SYS-CON

GOOD 'OL CF' AND THE NEW FRONTIER PG 5

ColdFusion Developer's Journal

ColdFusionJournal.com

July 2006 Volume:8 Issue: 7

Presorted
Standard
US Postage
PAID
St. Croix Press

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

BALANCE

Designer/developer, front-end/back-end, clients/sanity. . .web development is a balance and we can help you maintain it. Join now and experience a wealth of training resources tailored to the tools you use every day.

www.communitymx.com

Visit www.communitymx.com/trial/ for your free 10 day trial.

COMMUNITY MX

For answers call us at 1-866-EDGEWEB
3 3 4 3 9 3 2

When calling your web host for support you want answers, not an annoying song stuck in your head from spending all day on hold. At Edgewebhosting.net, we'll answer your call in two rings or less. There's no annoying on-hold music, no recorded messages or confusing menu merry-go-rounds. And when you call, one of our qualified experts will have the answers you're looking for. Not that you'll need to call us often since our self-healing servers virtually eliminate the potential for problems and automatically resolve most CF, ASP, .NET, SQL, IIS and Linux problems in 60 seconds or less with no human interaction.

Sleep soundly, take a vacation, and be confident knowing your server will be housed in one of the most redundant self-owned datacenters in the world alongside some of the largest sites on the Internet today and kept online and operational by one of the most advanced teams of skilled Gurus, DBAs, Network and Systems Engineers.

By the Numbers:

- 2 Rings or less, live support
- 100 % Guarantee
- 99.999% Uptime
- 2.6 GBPS Redundant Internet Fiber Connectivity
- 1st Tier Carrier Neutral Facility
- 24 x 7 Emergency support
- 24 Hour Backup
- Type IV Redundant Datacenter

EDGE WEB HOSTING

For a new kind of easy listening,
talk to EdgeWebHosting.net

http://edgewebhosting.net

CFDJ Readers' Choice Award · MXDJ Readers' Choice Award
2003 - 2006

∘ Shared Hosting ∘ Managed Dedicated Servers ∘ Managed Colocation ∘ Semi-Private Servers
∘ ColdFusion ∘ BlueDragon ∘ ASP ∘ .NET ∘ .Linux ∘ .Java ∘ SQL Server ∘ .MySQL ∘ Self-Healing Servers

editorial

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

technical editor
Raymond Camden raymond@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
135 Chestnut Ridge Rd., Montvale, NJ 07645
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
135 Chestnut Ridge Rd., Montvale, NJ 07645

Good 'ol CF" and the New Frontier

By Simon Horwith

Let's face it, the web is a frequently changing landscape – more so now than it has been since its inception. When the term "Web 2.0" first set the industry on fire, I have to admit, I was not terribly excited.

The idea, after all, is based by and large on the idea of applications being, well, like the kinds of applications that Macromedia had already been talking about and showing us for about four years… since the release of Flash MX and the term "RIA" was coined. Still, I did think it was great that the industry as a whole was finally on the same page about the importance of delivering great user experiences and about how that will be achieved, though, as far as implementation technology goes, AJAX (not Flash) received all the industry attention.

Whether they want to spin it this way or not, the reality is that Flash 9 and the Flex 2 platform/product suite is Adobe's answer to AJAX. Though it'd be fair to say that Flex 2 competes with AJAX, it's also fair to say that the two technologies simply complement each other very well. Adobe has made it clear that they think AJAX is neat and that they are in no way trying to compete with it; and to compare Flex 2 with AJAX is like comparing apples with oranges… Flex 2 is an entirely different beast and is way out of the league of AJAX. I agree with this line of thought: Flex 2 offers significant performance benefits, a true rapid development environment, visual effects, widgets, and other features, and platform ambiguity… all out of the box. The same cannot be said for AJAX.

Getting back to discussing industry trends in general, there has been a recent trend in online software services and social networking applications. The MySpace phenomenon is the best known example of social networking online – users create their own page (their own 'space' on the Web) and via that page are able to interact with and link to their other MySpace 'friends'. Many other social networking sites sprang up – based on sharing photos, videos, etc., with "friends". In addition to social networking, web apps that are focused on user convenience and productivity have also been gaining in popularity. By "convenience" I mean that the software is centralized (on the Web) and accessible from anywhere, is free, and runs in a browser with nothing more to install. Examples of this type of software include Gmail and Google Calendar.

We are now on the cusp of realizing a whole new Web. The ability to create more intelligent, functional, and better looking Web applications (RIAs) using Flex and/or AJAX, coupled with the popularity of online social networking applications, means a new type of experience for people who use the Web… no, for people who use computers. For business users and developers, this includes not just socially aware applications but a whole new class of online collaboration services as well. Another category of Web applications radically changing are search engines.

– continued on page 43

About the Author
Simon Horwith is the editor-in-chief of Cold-Fusion 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. He has been using ColdFusion since version 1.5 and specializes in ColdFusion application architecture, including architecting applications that integrate with Java, Flash, Flex, and a myriad of other technologies. In addition to presenting at CFUGs and conferences around the world, he has also been a contributing author of several books and technical papers. You can read his blog at www.horwith.com.*
simon@horwith.com

CF_UNDERGROUND VIII PRE-MAX EVENT

```
<CF_BEER
  FROM = "[imported/domestic]"
  COLOR = "[light/dark]"
  FERMENT = "[lager/ale/bock]"
  FLAVOR = "[porter/stout]"
  AUTOREFILL = "[on/off]">
```

DATE: SUNDAY, OCTOBER 22, 2006

LOCATION: THE BEACH, LAS VEGAS

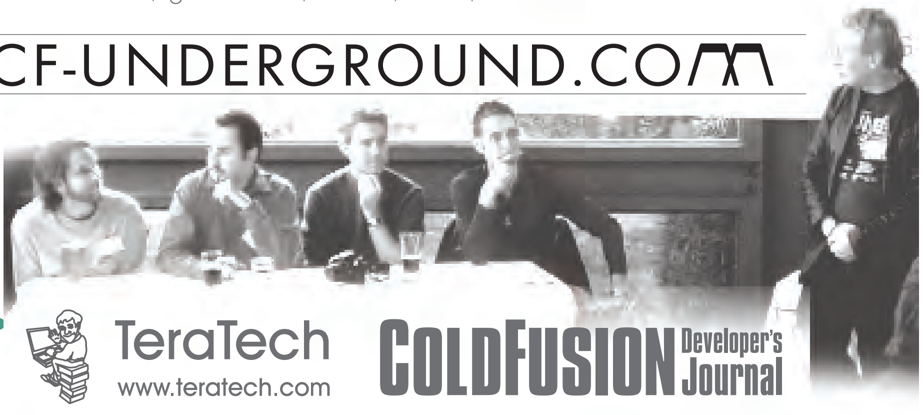
365 Convention Center Drive, Las Vegas, Nevada

Speakers include:
Simon Horwith
Michael Smith
Glenda Vigoreaux
and more...

Meet with old friends and new before the MAX event. A full day of networking, ColdFusion, great talks, drinks, food, and fun.

WWW.CF-UNDERGROUND.COM

only $99



TeraTech
www.teratech.com

COLDFUSION Developer's Journal

SYS-CON MEDIA

president & 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 director
Robyn Forma robyn@sys-con.com

advertising manager
Megan Mussa megan@sys-con.com

associate sales manager
Kerry Mealia kerry@sys-con.com
Lauren Orsi lauren@sys-con.com

sys-con events

president, events
Grisha Davida grisha@sys-con.com

national sales manager
Jim Hanchrow jimh@sys-con.com

customer relations

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

manager, jdj store
Brunilda Staropoli bruni@sys-con.com

sys-con.com

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

web designers
Stephen Kilmurray stephen@sys-con.com
Wayne Uffelman wayne@sys-con.com

online editor
Roger Strukhoff roger@sys-con.com

accounting

financial analyst
Joan LaRose joan@sys-con.com

accounts payable
Betty White betty@sys-con.com

accounts receivable
Gail Naples gailn@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

Working with Web Services

Flex class introspection to gain strong typing

By Jon Hirschi

Recently I worked on a project using Flex to create a front-end for Java-based Web Services. I ran into a favorite in the Java world: Transfer Objects/Value Objects. Early in the project, I was a bit skeptical of the transfer object model but, later on, I came to respect the value that it offers.

Transfer objects let two separate systems trade descriptive data objects without fear that either system will mistype the data or change the variable names. Transfer objects are an explicit agreement on how data will be encapsulated. They make integration easier because everyone can separate and build their system knowing what variable names and types will be traded. When someone modifies his version of the transfer object, the compile process should catch the changes and alert the file locations to fix the problems.

Flex automatically does the work of transferring a class type when using remote objects (one of the many reasons to use remote objects). However, when using SOAP Web Services, strong typing is broken because the class associations aren't automatically transferred to internal Flex classes. It's unfortunate, and leaves a bit of a gap when trying to use strong typing and compile-time error checking.

What's the value of strong typing? It makes it easier to debug the application. Instead of complaining at some obscure point when a user clicks on a certain button in the middle of a complicated workflow, it breaks during the compile process where the problem can be easily found and fixed. What could have been a two- hour search for a misnamed variable becomes a five-minute change.

The problem here though is how to get the untyped SOAP objects created by Flex into typed objects that Flex knows about – internal implementations of the transfer objects. The values themselves are typed, but there's no association between this data and the transfer object. Further, Flex can't be told that x SOAP object is actually of type y class. For example, trying to cast an incoming Web Service object into a user-defined TO class will cause a runtime error:

```
public var dpHistory:HistoryTo = event.result.Histo-
ryTo as HistoryTo;
```

or

```
public var dpHistory:HistoryTo = HistoryTo(event.
result.HistoryTo);
```

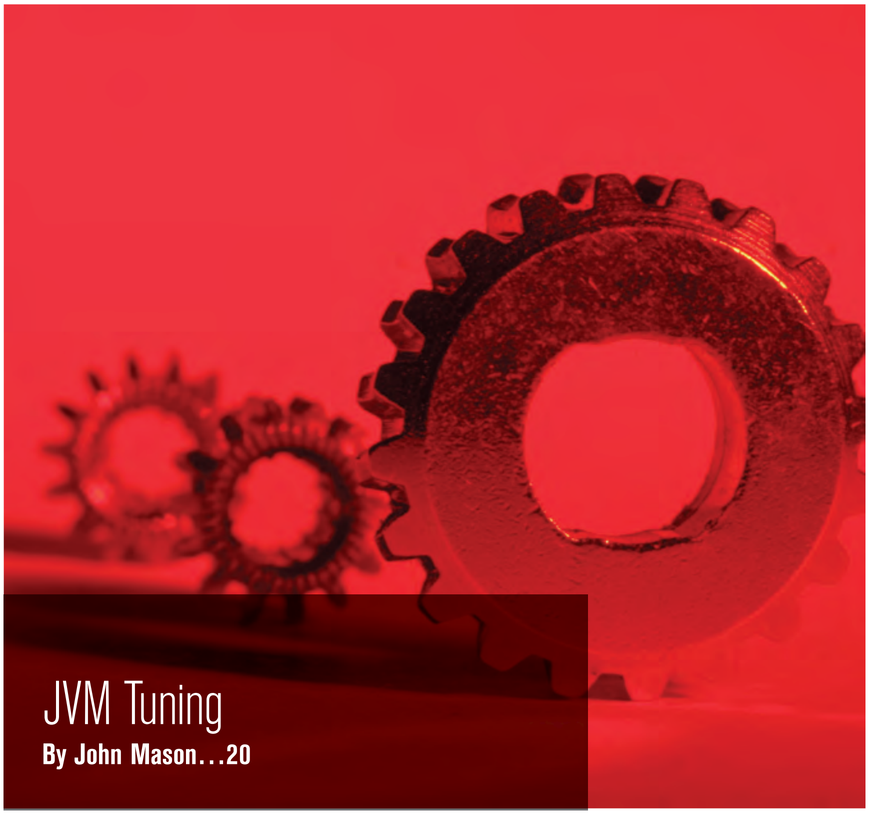
When I started trying to use strong typing of classes on the Flex side, I tried to do the transition by writing a custom transfer function for each type of class I had. For example:

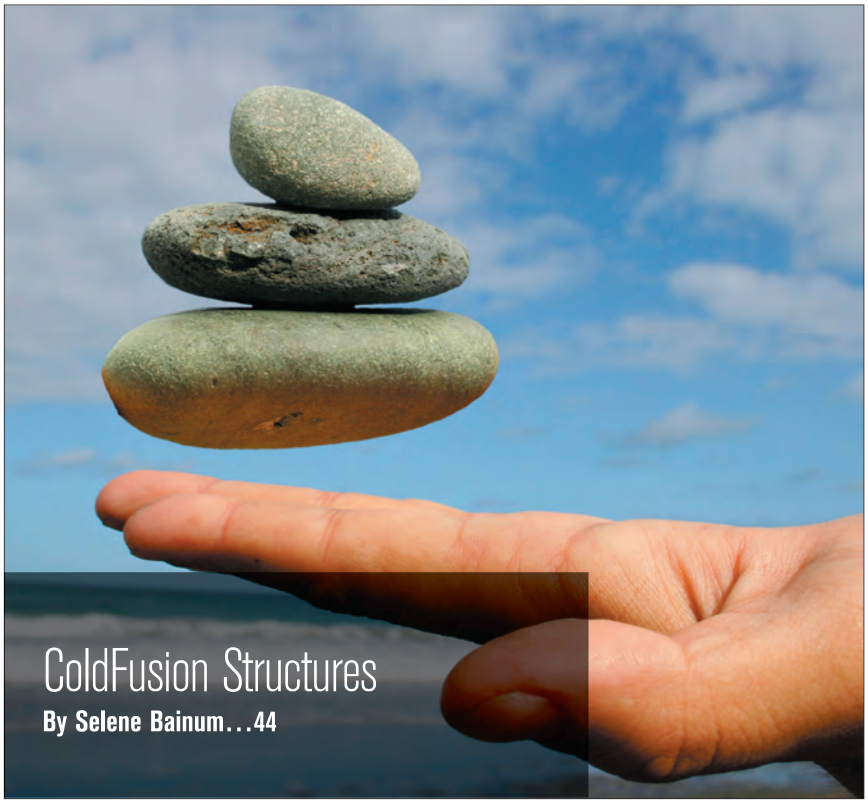
```
var dpHistory:ArrayCollection = new ArrayCollection;
var tempObj:HistoryTo;
For (var i=0;i<webServiceReturn.result.HistoryTos.
length; i++)        {
tempObj = new HistoryTo;
tempobj.message = webServiceReturn.result.History-
Tos.getItemAt(i).message;
tempobj.objectName = webServiceReturn.result.Histo-
ryTos.getItemAt(i).objectName;
…….
dpHistory.addItem(tempObj);
}
```

What a pain! It's error prone and doesn't allow for reuse. It's boring and adds to the amount of upkeep you have to do if anything changes. In the beginning, I avoided using transfer objects simply because that I didn't want to go through the pain of actually coding up every one of my Web Service returns like this. There are a lot of benefits to be gained by using transfer objects, but using the methodology above was horrible getting to them.

– continued on page 9

TABLE OF CONTENTS





– continued from page 7

To me, the drawbacks (i.e., less code reuse, more duplicate coding, etc.) appeared to outweigh the benefits. Sure, these functions can be written up and saved as an ActionScript file somewhere and included when needed, but it looked like one transfer function was needed per transfer object. And that can be a real pain, especially in a large project with numerous transfer objects.

I figured there had to be a better way. After hunting around a bit, I came up with a solution that worked for my project. All that was needed was a generic object transfer utility to translate the incoming objects into typed classes that are created on the Flex side. Once the transfer objects have been written on the Flex side, then class introspection can be used to dynamically populate the values from untyped Web Service objects. Let me show you how I did it.

First off, create a transfer object class. One of the beautiful things about Flex is the automatic getters and setters for public variables on classes. This means that creating a transfer object can be very simple. Flex is flexible enough to use explicit getters and setters as public properties or to take public properties/variables and create automatic getters and setters for them. Creating custom transfer objects then becomes a breeze and literally takes just a few minutes to put together.

For example, a simple class might be a history display class:

```
package comp.myProject.to       {
public class HistoryTo          {

           public var creationDate:Date = new Date;
           public var message:String = new String;
           public var objectName:String = new String;
           public var objectType:String = new String;

           public function HistoryTo()     {
           }

    }
}
```

Once the custom class (transfer object) has been created on the Flex side, it can be used to store variables. Flex Builder knows about this class and can display hinting on the class names, warn when a property name has been misspelled, and make sure that the properties and values maintain their type across the application. The biggest challenge is filling the class with the proper information.

With a plain object or dynamic class, getting the dynamic values contained inside is done by looping through its properties (dynamic properties are values added to a dynamic class at runtime, like adding variables to a plain object). For example:

```
    for (item in myObject) {
           // do something here
           myVar = myObject[item];
}
```

Unfortunately, classes don't support this method for finding static properties. This is where class introspection comes to the rescue. Class introspection returns a listing of all static properties of a class and the types of those properties. Best of all, class introspection is actually very easy in Flex. For example, there's a function in Flex that returns an XML structure of all the class properties:

```
    import flash.utils.describeType;

var classInfo:XML = describeType(HistoryTo);
```

The function describeType returns an XML structure that can be looped over to get a listing of the public static properties, class methods, class name, and class base reference (if it extends a class). The function describeType won't return dynamic properties, private properties, or private functions. In the example, the classInfo variable now contains an XML object that can be looped through and/or treated like any other XML-type data source. For instance, it could output the results in a tree (a la the old object inspector for Flex 1.5). In this case, it can be used to loop through the Flex-side transfer classes, pull out the variables and types, check them against the Web Service objects, and transfer the Web Service object values to the appropriate class variables.

The root node contains the name of the class and the name of the class that it extends (if any):

```
var className:String = classInfo.@name;
```

To get the list of public variables on the class is relatively simple as well.

The classInfo..variable contains an array of variable names and variable types. Looping through them allows for assignment from one variable to the next:

```
var vName:String;
var vType:String;

for each (var v:XML in classInfo..variable) {
    // set the xml name and type options to strings for easy comparison
vName = v.@Name;
    vType = v.@Type;
//do something here
}
```

With a list of the variables on the class, loop through and assign values to the class variables. Looping through the known classes lets us look into the Web Service object to see if the variable is there. If the variable exists, then the value is automatically transferred to the class. Because internal classes are static, an error is thrown if an attempt is made to set a variable that hasn't been written into the class at compile time.

```
if (translateFrom.hasOwnProperty(vName) && translateFrom[vName] != null)
{
// in the example, translateFrom is the incoming Web Service object or
untyped  object - check the incoming untyped object to see if it has the
property being looked for.
// do something here
}
```

From here, it's a matter of checking the type to transfer the variable accordingly. I've found some variable types from Java

YUM!
INTERSPEC
TACULAR

Adobe® Creative Suite 2

Say hello to the next generation.

It's found in the world's most inspirational places. It's pushing the boundaries of art direction and design. Adobe® Creative Suite 2. Today's most talented designers and art directors are working smarter and faster because of it. Just ask the masterminds behind INTERspectacular—they rely on Adobe CS2 to make their ideas tangible reality. See what Adobe Creative Suite 2 can do for you. It's everything but the idea. Better by Adobe.™

Luis Blanco and Michael Uman,
INTERspectacular

adobe.com/go/designer

Adobe

don't transfer over correctly. For example, longs sometimes don't transfer over as simple numbers and Booleans don't transfer over to a simple Boolean value. Sometimes they transfer over as a complexString value type that contains an XML snippet from the SOAP packet that Flex received. In most cases, when using the value in a complexString, it matches correctly and the fact that the object isn't strictly the type needed is barely noticeable. However, in certain cases, the difference becomes very noticeable. I've found that fixing it at the source helps eliminate problems down the road.

```
case "Number" :
translateTo[vName] = new Number(translateFrom[vName]);
    // translateTo is the class that is being populated
break;
```

This object translator can be extended to work with an array of objects as well. Once the class is inspected, you can get the name of the class. This lets you create a reference to that class and thereby dynamically create new instances of that class:

```
var classRef:Class = Class(getDefinitionByName(className));
```

Then when looping through the array of class objects create a new class object to store the data for each object in the array. From the example code, arrayOfObjects is the incoming array, classinfo is the describeType xml:

```
if (arrayOfObjects != null)      {
    for (i=0;i<arrayOfObjects.length;i++)   {
       tempObj = new classRef();
   returnArray.push(translateObject(arrayOfObjects[i],tempObj,classInfo
));
 }
}
```

Notice that most of the functions in the class are static functions:

```
public static function translateObject(translateFrom:Object,translateTo:
Object,classInfo:XML=null,upperCaseFromVar:Boolean=false):Object  {
```

Making the class functions static means that they can be used without creating an instance of the class. Static functions aren't tied to any particular instance of the class, but to all instances of that class. In effect it becomes like any other utility class that Flex uses. An example of the difference between static functions and instance functions is best illustrated by the Date class. When using the date class to store date information an instance has to be created then functions are available to modify or get the information stored in that date object. For example:

```
    var myDate:Date = new Date;
    var myMonth:Number = myDate.getMonth;
```

The function getMonth is only available on an instance of the Date class. It uses information stored in that instance to return its value. The Date class also has a static function:

```
Date.parse()
```

The parse function is a static function available on the class itself, not on a particular instance. Likewise, once the object translator class is imported into the application file, any of its functions can be used from the class itself:

```
import comp.ebay.utils.ObjectTranslator;

if (eventObj.result.HistoryTos != null )            {
dpHistory.source = ObjectTranslator.translateArrayObjects(eventObj.
result.HistoryTos.source,HistoryTo);

}
```

This returns an array of objects. However, there are times when you have to translate a single object. The function, translateObject, returns a typed object under the guise of an untyped object. This is done so that any kind of class can be translated. If a type of class were typed for return on the translator object, it would only be able to translate that specific class (which would be of little value). Returning an untyped object allows for flexibility in what this function gets used for. Now casting the returned object as a specific class or transfer object will work because intrinsically it IS the internal transfer class that Flex knows about. For example:

```
var historyItem = HistoryTo(ObjectTranslator.translateObject(event.
result,HistoryTo));
```

and

```
var historyItem = ObjectTranslator.translateObject(event.
result,HistoryTo) as HistoryTo;
```

There's one additional benefit for ColdFusion users that can be gained from a generic object translator. Pre-ColdFusion 7.02 variable names are all uppercased when sent through Web Services. In the example objectTranslator class (see Listing 1) some quick code has been added that checks for uppercased variable names on incoming objects, moving them into the variable names of the correct case.

In summary, once all Web Service objects have been translated to internal classes, strong typing can be maintained by casting any Web Service objects to the transfer classes. If variable names on the transfer objects have to change (which they often do), it's a much smaller job to catch all of these problems at compile time than it is to debug and find them at runtime. In a language like Flex that's strongly typed, using transfer objects makes for more efficient use of time and energy. Problems can be caught sooner and with less impact to users. Using class introspection offers the ability to look into a class and see the properties and methods inside the class. Those properties can be used to fill transfer objects and eliminate much duplicate code.

