SYS-CON

THE DARK SIDE OF GATEWAYS PG 28

COLDFUSION Developer's Journal

ColdFusionJournal.com

February 2007 Volume:9 Issue: 2

ColdFusion/.NET Integration

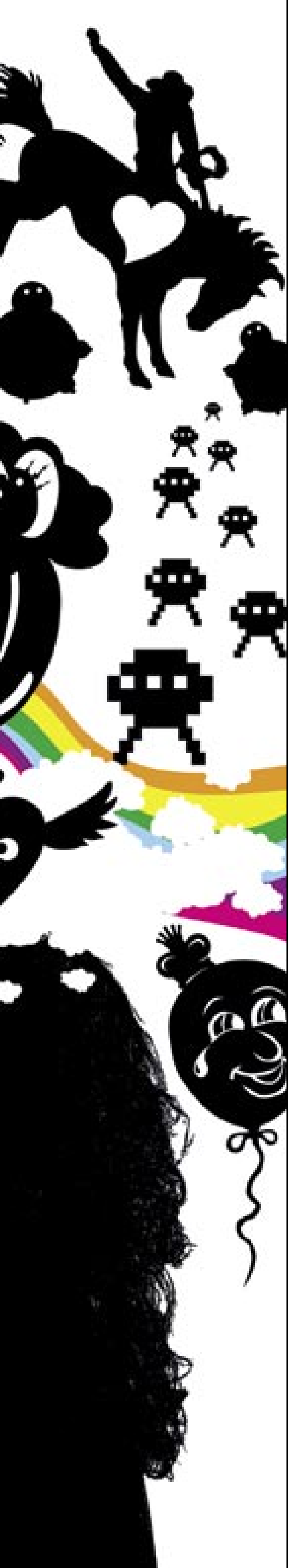
16

Scorpio and Apollo - Coming Soon to a Town Near You! 5

Data Table Gateways 7

A Flex 2 DVD Library Application 30

YUM!
INTERSPECTACULAR



Your Web site...
Your Web site Powered
by Hot Banana

COLDFUSION Developer's Journal

editorial

editor-in-chief
Simon Horwith simon@sys-con.com

executive editor
Nancy Valentine nancy@sys-con.com

research editor
Bahadir Karuv, PhD bahadir@sys-con.com

production

lead designer
Abraham Addo abraham@sys-con.com

art director
Alex Botero alex@sys-con.com

associate art directors
Louis F. Cuffari louis@sys-con.com
Tami Beatty tami@sys-con.com

editorial offices

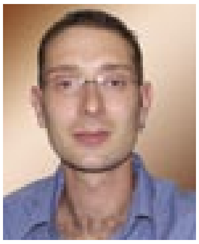
SYS-CON MEDIA
577 Chestnut Ridge Rd., Woodcliff Lake, NJ 07677
Telephone: 201 802-3000 Fax: 201 782-9638
COLDFUSION DEVELOPER'S JOURNAL (ISSN #1523-9101)
is published monthly (12 times a year)
by SYS-CON Publications, Inc.

postmaster: send address changes to:
COLDFUSION DEVELOPER'S JOURNAL
SYS-CON MEDIA
577 Chestnut Ridge Rd., Woodcliff Lake, NJ 07677

Worldwide Newsstand Distribution
Curtis Circulation Company, New Milford, NJ
FOR LIST RENTAL INFORMATION:
Kevin Collopy: 845 731-2684, kevin.collopy@edithroman.com
Frank Cipolla: 845 731-3832, frank.cipolla@epostdirect.com

For promotional reprints, contact reprint
coordinator Megan Mussa, megan@sys-con.com.
SYS-CON Publications, Inc., reserves the right to
revise, republish and authorize its readers to use
the articles submitted for publication.

Scorpio and Apollo – Coming Soon to a Town Near You!



By Simon Horwith

Yes, you read the title correctly: "Scorpio," the highly anticipated eighth major release of ColdFusion is coming soon to a town near you. Even before the release of the highly successful ColdFusion MX 7 two years ago, the ColdFusion team was already hard at work scheming, experimenting, planning, building, creating...and the result is "Scorpio."

Building on top of the powerful platform introduced in ColdFusion MX, and the solid feature set of ColdFusion MX 7, Scorpio piles on new features and technologies for developers, administrators, technical decision makers, and more. The official release of Scorpio is projected to be around the mid-2007 timeframe, but you don't have to wait until then to see it for yourself.

Ben Forta (as well as Tim Buntel in some locations) will be demonstrating lots of Scorpio throughout an extended user group tour in the spring, where attendees will get to see Scorpio in action, as well as gain access to the pre-release beta, and get the chance to win Adobe software. At the time of this writing there are 25 presentations scheduled from April 23 to May 21, but the schedule of dates/locations is being updated regularly. To check if a ColdFusion user group near you is participating, visit the schedule at http://labs.adobe.com/wiki/index.php/Scorpio.

What are the new features in Scorpio? Scorpio is currently in beta and members of the beta are under NDA, so new features can't be discussed. It has been announced that in addition to the user group tour dates, Scorpio topics will also be discussed at the CFObjective and CFUnited conferences. I and many other developers blogged about features we'd like to see – google the phrase "ColdFusion 8 wish list" (or "Scorpio wish list") to find many of those entries. You can also visit the Scorpio wiki at http://labs.adobe.com/wiki/index.php/Scorpio and look at the "Things that customers are looking for" section at the bottom of the page to see other features being requested. Of course, a wish list isn't the same as a feature list, so there are no guarantees, but Adobe does pay a lot of attention to what the community and the customers are asking for most often. Even if I could post a feature list, it's important to note that the product is currently in beta, so a feature here today could be gone tomorrow (and vice versa). One thing's for sure though – this truly is a very highly anticipated release (just look at the number of hits you get searching for wish lists) and the server team promises it's going to be another major feature-packed release (as opposed to a "patch" release), so keep your eyes and ears open for the tour and get ready to code. If you are not already on the Scorpio Beta and would like to apply to be on it, the URL is https://prerelease.adobe.com/callout/apply.html?callid={E9F64ADB-DADA-485E-BFFE-60E0D783EBEF} (there's also a link on the Scorpio Wiki referenced earlier).

– CONTINUED ON PAGE 12

About the Author

Simon Horwith is the editor-in-chief of ColdFusion Developer's Journal *and is the CIO at AboutWeb, LLC, a Washington, DC based company specializing in staff augmentation, consulting, and training. Simon is a Macromedia Certified Master Instructor and is a member of Team Macromedia.*
simon@horwith.com

SYS-CON MEDIA

founder & ceo
Fuat Kircaali fuat@sys-con.com

group publisher
Jeremy Geelan jeremy@sys-con.com

advertising

senior vp, sales & marketing
Carmen Gonzalez carmen@sys-con.com

vp, sales & marketing
Miles Silverman miles@sys-con.com

advertising sales director
Megan Mussa megan@sys-con.com

associate sales manager
Corinna Melcon corinna@sys-con.com

sys-con events

president, events
Carmen Gonzalez carmen@sys-con.com

events manager
Lauren Orsi lauren@sys-con.com

customer relations

circulation service coordinator
Edna Earle Russell edna@sys-con.com

sys-con.com

vp, information systems
Robert Diamond robert@sys-con.com

web designers
Stephen Kilmurray stephen@sys-con.com
Richard Walter richard@sys-con.com

accounting

financial analyst
Joan LaRose joan@sys-con.com

accounts payable
Betty White betty@sys-con.com

subscriptions

Subscribe@sys-con.com
Call 1-888-303-5282

For subscriptions and requests for bulk orders,
please send your letters to Subscription Department
Cover Price: $8.99/issue
Domestic: $89.99/yr (12 issues)
Canada/Mexico: $99.99/yr
All other countries $129.99/yr
(U.S. Banks or Money Orders)
Back issues: $12 U.S. $15 all others

Data Table Gateways

Working on a collection of records

By Jeffry Houser

In my previous article I wrote about Data Access Objects. Data Access Objects, or DAOs for short, are a way to separate your insert, select, delete, and update queries from other business logic. This lets you switch from one data storage mechanism to another easily. Whenever people talk about DAOs they also talk about Data Gateways.

I've also heard them called Table Gateways or more commonly gateway objects. The two often go together and are similar in concept. Data Access Objects are designed to work on a single record, whereas Table Gateways are designed to work on a collection of records.

MyFriends' RSSCategories

In the article on DAOs, I took a component from my RSS Aggregator project, MyFriends, and changed its implementation to use a Data Access object. You can download the aggregator code from the software pod on my blog at www.jeffryhouser.com. For this article, I thought I'd take the same data, RSSCategories, and create a gateway component.

When you enter RSS feeds into the system, they can be categorized in any way you like, and the category information is stored in the RSSCategory table. The table has two columns, a primary key, CategoryID, and a category name column called reasonably enough Category. When creating the DAO, I took an existing component and modified it to use the DAO pattern. Currently, the system doesn't have a gateway component yet, so in this case we'll start from scratch.

Generic Properties & Methods

When creating a Gateway object there are often generic properties and methods that I use. You can encapsulate these into a GenericGateway component. All future gateways will inherit from the gateway.

Here are the generic properties:

- **DSN:** When you're accessing a database from ColdFusion, you need to know the name of the datasource. This property holds that.
- **ColumnList:** The columnlist property will contain the name of the database columns that you want to retrieve from the database. I usually default this to '*'.
- **MaxRows:** How many rows do you want to return? In most cases, you want to return all of the rows, so I default the maxrows property to –1.
- **OrderList:** The orderlist property contains a list of all the fields that you're going to order the query results by. The order list is going to be dependent on the columns in the query. I default this to a blank string.
- **Criteria:** This is usually a set of properties that you use to define the selection criteria from the query. Perhaps you only want users whose names begin with the letter A. Maybe you only want products whose price is less than $5. I don't implement generic methods in the GenericGateway for these, since they're often specific to the query you want to run.

Here are the generic methods:

- **Getters and Setters:** The getters and setters are inherited from the Base Com-

TABLE OF CONTENTS

ponent. I use Hal Helm's generic getter and setter methods, available from his Web site at http://halhelms.com/webresources/BaseComponent.cfc.
- **Init:** The init method is one that will define the generic properties of the component such as the DSN, ColumnList, MaxRows, and OrderList.
- **Exceute:** The execute method is one that will piece together the query from the various property information, run it, and return the query.
- **Criteria methods:** In some cases, the criteria can be more complex than you would set with a simple getter or setter. Perhaps you want to use a range of numbers in your query. Maybe you want to test for equality and similarity using wildcards. Sometimes these are implemented better with their own criteria setting method from inside the sub-component. In most cases, I don't have any additional criteria setting methods. You can use your judgment.

Next I'll show the implementation of the GenericGateway object and then the implementation of an RSSCategories gateway.

The Generic Gateway Object

You can take a look at the code in the Generic Gateway object in Listing 1. It starts out with the cfcomponent tag. Act surprised. The component extends the BaseComponent. The pseudo-constructor code initializes four generic properties: dsn, columnlist, maxrows, and orderlist. There's a single method named init. The init method accepts the four separate arguments, one for each property. If that property is defined, it overrides the default. This is a pretty generic component. I leave the implementation of the execute method for the specific components that inherit from the generic gateway.

Writing the RSSCategoryGateway Object

The RSSCategoryGateway.cfc can be seen in Listing 2. The component extends the GenericGateway, thus inheriting all its methods and properties. It adds two instance variables to the mix, CategoryID and Category. In this case, I'm not changing any of the default values, or I would override them as part of the pseudo-constructor.

