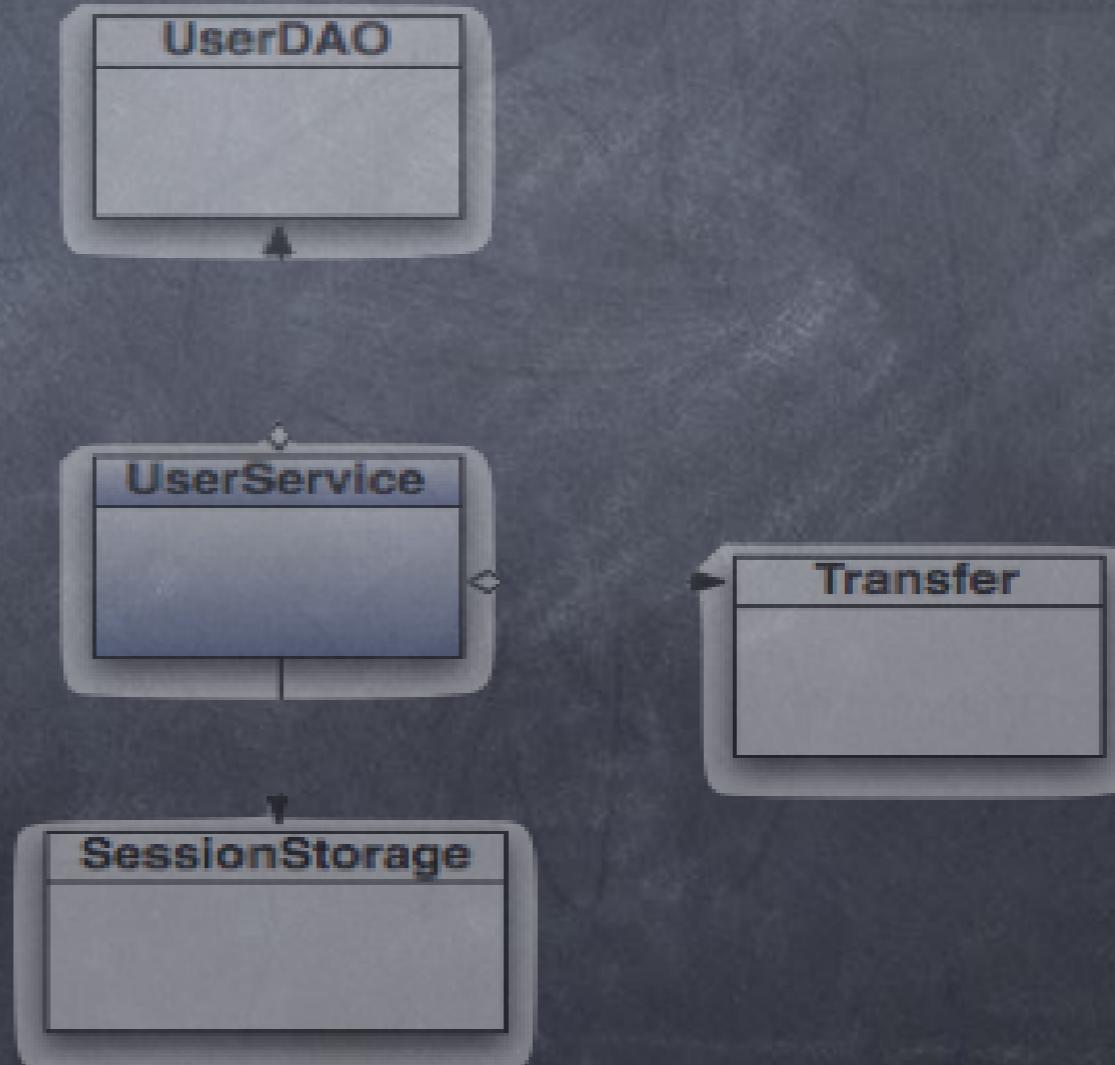




# Mock it baby!

---

## Leveraging Mock Objects





# The Obligatory Who am I?

- Husband
- Dad
- Coach
- Youth leader
- Co-owner of Computer  
Know How



- Member of  
TEAM ColdBox



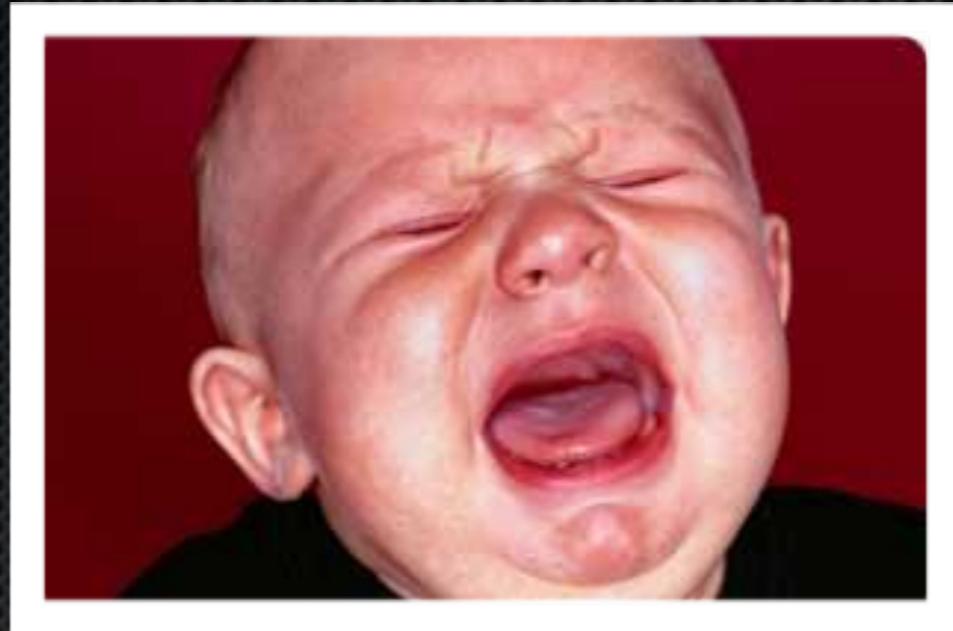


# What we will cover?

- Unit Testing Recap
- Testing Toolbox
- What is Mocking?
- What is a Mock Object
- Why Mock?
- Mocking Frameworks
- Practical Mocking with MockBox



# Why Unit Testing?



- Can improve code quality -> quick error discovery
- Code confidence via immediate verification
- Can expose high coupling
