



Introduction to Reactor for ColdFusion

Teddy R. Payne
teddyrpayne@gmail.com

June 7, 2006

What to expect

- What is Reactor for ColdFusion?
- What software is supported?
- Installation
- Creating a Reactor project
- Configuring Reactor
- Record Factory
- Gateway Factory
- Is there more to Reactor?
- Questions and Answers
- How to get more involved

What is Reactor for ColdFusion?

“Inline Dynamic Database Abstraction” - Doug Hughes

- What does this mean to the non-enlightened?
 - Reactor
 - is a code generator (CFCs) to help manipulate your database data
 - creates reusable components
 - removes the dependency of database specific queries
 - is independent of ColdFusion frameworks
 - allows for customizing and extending the behavior of its created objects

ColdFusion Support

- Reactor for ColdFusion accesses database metadata using ColdFusion Components (CFCs)
- Thus, only the product versions that support CFCs can utilize Reactor for ColdFusion
 - CFMX 7
 - CFMX 6.1

Database Support

- Microsoft SQL Server
 - 2005
 - 2000
- MySQL
 - 5.x
 - 4.x¹
- Oracle²
 - 9
 - 8
- PostgreSQL
 - Planned

¹ MySQL 4.x does not support views

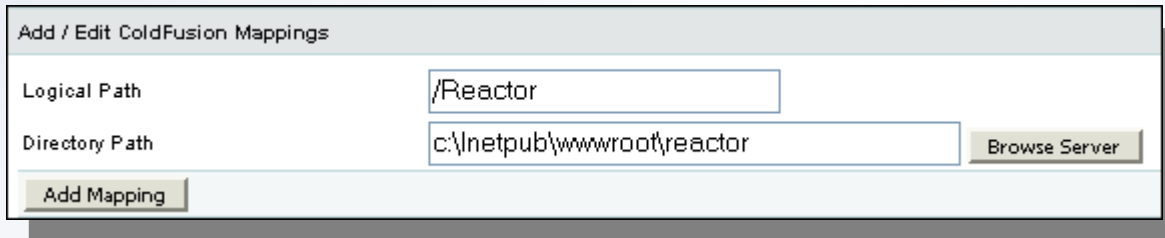
² Oracle Reactor support was not written by Doug Hughes. Additional future development is planned. Current status is that there is an issue with case sensitivity database objects.

Presentation Environment

- IIS 6.0
- ColdFusion MX 7
- Microsoft SQL Server 2000
- Eclipse 3.1
 - CF Eclipse 1.2.0
 - Adobe RDS
 - Oxygen XML Editor 7.1.0
- Tortoise SVN 1.3.3
- Reactor Build 269 (06/04/2006)

Installing Reactor

- Download the source code via SVN
 - <http://svn.reactorframework.com/reactor/trunk/>
- Copy the source code to your web root
 - ex. c:\inetpub\wwwroot\reactor
- Create a directory mapping from your ColdFusion Administrator

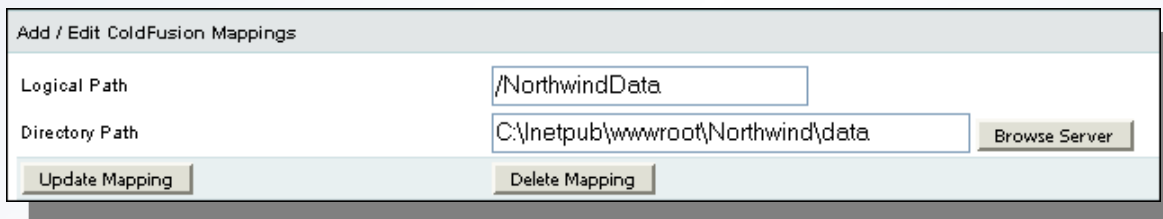


The screenshot shows the 'Add / Edit ColdFusion Mappings' dialog box. It has two input fields: 'Logical Path' with the value '/Reactor' and 'Directory Path' with the value 'c:\inetpub\wwwroot\reactor'. There is a 'Browse Server' button next to the 'Directory Path' field. At the bottom left, there is an 'Add Mapping' button.

