RAD OO

Peter Bell
Railo US

Who Am I?

- **Programmer** - 50-80 projects/yr
- **Entrepreneur** - Profitable/practical
- **Researcher** - DSM Forum, ooPSLA
- **Writer/Presenter** - Code gen, DSM, DDD, Agile/lean, CFML, Groovy, Flex, js
- **Railo US**
  - Fast, open source CFML engine
  - Development consultancy

RAD OO . . .

- Patterns - quickly build OO apps
- Proven (> 100 projects)
- Collection of tools

RAD OO . . .

- IS NOT . . .
  - Standard approach
  - Suitable for all projects
  - Hack job
- IS . . .
  - Fast
  - DRY
  - Maintainable

RAD OO Uses

- **Iterating Business Objects**
- **Base Classes**
- **DSLs for Model and Controller**
- **Optional Classes**
- **Custom Data Types**

Traditional view (initial)

- Complexity happens …

+

Traditional view (initial)

- Complexity happens . . .

```
<cfoutput query="ProductList">
    #Title#<br />
    #Image#<br />
    #SKU#<br />
    #Price#<br /> <br />
</cfoutput>
```

+

Traditional view (later)

```
<strong>#Title#</strong><br />
<cfif len(image)><img src="/ibo/sample/images/#Image#"
width="80"><br />
<cfelse><img src="/ibo/sample/images/no-image.gif"
width="80"><br />
</cfif>
<cfif Len(SKU)>#SKU#<br /></cfif>
<cfif SalePrice><strong>On sale - only
$#Numberformat(SalePrice, "9.99")#<br /></strong>
<cfelse>Price: $#Numberformat(Price, "9.99")#<br />
</cfif>
```

Traditional view (not DRY)

- Product list (multiple?)
- Product detail (multiple?)
- Admin product list
- Cart display
- Order display
- ….

Iterating Business Object

- Encapsulates recordset
- Handles iteration
- Generic get/set - can overload
- Pattern - not implementation
- E.g. Can drop generic get/set

IBO (initial)

```
<cfloop condition="#ProductList.next()#">
     #ProductList.display("Title")#<br />
     #ProductList.display("Image")#<br />
     #ProductList.display("SKU")#<br />
     #ProductList.display("Price")#<br />
     #ProductList.display("SalePrice")#<br /><br />
</cfloop>
```

OR

```
<cfloop condition="#ProductList.next()#">
     #ProductList.displayTitle()#<br />
     #ProductList.displayImage()#<br />
       . . .
</cfloop>
```

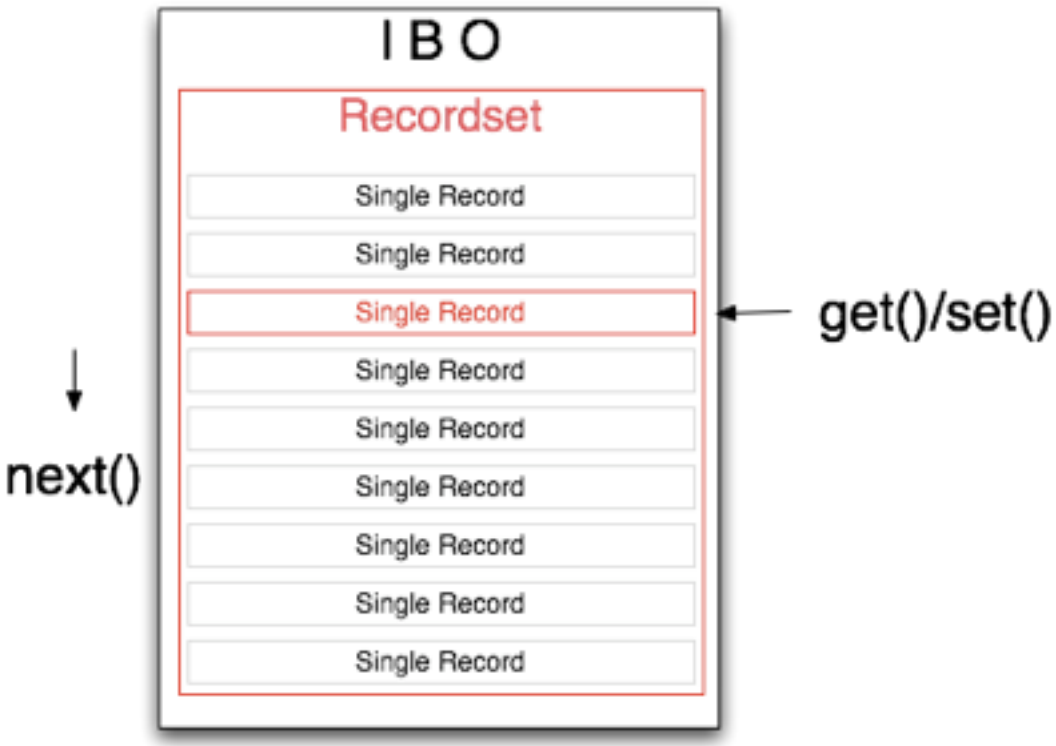
IBO (over time)

```
<cfloop condition="#ProductList.next()#">
     #ProductList.display("Title")#<br />
     #ProductList.display("Image")#<br />
     #ProductList.display("SKU")#<br />
     #ProductList.display("Price")#<br />
     #ProductList.display("SalePrice")#<br /><br />
</cfloop>
```

