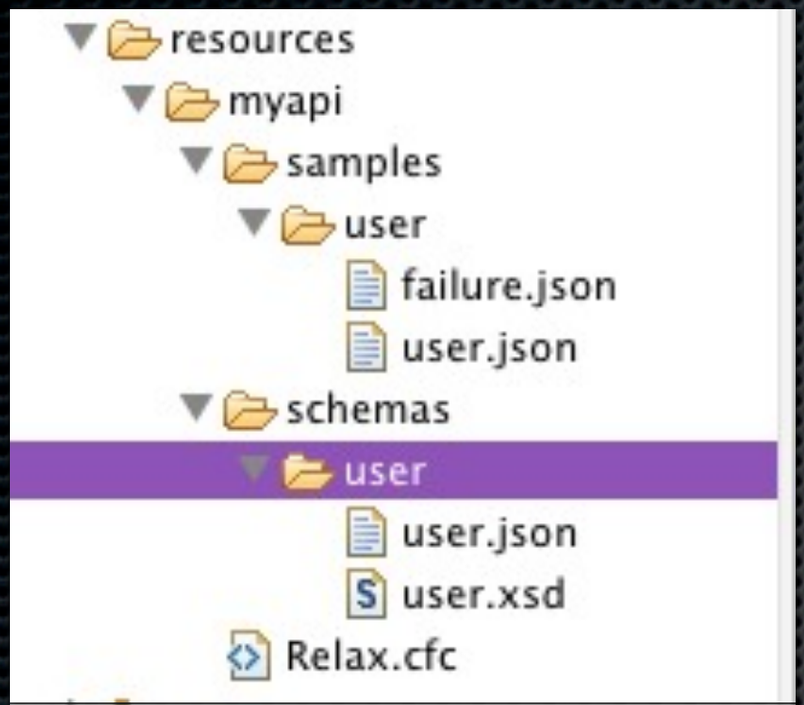
**relax/ModuleConfig.cfc**

# Relax Configuration

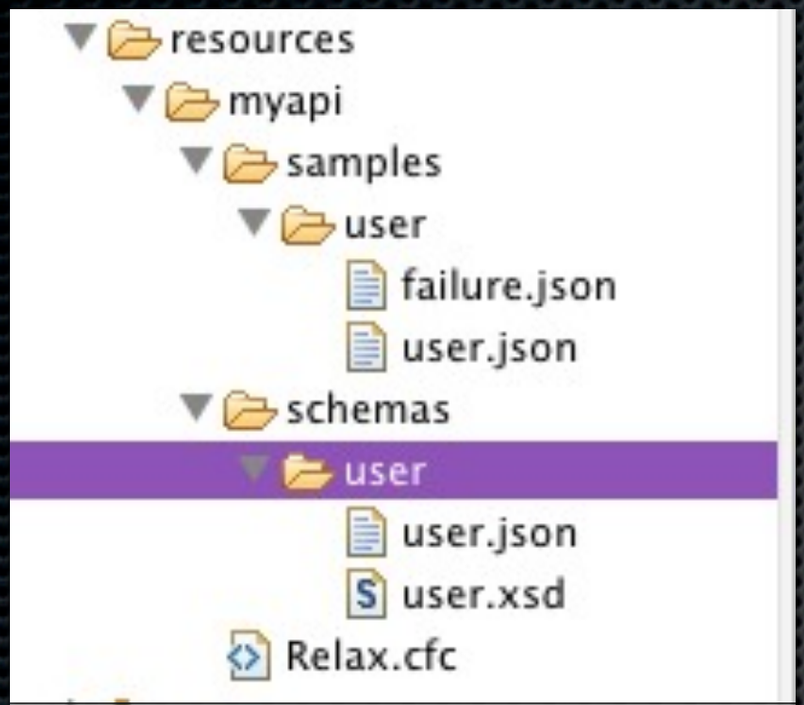
```
// Relax Configuration Settings
settings = {
    // Relax Version: DO NOT ALTER
    version = this.version,

    // Relax DSL component that has the resource definitions, this is an instanatiation path
    configCFC = "resources.myapi.Relax",
    // History stack size, the number of history items to keep on requests
    maxHistory = 10,
    // logbox integration information needed for log viewer to work
    // this means that it can read tables that are written using the logbox's DB Appender.
    relaxLogs = {
        // THE CF DATASOURCE NAME
        datasource = "relax",
        // THE DB TO USE FOR LOGS, AVAILABLE ADAPTERS ARE: mysql, mssql, postgres, oracle
        adapter = "mysql",
        // THE TABLE WHERE THE LOGS ARE
        table    = "api_logs",
        // PAGING MAX ROWS
        maxRows = 50,
        // PAGING CARROUSEL BANDGAP
        bandGap = 3
    }
};
```

