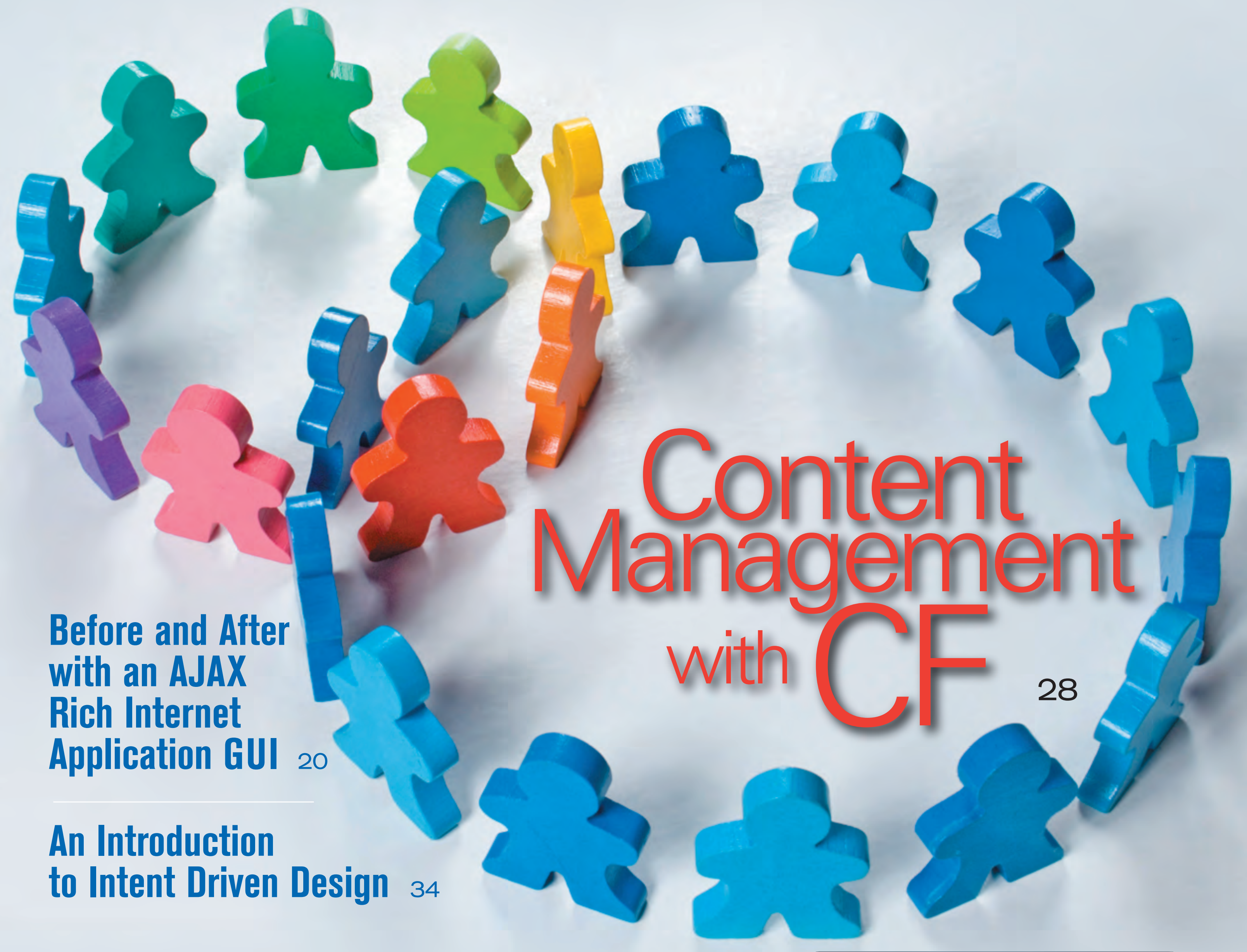
ADOBE FLEX 2: ADVANCED DATAGRID PG 7

ColdFusion Developer's Journal

ColdFusionJournal.com

November 2006 Volume:8 Issue: 11



Content Management with CF 28

Before and After with an AJAX Rich Internet Application GUI 20

An Introduction to Intent Driven Design 34

Presorted
Standard
US Postage
PAID
St. Croix Press

Create Tomorrow's Rich Internet Applications Today

The combination of the rich user experience of Adobe® Flex™ 2 and the productivity of Macromedia® ColdFusion® MX 7 provides a fast, easy way to create rich Internet applications.

Adobe has created a deep level of integration and productivity tools for Flex and ColdFusion making it easy for ColdFusion developers to get started and become productive building Flex applications.

Download a free trial today at
www.adobe.com/products/coldfusion/flex2

Better by Adobe™

Adobe®

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

BlueDragonCertified V.I.P. Hosting Partner

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

Making Heads or Tails of the Many CF Conferences

By Simon Horwith

ColdFusion developers today are presented with more options than ever before – more developer tools, more conferences and learning resources, and more development choices in the form of frameworks and language features.

With the recent passing of yet another MAX conference, I want to take the opportunity to explore many of the decisions in educational community activities (conferences) that ColdFusion developers face today. I realize that for many developers, attending every conference is not a realistic option. Obviously, location and timing can play a large part in the decision of whether to attend a conference. But what if timing and cost aren't an issue? How do you decide which conference(s) to attend, then?