- Will encourage refactoring to produce > testable code
- **Remember:** Testing is all about behavior and expectations

# Unit Testing Basics

SUT

```
component{
    function add(a,b) {
        return a + b;
    }
}
```

Unit Test

```
component extends="mxunit.framework.TestCase){

    function setup(){
        calculator = new Calculator();
    }

    function testAdd(){
        r = calculator.add(1,4);
        assertEquals( 5, r );
    }

}
```

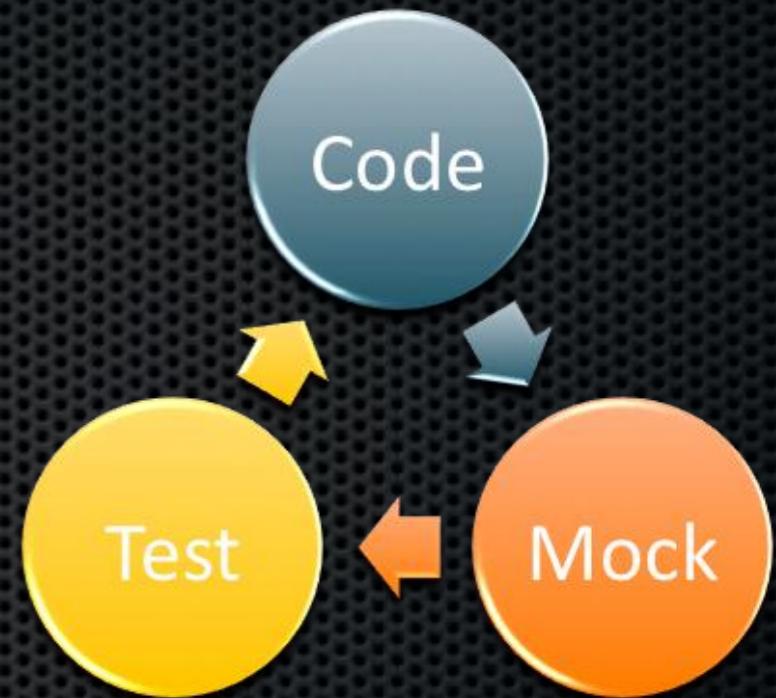
# TDD Process

- Test Driven Development
- Can be a new development paradigm for some
- Work from the IDE