There's one new method in this gateway, the execute method. Of course, this component inherits all the methods from the genericGateway, and its parent is the BaseComponent. The execute method, you'll remember, will piece together the query, run it, and return the results. That's exactly what this one does.

The method vars the query name so it stays local to the method. Then it has a cfquery tag. The query selects the columnlist from the table. Since this isn't intended to be generic, I didn't use the tablename as a variable. Then I enter the where conditions. I don't know if there will be any conditions or not, so I use an SQL trick, where 0=0; 0 is always equal to 0, so this condition will always be true no matter what data is returned from the query. Using this as the first condition the query lets me use, which remains true of all future conditions, since there will always be a prior condition. The code checks to see if the CategoryID is zero. If it is, do nothing. If it's not, filter the output based on the CategoryID value. I set up CategoryID to test equality, although it could easily do a greater than or less than, or something else completely depending on the data you're trying to retrieve. Next it checks the Category instance variable. If it's an empty string, do nothing. Otherwise, add in the Category check clause. I added a wildcard to the category field.

If there are no criteria, a finished query may be as simple as:

```
Select *
From RSSCAtegories
Where 0=0
```

If both criteria are used together, the query may turn out something like:

```
Select *
From RSSCAtegories
Where 0=0
 And RSSCAtegories.CategoryID = 1
 And RSSCAtegories.Category like ‘A%’
```

This may seem like a lot of overhead when using a table with just a two fields. But, with larger tables, or more complicated queries, the benefits can be seen more easily. You might use this component when creating a system for editing the categories, but another gateway when creating reports based on categories and RSSFeeds in those categories.

Using the Gateway

I can show you a simple example of how we can use the component RSSCategoryGateway component. First we need to create an instance and run the init method:

```
variables.RSSCategories = CreateObject(“component”,"#request.ComponentLoc#.RSSCategoryGateway");
variables.RSSCategories.init(‘*’,’category’,-1, request.DSN);
```

This was put inside a CFSCript block. The init method just resets the defaults in this case with the exception of the DSN. A blank DSN won't do us any good. This piece of code will run the simple query, with no filters:

```
ResultsNoFilter = variables.RSSCategories.execute();
```

You can dump ResultsNoFilter to see all

entries in the RSSCategories table. Let's add a filter:

```
variables.RSSCategories.set(‘category’,’A’);
ResultsCategoryFilter = variables.RSSCategories.execute();
```

You can easily dump the results to see all the categories that start with the letter A. This is a simple concept that has a lot of power especially when dealing with complicated queries.

Final Thoughts

As with DAO objects, I feel that Gateways implementations in ColdFusion are severely lacking in documentation. Everyone talks about why to use them; no one talks about how to implement them. I hope this article helped give you a head start on using gateways. It's easy for me to think of variations of this implementation that can achieve the same level of encapsulation and reuse, and still meet the definition of a gateway.

I'm now entering my third year of writing this column. Sometimes it's hard to figure out what to write about in a beginner's column that hasn't been done ad nauseam. I'd love to get some feedback from readers on what they want me to discuss in the coming year. To contact me, just go to my blog at www.jeffryhouser.com and fill out the contact form.

And one last plug, for those who are dying to hear the sound of my voice, I'm the co-host of a Flex-related podcast at www.theflexshow.com. Give a listen if you're interested in Flex!

About the Author

Jeffry Houser has been working with computers for over 20 years and in Web development for over 8 years. He owns a consulting company and has authored three separate books on ColdFusion, most recently ColdFusion MX: The Complete Reference (McGraw-Hill Osborne Media).

jeff@instantcoldfusion.com

Listing 1

```
<cfcomponent extends=”BaseComponent”>

<cfscript>
 variables.instance.dsn = “”;
 variables.instance.columnlist = “*”;
 variables.instance.maxrows = -1
 variables.instance.orderlist = “”;
</cfscript>

<cffunction name=”Init” access=”Public” returntype=”void” hint=”I init
the component”>
 <cfargument name=”columnlist” type=”string” required=”false”>
 <cfargument name=”orderlist” type=”string” required=”false”>
 <cfargument name=”maxrows” type=”numeric” required=”false”>
 <cfargument name=”dsn” type=”string” required=”false”>

 <cfif IsDefined(“arguments.columnlist”)>
  <cfset variables.instance.columnlist = arguments.columnlist>
 </cfif>
 <cfif IsDefined(“arguments.orderlist”)>
  <cfset variables.instance.orderlist = arguments.orderlist>
 </cfif>
 <cfif IsDefined(“arguments.maxrows”)>
  <cfset variables.instance.maxrows = arguments.maxrows>
 </cfif>
 <cfif IsDefined(“arguments.dsn”)>
  <cfset variables.instance.dsn = arguments.dsn>
 </cfif>

</cffunction>

</cfcomponent>
```

Listing 2

```
<cfcomponent extends=”GenericGateway”>

<cfscript>
 variables.instance.CategoryID = 0;
 variables.instance.Category = “”;
</cfscript>


<cffunction name=”execute” access=”public” returntype=”query”>

 <cfset var getCategories = “”>
 <cfquery name=”getCategories” datasource=”#variables.instance.dsn#”>
  select #variables.instance.columnlist#
  from RSSCAtegories
  where 0=0
  <cfif variables.instance.CategoryID NEQ 0>and RSSCAtegories.CategoryID
= #variables.instance.CategoryID# </cfif>
  <cfif variables.instance.Category NEQ “”>and RSSCAtegories.Category
like ‘#variables.instance.Category#%’ </cfif>
  </cfquery>

 <cfreturn getCategories>

</cffunction>

</cfcomponent>
```

Download the Code...
Go to http://coldfusion.sys-con.com

cfunited
Adobe
New Atlanta COMMUNICATIONS
CFDynamics
Universal Mind
<SEEFUSION>
ColdFusion Developer's Journal
cfunited
June 27- 30, 2007
Washington D.C area
Apollo
Step through your CFML code with FusionDebug
Using Structures with CFCs
ColdFusion Application Security
CSS - Back to Basics
Flex and Ajax: perfect match
Object Oriented Flex
Testing, Monitoring & Tuning CF w/ Open Source tools
Testing for Accessibility
Coding with XML
Flex 2 for ColdFusion Developers
Flex Data Services & CF Scorpio – Faster, Better & Cooler!
Custom Tags
AJAX Development with ColdFusion Frameworks
Integrating Spry and ColdFusion
Creating and Consuming WebServices
Beyond Basic SQL For CF
Working with RSS
Flex 2 for ColdFusion Developers
Programing with ColdSpring
Creating and Consuming WebServices
CSS - Back to Basics
Flex Data Services & CF Scorpio
Coding with XML
Step through your CFML code
ColdFusion Application Security
Flex and Ajax: perfect match
Using Structures with CFCs
Testing, Monitoring & Tuning CF w/ Open Source tools
TeraTech
HostMySite.com
AboutWeb
Paper | Thin
ShadoCMS with ZoomFLEX
Fig Leaf Software
Your seat

the premier coldfusion technical conference

Don't miss out on the Premier ColdFusion conference, CFUNITED-07 where you can learn from over 50 expert speakers and authors such as Ben Forta, Michael Smith, Simon Horwith, Hal Helms, Ray Camden, Charlie Arehart. Four days of sessions, networking and learning with your programming peers from around the world. Guarantee yourself a seat and register today.

"CFUNITED is a great way to not only network with fellow ColdFusion afficianados, but also to learn new techniques and methods to help your career or business. You also can get inside scooops on developments in ColdFusion and learn best practices and much more."
-Angela T , Attendee

"It is run by developers for developers. The sessions are based on real world experience and real user case studies."
-Rich P , Attendee

"CFUNITED is a great opportunity to launch your career into the next level, allowing developers to learn new tips, tricks and techniques, all while bonding with fellow ColdFusion users many of us only know by name."
-Constanty D , Attendee

"Learning more, networking more, getting NEW ideas...and now, to support CF's continued existence."
-Howard P, Attendee

TeraTech Inc.
405 East Gude Dr, Ste 207
Rockville, MD 20850
301.424.3903

www.cfunited.com

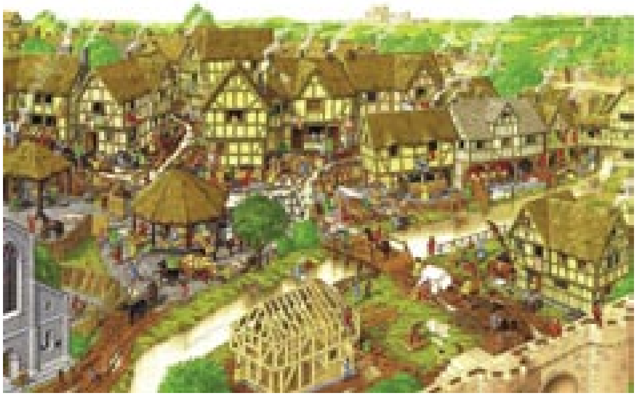
– CONTINUED FROM PAGE 5

Scorpio and Apollo – Coming Soon to a Town Near You!

One other very recent noteworthy announcement from Adobe is the availability of the Apollo alpha. Apollo is a cross-operating system runtime environment developed by Adobe that allows Web developers to leverage their current knowledge of Flex, ActionScript/Flash, HTML, JavaScript, HTML, and AJAX to build desktop applications (or, as the Adobe Apollo Wiki describes it, "to build and deploy rich Internet applications [RIAs] to the desktop"). Apollo is currently supported on Windows and OSX and plans are in the works to support Linux in the near future. There are many planned features such as PDF support, the ability to launch external applications, and many more (a more complete list is on the Apollo FAQ at http://labs.adobe.com/wiki/index.php/Apollo:developerfaq#What_are_some_of_the_features_that_are_not_included_in_the_Apollo_alpha.3F).

Many developers and clients have asked me why a ColdFusion developer should be interested in Apollo, and how Apollo impacts the role of the CF developer in an organization. These are very good questions that I plan to address in more detail in a future editorial, but for now I'll offer this brief bit of insight.

As for why a ColdFusion developer may be interested in Apollo, there are two ways in which I see Apollo being extraordinarily useful for us. One is the obvious fact that using ColdFusion alone, your application is Web based and runs on an application server, whereas Apollo let's you build something that runs on the desktop. This means your end users don't need to remember or bookmark any URL (just access the application from their desktop), your applications can do things that Web applications can't traditionally do such as access the file system and other resources on the client machine, and end users can even use your application when they don't have any Internet connectivity or when your servers are not online. This last benefit also makes Apollo ideal when users may want to take the application home or to some other remote location, do their work (storing data locally and using data they've retrieved from the server for offline use), and then possibly synch the work they've done offline with your servers when they return to the office or obtain connectivity. I also see Apollo as being very useful in that developers will be able to distribute prototypes, demos, and/or finished products to end users not only via the Web but on CD and other physical storage devices. Can't you already do this with Java? Well, kind of but not entirely and the short answer to that question is "no" (yes, you can run a desktop app written in Java anywhere you have a runtime and yes, you can do some simple Web scripting rendering with Java) – but Apollo really is different, and Apollo allows Web developers to leverage their existing skills to build these applications – the significance of which cannot be overstated.