What should you look for in a conference? The first criterion has to be subject matter. Do you have interest in learning about technologies and products other than ColdFusion? Why would a CF developer want to attend a conference in which ColdFusion is not the only topic? Some CF developers are also designers and/or develop in other programming languages as well as CF – they might also want to learn about other tools and technologies. Another reason is that some developers may be so proficient in CF that they won't learn as much from attending sessions on CF topics, but they're interested in learning something new and have a lot to learn by attending topics on other technologies.

Another very important criterion is the speaker line-up and topics that they'll be discussing. There are many folks in the ColdFusion community who speak at conferences and user groups – some are great speakers and some… well, let's just say that some are not. A great speaker can give a presentation on any subject and it will be enjoyable. There aren't many of them out there, but certainly, there are some speakers worth seeing talk about any topic at all. I don't recommend getting your hopes up about seeing someone speak unless you know they have a lot of experience presenting and/or teaching.

The last thing I recommend taking into consideration is attendee feedback from prior events. While location, timing, subject matter, and speaker line-up do say a lot about an event, there are certain ambiguous characteristics that all the facts and figures cannot reveal. Some are aspects of an event that aren't deliberate – they are a product of attendee/speaker attitude and expectation, and have to do with the atmosphere. Other aspects are how well the event is run – that can make a huge difference for attendees. Even something as simple as catering can be the deciding factor. There is no substitute for past attendee testimony, but keep in mind that different people have different experiences, different likes and dislikes, and that they may also be biased or let other opinions influence their feedback.

My advice would be to try each year to attend at least one large conference, one grassroots event, and to regularly attend your local user group meetings if at all possible. The development community needs these events, all of these events, to bring us all together in order to explore new ideas and share our discoveries with one another.

About the Author

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

simon@horwith.com

ColdFusion Developer's Journal

editorial advisory board
Jeremy Allaire, *founder emeritus*, **macromedia, inc.**
Charlie Arehart, *CTO*, **new atlanta communications**
Michael Dinowitz, **house of fusion, fusion authority**
Steve Drucker, *CEO*, **fig leaf software**
Ben Forta, *products*, **macromedia**
Hal Helms, *training*, **team macromedia**
Kevin Lynch, *chief software architect*, **macromedia**
Karl Moss, *principal software developer*, **macromedia**
Michael Smith, *president*, **teratech**
Bruce Van Horn, *president*, **netsite dynamics, LLC**

editorial

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

executive editor
Nancy Valentine nancy@sys-con.com

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

production

lead designer
Abraham Addo abraham@sys-con.com

art director
Alex Botero alex@sys-con.com

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

editorial offices

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

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

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

For promotional reprints, contact reprint coordinator Megan Mussa, megan@sys-con.com.
SYS-CON Publications, Inc., reserves the right to revise, republish and authorize its readers to use the articles submitted for publication.
All brand and product names used on these pages are trade names, service marks, or trademarks of their respective companies.

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

SYS-CON MEDIA

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

group publisher
Jeremy Geelan jeremy@sys-con.com

advertising

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

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

customer relations

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

sys-con.com

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

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

accounting

financial analyst
Joan LaRose joan@sys-con.com

accounts payable
Betty White betty@sys-con.com

subscriptions

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

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

PART 2

Adobe Flex 2: Advanced DataGrid

Drop-in RadioButtonGroupBox; runtime computed styles vs itemRenderers; masked input and numeric input controls

By Yakov Fain, Victor Rasputnis, & Anatole Tartakovsky

In Part 1 (CFDJ, Vol. 8, issue 10) we introduced the destination-aware grid, formatters, and renderers. In this article we are continuing our discussion about datagrid renderers and…

RadioButtonGroupBox as Drop-In Renderer

We can apply similar techniques to RadioButton controls. The following code snippet suggests how the group of RadioButton controls can be used as a drop-in item renderer (and editor). Instead of an onValue/offValue pair, we are introducing an array of options (we could have gone further and upgraded <fx:options> to <fx:dataProvider>, which is similar to the ButtonBar and LinkBar controls):

```
<fx:DataGridColumn dataField="STATUS" width="300"
headerText="Status" rendererIsEditor="true"
itemRenderer="com.theriabook.containers.RadioBut-
tonGroupBox">
  <fx:options>
    <mx:Array id="options">
      <mx:Object data="A" label="Active"/>
      <mx:Object data="T" label="Terminated"/>
      <mx:Object data="L" label="On leave"/>
     </mx:Array>
  </fx:options>
</fx:DataGridColumn>
```

To support this use case, we need to build the renderer class and make the DataGridColumn pass it the array of options. The latter can be done by adding the following options setter to our DataGridColumn:

```
package com.theriabook.controls.dataGridClasses{
 .  .  .  .  .  .
 public class DataGridColumn  extends mx.controls.
dataGridClasses.DataGridColumn {
  .  .  .  .  .  .
  public function set options(val:Array):void {
    if (itemRenderer) UIClassFactory(itemRenderer).
properties = {options:val};
  }
 }
}
```

Now let's build the renderer. By definition, to be an item renderer, the component has to implement the IListItemRenderer interface. To qualify as drop-in, a component has to additionally implement IDropInListItemRenderer. The Standard CheckBox implements both interfaces. Because of that, when we extended CheckBox in the previous article, we did not have to mention a single implementation and just merrily used data and listData at our convenience.