About the Author

Jon Hirschi is a senior Web engineer who works at eBay, Inc., creating next-generation RIA applications using Flex with Java and ColdFusion. He has worked with ColdFusion for the past seven years.

jhirschi@ebay.com

Listing 1

```
package com.ebay.utils     {
 import mx.collections.ArrayCollection;

 public class ObjectTranslator       {

    import mx.utils.ArrayUtil;
    import flash.utils.*;
    import mx.core.DeferredInstanceFromClass;


    public function ObjectTranslator()          {
       //TODO: implement function

    }

    public static function translateArrayObjects(arrayOfObjects:
Array,translateToClass:Class,upperCaseFromVar:Boolean=false):Array {
       // describeType describes a class and all of its properties
       var classInfo:XML = describeType(translateToClass);
       // get the class name, in order to instantiate more classes
       var className:String = classInfo.@name;
       var i:Number = new Number;
       var returnArray:Array = new Array;
       // get a reference to the class to create new classes
       var classRef:Class = Class(getDefinitionByName(className));
       var tempObj:Object = new classRef;

       if (arrayOfObjects != null)  {
          for (i=0;i<arrayOfObjects.length;i++)   {
             // for each new object, create a new class of the
class type that was passed in.
             tempObj = new classRef();
             // send it off to translateObject function. send it
the class to populate
             returnArray.
push(translateObject(arrayOfObjects[i],tempObj,classInfo,upperCase-
FromVar));
          }
       }
       return returnArray;  // return the translated array
    }


    public static function translateObject(translateFrom:
Object,translateTo:Object,classInfo:XML=null,upperCaseFromVar:
Boolean=false):Object  {
       var itemType:String;
       var itemName:String;
       var subClassInfo:XML;
       var tempArray:Array;
       var vType:String;
       var vName:String;
       var fromName:String;

       // if class information has already been sent in, we don't
need to re-fetch it.
       if (classInfo == null) {
          classInfo = describeType(translateTo);
       }
       for each (var v:XML in classInfo..variable) {
          try    {
             // xml values don't exactly match the variable name
unless it knows for sure it's a string
             vName = v.@name;
             vType = v.@type;
             if (upperCaseFromVar) {
                fromName = vName.toUpperCase();
             } else {
                fromName = vName;
             }
             // check to see if the untyped object has a property
that matches the typed property
             // if it does, then translate it.  if not, skip it.
             // this is done to maintain integrity with the appli-
cation and the internal classes.
             // internal classes are static (unless they've been
made dynamic) an error will result if
             // and unknown variable is set in it.
             if (translateFrom.hasOwnProperty(vName) &&
translateFrom[vName] != null) {
                // Only need to check for a few types here. vari-
ables should come across
                // in a limited number of base object types, ie
numbers, ints, dates, strings, etc.
                // anything else is assumed to be a class and will
be sent for translation as well.
                // the one case that is not dealt with here is when
an array of objects underneath
                // an array of objects.  This can be theoretically
set up, but there's no clear
                // an array of objects has to be a special type of
array, which in most cases only limits
                // the objects that can be store in it to a specific
class.  the simplest way around
                // this is to send back a simple array of untyped
objects which can then be translated
                // at the point that they are needed.

                switch (vType)  {
                   case "Array":
                      // in beta 3 SOAP sequences were typed as
arrays by FLEX.
                      if (translateFrom[vName] is ArrayCollec-
tion)   {
                         translateTo[vName] =
translateFrom[fromName].source;
                      } else if (translateFrom[fromName] is Ar-
ray)       {
                         translateTo[vName] =
translateFrom[fromName];
                      }
                      break;
                   case "mx.collections::ArrayCollection":
                      // in the GA release it looks like SOAP
sequences are
                      // typed as arrayCollections.  Support has
been left in for both
                      // types... just in case, so this function
can be used for both types.
                      if (translateFrom[fromName] is ArrayCollec-
tion)   {
```

```
                                translateTo[vName] =
translateFrom[fromName];
                            } else if (translateFrom[fromName] is Ar-
ray)        {
                                translateTo[vName].source =
translateFrom[fromName];
                            }
                            break;
                        case "Number" :
                            // sometimes numbers come across as a com-
plexString.  this can
                            // cause a mess at runtime if there is a
need to check against a
                            // simple number type.
                            translateTo[vName] = new Number(translateFr
om[fromName]);
                            break;
                        case "int" :
                            // ditto for ints
                            translateTo[vName] = new int(translateFrom[
fromName]);
                            break;
                        case "uint" :
                            // ditto for uints
                            translateTo[vName] = new uint(translateFrom
[fromName]);
                            break;
                        case "Date":
                            // ditto for dates.  what's nice about this
function is that with a little
                            // work, dates can automatically be trans-
fered from a UTC format or a string
                            // format (used sometimes for easy trans-
port)  into an actual date format
                            // which should be easier to use.
                            if (translateFrom[fromName] is Date) {
                                translateTo[vName] =
translateFrom[fromName];
                            } else if (translateFrom[fromName] is
String) {
                                translateTo[vName] = parseDateFromString
(translateFrom[fromName]);
                            } else if (translateFrom[fromName] is Number
|| translateFrom[fromName] is int || translateFrom[fromName] is uint)
{
                                translateTo[vName] = parseUtcDate(transl
ateFrom[fromName]);
                            }
                            break;
                        case "Boolean":
                            // Booleans get typed as a complexString
sometimes too.
                            translateTo[vName] = getBooleanValue(transl
ateFrom[fromName]);
                            break;
                        case "String":
                            translateTo[vName] = new String(translateFr
om[fromName]);
                            break;
                        case "Object":
                            translateTo[vName] =
translateFrom[fromName];
                            break;
                        default :
                            // assuming that anything that makes it here
is a sub class...
                            // send it to the this function again to
decode it...
                            translateTo[vName] = translateObject(transl
ateFrom[fromName],translateTo[vName]);
                            break;
                    }
                }
            } catch (e:Error)  {
                // do nothing // we will loose data, but our app will
stay in tact;
                //if (translateFrom.hasOwnProperty(v.@name))    {
                //  translateTo[v.@name] = translateFrom[v.@name];
                //}
                trace("an error occured in ObjectTranslator. The fol-
lowing fields were not captured correctly - to:" + vName + "  from:" +
fromName);
                trace(translateTo[vName]);
                trace(translateFrom[fromName]);
            }

        }
        return translateTo;
    }

    public static function translateArrayCollection(collection:Arra
yCollection,translateToClass:Class,upperCaseFromVar:Boolean=false):
ArrayCollection    {
        // this is a convenience function.  in the GA release of FLEX
2, objects are now coming back as
        // array collections.  if an array collection is expected,
this function will check to make sure
        // that it is not null.  it returns a completely new collec-
tion, which may or may not be what is desired.
        var returnCollection:ArrayCollection = new ArrayCollection;
        if (collection != null)   {
            if (collection.length > 0 )  {
                returnCollection.source = translateArrayObjects(collec
tion.source,translateToClass,upperCaseFromVar);
            }
        }
        return returnCollection;
    }

    public static function translateArrayCollectionAsArray(coll
ection:ArrayCollection,translateToClass:Class,upperCaseFromVar:
Boolean=false):Array      {
        // this is a convenience function.  in the GA release of FLEX
2, objects are now coming back as
        // array collections.  if an array collection is expected,
this function will check to make sure
        // that it is not null.  it returns a completely new Array,
which may or may not be what is desired
        var returnArray:Array = new Array;
        if (collection != null)   {
            if (collection.length > 0 )  {
                returnArray = translateArrayObjects(collection.
source,translateToClass,upperCaseFromVar);
            }
        }
        return returnArray;
    }

    public static function parseUtcDate(utcNumber:Number):Date{
        var returnDate:Date;
        // there should be a minimum length here for a valid utc Date
        if (utcNumber.toString().length > 6)  {
            returnDate = new Date;
            returnDate.setTime(utcNumber);
        }
        return returnDate;
    }

    public static function parseDateFromString(dateString:String):
Date    {
        var returnDate:Date;
        // check to see if we have a utc date as a string
        // guessing that dateString should be at least 6 chars long in
order to be
        // considered a viable date.... this could be a problem
though, because theoretically,
        // a time of zero should still be valid.
        if (dateString.length > 6 && parseInt(dateString).toString().
length > 6 )    {
            // we probaly have a utc date here
            returnDate = parseUtcDate(parseInt(dateString));
        } else {
            //  we didn't get a utc date so let's try to parse it...
            returnDate = parseUtcDate(Date.parse(dateString));
        }

        return returnDate;
    }

    public static function getBooleanValue(bValue:Object):Boolean
{
        var returnBoolean:Boolean = false;
        var stringBoolean:String = bValue.toString().toLowerCase();
        if (stringBoolean == "true" || stringBoolean == "yes" ||
stringBoolean == "1" || stringBoolean == "-1")   {
            returnBoolean = true;
        }
        return returnBoolean;
    }

    public static function clone(source:Object):*{
        var myBA:ByteArray = new ByteArray();
        myBA.writeObject(source);
        myBA.position = 0;
        return(myBA.readObject());
    }

  }
}
```

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

Efficient Web Content Management

CommonSpot™ Content Server is the ColdFusion developer's leading choice for efficient Web content management.

Our rich Web publishing framework empowers developers with out-of-the-box features such as template-driven pages, custom content objects, granular security, and powerful ColdFusion integration hooks (just to name a few), allowing you to rapidly build and efficiently deploy dynamic, high performance Web sites.

CommonSpot's open architecture and extensive APIs enable easy customization and integration of external applications. With CommonSpot, you can design and build exactly what you need. Best of all, CommonSpot puts content management in the hands of content owners. With non-technical users responsible for creating and managing Web content, developers are freed to focus on strategic application development.

Evaluate CommonSpot today.

To see your site *running* under CommonSpot, call us at 1.800.940.3087.

features.

- 100% browser-based
- Content object architecture
- Template driven pages
- 50+ standard elements
- Content reuse
- Content scheduling
- Personalization
- Flexible workflow
- Granular security
- Mac-based authoring
- 508 compliance
- Full CSS support
- Custom metadata
- Taxonomy module
- Extensible via ColdFusion
- Web Services content API
- Custom authentication
- Replication
- Static site generation
- Multilanguage support

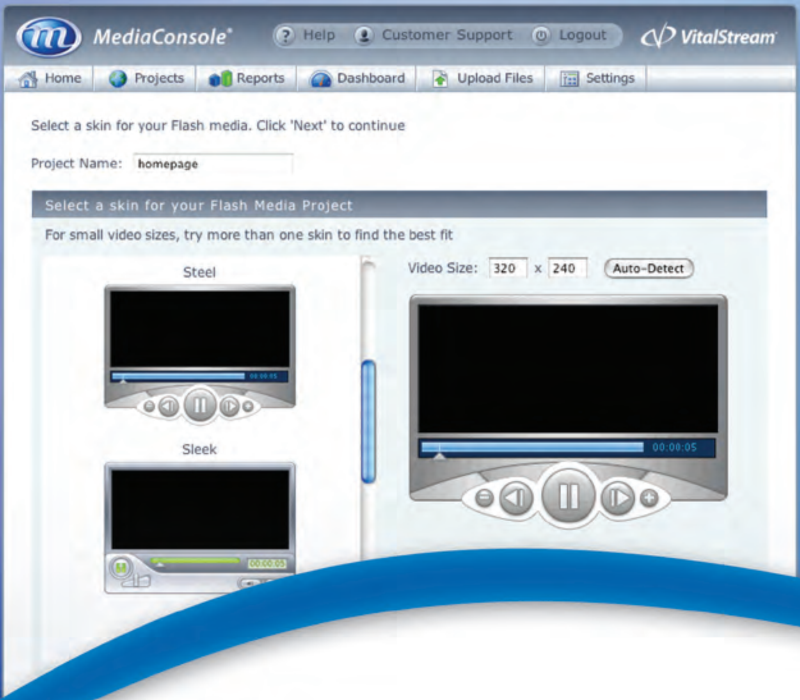
1.800.940.3087
www.paperthin.com

Paper Thin

Innovative Tools

To Manage, Deliver and Track Streaming Media On the Internet

MediaConsole



Select a skin for your Flash media. Click 'Next' to continue

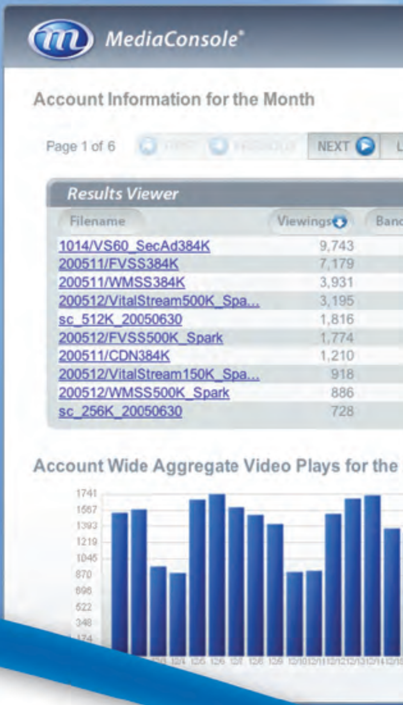
Project Name: homepage

Select a skin for your Flash Media Project

For small video sizes, try more than one skin to find the best fit

Video Size: 320 x 240

Real-time



Account Information for the Month

Page 1 of 6

Results Viewer

Filename	Viewings
1014/VS60_SecAd384K	9,743
200511/FVSS384K	7,179
200511/WMSS384K	3,931
200512/VitalStream500K_Spa...	3,195
sc_512K_20050630	1,816
200512/FVSS500K_Spark	1,774
200511/CDN384K	1,210
200512/VitalStream150K_Spa...	918
200512/WMSS500K_Spark	886
sc_256K_20050630	728

Account Wide Aggregate Video Plays for the

Using the Right Tools For the Job Is Critical to The Success of Your Business

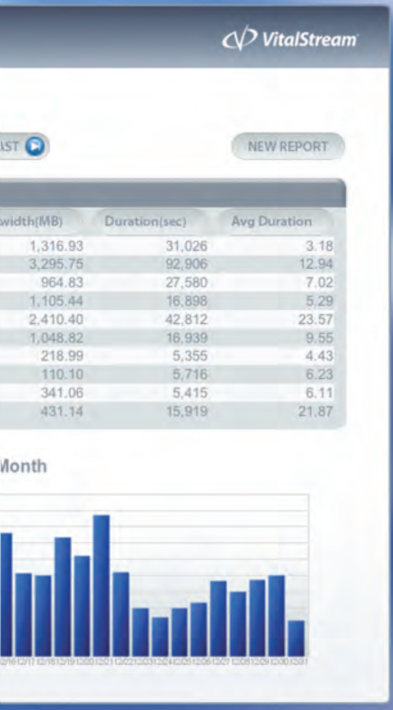
VitalStream reliably delivers streaming media to large global audiences using an award-winning content delivery network that powers all of our services. These services come with innovative tools that support and enable a variety of business models for online media distribution including advertising, subscriptions, pay-per-view and more.

Introducing the New VitalStream Media Player for Flash

Continuing its commitment to innovation, VitalStream is proud to announce the new Media Player for Flash to help speed up your workflow by allowing you to test and play streaming Flash video locally on your computer before you upload the files to our servers.

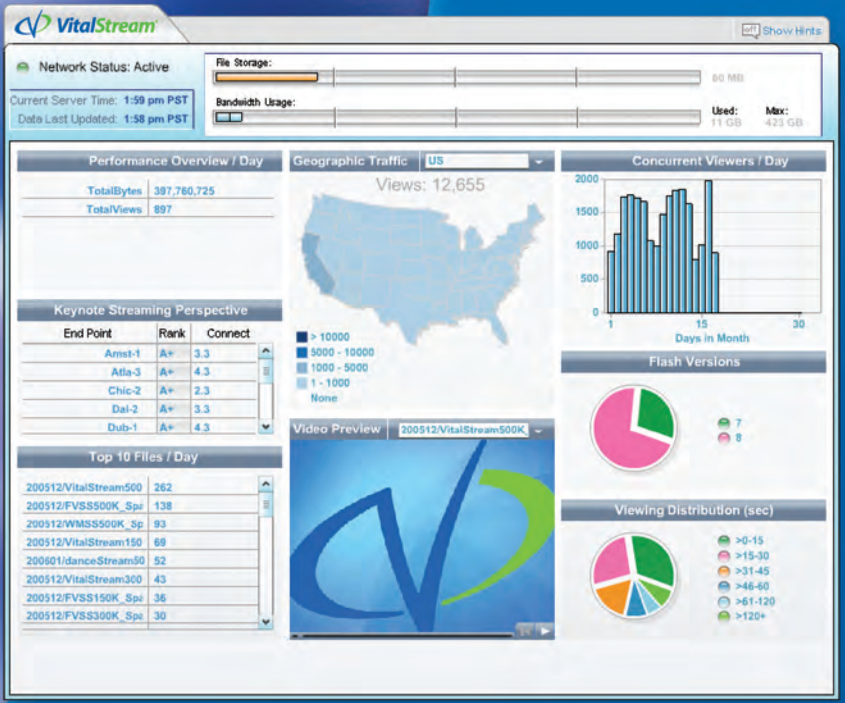
Download this player for free at http://www.vitalstream.com/tools/mxdj.asp

Reporting



width(MB)	Duration(sec)	Avg Duration
1,316.93	31,026	3.18
3,295.75	92,906	12.94
964.83	27,580	7.02
1,105.44	16,898	5.29
2,410.40	42,812	23.57
1,048.82	16,939	9.55
218.99	5,355	4.43
110.10	5,716	6.23
341.06	5,415	6.11
431.14	15,919	21.87

Reporting Dashboard



Network Status: Active

Current Server Time: 1:59 pm PST
Data Last Updated: 1:58 pm PST

Performance Overview / Day

TotalBytes	397,760,725
TotalViews	897

Keynote Streaming Perspective

End Point	Rank	Connect
Amst-1	A+	3.3
Atla-3	A+	4.3
Chic-2	A+	2.3
Dal-2	A+	3.3
Dub-1	A+	4.3

Top 10 Files / Day

200512/VitalStream500	262
200512/FVSS500K_Spa	138
200512/WMSS500K_Sp	93
200512/VitalStream150	69
200601/danceStream50	52
200512/VitalStream300	43
200512/FVSS150K_Spa	36
200512/FVSS300K_Spa	30

To learn more about VitalStream, call (800) 254-7554 or visit www.vitalstream.com

VitalStream®

The World Leader In Audio and Video Streaming

ColdFusion Can Save You Big Bucks

What happens when you discover that automation can cost so much

By Adedeji Olowe

How does a bank justify using over $200 million in shareholder funds to its shareholders? That's the million-dollar question that CEOs of banks in Nigeria were trying to answer in the aftermath of the forced capitalization imposed on them by the Central Bank of Nigeria (CBN).

In July 2004, the new CBN governor, Professor Charles Soludo, handed down a directive that every bank in Nigeria should increase its capital from about $16 million to $200 million in 18 months or risk losing its license. The options weren't any too pretty: merger and recapitalization. At the end of the intense bloodbath that followed, only 25 banks survived out of 89. But the reality that confronted the CEOs was that it was easier surviving the first recapitalization hurdle than justifying to shareholders the $200 million booty with bottom-line profits.

From a simple perspective, it should have been relatively easy. Since every bank is awash with cash, it's just a matter of creating a retail strategy, opening more branches, loaning money to multinationals, etc. But there was no simple perspective; it was way more difficult especially when every one of the 25 banks was targeting almost the same set of customers.

Just like its counterparts in every country of the world, each Nigerian bank runs a main banking application such as FlexCube, Finacle, Bankmaster, and Phoenix. The survival of these banks was (whether they liked it or not) directly linked to the capabilities of these applications, in addition to other things. The new mega-bank status of these banks exposed the relative weaknesses of some of these applications. Weaknesses that were relatively harmless when these banks were like mom-and-pop organizations were now threats to their very existence.

First City Monument Bank Plc (FCMB) was in this situation when it merged with three other banks. The application it used then, Equation, was slowing down its move to implement one of the best retail banking strategies the country had ever seen. Without skipping a beat, however, the management went for a new core banking application, Finacle, which has been judged to be one of the best in the world. Unfortunately, Finacle is pretty expensive to implement. For a newly minted mega-bank like FCMB, the implementation cost would carve massive hole in its the bottom-line profits.

Apart from the retail strategy and new banking application, FCMB was also trying for the Holy Grail of banking efficiency in Nigeria; the automation of all its processes. It didn't take a seer to see what they were trying to achieve. Lashing together a good strategy with a world-class application supported by a fully automated processes then justifying to shareholders at the next annual meeting the use of their $200 million equity should be a piece of cake.

Or so it seems until you discover that automation can cost so much. While considering the technologies that could be used to automate its processes, the bank discovered Infopool. Infopool is a Web-based ColdFusion integration-and-automation platform that had been successfully deployed in another Nigerian bank. Based on the efficacy of the application, FCMB realized that the cost of implementing Finacle could be significantly reduced by deploying Infopool.

Aside from the cost of the heavyduty IBM minicomputers needed to run the massive database and Finacle applications, the bulk of the cost of implementing the Finacle application is the license. Every user license runs into four digits. Multiply that by over a thousand user licenses and the total cost of acquiring the end-user licenses stops looking attractive. Meanwhile, most users would be using the application only for enquiries while the rest, the operations staff, would use it to post financial transactions.

Consequently it was decided that the automation process should also integrate the Finacle database so that all non-operation staff could make enquiries on Infopool.

The integration team swung into action and was able to complete the enquiry modules on Infopool the day before Finacle was officially launched. The total cost of implementing the automation, which included the enquiry modules and other workflow portions, was less than the cost of the 10-user Finacle licenses. Using ColdFusion, Infopool was able to achieve a single sign-on with the Microsoft Active Directory running the bank's network. As a result, the user only needs to log in once to access different applications based on the profile configured by the systems control department.

Of all the recent technology implementations in the industry, Infopool has the best return on investment (ROI) and the lowest total cost of ownership (TCO). It also lets management, using the management information dashboard, make intelligent decisions, quickly and accurately.

None of the things done with ColdFusion is a novelty or a ground-breaking technique. But as far as the management of FCMB is concerned, ColdFusion saved the day. For the bank, over a million dollars was saved in direct cost. It's not the technology that's important, but the fact that ColdFusion has been immensely instrumental in rapidly deploying a user-friendly platform with rapid access to customer information and automated processes. The bank has been able to live up to its promise to customers of superior customer care. It's been able to provide cutting-edge services to its old and new customers, which is actually the crux of its retail strategy. Happy customers mean more revenue. More revenue, all things being equal, translates to bottom-line profit.

This is exactly what First City Monument Bank Plc needed to show its shareholders. All made possible by the power of ColdFusion.

About the Author

Adedeji Olowe, a Web application developer at First City Monument Bank Plc in Lagos Nigeria, has been using ColdFusion for several years. He has experience developing and extending enterprise applications for companies in the financial industry as well as in Active Directory integration.

adedeji@olowe.com

Other companies in this magazine spent a lot of time on pretty ads. As you can see, we did not. We spent our time hiring the best people and training them to deliver outstanding support for your website. We spent our time building a state of the art datacenter and staffing it with people who care about your website like it's their own. Compassion, respect, credibility, ownership, reliability, "never say no," and exceed expectations are words that describe our service philosophy. From the first time you interact with us, you'll see what a difference it really makes. And you'll also forgive us for not having a pretty ad.