**RED (Write the unit test, obviously fails)**

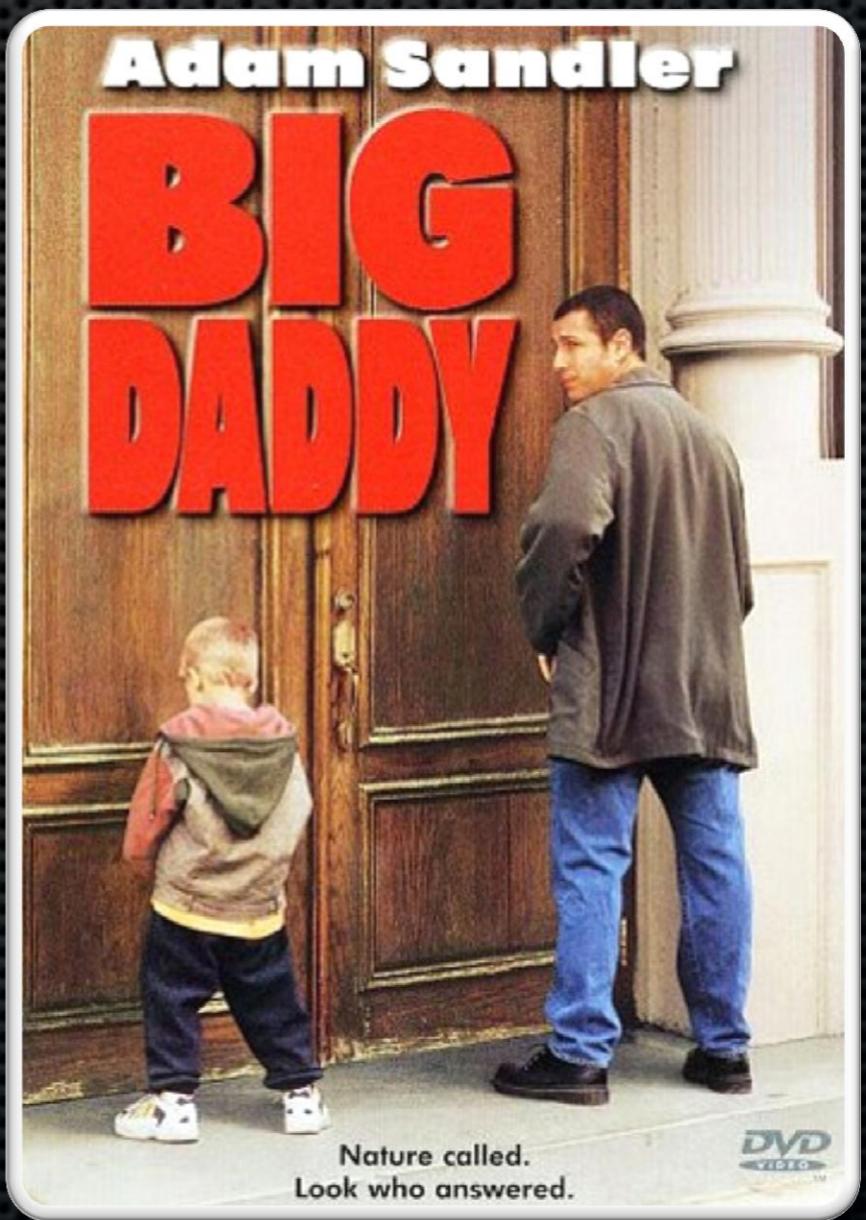
**GREEN (Write the code that makes the unit test pass)**

**RINSE AND REPEAT** (This is a cycle as you code)



# The 6 P's

- Prior
- Planning
- Prevents
- Piss
- Poor
- Performance



# Testing ToolBox

- MXUnit
- ColdFusion Builder OR CFEclipse
- A mocking framework (MockBox)
- ANT
- Integration Testing
- ColdBox
- Selenium
- BadBoy
- JMeter or Webstress Tool
- Jenkins





# What is Mocking?



Lets have a  
moment?

