ColdBox

Ortus Solutions
A Software Revolution

Just Mock It!

Leveraging Mock Objects

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, CodexWiki, or anything Box!
- Documentation Freak!

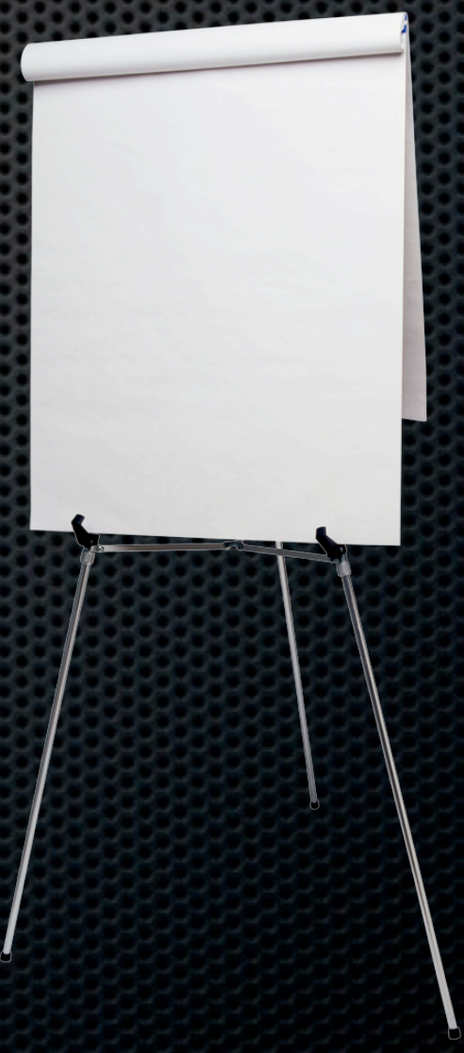
Ortus Solutions
A Software Revolution

Professional Open Source

- **ColdBox Platform** is POSS
- Professional Training Courses
- Books
- Support & Mentoring Plans
- Architecture & Design Sessions
- Code Reviews & Sanity Checks
- We can even brew coffee!

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



Unit Testing

*"unit testing is a software **verification** and **validation** method in which a programmer tests if **individual** units of source code are fit for use. A **unit** is the smallest testable part of an application"*
- wikipedia

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

- Use MXUnit - www.mxunit.org
- Unit Test CFC inherits from **mxunit.framework.TestCase**
- 1-1 Relationship between SUT CFC and Unit Test CFC
  - Calculator.cfc -> CalculatorTest.cfc
- 1-1 Relationship between SUT methods and Unit Test Methods
  - add() -> testAdd()
- Assert towards expectations of results or internal states
- Can test private methods via **makePublic()**

Unit Testing Basics

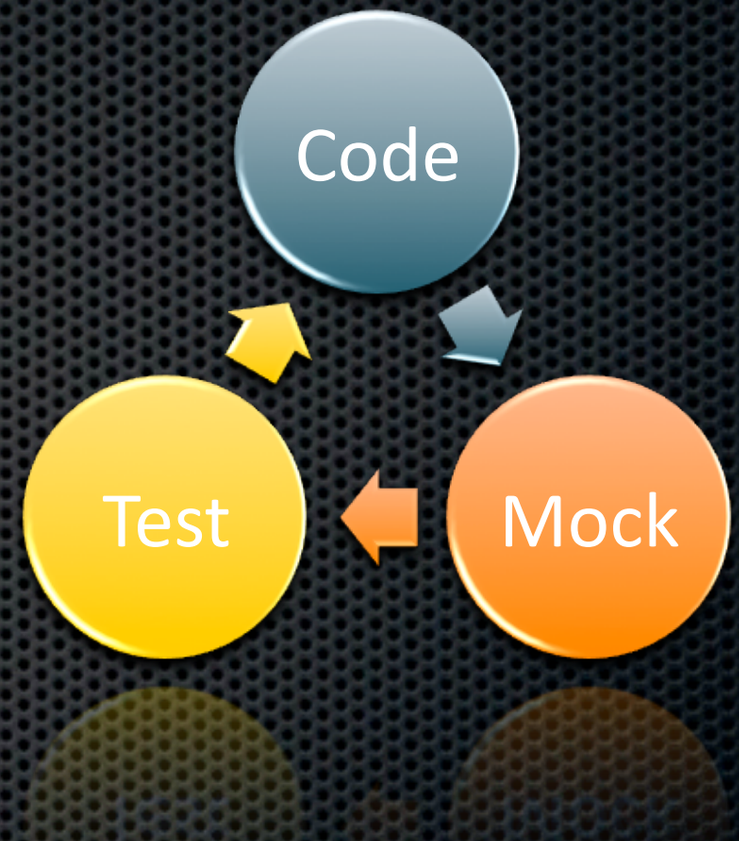
SUT

```
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 );
  }

}
```