relax/ModuleConfig.cfc

# Prepare API Resources



# Prepare API Resources




- Create directory structure to model your API
  - A CFC that will model your API
  - Samples/Schemas
  - Organize by resources

# Relax Modeling DSL

# Relax Modeling DSL

- Create a simple CFC
- Create a set of public properties (Implicit structs and arrays)

# Relax Modeling DSL

- Create a simple CFC
- Create a set of public properties (Implicit structs and arrays)

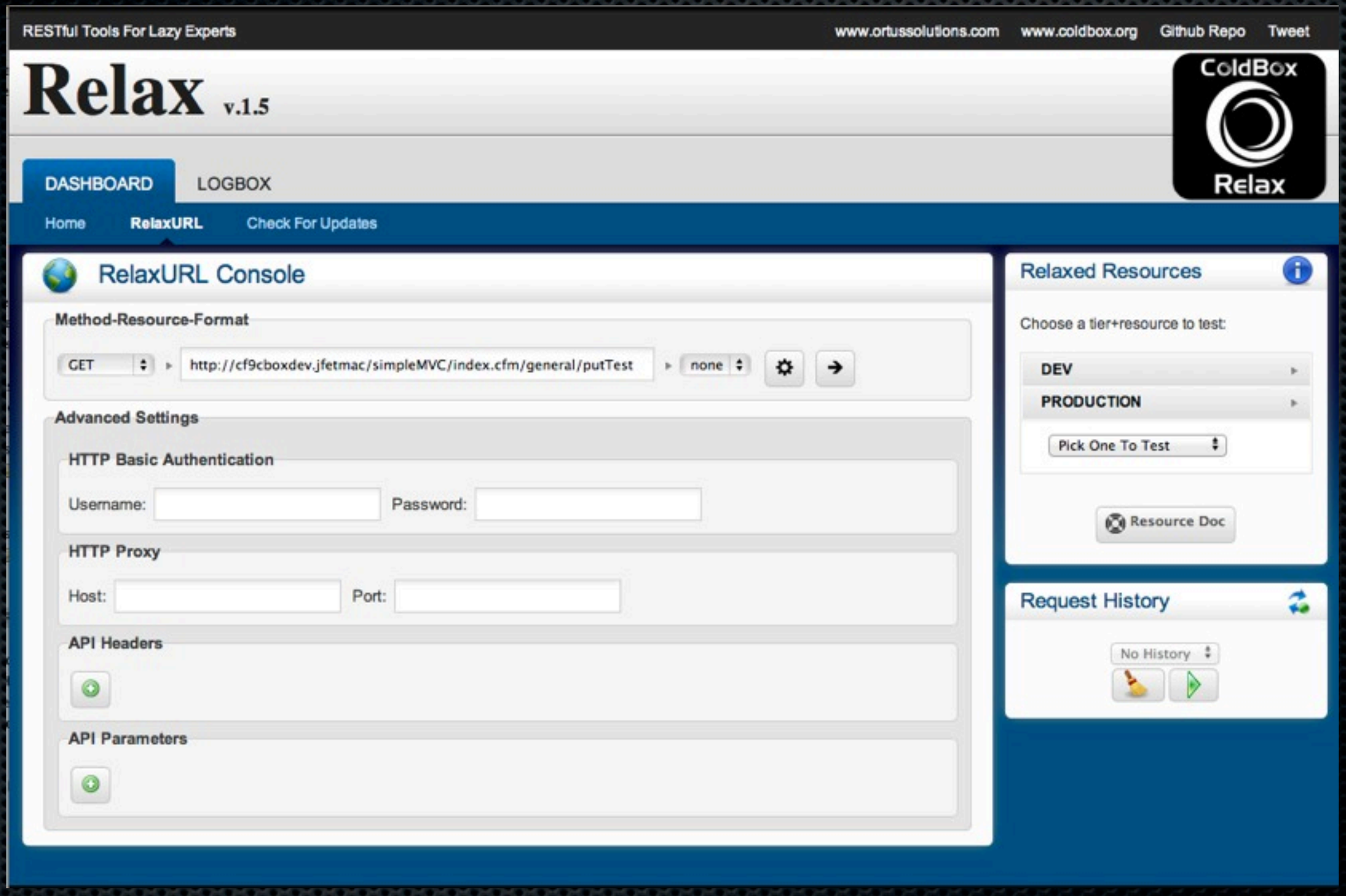
| Key | Description |
|---|---|
| **this.relax** | Service title, description, tiered entry points, extension detection, valid format extensions, etc |
| **this.globalHeaders** | Define service global headers: [name,description,required,default,type] |
| **this.globalParameters** | Definite service global parameters: [name,description,required,default,type] |
| **this.resources** | A collection of resources that identify your service |