OR

```
<cfloop condition="#ProductList.next()#">
     #ProductList.displayTitle()#<br />
     #ProductList.displayImage()#<br />
       . . .
</cfloop>
```

The Concept


IBO
Recordset
Single Record
Single Record
Single Record
Single Record
Single Record
Single Record
Single Record
Single Record
Single Record
get()/set()
next()

Using the IBO

CREATE:

```
<cfscript>
	var ProductList = beanFactory.getbean("Product");
	ProductList.load(GetProducts);
</cfscript>
```

ITERATE:

```
<cfset ProductList.reset()>
<cfloop condition="#ProductList.next()#">
	#ProductList.get("Title")#<br />
</cfloop>
```

PERFORMANCE:

- Single bean - quick to create

Base Classes

- Most code similar
- Refactor to parameterized calls
- Samples: BaseService, BaseController, BaseDAO

Base Classes

- Most code similar
- Refactor to parameterized calls
- Samples: BaseService, BaseController, BaseDAO

```
<cffunction name="getProspectList" returntype="any">
        <cfargument name="SalesRep" required="true" type="numeric">
        <cfreturn getbyFilter(Filter="Prospect = 1 AND AssociatedRep =
#SalesRep#" , PropertyNameList="FirstName,LastName" , Order="LastName,FirstName")>
</cffunction>
```

Domain Specific Languages

- Coding sucks
  - Slow
  - Repetitive
  - Tests
  - Not business user readable
  - Code == Fail

“Ideal” DSLs

```
Product  extends: BaseObject tableName: tbl_Product
Identity: ProductID

Properties:
  Title title required
  Price money optional default:0
  Description WYSIWYG optional

ClassMethods:
  AdminList: Title,Price OrderBy Title
  DefaultAdd: Title,Price,Description
  QuickAdd: Title,Price multiple:5
  DefaultEdit: ID, Title,Price,Description

Relationship
  has-many Category associated optional
```

First Cut

```
<?xml version="1.0" encoding="UTF-8"?>
<businessObject>
    <title>Product</title>
    <extends>BaseObject</extends>
    <properties>
        <field title="Title" name="Title" columnName="Title" dataType="string"
sqlDataType="varchar(150)" />
    </properties>

    <classMethods>
        <getByProperty name="AdminEditForm" propertyName="ProductID"
propertyNameList="Title,CategoryList,DisplayOrder" />
    </classMethods>

</businessObject>
```

First Cut

```
<?xml version="1.0" encoding="UTF-8"?>
<businessObject>
    <title>Product</title>
    <extends>BaseObject</extends>
    <properties>
        <field title="Title" name="Title" columnName="Title" dataType="string"
sqlDataType="varchar(150)" />
    </properties>

    <classMethods>
        <getByProperty name="AdminEditForm" propertyName="ProductID"
propertyNameList="Title,CategoryList,DisplayOrder" />
    </classMethods>

</businessObject>
```

+

- Automatically process
- Enforce constraints
- Generate docs
- Less tests required