HostMySite.com

WEB HOSTING • MANAGED DEDICATED SERVERS • COLOCATION • VPS • ECOMMERCE • BLOGGING • EMAIL

JVM Tuning

The application, machine, and JVM version make a difference

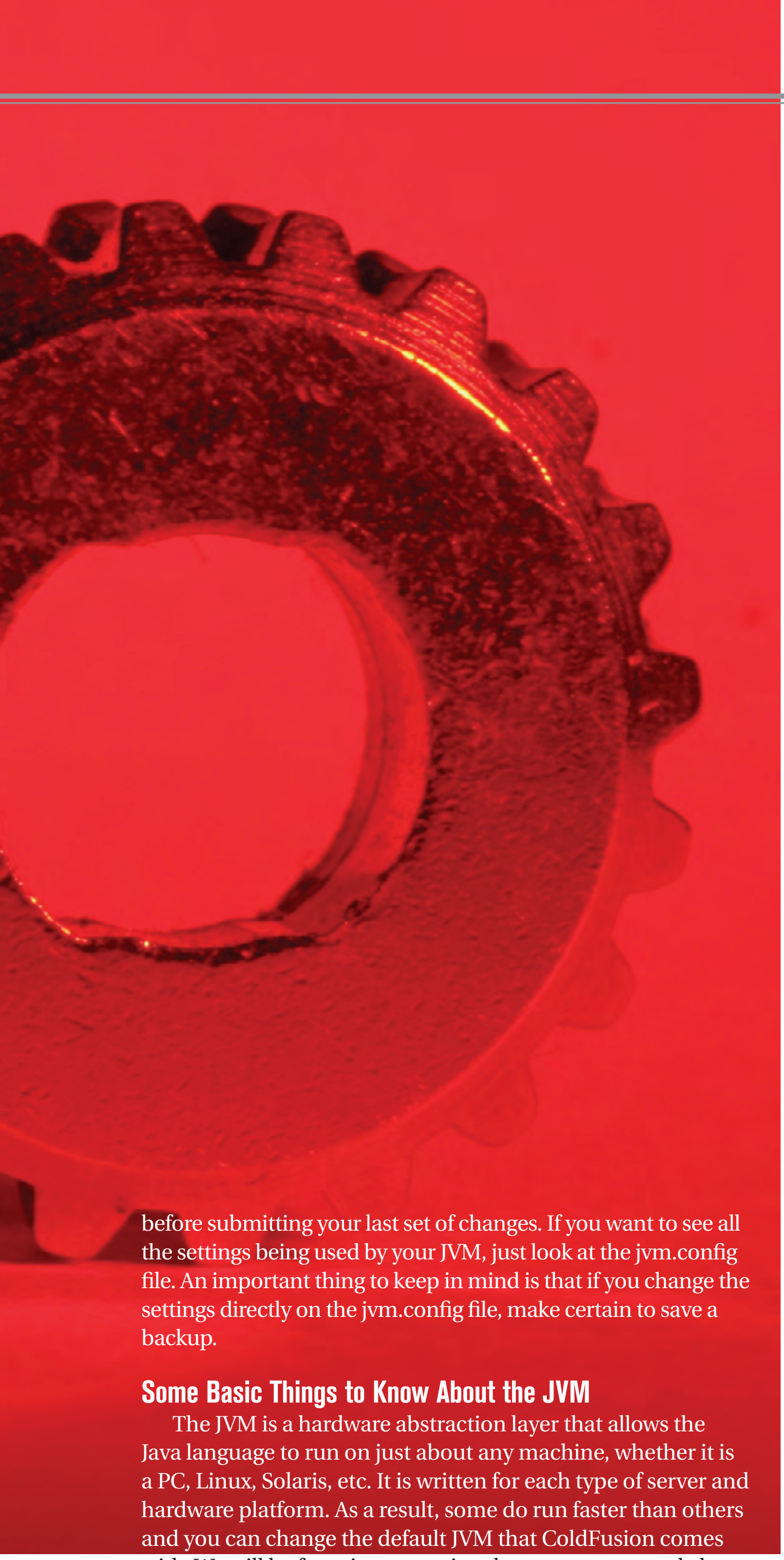
By John Mason

As ColdFusion programmers or system administrators, there are times when we go through the CF Admin interface to try and optimize the server. A particular section located within the standalone version of the ColdFusion Administrator is simply called Java and JVM. When you reach this section, unless you have a healthy dose of Java experience, you may scratch your head and think, “I don’t really know much about this” and skip to something else. But the catch is, this section is the most powerful and important area to look at.

It is common knowledge that Adobe’s ColdFusion runs on Java, but Java runs on top of the Java Virtual Machine (commonly called the JVM or VM). The JVM is what allows Java to speak to the physical machine and controls processing and memory allocation. Because of this, changes to the JVM settings can have a huge impact on how your ColdFusion applications run. If you have already optimized your ColdFusion code and settings: scoping variables, caching, etc., it may be time to take a closer look at your JVM settings.

Java and JVM Administrator Section

If you are running the standalone version of ColdFusion MX, the Java and JVM section of your ColdFusion Administrator is a very simplified admin with which to view your JVM settings. If you are using the J2EE configuration, this feature is not listed, but you can still access the configuration file directly. Several important settings are shown within the CF admin, including the important ones like the Heap size, Permanent size and certain other JVM arguments. As you change or modify these settings, the admin will modify the jvm.config file, which is usually located at either the “CFusionMX7\runtime\bin” or “JRUN4\bin directories, depending on your configuration. It will also save a jvm.bak file, a backup file of the previous settings you had



before submitting your last set of changes. If you want to see all the settings being used by your JVM, just look at the jvm.config file. An important thing to keep in mind is that if you change the settings directly on the jvm.config file, make certain to save a backup.

Some Basic Things to Know About the JVM

The JVM is a hardware abstraction layer that allows the Java language to run on just about any machine, whether it is a PC, Linux, Solaris, etc. It is written for each type of server and hardware platform. As a result, some do run faster than others and you can change the default JVM that ColdFusion comes with. We will be focusing on tuning the one you currently have installed.

Permanent Space

The JVM must have memory in order to function. One part of the machine's memory is set for the JVM to reside in. This is called the Permanent space. The ColdFusion application server runs in this space as well. Some people think the Permanent space is a part of the heap space. It really is not and has its own distinct purpose. The Permanent space holds items like the class objects that ColdFusion creates during compilation and runtime of templates.

The Heap Space

The memory used to store items such as variables is called the Heap space. The Heap space is actually divided up into smaller spaces called "generations." There are several generations within the heap, but the few general areas we are most concerned with are called:

- Young (New) generation space
- which contains
  a) Eden space
  b) Survivor 0 space
  c) Survivor 1 space
- Tenure (Old) generation space

The specific generation an object falls into is dependent on its age. Any "new born" memory objects that need to be placed into memory are always created in the Eden space first. When Eden fills up, the memory items that are still referenced or "alive" move to Survivor 0 and Eden is cleared out of any old objects that are no longer required by the JVM. This provides room for new incoming objects. When the Survivor 0 space fills up, once again the objects that are still "alive" are moved to Survivor 1 space and Survivor 0 is cleared out. If an object is needed for an extended period of time, it will move into the Tenure space and stay there until it dies. So, all objects in the JVM memory start off in the Eden space and will either die at some point or end up in the Tenure space. Naturally, you will want to see request-scope variables stay in the Eden space, session-scope variables in one of the Survivor spaces, and application-scope variables in the Tenured space.

What Is Garbage Collection?

The memory spaces are controlled by the JVM through what is called the "Garbage Collector" or "GC" for short. While it can sound relatively simplistic, Garbage collection is actually a very complex subject.

The garbage collector moves things around in the memory heap space and decides whether certain items need to be kept or discarded. If there were no garbage collection, the JVM would run out of memory within a very short period of time. The garbage collector either kills objects or moves them into an older generation space. It also changes the sizes of the spaces depending on your JVM settings and its needs. These are minor garbage collections and are needed for the JVM to run properly. But, as you might expect, over time the Tenure space might fill up. When this happens, the system will run a full garbage collection and try to free up space. This is very intensive, and you don't want to see the system doing this full GC very often. If, for some reason when it runs a full GC and it can't free up the memory, the JVM will crash, taking ColdFusion down with it. This is when you might start noticing java.lang.OutOfMemoryError errors appearing in your exception logs. On a browser, you might see a 500 error saying simply "Java heap space."

If you are wondering why the JVM has so many memory spaces, it is because of the garbage collector. Early JVM versions had collectors that had to scan the entire memory space to do a

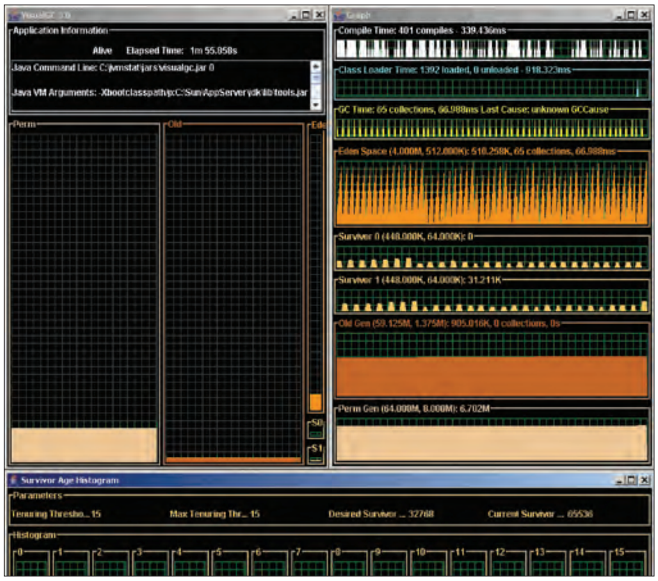

VisualGC 3.0
Application Information
Alive Elapsed Time: 1m 55.858s
Java Command Line: C:\jvmstat\jars\visualgc.jar 0
Java VM Arguments: -Xbootclasspath/p:C:\Sun\AppServer\jdk\lib\tools.jar
Perm
Old
Eden
S0
S1
Graph
Compile Time: 401 compiles - 339.436ms
Class Loader Time: 1392 loaded, 0 unloaded - 918.323ms
GC Time: 65 collections, 66.988ms Last Cause: unknown GCCause
Eden Space (4.000M, 512.000K): 510.258K, 65 collections, 66.988ms
Survivor 0 (448.000K, 64.000K): 0
Survivor 1 (448.000K, 64.000K): 31.211K
Old Gen (59.125M, 1.375M): 905.016K, 0 collections, 0s
Perm Gen (64.000M, 8.000M): 6.702M
Survivor Age Histogram
Parameters
Tenuring Thresho... 15
Max Tenuring Thr... 15
Desired Survivor ... 32768
Current Survivor ... 65536
Histogram
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15

Figure 1:

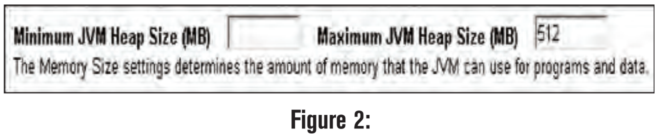
garbage collection. This took up time and resources. By dividing up the space, the collector can run fast minor GCs and do full GCs only when it absolutely needs to. Under this setup, even the full GCs run faster. So it helps boost the performance.

When the garbage collector is at work, just about everything within the JVM is put on hold. Naturally, the JVM can't have an application thread writing to a memory space that is being deleted by the collector. So the threads have to wait whenever the collector is in process. This is important because the collector could take a few milliseconds doing a minor GC or it could take a lot longer when doing a full GC to clear up the Tenured space. The speed and number of collectors depends a great deal on processor speed, heap size, and the type of collector you are using. Therefore, you need to see what your garbage collector is doing.

How Do You Determine Your Current Heap Size Usage and Garbage Collection Time?

How do you determine how much memory the JVM is currently using or how well the garbage collector is running? There are several different ways, but one of the easiest is to use is the JVMStat and Visual Garbage Collector Tools from Sun (http://java.sun.com/performance/jvmstat/). If you use a JVM that isn't from Sun, it should have its own statistic tool that you can use. If you are trying to optimize your ColdFusion server, I would highly recommend using these tools. They will provide you with the basic information you need to make the right setting changes to your JVM.

Once you have installed the tools on your machine, you'll be able to pull up a graphical representation of the heap space,



Minimum JVM Heap Size (MB) [ ] **Maximum JVM Heap Size (MB)** 512

The Memory Size settings determines the amount of memory that the JVM can use for programs and data.

Figure 2:

permanent space and garbage collector (see Figure 1).

This provides the most detail view of your JVM settings.

You now have some basic variables to look at. A few other things to take note of are the number and type of processors that are on your server. Also, consider the server's physical memory and its current memory usage. Measure these when the server is under load so you can see the changes in items like memory usage. Microsoft provides a simple and free load tester called "Microsoft Web Application Stress Tool." There is also OpenSTA. Use this to apply enough load to send your CPU usage to 100%. At this point, note the heap size, the amount of the heap being used, the physical memory usage, and the number of full GCs. If the Tenured space is filling up quickly and the garbage collector is doing a full GC every few minutes, you have a problem. You don't want the Tenured space to fill up frequently. If this happens more than once an hour, you will most likely need to adjust your JVM.

How to Make the JVM Setting Changes

Changing the Heap size is very easy through the ColdFusion administrator (see Figure 2).

The default settings are shown in Figure 2. There is a strong argument to set the Minimum and Maximum settings to the same size. The reason is that it will keep the JVM from having to calculate this for itself, thereby allowing the JVM to focus on more important things. But when you do this, more attention will need to be given to the JVM to see if any java.lang.OutOfMemoryError errors appear in your exception logs.

How high can you increase the Heap size? This depends on the JVM version and hardware being used. You will need to do some research here. Most recent JVMs on a 32-bit system usually have a limit near 2GB. Sixty-four bit systems can run much higher. If your server has only 2GB of memory, it would be safe to only go up to 1GB for the JVM. Remember, your other applications need memory too.

You can actually lower your server's performance by setting your Heap size too high. If, for example, your JVM never exceeds 100MB of memory, and yet you have the Heap size set to 1024MB, you are asking the garbage collector to manage a bigger memory space than it really needs to. Look at your Tenured space and see how often it fills up. If it rarely seems to fill up, there is no reason to increase your Heap size. If you lower your Heap size, pay attention to your exception logs for java.lang.OutOfMemoryError errors because this will serve as a warning that the JVM is running out of memory.

The Permanent space size can also be controlled. This setting can be found in the JVM arguments section of the CF Admin. In the arguments, you should see something like this, "-XX:MaxPermSize=128m". This argument simply tells the JVM that the maximum allocation of memory for the Permanent space is 128MB. Depending on what your ColdFusion application server is doing, it may be necessary to move this setting up to 192MB or 256MB. If your server is creating a lot of class objects, then it may be necessary to move this limit up.

The garbage collector can also be changed and tuned. The default garbage collector used by ColdFusion 6 and 7 is called the Parallel Young Generation collector (or Parallel Scavenging), which is a part of the family of Throughput collectors. You will need to go directly to the jvm.config file to see its setting.

Under the JVM arguments, you should see the following, "-XX:+UseParallelGC". There are actually several others to choose from. The two major families are Low Pause Collectors and Throughput Collectors. The differences are in how they treat the balance between GC pauses versus processor performance. The Low Pause family of collectors will allow applications to run during part of the GC process. It does this by not moving "live" objects when it's removing dead objects from the Tenure space. This, however, costs the collector in performance since it has more complexity to its task. So Low Pause collectors give you smaller GC pauses but with more processor time than with the Parallel collectors. Throughput collectors have longer GC pauses but greater CPU throughput because the application threads are not sharing the processor time with the GC. Typically, applications with large real-time data prefer the Low Pause collectors, but you will need to see if you have the available processor resources to make this type of switch.

In summary, if you feel that the default collector is not very efficient, research the others available and test to see if they can help your system. Note, other JVMs sometimes carry with them other collectors. For example, the JRockit JVM from BEA has the Dynamic Garbage Collector, which is not available with Sun.

Finally, How and When to Optimized the JVM

In most cases, you do not want to see the Tenure space fill up very often because when it does, the JVM has to run a full garbage collection, which takes up time and resources. Plus, you run the risk of the JVM not being able to flush out the memory in time and crashing your ColdFusion server. So adjust your JVM heap size to fit the needs of your system. If you want to switch out the garbage collector, I would advise you to do plenty of research. Sun has a lot of documentation on this subject regarding this, and O'Reilly has an excellent book called Java Performance Tuning by Jack Shirazi. As always, test any setting changes on a development server first before applying to a production environment.

As you can see, the JVM settings are extremely powerful and can do a lot to improve your ColdFusion application performance. My hope is that this article encourages you to learn as much as you can about the JVM and its settings. If you are interested in learning more about this and our research, please visit http://labs.fusionlink.com for more information.

About the Author

John Mason works at FusionLink, which specializes in ColdFusion hosting services. He is a Certified Advanced ColdFusion developer and has been working with ColdFusion since version 2.

mason@fusionlink.com

Get more with Hot Banana!

Web Content Management + Active Marketing

Hot Banana software builds efficient, affordable, fast, very easy-to-use and measurable Web sites. And they're ColdFusion-based and optimized for marketing performance. That's what makes us different and better!

Call us today at 1-866-296-1803 to schedule a live 1-on-1 demo, or visit hotbanana.com/cfdj to learn more.

hotbanana®

"I was totally intimidated by Java, but I knew I had to learn it. Your class taught me what I honestly thought I couldn't be taught." - Sharon T

Java for ColdFusion Programmers?

Java for ColdFusion Programmers, the five-day Hal Helms, Inc. training class is designed for ColdFusion programmers; no previous Java experience is needed. You'll learn how to think in objects and program in Java.

For class information and registration, come to halhelms.com.

My First Flex Application

For this article I used Flex 2 beta3. Since then Flex 2 has been released so you should download that.

By Mary McDonald

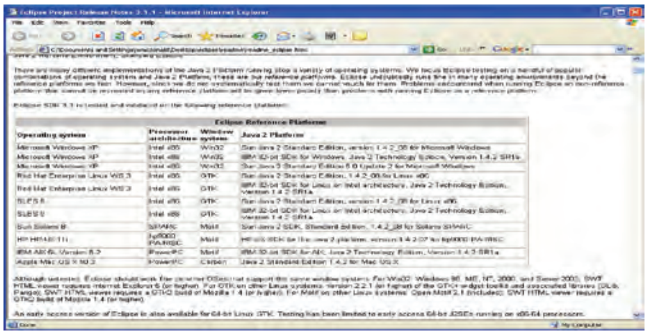
We'll walk through the following steps to write our first Flex application.

1. Install the free Eclipse integrated development environment
2. Install the Flex Builder 3 plug-in for Eclipse
3. Write the application
4. Test/debug the application

When you set out to write your first Flex application you'll have to choose what tool you'll use to write it with. Currently the most popular tool to use is the free Eclipse IDE. If you don't have Eclipse installed go to http://www.eclipse.org/ to install it. You'll need to install Eclipse 3.1 to use Flex 2 Beta 3.
Select the Downloads link

Select the Eclipse SDK 3.1.2 for Windows to download, if you have Windows.

Eclipse can be used with many operating systems. I'll be using Windows XP with the Sun Java 2 Standard Edition Platform version 1.4.2_08 for Microsoft Windows. See the Eclipse readme_eclipse.html for more information.

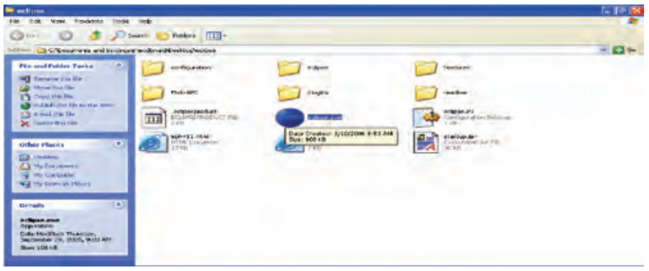


Eclipse Reference Platforms

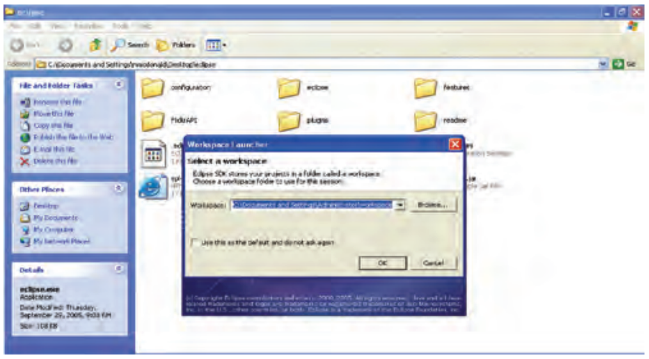
Operating system	Processor architecture	Window system	Java 2 Platform
Microsoft Windows XP	Intel x86	Win32	Sun Java 2 Standard Edition, version 1.4.2_08 for Microsoft Windows
Microsoft Windows XP	Intel x86	Win32	IBM 32-bit SDK for Windows, Java 2 Technology Edition, Version 1.4.2 SR1a
Microsoft Windows XP	Intel x86	Win32	Sun Java 2 Standard Edition 5.0 Update 2 for Microsoft Windows
Red Hat Enterprise Linux WS 3	Intel x86	GTK	Sun Java 2 Standard Edition, 1.4.2_08 for Linux x86
Red Hat Enterprise Linux WS 3	Intel x86	GTK	IBM 32-bit SDK for Linux on Intel architecture, Java 2 Technology Edition, Version 1.4.2 SR1a
SLES 8	Intel x86	GTK	Sun Java 2 Standard Edition, version 1.4.2_08 for Linux x86
SLES 9	Intel x86	GTK	IBM 32-bit SDK for Linux on Intel architecture, Java 2 Technology Edition, Version 1.4.2 SR1a
Sun Solaris 8	SPARC	Motif	Sun Java 2 SDK, Standard Edition, 1.4.2_08 for Solaris SPARC
HP HP-UX 11i	hp9000 PA-RISC	Motif	HP-UX SDK for the Java 2 platform, version 1.4.2.07 for hp9000 PA-RISC
IBM AIX 5L Version 5.2	PowerPC	Motif	IBM 32-bit SDK for AIX, Java 2 Technology Edition, Version 1.4.2 SR1a
Apple Mac OS X 10.3	PowerPC	Carbon	Java 2 Standard Edition 1.4.2 for Mac OS X

There are various plug-ins available for Eclipse. There's a Flex plug-in we'll install so we can use Eclipse to write our Flex app. The Eclipse plug-in for Flex is included in the Flex download. Go to http://labs.adobe.com/ to download the Flex 2.0 Beta 3.

Open the Eclipse IDE by clicking the Eclipse icon.

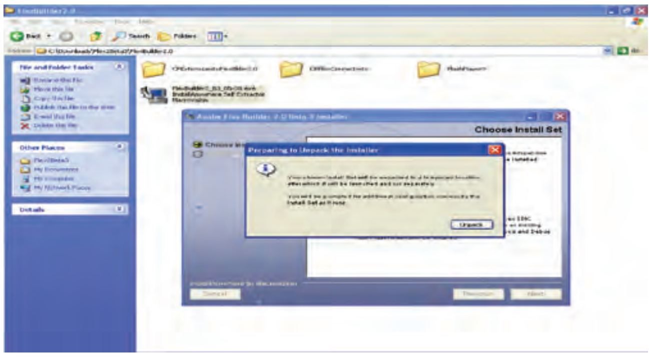


Eclipse will ask you where you want your workspace to be located. Just leave it in the default directory for now. Press OK.

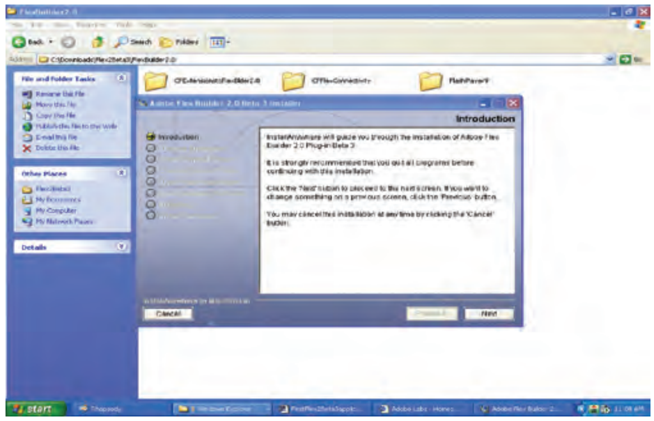


We have to install the Flex Plug-in into Eclipse. I downloaded the Flex Builder 2 Beta 3 and selected the install option.