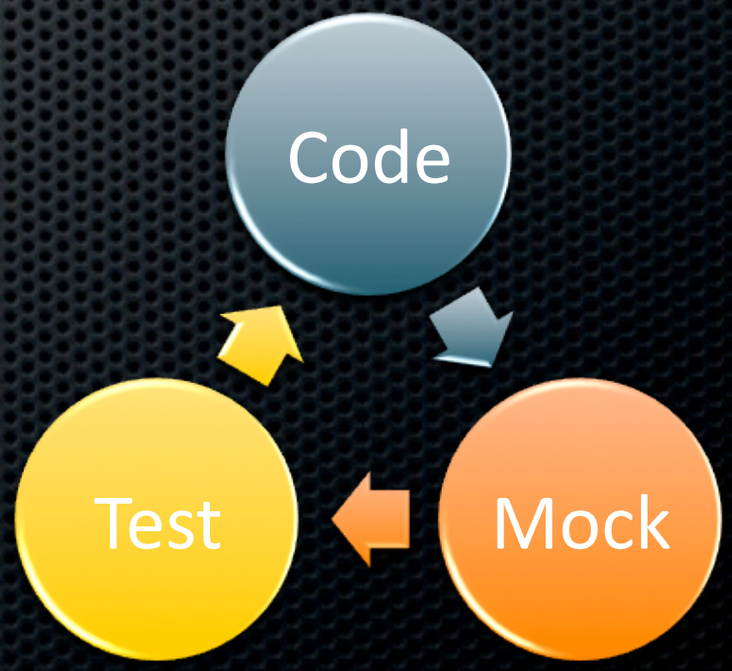
TDD Process



- Test Driven Development
- Can be a new development paradigm for some
- Work from the IDE
  - Write software units
  - Confirm expectations and behavior via unit testing and mocking
  - Continue writing your software units
  - Rinse & Repeat

Important Tests

- Unit Testing
  - Test behavior of **individual** objets
- Integration Testing
  - Included with ColdBox
  - Test entire application headlessly
  - Test entire controller layer top-down
- UI verification testing
  - Verification via HTML/Visual elements
  - Selenium is great!



Testing ToolBox

- MXUnit
- ColdFusion Builder OR CFEclipse
- A mocking framework
- ANT
- Integration Testing
  - ColdBox
- Selenium
- JMeter or Webstress Tool

What is Mocking?

is that when you hit people in the face?