Regarding how Apollo impacts the role of ColdFusion or ColdFusion developers in an organization – that's tough to say so early in the game and it also depends largely on the nature of the application(s) on a project-by-project basis. In an Apollo application, the role of ColdFusion will typically be that of a service provider. It's a typical client/server relationship where data persistence is (most likely) being performed on the client and business logic may or may not be performed on the client depending on whether this logic must be server-side or may need to be available to applications other than the Apollo client apps. Logic written using CFCs can be used by Apollo, Flex 2, and ColdFusion applications and can be exposed as a Web service that's consumable by any other SOAP-enabled environment. Logic in an Apollo app is available to the Apollo app. On the other hand, logic in the Apollo app runs client side, thus greatly reducing server load as well as dependence on an active Internet connection. It's important to keep in mind that in addition to offering services that provide database interaction and/or perform business logic on the server, ColdFusion will be a good candidate to provide services that generate/manipulate documents, leverage products deployed on the enterprise (Verity, mail servers, Java/COM/CORBA objects, etc.), and that access other server-side resources (such as the servers local file system).

If you aren't interested in developing desktop applications or distributing your current content to the desktop, then Apollo may not be something you need immediately dive into, but it's definitely something to better familiarize yourself with in the event that it suits your needs in the future. If nothing else, it's a skill that will most likely make you a more marketable developer and could make your contributions in the workplace more valuable. Though the features are yet to be publicly announced, Scorpio certainly looks to be a ColdFusion release with many significant new features that allow you to do more, more easily and/or in better ways, so I strongly recommend attending the user group presentation nearest you and signing up for the Scorpio beta. If you're fortunate enough to not only be a CF developer but to also be one of the early adopters of Flex, then 2007 looks to be a very productive, interesting, and fun year for you and your organization. Keep your eye on the Adobe Labs site for breaking news, downloads, and other information, as Adobe is doing a fantastic job releasing loads of "goodies" at a great rate.

2007
VIRTUALIZATION
CONFERENCE + EXPO
www.virtualizationconference.com

Register Now!
EARLY-BIRD SAVINGS!
$200

ROOSEVELT HOTEL
NEWYORK CITY

June 25-27, 2007

Virtualization: Solutions for the Enterprise.

Delivering The #1 i-Technology Educational and Networking Event of the Year!

As today's CIOs and IT managers rise to the challenge of using their enterprise computing infrastructure as cost-effectively as possible while remaining responsive in supporting new business initiatives and flexible in adapting to organizational changes, Virtualization has become more and more important.

A fundamental technological innovation that allows skilled IT managers to deploy creative solutions to such business challenges, Virtualization is a methodology whose time has come. The fast-emerging age of Grid Computing is enabling the virtualization of distributed computing, of IT resources such as storage, bandwidth, and CPU cycles.

But Virtualization can also apply to a range of system layers, including hardware-level virtualization, operating system level virtualization, and high-level language virtual machines.

Register Online!
www.VirtualizationConference.com

THE FIRST MAJOR VIRTUALIZATION EVENT IN THE WORLD!

HURRY, FOR EXHIBITOR AND SPONSORSHIP OPPORTUNITIES
CALL 201-802-3020

— BROUGHT TO YOU BY —
ENTERPRISE OpenSource MAGAZINE
SOAWORLD MAGAZINE

SYS-CON EVENTS For more great events visit www.EVENTS.SYS-CON.com

Scary Question.

Exactly who is developing your next app?

ColdFusion/.NET Integration

Beyond Web services

By Catalin Sandu

As both a .NET programmer and ColdFusion developer, I always wondered how I could leverage the world of .NET in ColdFusion. Both platforms come with powerful features and using them together might be a wonderful friendship, if one could only make them cooperate. There are two worlds out there and none of them is an island.

Apart from this, and starting with Windows Vista, Microsoft will include the last incarnation of the .NET Framework in its flagship operating system, with all the bells and whistles that come with it. For those who like their ColdFusion environment hosted on a Windows server, this means they'll always have all the features provided in the .NET platform at hand. Even if a switch to Vista won't be an option in the near future for current projects, you may still wonder how you can put them to work for your ColdFusion application.

The promise of the next version of ColdFusion (code named "Scorpio" and due out later this year) is that it will support .NET natively. This also means that access to this platform will be of interest to ColdFusion developers, in the future anyway. Still, apart from the ability to use .NET objects directly from CFML, there are other ways of making ColdFusion and the .NET Framework talk to each other.

Here we're going to explore some of these methods. This article assumes a basic knowledge of event gateways and how to use external objects from ColdFusion (Web services and COM objects), as well as a fair understanding of the .NET architecture. On the one hand, there's your world, ColdFusion. On the other side is .NET. So what are our bridging options?

Consuming Web Services

Of course, the first answer that comes to mind is to build and deploy ASP.NET Web services. You don't have to know the intricate details of the Simple Object Access Protocol (SOAP) or Web Services Definition Language (WSDL) to successfully write such a component in .NET (or ColdFusion for that matter) – at least for most developing tasks. The .NET Framework will take care of all details for you.

For example, the class that will implement the methods exposed by a Web service doesn't even need to be derived from the Web Service class found in .NET, although doing so will provide the service with direct access to different common ASP.NET objects (session state, application state, request context, and so on). On the other hand, the methods that should be public for the Web service must be decorated with the WebMethod attribute. Not all methods should be decorated this way, effectively making those methods private to the implementation of the component.

Consuming a .NET Web service from ColdFusion is just a matter of using CFOBJECT or CreateObject, depending on the developer's preferences. Care should be taken when defining method parameters and return values so that both platforms understand each other (for example, you can't pass a ColdFusion structure to .NET as is).

Anyway, this road of Web services has already been taken and discussed in many articles about ColdFusion and .NET. I won't go further than that and start exploring other interoperability options.

Talking .NET 3.0: Windows Communication Framework Services

In the last release of the .NET Framework, Microsoft complemented the previous version with a few interesting components. In fact, .NET 3.0 is the old 2.0 platform; the Common Language Runtime (CLR) that sits at the base of the framework hasn't been upgraded at all. On top of it there are now four main technologies written entirely in .NET 2.0: Windows Presentation Foundation (WPF), Windows Workflow Foundation (WF), Windows CardSpace (WCS), and Windows Communication Framework (WCF), which is the focus here.

There are many technologies that are meant for making two applications talk to each other. Depending on the project requirements, one can choose between named pipes, message queue-based communication, peer-to-peer protocols, and so on, including Web services. The rationale behind WCF is to provide a unifying programming model for all existent communication technologies, at the same time offering the possibility of defining your own custom transport protocol. Instead of trying to learn multiple technologies, a programmer would have to know WCF. All components written according to the WCF model can easily talk to applications exposing their functionality through any protocol.

To expose a class to the outside world through WCF, one must define an interface and decorate it with the ServiceContract attribute, while the methods defined for that interface should be decorated with an OperationContract attribute (the ones that aren't decorated as such won't be exposed at all). The C# example below defines an interface (in WCF speak, a contract) exposing a single operation:

```
[ServiceContract]
public interface IHello {
    [OperationContract]
    string SayHello(string name);
}
```

The actual implementation must be provided in a class that derives from this interface. To start the service, several other steps must be followed, such as configuring the service (that is, informing the WCF system about the address where the service will be exposed; the configuration is done in a special XML file called app.config), and hosting it (meaning, running the application – there are also several options here, such as compiling the service as a console application, running it inside IIS, implementing it as a Windows service, and so on).

The good news for a ColdFusion developer is that the WCF components can be configured to act as Web services. From a ColdFusion point-of-view, such a WCF object is just another Web service exposing its functionality via WSDL, thus being accessible through CFOBJECT/CreateObject.

I have provided a full sample on how to define and consume a WCF service from ColdFusion. It includes both the source code for the contract (see Listing 1) and the configuration XML file (see Listing 2), as well as explanatory comments for the relevant lines of code. The service has been implemented as a simple console application so you can see what is happening when it's invoked: every time an operation is called from ColdFusion, the service will print a log message.

Note that you should first start the service before consuming it from ColdFusion. The sample will run on localhost on port 4500. You can access it through http://localhost:4500/<WCFServiceName>?WSDL.

Of course, console applications may not be the best choice for hosting a service; a better scenario would be to host it in IIS (as a regular Web service), or simply as a Windows Service. But this also depends on the project type you're working on.

Using the Socket Event Gateway

One of the greatest additions to the ColdFusion application server is the event gateway system. Gateways are classes written in Java according to a well-defined application programming interface (API); they are capable of reacting to external events and handling requests not coming through the usual HTTP channel via a Web browser. Event gateways open up your application to the world of asynchronous programming, and so to a whole range of applications. You can even roll your own gateway type, or use the ones provided out-of-the-box in ColdFusion.

Among the gateway samples coming with ColdFusion is the socket event gateway. You can check out the source code for this type of gateway at <ColdFusion Install Folder>\gateway\src\examples\socket\SocketGateway.java. To use it you have to define a gateway instance first. The default configuration for this socket event gateway will use port 1225 for communication, but you can change it in the configuration file (see <ColdFusion Install Folder>\gateway\config\socket.cfg). After the gateway instance has been defined, it should be started from the ColdFusion Administrator.

Keep in mind that event gateways are a feature of the Enterprise and Developer editions of ColdFusion.

Writing a simple application that sends and gets data over a socket connection is very easy in .NET. All the classes you'll need are located in the System.Net.Sockets namespace. Here's how you can open up a connection and send some data to a socket in C#:

```
Socket s = new Socket();
s.Connect(host, port_number);
s.Send(your_message);
```

The Send method has a series of overloads that might be helpful; it returns the number of bytes that were actually sent. The Socket class also has different properties that can control the communication. To read data from a socket you use the Receive method as follows:

```
byte[] buffer = new byte[number_of_bytes_to_read];
s.Receive(buffer);
```

Listing 3 is a very simple application (again, a console application) that will send a series of strings to a socket opened on port 1225. The gateway instance (the component in Listing 4) does nothing but return the received string back to the .NET application, which will display it back to you.

Of course, sending simple strings over the wire isn't spectacular, but you can define your own protocol to be followed by all clients that might communicate with your gateway. One possible scenario is sending XML fragments between the client and the server (the gateway instance); what's more, you can use Web Development Data Exchange (WDDX) for this. You already have WDDX support on the ColdFusion side, while in .NET you can use the COM library found on www.openwddx.com.

Expose .NET Components as COM Objects

Let's come back for a while to our old friend CFOBJECT (and its cousin CreateObject). Sure you know that besides Web services you can instantiate COM objects on Windows as well. All you have to do is replace "web service" with "com" for the object type parameter and supply the component programmatic identifier (ProgID). Then you're free to use it whatever way you like. After you're done with the object, you mustn't forget to call ReleaseComObject so all resources associated with the COM object are freed up properly.

With careful design, most .NET assemblies can be exposed as COM objects, making them accessible for COM clients – including ColdFusion pages. The magic is done with a little help from a very neat thing called COM Callable Wrapper (CCW) that comes with the .NET Framework. CCW acts as a proxy between your .NET classes and the COM system. It takes care of reference counting for you, knows how to convert .NET types so that they are visible to the outside world, and comes with built-in implementations for a few standard COM interfaces.

There are some rules to be observed to make a .NET class COM-visible. First of all, you must mark your assembly as being COM-visible. Do that by using the ComVisible attribute in the AssemblyInfo file for your project (AssemblyInfo.cs for a C# project, AssemblyInfo.vb for VB.NET, and so on). Assuming you're writing a C# solution, this is what you need to write to make your assembly visible for COM clients:

```
[assembly: ComVisible(true)]
```

By default, ComVisible is set to false; you have to change this as shown above. This way, the .NET objects already defined in the assembly will be available to COM consumers for free. A few restrictions still apply as you'll see in a moment; what should be mentioned here is that you still have the option to control what is exposed or not. Simply by applying the ComVisible(false) attribute to one of your types (class, interface, whatever), that type becomes hidden again – you can even apply this attribute to methods, properties, and data members.