# Resource Modeling DSL

# Resource Modeling DSL

| Resource Keys | Description |
|---|---|
| **pattern** | The SES pattern also used for coldbox route generation |
| **handler** | The handler that responds to the pattern used for route generation |
| **action** | Simple or RESTful action used for route generation |
| **placeholders** | Collection that describes the pattern placeholders: [name,description,type,required,default] |
| **description** | Resource description, can use HTML |
| **methods** | HTTP methods implemented by this resource |
| **headers** | A collection of headers  this resource responds to |
| **parameters** | A collection of parameters this resource responds to |
| **response** | Holds more metadata about resource response schemas and samples |
| **response.schemas** | A collection of different schema representations: [format,description,body] |
| **response.samples** | A collection of different sample representations: [format,description,body] |

# Start it up


RESTful Tools For Lazy Experts
www.ortussolutions.com
www.coldbox.org
Github Repo
Tweet
Relax v.1.5
DASHBOARD
LOGBOX
Home
RelaxURL
Check For Updates
RelaxURL Console
Method-Resource-Format
GET
http://cf9cboxdev.jfetmac/simpleMVC/index.cfm/general/putTest
none
Advanced Settings
HTTP Basic Authentication
Username:
Password:
HTTP Proxy
Host:
Port:
API Headers
API Parameters
Relaxed Resources
Choose a tier+resource to test:
DEV
PRODUCTION
Pick One To Test
Resource Doc
Request History
No History

# Start it up

RESTful Tools For Lazy Experts

www.ortussolutions.com www.coldbox.org Github Repo Tweet

**Relax** v.1.5

DASHBOARD LOGBOX

Home RelaxURL Check For Updates

## Welcome To Relax

Welcome to your Relax Console. We have succesfully read the *Relax DSL CFC* you configured at **resources.myapi.Relax.cfc**. Below is the RESTful documentation. From here you can also tap into our **RelaxURL** console to test the resources or any web RESTful service or view our awesome **RelaxLogs** log viewer.

- Service Overview
- Defined Resources
- HTTP Codes
- Generated Routes

# My RESTFul Service

A very cool RESTFul Service

## Service Entry Point(s)

DEV

```
1 | http://dev.myapi.com
```

PRODUCTION

```
1 | http://www.myapi.com
```

## API Return Formats

This service can detect the incoming resource extension in order to provide to you the resource represented according to the extension:

```
1 | resource.{format}
2 | resource.json
3 | resource.xml
```

| Service Extension Detection: | Yes |
|---|---|

## RelaxURL

**RelaxURL**, pronounced "Relax-ER", is a utility to help you test against your Relaxed RESTful service definitions or any other RESTful web service.