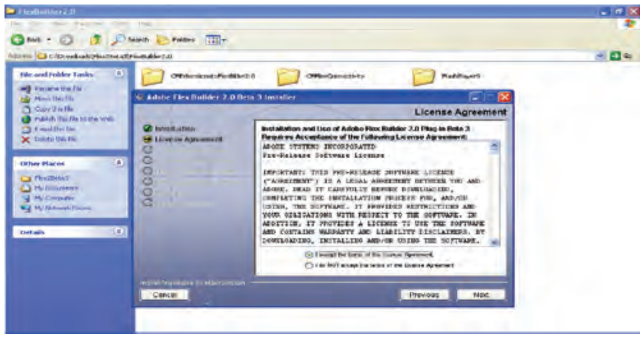
I'll choose the Flex Builder plug-in and Flex SDK option to install Flex Builder as an Eclipse plug-in.



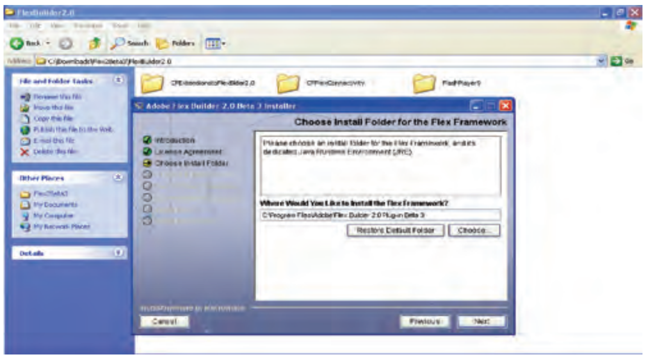
Press the Unpack button to unpack to a temporary installation area.



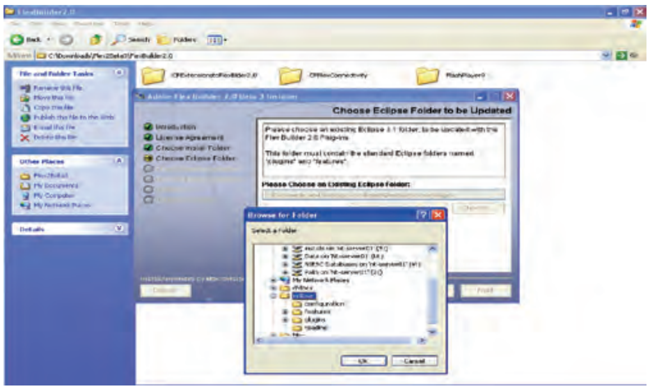
Press the Next button.



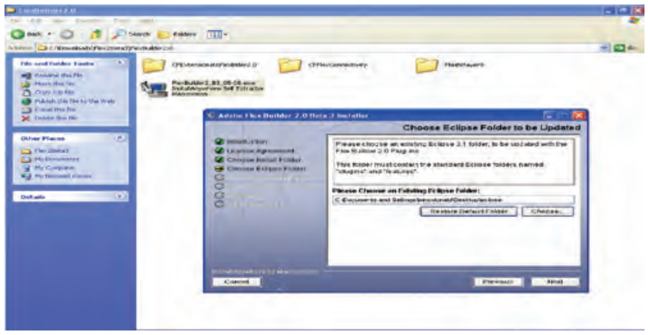
Accept the license.


Adobe Flex Builder 2.0 Beta 3 Installer
Choose Install Folder for the Flex Framework
Where Would You Like to Install the Flex Framework?
Restore Default Folder
Choose...
Cancel
Previous
Next

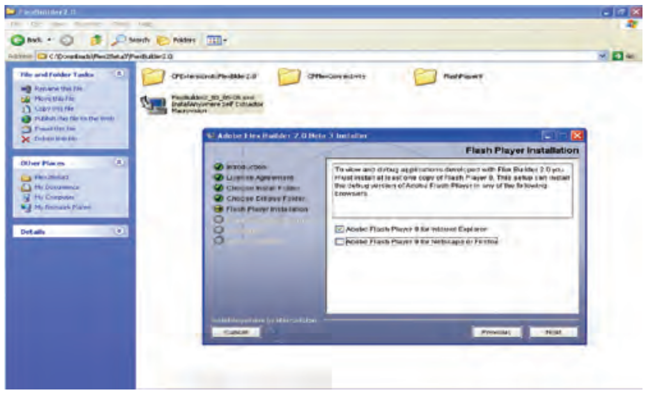
Accept the default directory to install the Flex Framework.


Adobe Flex Builder 2.0 Beta 3 Installer
Choose Eclipse Folder to be Updated
Please Choose an Existing Eclipse Folder:
Browse for Folder
OK
Cancel

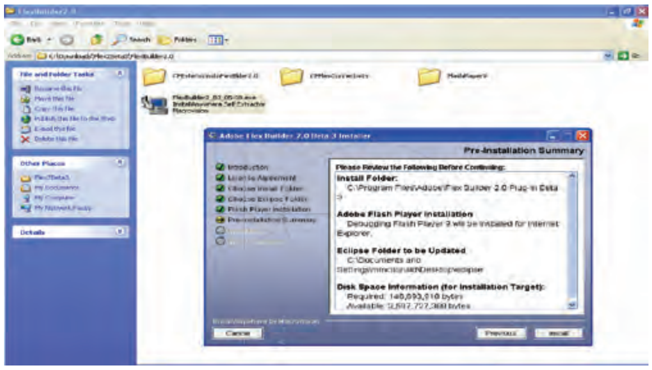
The My Eclipse folder is in my desktop, so that's where I installed the Flex plug-in.


Adobe Flex Builder 2.0 Beta 3 Installer
Choose Eclipse Folder to be Updated
Please Choose an Existing Eclipse Folder:
Restore Default Folder
Choose...
Cancel
Previous
Next

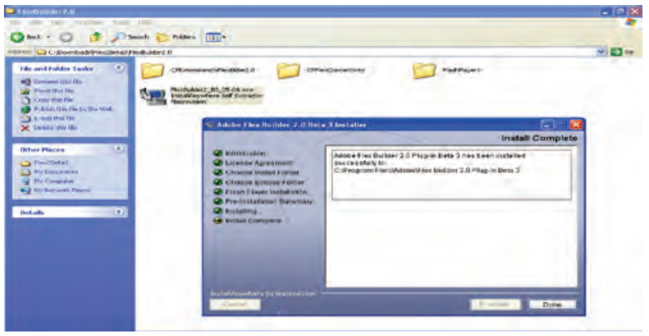
After you press OK, the path to the Eclipse plug-in is entered. Press Next.


Adobe Flex Builder 2.0 Beta 3 Installer
Flash Player Installation
Adobe Flash Player 9 for Internet Explorer
Adobe Flash Player 9 for Netscape or Firefox
Cancel
Previous
Next

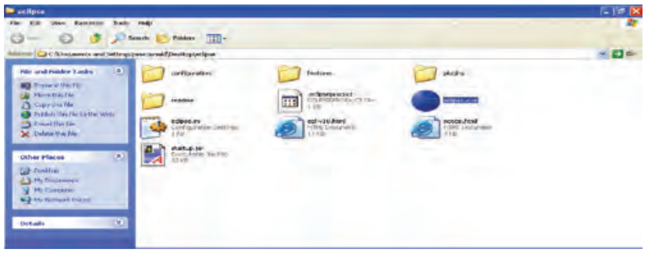
You'll have to install Flash Player 9.0 to view applications developed with Flex 2 for the browser of your choice.


Adobe Flex Builder 2.0 Beta 3 Installer
Pre-Installation Summary
Install Folder:
Adobe Flash Player Installation
Eclipse Folder to be Updated
Disk Space Information (for Installation Target):
Cancel
Previous
Install

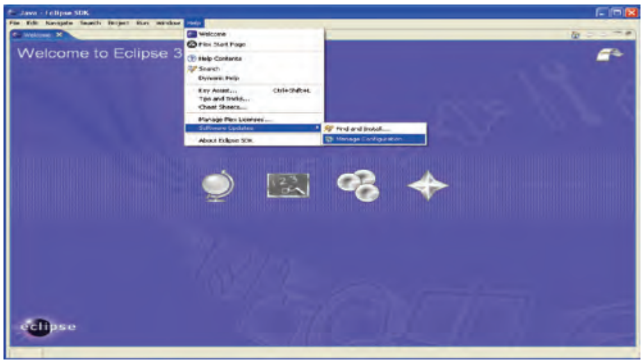
Check your pre-installation summary to verify that everything is correct then press the Install button.



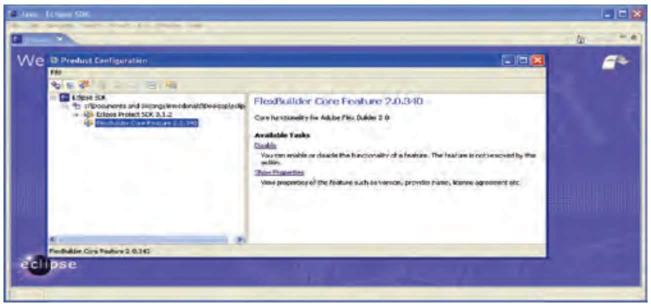
Once installed, you'll get a message that the software has been successfully installed.



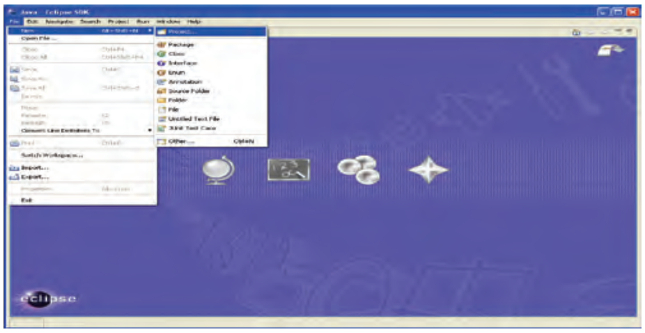
Open the Eclipse program. Make a note of the default workspace where Eclipse will be looking for your applications or choose a directory where you want Eclipse to look for your applications.



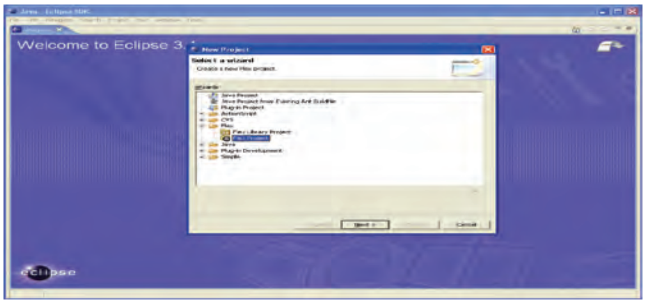
To verify that the Flex Builder 2 Beta 3 plug-in was installed, select Help/Software Updates/Manage Configurations.



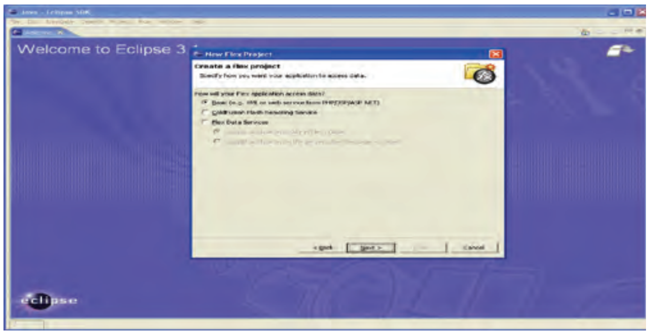
You should see the Flex Builder 2 Beta 3 plug-in zorn.feature.core 2.0.340.



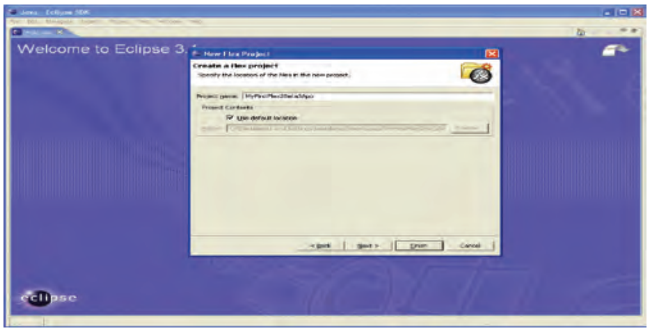
Open a new project in Eclipse.



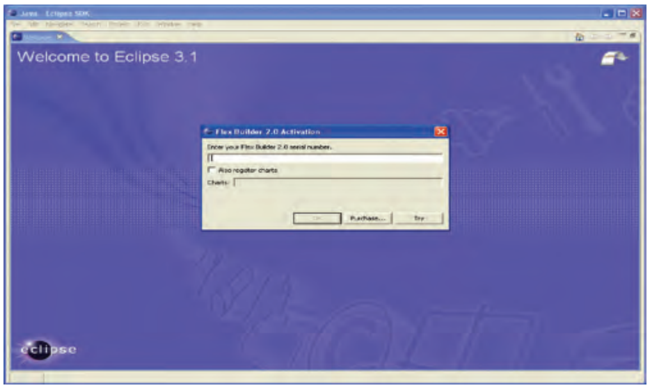
Under Flex, select Flex Project and press the Next button.



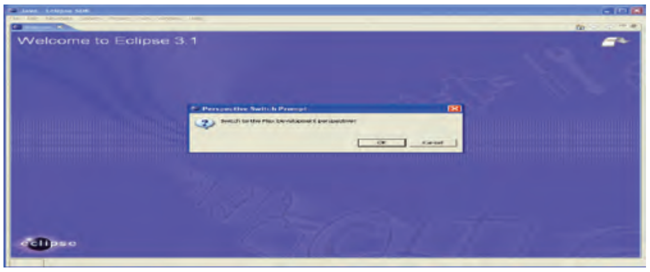
We won't use any Flex server technology for this application, so select the Basic radio button and press the Next button.



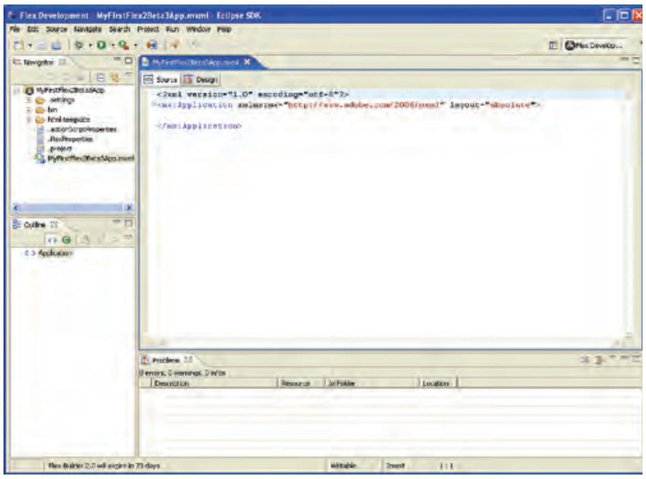
Name the project and press the Finish button. I called mine MyFirstFlex2Beta3App.



Enter the Flex Builder 2.0 serial number or select Try to try Flex Builder.

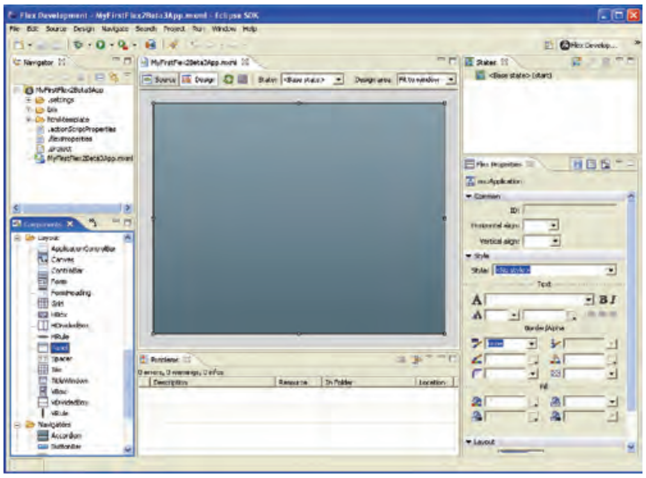


Select OK to switch to the Flex Development Perspective.

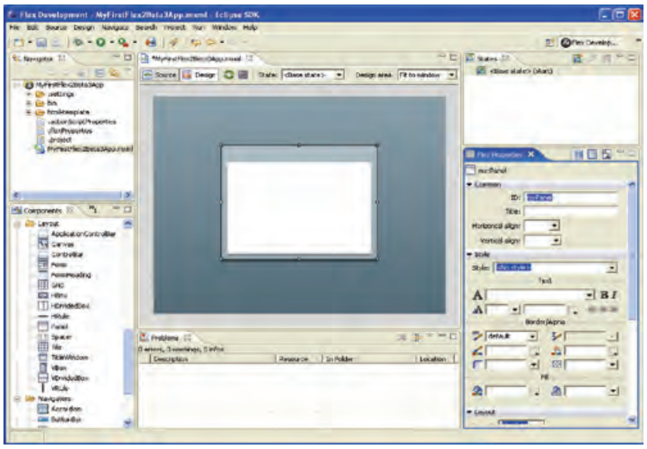


```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute">

</mx:Application>
```

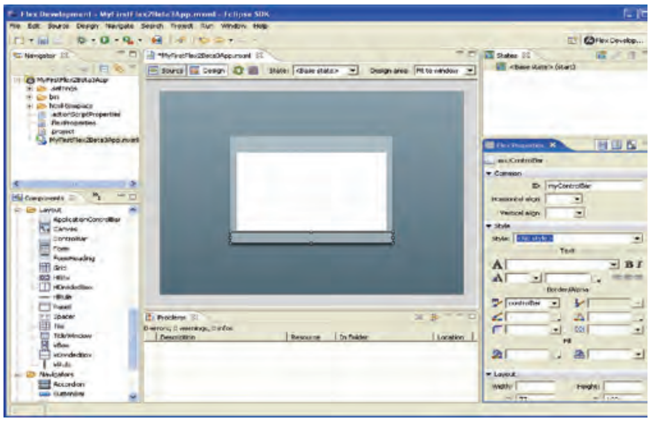
Flex defaults to the Source view. Select the Design tab to switch to the Design view.



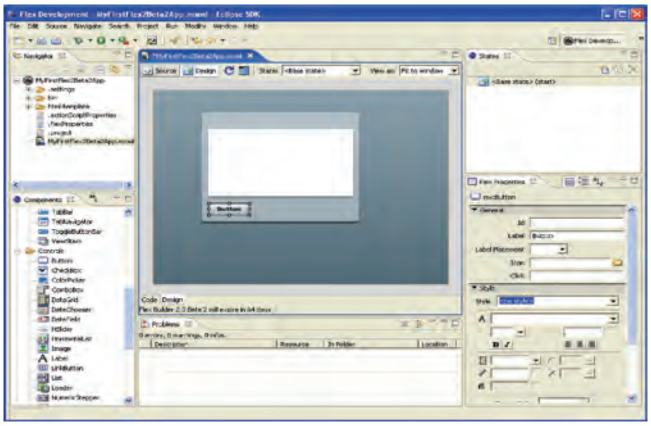
In the Design view locate the Panel component under the Components tab on the lower left and drag the Panel to the stage.



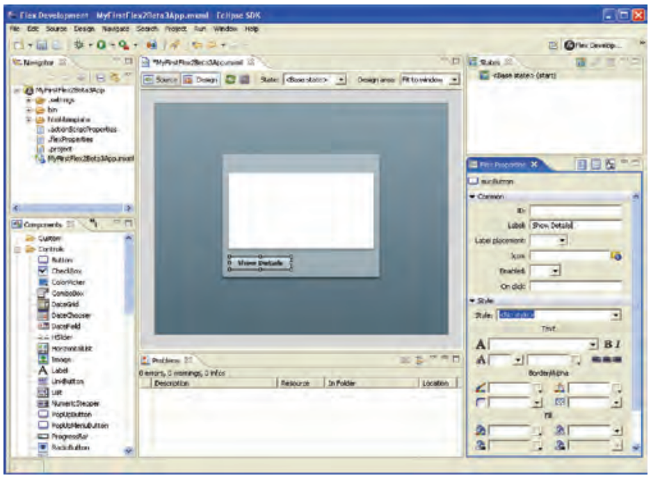
In the Flex Properties tab on the lower right, type in myPanel next to the id field under the General area. Next, locate the ControlBar component on the lower left and drag it to the bottom of the Panel.



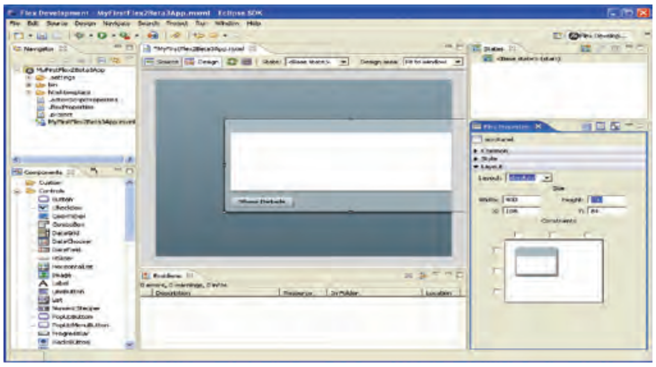
Next to the id in the Flex Properties type myControlBar in the General area.



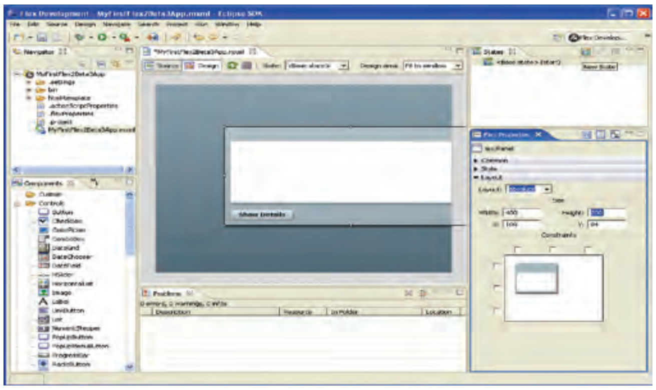
Locate the Button component under the Controls folder on the lower-left Components tab. Drag the Button component and put it over the ControlBar component.



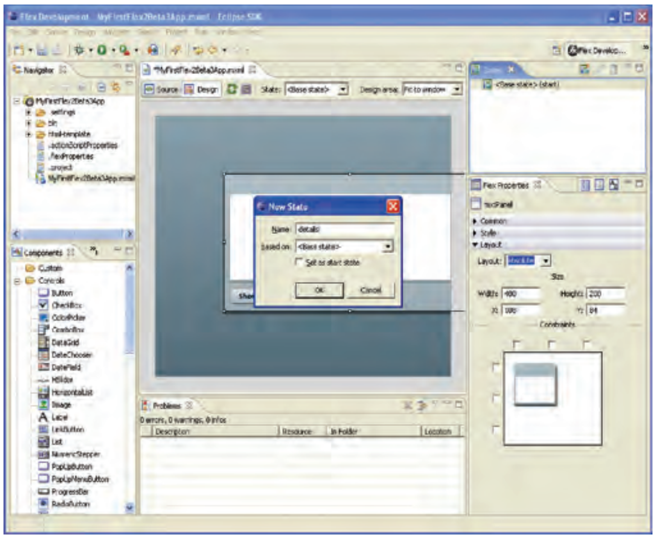
Change the label to Show Details.



Click on the Panel again. Under Flex Properties, locate the Layout area. Change the Layout of the Panel to Width = 400, Height = 200.