To effectively expose a .NET class to COM, a developer needs to know some rules. First, the class must be public, non-abstract, and have a default constructor. Second, what will be exposed (either data members or methods) must be marked as public; no protected, private, or internal members will be available to COM. What's more, all static stuff goes the same way; they won't be visible from outside.

Most of the COM-exposing stuff is done via some clever class- and method-level attributes. For example, you can assign your own dispatch identifiers for the class methods and properties, and you can choose whatever ProgID you want for a class and so on. To describe them all is beyond the scope of this article, however, you're free to explore the available documentation on msdn.microsoft.com.

Here's a short example for a class that can be exposed to COM:

```
public class ExposedClass {
   public void DoThis();     // This method is exposed
   public int someData;      // This data member will be exposed as a
                                COM property
   private long hidden;        // NOT exposed data
   public static float sharedStuff;        // NOT exposed because of the
                                                  static modifier
   [ComVisible(false)] public void DoThat(); // NOT exposed, because of
                                                  ComVisible(false)
}
```

Finally, the assembly needs to be registered in the Windows registry so that COM can properly instantiate your exposed .NET stuff. However, since the resulting DLL isn't a true COM component, you won't be able to do this via RegSvr32.exe. For this, you have to use RegAsm.exe, or the assembly registration tool, coming with the .NET Framework:

```
regasm.exe YourAssembly.dll
```

To unregister an assembly, just add the /unregister (or /u) switch to the end of the command above. The registration story isn't complete without mentioning that the recommended way of doing this is to create a strong name for your assembly (via the sn.exe utility from .NET), and to put it in the Global Assembly Cache (GAC) – although for testing purposes you can forget about that. If the assembly isn't located in the GAC, you must add the /codebase switch when running the RegAsm.exe tool; this will put the correct path to your DLL in the registry so that COM clients will instantiate your objects without any problems. Listing 5 is a complete example of a .NET class that can be used from any COM client.

Coming back to ColdFusion, all you have to do is use CFOB-

JECT on the new COM object. One testing idea is to CFDUMP the variable created via CFOBJECT so you can see that the .NET class is fully visible from ColdFusion; you'll get a list of all of the methods and properties for that component (along with the usual stuff inherited from the IDispatch and IUnknown interfaces, such as GetIDsOfNames, Invoke, AddRef, and so on).

There's more than meets the eye in the .NET-COM story, a subject that requires its own series of articles to be discussed properly. Anyway, this should be enough information to get you started, if this is the road you want to take.

Writing CFX Tags in Managed C++

For the next stop in the ColdFusion/.NET interoperability game we need to move further down the list of features offered by ColdFusion. One of the many ways through which the performance of the ColdFusion application server can be improved is to use that special API for creating ColdFusion Extension tags (CFX). They can be written in C++ or Java and must be registered for use via the ColdFusion Administrator.

Java CFX tags are in fact components that must target the Java platform (obviously), and, as such, this language isn't a valid option if you need access to .NET. While Visual J# can be more or less considered a variant of Java, it still remains a language that targets the .NET platform, and the resulting assemblies can't be used as Java CFX tags. So what's left for you is C++.

CFX tags written in C++ are dynamic libraries (DLLs) that must export a special function that will be called by ColdFusion whenever the tag is used from a CFML page. ProcessTagRequest is the name suggested by the ColdFusion Administrator for this function, but any name will do as long as it's a valid C++ identifier. Beyond that, you can implement whatever logic you want in the CFX tag.

The best news when writing a CFX in Visual C++ is that you can combine managed code with the more mundane version of the language. The whole world of .NET is open for you to explore in a CFX tag – the only catch is that you have to compile your Visual C++ project with support for the Common Language Runtime (CLR), but this is already turned on for you when the solution is created.

The fine print in this story is that you need to convert everything that's expressed as .NET variables to string values whenever you need to return such variables back to ColdFusion. However, this isn't a real problem, since all .NET constructs ultimately derive from the System.Object class, which in turn has a handy method called ToString that you can use as is, or override for new .NET objects. You'd want to consult the code in Listing 6 for a C++ sample tag to see this in action.

Of course, the CFX tag can be designed in such a way that the .NET Framework will be used for its internal workings only, and never for building return variables. The overhead mentioned above won't be a bother in this case.

CLR Programming in SQL Server 2005

The last integration option comes from an unexpected source: the database-related tags from ColdFusion, CFQUERY, and CFSTOREDPROC – as well as their associated tags. Actually, this is possible only when using SQL Server 2005 as a back-end database server.

SQL Server supports the creation of various database objects using the .NET Framework – a feature better known as CLR integration. Be aware though that the default installation of SQL Server 2005 sets the CLR integration flag to disabled and it should be activated explicitly. There are two methods to do this. First there's the Transact-SQL way via the sp_configure system stored procedure. When called with no arguments, it will return a list with all configuration options in SQL Server, along with their current state. The option that should be modified is clr enabled, as shown in the following snippet:

```
sp_configure ‘clr enabled’, 1
reconfigure
```

The call to reconfigure is mandatory to apply the changes done via sp_configure. To disable the CLR integration again, use zero for the second parameter of the stored procedure.

The second method is to open up the Surface Area Configuration tool found in the All Programs/Microsoft SQL Server 2005/Configuration Tools folder. Click on Surface Area Configuration for Features for the computer you want to configure, and locate the CLR Integration option for your SQL Server 2005 instance. Check Enable CLR integration here and then apply the changes.

Coming back to programming, the .NET namespace that contains everything that's needed to build managed database objects in SQL Server 2005 is Microsoft.SqlServer.Server. These objects are stored procedures, triggers, user-defined types, functions, and aggregates – most of them being usable from ColdFusion via CFQUERY or CFSTOREDPROC.

I'm going to show you how to build a managed stored procedure in Visual Studio; the other CLR objects can be created more or less in the same way.

The first step is to create a database project in Visual Studio in your language of choice. Once created, you can add any kind of managed database object to this solution. All of them follow the same pattern: they are created in a separate class, and the actual implementation is found in a public static method (Shared in Visual Basic .NET).

Note that a special attribute decorates this static method. For stored procedures it's SqlProcedure, for user-defined functions it's SqlFunction, and so on. They're added using the full .NET namespace path (Microsoft.SqlServer.Server.SqlProcedure, for example), but you can safely remove the root if you want (as long as you don't forget to add a using or Import directive at the beginning of the file).

The most important object you'll be using here is SqlContext. It provides access to a SqlPipe object (via the Pipe property), which in turn can be used to send some data back to the caller:

```
public partial class StoredProcedures {
    [SqlProcedure]
    public static void DotNetStoredProcedure() {
            SqlContext.Pipe.Send(“String from a .NET stored procedure”);
    }
}
```

Of course, the sample above is sending back just a simple string, but you also have methods for creating and returning result sets from a stored procedure. Listing 7 shows you how it's done; it will return a result set containing all the languages supported by the .NET Framework (that is, the ISO codes), along with their English and native names.

Once compiled, the resulting assembly must be deployed on SQL Server 2005. This can be done by Visual Studio for you (go to Build/Deploy Solution in the menu), or manually via SQL.

This last method involves two separate statements: one for registering the assembly with SQL Server 2005 (see the documentation on CREATE ASSEMBLY), and the other for creating the stored procedure or whatever other managed object needs to be deployed (using the CREATE PROCEDURE/FUNCTION/TRIGGER/TYPE/AGGREGATE family of statements) – you use the appropriate DROP command to get rid of them from the database. Having done all this, it's just a matter of CFQUERY/CFSTOREDPROC on the ColdFusion side to actually use them in your project. From a ColdFusion point-of-view, you're consuming database objects, and that's all. Everything is transparent to you.

Conclusion

There are lots of options when considering using ColdFusion and .NET together. This article discussed some of them; other possibilities may still exist, of course. For example, using CFEXECUTE to run an external application (which can be a .NET one, too), but this method has its own risks and I didn't bother to bring it to the table.

By now you saw that I didn't mention anything about New Atlanta's BlueDragon for .NET. This is a perfectly valid option, too, and you're free to try it, if this is your application server of choice. In fact, most methods described here apply to BlueDragon as well (the obvious exception being event gateways, which aren't supported on BlueDragon).

There's more to be said for each of the techniques described in this article. Still, I'm sure you have enough to start thinking about how can you extend ColdFusion to the .NET world.

About the Author

Catalin Sandu is a software developer at RomSoft (www.rms.ro) with 10 years of experience. He is both a Microsoft Certified Professional (on C++ and .NET), and an Advanced ColdFusion MX 7 Developer. Catalin is also a member of the British Computer Society since 2005.

cat@rms.ro

Listing 1

```
using System;
using System.Collections.Generic;
using System.Text;

// Mandatory namespace for WCF (you need .NET 3.0 for this)
using System.ServiceModel;

namespace WCFHello
{
    /*
     * Define an interface for exposing to WCF.
     * Use the [ServiceContract] attribute to expose
     * the interface as a WCF service.
     * */

    [ServiceContract]
    interface ISayHello
    {
        /*
         * Use the [OperationContract] attribute for
         * every method that is exposed to WCF.
         * */

        [OperationContract] string GetGreeting(string name);
        [OperationContract] int    GetRandomInteger();
    }

    /*
     * We implement the above interface as a class that is
     * private to our assembly
     * */

    class SayHello : ISayHello
    {
        // This is a console application.
        // For every call to our WCF service,
        // we’ll write a short message for our user

        public string GetGreeting(string name)
        {
            Console.WriteLine(“ISayHello.GetGreeting called”);
            return “Hello world, and hi there, “ + name;
        }

        public int GetRandomInteger()
        {
            Console.WriteLine(“ISayHello.GetRandomInteger called”);

            Random r = new Random();

            return r.Next();
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            // This is a self-hosted WCF service; use
            // a ServiceHost instance for this
            ServiceHost sh = new ServiceHost(typeof(SayHello));

            sh.Open();
```

```
            // Wait for calls
            Console.WriteLine(“The WCF service is up and running, press
<ENTER> to stop it”);
            Console.ReadLine();

            sh.Close();
        }
    }
}
```

Listing 2

```
<?xml version=”1.0” encoding=”utf-8” ?>
<configuration>
    <system.serviceModel>
        <behaviors>
            <serviceBehaviors>
                <behavior name=”WCFSayHelloMetadata”>
                    <serviceMetadata httpGetEnabled=”true” />
                </behavior>
            </serviceBehaviors>
        </behaviors>
        <services>
            <service behaviorConfiguration=”WCFSayHelloMetadata”
                            name=”WCFHello.SayHello”>
                <endpoint binding=”basicHttpBinding” bindingConfigura
                            tion=””
                    name=”WCFSayHelloEndpoint” contract=”WCFHello.
                                ISayHello” />
                <host>
                    <baseAddresses>
                        <add baseAddress=”http://localhost:4500/WCFSay
                            Hello” />
                    </baseAddresses>
                </host>
            </service>
        </services>
    </system.serviceModel>
</configuration>
```