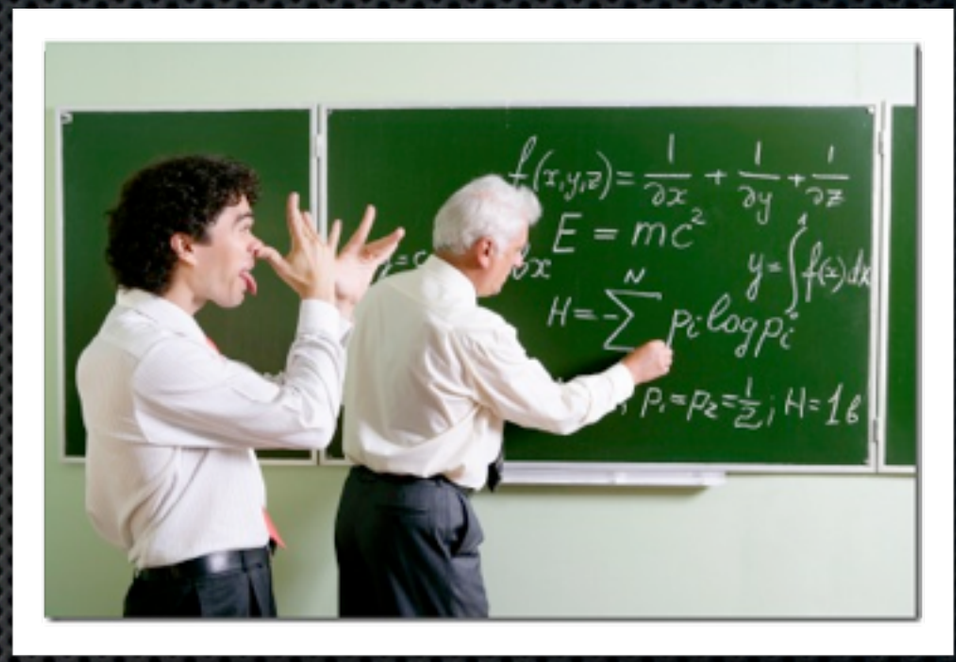
Mocking

“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?

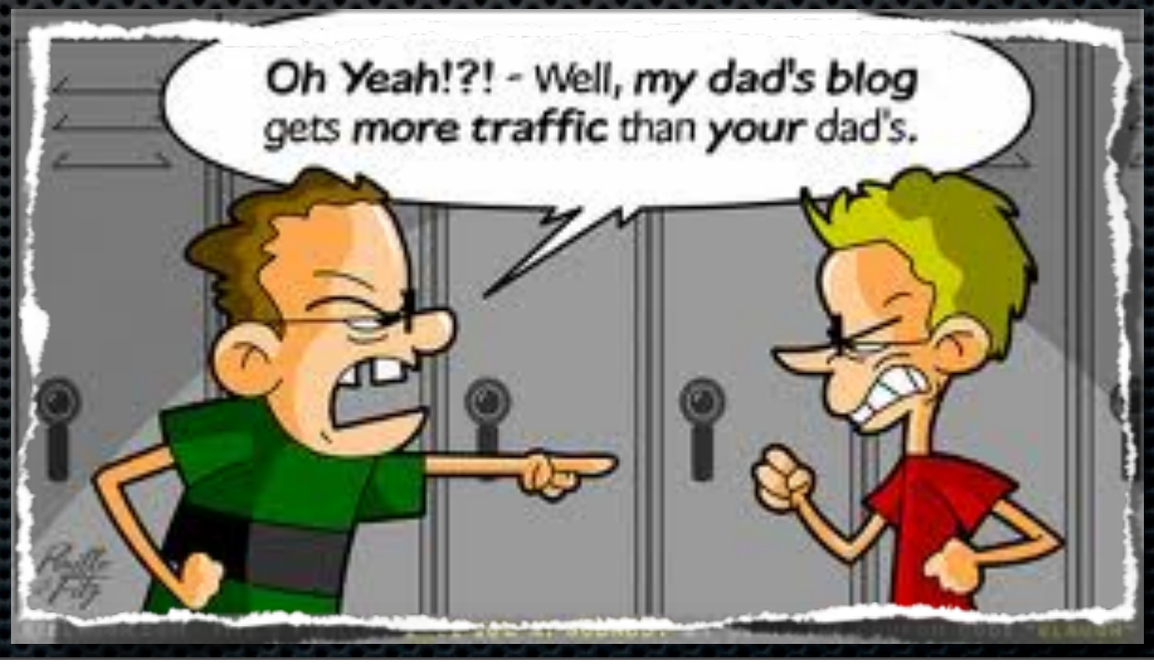


- Because you are immature!
- Isolate your SUT -> Software Under Test
- To build against interfaces & contracts
- Building against missing integration pieces
- To control data and expectations
- Mock components whose behavior is undesirable or hard to control