# Mocking



*“To treat with ridicule or contempt; to imitate, to counterfeit”*

# Mock Object

*"A mock object is an object that takes the place of a 'real' object in such a way that makes testing easier and more meaningful, or in some cases, **possible** at all"*

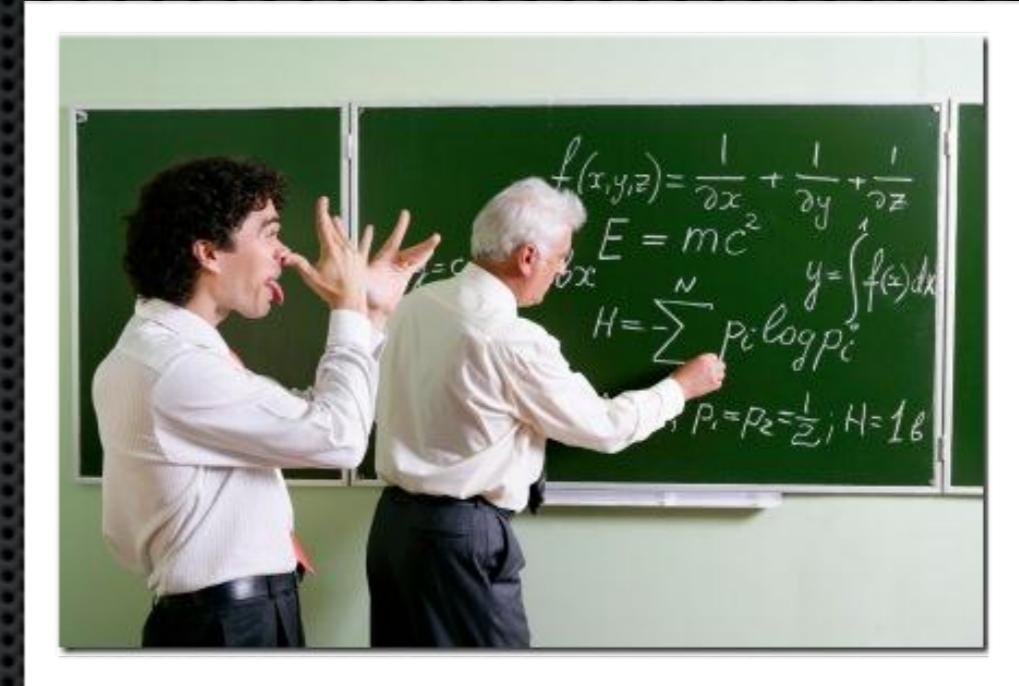
*by Scott Bain - Emergent Design*





# Why Mock?

- Because you are immature!
- Isolate your SUT -> Software Under Test
- To build against interfaces & contracts
- Building against missing integration pieces
- To control data and expectations
- Mock components whose behavior is undesirable or hard to control



*A mock is essentially the interface without any real implementation*

# Why Mock?

- How do you test when helper components that are not built yet?
- How do you do controlled exceptions?
- How do you test & control external API calls?
- How do you control results from ColdFusion tags or functions?
- How do you control network connections? Do you pull the network plug?



# Why Mock?

- How do you test the following?

```
<cfdirectory action="list" directory="#arguments.path#" name="qResults">

<cfhttp url="#arguments.urlPath#" results="qResults">

<cfmail to="#to#" from="#from#" subject="#subject#">#content#</cfmail>

<cfquery />

function init(){
    var helper = new Helper();
}

private function getData(){ return data; }
```

- Some code is untestable or we would need some serious world of hurt to test

# Benefits

- Build test friendly code (refactor) and use a DI engine (WireBox of course!)
- Leverage utility components that can be easily mocked
- Refactor to remove complexities and isolate dependencies
- Smaller and more focused methods
- Improve code reuse
- Our tests can say a lot about our code



# When to Mock?

You're  
*mocking*  
me, aren't  
you?



- . When it's appropriate...
- . When using TDD (*Test Driven Development*)
  - . When you're early in the development cycle
  - . When you're agile and you need results from functionality not built yet

# What are we mocking?

- Entire Components
- Methods
- Properties
- Data
- Exceptions
- Curt??