Listing 3

```
using System;
using System.Text;
using System.Net;
using System.Net.Sockets;

namespace NETSockets
{
    class Program
    {
        static void Main(string[] args)
        {
            // Construct the endpoint (localhost:1225)
            IPHostEntry ipHostInfo = Dns.GetHostEntry(“localhost”);
            IPAddress   ipAddress  = ipHostInfo.AddressList[0];
            IPEndPoint  remoteEP   = new IPEndPoint(ipAddress, 1225);

            Console.WriteLine(“Connecting to the ColdFusion Socket
Gateway...\r\n”);

            // Create the client socket and connect to our endpoint
            Socket client = new Socket(
                AddressFamily.InterNetwork,
                SocketType.Stream,
                ProtocolType.Tcp);
            client.Connect(remoteEP);

            int numBytes  = 0;
            byte[] buffer = null;



            // Write the first response from our socket.
            // This is the welcome message coming from ColdFusion.
            if (client.Poll(-1, SelectMode.SelectRead))
            {
                // Read & show the message
```

CFDJ Advertiser Index

ADVERTISER	URL	PHONE	PAGE
Adobe	www.adobe.com/products/coldfusion/flex2		2,3
AJAXWorld Bootcamp	www.AJAXbootcamp.com	201-802-3000	49
CFDynamics	www.cfdynamics.com	866-233-9626	35
Cfunited	www.cfunited.com		11
CFDJ	http://coldfusion.sys-con.com/	201-802-3000	25
Cynergy Systems	www.cynergysystems.com		14, 15
EdgeWebHosting	edgewebhosting.net	1-866-334-3932	4
Hal Helms	halhelms.com		45
HostMySite	www.hostmysite.com	877-215-4678	21, 36
Hotbanana	hotbanana.com/cfdj	866-296-1803	4
Intergral	www.fusiondebug.com		6
Paperthin	www.paperthin.com	1-800-940-3087	37
SOAWorld	www.soaworld2007.com	201-802-3020	33
Virtualization Conf & Expo	www.events.sys-con.com	201-802-3020	13
VitalStream	www.vitalstream.com/tools/mxdj.asp	800-254-7554	14-15

General Conditions: The Publisher reserves the right to refuse any advertising not meeting the standards that are set to protect the high editorial quality of. All advertising is subject to approval by the Publisher. The Publisher assumes no liability for any costs or damages incurred if for any reason the Publisher fails to publish an advertisement. In no event shall the Publisher be liable for any costs or damages in excess of the cost of the advertisement as a result of a mistake in the advertisement or for any other reason. The Advertiser is fully responsible for all financial liability and terms of the contract executed by the agents or agencies who are acting on behalf of the Advertiser. Conditions set in this document (except the rates) are subject to change by the Publisher without notice. No conditions other than those set forth in this “General Conditions Document” shall be binding upon the Publisher. Advertisers (and their agencies) are fully responsible for the content of their advertisements printed in ColdFusion Developer’s Journal. Advertisements are to be printed at the discretion of the Publisher. This discretion includes the positioning of the advertisement, except for “preferred positions” described in the rate table. Cancellations and changes to advertisements must be made in writing before the closing date. “Publisher” in this “General Conditions Document” refers to SYS-CON Publications, Inc.This index is provided as an additional service to our readers. The publisher does not assume any liability for errors or omissions This index is provided as an additional service to our readers. the publisher does not assume any liability for errors or omissions.

```
                numBytes = client.Available;
                buffer = new byte[numBytes];
                client.Receive(buffer, 0, numBytes, SocketFlags.None);
                Console.WriteLine(Encoding.ASCII.GetString(buffer, 0,
                    numBytes));
            }

            while (true)
            {
                Console.Write(“> “);

                // Read a string from the console
                string input = Console.ReadLine();

                if (input.Equals(“exit”))
                {
                    // Writing <exit> tells us to stop
                    break;
                }

                // Send our string to the socket
                client.Send(Encoding.ASCII.GetBytes(input + “\r\n”));

                // Retrieve the answer from the socket
                if (client.Poll(-1, SelectMode.SelectRead))
                {
                    numBytes = client.Available;
                    buffer = new byte[numBytes];
                    client.Receive(buffer, 0, numBytes, SocketFlags.
                        None);
                    Console.WriteLine(Encoding.ASCII.GetString(buffer,
                        0, numBytes));
                }
            }

            // Don’t forget to close our connection
            client.Shutdown(SocketShutdown.Both);
            client.Close();
            client = null;

            Console.WriteLine(“Disconnected from server. Press <ENTER>
to close the app”);
            Console.ReadLine();
        }
    }
}
```

Listing 4

```
<cfcomponent>

  <cffunction name=”onIncomingMessage” output=”no”>
      <cfargument name=”CFEvent” type=”struct” required=”yes”>

      <!--- Echo the received message --->
      <cfset retValue = StructNew()>
      <cfset retValue.DestinationID = CFEvent.OriginatorID>
      <cfset retValue.Message       = “Hello, “ & CFEvent.data.MESSAGE>

      <cfreturn retValue>
  </cffunction>

</cfcomponent>
```

Listing 5

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Runtime.InteropServices;

namespace COMExposed
{
    /*
     * Define a class and expose it to COM.
     * Assuming that your compiled assembly is called
     * COMExposed.dll, then the ProgId of this
     * class is “COMExposed.MiniCalc”
     *
     * You can define your own ProgID by decorating
     * the class like this: [ProgId(“YourProgram.YourComponent”)]
     * Remember that a ProgID is limited to 39 characters
     *
     * We expose the class to COM as ClassInterfaceType.AutoDual,
     * thus making our class available to both early- and
     * late-binding COM clients.
     *
     * The other options for ClassInterfaceType would be
     * None (IUnknown-derivative class) and
     * AutoDispatch (to indicate the class support only
     * late binding clients; AutoDispatch is the default value).
     * Note that ColdFusion uses only COM objects that
     * implement the IDispatch interface.
     * */

    [ClassInterface(ClassInterfaceType.AutoDual)]
    public class MiniCalc
    {
        // Default constructor to make CCW happy
        public MiniCalc() { }

        // The DispId  attribute is optional; it allows you to
        // choose your own DISPID for a method or property.
        // If omitted, you’ll end up with automatically generated
        // DISPIDs.

        [DispId(1)] public int Add     (int a, int b) { return a + b; }
        [DispId(2)] public int Subtract(int a, int b) { return a - b; }
        [DispId(3)] public int Multiply(int a, int b) { return a * b; }
        [DispId(4)] public int Divide  (int a, int b) { return a / b; }
    }
}
```

Listing 6

```
#include “stdafx.h”

using namespace System;
using namespace System::Text;
using namespace System::Reflection;
using namespace System::Runtime::InteropServices;

void ProcessTagRequest(CCFXRequest* pRequest)
{
  try
  {
      // Get the currently executing assembly
      System::Reflection::Assembly^ curAsm =
          System::Reflection::Assembly::GetExecutingAssembly();

      // Construct the output string value as a concatenation
      // between this assembly’s name, a random number,
      // and the current date
      StringBuilder^ sb = gcnew StringBuilder();
      sb->AppendFormat(“The assembly full name is \”{0}\”<br>”, curAsm-
>FullName);
      sb->AppendFormat(“Here’s a random number: \”{0}\”<br>”, Random().
Next());
      sb->AppendFormat(“Here’s the current date: \”{0}\”<br>”, Date-
Time().Now);

      // Store it in a managed string
      String^ sOutput = sb->ToString();

      // Convert the managed string to a normal string so
      // that we can return it to ColdFusion
      IntPtr ptr             = Marshal::StringToHGlobalAnsi(sOutput);
      const char* lpszOutput = (const char*)ptr.ToPointer();

      // Write this string in the browser; we could as well return
      // this string using pRequest->SetVariable or to add it
      // to a ColdFusion query.
      pRequest->Write(lpszOutput);

      // Don’t forget to free the output string
      Marshal::FreeHGlobal(IntPtr((void*)lpszOutput));
  }
  catch(CCFXException* e)
  {
      pRequest->ReThrowException(e);
  }
  catch(...)
  {
      pRequest->ThrowException(“General Error”, “”);
  }
}
```

Listing 7

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Data.SqlTypes;
using Microsoft.SqlServer.Server;
using System.Globalization;

public partial class StoredProcedures
{
    [Microsoft.SqlServer.Server.SqlProcedure]
    public static void GetDotNetLanguages()
    {
        // Retrieve a list of all supported cultures in .NET
        CultureInfo[] cultures =
            CultureInfo.GetCultures(CultureTypes.AllCultures);

        // Create a SqlDataRecord  object that will hold
        // all columns and column types that we’ll return from
        // our stored procedure
        SqlDataRecord record = new SqlDataRecord(
            new SqlMetaData(“Culture”, SqlDbType.NVarChar, 100),
            new SqlMetaData(“English Name”, SqlDbType.NVarChar, 100),
            new SqlMetaData(“Native Name”, SqlDbType.NVarChar, 100),
            new SqlMetaData(“Display Name”, SqlDbType.NVarChar, 100));

        SqlContext.Pipe.SendResultsStart(record);

        foreach (CultureInfo ci in cultures)
        {
            record.SetString(0, ci.Name);
            record.SetString(1, ci.EnglishName);
            record.SetString(2, ci.NativeName);
            record.SetString(3, ci.DisplayName);

            SqlContext.Pipe.SendResultsRow(record);
        }

        SqlContext.Pipe.SendResultsEnd();
    }
};
```

Download the Code...
Go to http://coldfusion.sys-con.com

Subscribe Today!

SAVE 16%

12 Issues for $89^{99}

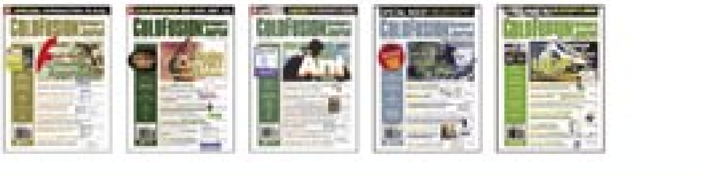
OFFER SUBJECT TO CHANGE WITHOUT NOTICE

AN ADVANCE LOOK AT BLAC
COLDFUSION
Leveragin
or Just F
Training: Fast Track to Fl
CFML: Harnessing the P
SQL Server Using Store
Foundations: A We
<BF> on <CF>: C
and Data Abstra
PDFs: Creatin
Your CF Appl
SPECIAL ISSUE: COLDFUSION SECURITY
COLDFUSION Developer's Journal
2005 East
SANDBOX SECURITY
Security: ColdFusion Security Best Practices Knowing the security risks are there is half the battle
Security: Public-Key Encryption Making strong encryption nearly painless
cfeUnit: ColdFusion Unit Testing Framework Paul Kenney's version
<bf> on <cf>: Defending ColdFusion Against...
Real-World Uses: Using ColdFusion for Network Monitoring Find the exact piece of data you need
Security: Top Ten Web Security Tips You can't be too careful
SYS-CON MEDIA

- Exclusive feature articles
- Latest *CFDJ* product reviews
- Interviews with the hottest names in ColdFusion
- Code examples you can use in your applications
- *CFDJ* tips and techniques

That's a savings of $29.89 off the annual newsstand rate. Visit our site at www.sys-con.com/coldfusion or call 1-800-303-5282 and subscribe today!

COLDFUSION Developers Journal



ColdFusion U

For more information go to...

U.S.