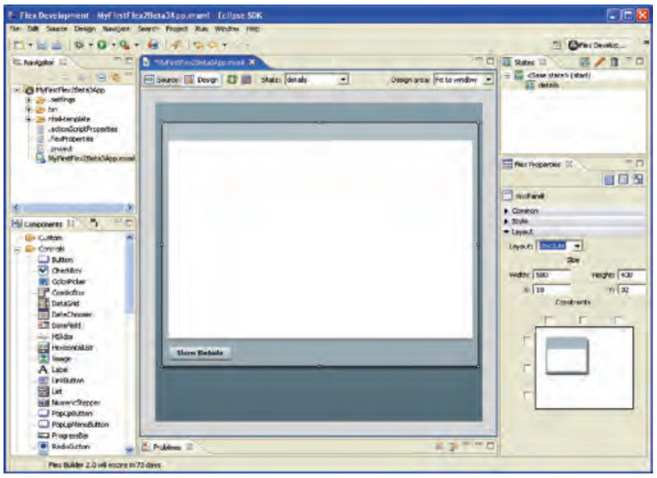


Locate the Add New State icon in the upper right corner. Click on it to Add a New State.

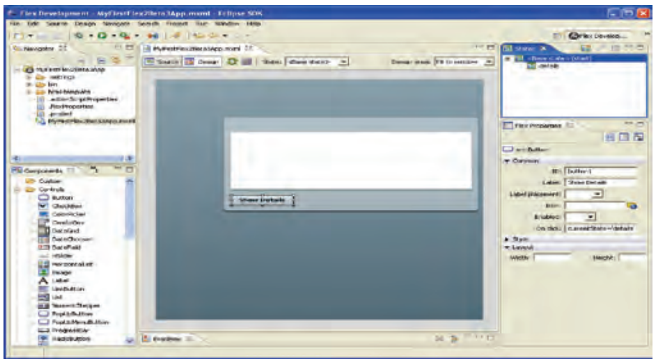


Name: details
Based on: <Base state>
Set as start state

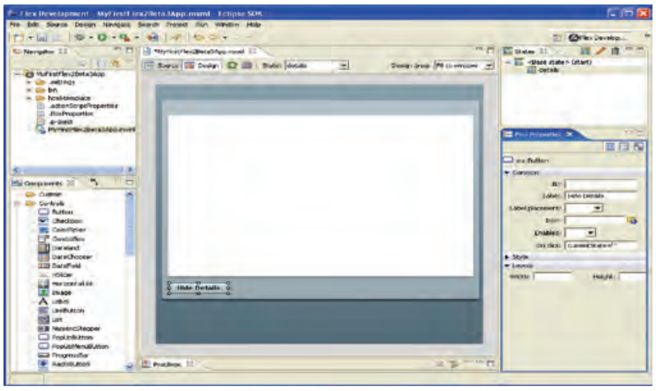
We'll call this state details. Type details in the Name field and press OK.



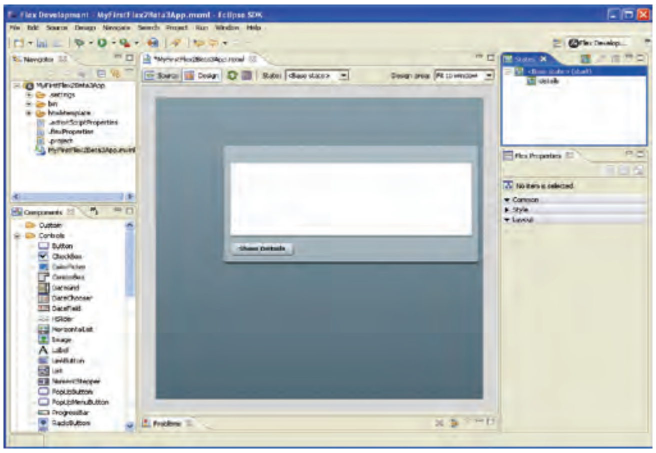
Change the Width on the details Panel to 500 and the Height to 400. Select the Button and delete it. Drag a new button component to the BarControl and change the label to Hide Details.



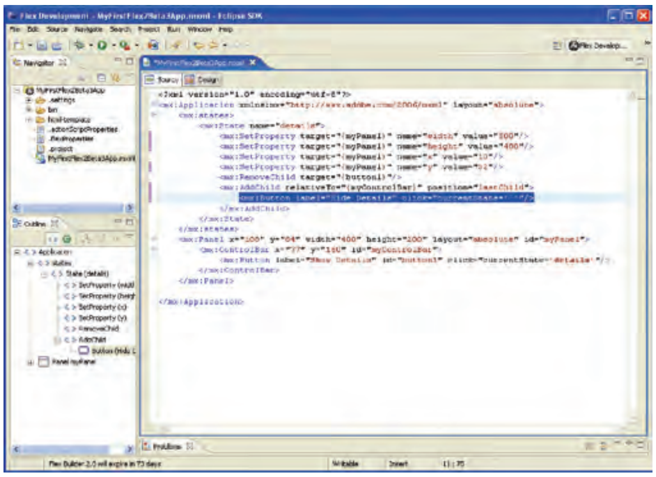
Click the Base State on the upper-right-hand corner. Click the button. In Flex Properties, under the Common area, Type currentState='details' next to On click to make the base state "transform" or turn into the details state.



Click the details State on the upper-right-hand corner. Click the button. In Flex Properties, under the General area, Type currentState=' ' next to Click to make the details state "transform" or turn back into the base state.



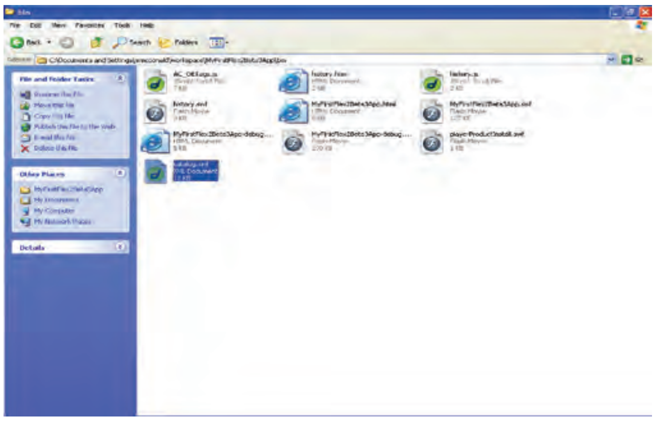
Click on the Base state and select the Source tab to display the source code for the Base state.



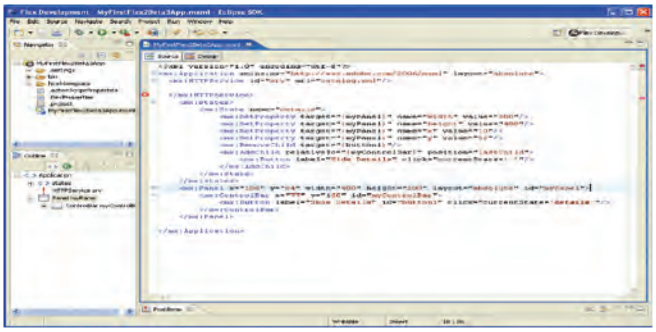
```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute">
    <mx:states>
        <mx:State name="details">
            <mx:SetProperty target="{myPanel}" name="width" value="500"/>
            <mx:SetProperty target="{myPanel}" name="height" value="400"/>
            <mx:SetProperty target="{myPanel}" name="x" value="10"/>
            <mx:SetProperty target="{myPanel}" name="y" value="52"/>
            <mx:RemoveChild target="{button1}"/>
            <mx:AddChild relativeTo="{myControlBar}" position="lastChild">
                <mx:Button label="Hide Details" click="currentState=''"/>
            </mx:AddChild>
        </mx:State>
    </mx:states>
    <mx:Panel x="100" y="64" width="400" height="200" layout="absolute" id="myPanel">
        <mx:ControlBar x="77" y="160" id="myControlBar">
            <mx:Button label="Show Details" id="button1" click="currentState='details'"/>
        </mx:ControlBar>
    </mx:Panel>

</mx:Application>
```

Flex uses MXML markup language. Your code should look similar to this.

Next, we will add some data to our panels using the HTTPService command to read an XML file located currently in your application bin folder. Copy the Catalog.xml file to your application's bin folder.



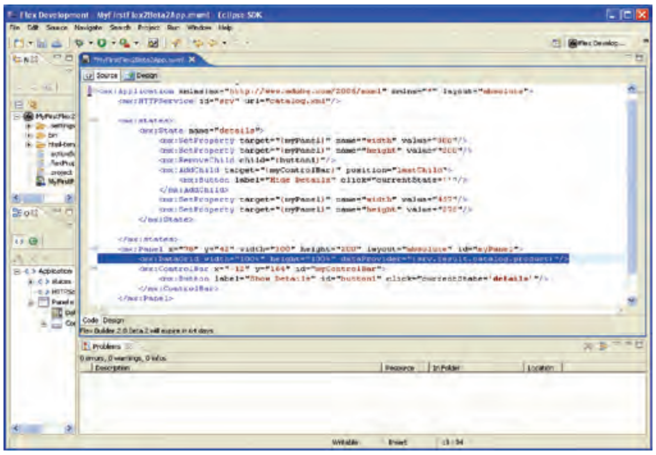
We're going to add three lines to our source code. The first line will define the service we'll be using.



```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute">
    <mx:HTTPService id="srv" url="catalog.xml"/>

    </mx:HTTPService>
    <mx:states>
        <mx:State name="details">
            <mx:SetProperty target="{myPanel}" name="width" value="300"/>
            <mx:SetProperty target="{myPanel}" name="height" value="400"/>
            <mx:SetProperty target="{myPanel}" name="x" value="10"/>
            <mx:SetProperty target="{myPanel}" name="y" value="52"/>
            <mx:RemoveChild target="{button1}"/>
            <mx:AddChild relativeTo="{myControlBar}" position="lastChild">
                <mx:Button label="Hide Details" click="currentState=''"/>
            </mx:AddChild>
        </mx:State>
    </mx:states>
    <mx:Panel x="100" y="64" width="400" height="200" layout="absolute" id="myPanel">
        <mx:ControlBar x="77" y="160" id="myControlBar">
            <mx:Button label="Show Details" id="button1" click="currentState='details'"/>
        </mx:ControlBar>
    </mx:Panel>

</mx:Application>
```

Right above the states tags type:

```
<mx:HTTPService id="srv" url="catalog.xml"/>.
```



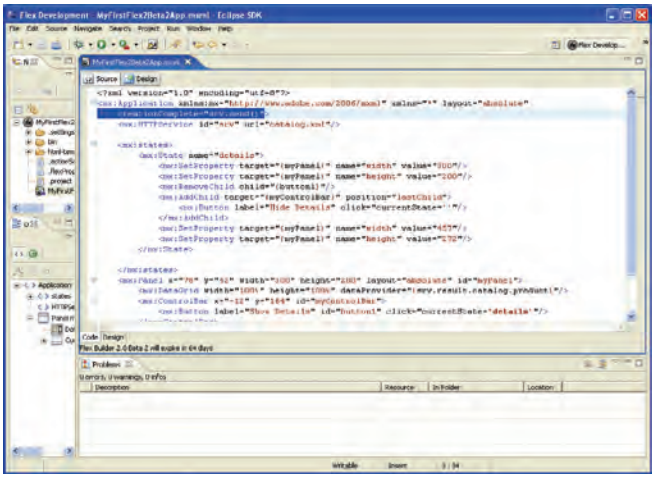
```
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" xmlns="*" layout="absolute">
    <mx:HTTPService id="srv" url="catalog.xml"/>

    <mx:states>
        <mx:State name="details">
            <mx:SetProperty target="{myPanel}" name="width" value="300"/>
            <mx:SetProperty target="{myPanel}" name="height" value="200"/>
            <mx:RemoveChild child="{button1}"/>
            <mx:AddChild target="{myControlBar}" position="lastChild">
                <mx:Button label="Hide Details" click="currentState=''"/>
            </mx:AddChild>
            <mx:SetProperty target="{myPanel}" name="width" value="457"/>
            <mx:SetProperty target="{myPanel}" name="height" value="272"/>
        </mx:State>

    </mx:states>
    <mx:Panel x="78" y="42" width="300" height="200" layout="absolute" id="myPanel">
        <mx:DataGrid width="100%" height="100%" dataProvider="{srv.result.catalog.product}"/>
        <mx:ControlBar x="-12" y="164" id="myControlBar">
            <mx:Button label="Show Details" id="button1" click="currentState='details'"/>
        </mx:ControlBar>
    </mx:Panel>
```

Flex Builder 2.0 Beta 2 will expire in 64 days

0 errors, 0 warnings, 0 infos

Next, define the container that will hold the data we're retrieving. Right before the mx:ControlBar tag type:

```
<mx:DataGrid width="100%" height="100%"dataProvider="{srv.result.
catalog.product}"/>
```



```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" xmlns="*" layout="absolute"
    creationComplete="srv.send()">
    <mx:HTTPService id="srv" url="catalog.xml"/>

    <mx:states>
        <mx:State name="details">
            <mx:SetProperty target="{myPanel}" name="width" value="300"/>
            <mx:SetProperty target="{myPanel}" name="height" value="200"/>
            <mx:RemoveChild child="{button1}"/>
            <mx:AddChild target="{myControlBar}" position="lastChild">
                <mx:Button label="Hide Details" click="currentState=''"/>
            </mx:AddChild>
            <mx:SetProperty target="{myPanel}" name="width" value="457"/>
            <mx:SetProperty target="{myPanel}" name="height" value="272"/>
        </mx:State>

    </mx:states>
    <mx:Panel x="78" y="42" width="300" height="200" layout="absolute" id="myPanel">
        <mx:DataGrid width="100%" height="100%" dataProvider="{srv.result.catalog.product}"/>
        <mx:ControlBar x="-12" y="164" id="myControlBar">
            <mx:Button label="Show Details" id="button1" click="currentState='details'"/>
```

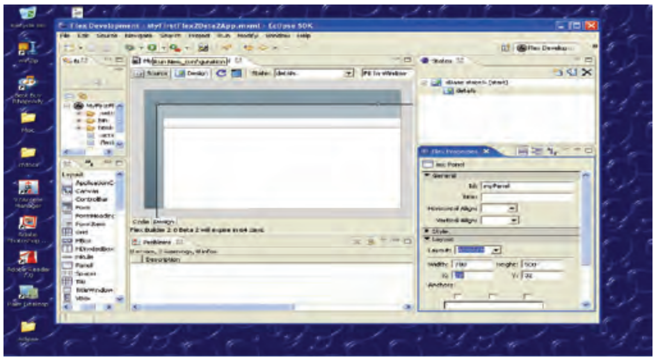
Flex Builder 2.0 Beta 2 will expire in 64 days

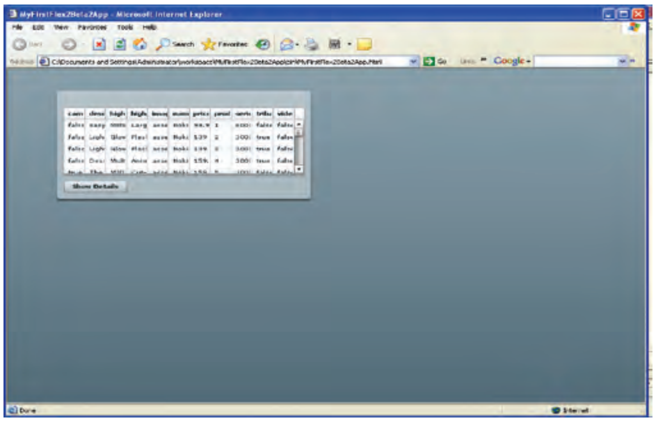
0 errors, 0 warnings, 0 infos

Get the data when the application starts up by adding the following line to the mx:Application tag:

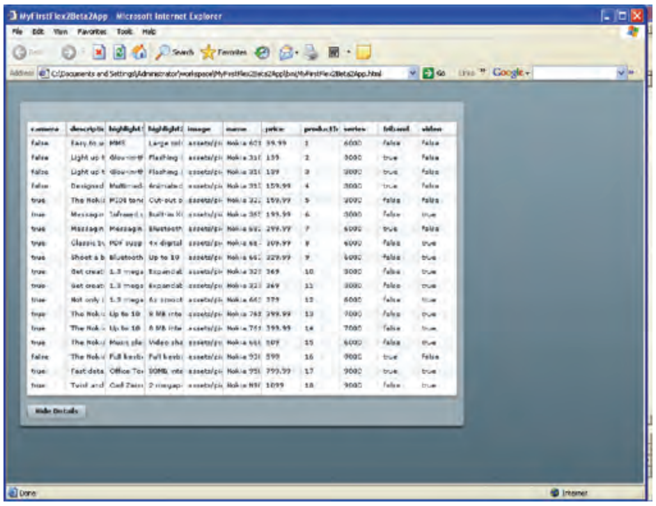
```
creationComplete="srv.send()"
```

Now save and run your application.

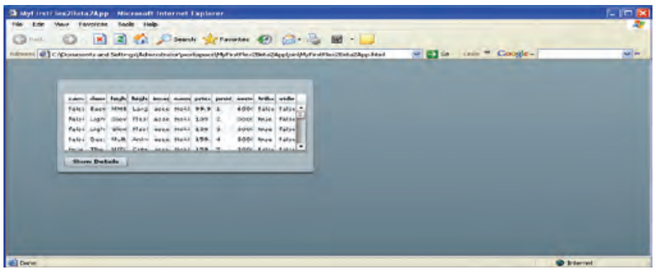




Press the Show Details Button.



Here the Details are displayed. Press the Hide Details button to return to the previous screen.



Congratulations! You've just built your first Flex application.

About the Author

Mary McDonald has served the last four years as the Northern Indiana Adobe Users Group Manager (see www.ninmug.org), has attended many conferences including CFUNITED, MAX, MXNorth and TodCon. Out of her 16 years working as a software developer, Mary has worked the last 6 years with ColdFusion, SQL, Flash, and Flex and is a Certified 5.0 ColdFusion Developer. Mary's other interests include music, photography, traveling, dogs, and most recently Tai Chi.

mary_mcdonald@earthlink.net

CFDJ Advertiser Index

ADVERTISER	URL	PHONE	PAGE
Adobe	www.adobe.com		11
AjaxWorld	www.ajaxseminar.com	201-802-3022	34
CFDynamics	www.cfdynamics.com	866-233-9626	3
CFDJ	http://coldfusion.sys-con.com/	800-303-5282	
CommunityMX	www.communitymx.com/trial		4
EdgeWebHosting	edgewebhosting.net	1-866-334-3932	5
Fusion debug	www.fusondebug.com		31
Hal Helms	halhelms.com		23
HostMySite	www.hostmysite.com	877-215-4678	19+52
Hotbanana	hotbana.com/cfdj	866-296-1803	23
InterAKT	www.interaktonline.com/		51
Paperthin	www.paperthin.com	1-800-940-3087	15
Real World Flex	www.flexseminar.com	201-802-3020	39
TeraTech	www.teratech.com		7
VitalStream	www.vitalstream.com/tools/mxdj.asp	800-254-7554	16-17

Adobe
Solution Partner
2006

INTERGRAL
information solutions

</cf_bugs>

FusionDebug™
The new interactive CF debugger.

fusion debug™
www.fusiondebug.com

Using FusionDebug™ you can:

- interactively step through CF code
- set breakpoints
- step directly into custom CF tags and CFC's
- set watch expressions
- examine and modify variables during execution
- view stack frames
- integrate with CFEclipse and the Eclipse editor

ColdFusion

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/

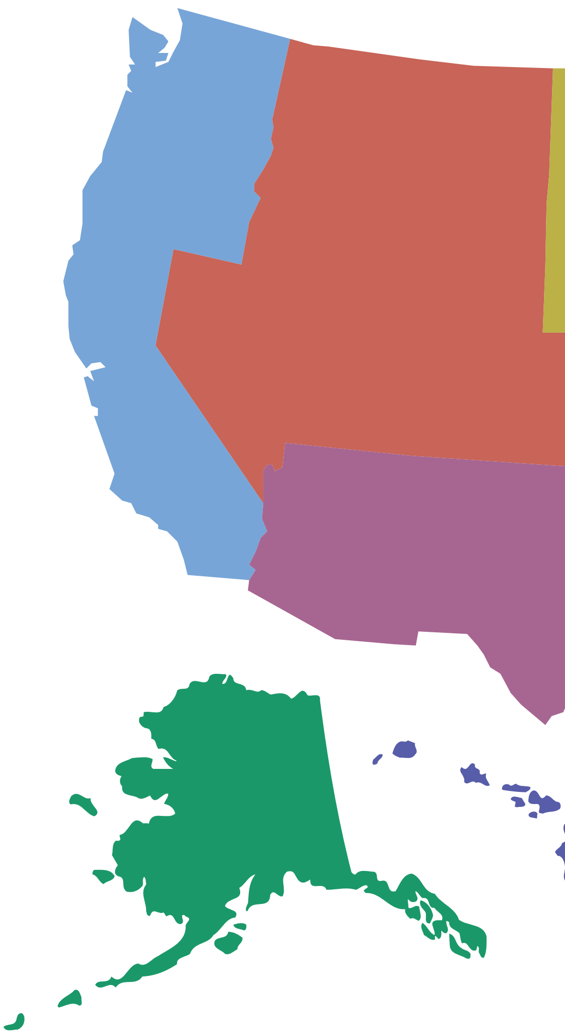
Tennessee
Nashville, TN CFUG
http://www.ncfug.com

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



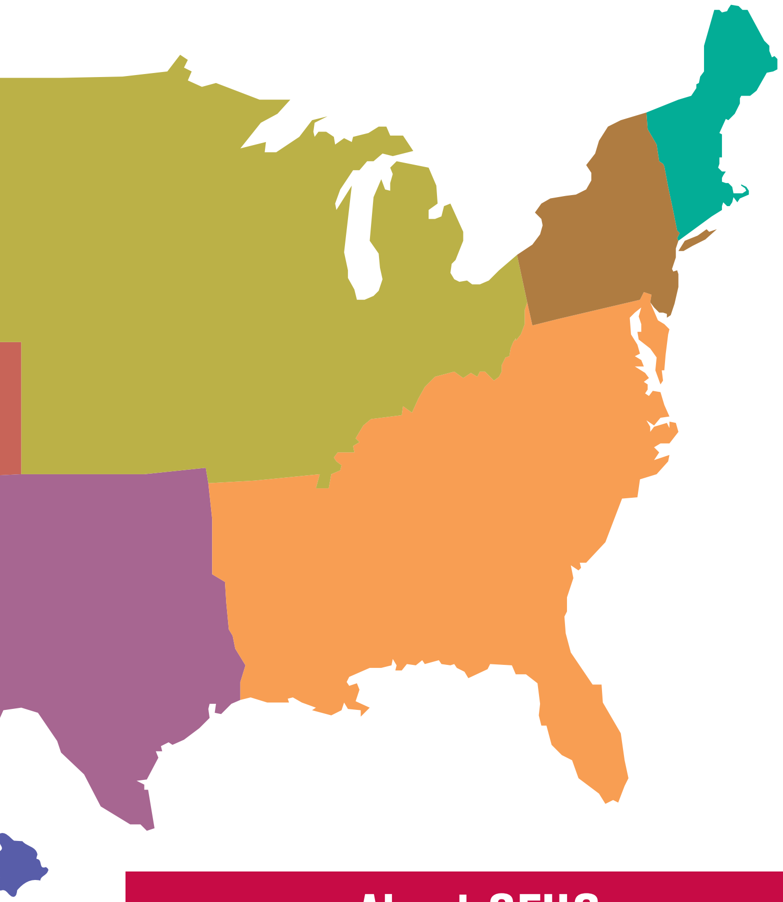
Utah
Salt Lake City, UT CFUG
www.slcfug.org

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

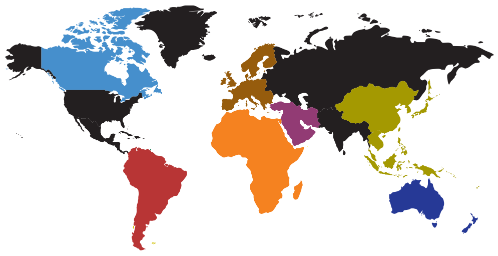
Washington
Seattle CFUG
http://www.seattlecfug.com/

Jser Groups

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



INTERNATIONAL



Australia ACT CFUG http://www.actcfug.com	**South Africa** Joe-Burg, South Africa CFUG www.mmug.co.za
Australia Queensland CFUG http://qld.cfug.org.au/	**South Africa** Cape Town, South Africa CFUG www.mmug.co.za
Australia Victoria CFUG http://www.cfcentral.com.au	**Spain** Spanish CFUG http://www.cfugspain.org
Australia Western Australia CFUG http://www.cfugwa.com/	**Sweden** Gothenburg, Sweden CFUG www.cfug-se.org
Canada Kingston, ON CFUG www.kcfug.org	**Switzerland** Swiss CFUG http://www.swisscfug.org
Canada Toronto, ON CFUG www.cfugtoronto.org	**Turkey** Turkey CFUG www.cftr.net
Germany Central Europe CFUG www.cfug.de	**United Kingdom** UK CFUG www.ukcfug.org
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	

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.