# Refactor Example

## Original

```
<cfdirectory action="list" directory="/myapp/path" name="qResults">
```

## Refactored

```
<cffunction name="getFiles" output="false" returnType="query">
  <cfargument name="path">

  <cfset var qResults = "">
  <cfdirectory action="list" directory="#arguments.path#" name="qResults">

  ... Process Here ...
  <cfreturn qResults>
</cffunction>
```

# Refactor Example

Original

```
<cffeed action="read" source="http..." query="results">
```

Refactored

```
<cffunction name="getFeeds" output="false" returnType="struct">
  <cfargument name="feedURL">
  <cfargument name="timeout">

  <cfset var results = {}>

  <cffeed action="Read" source="#arguments.feedURL#" query="results"
    timeout="#arguments.timeout#"

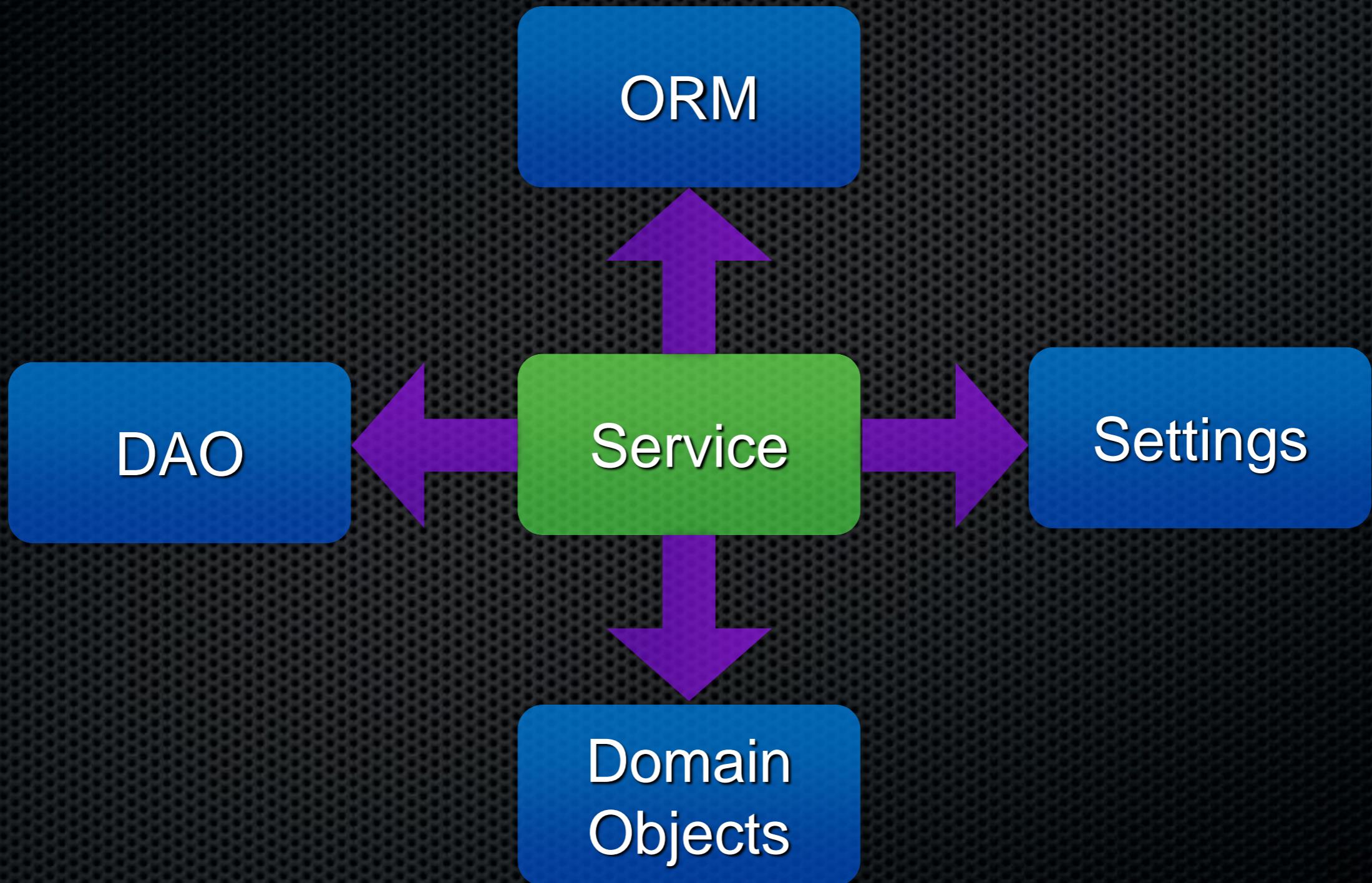
  <cfreturn results>
</cffunction>
```