Alabama
Huntsville, AL CFUG
www.nacfug.com

Arizona
Phoenix, AZ CFUG
www.azcfug.com

California
Bay Area CFUG
www.bacfug.net

California
Sacramento, CA CFUG
http://www.saccfug.com/

California
San Diego, CA CFUG
www.sdcfug.org/

Colorado
Denver CFUG
http://www.denvercfug.org/

Connecticut
SW CT CFUG
http://www.cfugitives.com/

Connecticut
Hartford CFUG
http://www.ctmug.com/

Delaware
Wilmington CFUG
http://www.bvcfug.org/

Florida
Jacksonville CFUG
http://www.jaxfusion.org/

Florida
South Florida CFUG
www.cfug-sfl.org

Georgia
Atlanta, GA CFUG
www.acfug.org

Illinois
Chicago CFUG
http://www.cccfug.org

Indiana
Indianapolis, IN CFUG
www.hoosierfusion.com

Louisiana
Lafayette, LA MMUG
http://www.acadianammug.org/

Maryland
California, MD CFUG
http://www.smdcfug.org

Maryland
Maryland CFUG
www.cfug-md.org

Massachusetts
Boston CFUG
http://bostoncfug.org/

Massachusetts
Online CFUG
http://coldfusion.meetup.com/17/

Michigan
Detroit CFUG
http://www.detcfug.org/

Michigan
Mid Michigan CFUG
www.coldfusion.org/pages/index.cfm

Minnesota
Southeastern MN CFUG
http://www.bittercoldfusion.com

Minnesota
Twin Cities CFUG
www.colderfusion.com

Missouri
Kansas City, MO CFUG
www.kcfusion.org

Nebraska
Omaha, NE CFUG
www.necfug.com

New Jersey
Central New Jersey CFUG
http://www.cjcfug.us

New Hampshire
UNH CFUG
http://unhce.unh.edu/blogs/mmug/

New York
Rochester, NY CFUG
http://rcfug.org/

New York
Albany, NY CFUG
www.anycfug.org

New York
New York, NY CFUG
www.nycfug.org

New York
Syracuse, NY CFUG
www.cfugcny.org

North Carolina
Raleigh, NC CFUG
http://tacfug.org/

Ohio
Cleveland CFUG
http://www.clevelandcfug.org

Oregon
Portland, OR CFUG
www.pdxcfug.org

Pennsylvania
Central Penn CFUG
www.centralpenncfug.org

Pennsylvania
Philadelphia, PA CFUG
http://www.phillycfug.org/

Pennsylvania
State College, PA CFUG
www.mmug-sc.org/

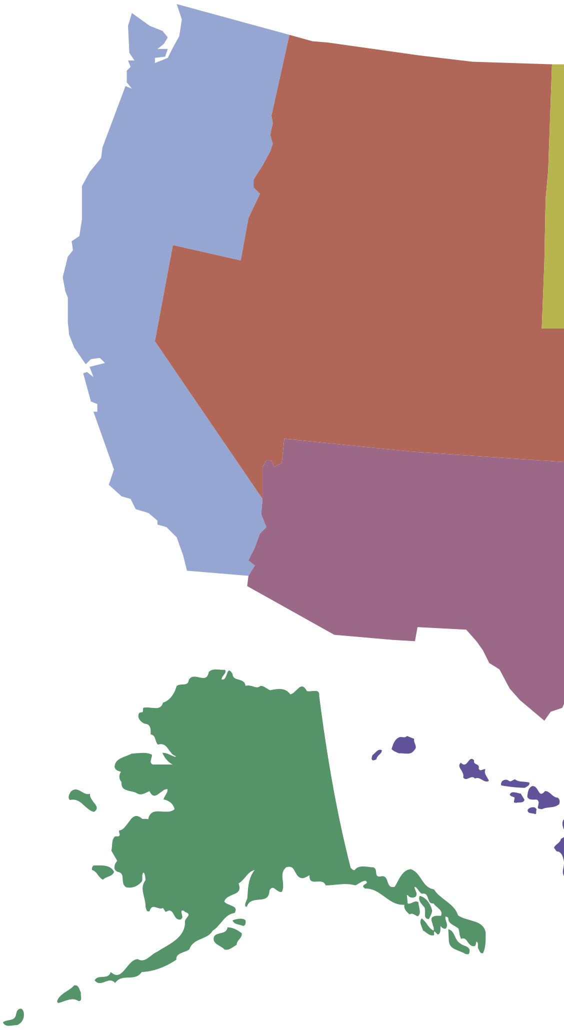
Tennessee
Nashville, TN CFUG
http://www.ncfug.com

Tennessee
Memphis, TN CFUG
http://mmug.mind-over-data.com

Texas
Austin, TX CFUG
http://cftexas.net/

Texas
Dallas, TX CFUG
www.dfwcfug.org/

Texas
Houston Area CFUG
http://www.houcfug.org



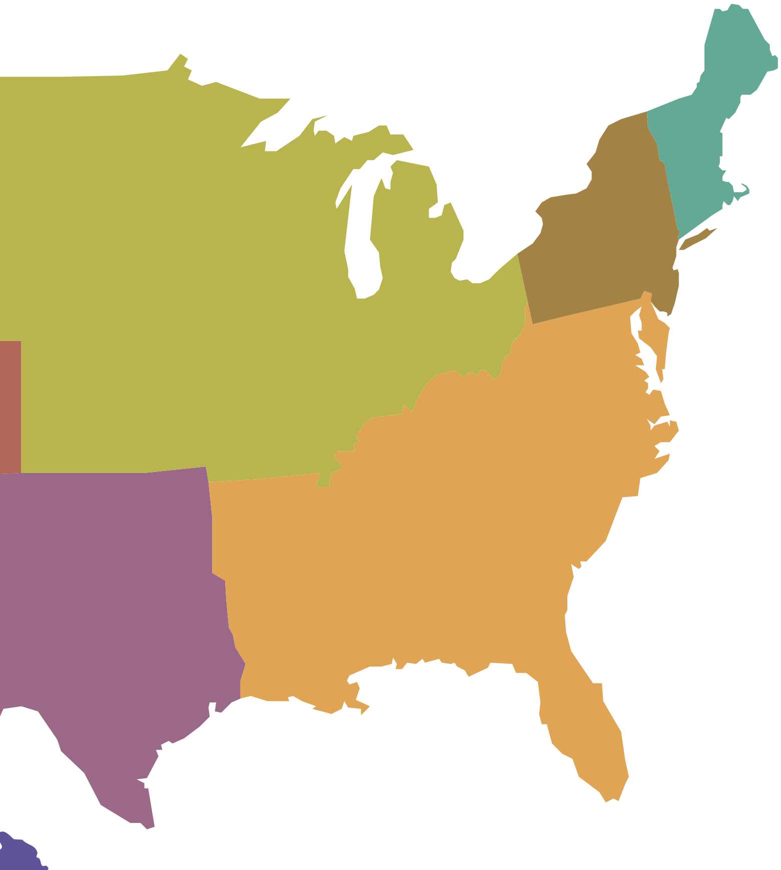
Utah
Salt Lake City, UT CFUG
www.slcfug.org

Virginia
Charlottesville CFUG
http://indorgs.virginia.edu/cfug/

Washington
Seattle CFUG
http://www.seattlecfug.com/

User Groups

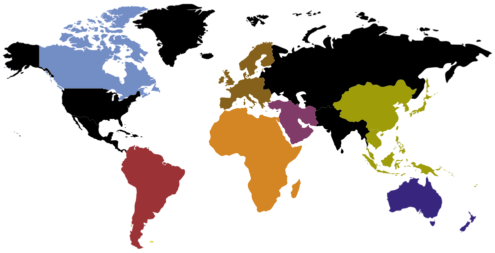
http://www.macromedia.com/cfusion/usergroups



About CFUGs

ColdFusion User Groups provide a forum of support and technology to Web professionals of all levels and professions. Whether you're a designer, seasoned developer, or just starting out – ColdFusion User Groups strengthen community, increase networking, unveil the latest technology innovations, and reveal the techniques that turn novices into experts, and experts into gurus.

INTERNATIONAL



Australia
ACT CFUG
http://www.actcfug.com

Australia
Queensland CFUG
http://qld.cfug.org.au/

Australia
Victoria CFUG
http://www.cfcentral.com.au

Australia
Western Australia CFUG
http://www.cfugwa.com/

Canada
Kingston, ON CFUG
www.kcfug.org

Canada
Toronto, ON CFUG
www.cfugtoronto.org

Germany
Central Europe CFUG
www.cfug.de

Italy
Italy CFUG
http://www.cfmentor.com

New Zealand
Auckland CFUG
http://www.cfug.co.nz/

Poland
Polish CFUG
http://www.cfml.pl

Scotland
Scottish CFUG
www.scottishcfug.com

South Africa
Joe-Burg, South Africa CFUG
www.mmug.co.za

South Africa
Cape Town, South Africa CFUG
www.mmug.co.za

Spain
Spanish CFUG
http://www.cfugspain.org

Sweden
Gothenburg, Sweden CFUG
www.cfug-se.org

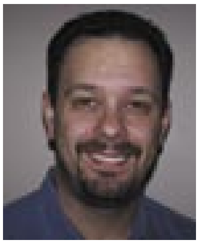
Switzerland
Swiss CFUG
http://www.swisscfug.org

Turkey
Turkey CFUG
www.cftr.net

United Kingdom
UK CFUG
www.ukcfug.org

The Dark Side of Gateways

How to cope with a sort of denial of service attack from the inside



By Dave Ferguson

Gateways; you've heard about them. You've read the hype. You've probably created one or two. You've seen applications do wonderful things with them. There have been articles written about them.

Don't be fooled. Not everything is always the way it seems. There's a dark side to the gateway system. A side where things don't go as expected.

I think the gateway system is like an exotic sports car with a child at the wheel. It is full of power, has extensive form and tons of function. But it seems like someone with no knowledge or understanding is at the controls.

Gateways can be very powerful; so powerful that they can flood a server from the inside creating what could be called a sort of denial of service attack from the inside. It's obviously not an attack but the effect is about the same. The server becomes overloaded handling gateway requests and can run out of resources to handle normal traffic.

This usually occurs when you have multiple gateways and they all start firing at the same time. It can even be caused by an asynchronous gateway that's repeatedly fired and doesn't terminate. You can even have a condition where you have hung gateways that aren't terminating.

Due to the nature of gateways there's no easy way to recover from this. Your only recourse is to restart the ColdFusion service. Although restarting the service will fix the problem, you'll have lost whatever was running at restart. You could end up with corrupt data or a whole slew of other issues.

My goal is to help you take control of the gateway system. I'm going to show you how to put a "Gatekeeper" on top of gateways; a method in which all gateway requests are queued. Once queued they can be controlled better and prevent the gateways from overrunning the server. But creating a "Gatekeeper" won't save you from crashes.

The amount of work is minimal. All you need to do is modify your existing gateway cfcs. Just a couple of minor tweaks to your current cfcs and you'll be off and running.

The queue system has two parts: the "Gatekeeper" and a "Dispatcher." They each have their own job but work together to create the queue. Once you actually have this you can create a monitor to keep tabs on everything. This article won't cover this part but the online code contains code for a monitor.

The Gatekeeper takes incoming gateway requests and converts them to server vars that will be used by the Dispatcher and the Monitor. The Dispatcher takes the server vars created by the Gatekeeper and creates asynchronous calls to the appropriate cfcs. The Monitor does just what its name suggests. It keeps tabs on the jobs running and will stop the processing of new requests if things go badly.

The final part is the original gateway cfcs themselves. They need a slight alteration that will let them work with the queue. It starts with the alterations to the current gateway cfcs. The code in Listing 1 demonstrates the code changes needed. Copy the onAdd function and create a new function called GKonAdd. Then the onAdd event is altered to send the event to the Gatekeeper.

Listing 2 is the Gatekeeper itself. All it does is take the event passed to it and stores it in a server variable called "server.cfcevents." You can use either the application or the server scope for this. However, if you use application you have to deal with application timeouts.

The Gatekeeper creates a structure for each gateway that's using it. The structure name is based on the gateway id from the CF administrator. After adding the structure it then fires the Dispatcher asynchronous gateway. You'll have to create this gateway in the CF administrator.