## Export Lounge

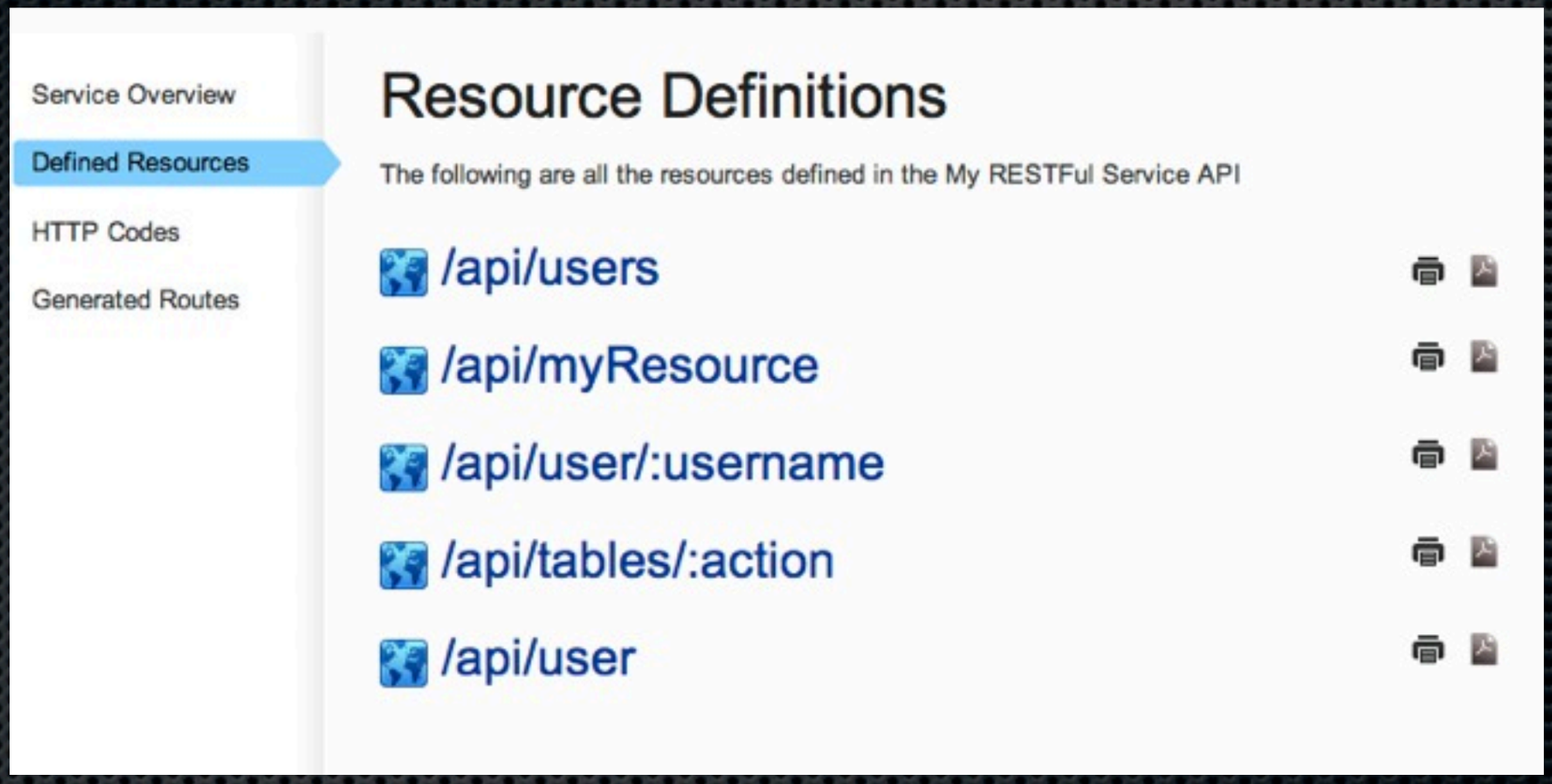
You can export your Relaxed Service Documentation in several formats:

## Need Help?

Ortus Solutions A Software Revolution

**Ortus Solutions** is the company behind anything ColdBox. Need professional support, architecture analysis, code reviews, custom development or anything ColdFusion, ColdBox related? Contact us, we are here to help!