Rich Internet Applications: AJAX,

SANTA CLARA SILICON VALLEY

SYS-CON Events is proud to announce the first-ever AjaxWorld Conference & Expo 2006!

The world-beating Conference program will provide developers and IT managers alike with comprehensive information and insight into the biggest paradigm shift in website design, development, and deployment since the invention of the World Wide Web itself a decade ago.

The terms on everyone's lips this year include "AJAX," "Web 2.0" and "Rich Internet Applications." All of these themes play an integral role at AjaxWorld. So, anyone involved with business-critical web applications that recognize the importance of the user experience needs to attend this uniquely timely conference – especially the web designers and developers building those experiences, and those who manage them.

CALL FOR PAPERS NOW OPEN!

We are interested in receiving original speaking proposals for this event from i-Technology professionals. Speakers will be chosen from the co-existing worlds of both commercial software and open source. Delegates will be interested learn about a wide range of RIA topics that can help them achieve business value.

Flash, Web 2.0 and Beyond...

REGISTER TODAY AND SAVE!

October 3-4, 2006

Santa Clara Convention Center
Hyatt Regency Silicon Valley
Santa Clara, CA

To Register
Call 201-802-3020 or
Visit www.AjaxWorldExpo.com

May 7-8, 2007

First International AjaxWorld Europe
Amsterdam, Netherlands

"Over the two information-packed days, delegates will receive four days' worth of education!"

Early Bird*
(Register Before August 31, 2006) **$1,495****
See website or call for group discounts

Special Discounts*
(Register a Second Person) **$1,395****
See website or call for group discounts

(5 Delegates from same Company) **$1,295/ea.****
See website or call for group discounts

On-Demand Online Access
(Any Event) **$695**

*Golden Pass access includes Breakfast, Lunch and Coffee Breaks, Conference T-Shirt, Collectible Lap-Top Bag and Complete On-Demand Archives of sessions in 7 DVDs!

**OFFER SUBJECT TO CHANGE WITHOUT NOTICE, PLEASE SEE WEBSITE FOR UP-TO-DATE PRICING

"It Was The Best AJAX Education Opportunity Anywhere in the World!" —John Hamilton

Topics Include...

Themes:
> Improving Web-based Customer
> Interaction
> AJAX for the Enterprise
> RIA Best Practices
> Web 2.0 – Why Does It Matter?
> Emerging Standards
> Open Source RIA Libraries
> Leveraging Streaming Video

Technologies:
> AJAX
> The Flash Platform
> The Flex 2 Framework & Flex Builder 2
> Microsoft's approaches: ASP.NET, Atlas, XAML with Avalon
> JackBe, openLaszlo
> JavaServer Faces and AJAX
> Nexaweb
> TIBCO General Interface

Verticals:
> Education
> Transport
> Retail
> Entertainment
> Financial Sector
> Homeland Security

GROUP DISCOUNTS AVAILABLE:
— 5 Delegates from Same Company — for only $995 (each)
— Register a Second Person — for only $1195

LIVE SIMULCAST!
AROUND THE WORLD ON SYS-CON.TV

Receive FREE WebCast Archives of Entire Conference!

The best news for this year's conference delegates is that your "Golden Pass" registration now gives you full access to all conference sessions. We will mail you the complete content from all the conference sessions in seven convenient DVDs after the live event takes place.

► **This on-demand archives set is sold separately for $995**

SYS-CON EVENTS
For more great events visit www.EVENTS.SYS-CON.com
VISIT WWW.AJAXWORLDEXPO.COM FOR THE MOST COMPLETE UP-TO-DATE INFORMATION

DirectoryWatcher Event Gateway: Ditching the Scheduler

A textbook case for using the gateway

By Jeff Peters

I remember sitting at a ColdFusion conferences (probably CFUN) back when CFMX 7 was still in its pre-beta stage and watching as Ben Forta revealed some of the new features. There was a mixed reaction when he talked about event gateways.

While the idea of having messaging built into ColdFusion seemed appealing to most of the audience, the idea of watching a directory for its content to change wasn't as convincing. As I looked around at people's faces, I saw a mixture of excitement, inspiration, and blank stares.

I would guess the blank stares belonged to those for whom directories are relatively static things. Files on the server only change at the behest of a developer, so what's the big deal about having the server watch to see when they change? Let's face it: directory listings are pretty boring on first consideration.

I, on the other hand, couldn't wait to get my hands on CFMX 7 in production, because the DirectoryWatcher event gateway promised to solve a problem that had become all too familiar at that point in my career. I'll bet it's a situation that you may have run into yourself at some point, though I'm sure the implementation details will be a bit different in your environment.

The project we were working on at that time involved interfaces with two other systems. For the first system, we had a very smooth interface that employed a Data Migration Zone (DMZ) database to allow our Oracle database to exchange data with the other system's Oracle database without either of us needing authorization to the other system. It all ran beautifully and automatically.

The second system, though, was a bit less elegant. Our interface on that end was done through the exchange of XML datasets between the other system (Microsoft SQL Server) and ours (Oracle). While the XML part of the equation was quite nice, the channel that had been chosen for transfers was FTP. So the other system was allowed to upload files to our server using an FTP script. To import the XML data from an uploaded file into our system we ran a CFML template that would read any files in the directory, parse the XML, insert the data into our system, and move the file to a permanent directory on the server. This template was scheduled to run once every 15 minutes, using the CF Scheduler. Figure 1 shows how the process worked.

Whenever the customer system needed to send us data, it

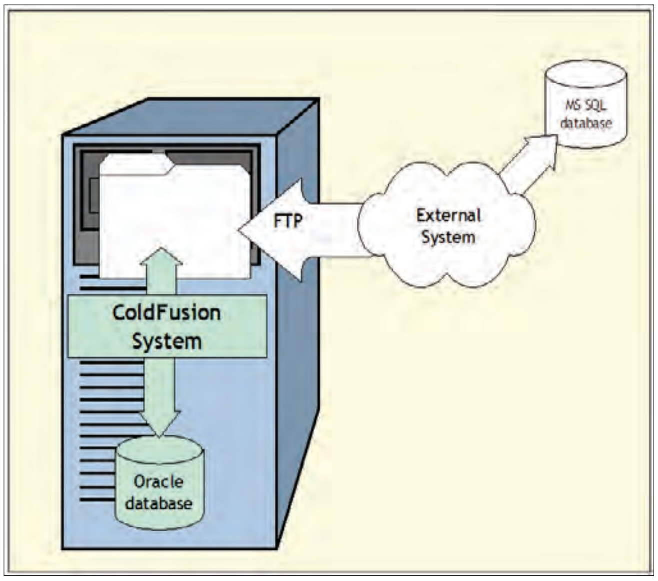


Figure 1:

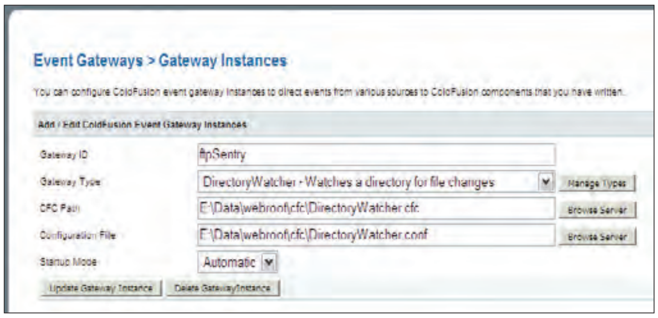
would generate an XML dataset, save it according to an agreed-on naming convention, and send it to our server via FTP. Once the dataset was on our server, the scheduled job would detect the new file the next time the scheduled job ran, and handle it accordingly.

Now, considering possible alternatives (like e-mail or even snail mail), our scheduled template solution wasn't bad at all. It worked. However, the schedule meant that inbound data might sit in a queue of sorts (the FTP directory) for up to 15 minutes, waiting on the scheduled template to run. That's why I was immediately interested in the DirectoryWatcher event gateway. I wanted a way to get rid of the time lag between when we got a dataset and when it got processed into the database.

In fact this situation turned out to be a textbook case for using the gateway. We already had the process in place and tested; the only thing we had to do was set up the gateway to kick off the process instead of having it wait for the next scheduled time. All we had to do was set up the gateway and write a configuration file for it, and a CFC to handle the events.

Setting up the DirectoryWatcher is an exercise in simplicity. In the ColdFusion Administrator, select Gateway Instances under the Event Gateways menu item on the left, and fill out the form for the new gateway instance. An example is shown in Figure 1.

The Gateway ID is an arbitrary string. I used "ftpSentry"



Event Gateways > Gateway Instances

You can configure ColdFusion event gateway instances to direct events from various sources to ColdFusion components that you have written.

Add / Edit ColdFusion Event Gateway Instances

Gateway ID	ftpSentry
Gateway Type	DirectoryWatcher - Watches a directory for file changes
CFC Path	E:\Data\webroot\cfc\DirectoryWatcher.cfc
Configuration File	E:\Data\webroot\cfc\DirectoryWatcher.conf
Startup Mode	Automatic

Figure 2:

since I wanted this instance to watch over my FTP directory. The Gateway Type is, of course, DirectoryWatcher.
CFC Path and Configuration File are full paths to the gateway's handler CFC and configuration file, respectively. In the example, I used DirectoryWatcher.cfc and DirectoryWatcher.conf, respectively.

After clicking OK, the new instance shows up in the list of Configured ColdFusion Event Gateway Instances, as shown in Figure 2 (the new instance is highlighted there). Note that it's not yet running — we'll take care of that after we write the handler CFC and configuration file. Also, I named the configuration file with a .conf extension — a bit of Linux influence sneaking in there — where the examples use .cfg. ColdFusion doesn't care what extension we use.

The configuration file for a DirectoryWatcher gateway is very simple — just name=value pairs, each on its own line. At a minimum, it contains the directory we want to watch in the form directory=c:/ftpInbox. The other values are optional and we didn't need any of them for this example, but I'll mention them briefly for reference purposes.

The recurse option allows the gateway to watch all sub-directories of the specified directory. Its default value is no. If it's set to yes, all sub-directories of the specified directory will be watched.

The interval option specifies how long to wait between checks of the directory. Its default is 60000 (60 seconds).
The extensions option specifies which filename extensions to watch. Its default is * for all extensions. If we want to watch more than one extension, the value should be a comma-delimited list. So if we wanted to watch for XML and CSV files, we would need extensions=xml,csv.

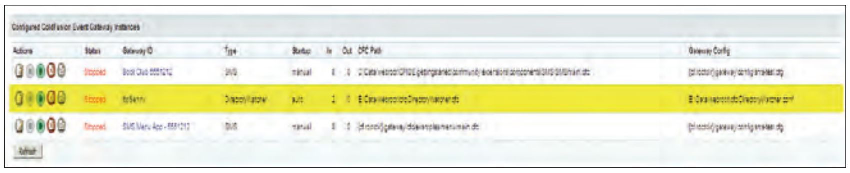
The xFunction options specify which function in the gateway's CFC will be called in response to an action in the directory. They are changeFunction, addFunction, and deleteFunction and their default values are onChange, onAdd, and onDelete, respectively.

For simplicity's sake, we used all the defaults, so our configuration file only has one line: directory=c:/ftpInbox.
The only thing left to do now is set up the handler CFC for the gateway. Since we already had the code in place to handle our import tasks, the CFC really only needed one function — onAdd — to call that code. But I liked the idea of monitoring the FTP directory for all file activities and keeping a log, so I started with the sample CFC from the DirectoryWatcher example that comes with CFMX and added a bit of code to the onAdd function to suit our needs. The code is in Listing 1.

Each event triggers its corresponding function, and each function writes a line to the ftpSentry log. In addition, the onAdd event uses <cfmodule> to call the desired template, and writes an error message to the log if any errors are thrown. Some example entries from the log are shown in Listing 2.
You can see how the file ASDO200607112876.xml was added to the directory then deleted a minute later. This is due to the automated process reading the file, adding its data to the database then moving the file to an archive directory.

So using the DirectoryWatcher event gateway improved our customer response time by reducing the scheduled wait from a 15-minute maximum to one-minute maximum. The other benefit is that our handler code doesn't fire unless a change is made to the directory. Under the scheduler scenario, our process read the directory every 15 minutes and ran some logic to determine if there were new files present. If there were, it would call the template to read and insert the data and move the file to the archive directory. The new approach with the event gateway is much simpler in terms of the code we had to write and more efficient in processing terms.

Of course, this is just one application of the DirectoryWatcher event gateway. You can use instances of it in your own system to trigger all manner of processes based on the activity on your server. For example, you could monitor a directory for file-size limits, and remove any new files after the directory reaches the limit. Or maybe you want to remove older files first, providing space for new files. Really the sky's the limit in terms of file manipulation. The DirectoryWatcher just



Configured ColdFusion Event Gateway Instances

Actions	Status	Gateway ID	Type	Startup	In	Out	CFC Path	Gateway Config
	Stopped	Book Club - 5551212	SMS	manual	0	0	[illegible]	{cf.rootdir}/gateway/config/smstest.cfg
	Stopped	ftpSentry	DirectoryWatcher	auto	2	0	E:\Data\webroot\cfc\DirectoryWatcher.cfc	E:\Data\webroot\cfc\DirectoryWatcher.conf
	Stopped	SMS Menu App - 5551212	SMS	manual	0	0	{cf.rootdir}/gateway/cfc/examples/menu/main.cfc	{cf.rootdir}/gateway/config/smstest.cfg

Figure 3:

gives you a window into your filesystem so you can automate these tasks.

About the Author

Jeff Peters is an application architect and program manager for OTS, Inc. in Vienna, Virginia. He is the author of several books on ColdFusion and Fusebox (www.protonarts.com). You can listen to Jeff weekly with Hal Helms on the "Helms and Peters Out Loud" podcast (www.helmsandpeters.com)

.

jeff@grokfusebox.com.

Listing 1: DirectoryWatcher.cfc

```
<cfcomponent>
  <cffunction name="onAdd" output="no">
    <cfargument name="CFEvent" type="struct" required="yes">
    <cfset data = CFEvent.data>
    <cflog application="No"
                 file="ftpSentry"
                 text="ACTION: #data.type#;  FILE: #data.filename#;
TIME: #timeFormat(data.lastmodified)#">
    <cftry>
      <cfmodule template="/myApp/index.cfm?fuseaction=system.check-
InFTP">
      <cfcatch type="any">
        <cflog application="no"
               file="ftpSentry"
               text="ERROR: #cfcatch.Message#; TIME:
#timeFormat(Now())#">
      </cfcatch>
    </cftry>
  </cffunction>

  <cffunction name="onDelete" output="no">
    <cfargument name="CFEvent" type="struct" required="yes">
    <cfset data = CFEvent.data>
    <cflog application="No"
                 file="ftpSentry"
                 text="ACTION: #data.type#;  FILE: #data.filename#
TIME: #timeFormat(data.lastmodified)#">
  </cffunction>

  <cffunction name="onChange" output="no">
    <cfargument name="CFEvent" type="struct" required="yes">
    <cfset data = CFEvent.data>
    <cflog application="No"
                 file="ftpSentry"
                 text="ACTION: #data.type#;  FILE: #data.filename#;
TIME: #timeFormat(data.lastmodified)#">
  </cffunction>
</cfcomponent>
```

Listing 2: ftpSentry.log

```
"Severity","ThreadID","Date","Time","Application","Message"
"Information","Thread-11","07/11/06","12:33:56",,"C:\CFusionMX7\logs\
ftpSentry.log initialized"
"Information","Thread-11","07/11/06","12:33:56",,"ACTION: ADD;  FILE:
C:\ftpSentry\ASD0200607112876.xml;       TIME: 07:45 PM"
"Information","Thread-11","07/11/06","12:34:56",," ACTION: DELETE;
FILE: C:\ftpSentry\C:\ftpSentry\ASD0200607112876.xml"
```

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

REALWORLD
FLEX SEMINAR

www.flexseminar.com

Learn How to Build the Next Generation of Web Apps **from the Experts!**

August 14, 2006
The Roosevelt Hotel
New York City

Adobe Flex 2 is a complete, powerful application development solution for creating and delivering cross-platform rich Internet applications (RIAs), within the enterprise and across the web. Not until now has there been a way for enterprise programmers and architects to work with existing tools of choice, familiar programming models and integration with existing systems and infrastructure. Multi-step processes, client-side processing, direct manipulation and data visualization are all key factors in the Flex solution.

The "Real-World Flex" One-Day Seminar will delve deep into the central workings of Flex so that the seminar delegates can integrate this timely new technology into their applications, creating powerful interactive content. Delegates will learn from the experts who not only created the product but also those who are using it to the massive benefit of their employers, their customers and the Web.

Early Bird: (Hurry for Early Bird Pricing) $395*	**Conference Price:** (Register Onsite) $495*	* Golden Pass access includes Breakfast, Lunch and Coffee Breaks, Conference T-Shirt, Collectible Lap-Top Bag and Complete On-Demand Archives of sessions in 7 DVDs!
OFFER SUBJECT TO CHANGE WITHOUT NOTICE, PLEASE SEE WEBSITE FOR UP-TO-DATE PRICING		

"Go Beyond AJAX with Flex."

The list of topics at the Real-World Flex Seminar includes:

- ✓ *The Flex Approach to RIA Development*
- ✓ *Bridging Flex and AJAX*
- ✓ *Integrating The SPRY Framework*
- ✓ *Using Flex Builder 2*
- ✓ *Flex for Java Developers*
- ✓ *ActionScript 3.0 Tips and Tricks*
- ✓ *How To Use ColdFusion with Flex*
- ✓ *Leveraging Flash*
- ✓ *Preparing for Adobe Apollo*
- ✓ *MXML Master Class*

Who Should Attend?

- ✓ *CEOs and CTOs*
- ✓ *Senior Architects*
- ✓ *Project Managers*
- ✓ *Web programmers*
- ✓ *Web designers*
- ✓ *Technology Evangelists*
- ✓ *User Interface Architects*
- ✓ *Consultants*
- ✓ *Anyone looking to stay in front of the latest Web technology!*

Sponsored by Adobe Web Developer's & Designer's Journal

Produced by SYS-CON EVENTS *The World's Leading i-Technology Event Producer* www.events.sys-con.com

Writing an RSS Aggregator

Completing the task

By Jeffry Houser

Two months ago I put together an article about building an RSS aggregator (CFDJ, Vol. 8, issue 5). Before reading this you might want to refresh your mind on the original article. Go over here – http://coldfusion.sys-con.com/read/235976.htm – to read it.

I discussed what an aggregator is and why we care to write one. RSS is a version of XML that is used to make syndicating data easy. Most blogs have an RSS feed attached to them. An aggregator takes a bunch of blog feeds and combines them. Weblogs.macromedia.com and FullAsaGoog.com are two great examples of aggregators in the ColdFusion community.

The last article stepped you through the thought process of designing the database and object model. We built two of the components: an RSSCategory component that is used to categorize the RSS feeds and an RSSFeed component that is used to enter an RSS feed into the database. We also wrote some admin code to enter a new RSS feed into the database. The article was, unfortunately, lacking the real meat of things, which is the RSSAggregator component. In this article, we'll flesh it out along with the item component and the scheduled task for running in. Before we do that, I did find one bug so let's fix that.

One Quick Bug Fix

While testing the code from the last article, I discovered a bug. It happens to the best of us, right? When entering the feed into the database (Feedip.cfm), I had written some code to retrieve the feed using cfhttp and then parse the XML to get the feed's title. This worked fine for standard RSS feeds, and referenced the title like this:

```
MyXMLVar.rss.channel.title.xmltext
```

The problem here is that the root element, RSS, is hard coded. When I tried to run this code against the weblogs.macromedia.com site, it didn't work. The reason is the RSS feed offered by weblogs.macromedia.com is RDF. The root element isn't RSS, it is rdf:RDF. The fix for this was easy:

```
MyXMLVar.xmlroot.channel.description.xmltext
```

Instead of hard coding the RSS root name, I used the xmlroot value. RDF and RSS handle items differently too, so this will come into play in some of the code from this article.

Writing the Item Component

In RSS, an item is the equivalent of a single blog post. For each item, we are storing the link to the original item, the title, and the description. There can be a lot more data associated with this, but for the sake of these articles, I decided to keep it simple. The component also has some internal values, such as a primary key named ItemID and a foreign key named RSSFeedID. The RSSFeedID tells us which item the RSS Feed has. The Item component code is shown in Listing 1.

The component starts with the cfcomponent tag (of course) and the pseudo constructor code. The pseudo constructor sets up the instance variables of the component. Once again, our components are borrowing Hal Helms basecomponent from http://halhelms.com/webresources/BaseComponent.cfc. I use this instead of writing manual getter and setter methods.

Other than inherited methods, this component contains an init method and a commit method. The init method takes an ItemID and the datasource and loads all the relevant information from a database. The commit method will insert, or update, the information in the database as needed.

Creating the Aggregator

The RSSAggregator component is shown in Listing 2. This component is a bit different than many I usually write, since its purpose is not to get data in and out of a database. It is pulling data from some far away off place and putting it in our database. This component does not contain an instance variable, and as such does not have a pseudo constructor. There are three methods, which I can explain in more detail.

The first method is GetAllFeeds. It is a private method, so it cannot be called outside of the CFC. It runs a query to retrieve all the feeds that are being watched in the database. The method returns the query. There is nothing special about this.

The second method it called ItemExists. It accepts an item component (which you learned about in the previous section of this article) and checks to see whether this item already exists in the database. If it does, it returns true, otherwise it returns false. I made the assumption that each item has a unique URL pointing to it, so that is the value the code checks to see if the item is unique.