The Dispatcher has a case for each gateway event as well as how many concurrent requests can be running. The Dispatcher is designed to run forever once it starts. It contains code to make it pause as well as detect if it's already running when it starts. If a Dispatcher instance is already running the new instance will shut down. The code for the Dispatcher is very involved and lengthy so it's not in this article but it is available online.

Once you have the Dispatcher set up for your gateways you're ready to go. If you need to you can also set up special conditions in the Dispatcher that will let a request be run immediately and bypass the queue.

Unfortunately I haven't figured out how to do crash recovery. You could write events to a text file instead of the application or server scope. However, then you're limited as to what data you can store. This also adds excess overhead.

And there you have it. You're now back in control of your system. You no longer have to worry about your gateways taking over your system. Hopefully in the future functionality like this will be core to CF and a workaround like this won't be necessary.

But until then we have to do what we can to remain in control of our systems.

About the Author

Dave Ferguson is a system architect and principal programmer. He has been doing website design and development for over 10 years. He is also a Certified Advanced ColdFusion Developer. You can read his blog at www.dkferguson.com/BlogCFC.

dave@dkferguson.com

Listing 1

```
<CFCOMPONENT HINT=”LISTING 1” EXTENDS=”GateKeeper”>
  <CFFUNCTION NAME=”onAdd” OUTPUT=”FALSE”>
     <CFARGUMENT NAME=”CFEvent” TYPE=”struct” REQUIRED=”yes”>

     <CFSCRIPT>
        props = structNew();
        props.event = CFEvent;
        storeEvent(props);
     </CFSCRIPT>

  </CFFUNCTION>
  <CFFUNCTION NAME=”GKonAdd” OUTPUT=”FALSE”>
     <CFARGUMENT NAME=”CFEvent” TYPE=”struct” REQUIRED=”yes”>

        <!--- original onAdd event code --->

  </CFFUNCTION>
</CFCOMPONENT>
```

Listing 2

```
<CFCOMPONENT EXTENDS=”functions” HINT=”Handles incomming gateway re-
quests and put them into queue”>
   <CFFUNCTION NAME=”storeEvent” OUTPUT=”FALSE” ACCESS=”PUBLIC”
HINT=”Processes ingomming messages”>
      <CFARGUMENT NAME=”inEvent” TYPE=”struct” REQUIRED=”yes”>

     <CFSCRIPT>

        try {
            GatewayName = trim(inEvent.event.GATEWAYID);
            if (NOT IsDefined(“server.CFCEvents.#GatewayName#”)){
               “server.CFCEvents.#GatewayName#” = ArrayNew(2);
            }
            next = ArrayLen(server.CFCEvents[GatewayName])+1;
            server.CFCEvents[GatewayName][next][1] = now();
            server.CFCEvents[GatewayName][next][2] = inEvent.event;

            fireDispatcher = SendGatewayMessage(“dispatcher”);
        } catch (any ept){
            MailAlert(ept); // send alert
        }

    </CFSCRIPT>

   </CFFUNCTION>
</CFCOMPONENT>
```

Download the Code...
Go to http://coldfusion.sys-con.com

A Flex 2 DVD Library Application

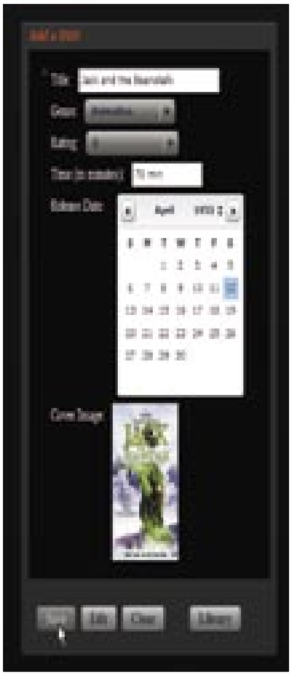
Or how I saved some money

By Michael Givens

I have been collecting DVD movies for years now and, on some occasions, I have forgotten that I already owned a particular movie. More times than I hate to admit, I have ended up with two copies.

At my wife and budget's insistence, I created a Flex 2 application with a ColdFusion, CFC-based back end that allows me to store a DVD inventory with the expected data inputs (see Figure 1) into a mySQL database. This Flex 2 application displays the inventory in a DataGrid (see Figure 2) with an itemRenderer that displays the DVD cover art, and it utilizes an Amazon Web service (http://labs.insideflex.com/flextraining/movies/bin/ExampleAWSCall.xml) to dynamically locate the URL of a medium-sized image of the cover. In addition, a back-end CFC function automatically calls the YouTube API to grab the URL of a trailer for the movie, based on the title entered (see Figure 3).

Let's take a peek at the MXML code (see Listing 1). (The end of Listing 1 and all of Listing 2 can be downloaded from http://coldfusion.sys-con.com.) First, the database is a simple table (no normalization was used, but that certainly could be added). The movies table structure (http://labs.insideflex.com/flextraining/movies/bin/movies.sql.txt) is shown in Figure 4. Next, I used the Flex Builder 2's ColdFusion CFC Value Object Wizard to quickly create the server-side components from the database structure. Three CFCs and one ActionScript class later, the back end was complete (see Listing 2). The creationComplete event, in the dvdLibrary.mxml, calls a function, pageLoaded(), that triggers a RemoteObject call to the movieGateway component's getAllAsQuery() function. The returned data is pushed into a Bindable ArrayCollection, arcMovies, which serves as the dataProvider for the DataGrid, dgMovies. ItemRenderers are used to format the cells of the DataGrid, including a CoverRenderer to display the cover art for the DVD's Image data (this data



Add a DVD

Title: Jack and the Beanstalk
Genre: Animation
Rating:
Time (in minutes): 90 min
Release Date: April
Cover Image:

Figure 1

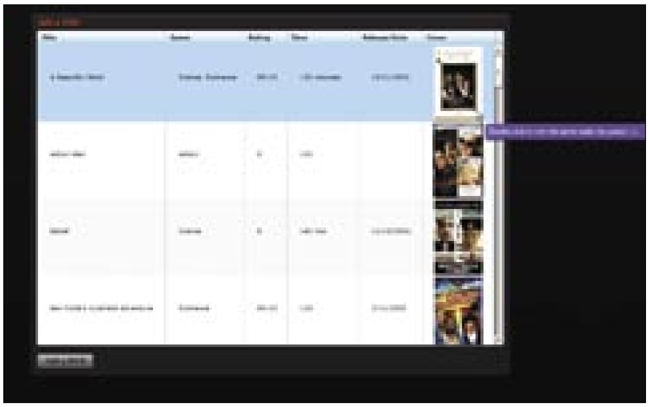


Figure 2

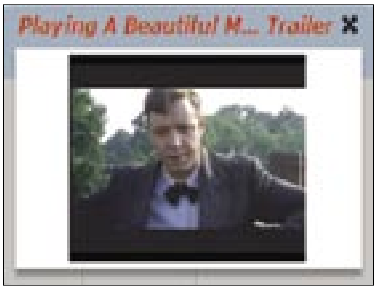


Figure 3

is provided by the Amazon Web service (1)). In addition, a call to a YouTube (2) Web service supplies a movie trailer for each DVD added to the library. Adding a new DVD has only one required field – the DVD's title. The mx.events.ValidationResultEvent and mx:StringValidator are used to ensure that a title is entered (3). The title is passed to both of the Web services and saved to the database. A TitleWindow (4) is utilized to display a larger image of the cover artwork as well as to display the movie trailer.

Building this project was fun as well as useful. The full source code is available at http://labs.insideflex.com/flextraining/movies/bin/srcview/index.html.

Now there is simply no excuse for accidentally buying a DVD I already own. I will soon be entering my entire collection of DVDs into this new Flex 2 DVD library. From now on, I plan to check this library before heading out to the video store. For the final online version of this Web application (http://labs.insideflex.com/flextraining/movies/bin/dvdLibrary.html), I've added search functionality so that as the library grows, it will still be easy to filter out and find the movies. I already have plans to port this Flex Web application to an Apollo desktop application in the coming weeks. If you would like to keep an eye on my progress, please visit my blog, www.flexination.info, when you have time. If you have any follow-up questions, feel free to ping me at the e-mail address listed with the article.

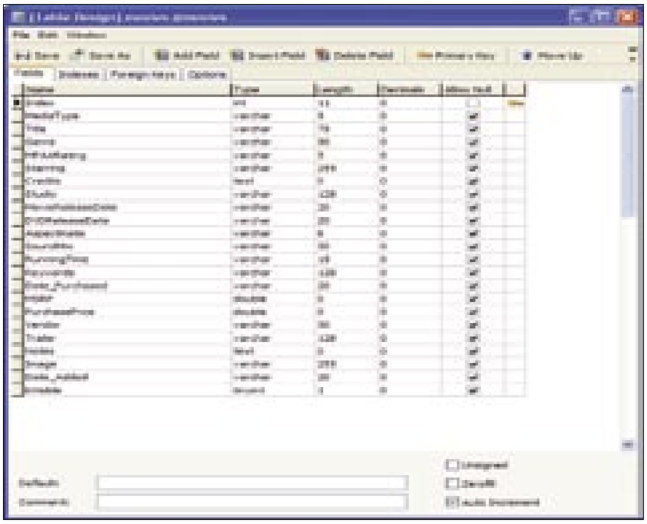


Figure 4

About the Author

Michael Givens is the CTO of U Saw It Enterprises, a Web technology consulting firm based in Marietta, GA. As an experienced Web technology specialist, he is willing to shift gears at a moments notice to the client's technology of choice. He is both an Adobe Community Expert and an Adobe Corporate Champion known to share his experience and evangelism of all things Adobe. Certified in both ColdFusion 5 and as an Advanced CFMX Developer, he has been using ColdFusion since the days of Allaire Spectra. Michael blogs at http://www.flexination.info/.