# Refactoring Thoughts

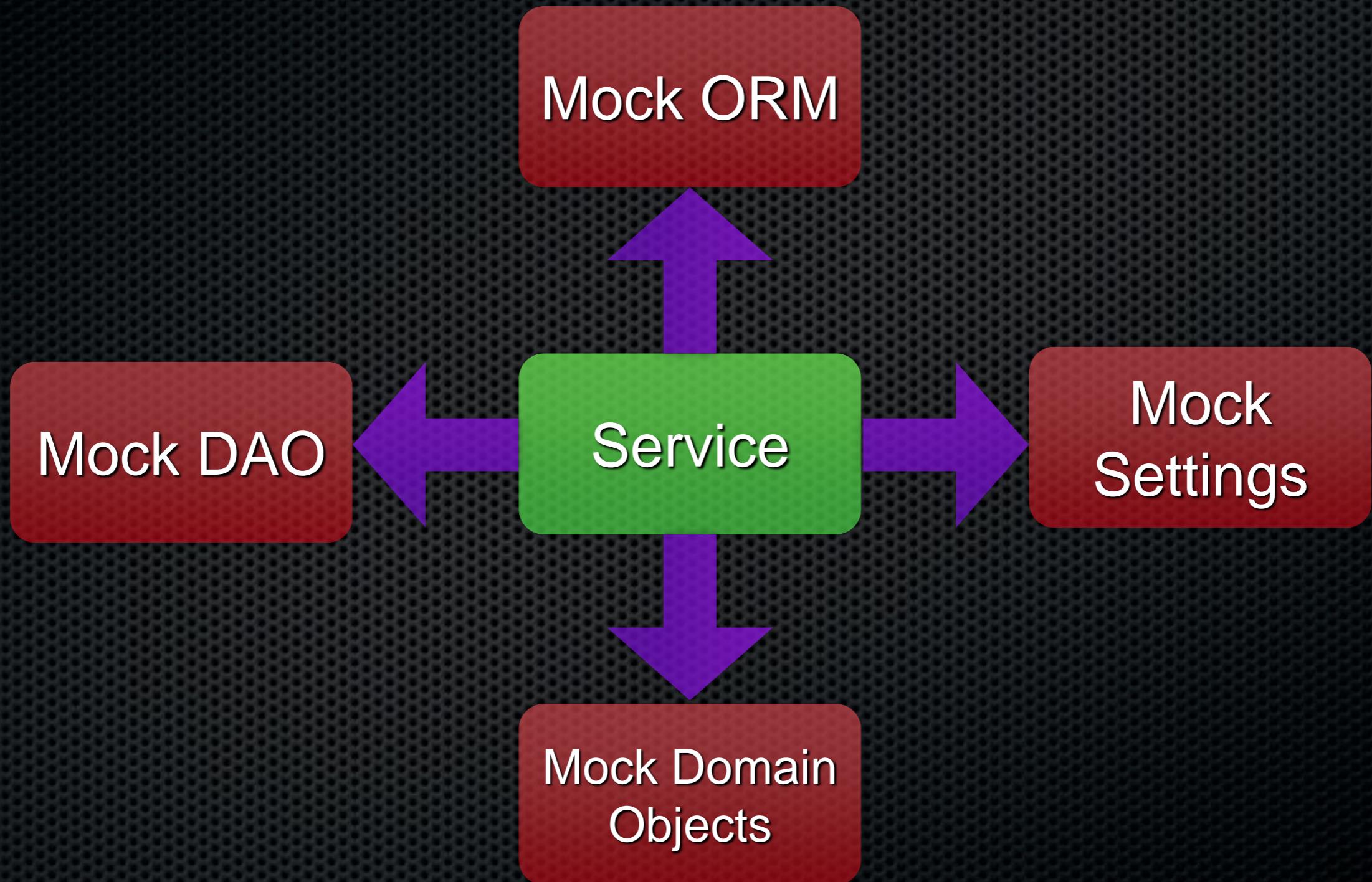


- First thoughts
  - Dumb!
  - I'll end up with lots of small utility methods
  - More work?
- Mature thoughts
  - Cool!
  - I'll have more granular and reusable methods that can be mocked easily
  - Makes my code cleaner
  - Logical code separation

# Typical Example



# Typical Example





# Mocking Frameworks

- **MockBox** by ColdBox
- MightyMock by MXUnit
- ColdMock
- Showcase MockBox as well... We built it