# Start it up


Service Overview
Defined Resources
HTTP Codes
Generated Routes
Resource Definitions
The following are all the resources defined in the My RESTFul Service API
/api/users
/api/myResource
/api/user/:username
/api/tables/:action
/api/user

# Start it up

Service Overview

Defined Resources

HTTP Codes

Generated Routes

## Generated ColdBox Routes

The following are the generated ColdBox URL Mapping routes that you can use to copy and paste into your application.

```
// AutoReload defautls to true since you must be developing, else true fo
setAutoReload( true );

// Sets automatic route extension detection and places the extension in the
setExtensionDetection( true );

// Setup the list of valid extensions for this application
setValidExtensions('xml,json,jsont,wddx');

// Throw exception 406 when an invalid extension is detected?
setThrowOnInvalidExtension( false );

// Route Definitions

addRoute(pattern="/api/users",handler="rest.user",action="list");

addRoute(pattern="/api/myResource",handler="rest.myUser",action={POST="crea

addRoute(pattern="/api/user/:username",handler="rest.user",action="{'get':'

addRoute(pattern="/api/tables/:action",handler="rest.table");

addRoute(pattern="/api/user",handler="rest.user");
```

# Start it up

## RelaxLogs

This is the contents of the log table you configured Relax with. You can page through it and also do dynamic filtering of the current viewable window.

Log Filter:

**Total Records:** 70 **Total Pages:** 2

*Max Records* 50

1 2

| Type | Logdate | Message | Actions |
|---|---|---|---|
| | 8/2/11 2:45:41 PM | Request initiated to relax:logs.index | |
| | 8/2/11 2:45:15 PM | Request initiated to relax:home.index | |
| | 8/2/11 2:43:57 PM | Request initiated to relax:home.relaxer | |
| | 8/2/11 2:09:12 PM | Request initiated to relax:home.index | |
| | 8/1/11 7:46:29 PM | Request initiated to relax:home.index | |
| | 8/1/11 7:46:25 PM | CacheBox Cache: TEMPLATE has been initialized successfully for operation | |
| | 8/1/11 7:46:24 PM | CacheBox Cache: default has been initialized successfully for operation | |
| | 7/25/11 10:20:29 PM | CacheBox Cache: default has been initialized successfully for operation | |
| | 7/25/11 10:20:29 PM | CacheBox Cache: TEMPLATE has been initialized successfully for operation | |
| | 7/25/11 10:20:29 PM | Request initiated to relax:home.relaxer | |
| | 7/22/11 11:36:05 AM | Request initiated to relax:home.relax | |
| | 7/22/11 11:35:53 AM | Request initiated to relax:home.index | |
| | 7/22/11 11:35:52 AM | CacheBox Cache: default has been initialized successfully for operation | |
| | 7/22/11 11:35:52 AM | CacheBox Cache: TEMPLATE has been initialized successfully for operation | |
| | 7/14/11 3:39:25 PM | Request initiated to relax:home.relaxer | |

## RelaxLogs Settings

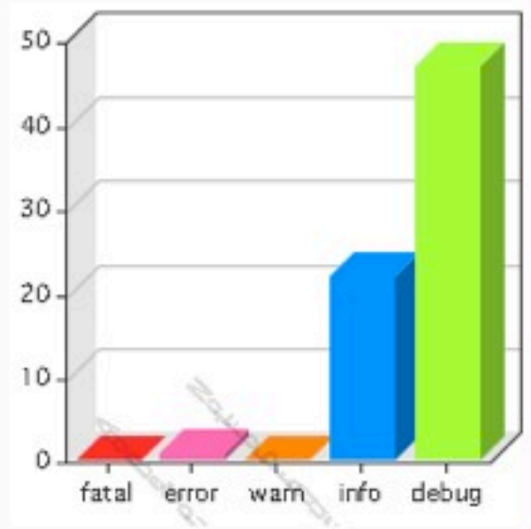
The following are the settings the RelaxLogs are configured with:

| | |
|---|---|
| **Datasource:** | relax |
| **Adapter:** | mysql |
| **Table:** | api_logs |
| **Max Rows:** | 50 |

Purge Logs

## Severity Showdown!

Checkout your Log Severity Showdown




## RelaxLogs Setup

To learn more how to configure Relax to read your LogBox logs then visit our help section.