Add / Edit ColdFusion Mappings	
Logical Path	<input type="text" value="/Reactor"/>
Directory Path	<input type="text" value="c:\inetpub\wwwroot\reactor"/> <input type="button" value="Browse Server"/>
<input type="button" value="Add Mapping"/>	

Creating your Reactor project

- Create your project folder for your Reactor project
 - ex. c:\inetpub\wwwroot\Northwind
- Reactor needs a folder to store dynamic data objects for your project, so create an arbitrary folder for data
 - ex. c:\inetpub\wwwroot\Northwind\data
- The dynamic Reactor objects can be referenced through relative paths, but it is recommended to create a mapping for your data folder again from your ColdFusion administrator



The screenshot shows the 'Add / Edit ColdFusion Mappings' dialog box. It has two input fields: 'Logical Path' with the value '/NorthwindData' and 'Directory Path' with the value 'C:\inetpub\wwwroot\Northwind\data'. To the right of the 'Directory Path' field is a 'Browse Server' button. At the bottom of the dialog are two buttons: 'Update Mapping' and 'Delete Mapping'.

- Create a reactor.xml file in your project folder
 - ex. c:\inetpub\wwwroot\Northwind\reactor.xml

Configuring your Reactor

- In your reactor.xml, you will need the following initial XML to get your Reactor project configured

```
<reactor>
  <config>
    <project value="Northwind" />
    <dsn value="Northwind" />
    <type value="mssql" />
    <mapping value="/NorthwindData" />
    <mode value="development" />
  </config>
</reactor>
```

- All child elements of the <config> node are required

Configuring your Reactor (cont.)

- `<project value="Northwind" />`
 - The name of Reactor project which is similar to `<cfapplication />`
- `<dsn value="Northwind" />`
 - The data source name of the database
- `<type value="mssql" />`
 - The database type of the schema
 - "mssql"
 - MSSQL 2000,2005
 - "mysql4"
 - MySQL 4
 - "mysql"
 - MySQL 5 +
- `<mapping value="/NorthwindData" />`
 - The mapping to where the dynamic objects will be stored

Configuring Reactor (cont.)

- `<mode value="development" />`
 - This dictates the the behavior of how Reactor reads the schema and instantiates objects
 - “always”
 - Reactor will recreate all objects and reread all database objects every time Reactor is instantiated
 - Load : Heavy
 - “development”
 - Reactor checks to see if the database schema has changed, if the schema has changed, reload only the necessary objects
 - Load : Medium
 - “production”
 - After the initial load, Reactor will instantiate the object and return without checking for changes to the database
 - Load : Light

Instantiating the Reactor Factory

```
<cfset Reactor = CreateObject("Component",  
    "reactor.reactorFactory").init(expandPath("reactor.xml"))>
```

- Once the Reactor factory has been created, the following methods are now exposed
 - `Reactor.CreateRecord()`
 - `Reactor.CreateGateway()`
 - `Reactor.CreateMetaData()`
 - `Reactor.CreateValidator()`
 - `Reactor.CreateDictionary()`
 - `Reactor.CreateTO()`
 - `Reactor.CreateIterator()`
 - `Reactor.CreateDAO()`

Reactor.CreateRecord()

Reactor.CreateRecord()

- The record factory provides the ability to work with one record at a time
- At creation, methods to “get” and “set” the value of each representative table column are stored in the record factory
- Database persistence is simplified into the data access layer using intuitive methods
 - load()
 - save()
 - delete()

Reading a Record

Northwind.dbo.Region

- RegionID int PK
- RegionDescription nchar(50)

```
<cfset Reactor =  
    CreateObject("Component","reactor.reactorFactory").init(expandPath("reactor.xml"))>
```

```
<cfset Region = Reactor.createRecord("Region")>
```

```
<cfset Region.setRegionID(1)>
```

```
<cfset Region.load()>
```

```
<cfoutput>#Region.getRegionID()# - #Region.getRegionDescription()#<br /></cfoutput>
```


Reading a Record Explained

```
<cfset Reactor =  
    CreateObject("Component","reactor.reactorFactory").init(expandPath("reactor.xml"))>
```

- Create the Reactor factory using the reactor.xml configuration file

```
<cfset Region = Reactor.createRecord("Region")>
```

- Create the Record factory for the Region table

```
<cfset Region.setRegionID(1)>
```