This is not the case now. Had RadioButtonGroup been at least a UIComponent, we'd need to implement the IDataRenderer and IDropInListItemRenderer interfaces and be done. But RadioButtonGroup is not even a DisplayObject, so we will base our renderer on mx.containers.Box with RadioButtonGroup embedded:

```
  private var group:RadioButtonGroup=null ;

  public function RadioButtonGroupBox() {
   super();
   group = new RadioButtonGroup();
  }
```

Having RadioButtonGroup is just the beginning. Whenever our component gets assigned any options, we need to translate them into a set of RadioButton controls.

TABLE OF CONTENTS

Each RadioButton will be added as a child of the renderer (container):

```
private var _options:Array=null;
public function set options(opt:Array):void {
 var i:int;
 . . . .
 _options=opt;
 for (i= 0; i < opt.length; i++) {
  var rb:RadioButton = new RadioButton();
  rb.label = opt[i].label;
  rb.value = opt[i].data;
  addChild(rb);
 }
}

override public function addChild(child:DisplayObject):DisplayObject {
 if (child is RadioButton) {
  (child as RadioButton).group = group;
  group.addInstance(child as RadioButton);
 }
 return super.addChild(child);
}
}
```

Please note how subscribing a RadioButton to the group is delegated to the overridden method addChild():

```
(child as RadioButton).group = group;
group.addInstance(child as RadioButton);
```

Had we done it directly in the options setter, there wouldn't be any need in addChild(). Why did we take the convoluted way? We did it to enable the potential use of RadioButtonGroupBox as a regular container, outside of the renderer context. In other words, no matter how a RadioButton gets added to the component – via options or not – it will get associated with the group.

Next, since we want the component to be a drop-in renderer, we need to implement the IDropInListItemRenderer interface so that the extra information about the hosting List is at our fingertips:

```
private var _listData:BaseListData=null;
public function get listData():BaseListData          {
 return _listData;
}
public function set listData(value:BaseListData):void          {
 _listData = value;
}
```

Once we have the listData, we can offer the following override of the data setter of IDataRenderer:

```
override public function set data(item:Object):void  {
 super.data = item;
 if( item!=null ) {
  group.selectedValue = item[DataGridListData(listData).dataField];
 }
}
```

Similarly, while implementing the property value, we again consider both use cases: the standalone component and the item renderer. In case of the item renderer, our component updates the underlying data:

```
public function get value():Object {
 return group.selectedValue;
}
public function set value(v:Object) : void {
 group.selectedValue = v;
 if (listData) {
  data[DataGridListData(listData).dataField] = group.selectedValue;
 }
}
```

Finally, how about capturing the selection of a radiobutton? Since we need to listen to the change event on the RadioButton-Group, we'll set up the listener right in the constructor method, handling the Event.CHANGE with the anonymous function:

```
public function RadioButtonGroupBox() {
 . . . .
 group = new RadioButtonGroup();
 group.addEventListener(Event.CHANGE,
  function event:Event):void {
   value = event.target.selectedValue;
  }
 );
}
```

The complete code of the RadioButtonGroupBox is presented in Listing 1. (Listings 1–13 can be downloaded from the online version of this article at http://coldfusion.sys-con.com.) See if you can discover the discrepancies between the listing and what we outlined in our snippets:

- We've added the text property, so we don't have to specify editorValue="value" in the item editor use case.
- We've made the properties text and value bindable by the "change" and "valueCommit" events; the dispatching of events is being done by an anonymous handler of the Event. CHANGE and setter of value correspondingly.
- We've allowed dynamic re-assigning of options by removing the existing dynamic RadioButtons prior to building the new ones from the options array.

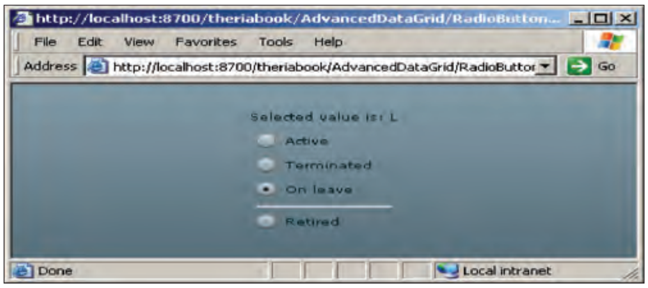
To take a standalone RadioButtonGroupBox for a spin, we wrote a small application – RadioButtonGroupBoxStandAloneDemo (see Listing 2). When you run the application, you can check that it's still a normal Box container that's enriched with an ability to dynamically create as many RadioButton controls as there are option values. We also squeezed in the property value, similar to the one in the ComboBox and CheckBox controls (see Figure 1).

Now let's test the itemRenderer/itemEditor scenario. Not that it's required, but we would prefer to lay out the radio buttons horizontally. Accordingly, we will create RadioButtonGroupHBox as a simple extension of the RadioButtonGroupHBox, taking care of the box's direction and adding a couple of padding pixels:

The screenshot of the test application RadioButtonGroupHBoxDemo is presented in Figure 2. As you may notice in Listing 4, we declared the entire DataGrid as editable, and marked the "status" column with rendererIsEditor="true":