# RESTful Routing

# RESTful Routing

```
addRoute(pattern="API/settings",  handler="General",action="settings");
addRoute(pattern="API/echo/:name?",handler="General",action="{'POST':'echo'}");
addRoute(pattern="API/user/:username?",
        handler="admin:rest.user",
        action="{'GET':'index',‘PUT’:‘update’,‘POST’:‘add’,‘DELETE’:‘remove’}");
```

- Pattern = The URI of the resource (Dynamic)
- Handler = The controller to handle the event
- Action = Implicit Struct or JSON Struct
  - Maps HTTP verbs to handler methods

# RESTful Handlers

# RESTful Handlers

- Create your domain objects and TEST them!
- Create a base REST handler
  - AOP interceptions
  - Data uniformity
  - Error uniformity
- Create our RESTful Event Handlers

# Base Handler

# Base Handler

```
component{

    function aroundHandler(event,targetAction){
        log.debug("Incoming request for route: #event.getCurrentRoute()#");
        event.setValue("results", prepareResults());

        arguments.targetAction(event);

        event.renderData(data=event.getValue("results"),type="json");
    }
    private function prepareResults(){
        var results = {
            error      = false,
            messages   = "",
            data       = ""
        };
        return results;
    }
    function onError(event,faultAction,exception){
        var rc = event.getCollection();
        rc.results.error   = true;
        rc.results.messages = "The action:#faultaction# failed:
                               #exception.detail# #exception.message#";

        log.error(rc.results.messages, exception);

        // Headers
        event.setHTTPHeader(500,rc.results.messages);
        event.renderData(data=rc.results,type="json");
    }
}
```

# Base Handler

```
component{

    function aroundHandler(event,targetAction){
        log.debug("Incoming request for route: #event.getCurrentRoute()#");
        event.setValue("results", prepareResults());

        arguments.targetAction(event);

        event.renderData(data=event.getValue("results"),type="json");
    }
    private function prepareResults(){
        var results = {
            error      = false,
            messages   = "",
            data       = ""
        };
        return results;
    }
    function onError(event,faultAction,exception){
        var rc = event.getCollection();
        rc.results.error   = true;
        rc.results.messages = "The action:#faultaction# failed:
                               #exception.detail# #exception.message#";

        log.error(rc.results.messages, exception);

        // Headers
        event.setHTTPHeader(500,rc.results.messages);
        event.renderData(data=rc.results,type="json");
    }
}
```

# Base Handler

```
component{

    function aroundHandler(event,targetAction){
        log.debug("Incoming request for route: #event.getCurrentRoute()#");
        event.setValue("results", prepareResults());

        arguments.targetAction(event);

        event.renderData(data=event.getValue("results"),type="json");
    }
    private function prepareResults(){
        var results = {
            error      = false,
            messages   = "",
            data       = ""
        };
        return results;
    }
    function onError(event,faultAction,exception){
        var rc = event.getCollection();
        rc.results.error   = true;
        rc.results.messages = "The action:#faultaction# failed:
                               #exception.detail# #exception.message#";

        log.error(rc.results.messages, exception);

        // Headers
        event.setHTTPHeader(500,rc.results.messages);
        event.renderData(data=rc.results,type="json");
    }
}
```

# Base Handler

```
component{

    function aroundHandler(event,targetAction){
        log.debug("Incoming request for route: #event.getCurrentRoute()#");
        event.setValue("results", prepareResults());

        arguments.targetAction(event);

        event.renderData(data=event.getValue("results"),type="json");
    }
    private function prepareResults(){
        var results = {
            error      = false,
            messages   = "",
            data       = ""
        };
        return results;
    }
    function onError(event,faultAction,exception){
        var rc = event.getCollection();
        rc.results.error   = true;
        rc.results.messages = "The action:#faultaction# failed:
                               #exception.detail# #exception.message#";

        log.error(rc.results.messages, exception);

        // Headers
        event.setHTTPHeader(500,rc.results.messages);
        event.renderData(data=rc.results,type="json");
    }
}
```