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

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">
```

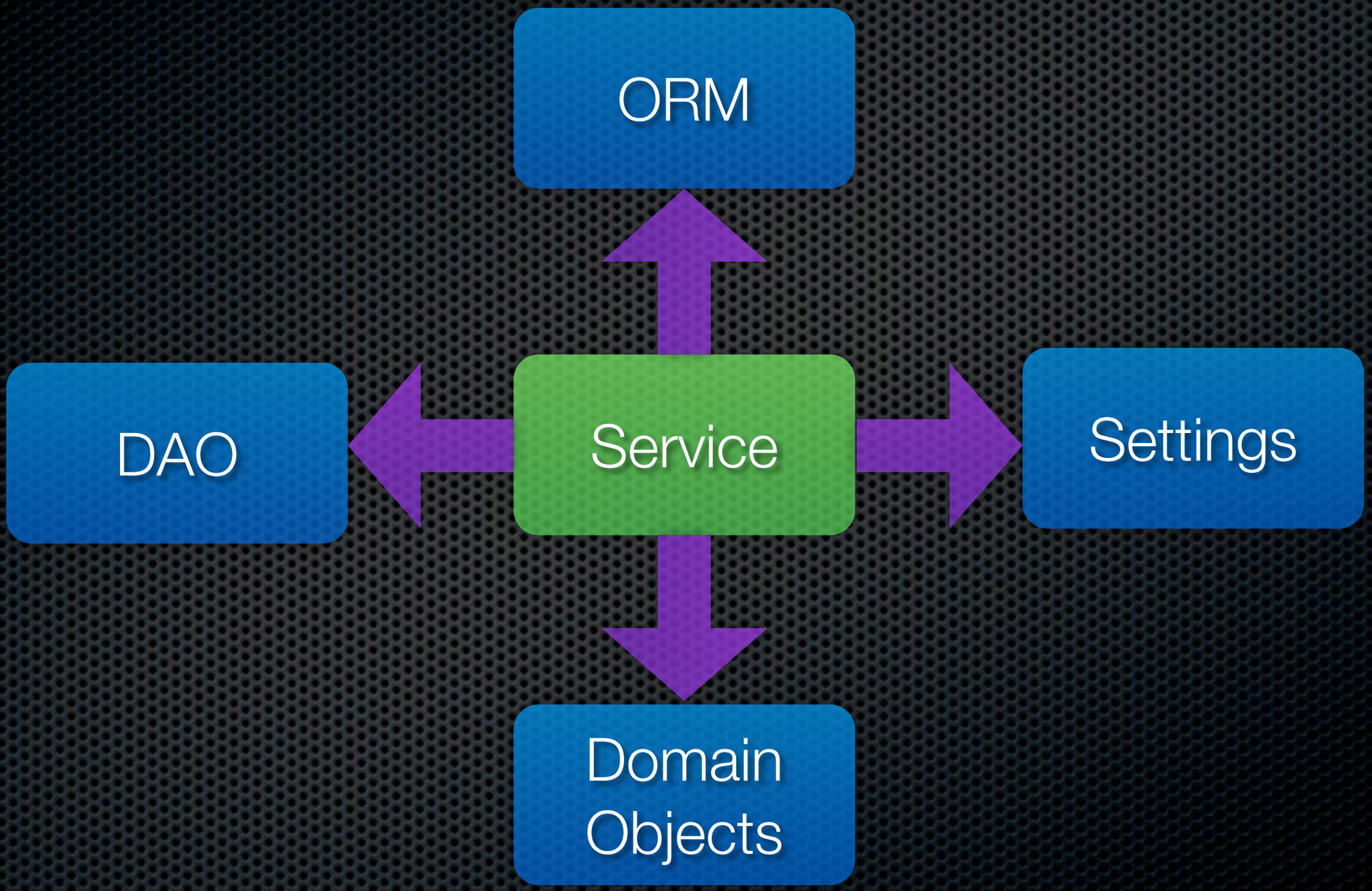
Refactored

