COLDFUSION SUMMIT EAST 2018

POWER OF SIMPLICITY IN FW/1

WHO AM I?

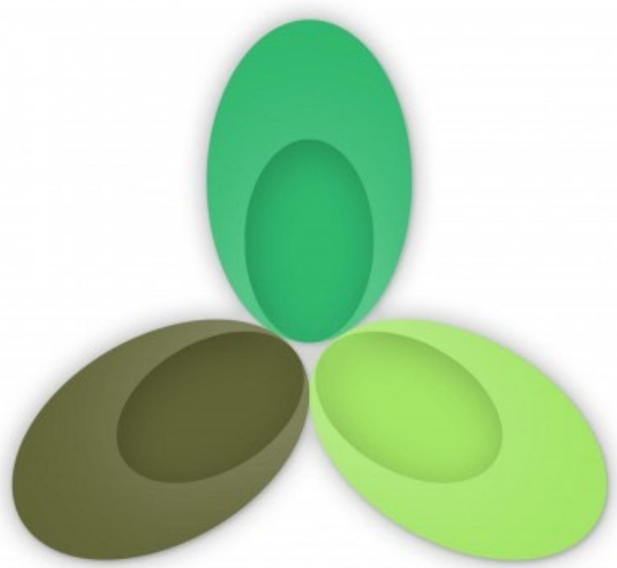
- Full Stack Web Developer of 15+ years
- HD Web Studio Owner and Visionary
- Mother to a Teenage Daughter 😳
- Travel Enthusiast
- U2 Fan
- Amateur Latin Dancer

WHAT IS THIS ABOUT?

- Getting started with FW/1
- Brief setup and installation outline
- Advanced features of the framework (subsystems, DI/1, AOP/1)
- REST support for APIs
- Skinning and multi-site setup

GETTING STARTED WITH FW/1

- Lightweight, one file MVC framework
- Convention over Configuration
- No XML
- SES URLs support
- REST support (enhanced in v4.0)



INSTALLATION AND SETUP

"Just this once, let it be easy." –
Cynthia Lord, Rules

INSTALLATION

- FW/1 4.2 supports Adobe ColdFusion 10 or later and Lucee 4.5.0 or later.
- The framework folder should either be copied to your webroot (the simplest way to get started) or else made accessible via a mapping for **/framework**
- Extend your Application.cfc to FW/1 core file
- Alternatively install via CommandBox:

```
box install fw1
```

APPLICATION OUTLINE

- FW/1 assumes your application is made up of ***sections*** and ***items*** called by ***action***:

  index.cfm?action=main.default

- Default section is ***main*** & default Item is ***default*** (this can be overwritten)

- Supports SES URLs

  index.cfm/blog/list => blog/list

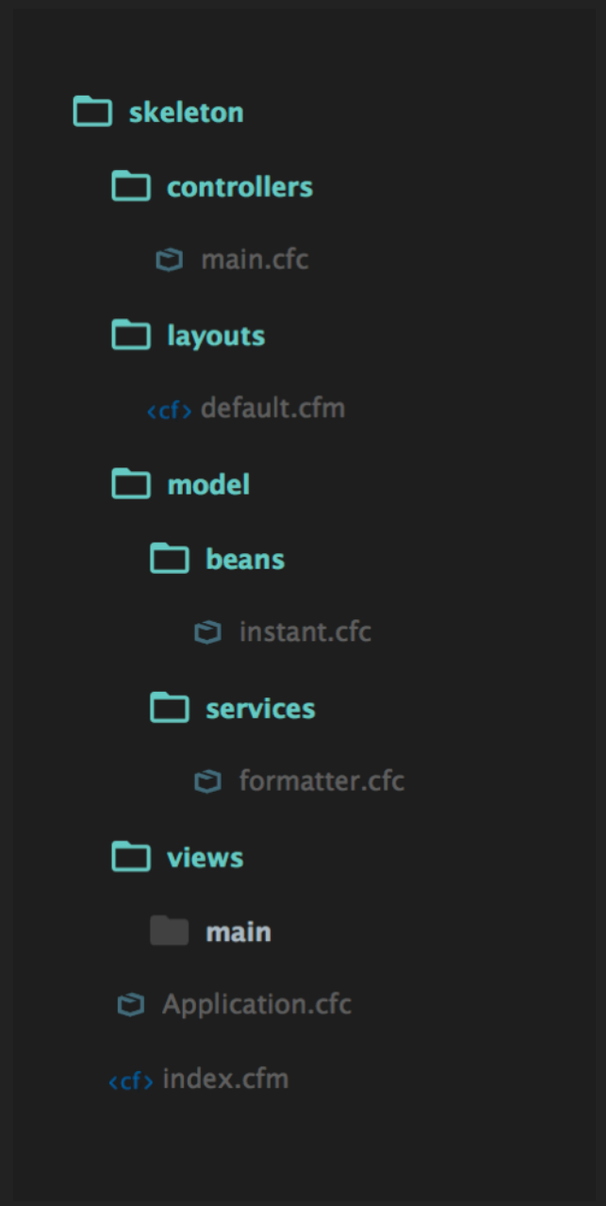
  (using server rewrite engine, set SESOmitIndex = true)