# Base Handler

```
component{

    function aroundHandler(event,targetAction){
        log.debug("Incoming request for route: #event.getCurrentRoute()#");
        event.setValue("results", prepareResults());

        arguments.targetAction(event);

        event.renderData(data=event.getValue("results"),type="json");
    }
    private function prepareResults(){
        var results = {
            error      = false,
            messages   = "",
            data       = ""
        };
        return results;
    }
    function onError(event,faultAction,exception){
        var rc = event.getCollection();
        rc.results.error   = true;
        rc.results.messages = "The action:#faultaction# failed:
                               #exception.detail# #exception.message#";

        log.error(rc.results.messages, exception);

        // Headers
        event.setHTTPHeader(500,rc.results.messages);
        event.renderData(data=rc.results,type="json");
    }
}
```

# Base Handler

```
component{

    function aroundHandler(event,targetAction){
        log.debug("Incoming request for route: #event.getCurrentRoute()#");
        event.setValue("results", prepareResults());

        arguments.targetAction(event);

        event.renderData(data=event.getValue("results"),type="json");
    }
    private function prepareResults(){
        var results = {
            error      = false,
            messages   = "",
            data       = ""
        };
        return results;
    }
    function onError(event,faultAction,exception){
        var rc = event.getCollection();
        rc.results.error   = true;
        rc.results.messages = "The action:#faultaction# failed:
                               #exception.detail# #exception.message#";

        log.error(rc.results.messages, exception);

        // Headers
        event.setHTTPHeader(500,rc.results.messages);
        event.renderData(data=rc.results,type="json");
    }
}
```

# Base Handler

```
component{

    function aroundHandler(event,targetAction){
        log.debug("Incoming request for route: #event.getCurrentRoute()#");
        event.setValue("results", prepareResults());

        arguments.targetAction(event);

        event.renderData(data=event.getValue("results"),type="json");
    }
    private function prepareResults(){
        var results = {
            error      = false,
            messages   = "",
            data       = ""
        };
        return results;
    }
    function onError(event,faultAction,exception){
        var rc = event.getCollection();
        rc.results.error   = true;
        rc.results.messages = "The action:#faultaction# failed:
                               #exception.detail# #exception.message#";

        log.error(rc.results.messages, exception);

        // Headers
        event.setHTTPHeader(500,rc.results.messages);
        event.renderData(data=rc.results,type="json");
    }
}
```

# RESTful Event Handlers

# RESTful Event Handlers

```
component cacheTimeout="30"{
    // DI
    property name="userService" inject;
    // HTTP Security
    this.allowedMethods = {
        index = "get,post", echo="post,put", remove="delete"
    };

    function index(event){
        var rc = event.getCollection();
        rc.results.data = userService.get(id=event.getValue("id",0));
    }

    function echo(event){
        rc.results.data = "My name is #event.getValue("name","nobody")#");
    }
}
```

- Handler Persistence
- Autowire Domain Objects
- HTTP Action Security
- AOP Transformations

# RESTful Event Handlers

```
component cacheTimeout="30"{
    // DI
    property name="userService" inject;
    // HTTP Security
    this.allowedMethods = {
        index = "get,post", echo="post,put", remove="delete"
    };

    function index(event){
        var rc = event.getCollection();
        rc.results.data = userService.get(id=event.getValue("id",0));
    }

    function echo(event){
        rc.results.data = "My name is #event.getValue("name","nobody")#");
    }
}
```

- Handler Persistence
- Autowire Domain Objects
- HTTP Action Security
- AOP Transformations

# RESTful Event Handlers

```
component cacheTimeout="30"{
    // DI
    property name="userService" inject;
    // HTTP Security
    this.allowedMethods = {
        index = "get,post", echo="post,put", remove="delete"
    };

    function index(event){
        var rc = event.getCollection();
        rc.results.data = userService.get(id=event.getValue("id",0));
    }

    function echo(event){
        rc.results.data = "My name is #event.getValue("name","nobody")#");
    }
}
```

- Handler Persistence
- Autowire Domain Objects
- HTTP Action Security
- AOP Transformations