# Key Features

- Mock Objects with or without implementations
- Mock methods & properties in any scope
- Create Stub Objects -> Non-existent objects
- Mock exceptions
- Mock arguments to results
- Logging & Debugging
- Verification methods
- State Machine Results



# Setting up MockBox

## ColdBox Embedded

```
mockBox = createObject("component","coldbox.system.testing.MockBox").init();
```

## ColdBox Base Tests = Easier Integration

```
mockBox = getMockBox();
```

## Standalone

```
mockBox = createObject("component","mockBox.system.testing.MockBox").init();
```

# Creation Methods

- *CreateMock()*
- *CreateEmptyMock()*
- *PrepareMock()*
- *CreateStub()*

Creates & Decorates  
Objects Dynamically!

```
user = mockBox.createMock("model.User");  
  
dao = mockBox.createEmptyMock("model.UserDAO");  
  
mockBox.prepareMock( service );  
  
nonExistentService = mockBox.createStub();
```

# \$()

- Shortcut for mockMethod
- Arguments
- method
- returns
- preserveReturnType
- throwException
- throwType
- throwDetail
- throwMessage
- callLogging

```
// Cascaded mocks
mockUser.$("isFound",true).$("isDirty",true);

// Mock Exception
mockUser.
$(method="save",
throwsException=true,
throwType="IllegalStateException",
throwMessage="Invalid User Data");

// Mock Return Objects
mockRole = mockBox.createMock("Role");
service.$(method="getRole",returns=mockRole);
```

\$()

## Setup

```
mockUser      = mockBox.createEmptyMock("model.User").init();
userService = mockBox.createMock("model.UserService").init();

userService.$("get", mockUser);
```

## Mock methods

```
//Technique 1
user.$("getName", "Luis Majano");

//Technique 2
user.$("getName").$results("Luis Majano", "Curt Gratz", "Diego Maradona");
```

# \$args()

- Argument directed results
- MUST be chained via **\$results()**
- You can use:
  - Named parameters
  - Positional parameters
  - Argument Collection - CF Upper Cases Arguments

```
// Call to Mock
if( dao.getSetting("userAudit") ){
    startAudit( dao.getSetting("auditTables") );
};

// Mocking Calls
dao.$("getSetting").$args("userAudit").$results(true);
dao.$("getSetting").$args("auditTables").$results("user,order,product");
```

# \$args()

## Named Parameters

```
saveUser(fname="luis", lname="majano");
```

## Positional Parameters

```
saveUser("luis", "majano");
```

## Argument Collection

```
data = {  
  fname = "luis", lname = "majano"  
};  
saveUser(argumentCollection=data);
```

# \$results()

- State machine your results
- Repetition sequence
- **\$results(1,2,3) + Called 5 Times = 1,2,3,1,2**

```
// Using Single result set
dao.$("getSetting").$args("userAudit").$results(true);

// Using State Machine
user.$("getVisitCount").$results(5,6,700);
```

# \$property()

- Mock any property on any scope
- Great for settings, aggregation or composition mocking

```
// Mock a setting on the variables scope
service.$property("cacheActive","variables",true);

// Mock a file utility object
mockUtil = mockbox.createEmptyMock("util.FileUtils");
service.$property("fileUtil","variables", mockUtil);

// Mock in the variables.instance scope path
service.$property("isDirty","instance",true);
```



# Code - Discussions



# Resources

- Unit Testing
  - [www.mxunit.org](http://www.mxunit.org)
- Mocking
  - [www.mxunit.org](http://www.mxunit.org)
  - [wiki.coldbox.org/wiki/MockBox.cfm](http://wiki.coldbox.org/wiki/MockBox.cfm)
- ColdBox Resources
  - [www.coldbox.org](http://www.coldbox.org)
  - [wiki.coldbox.org](http://wiki.coldbox.org)
  - [groups.google.com/group/coldbox](http://groups.google.com/group/coldbox)
- Professional Support & Training
  - [www.ortussolutions.com](http://www.ortussolutions.com)



Curt Gratz &  
Computer Know How  
[gratzc@compknowhow.com](mailto:gratzc@compknowhow.com)



Blog: <http://www.coldbox.org/blog>  
Twitter: gratzc  
G-Talk: [gratzc@compknowhow.com](mailto:gratzc@compknowhow.com)

# Q & A



Don't forget the evals

Thanks!