FOLDER STRUCTURE

- Controllers - traffic cops
- Layouts - wrappers
- Views - user interaction
- Model - database relationships, business logic

?action=main.item

main.cfc - controller, if present, invokes the item() method on it (and then displays the matching view)



AND WE'RE OFF TO THE RACES

APPLICATION.CFC

- **this = {}**
- **variables.framework = {}**
- **setupApplication()** instead of **onApplicationStart()**
- **setupSession()** instead of **onSessionStart()**
- **setupRequest()** instead of **onRequestStart()**
- **before()**
- **setupView()**
- **setupResponse()**
- **after()**

CONTROLLERS

To access FW/1 API inside a controller and/or bean factories
add “accessors=true” to your component tag, and declare dependencies with the property keyword:

```
property beanFactory; // make your controller bean factory aware
property framework;   // make your controller framework aware
```

- **Structure:**

  before()

  item()

  after()

Each controller method is passed the request context as a single argument called rc, of type struct, pass it into individual methods as needed.

COMMONLY USED FUNCTIONS

CONTROLLERS

- Functions:

variables.fw.populate()

populate (cfc, **keys** = “”, **trustKeys** = false, **trim** = false, **deep** = false, **properties** = ‘’)

variables.fw.redirect()

redirect (**action**, **preserve** = “[none,all]”, **append** = “[none,all]”, **path**, **queryString**, **statusCode** = ‘302’, **header**)

variables.fw.renderData()

skips views and layouts and instead renders data in the specified content type format ("html", "json", "jsonp", "rawjson", "xml", or “text")

variables.fw.setView(), variables.fw.setlayout()

```
setView(“emails/template", {emailView = “emails/thankYouLetter”});
```

variables.fw.abortController()

COMMONLY USED FUNCTIONS

VIEWS AND LAYOUTS

- Functions:

view()

```
#view('calendar/categories')#
#view('thisView',{args='someArgs'},view('onMissingView'))#
```

buildURL()

```
#buildURL( action = 'section.item', queryString =
'arg=val' )#
```