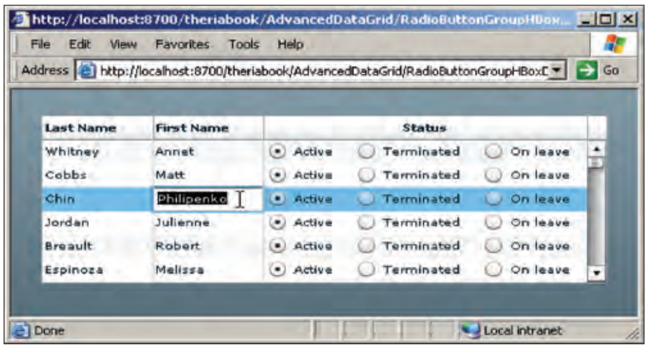
Computed Column Color

Getting back to the DataGrid, we will look at controlling the styles of the DataGridColumn: color, backGroundcolor, font and so on. We'll focus on defining the styles in such a way that they are re-evaluated along with the data changes. Many times we meet business requirements that call the styles to be different


http://localhost:8700/theriabook/AdvancedDataGrid/RadioButton...
File Edit View Favorites Tools Help
Address http://localhost:8700/theriabook/AdvancedDataGrid/RadioButto
Go
Done
Local intranet

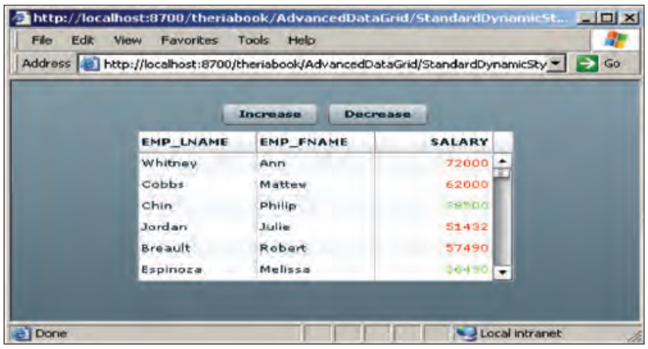
Selected value is: L
Active
Terminated
On leave
Retired

Figure 1: The RadioButtonGroupBoxStandaloneDemo


http://localhost:8700/theriabook/AdvancedDataGrid/RadioButtonGroupHBox...
File Edit View Favorites Tools Help
Address http://localhost:8700/theriabook/AdvancedDataGrid/RadioButtonGroupHBoxD
Go
Done
Local intranet

Last Name	First Name	Status		
Whitney	Annet	● Active	○ Terminated	○ On leave
Cobbs	Matt	● Active	○ Terminated	○ On leave
Chin	Philipenko	● Active	○ Terminated	○ On leave
Jordan	Julienne	● Active	○ Terminated	○ On leave
Breault	Robert	● Active	○ Terminated	○ On leave
Espinoza	Melissa	● Active	○ Terminated	○ On leave

Figure 2: RadioButtonGroupHBoxDemo


http://localhost:8700/theriabook/AdvancedDataGrid/StandardDynamicSt...
File Edit View Favorites Tools Help
Address http://localhost:8700/theriabook/AdvancedDataGrid/StandardDynamicSty
Go
Increase
Decrease
Done
Local intranet

EMP_LNAME	EMP_FNAME	SALARY
Whitney	Ann	72000
Cobbs	Mattew	62000
Chin	Philip	38500
Jordan	Julie	51432
Breault	Robert	57490
Espinoza	Melissa	36490

Figure 3: StandardDynamicStyleDemo

from row to row. Supposedly, we need to highlight high paid employees ($50K or more) in red, otherwise the salary value is shown in green. One solution is to use the in-line itemRenderer:

```
<mx:DataGridColumn   dataField="SALARY" textAlign="right">
  <mx:itemRenderer>
   <mx:Component>
    <mx:Label>
     <mx:Script>
   <![CDATA[
    override protected function updateDisplayList(unscaledWidth:
        Number,
           unscaledHeight:Number):void
    {
     super.updateDisplayList(unscaledWidth, unscaledHeight);
     if (data && listData) { // Check that we are in a List
      if (data.SALARY > 50000) {
       setStyle("color", "red");
      } else {
       setStyle("color", "green");
      }
     }
    }
    ]]>
     </mx:Script>
    </mx:Label>
   </mx:Component>
  </mx:itemRenderer>
</mx:DataGridColumn>
```

As a component base we've used Label, an immediate descendant of the UIComponent, because updateDisplayList(), an UIComponent method, would be out of reach for a standard DataGridItemRenderer based on UITextField. Alernatively, we could have achieved the same functionality with a more elegant binding expression syntax:

```
<mx:DataGridColumn   dataField="SALARY" textAlign="right">
 <mx:itemRenderer>
  <mx:Component>
   <mx:Label
      color="{data.SALARY&gt;50000?255*256*256:255*256}"
    >
   </mx:Label>
  </mx:Component>
 </mx:itemRenderer>
</mx:DataGridColumn>
```

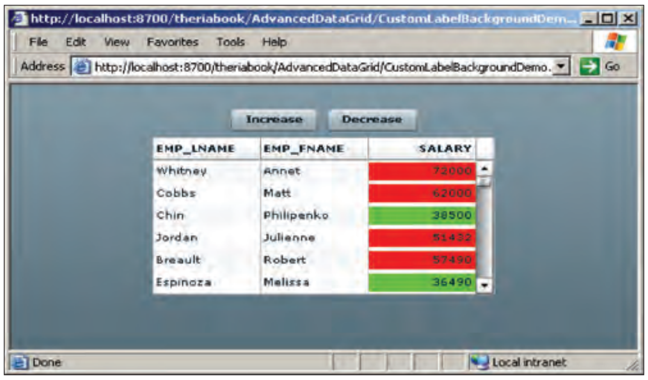
Listing 5 presents the complete code of the sample application StandardDynamicStyleDemo. In addition to the DataGrid, we have thrown in "Increase" and "Decrease" buttons, which allow us to modify the salary values in increments of 10K. When you run a StandardSynamicStyleDemo, you will see the picture shown in Figure 3.