The third method is an init method. This is the one that retrieves the feeds and stores the data in the database, if relevant. This is the only public method in the component; the getallfeeds and ItemExists methods are used by init. The init method starts by setting some local variables. These are the local variables:

- ***GetAllFeeds:*** This variable calls the GetAllFeeds function. It contains a list of all feeds that we want to pull data from.
- ***Error:*** The second local variable is the error variable. The most likely cause of an error when running this code will be that a feed times out. If the feed times out, we won't want to attempt to process the data. It defaults to false, which is no error.
- ***TempItem:*** As we loop over the feed data, this item object will be used to decide whether the data is duplicated or needs to be saved. The item component also contains the SQL for saving the item component.
- ***ItemArray:*** The ItemArray value is a temporary value that will contain an array of all items in the current feed.
- ***TempItemindex:*** When we look over the ItemArray array, a counter will be needed to keep track of which item is being examined at the moment. This is the counter variable.
- ***MyxMLVar:*** The MyXMLVar will contain the returned XML feed.

While initializing the local variables, the GetAllFeeds query was executed. The code starts by looping over it. The error variable is initialized to false. You want to make sure to initialize it each time through the loop, so that you aren't processing the current feed based on the result of the previous feed.

Next comes a try block. Inside the try block is code to retrieve the RSS feed. If the feed times out, a catch block switches the error value to try. Exiting the try block, if the error is false, the code processes the feed. If the error is true, skip the processing and go right to the next feed. Although left out at this time, there should probably be some sort of logging for feeds that cause errors.

Earlier I spoke about the differences between RSS and RDF. Most blogs I read pass out data in the RSS format, but weblogs.macromedia.com was using RDF. You can read more about RDF at http://www.w3.org/RDF/. In RSS, items are stored inside the channel. In RDF they are not. The ItemArray is initialized differently depending on the root. (If this code tries to parse another flavor of XML, it will cause problems.) The next code block uses cfloop to loop over the Item Array. It creates an item object using the tempitem variable. It sets the relevant instance data, then it uses the ItemExists function to check whether or not the item exists yet. If it doesn't, the commit method is run to save the data. Otherwise, nothing happens. The loop ends, and the method returns true. This is simple stuff, right?

The Scheduled Task

There is one last bit of code to examine in this article and that is the actual scheduled task code. The bulk of the code is located in the component, so all the scheduled task does is create an instance of the RSSAggregator and run the init method. The code looks like this:

```
<cfscript>
 variables.RSSAggregator = CreateObject("component","#request.Component-
Loc#.RSSAggregator");
 variables.RSSAggregator.init(request.dsn);
</cfscript>
```

It is probably one of the easiest scheduled tasks I've ever written.

Conclusion

This app is far from complete. Most RSS feeds contain a lot more data than just link, description, and title. A full-featured app would address those features. I left them out in the interest of length. Also, we've only built code for collecting RSS data. There is no "view' portion of this app. The only way to view the data you're collecting is to open up the database. That isn't conducive to good usability.

Every good project needs a code name, and I decided to give this project one. After some deep soul-searching, I've decided to name this project MyFriend. There are two reasons for this. The first is that I modeled the whole idea after the LiveJournal friends list. The second is that, My Friend is the name of a song that my first band recorded the first time we went into a recording studio. It was my first time in a professional recording studio, and I had been playing bass for less than a month. The results came out better than you might have expected, really. I'm a sentimental freak. Check out my www.jeffryhouser.com for the latest version of this code and let me know what you think.

About the Author

Jeff Houser has been working with computers for over 20 years. He owns a DotComIt, a web consulting company, manages the CT Macromedia User Group, and routinely speaks and writes about development issues. You can find out what he's up to by checking his Blog at www.jeffryhouser.com.

jeff@instantcoldfusion.com

Listing 1

```
<cfcomponent extends="basecomponent">

<cfscript>
 variables.instance.ItemID = "";
 variables.instance.RSSFeedID = "";
 variables.instance.Title = "";
 variables.instance.link = "";
 variables.instance.description = "";
</cfscript>

<cffunction name="Init" access="public" returntype="Boolean">
 <cfargument name="ItemID" type="string" required="true">
  <cfargument name="dsn" type="string" required="true">
   <cfset var qGetItem = "">

  <cfquery name="qGetItem" datasource="#arguments.dsn#">
   select *
   from items
   where items.ItemID = <cfqueryparam value="#arguments.itemID#"
 cfsqltype="cf_sql_varchar">
 </cfquery>

  <cfif qGetItem.recordcount is 0>
   <cfreturn false>
  </cfif>
 <cfscript>
  variables.instance.ItemID = qGetItem.ItemID;
 variables.instance.RSSFeedID = qGetItem.RSSFeedID;
  variables.instance.Title = qGetItem.Title;
  variables.instance.link = qGetItem.link;
  variables.instance.description = qGetItem.description;
 </cfscript>

 <cfreturn true>

</cffunction>

<cffunction name="Commit" access="public" returntype="Boolean">
 <cfargument name="dsn" type="string" required="true">

 <cfset var qUpdateItem = "">

 <cfif variables.instance.ItemID is "">
  <cfset variables.instance.ItemID = createuuid()>
  <cfquery name="qUpdateItem" datasource="#arguments.dsn#">
   insert into items (ItemID, RSSFeedID, Title, Link, Description)
   values (
    '#variables.instance.ItemID#',
    <cfqueryparam value="#variables.instance.RSSFeedID#"
cfsqltype="cf_sql_varchar">,
    <cfqueryparam value="#variables.instance.Title#" cfsqltype="cf_
sql_varchar">,
    <cfqueryparam value="#variables.instance.Link#" cfsqltype="cf_sql_
varchar">,
    <cfqueryparam value="#variables.instance.Description#"
cfsqltype="cf_sql_varchar"> )
  </cfquery>
 <cfelse>
  <cfquery name="qUpdateItem" datasource="#arguments.dsn#">
   update items
   set RSSFeedID = <cfqueryparam value="#variables.instance.RSS-
FeedID#" cfsqltype="cf_sql_varchar">,
    Title = <cfqueryparam value="#variables.instance.Title#"
cfsqltype="cf_sql_varchar">,
    Link = <cfqueryparam value="#variables.instance.Link#"
cfsqltype="cf_sql_varchar">,
    Description = <cfqueryparam value="#variables.instance.Descrip-
tion#" cfsqltype="cf_sql_varchar">

   where ItemID = <cfqueryparam value="#variables.instance.ItemID#"
cfsqltype="cf_sql_varchar">

  </cfquery>
 </cfif>

 <cfreturn true>

</cffunction>

</cfcomponent>
```

Listing 2

```
<cfcomponent extends="basecomponent">

<cffunction name="GetAllFeeds" access="private" returntype="query">
  <cfargument name="dsn" type="string" required="true">

  <cfset var qGetFeeds = "">

  <cfquery name="qGetFeeds" datasource="#arguments.dsn#">
   select * from RSSFeeds
 </cfquery>

  <cfreturn qGetFeeds>
 </cffunction>

 <cffunction name="ItemExists" access="private" returntype="Boolean">
  <cfargument name="item" type="item" required="true">
  <cfargument name="dsn" type="string" required="true">

  <cfset var qCheckItem = "">

  <cfquery name="qCheckItem" datasource="#arguments.dsn#">
   select itemID
   from items
   where items.link = <cfqueryparam value="#arguments.item.
get('link')#" cfsqltype="cf_sql_varchar">
  </cfquery>

  <cfif qCheckItem.recordcount gte 1>
   <cfreturn true>
  </cfif>
  <cfreturn false>

 </cffunction>

 <cffunction name="Init" access="public" returntype="void" hint="Gets
all RSS feeds, parses them, and saves new items in the database">
  <cfargument name="dsn" type="string" required="true">

  <cfset var GetAllFeeds = GetAllFeeds(arguments.dsn)>
  <cfset var error = false>
  <cfset var TempItem = CreateObject('component','item')>
  <cfset var ItemArray = "">
  <cfset var TempItemindex = 0>
  <cfset var MyXMLVar = "">

  <cfloop query="GetAllFeeds">
   <cfset error = false>
   <cftry>
    <cfhttp url="#GetAllFeeds.link#" timeout="30"></cfhttp>

    <cfcatch type="any">
     <!--- most likely cause of error is server timeout ---->
     <cfset error = true>

    </cfcatch>

   </cftry>

   <cfif error is false>
    <cfset MyXMLVar = xmlparse(cfhttp.filecontent)>

    <cfif MyXMLVar.xmlroot.xmlname is "rss">
     <cfset ItemArray = MyXMLVar.xmlroot.channel.item>
    <cfelseif MyXMLVar.xmlroot.xmlname is "rdf:RDF">
     <cfset ItemArray = MyXMLVar.xmlroot.item>
</cfif>

    <cfloop from="1" to="#arraylen(ItemArray)#" index="TempItemindex">

     <cfset TempItem = CreateObject('component','item')>
     <cfset TempItem.set('RSSFeedID',GetAllFeeds.RSSFeedID)>
     <cfset TempItem.set('Title',ItemArray[TempItemindex].Title.
xmlText)>
     <cfset TempItem.set('link',ItemArray[TempItemindex].link.
xmlText)>
     <cfset TempItem.set('description',ItemArray[TempItemindex].
description.xmlText)>
    <cfif not ItemExists(TempItem ,arguments.dsn)>
      <cfset TempItem.commit(arguments.dsn)>
     </cfif>

   </cfloop>
   <cfelse>
    <!---- should add some type of error handling here ---->
</cfif>

  </cfloop>

 </cffunction>

</cfcomponent>
```

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

Good 'ol CF" and the New Frontier

– continued from page 5

Here are just a few examples of these types of Web applications that should give you a good idea of what's coming:

- Live Plasma (http://www.liveplasma.com/) – a visual search engine for music/video information… with a slick Flash UI
- Krugle (http://www.krugle.com/) – we've all used Google to look for code or tech advice – this search engine is devoted to just that. Yes, a search engine for programmers!
- Vyew (http://www.vyew.com/content/) - what's that in the sky? It's a WebEx, it's Breeze, no, wait… it's Vyew. Yes – free online collaboration and presenting much like what you get with a product like Breeze.
- Dabble DB (http://www.dabbledb.com/) – this site, like many of the next generation apps I've been watching, is still in beta, but it's soon to go "V 1". Dabble DB offers an easy to use interface to online databases that users can create or import to their servers. It's very robust, data can be shared or extracted (as PDF, RSS, and many other formats), and includes not just standard AJAX data views but also calendar views of your online data.

So what does all of this have to do with ColdFusion? Well, nothing and everything. The bulk of these applications are developed using client side technologies, and we all know that ColdFusion is a server-side programming environment. But CF is still an environment for developing dynamic Web applications, and many of these next generation Web applications, especially the more robust web apps, do still require some server side component. I've written in the past about the fact that CF serves an important role in intelligent client side apps by offering database access and other functionality as part of a services tier… but is that all CF will be reduced to in the future?

With much optimism and enthusiasm I say, "no". The ColdFusion server team is now aggressively working on the next major release of CF – currently code named "Scorpio." While many of the features have yet to be decided or publicly announced, it is clear that the CF Team has no intention of "laying down". At CFUnited, the premier CF annual event held last month in Washington, DC, two cool features were shown during the opening keynote. One was the ability to create online presentations (rendering as Flash) dynamically using a few simple CF tags – think of it kind of like a merger between CF and Breeze (though that's a very general statement to make). The other was the ability to render, manipulate, and access data within PDF Forms… yet another new feature that will make ColdFusion an integral part of how companies do business on the Web.

While those two new possible features (I say "possible" because until the final release of the product, everything's open to change) don't necessarily have anything to do with creating the types of next generation web applications I discussed earlier, other features certainly may. ColdFusion MX 7.02, the free upgrade formerly code named "Mystic," added terrific support for Flex 2. Coupled with the Flex Builder 2 CF wizards, the current version of CF and FB2 make it easier than ever to supply data and services to Flash applications via CF. This is more than just Flash Remoting, though CF MX 7.02 does have a new Flash Gateway that supports the AMF 2 protocol. The CF Wizards make Flex/CF application development ridiculously fast and easy, and, being a guy who typically likes to write everything manually, are the first wizards I've seen in any CF-related product that I not only enjoy but find extremely useful (yes, they also generate really decent code). The new features in CF also includes a gateway for pushing content to Flex 2 and the ability to integrate CF and Flex Data Services. Flex Data Services, among other things, makes it very easy to coordinate data persistence on the server by controlling who can access and modify data at the record level. It also allows content pushes to Flex applications – another very useful feature that AJAX doesn't nativly support but that would be extremely useful if one were, say, building a social networking application.;)

I didn't want to talk about next generation Web applications and CF just to point out the strength of Flex and the new hooks for integrating and communicating between the two. Not too long ago, Damon Cooper blogged about a new feature that one of his server team members prototyped for CF 8 (the feature is an enhancement to give more programmatic control over asynchronous threading). After reading that, and knowing that the server team is currently hard at work coming up with features for the next release, I decided to post an "open letter to the CF Server Team" entry on my blog in which I listed many of the new features that I would like to see in CF 8. This caused a kind of chain reaction in the CF blog community and to date, dozens of bloggers have also let the CF Server Team know what they'd like to see in the next release. Yes, Damon and his crew have been commenting on the blogs and letting the community know that they're paying a lot of attention.

This type of feedback will help ensure that going forward, each next release of CF will have the features that we, the CF development community, needs in order to meet the demands of a rapidly changing Web and the day-to-day demands in our respective office environments. If you don't already have a blog, you can get one at any number of places, including from sys-con at http://www.blog-n-play.com/. I'll conclude my editorial this month with a simple question – one that you can now go blog about immediately… "What would you like to see in ColdFusion 8?"

feature

ColdFusion Structures

Making applications more modular and efficient

By Selene Bainum

Way back when ColdFusion 4.5 was released, the concept of structures (associative arrays to some of you) was introduced. Never one to be receptive to change – not to mention having no background in other programming languages – I shunned structures for the most part and kept on my merry way working with arrays and lists. Over the years, however, I have come to appreciate the simplicity and functionality of structures and embrace them as my favorite ColdFusion data type.

If you haven't worked with other languages such as the various flavors of C or Java, the concept of what structures are may be hard to grasp – so I hope to make it easier for you.

Like arrays and query results, structures are considered to be complex data types – as opposed to numbers or strings. Structures are made up of key/value pairs logically grouped together. For instance, the following structure contains information related to a contact:

```
Contact.FirstName = "Selene"
Contact.LastName = "Bainum"
Contact.EmailAddress = "selene@webtricks.com"
```

ColdFusion has many built-in structures that you have worked with, but may not realize: Server, Application, Session, Request, Form, URL, and Variables. So the Session structure is a logical grouping of all session information, the Application structure is a logical grouping of all application information and so on.

Structure Notations

There are two ways to access the values of structure keys: dot notation and indexed notation. You're already familiar with dot notation: Form.FieldName, Session.SessionID, Application.DSN, etc. These are all examples of structures and keys. FieldName is a key of the Form structure, SessionID is a key of the Session structure, and DSN is a key of the Application structure.

Dot notation is easy and familiar, but it has several limitations. First, the key name must be a syntactically correct ColdFusion variable name – you probably already know your form field names can't start with a number or contain spaces. Second, the name of the key must be a static value. If it's not a static value, you must use the Evaluate() function, which can be resource expensive.

Indexed notation is much more flexible: key names can start with a number and/or contain spaces and variables can be used to represent all or part of the key name without needing to use Evaluate(). With indexed notation, the key name is placed between brackets, with the left bracket being immediately to the right of the structure name. If the key is a numeric value, quotes are optional around the key name, but quotes are required for a string key name. If there are no quotes around a string key, ColdFusion will assume the value is the name of a variable and will attempt to retrieve the value of that variable to use as the key name.

For example:

```
Form[1]  // = Form["1"]
Session["SessionID"]
varA = "DSN"
varB = 1
Request[varA]  // = Request["DSN"]
Request["DSN" & varB] or Request["DSN#varB#"]  // = Request["DSN1"]
```

Looking at indexed notations, you may notice some similarities between structures and arrays. They are, with the following notable differences:

- Structure values can be referenced by a string name or a numeric value
- Structures can only be one dimension in depth
- The keys are stored in no particular order
- Structures are passed by reference, not value

Creating/Modifying Structures

It isn't always necessary to create a new empty structure before adding keys to it, but it's a good idea because there are times when it's required. To create an empty structure without any keys, you can use the StructNew() function or the cfparam tag in conjunction with StructNew():

```
<cfset Cart = StructNew() />
<cfparam name="Cart.BillingStruct" default="#StructNew()#" />
```

In the example above, a new empty structure called Cart will be created. Notice that StructNew() uses no arguments. If the structure already existed, it will be overwritten with the empty structure. Using cfparam, however, the new empty structure of Cart.BillingStruct will only be created if it doesn't already exist. Both methods are very useful, depending on your desired results.

If you've worked with any of the built-in ColdFusion data types, you have already set and modified the key/value pairs of structures. Using dot or indexed notation, you can easily set and update values of structure keys:

```
<cfset Application.DSN = "mydb" />
<cfset Cart["CCNum"] = "xxxx" />
<cfparam name="Cart.ExpDate" default="12/06" />
```

In the examples above, the structure will be created if it doesn't already exist, the key will be added if it isn't there already and the value of the key will be set. If the key already existed, its value will be overwritten with the new value.

Working with Structures

There are many structure functions that are quite useful when working with structures.

As you already know, ColdFusion has many decision functions, and several work with just structures. The IsStruct() function can be used to determine if an object is a structure or not:

```
<cfscript>
  Cart = StructNew();
  BillingFirstName = "Selene";
  Test1  = IsStruct(Cart);  // returns "yes"
  Test2 = IsStruct(BillingFirstName); // returns "no"
</cfscript>
```

The StructIsEmpty() function can be used to determine whether or not a structure contains any keys:

```
<cfscript>
  Cart = StructNew();
  BillingStruct.FirstName = "Selene";
  ShippingFirstName = "Dave";
  Test1 = StructIsEmpty(Cart);   // returns "yes"
  Test2 = StructIsEmpty(BillingStruct);  // returns "no"
  Test3 = StructIsEmpty(ShippingFirstName);  // error
</cfscript>
```

The values of Test1 and Test2 aren't surprising. However, setting the value of Test3 will return an error because the object being passed to the function isn't a structure.

You're probably very familiar with the IsDefined() function to test for the existence of an object in ColdFusion. The variable passed in can be of any data type, so the function is extremely useful. There's also a function called StructKeyExists() that's used only to test the existence of a key in a particular structure:

```
<cfscript>
  Cart = StructNew();
  BillingStruct.FirstName = "Selene";
  ShippingFirstName = "Dave";
  Test1 = StructKeyExists(Cart, "CCNum");  // returns "no"
  Test2 = StructKeyExists(BillingStruct, "FirstName");  // returns "yes"
  Test3 = StructIsEmpty(ShippingFirstName);  // error
</cfscript>
```

Again, the setting of Test3 will return an error because ShippingFirstName isn't a structure.

By Reference Versus by Value

Objects and variables are passed by one of two ways: reference or value. When an object is passed by value, a new object is created that has the same value as the original. When an object is passed by reference, the new object is simply a pointer to the value of the original object stored in memory. Most of us are used to variables that are passed by value. For example:

```
<cfset BillingFirstName = “Selene” />
<cfset ShippingFirstName = BillingFirstName />
<cfset BillingFirstName = “Dave” />
<cfoutput>#ShippingFirstName# - #BillingFirstName#</cfoutput>
```

The output of this code would be “Selene - Dave” because the value of BillingFirstName – in this case “Selene” – is passed to ShippingFirstName as its value. Even after BillingFirstName is changed to “Dave” the value of ShippingFirstName doesn’t change.

The same is not true for objects that are passed by reference. If the example above was passed by reference as opposed to value, the output would be “Dave - Dave” because the value of BillingFirstName wouldn’t be copied to the value of ShippingFirstName, rather a reference (or pointer) to BillingFirstName would be saved to ShippingFirstName. This means that when the value of BillingFirstName changes, the value of ShippingFirstName is also changed because ShippingFirstName just points to the value of BillingFirstName.

A good example of this is the billing and shipping information stored for an online purchase. Typically, the purchaser fills out his billing information and the next page contains the same information pre-populated in the shipping form. The purchaser can modify any information and then proceed with the checkout process. The following code would take the information entered into the billing form and then copy it to the shipping structure:

```
<cfscript>
  Session.BillingStruct = StructNew();
  Session.BillingStruct.FirstName = Form.FirstName;
  Session.BillingStruct.LastName = FormLastName;
  Session.ShippingStruct = Session.BillingStruct;
</cfscript>
```

On the shipping information page, the form is populated with the information in the shipping structure, which is a reference to the billing structure. When the shipping information is posted, the structure can be updated:

```
<cfscript>
  Session.ShippingStruct.FirstName = Form.FirstName;
  Session.ShippingStruct.LastName = FormLastName;
</cfscript>
```

If the purchaser changed no information, there’s no problem.

However, any changes made will actually be made to Session.BillingStruct since Session.ShippingStruct is just a pointer. That means the billing information may no longer be correct.

To avoid this potential pitfall, the StructCopy() function can be used:

```
<cfscript>
  Session.BillingStruct = StructNew();
  Session.BillingStruct.FirstName = Form.FirstName;
  Session.BillingStruct.LastName = FormLastName;
  Session.ShippingStruct = StructCopy(Session.BillingStruct);
</cfscript>
```

When StructCopy() is used, the new structure will no longer be a pointer to the existing one, rather it will be a duplicate of the original structure. If a change is now made to Session.ShippingStruct, the values in Session.BillingStruct won’t be affected and vice-versa.

Looping Over a Structure

There are two easy ways to loop over structures in ColdFusion: using the cfloop tag and a for loop in cfscript:

```
<cfscript>
  BillingStruct = StructNew();
  BillingStruct.FirstName = “Selene”;
  BillingStruct.LastName = “Bainum”;

  for(key in BillingStruct) {
    writeOutput(key & “ = “ & BillingStruct [key] & “<br />”);
  }
</cfscript>

<cfloop collection="#BillingStruct#” item="key">
  <cfoutput>#key# = #BillingStruct[key]#<br /></cfoutput>
</cfloop>
```

Both the for loop and the cfloop examples above will return exactly the same results. The for loop method is great if you’re only performing actions in the loop that can be done in cfscript. The following example will copy all of the fields in a form post to another structure:

```
<cfparam name="Cart.BillingStruct” default="#StructNew()#” />
<cfscript>
  for(field in Form) {
    Cart.BillingStruct[field] = Form[field];
  }
</cfscript>
```

Why wouldn’t you just set Cart.BillingStruct equal to StructCopy(Form) you ask? If Cart.BillingStruct already existed and had keys, the StructCopy() function would overwrite the existing structure, whereas looping over the form and setting each field individually retains any previously existing Cart.BillingStruct keys that may not exist in the Form structure.

Combining Arrays and Structures

Structures can be extremely useful when used in combination with other structures and arrays. While structures can only be one dimension, the value of a key can be an array or even another structure, thus appearing to be multi-dimensional.

Throughout this article I’ve referred to a Cart structure that would hold the information from a shopping cart. Listing 1 provides a peek at a fuller version of this structure.

The first part of this example just sets form variables that would be similar to those passed by a shopping cart. The second part of the example creates a complex structure of structures and arrays of structures to hold all the information. What can you do with this structure? Pass it to a function!

Structures with Components and Functions

One of the most useful applications for structures that I've found is using them in conjunction with ColdFusion Components (cfcs). One of the drawbacks with a component is declaring every possible argument that may be passed to a function. If you're creating a function to insert a record with 20 columns, you may have to declare 20 arguments. However, if you pass a structure of those 20 elements into the function, you only have to declare one argument – the structure! This is also very helpful when it comes to adding and removing values – you don't have to update your argument declarations and your function calls – you only have to update the elements in the structure. The drawback, however, is that you can't force a structure element to be required or of a certain type without additional logic – validation that declaring the arguments of a function lets you do quite easily.