info@webmxml.com

Listing 1

```
Flex 2 UI code
dvdLibrary.mxml:
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
pageTitle="DVD Library" creationComplete="pageLoaded()"
layout="vertical" backgroundColor="black" viewSourceURL="srcview/index.
html">
    <mx:HTTPService id="AWSECommerceService" resultFormat="e4x"
result="getImages_handler(event)" fault="ro_fault(event)"
showBusyCursor="true"/> (1)
    <mx:HTTPService id="AWSECommerceServiceFinal" resultFormat="e4x"
result="getImageFinal_handler(event)" fault="ro_fault(event)"
showBusyCursor="true"/>
```

```
    <mx:RemoteObject id="roMovies"
        destination="ColdFusion"
        source="flextraining.movies.info.flexination.cfc.movieGateway"
        showBusyCursor="true">
        <mx:method name="save" result="save_handler(event)" fault="ro_
fault(event)"/>
        <mx:method name="getAllAsQuery" result="getAllAsQuery_
handler(event)" fault="ro_fault(event)"/>
        <mx:method name="writeAmzWSToXML" result="writeAmzWSToXML_
handler(event)" fault="ro_fault(event)"/>
        <mx:method name="cleanupXML" result="cleanupXML_handler(event)"
fault="ro_fault(event)"/>
        <mx:method name="proxyToYouTube" result="getTrailer_
handler(event)" fault="ro_fault(event)"/>
```

```
        <mx:method name=”getAllByTitle” result=”getAllByTitle_
handler(event)” fault=”ro_fault(event)”/>
    </mx:RemoteObject>

    <mx:Style>
        Panel {
           borderColor: #666666;
           borderAlpha: 0.4;
           roundedBottomCorners: true;
           headerHeight: 22;
           backgroundAlpha: 1;
           backgroundColor: #000000;
           titleStyleName: “mypanelTitle”;
        }

        .mypanelTitle {
           color: #ff3300;
           textAlign: left;
           fontSize: 12;
           fontWeight: bold;
           fontStyle: italic;
           paddingLeft: 20;
        }
    </mx:Style>
    <mx:Script>
        <![CDATA[
            import mx.managers.PopUpManager;
            import mx.rpc.events.ResultEvent;
            import mx.rpc.events.FaultEvent;
            import mx.controls.Alert;
            import mx.utils.ObjectUtil;
            import mx.collections.ArrayCollection;
            import mx.managers.PopUpManager;
            import mx.containers.TitleWindow;
            import mx.events.ValidationResultEvent;
            import mx.controls.ToolTip;
            import mx.managers.ToolTipManager;
            import info.flexination.actionscripts.movie;

            [Bindable] private var arcMovies:ArrayCollection;
            [Bindable] private var oMovie:movie = new movie;
            [Bindable] private var iMovie:Number = 0;
            [Bindable] private var sGenre:String = “”;
            [Bindable] private var sRating:String = “”;
            [Bindable] private var sSearchType:String = “Title”;
            [Bindable] private var blnCleanupSuccess:Boolean = false;
            [Bindable] private var XMLFileName:String = “”;
            [Bindable] private var xmlSuccess:XML;
            [Bindable] private var TrailerURL:String;
            [Bindable] private var ImgURL:String = “”;
            [Embed(source=”images/search-icon.png”)]
            [Bindable] private var imgSearch:Class;
            private var vResult:ValidationResultEvent; (3)
            private var msg:String = “”; (3)
            private var count:Number = 0; (3)
            private var msgprefix:String = “”; (3)
            private var ErrorTip:ToolTip; (3)

            private function pageLoaded():void {
                //retrieve all the movies
                roMovies.getAllAsQuery();
            }

            private function doSave():void {
                // call the roMovies save method
                if (iMovie>0) {
                    oMovie.Index = iMovie;
                    progressUpdateIndicator();
                } else {
                    oMovie.Index = iMovie;
                    progressSaveIndicator();
                }
                oMovie.MediaType = “DVD”;
                oMovie.Title = tiTitle.text;
                oMovie.Genre = sGenre;
                oMovie.MPAARating = sRating;
                oMovie.Starring = “”;
                oMovie.Credits = “”;
                oMovie.Studio = “”;
                oMovie.MovieReleaseDate = dteRelease.text;
                oMovie.DVDReleaseDate = “”;
                oMovie.AspectRatio = “”;
                oMovie.SoundMix = “”;
                oMovie.RunningTime = tiTime.text;
                oMovie.Keywords = “”;
                oMovie.Date_Purchased = “”;
                oMovie.MSRP = 0;
                oMovie.PurchasePrice = 0;
                oMovie.Vendor = “”;
                oMovie.Notes = “”;
                oMovie.Image = ImgURL;
                oMovie.Date_Added = date_updated();
                //Alert.show(ObjectUtil.toString(oMovie));
                //create a new movie
                roMovies.save(oMovie);
                //retrieve all the active movies
                roMovies.getAllAsQuery();
                // clear out all the fields
            }

            private function clearForm():void {
                iMovie = 0;
                tiTitle.text = ‘’;
                sGenre = “”;
                cbxGenre.selectedIndex = 0;
                sRating = “”;
                cbxRating.selectedIndex = 0;
                tiTime.text = ‘’;
                dteRelease.text = ‘’;
                oMovie.Trailer = “”;
                imgCoverArt.source = null;
                ImgURL = ‘’;
                btnSave.label = ‘Save’;
            }

            private function date_updated():String {
                var date:Date = new Date;
                var current:String = dateformatter.format(date.get-
Month() + 1 + ‘/’ + date.getDate() + ‘/’ + date.getFullYear())  + ‘ ‘ +
date.getHours() + ‘:’ + date.getMinutes() + ‘:’ + date.getSeconds();
                //Alert.show(current);
                return current;
            }

            private function save_handler(event:ResultEvent): void {
                iMovie = event.result as Number; // change flag to an
update one
            }

            private function getAllAsQuery_handler(event:ResultEvent):
void {
                arcMovies = event.result as ArrayCollection;
            }

            private function getAllByTitle_handler(event:ResultEvent):
void {
                arcMovies = event.result as ArrayCollection;
            }
```

> “Businesses that ignore the potential of SOA will find themselves outpaced by rivals who improve their agility and transform themselves into new kinds of enterprises
>
> — Yafim Natis, Gartner Analyst

3-DAY EVENT!

SOAWorld 2007 Plus Enterprise OpenSource Conference & Expo 2007

TOPICS INCLUDE:

SOA Web Services
- AJAX and SOA
- Web 2.0
- Universal SOA
- Protecting Web Services
- Troubleshooting SOA
- Governance
- Open-Source SOA
- XBRL
- Service Virtualization

Open Source
- Open Source Business Models
- Open Source ESB
- OpenAjax Alliance
- SaaS and Open Source
- Spring, Hibernate and Eclipse
- Seam
- Open Source Penetration
- Monetizing Open Source
- Open Source Databases
- AMQP
- Open Source Middleware

June 25-27, 2007
Roosevelt Hotel / New York City

Register Online! www.SOAWorld2007.com

SOAEOSCONFERENCE.SYS-CON.COM

REGISTER ONLINE TODAY
SAVE $200!
(HURRY FOR EARLY-BIRD DISCOUNT)

11th International 2007 SOAWORLD™ CONFERENCE & EXPO

2nd Annual ENTERPRISE> 2007 OPENSOURCE CONFERENCE+EXPO

2007 is to many industry insiders shaping up to be a major inflection point in software development and deployment, with SOA, Web Services, Open Source, and AJAX all converging as cross-platform and cross-browser apps become the rule rather than the exception.

Accordingly the 11th International SOA Web Services Edge 2007 again seeks to offer comprehensive coverage and actionable insights to the developers, architects, IT managers, CXOs, analysts, VCs, and journalists who'll be assembling as delegates and VIP guests in The Roosevelt Hotel in downtown Manhattan, June 25-27, 2007

Co-located with the 2nd Annual Enterprise Open Source Conference & Expo, the event will deliver the #1 i-technology educational and networking opportunity of the year. These two conference programs between them will present a comprehensive view of all the development and management aspects of integrating a SOA strategy and an Open Source philosophy into your enterprise. Our organizing principle is that delegates will go away from the intense two-day program replete with why-to and how-to knowledge delivered first-hand by industry experts.

Visit soaeosconference.sys-con.com for the most up-to-the-minute information including... Keynotes, Sessions, Speakers, Sponsors, Exhibitors, Schedule, etc.

BROUGHT TO YOU BY:

SOA World Magazine focuses on the business and technology of Service-Oriented Architectures and Web Services. It targets enterprise application development and management, in all its aspects.

Enterprise Open Source Magazine EOS is the world's leading publication showcasing every aspect of profitable Open Source solutions in business and consumer contexts.

SYS-CON EVENTS For more great events visit www.EVENTS.SYS-CON.com

Exhibit and Sponsorship Info:
Call 201-802-3020 or email events@sys-con.com

```
            private function ro_fault(event:FaultEvent): void {
                // dump error message
                Alert.show(ObjectUtil.toString(event.fault));
            }

            private function doView(): void {
                if (dgMovies.selectedItem.Trailer!=“”) {
                    var tw:MovieTrailer=MovieTrailer(PopUpManager.
createPopUp(this, MovieTrailer , false));
                    tw.movietitle = dgMovies.selectedItem.Title;
                    tw.trailer = dgMovies.selectedItem.Trailer;
                    tw.x = 620;
                    tw.y = 150;
                 } else {
                     Alert.show(“Trailer unavailable...”, “Ooops”);
                 }
            }

            private function getGenre(selIdx:Number): void {
                //Alert.show(selIdx.toString());
                switch (selIdx) {
                    case 0:
                        sGenre = “Select Genre...”;
                        break;
                    case 1:
                        sGenre = “Action”;
                        break;
                    case 2:
                        sGenre = “Animation”;
                        break;
                    case 3:
                        sGenre = “Comedy”;
                        break;
                    case 4:
                        sGenre = “Drama”;
                        break;
                    case 5:
                        sGenre = “Family”;
                        break;
                    case 6:
                        sGenre = “Horror”;
                        break;
                    case 7:
                        sGenre = “Musical”;
                        break;
                    case 8:
                        sGenre = “Romance”;
                        break;
                    case 9:
                        sGenre = “Sci-Fi”;
                        break;
                    case 10:
                        sGenre = “Western”;
                        break;
                }
            }

            private function getRating(selIdx:Number): void {
                //Alert.show(selIdx.toString());
                switch (selIdx) {
                    case 0:
                        sRating = “Select Rating...”;
                        break;
                    case 1:
                        sRating = “G”;
                        break;
                    case 2:
                        sRating = “PG”;
                        break;
                    case 3:
                        sRating = “PG-13”;
                        break;
                    case 4:
                        sRating = “R”;
                        break;
                    case 5:
                        sRating = “NC-17”;
                        break;
                    case 6:
                        sRating = “Adult”;
                        break;
                }
            }

            private function getSearchType(selIdx:Number): void {
                //Alert.show(selIdx.toString());
                switch (selIdx) {
                    case 1:
                        sSearchType = “Title”;
                        break;
                    case 2:
                        sSearchType = “Genre”;
                        break;
                    case 3:
                        sSearchType = “MPAARating”;
                        break;
                    case 4:
                        sSearchType = “RunningTime”;
                        break;
                    case 5:
                        sSearchType = “MovieReleaseDate”;
                        break;
                    default:
                        sSearchType = “Title”;
                        break;
                }
            }

            private function progressSaveIndicator():void {
                var myTimer:Timer = new Timer(2000, 0);
                myTimer.addEventListener(“timer”, onTimer);
                myTimer.start();
                lblProgress.setStyle(“color”, “green”);
                lblProgress.setStyle(“fontWeight”, “bold”);
                lblProgress.text = “Saving”;
            }

            private function progressUpdateIndicator():void {
                var myTimer:Timer = new Timer(2000, 0);
                myTimer.addEventListener(“timer”, onTimer);
                myTimer.start();
                lblProgress.setStyle(“color”, “green”);
                lblProgress.setStyle(“fontWeight”, “bold”);
                lblProgress.text = “Updating”;
            }

            private function onTimer(event:TimerEvent):void {
                lblProgress.text = “”;
            }
```

Download the Code...
Go to http://coldfusion.sys-con.com

ColdFusion Hosting is our Complete Focus

POWERFUL HOSTING PLANS

FREE SQL server access | FREE account setup | Unlimited email accounts Generous disk space & data transfer
30 day money-back guarantee | Great value

RELIABLE NETWORK

99.99% average uptime | State-of-the-art data center with complete redundancy in power, HVAC, fire suppression, bandwidth and security | 24/7 network monitoring

FANTASTIC SUPPORT SERVICES

24/7 suppport services | Knowledgeable phone support | We focus on your individual needs

CFDynamics

866.233.9626 › CFDYNAMICS.COM

For years we have been involved in the Cold Fusion community and have come to know what developers and project managers look for in a web host. The combination of our powerful hosting plans, reliable network, and fantastic suppport sets us apart from other hosts. ***Real service. Real satisfaction. Real value. Real support. Real Freedom.***

macromedia ALLIANCE PARTNER

BlueDragon Certified V.I.P. Hosting Partner