Computed Column Background

We can't apply the same technique for a background color, 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


http://localhost:8700/theriabook/AdvancedDataGrid/CustomLabelBackgroundDem...
File Edit View Favorites Tools Help
Address http://localhost:8700/theriabook/AdvancedDataGrid/CustomLabelBackgroundDemo.
Go
Increase
Decrease
Done
Local intranet

EMP_LNAME	EMP_FNAME	SALARY
Whitney	Annet	72000
Cobbs	Matt	62000
Chin	Philipenko	38500
Jordan	Julienne	51432
Breault	Robert	57490
Espinoza	Melissa	36490

Figure 4: Background color computed with binding expression

because Label does not support the backgroundColor style. In comparison, TextInput does and, although, we could have resorted to TextInput instead of Label, the mere absence of backgroundColor does not seem reason enough to give up a lightweight Label in favor of the heavier TextInput. After all, the beauty of Flex is that the framework is extensible. Our custom Label extended with backgroundColor support is shown in Listing 6.

We have defined not one, but two styles – backgroundAlpha and backgroundColor – and we use both values with graphics. beginFill() inside the overridden implementation of updateDisplayList(). Once we add the Label.as to theriabook.swc and register it within the component manifest XML, the DataGridColumn can be redefined as is done in the following snippet:

```
    <mx:DataGridColumn    dataField="SALARY" textAlign="right">
     <mx:itemRenderer>
      <mx:Component>
       <fx:Label
        backgroundColor="{data.SALARY&gt;50000?255*256*256:255*256}"
       />
      </mx:Component>
     </mx:itemRenderer>
    </mx:DataGridColumn>
```

The difference between the above approach and the way we defined a similar DataGridColumn in Listing 5 is that now we compute backgroundColor instead of the color and use fx:Label instead of the Label from mx namespace. If you do the replacement and run the program, you'll see the picture as shown in Figure 4.

Despite the decent result, we still don't feel satisfied: employing the powerful mechanism of item renderers only for the purpose of managing styles seems to be design overkill. Let's speak our mind: we are after dynamic runtime styles, right? Wouldn't it be nice if we added an extra DataGridColumn attribute, called, say, runtimeStyles, where we could list all the styles and the abstract from the implementation. Here is an example:

```
    <fx:DataGridColumn    dataField="SALARY" textAlign="right"
formatString="money">
     <fx:runtimeStyles>
       <mx:Object
        backgroundColor="{function(item:Object):String {return (item.
SALARY&gt;50000)?'red':'green';}}"
       />
     </fx:runtimeStyles>
    </fx:DataGridColumn>
```

This approach would let developers concentrate on the substance rather than on the process. Let's make it happen.

Runtime Column Styles Unleashed

Here is the plan: we will upgrade the default itemRenderer of the fx:DataGrid from UITextField to our custom fx:Label by extending fx:DataGridColumn with an extra property – runtimeStyles. Finally, we will intercept data changes to an item renderer, whether default or not, to re-assign all runtimeStyles on each.

Here is how the constructor of the standard DataGrid assigns itemRenderer:

```
package mx.controls {
public class DataGrid extends DataGridBase implements IIMESupport
{
   public function DataGrid()
  {
      super();
          itemRenderer = new ClassFactory(DataGridItemRenderer);
          .  .  .  .  . .
   }
 }
}
```

We would have to fight the instant temptation to replace the assignment of the itemRenderer with:

```
  itemRenderer = new ClassFactory(com.theriabook.controls.Label);
```

Here is why: a ClassFactory instance is a "factory object," which is used to generate instances of another class (aka generator class) with the newInstance() method. According to our plan, we need to intercept data changes to any instance of the generator class item renderer. In other words, we'll need to listen to the event FlexEvent.DATA_CHANGE on every instance of the com.theriabook.controls.Label created by "the factory." Hmm, what could be simpler than adding the needed event listener to the controls? Although we are content with the fx:Label as a default renderer, by no means do we propose to take the power of the custom item renderers away. To make the mechanism of the runtimeStyles control agnostic, we would like to listen to FlexEvent.DATA_CHANGE on the instances of any generator class.

It only sounds difficult. After all, a ClassFactory is nothing but an implementation of the IFactory interface with a single property – properties and single method – newInstance(). So we can easily wrap a standard ClassFactory inside our custom one for the purpose of intercepting the newInstance() call. We will call our wrapping class factory the UIClassFactory:

```
 function DataGrid() {
  super();
  itemRenderer = new UIClassFactory(ClassFactory(com.theriabook.controls.
Label));
 }
```

The constructor of the UIClassFactory would simply store the reference to the instance of the real class factory – cf. The newInstance() would delegate the call to the cf.newInstance(). In addition, it would also register the listener to FlexEvent. DATA_CHANGE event, as shown below:

```
 public class UIFactory implements IFactory
 {
  . . . . . .
  public function UIClassFactory( cf:ClassFactory ) {
   wrappedClassFactory = cf;
  }

  public function newInstance():* {
   var obj:* = wrappedClassFactory.newInstance();
   obj.addEventListener(FlexEvent.DATA_CHANGE, onDataChange);
   return obj;
  }

  private function onDataChange(event:FlexEvent):void{
   . . . . . .
  }
 }
```