Using the shopping cart example above, it would be very difficult – not to mention annoying – to pass all the individual arguments into a function. Instead, the example in Listing 1 shows how to call the function passing in the structure and how the function itself would process the structure:

While not a complete shopping cart processing example, you can see how you can take a complex structure and process it by accessing particular elements and looping over others dynamically. Structures like this greatly clean up as well as modularize your applications.

While it would be impossible to list all structure uses and functions in a single article, you should now know what you need to start using structures to help make your applications more modular and efficient.

About the Author

Selene Bainum is a software architect at INPUT. She has been a ColdFusion and SQL developer for over 10 years and runs www.webtricks.com.

selene@webtricks.com

Listing 1

```
<cfscript>
  // Create form variables
  FORM.BillingFirstName = "Selene";
  FORM.BillingLastName = "Bainum";
  FORM.ShippingFirstName = "Dave";
  FORM.ShippingLastName = "Bainum";
  FORM.ItemCount = 2;
  FORM.ItemID_1 = 2;
  FORM.ItemPrice_1 = 12.00;
  FORM.ItemQuantity_1 = 1;
  FORM.ItemID_2 = 12;
  FORM.ItemPrice_2 = 9.99;
  FORM.ItemQuantity_2 = 3;
  FORM.CCNum = "xxxx";
  FORM.ExpDate = "12/06";

  // Convert form to structure
  Cart = StructNew();
  Cart.BillingStruct = StructNew();
  Cart.BillingStruct.FirstName = FORM.BillingFirstName;
  Cart.BillingStruct.LastName = FORM.BillingLastName;
  Cart.ShippingStruct = StructNew();
  Cart.ShippingStruct.FirstName = FORM.ShippingFirstName;
  Cart.ShippingStruct.LastName = FORM.ShippingLastName;
  // Create an array to hold the items
  Cart.Items = ArrayNew(1);
  // Loop over the number of items in the cart
  for (i = 1; i LTE FORM.ItemCount; i = i + 1) {
    // Create a structure for each item
    Cart.Items[i] = StructNew();
    Cart.Items[i].ItemID = FORM["ItemID_" & i];
    Cart.Items[i].Price = FORM["ItemPrice_" & i];
    Cart.Items[i].Quantity = FORM["ItemQuantity_" & i];
  }
  Cart.CCNum = FORM.CCNum;
  Cart.ExpDate = FORM.ExpDate;
</cfscript>
```

Listing 2

```
<cfscript>
  // Call the function and pass in the Cart structure
  ProcessCart(CartStruct = Cart);
</cfscript>

<!--- Define the function. --->
<cffunction name="ProcessCart" access="remote" returntype="string">

  <!--- Define the argument. --->
  <cfargument name="CartStruct" type="struct" required="yes" />

  <cftransaction>

  <!--- Insert Cart info. --->
  <cfquery ...>
  INSERT INTO Cart(PurchaseDate, CCNum, ExpDate)
  VALUES (#Now()#, '#Arguments.CartStruct.CCNum#', '#Arguments.
CartStruct.ExpDate#')
  </cfquery>

  <!--- GetID. --->
  <cfquery name="GetID" ...>
  SELECT MAX(CartID) AS ThisID FROM Cart
  </cfquery>
  <cfset CartID = GetID.ThisID />

  <!--- Insert billing info. --->
  <cfquery ...>
  INSERT INTO CartBilling (CartID, FirstName, LastName)
  VALUES (#CartID#, '#CartStruct.BillingStruct.FirstName#', '#Cart-
Struct.BillingStruct.LastName#')
  </cfquery>

  <!--- Insert the shipping info. --->
  <cfquery ...>
  INSERT INTO CartShipping (CartID, FirstName, LastName)
  VALUES (#CartID#, '#CartStruct.ShippingStruct.FirstName#', '#Cart-
Struct.ShippingStruct.LastName#')
  </cfquery>

  <!--- Loop over the items and insert. --->
  <cfloop from="1" to="#ArrayLen(Arguments.CartStruct.Items)#"
index="i">
    <cfquery ...>
    INSERT INTO CartItem (CartID, ItemID, Price, Quantity)
    VALUES (#CartID#, #Arguments.CartStruct.Items[i].ItemID#, #Argu-
ments.CartStruct.Items[i].Price#, #Arguments.CartStruct.Items[i].
Quantity#)
    </cfquery>
  </cfloop>

  </cftransaction>

</cffunction>
```

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

A New Vision for ColdFusion

The language of innovation

By Hal Helms

During a recent conversation between Mike Britton, Brian Kotek, and myself, we were discussing the features that we'd like to see in ColdFusion 8. (A podcast of this discussion can be found at helmsandpeters.com.) I'd like to share with you some thoughts on the topic. Much that follows is taken from a talk I gave last year at the Cfobjective conference. It lays the groundwork for a vision for ColdFusion that concludes this article.

Marketers speak of the importance of a product's "positioning." The concept of positioning gained wide acceptance some 25 years ago in the book, Positioning: The Battle for Your Mind. With that book, the authors, Al Ries and Jack Trout, rocked the business world.

Ries's and Trout's thesis is that prospective buyers of a product – and that product can be popcorn, presidents, or programming languages – identify brands by a single word or concept. That word in your prospect's mind is your product's position. According to Ries and Trout, people are so overloaded with information and with the claims of products that marketing messages are distilled down to a single word or, at most, a short phrase.

Let's see if we can identify some commonly held positions for a few well-known brands. See what concept each of these brands occupies in your mind: Volvo, Ritz-Carlton, Haliburton.

For most of us, Volvo conjures up an image of safety; Ritz-Carlton, an image of luxury; and Haliburton, an image of corporate greed. Once established, a brand's position within a prospect's mind is remarkably stable. Marketers who understand the power of positioning are very careful about avoiding counter-positioning – sending a message that differs from the brand's established position. Years ago, Porsche decided to "extend" their brand by producing a reasonably priced car, the Porsche 914. The product was a failure. Why? To most people, a Porsche is a fast, sexy, and expensive car. The 914 was pure counter-positioning.

"Line extension," the strategy employed by Porsche, is very popular among business executives. After all, they reason, why not leverage the strength of our brand and appeal to a different market segment – people who would like a Porsche (who want to associate themselves with that position) but can't afford the brand's pricier offerings?

At best, the market ignores counter-positioning claims. This was the case with Porsche – very few people bought the car. At worst, though, a counter-positioning message confuses people and dilutes the strength of the brand. Chevrolet, at one time, held the position of a modestly priced commuter car. But through line extension – everything from the Nova to the Corvette – Chevrolet has undermined that position. What position does it hold now? As Chevrolet's dwindling sales show, it has no strong position in prospects' minds.

An example of a brand that understands the power of positioning is Toyota. When the company wanted to appeal to a different market segment, people who wanted a luxury car, they created an entirely new brand: Lexus. They established an entirely new brand that could lay claim to the "luxury car" position. The strategy was highly successful: Toyota is a reliable, mid-priced car. Lexus is an expensive, luxury car.

Let's move from the world of cars to one closer to us. What positions do these computer languages hold in the minds of IT executives? Java, COBOL, Ruby, LISP. For most, Java means "enterprise"; COBOL means "legacy"; and LISP means "esoteric". We can see from COBOL that positions can change over time. COBOL was not always viewed so negatively, but times change and, sometimes, peoples' perceptions change with the times.

Now, while you're still inside those executive's heads – and while you're marveling at just how much space there is in there – I want you to think about how those managers would answer this question: What is ColdFusion's position? What one word describes it, not from your point of view, but from theirs?

I've asked this question of several executives. Among words like "easy" and "simple," perhaps the most-commonly used term is "lightweight." And this is a positioning problem. In the world of IT, lightweight is a pejorative term. For IT managers, CIOs, and CTOs, who are very susceptible to herd mentality, Java and C# hold the positions of being industrial-strength, enterprise-grade languages. ColdFusion, alas, does not.

We've all heard, and been annoyed by, the raps against ColdFusion: it's not secure, it's not scalable, etc. And we know that's simply not true. So it might seem that we should concentrate our efforts on showing that ColdFusion is perfectly capable to take on enterprise-mission applications, which we know it to be.

Anyone who's heard the familiar argument that "ColdFusion is Java" or "ColdFusion is the fastest way to build Java applications" will recognize this attempt to set the record straight. Arguments like "ColdFusion is Java" are logical arguments, but positions are held on a deeper, gut-instinct level and, once held, are deeply persistent. It's not that the argument fails so much as the argument is ignored.

Napoleon, the great general, was once asked, "Sir, whose side do you think God is on?" His reply is just as applicable for positioning as it is for warfare. He said, "I have observed that God is most often on the side with the biggest battalions." Apparently, Napoleon is right. In one study by Ries and Trout, 25 different categories of products were examined to see which brands were leaders. The study then looked at the same categories 60 years later. Twenty of those 25 brands still held the number one position.

My question this is morning is this: How do we win this battle for people's minds as it pertains to ColdFusion? Perhaps we should ask another question: Why do we care about the fate of ColdFusion? After all, we learned ColdFusion and we can learn another language. Why bother engaging in a battle at all? I believe it's the same reason that Apple users are so fanatic about their machines and their OS and refuse to capitulate to Windows. It's the same reason that Firefox developers were too foolish to realize that they had no chance against the enormous resources Microsoft committed to IE.

In both of these cases, the side with the smaller battalions were fueled by a powerful idea: that they were right. Many of us do know multiple languages, yet we continue to prefer ColdFusion. While I believe that we are right – that ColdFusion has a vital role to play in the enterprise – we must be strategic in the way we approach the IT executives who make decisions about which languages will be used.

Let me give you an example of one way to approach your CIO. It's done in the form of an e-mail.

To: CIO
From: Me
Subject: ColdFusion in the enterprise
Yo.

There's been lots of talk lately about using Java for everything. Web stuff. Server-side stuff. You know.

I have two words for this: dumb idea. ColdFusion rocks, dude! Anything you can throw at it, ColdFusion can handle.

Jerry, our company Java weenie, said the other day in a meeting that ColdFusion isn't scalable and that it's not secure. That's bogus.

First, it's just not true. Scalability and security aren't magic features of a language. They're something good developers design for using a language.

Second, if the argument is Java is more scalable and more secure, then – newsflash! – ColdFusion is Java!

Hopefully, you can see now that your idea that maybe we should standardize on Java is just wrong. If you don't believe me, here are some books and articles you ought to read…

And then I proceed to offer a reading list to our CIO to back up my position. Good move, no?

No. If anything is more certain than death and taxes, it's this: people don't change their minds when told they're wrong. What do they do? They dig in. And then the side with the bigger battalions does win.

If people are unwilling to admit that a decision they made previously was wrong, they just may be able to make a new decision. To assist in this, we need new strategy. In the battle for minds, smaller competitors often go for the frontal attack. Its virtues: straightforwardness and simplicity. Its weakness: it seldom works.

Some years ago, RC Cola made a concerted effort to take on Coke and Pepsi. The CEO rallied the troops with a speech in which he said, "RC is, in every way, as good as Coke or Pepsi. But our product, every bit as good, is cheaper!" While this played well among the loyal RCers, it fared rather less well among the general public. This is the normal outcome when attacks are launched against a perceived weakness on the part of a leader. After all, when was the last time you bought an RC Cola?

Attacks against perceived weaknesses typically fail. Attacking a leader's strength, on the other hand, may just afford a real possibility for prospects to make new decisions. In the '70s, Avis was a distant second to the market leader, Hertz. Avis didn't choose to say, "We're better than Hertz." They knew that saying such a thing was tantamount to telling prospects, "You were wrong in your decision to think of Hertz as the leader." People don't like to be told they're wrong, and when told, they seldom change their minds.

Instead, Avis came up with a brilliant campaign, advertising themselves as not the leader. "We're No. 2," Avis advertised. In one ad, Avis offered prospects this to consider: "Rent from Avis. The line at our counter is shorter." They didn't attack the leader's weakness. They attacked the leader's strength. "We're just as good as Hertz – and cheaper" would have been a "me too" attack on a perceived weakness.

The brilliance of the campaign was that the very nature of Hertz's strength – of being the market leader – was susceptible to attack. And Avis' results were as different from RC Cola's as was its strategic attack. While I can't remember the last time I had an RC Cola, over the years since their campaign, I've rented quite a few cars, and a good number of those have been from Avis.

What does all this have to do with ColdFusion? Who are the Coke and Pepsi of enterprise development languages? Java and C#. The "ColdFusion is Java" argument is a "me too" strategy that lacks the power to help prospects make a new decision. What, though, if we attack, not Java and .NET's weakness, but their strength? What are the strengths of the J2EE and .NET platforms? They are perceived as secure, robust, and enterprise-capable. To use an analogy, these languages are to applications what steel I-beams and concrete are to buildings.

Certainly, if you want to build an true enterprise-class application – something that can scale to thousands of users, something that is mission-critical – something like J2EE is a pretty hard choice to argue with. But the decision doesn't end there. The philosopher and physicist, Eli Goldratt, has developed a system of thinking he refers to as the Theory of Constraints (or TOC). Goldratt stresses what he refers to as "system thinking" and his work was greatly influenced by the work of Edwards Deming, the man famous for teaching quality to the Japanese.

TOC states that within any system, there is one, and only one, constraint operating at any point. The mission of the manager is to lift that sole constraint on the system. Let's apply TOC to the case of enterprise development. What is the constraint in the system? We know that a great deal of enterprise software is deemed by the only jury that counts, the users of that system, as a failure. But why? What is the failure point? If the failure point is an inability to scale, or a lack of security, or is simply not robust enough, then J2EE and .NET may well be the answer and all the "ColdFusion is Java" talk in the world will fall on deaf ears.

Are any of these the constraint of the system? Ask users why a recent software implementation goes unused and they'll likely tell you this: the application doesn't do what we need it to do. This is the failure point. Users aren't getting what they need. How bad is the problem? Dr. Ralph Young of Northrup Grumman Information Technology division estimates that 85% of defects in developed software originate in failed requirements.

How do we discover user requirements? For years, I've been preaching the virtues of designing the user interface first – before the database schema, before the application modules, before the domain model. The reason is simple: to the user, the interface is the application. Alan Cooper, the father of Visual Basic, has written two books that count the user experience as the most important aspect of software. Jason Fried, the CEO of 37 Signals, the company behind BaseCamp and Ruby on Rails, recently said in an interview: "We do things a little differently: we design the user interface first."

The reason for the "user interface first" movement is simple: traditional methods of discovering user requirements just don't work. Incontrovertible evidence compels us to recognize that such techniques fail at uncovering what users really want.

Concentrating on the user experience, on the other hand, fosters innovation, as those of us who practice the technique know. Innovation occurs when a system is in place that allows for rapid response to user requirements, where applications are flexible enough to meet unanticipated – and unanticipatable – demands of users in the real world, where versions can be rolled out rapidly by a system agile enough to provide for quick responses. It's a world in which "lightweight" is a key to innovation and, perhaps, ultimately, to survival. It's a world perfectly suited to a language such as ColdFusion.

I used the analogy of concrete and steel earlier when talking about J2EE and .NET. Their problem is not their weakness, but their strength. If you know exactly what you are going to build and that system is going to remain stable (i.e., there will be no innovation), then Java or C# are great choices. But if the constraint on the system is the lack of ability to innovate, to (1) discover requirements and (2) respond rapidly to changing customer needs and requirements, then applying J2EE or .NET to the user interface may just make the situation worse. Their strength is their weakness.

Let me give you an example of how such business agility can work in the real world. A corporation I worked with was struggling with the idea of standardizing on J2EE for all application development. The problem was that most of the applications that they used on a daily basis were written in ColdFusion. They knew by experience that the Java team took much longer to deliver a similar application and that the application was more expensive to develop. Further, it was harder to change. Yet, they wanted to standardize on a single way to do certain things.

My recommendation to them was that they continue fostering innovation with ColdFusion, yet roll up those standardized aspects into Java components, once it was clear exactly what those components should do. An MVC framework such as Fusebox, Mach-II, or Model Glue made it a simple matter to swap out the domain model ColdFusion components for domain model Java components if and when it made sense to do so.

I'm happy to report that for 18 months, the company has followed this plan and reports great success doing so. Their "business agility quotient" increased substantially. A side benefit is that both their ColdFusion and Java developers are happier in their respective roles. Discovery and innovation happen within ColdFusion. Standardization and consensus occur within Java. And both work well together.

I think that a remarkable opportunity for propelling ColdFusion into the enterprise exists. After years of languishing, following the dot-com boom, the Web is suddenly hot again. "Web 2.0," whatever that term may actually mean, has caught the attention of the broader business market. And we stand to benefit: demand for Web programmers is outstripping the supply. Programmer salaries are rising. The old adage applies: a rising tide lifts all boats.

What of ColdFusion specifically? What part will it play in the resurgence of the Web? How can it best leverage these trends? Some ColdFusion programmers want ColdFusion to compete head-on with Java and C#. Give us interfaces. Give us abstract classes. Give us constructors. Make ColdFusion more like Java. It's the RC Cola strategy all over again.

What if ColdFusion were, instead, to focus all its efforts on empowering programmers to create fantastic user interfaces quickly and easily? This has always been a strength of ColdFusion. What if, to borrow an analogy from the Extreme Programming camp, "the dial was turned to 11"? What would this look like? What would ColdFusion 8 offer?

If this strategy were adopted, we would likely see the integration of Flex components with ColdFusion tags and functions. ColdFusion would offer the ability to use AJAX seamlessly. Flash widgets could easily be added to interfaces.

There's a temptation to want it all, to have a language that offers great user interface capabilities and all the server-side features of Java or C#. It's the classic appeal of line extension, a strategy that repeatedly, predictably fails. Are we really willing to believe that this time it will be different? I am not. I hope that Adobe will recognize the unique strengths of ColdFusion and augment these to lift the great constraint facing developers – the difficulty of understanding and quickly supplying the needs of users.

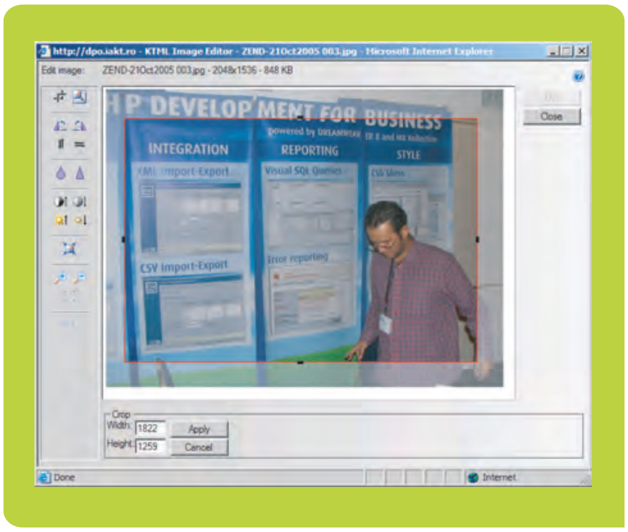
If the bloggers who want ColdFusion to be more like Java have their way, ColdFusion will just be another general-purpose language. Wanting to have it all is a fantasy. In the real world, defining what a brand is also means defining what a brand is not. Every R&D dollar that goes into making ColdFusion more like Java is a dollar that is not available for making ColdFusion the best choice for creating great user interfaces. We've seen what happens when a brand like Chevrolet tries to offer it all: they fail. Let's encourage Adobe to focus all their efforts on positioning ColdFusion as the language of innovation.

hal@halhelms.com

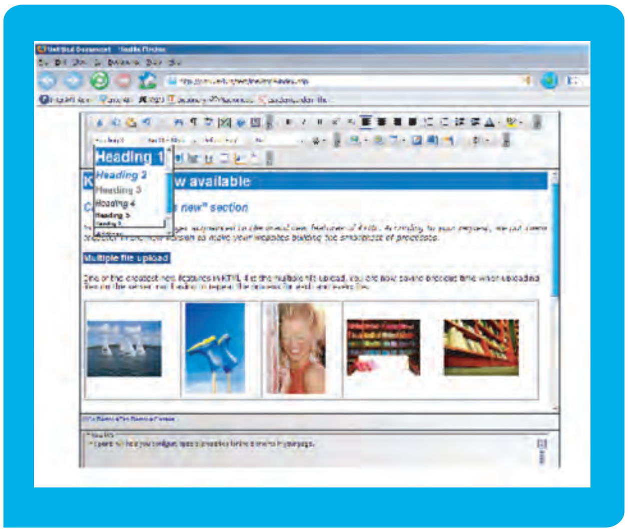
KTML4 Word editing in browser

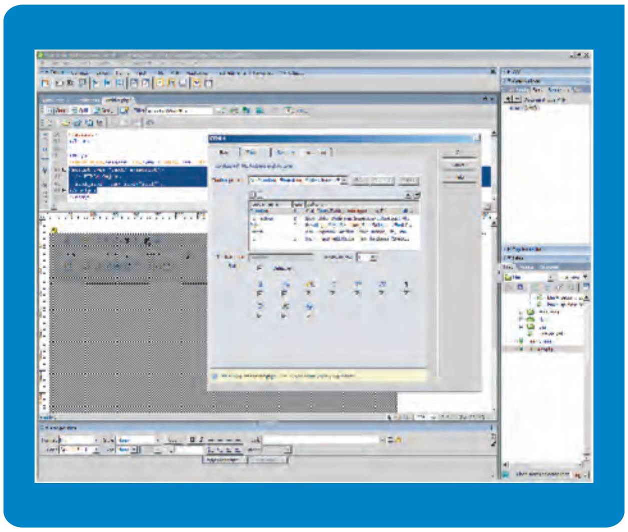
Along with the most affordable **UNLIMITED** licence

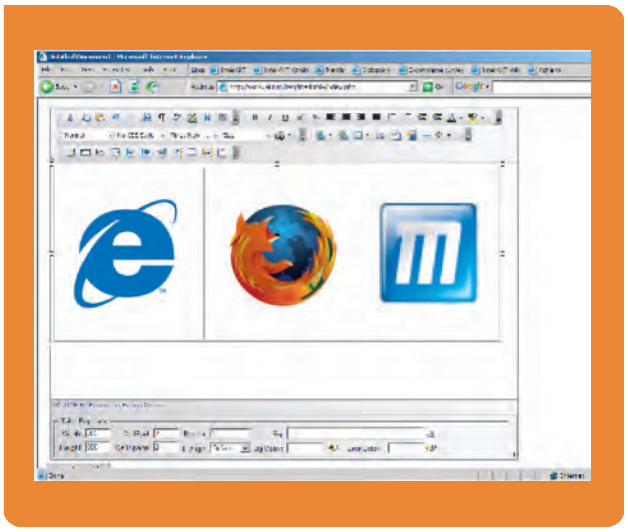
Revolutionary Image Editor



Word-like visual CSS styles







Fast Dreamweaver integration

Wide browser compatibility

Instant paste from Word
Incredible speed
Easy to use Word-like toolbars
Improved CSS authoring
Remote File Explorer
XHTML 1.1 compliant

Multiple file upload at once
HTML Table Editor
Support for multimedia (Flash, Avi)
Documents management (.doc, .pdf)
Page templates
WAI compliant

Please visit **www.interaktonline.com/ktml4/** for details

work smart

Interakt

Other companies in this magazine spent a lot of time on pretty ads. As you can see, we did not. We spent our time hiring the best people and training them to deliver outstanding support for your website. We spent our time building a state of the art datacenter and staffing it with people who care about your website like it's their own. Compassion, respect, credibility, ownership, reliability, "never say no," and exceed expectations are words that describe our service philosophy. From the first time you interact with us, you'll see what a difference it really makes. And you'll also forgive us for not having a pretty ad.

HostMySite.com

WEB HOSTING • MANAGED DEDICATED SERVERS • COLOCATION • VPS • ECOMMERCE • BLOGGING • EMAIL