- Set the value of the RegionID Primary Key (PK) for the record you are interested in reading

```
<cfset Region.load()>
```

- Load the data from the table with the primary key that was set

```
<cfoutput>#Region.getRegionID()# - #Region.getRegionDescription()#<br /></cfoutput>
```

- Get the values of the data that was loaded into the Record object

Reading a Record Explained (cont.)

```
<cfset Region = Reactor.createRecord("Region")>
```

```
<cfset Region.setRegionID(1)>
```

```
<cfset Region.load()>
```

- Is there a shorter way to achieve this?

```
<cfset Region = Reactor.createRecord("Region").load(RegionID=1)>
```

Creating a Record

Northwind.dbo.Region

- RegionID int PK Identity
- RegionDescription nchar(50)

```
<cfset Reactor =  
    CreateObject("Component","reactor.reactorFactory").init(expandPath("reactor.xml"))>
```

```
<cfset Region = Reactor.createRecord("Region")>
```

```
<cfset Region.setRegionID(5)>
```

```
<cfset Region.setRegionDescription("Northeastern")>
```

```
<cfset Region.save()>
```

Creating a Record Explained

```
<cfset Reactor =  
  CreateObject("Component","reactor.reactorFactory").init(expandPath("reactor.xml"))>
```

- Create the Reactor factory using the reactor.xml configuration file

```
<cfset Region = Reactor.createRecord("Region")>
```

- Create the Record factory for the Region table

```
<cfset Region.setRegionID(5)>
```

- Set the value of the RegionID PK for the record you are interested in creating

```
<cfset Region.setRegionDescription("Northeastern")>
```

- Set the value of the RegionDescription for the new record

```
<cfset Region.save()>
```

- Commit the data to the database

Creating a Record Explained (cont.)

Isn't that a little verbose using <cfset />?

<cfscript>

```
Reactor =  
    CreateObject("Component","reactor.reactorFactory").init(expandPath("reactor.xml"));
```

```
Region = Reactor.createRecord("Region");
```

```
Region.setRegionID(5);
```

```
Region.setRegionDescription("Northeastern");
```

```
Region.save();
```

</cfscript>

Updating a Record

Northwind.dbo.Region

- RegionID int PK Identity
- RegionDescription nchar(50)

<cfscript>

```
Reactor =  
    CreateObject("Component","reactor.reactorFactory").init(expandPath("reactor.xml"));
```

```
Region = Reactor.createRecord("Region").load(RegionID=5);
```

```
Region.setRegionDescription("Southeastern");
```

```
Region.save();
```

</cfscript>

Updating a Record Explained

Reactor =

```
CreateObject("Component","reactor.reactorFactory").init(expandPath("reactor.xml"));
```

- Create the Reactor factory using the reactor.xml configuration file

Region = Reactor.createRecord("Region").load(RegionID=5);

- Using the shortcut, load the data associated to RegionID PK with the value of 5

Region.setRegionDescription("Southeastern");

- Set the value of the RegionDescription for the record to be updated

Region.save();

- Commit the data to the database

Deleting a Record

Northwind.dbo.Region

- RegionID int PK Identity
- RegionDescription nchar(50)

<cfscript>

```
Reactor = CreateObject("Component",  
    "reactor.reactorFactory").init(expandPath("reactor.xml"));
```

```
Region = Reactor.createRecord("Region").load(RegionID=5);
```

```
Region.delete();
```

</cfscript>

Deleting a Record Explained

Reactor =

```
CreateObject("Component","reactor.reactorFactory").init(expandPath("reactor.xml"));
```

- Create the Reactor factory using the reactor.xml configuration file

Region = Reactor.createRecord("Region").load(RegionID=5);

- Using the shortcut, load the data associated to RegionID PK with the value of 5

Region.delete();

- Delete the record from the Region table with the RegionID PK of 5

Deleting a Record Explained (cont.)

```
<cfscript>
```

```
Reactor = CreateObject("Component",  
    "reactor.reactorFactory").init(expandPath("reactor.xml"));
```

```
Region = Reactor.createRecord("Region").load(RegionID=5);
```

```
Region.delete();
```

```
</cfscript>
```

- Is there a shorter way to achieve this?

```
<cfscript>
```

```
Reactor = CreateObject("Component",  
    "reactor.reactorFactory").init(expandPath("reactor.xml"));
```

```
Region = Reactor.createRecord("Region").delete(RegionID=5);
```

```
</cfscript>
```

Deleting a Record Explained (cont.)