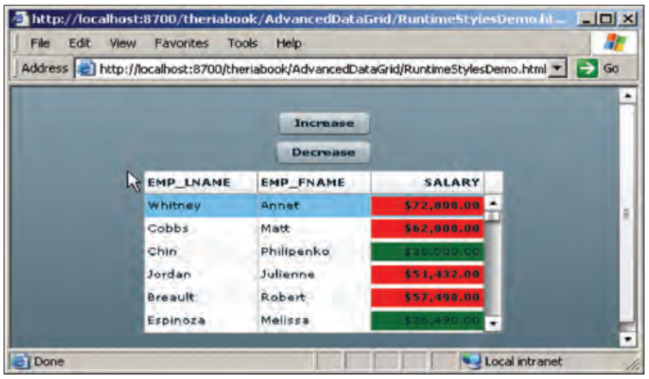
As a reminder, the properties of the factory-manufactured objects get assigned by iterating over properties of the specific property of the factory object. The name of this aggregating property is properties. That way all instances are initialized with the same values. Wrapping up the property properties is quite simple:

```
  public function set properties(v:Object):void       {
   wrappedClassFactory.properties = v;
  }
  public function get properties():* {
   return wrappedClassFactory.properties ;
  }
```

The complete code for UIClassFactory is available in Listing 7.

Let's talk about the onDataChange() handler. The implementations of both IDropInListItemRenderer and IDataRenderer events are sending us DATA_CHANGE events. We start by filtering out the events coming from the IDropInListItemRenderer part and leave only the ones coming from IDataRenderer. Then we single out the runtimeStyles property and treat it as a dynamic object carrying the property names that correspond to the style names. The values of these properties may be literal or, alternatively, the function references. In the latter case, we apply the call operator () prior to setting the style value with the setStyle().

One chore remains. As long as we want to communicate the runtimeStyles to any item renderer, including the ones that are individually set on a per column basis, we need to modify our DataGridColumn, overriding the implementation of the


http://localhost:8700/theriabook/AdvancedDataGrid/RuntimeStylesDemo.ht...
File Edit View Favorites Tools Help
Address http://localhost:8700/theriabook/AdvancedDataGrid/RuntimeStylesDemo.html Go
Increase
Decrease
Done
Local intranet

EMP_LNAME	EMP_FNAME	SALARY
Whitney	Annet	$72,000.00
Cobbs	Matt	$62,000.00
Chin	Philipenko	$36,500.00
Jordan	Julienne	$51,432.00
Breault	Robert	$57,490.00
Espinoza	Melissa	$36,490.00

Figure 5: A RuntimeStyleDemo application

DataGridColumn's itemRenderer setter:

```
 override public function set itemRenderer( val : IFactory ) : void {
  super.itemRenderer = new UIClassFactory(val as ClassFactory);
 }
```

The code for our DataGridColumn is shown in Listing 8, and Listing 9 has our test application RuntimeStyleDemo. Figure 5 depicts the RuntimeStylesDemo running.

What's the cost of our automation? We have replaced the ultralight UITextField with the heavier UIComponent – Label so there must be a potential for performance degradation. On the other end, the DataGrid recycles item renderers by maintaining a pool of them just enough to cover the visible portion of the column, which outright limits possible damage.

o far we have shown that it is possible to control runtime styles via anonymous or explicit functions (backgroundColor versus computedFontWeight in the above demo application). You can take our approach further and completely outsource the dynamic styling to a separate controller object flexibly instantiated via the getDefinitionByName() method. Come to think of it, you would completely shield developers from formatting and styling problems of a particular project.

Masked Input and Numeric Input

The input masking stops a user from entering non-appropriate characters. Stricter masks can prevent users from entering non-complete numbers, lesser than the required text, etc. – details are always implementation-specific. Taken to an extreme, masks can completely obliterate validation programming, albeit at a cost of fixing a specific user experience.

The first class of this section – MaskedInput – has been created by Peter Ent from Adobe. We will be using it for illustrative purposes only; the code walkthrough of MaskedInput is beyond the scope of this article. MaskedInput is a lightweight mask in which you can indicate the maximum number of positions in the mask and prescribe the type of characters the user can type. The movement of the insertion point is controlled by the control. For example, if you set a mask for entering a U.S. phone number as (###) ###-####, in response to the end user typing 6175551212, the control will display (617) 555-1212.

– CONTINUED ON PAGE 48

Meet Robert

A business executive at a popular social media site

Challenges

- Encountering poor video quality due to exponential growth in traffic.
- Increasing bandwidth costs due to growing audience size.
- Experiencing compatibility issues due to user-generated video arriving in multiple formats.

Solutions

- VitalStream Streaming Services – Improved quality of end-user video experience using a scalable and global content delivery network.
- VitalStream Advertising Services – Transformed the delivery of digital media from a cost center into a profit center.
- VitalStream Transcoding Services – Automatically converted all user-generated content into the leading streaming media format.