layout()

```
#layout("news/wide",articles)#
```

getBeanFactory()

```
#getBeanFactory().getBean("article")#
```

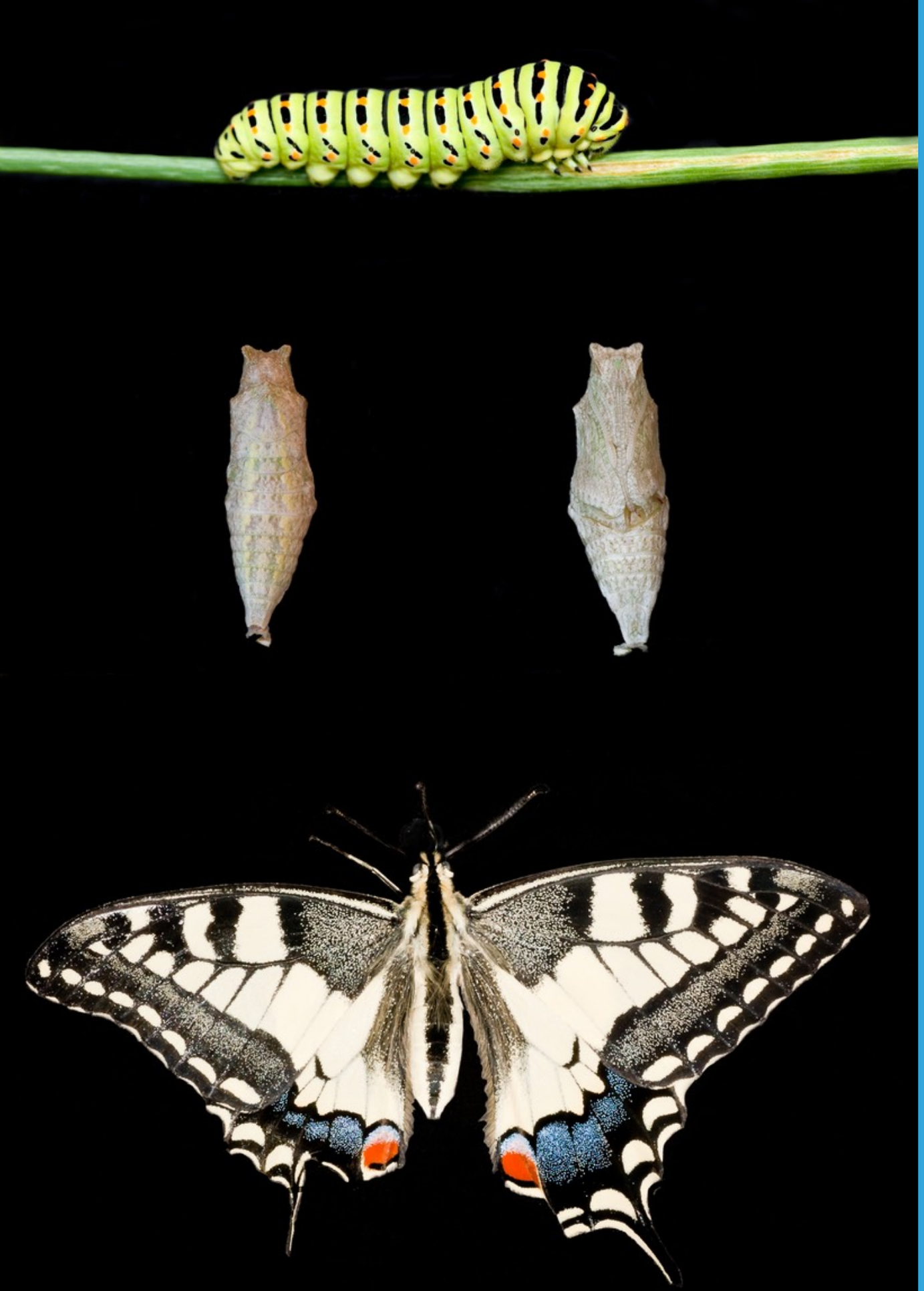
- Variables:

body

rc[]

local[]

framework[]



LIFECYCLE

Just when the caterpillar thought her life was over she began to fly . . .

ENVIRONMENTS

- **Environment control** - the ability to automatically detect your application environment (development, production, etc) and adjust the framework configuration accordingly.

  getEnvironment()

  variables.framework.environments

  setupEnvironment(env)

```
public function getEnvironment() {
    if ( findNoCase( "www", CGI.SERVER_NAME ) ) return "prod";
    if ( findNoCase( "dev", CGI.SERVER_NAME ) ) return "dev";
    else return "dev-qa";
}
variables.framework.environments = {
    dev = { reloadApplicationOnEveryRequest = true, error = "main.detailederror" },
    dev-qa = { reloadApplicationOnEveryRequest = false },
    prod = { password = "supersecret" }
};
```

SHARE RESOURCES

“Good programmers know what to write. Great ones know what to rewrite (and reuse)”

– Eric S. Raymond

SUBSYSTEMS

Subsystems give you a way of modularizing your FW/1 application as it grows.

- *Legacy subsystems:*
    - Each subsystem was in a top-level folder of the application.
    - Your “main” application was just a subsystem, like any other (by default it was the "home"subsystem).
    - In addition to the normal three-tier cascading layout for each subsystem, you could specify a subsystem that contained a fourth tier layout to wrap every page (via siteWideLayoutSubsystem).