- Is there a way to detect a deleted record?
- IsDeleted()
 - Provides a logic layer to assist in detecting deleted records
 - No method arguments
 - Returns a Boolean value ("true","false")

<cfscript>

```
Reactor =  
    CreateObject("Component","reactor.reactorFactory").init(expandPath("reactor.xml"));  
Region = Reactor.createRecord("Region").load(RegionID=5);  
Region.delete();  
Deleted = Region.IsDeleted();
```

</cfscript>

Deleted a Record Explained (cont.)

- IsDeleted() Gotcha

```
<cfscript>
```

```
Reactor =  
    CreateObject("Component","reactor.reactorFactory").init(expandPath("reactor.xml"));
```

```
Region = Reactor.createRecord("Region").delete(RegionID=5);
```

```
Deleted = Region.IsDeleted();
```

```
</cfscript>
```

- This will generate an error
- Must have a load() method prior to deletion or the Region object will have been deconstructed

Record Validation

```
<cfscript>
```

```
Reactor = CreateObject("Component", "reactor.reactorFactory").init(expandPath("reactor.xml"));
```

```
Region = Reactor.createRecord("Region").load(RegionID=99);
```

```
Region.Validate();
```

```
if(Region.HasErrors())
```

```
{
```

```
    ErrorCollection = Region._getErrorCollection();
```

```
    Error_Array = ErrorCollection.GetErrors();
```

```
    Error_Translated_Array = ErrorCollection.GetTranslatedErrors();
```

```
    Error_Count = ErrorCollection.Count();
```

```
    RegionDescription_Error =
```

```
        ErrorCollection.HasError("Region.RegionDescription.notProvided");
```

```
    RegionDescription_Translated_Error =
```

```
        ErrorCollection.GetTranslatedError("Region.RegionDescription.notProvided");
```

```
}
```

```
</cfscript>
```

Record Validation Explained

```
Region = Reactor.createRecord("Region").load(RegionID=99);
```

- Create a Dirty Read for purposes of generating validation errors

```
Region.Validate();
```

- Calling this method exposes three methods into the Region record
 - Validated()
 - Boolean method to return if the record has ever been validated
 - HasErrors()
 - Boolean method to return if the record has any errors
 - _getErrorCollection()
 - Returns the Error Collection object to extend the record object

```
if(Region.HasErrors())
```

- Boolean statement to test if the Region record has any validation errors

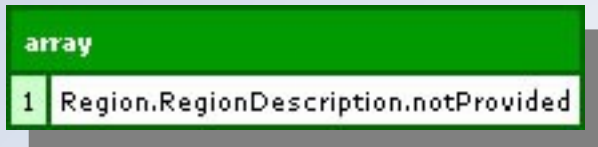
```
ErrorCollection = Region._getErrorCollection();
```

- Reference the error collection to expose the methods that detail the specific errors

Record Validation Explained (cont.)

```
Error_Array = ErrorCollection.GetErrors();
```

- Get the array with all of the validation errors



A screenshot of a Visual Basic array variable. The array is named 'array' and is of type 'array'. It contains one element at index 1 with the value 'Region.RegionDescription.notProvided'.

	array
1	Region.RegionDescription.notProvided

```
Error_Translated_Array = ErrorCollection.GetTranslatedErrors();
```

- Get the array of all of the errors with verbose descriptions



A screenshot of a Visual Basic array variable. The array is named 'array' and is of type 'array'. It contains one element at index 1 with the value 'The RegionDescription field is required but was not provided.'.

	array
1	The RegionDescription field is required but was not provided.

```
Error_Count = ErrorCollection.Count();
```

- Get the integer count of the total number of validation errors found

```
ErrorCollection.HasError("Region.RegionDescription.notProvided");
```

- Boolean method to return if the collection contains a specific error
- Not to be confused with HasErrors, which tests if the record has any validation errors

Record Validation Explained (cont.)

```
RegionDescription_Translated_Error =  
    ErrorCollection.GetTranslatedError("Region.RegionDescription.notProvided");
```

- Returns a string of the verbose description of the validation from the shorthand

The RegionDescription field is required but was not provided.