Providing End-to-End Solutions for Your Business

VitalStream is more than a rich media delivery company. We are your professional partner providing solutions that meet your unique business challenges. To learn more about VitalStream solutions, call 800-254-7554 or visit www.vitalstream.com/go/solutions/

VitalStream®

Digital Media Solutions for Your Business

Extending Component Metadata for Enhanced Documentation

Making your API easier to use

By Chip Temm

With the introduction of MX, ColdFusion programmers started to move away from the world of hard-to-maintain reams of scripts with includes, and into the world of clearly defined libraries with specific responsibilities. CustomTags were a good stab at re-usable code, but focused on the level of the function as opposed to grouping functions, which meant the method lacked a good way to define the return value and had scoping issues.

User defined functions (UDFs) gave us a clear return value, but still left us longing for classes. For several years now, we've had ColdFusion Components (CFCs) that give us a good chunk of the flexibility and power of object-oriented languages. Many have been honing their application programming interfaces (APIs) for years now, and the challenge has been to manage documentation and make the API usable to new developers joining the team, customers, or members of the CF community. Luckily, there is a language feature that can help us keep our documentation updated along with our code: component metadata. All of the attributes of the CFComponent, CFProperty, CFFunction, and CFArgument tags are available for us to view and use programmatically through reflection. We can ask components to tell us about themselves by passing an instance of them to CF's built-in getMetaData function.

```
<cfset example = createObject("component", "CFIDE.admi-
napi.administrator") />
<cfdump var="#getMetaData(example)#" />
```

In this article, we will explore ways of adding metadata to CFCs that can make your API easier to use.

One of the tenets of good API docs is that after reading them you should not be left with questions that can only be answered by reading the code. In some cases, the reader may not have access to the code. Even where the code is available, however, it's better left unread by API consumers. You want consumers of your API to code what is described (the interface) and not how it is done (implementation). When you encourage fellow developers to root around in your code, they may find ways to "optimize" their calls based on the internals of the functions. This can lead to reliance on "undocumented features," which were never really features at all. This practice violates the principle of information hiding in object-oriented programming. At the end of the day, the result can be brittleness – if you change your implementation, the client breaks.

If you use understandable names for your components, properties, functions, and arguments, and if you provide good type information (i.e., don't use the "any" type if possible), you are on your way to a good API. Names are important – think about them a lot. They provide the basic semantics of your API. Which has a clearer meaning: Person.getAgeYears() or Person.calcYears()? If you provide a description of every element (CFC, property, function, or argument) of your API using the hint attribute, you will have built a much more understandable system. Hints should explain more than the name can by itself. >From hints, consumers should get enough information to determine the right element for the use intended.

There are a number of types of information embedded in hints that you can see when scanning API docs. Sometimes usage information is there – we all like snippets or examples. We often see information on enumerated values for function arguments or returns. For example, ColdFusion's own administrative API's CFIDE.adminapi.base class has a function called getEdition, which the hint states can return the values "Developer, Evaluation, Standard, Enterprise, and Professional." On the flip side, CFIDE.adminapi.administrator has a function called setAdminProperty with an argument propertyName, which the hint states can only accept the values "migrationFlag, MXMigrationFlag, SetupWizardFlag, migrateCF5, migrateCF6, setupOdbc, and setupEnabldRds." There are other examples such as authorship (some classes may have a lot of functions contributed by many different authors), validity ("this function deprecated as of version 1.4"), and the ever-present TODO. All of these types of information can be broken out into their own metadata fields if you desire. One of the benefits of doing so is to assist in extracting different kinds of documentation from the code. Creating a new metadata field is as simple as adding it as a new attribute to your CFComponent, CFProperty, CFFunction, or CFargument tag. This worked in CF6, but is actually a documented feature of CF7! For the purpose of this article, we'll focus on the enumerated values example, as this is a metadata element that can contribute to different kinds of automation related to the documented code, such as testing. (Java5 programmers bear with us – CF has no enumerated types.)

Decomposing Metadata

If the hint for CFIDE.adminapi.base.setAdminProperty's propertyName argument tells us that it can only accept one of seven values, that's important information for client code to have, which it will use to adapt accordingly. What if a new version of CF was released that exposed more admin properties? Would our client code automatically cope? Imagine a CFCUnit test stub generator. If it could discover the range of allowed values programmatically, it could do more work for you – testing both in-range and out-of-range values.

Looking at the hint for setAdminProperty, we can decompose it into two kinds of information: a description about what the function does and an enumeration of potential return values. The existing hint attribute is meant to handle the description, but we could add our own new attribute inputRange and set its value to a list. Note: the example is hypothetical as this CFC is not editable.

```
<cffunction name="setAdminProperty" … >
<cfargument
name="propertyName"
hint="Return Migration or setup flag to be set."
return="string"
inputRange="migrationFlag, MXMigrationFlag, SetupWizardFlag,
migrateCF5, migrateCF6, setupSampleApps, setupOdbc,
setupEnabldRds"
>
```

Now, we can see the changed metadata for the argument if we cfdump the result of a getMetadata call, as shown in Figure 1.

The trick is to remember that there is no way to reference by name the metadata nested in the PARAMETERS array shown in Figure 1. You have to know the position of the argument in the array to access it. This is a pain. I have a utility called simplifyMetadata (see Listing 1) that converts all the nested arrays to structs, which allows easier access to nested information. After running the metadata through, what we end up with is shown in Figure 2.