First Cut Implementation

- Most code similar
- Refactor to parameterized calls
- *Samples: BaseService, BaseDAO, BaseController*
- Your language *will* vary …

First Cut

First Cut

```
<cffunction name="getAdminList" returntype="Query">
    <cfset var AdminList = "">
    <cfquery name="AdminList" datscource="#variables.datasource#">
        SELECT title , publicationdate
        FROM Article
        WHERE Live = 1
        ORDER BY PublicationDate
    </cfquery>
    <cfreturn AdminList>
</cffunction>
```

First Cut

```
<cffunction name="getAdminList" returntype="Query">
    <cfset var AdminList = "">
    <cfquery name="AdminList" datscource="#variables.datasource#">
        SELECT title , publicationdate
        FROM Article
        WHERE Live = 1
        ORDER BY PublicationDate
    </cfquery>
    <cfreturn AdminList>
</cffunction>

<cffunction name="getAdminList" returntype="any">
     <cfreturn getbyFilter(Filter="Live=1" ,
PropertyNameList="title,publicationdate" , Order="PublicationDate")>
</cffunction>
```

First Cut

```
<cffunction name="getAdminList" returntype="Query">
    <cfset var AdminList = "">
    <cfquery name="AdminList" datscource="#variables.datasource#">
        SELECT title , publicationdate
        FROM Article
        WHERE Live = 1
        ORDER BY PublicationDate
    </cfquery>
    <cfreturn AdminList>
</cffunction>

<cffunction name="getAdminList" returntype="any">
     <cfreturn getbyFilter(Filter="Live=1" ,
PropertyNameList="title,publicationdate" , Order="PublicationDate")>
</cffunction>
```

Code

First Cut

```
<cffunction name="getAdminList" returntype="Query">
    <cfset var AdminList = "">
    <cfquery name="AdminList" datscource="#variables.datasource#">
        SELECT title , publicationdate
        FROM Article
        WHERE Live = 1
        ORDER BY PublicationDate
    </cfquery>
    <cfreturn AdminList>
</cffunction>

<cffunction name="getAdminList" returntype="any">
     <cfreturn getbyFilter(Filter="Live=1" ,
PropertyNameList="title,publicationdate" , Order="PublicationDate")>
</cffunction>


<classMethods>
    <getByFilter name="getAdminList" filter="Live=1"
propertyNameList="title,publicationdate" order="PublicationDate" />
</classMethods>
```

Code

Optional Classes

- Only create CFC's with code
- All others instantiate base classes

Code

Lightwire Config

```
BusinessObjectList = ApplicationConfig.get("ObjectList");
For (i = 1; i lte listlen(BusinessObjectList); i = i + 1)
	{
	// Get current object name
	BusinessObjectName = ListGetAt(BusinessObjectList, i);
		// Configure Business Object Service Classes
	If (FileExistsinApplication("com.model.#BusinessObjectName#Service"))
	{addSingleton("lb.com.model.#BusinessObjectName#Service");}
	Else
	{
		If (FileExistsinLibrary("com.model.#BusinessObjectName#Service"))

	{addSingleton("lightbase3.library.com.model.#BusinessObjectName#Service");}
		Else
		{addSingleton("lightbase3.framework.com.base.BaseService",
"#BusinessObjectName#Service");};
	};

addConstructorDependency("#BusinessObjectName#Service","#BusinessObjectName#Metadata",
"Metadata");
	addMixinDependency("#BusinessObjectName#Service","#BusinessObjectName#DAO");
	addMixinDependency("#BusinessObjectName#Service","DBProvisioner");
};
```

Code

Custom Data Types

- Scaffolding sucks
- Add metadata, generate *real* application
- Django, Grails …
- Key element: custom data types
- Scaffolding with sense

Custom Data Types

- Displaying form fields
- Processing form fields
- Displaying formatted values

Custom Data Types

- .display()
- .field()
- .process()

Code

Conclusions

- Proven in over 100 projects
- Extremely productive
- Removes unnecessary flexibility
- Your milage may vary

Questions?

- Blog: www.pbell.com
- Email: peter@getrailo.com
- Skype: peterfbell
- Awesome website: http://getrailo.com
- Check out:
  - RIA Unleashed - November 13th
  - cf in nc October 17/18th