Dirty Reads

- What happens when you load a record with a primary key that doesn't exist?
- You have an instantiated object that has a value for the primary key, but the subsequent column methods are empty

```
<cfscript>
```

```
    Reactor =  
        CreateObject("Component","reactor.reactorFactory").init(expandPath("reactor.xml"));  
    Region = Reactor.createRecord("Region").load(RegionID=99);
```

```
</cfscript>
```

```
<cfoutput>
```

```
Result: #Region.getRegionID()# - #Region.getRegionDescription()#
```

```
</cfoutput>
```

Result: 99 -

Dirty Reads (cont.)

- IsDirty()
 - Provides a logic layer to assist in detecting dirty reads
 - No method arguments
 - Returns a Boolean value ("YES","NO")

<cfscript>

```
Reactor =  
    CreateObject("Component","reactor.reactorFactory").init(expandPath("reactor.xml"));  
Region = Reactor.createRecord("Region").load(RegionID=99);
```

```
DirtyRead = Region.IsDirty();
```

```
if(not DirtyRead)  
{  
    ...  
}
```

</cfscript>

Reactor.CreateGateway()

Reactor.CreateGateway()

- The gateway factory provides the ability to work with multiple records at a time
- Allows for creating customized query objects
- Join notation

Getting all records from a table

- GetAll()
 - Returns all records for the table passed into the gateway
 - No method arguments
 - Returns query

<cfscript>

```
Reactor =  
    CreateObject("Component","reactor.reactorFactory").init(expandPath("reactor.xml"));  
Shippers = reactor.createGateway("Shippers");  
qryShippers = Shippers.getAll();
```

</cfscript>

query - Top 3 of 3 Rows			
	COMPANYNAME	PHONE	SHIPPERID
1	Speedy Express	(503) 555-9831	1
2	United Package	(503) 555-3199	2
3	Federal Shipping	(503) 555-9931	3

Getting records by the column name

- GetByFields()
 - Returns all records for the table passed into the gateway
 - Optional column search criteria and sorting
 - Returns query

<cfscript>

```
Reactor =  
    CreateObject("Component","reactor.reactorFactory").init(expandPath("reactor.xml"));  
Shippers = reactor.createGateway("Shippers");  
qryShippers = Shippers.getByFields(ShipperID=3,sortByFieldList="ShipperID");
```

</cfscript>

```
qryShippers = Shippers.getByFields(ShipperID=3,sortByFieldList="ShipperID");
```

- Column names do not have to be in order
- sortByFieldList can contain more than one column name and performs a DESC order which is comma delimited
- Performs simple where clause statement based upon exact matching
- Searching by a column is optional if you just want to sort by column

Creating Custom Queries

```
<cfscript>
```

```
Reactor =  
    CreateObject("Component","reactor.reactorFactory").init(expandPath("reactor.xml"));
```

```
Suppliers = reactor.createGateway("Suppliers");
```

```
query = Suppliers.CreateQuery();
```

```
query.returnObjectFields("Suppliers","CompanyName,ContactName");
```

```
where = query.getWhere();
```

```
where.IsEqual("Suppliers","Country","Germany");
```

```
qrySuppliers = Suppliers.getByQuery(query);
```

```
</cfscript>
```

Creating Custom Queries Explained

```
Suppliers = reactor.createGateway("Suppliers");
```

- Create the gateway to the Suppliers table

```
query = Suppliers.CreateQuery();
```

- Instantiate a new query object for the Suppliers gateway

```
query.returnObjectFields("Suppliers","CompanyName,ContactName");
```

- Set the query to return the columns CompanyName and ContactName from Suppliers
- As of Build 269, returnObjectFields takes a list of fields instead of returning all fields

```
where = query.getWhere();
```

- Instantiate a new where object for the query

```
where.IsEqual("Suppliers","Country","Germany");
```

- Insert an IsEqual statement in the where object to set Country = "Germany"

```
qrySuppliers = Suppliers.getByQuery(query);
```

- Execute the query

Creating Custom Queries Explained (cont.)