SUBSYSTEMS 2.0

- Subsystems live in a subsystems folder.
- Your “main” application is just a regular FW/1 application – the top-level application.
- In addition to the normal three-tier cascading layout for each subsystem, your main application’s site-wide layout is also applied (i.e., layouts/default.cfm).

ENABLING

- Do not need to do anything to enable
- Requests for actions like **module:section.item** will cause FW/1 to automatically look for a subsystem called module in the subsystems folder of your application
- Legacy subsystems can still be used by setting **usingSubsystems** to **true**

ACCESSING

- index.cfm?action=module:section.item
- SES:

  index.cfm/module:section/item

- Further improve with custom rewrite rules and/or adding routes to the application.cfc:

  routes = [{'/module/*' = '/module:\1'}]

  /module/section/item

CONFIGURING

- Optional method that can be declared in Application.cfc:

```
function setupSubsystem(subsystem) {}
```

```
function setupSubsystem( subsystem ) {
    var bfConfigFilePath = '/subsystems/' & subsystem & '/config/coldspring.xml.cfm';
    // conditionally load bean factory for this subsystem by convention:
    try {
        if ( fileExists(expandPath('./') & bfConfigFilePath) ) {
            var bf = new coldspring.beans.DefaultXmlBeanFactory();
            bf.loadBeans( bfConfigFilePath );
            bf.setParent( getDefaultBeanFactory() );
            setSubsystemBeanFactory( subsystem, bf );
        }
    } catch ( any e ) {
        // ignore exceptions caused by bad paths etc
    }
}
```

APPLICATION OUTLINE

- Controllers

  subsystems/module/controllers/main.cfc

- Views

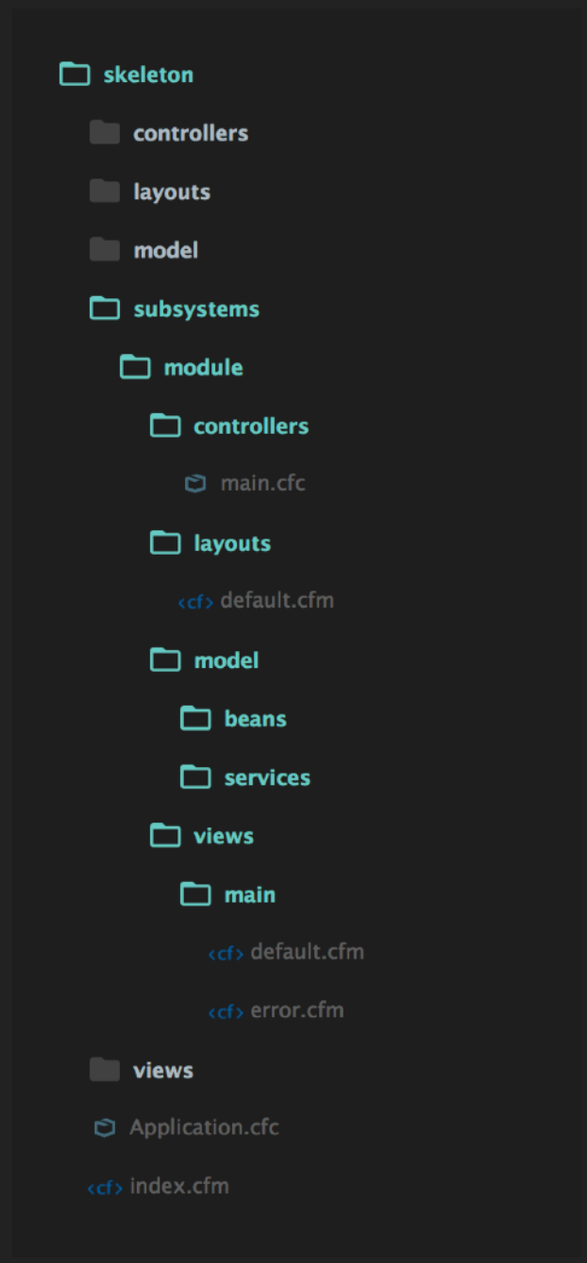
  subsystems/module/views/main/default.cfm

- Layouts

  subsystems/module/layouts/default.cfm

- Model

  subsystems/module/model/services/login.cfc



COMBINE THE FLAVORS

USING DI/1

DI/1 – default bean factory for FW/1

Bean Factory is a service that creates beans for you, so you don't have to use new() or createObject(), and populates them with any dependencies.