```
<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>
```

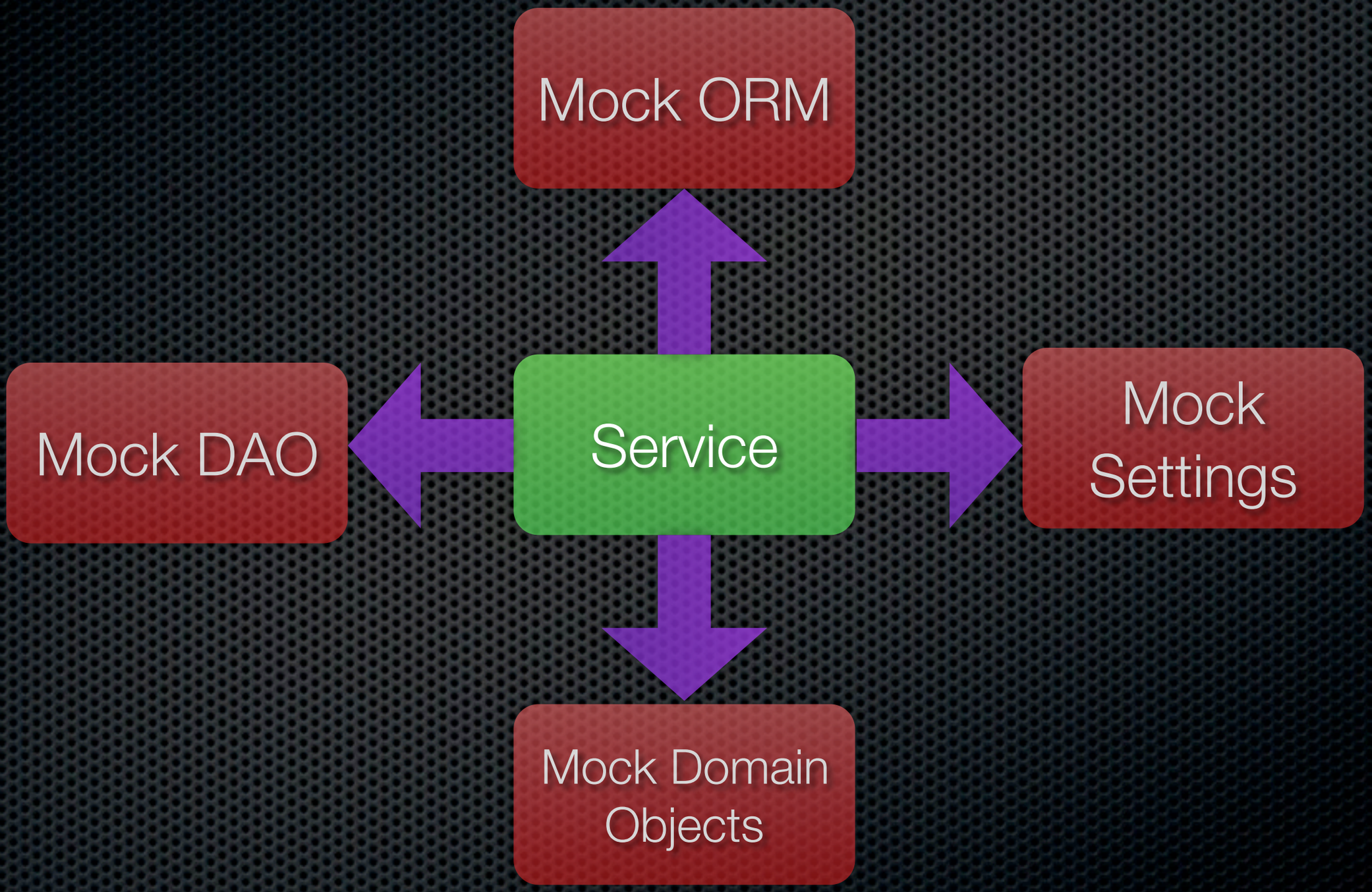
Refactoring Thoughts

- 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();
```

Injected Methods

Method	Description
$()	Mock a method
$property()	Mock a property
$results()	Mock results pattern
$args()	Argument driven results
$count([methodName])	Method call counter
$callLog()	Get call logging stats
$debug()	Get debugging data

Verification Methods

Method	Description
$times(count,[methodName])	Verify X calls
$never([methodName])	Verify never called
$atLeast(min,[methodName])	Verify at least calls
$atMost(max,[methodName])	Verify at most calls
$once([methodName])	Verify called once

**** Verification methods return boolean so they can be asserted***

$()

- 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();

userServie.$("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);
```

Verification Methods

```
function testVerifyCallCount(){
	test.$("displayData",queryNew(''));
	assertTrue( test.$never() );
	assertTrue( test.$never("displayData") );
	assertFalse( test.$times(1,"displayData") );
	assertFalse( test.$once("displayData") );

	test.displayData();
	assertEquals(true,  test.$verifyCallCount(1));
}
function testMockMethodCallCount(){
	test.$("displayData",queryNew(''));
	test.$("getLuis",1);

	assertEquals(0, test.$count("displayData") );
	assertEquals(-1, test.$count("displayData2") );
}
```

If all else fails?

```
<cfdump var="#targetObject.$debug()#">
```

Code - Discussions

Resources

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

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

Q & A

Thanks!