# RESTful Results

POST http://app.com/echo {name=”Louis Mahoney”}
StatusCode = 200

GET http://app.com/echo/?name=hacker
StatusCode = 403

# RESTful Results

POST http://app.com/echo {name="Louis Mahoney"}
StatusCode = 200

```
{error:'false', messages : '', data: 'My Name is Louis Mahoney'}

{error:'false', messages : '', data: 'My Name is Josue TierraAlta'}
```

GET http://app.com/echo/?name=hacker
StatusCode = 403

```
{error:'true', messages : 'HTTP Verb Security Action Echo cannot be executed via a
GET', data: ''}
```

# Integration Testing

# Integration Testing

- Top-Down test of your entire app
- Target specific events
- Assert transformations and data retrievals
- Code confidence
- HTTP Verb Mocking
- Data Mocking

# Integration Testing

# Integration Testing

```
component extends="coldbox.system.testing.BaseTestCase"{


    function testEchoNoName(){
        event = execute("general.echo");
        assertEquals();
    }

    function testEchoWithName(){
        URL.name = "Louis Mahoney";
        event = execute("general.echo");
        assertEquals();
    }

    function testEchoWithInvalidVerb(){
        mockContext = getMockBox()
                        .prepareMock(controller.getRequestService().getContext());
        mockContext.$("getHTTPVerb","GET");
        event = execute("general.echo");
        assertEquals();
    }

}
```

# Integration Testing

```
component extends="coldbox.system.testing.BaseTestCase"{

    function testEchoNoName(){
        event = execute("general.echo");
        assertEquals();
    }

    function testEchoWithName(){
        URL.name = "Louis Mahoney";
        event = execute("general.echo");
        assertEquals();
    }

    function testEchoWithInvalidVerb(){
        mockContext = getMockBox()
                        .prepareMock(controller.getRequestService().getContext());
        mockContext.$("getHTTPVerb","GET");
        event = execute("general.echo");
        assertEquals();
    }

}
```

# Integration Testing

```
component extends="coldbox.system.testing.BaseTestCase"{

    function testEchoNoName(){
        event = execute("general.echo");
        assertEquals();
    }

    function testEchoWithName(){
        URL.name = "Louis Mahoney";
        event = execute("general.echo");
        assertEquals();
    }

    function testEchoWithInvalidVerb(){
        mockContext = getMockBox()
                        .prepareMock(controller.getRequestService().getContext());
        mockContext.$("getHTTPVerb","GET");
        event = execute("general.echo");
        assertEquals();
    }

}
```

# Integration Testing

```
component extends="coldbox.system.testing.BaseTestCase"{

    function testEchoNoName(){
        event = execute("general.echo");
        assertEquals();
    }

    function testEchoWithName(){
        URL.name = "Louis Mahoney";
        event = execute("general.echo");
        assertEquals();
    }

    function testEchoWithInvalidVerb(){
        mockContext = getMockBox()
                        .prepareMock(controller.getRequestService().getContext());
        mockContext.$("getHTTPVerb","GET");
        event = execute("general.echo");
        assertEquals();
    }

}
```

# ColdBox Resources

- Official Site
  - www.coldbox.org
- Documentation
  - wiki.coldbox.org
- Google Group
  - groups.google.com/group/coldbox
- Training
  - www.coldbox.org/training
- Professional Support
  - www.ortussolutions.com

# ColdBox Resources

- Official Site
  - www.coldbox.org
- Documentation
  - wiki.coldbox.org
- Google Group
  - groups.google.com/group/coldbox
- Training
  - www.coldbox.org/training
- Professional Support
  - www.ortussolutions.com

**Luis Majano &**
**Ortus Solutions, Corp**
lmajano@ortussolutions.com

# ColdBox Resources

- Official Site
  - www.coldbox.org
- Documentation
  - wiki.coldbox.org
- Google Group
  - groups.google.com/group/coldbox
- Training
  - www.coldbox.org/training
- Professional Support
  - www.ortussolutions.com


**Luis Majano &**
**Ortus Solutions, Corp**
lmajano@ortussolutions.com

# Q & A

# Thanks!

# Q & A


# Thanks!