- *DI/1 - is a simple, convention-based Dependency Injection framework.*
- *Convention:*

  By default, FW/1 creates an instance of DI/1 to use as its bean factory, to manage controller CFCs and also CFCs that are part of your application model.

- *Configuration:*

  diEngine **diComponent** **diLocations** **diConfig**

DI/1 CONVENTIONS

- *CFCs found in a folder called '**beans**' are assumed to be transients; otherwise CFCs are assumed to be singletons.*
- *Bean Aliases:*
  - /model/beans/product.cfc would get the alias "productBean"
  - /model/services/product.cfc would get the alias "productService"

(setting can be overwritten)

- *Autowiring:*

```
component accessors=true {
    property userService;
    property securityService;
    //...
}
```

- *Direct Access:*

```
var user = fw.getBeanFactory().getBean(
    "user",
    {name : "Masha", email : "masha@hdwebstudio.com"}
);
```

USING AOP/1

Aspect Oriented Programming

AspectOriented Programming (AOP) provides a way to write "cross-cutting concerns" in a single, isolated place (an object) that is then applied automatically wherever it is needed.

Common Uses: logging, versioning, security.

- *AOP/1 is a simple Aspect Oriented Programming **extension** for DI/1 which allows you to define interceptors for your beans.*
- *Overwrite the default **di1** in framework variables to enable:*

  diEngine = "aop1"
- *Create interceptors and place them in model folder.*
- *Announce Interceptors in diConfig:*

```
diConfig = {
    interceptors =
          [{beanName = "pageService", interceptorName = "loggingInterceptor", methods = "read"}]
}
```

INVEST IN REST

When all else fails, take a nap.

REST SUPPORT IN FW/1

REST is an architectural style for designing distributed systems.

The easier your API is to consume, the more people will consume it.

- *Architecture*
  - *Endpoints and Filtering*
  - *Content Type*
  - *Status Codes*
- *Authentication and Stateless Principles*
- *HTTP access control (CORS)*

ARCHITECTURE, ENDPOINTS AND FILTERING

Designing Routes and API Controllers

- *Routes*

```
routes = [
    { "$RESOURCES" = "dogs,cats" };
]
```

will automatically generate a standard set of routes for each of the listed resources:

```
{ "$GET/dogs/$" = "/dogs/default" },
{ "$GET/dogs/new/$" = "/dogs/new" },
{ "$POST/dogs/$" = "/dogs/create" },
{ "$GET/dogs/:id/$" = "/dogs/show/id/:id" },
{ "$PATCH/dogs/:id/$" = "/dogs/update/id/:id", "$PUT/dogs/:id/$" = "/dogs/update/id/:id" },
{ "$DELETE/dogs/:id/$" = "/dogs/destroy/id/:id" },
{ "$*/dogs/$" = "/dogs/error" }
```

ARCHITECTURE, ENDPOINTS AND FILTERING

Designing Routes and API Controllers

- *Nested Routes*

```
variables.framework.routes = [
    { "$RESOURCES" = { resources = "posts", subsystem = "blog", nested = "comments" } },
];
```

results in all of the standard routes for "posts", and in addition generates nested routes for "comments"

```
{ "$GET/posts/:posts_id/comments" = "/comments/default/posts_id/:posts_id" }
```

ARCHITECTURE, ENDPOINTS AND FILTERING

Designing Routes and API Controllers

- *Default Routes*

  (can be overwritten)

```
variables.framework.resourceRouteTemplates = [
    { method = 'default', httpMethods = [ '$GET' ] },
    { method = 'new', httpMethods = [ '$GET' ], routeSuffix = '/new' },
    { method = 'create', httpMethods = [ '$POST' ] },
    { method = 'show', httpMethods = [ '$GET' ], includeId = true },
    { method = 'update', httpMethods = [ '$PUT','$PATCH' ], includeId = true },
    { method = 'destroy', httpMethods = [ '$DELETE' ], includeId = true },
    { method = 'error', httpMethods = [ '$*' ] }
];
```