- What is that equivalent to in SQL?

select

 CompanyName ,

 ContactName

from

 Suppliers

where

 Country = "Germany"

query - Top 3 of 3 Rows

	COMPANYNAME	CONTACTNAME
1	Heli Süßwaren GmbH & Co. KG	Petra Winkler
2	Plutzer Lebensmittelgroßmärkte AG	Martin Bein
3	Nord-Ost-Fisch Handelsgesellschaft mbH	Sven Petersen

Create a Compound Query

<cfscript>

```
Reactor = CreateObject("Component", "reactor.reactorFactory").init(expandPath("reactor.xml"));
```

```
Suppliers = reactor.createGateway("Suppliers");
```

```
query = Suppliers.CreateQuery();
```

```
query.returnObjectFields("Suppliers","CompanyName,ContactName");
```

```
where = query.getWhere();
```

```
where.IsEqual("Suppliers","Country","Germany");
```

```
where.setMode("and");
```

```
where_2 = where.CreateWhere();
```

```
where_2.IsEqual("Suppliers","ContactName","Martin Bein");
```

```
where.addWhere(where_2);
```

```
qrySuppliers = Suppliers.getByQuery(query);
```

</cfscript>

Create a Compound Query Explained

```
where = query.getWhere();
```

- Instantiate a new where object for the query

```
where.IsEqual("Suppliers","Country","Germany");
```

- Insert an IsEqual statement in the where object to set Country = “Germany”

```
where.setMode("and");
```

- Set the compound where clause to use “and”

```
where_2 = where.CreateWhere();
```

- Instantiate an additional clause into the where object

```
where_2.IsEqual("Suppliers","ContactName","Martin Bein");
```

- Insert an IsEqual statement in the second where object to set ContactName = “Martin Bein”

```
where.addWhere(where_2);
```

- Combine the two where objects together to form compound where clause

Create a Compound Query Explained (cont.)

- What is that equivalent to in SQL?

select

 CompanyName ,

 ContactName

from

 Suppliers

where

 Country = "Germany"

and

 ContactName = "Martin Bein"

query - Top 1 of 1 Rows

	COMPANYNAME	CONTACTNAME
1	Plutzer Lebensmittelgroßmärkte AG	Martin Bein

Create a Compound Query Explained (cont.)

- Is there an OO shortcut to compound where clauses?

<cfscript>

```
Reactor =  
    CreateObject("Component","reactor.reactorFactory").init(expandPath("reactor.xml"));  
Suppliers = reactor.createGateway("Suppliers");  
  
query = Suppliers.CreateQuery();  
  
query.returnObjectField("Suppliers","CompanyName");  
query.returnObjectField("Suppliers","ContactName");  
  
where = query.getWhere();  
  
where.IsEqual("Suppliers","Country","Germany").setMode("and").addWhere(where.CreateWhere().IsEqual("Suppliers","ContactName","Martin Bein"));  
  
qrySuppliers = Suppliers.getByQuery(query);
```

</cfscript>

Where Clause Methods

- IsBetween()
- IsEqual(), IsNotEqual()
- IsGt()
- IsGte()
- IsIn(), IsNotIn()
- IsLike()
- IsLt()
- IsLte()
- IsLike(), IsNotLike()
- IsNull(), IsNotNull()
- IsBetweenFields()
- IsEqualField(), IsNotEqualField()
- IsGtField()
- IsGteField()
-
- IsNotLike()
- IsLtField()
- IsLteField()

Is there more to Reactor?

- Joins
- Linked tables
- Transfer objects
- DAO objects
- Metadata objects
- Validator objects
- Iterator objects
- Dictionary objects
- Customizable objects
- Override objects
- ... And much more!

Questions and Answers

How to get more Involved

- Doug Hughes's Blog
 - <http://www.doughughes.net>
- Mailing List
 - Send “subscribe” in the subject of an email to reactor@doughughes.net
- Mailing List Archive
 - <http://www.mail-archive.com/reactor@doughughes.net/>
- Trac
 - <http://trac.reactorframework.com/reactor/trac.cgi>

Special Thanks

- President and Board of ACFUG
 - Thank you for giving me the opportunity to speak
- Doug Hughes
 - For his response to my endless questions
- Ken Adcock
 - For contributing related topic material and ideas
- Precia and John M.
 - Contributing your time proofreading