ARCHITECTURE, ENDPOINTS AND FILTERING

Designing Routes and API Controllers

Filtering

?limit=10: Reduce the number of results returned to the Consumer (for Pagination)

?offset=10: Send sets of information to the Consumer (for Pagination)

?animal_type_id=1: Filter records which match the following condition (WHERE animal_type_id = 1)

?sortby=name&order=asc: Sort the results based on the specified attribute (ORDER BY `name` ASC)

Matching Controllers and Methods

Return data directly to the caller, bypassing views and layouts, using the renderData() function.

```
variables.fw.renderData().data( resultData ).type( contentType );
```

data() **type()** **header()** **statusCode()** **statusText()** **jsonpCallback()**

ARCHITECTURE

Content Type

Based on the type in renderData() function the Content-Type HTTP header is automatically set to:

Content Type	Headers
html	text/html; charset=utf-8
json	application/json; charset=utf-8
jsonp	application/javascript; charset=utf-8
rawjson	application/json; charset=utf-8
xml	text/xml; charset=utf-8
text	text/plain; charset=utf-8

ARCHITECTURE

Status Code and Status Message

```
variables.fw.renderData().data(local.notification).type("json").statusCode(404).statusText(local.errorMessage);
```

renderer() function

```
variables.fw.renderData().data(result).type( "json" );
if ( someCondition ) {
    variables.fw.renderer().header( "X-Result", "Condition Happened" );
}
```

AUTHENTICATION AND STATELESS PRINCIPLES

Headers Argument

A controller that wants to inspect the HTTP headers can be defined like this:

```
function item(rc, headers) {
    ...
}
```

The headers argument is a struct of HTTP headers, and is the same as you would get from calling **getHttpRequestData().headers**. This argument was introduced in release 4.0 to better support REST APIs.

HTTP ACCESS CONTROL (CORS – CROSS-ORIGIN RESOURCE SHARING)

OPTIONS Support

```
preflightOptions = true;
```

Access-Control-Allow-Origin

By default FW/1 returns * here. The optionsAccessControl.origin setting will override this.

Access-Control-Allow-Methods

Determined by inspecting the matching routes, e.g., GET, POST, OPTIONS.

Access-Control-Allow-Headers

By default FW/1 returns Accept, Authorization, Content-Type here.
The optionsAccessControl.headers setting will override this.

Access-Control-Allow-Credentials

By default FW/1 returns true here. The optionsAccessControl.credentials setting will override this.

Access-Control-Max-Age

By default FW/1 returns 1728000 here (20 days, in seconds).
The optionsAccessControl.maxAge setting will override this.

CUSTOMIZATION IS KING

SKINNING

Possible with overwriting customizeViewOrLayoutPath()

```
function customizeViewOrLayoutPath( pathInfo, type, fullPath ) {
    // fullPath is: '#pathInfo.base##type#s/#pathInfo.path#.cfm'
    var defaultPath = '#type#s/#pathInfo.path#.cfm';

    if ( fileExists( expandPath( request.subsystembase & '/skins/'
        & request.context.skin & '/' & defaultPath ) ) ) {

        return request.subsystembase & 'skins/' & request.context.skin & '/' & defaultPath;
    } else {
        return request.subsystembase & 'skins/default/' & defaultPath;
    }
}
```

MULTI-SITE SETUP

Possible with overwriting customizeViewOrLayoutPath()

```
function customizeViewOrLayoutPath(pathInfo, type, fullPath) {

    var basePath = '';
    var path = '';

    if (!fileExists(ExpandPath(arguments.fullPath))) {

        local.path = "/fw1_core" & arguments.fullPath;

    } else {

        local.path = arguments.fullPath;

    }

    return local.path;

}
```

RESOURCES

- http://framework-one.github.io - official FW/1 Documentation
- https://github.com/framework-one/fw1 - official FW/1 repository
- https://developer.mozilla.org/en-US/docs/Web/HTTP/Access_control_CORS - HTTP access control (CORS)

QUESTIONS?

- Masha Edelen
- masha@hdwebstudio.com
- @mashaedelen