Now we have a value that we can programmatically access, which will tell us all possible input values for this function:

```
<cfscript>
   adminAPI = createObject("component", "CFIDE.adminapi.administrator");
   metadata = getMetadata(adminAPI);
   simpleMetadata = simplifyMetadata(metadata);//custom function to
convert nested arrays
</cfscript>
<cfoutput>
```

The input range for the setAdminProperty function of CFIDE.adminapi.administrator is:

```
simpleMetadata.functions.setAdminProperty.parameters.propertyName.
inputRange#
</cfoutput>
```

Driving Validation with Metadata

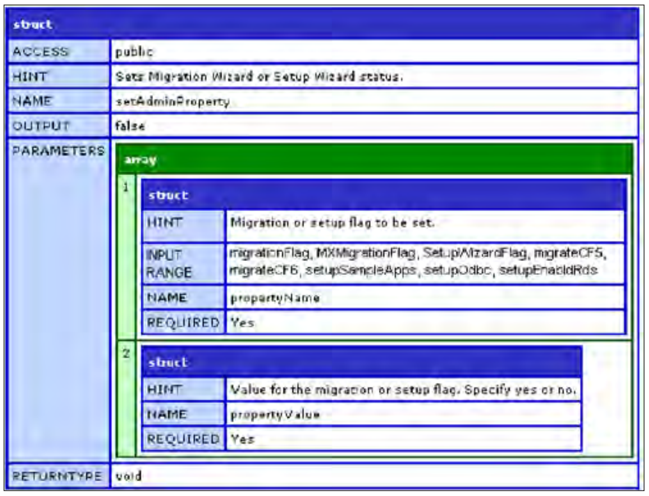
An interesting way to use this type of metadata about input ranges is to use it to help drive data validation for the argument. Let's say we have the function convertColorToHex(string colorName) in a CFC called UtilityClass. We want to inform the clients of this function that we only know how to convert Red, Green, and Blue. We can define the new metadata attribute called inputRange to expose this in a sensible way.

We can see in Listing 2 that while consumers of the function can see the input range without reading the code, the function itself does not validate the input range. If out-of-range input is sent, the function will err by saying "struct key does not exist."

To validate input, we can insert a check after we set up our color structure:

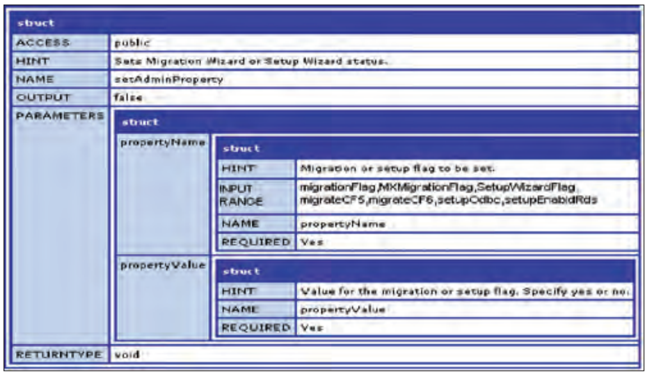
```
…
   if(not structKeyExists(_colorStruct, arguments.colorName)){
return "";
}
…
```

This, however, violates the expectation of the client accord-



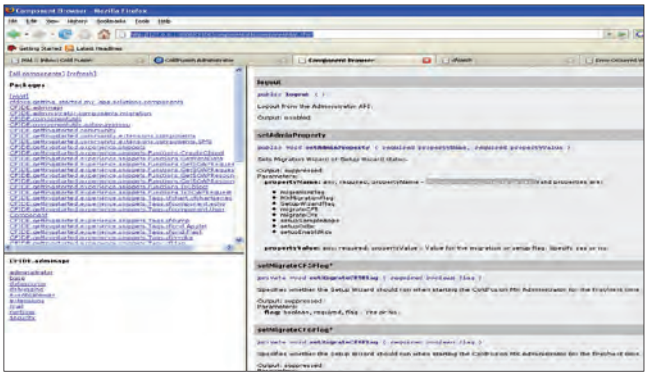
```
struct
ACCESS      public
HINT        Sets Migration Wizard or Setup Wizard status.
NAME        setAdminProperty
OUTPUT      false
PARAMETERS  array
            1  struct
               HINT      Migration or setup flag to be set.
               INPUT     migrationFlag, MXMigrationFlag, SetupWizardFlag, migrateCF5,
               RANGE     migrateCF6, setupSampleApps, setupOdbc, setupEnabldRds
               NAME      propertyName
               REQUIRED  Yes
            2  struct
               HINT      Value for the migration or setup flag. Specify yes or no.
               NAME      propertyValue
               REQUIRED  Yes
RETURNTYPE  void
```

Figure 1



```
struct
ACCESS      public
HINT        Sets Migration Wizard or Setup Wizard status.
NAME        setAdminProperty
OUTPUT      false
PARAMETERS  struct
            propertyName   struct
                           HINT         Migration or setup flag to be set.
                           INPUT RANGE  migrationFlag,MXMigrationFlag,SetupWizardFlag,
                                        migrateCF5,migrateCF6,setupOdbc,setupEnabldRds
                           NAME         propertyName
                           REQUIRED     Yes
            propertyValue  struct
                           HINT         Value for the migration or setup flag. Specify yes or no.
                           NAME         propertyValue
                           REQUIRED     Yes
RETURNTYPE  void
```

Figure 2



```
Component Browser - Mozilla Firefox

[all components] [refresh]
Packages
[illegible]

CFIDE.adminapi
[illegible]

logout
public logout ( )
Logout from the Administrator API.
Output: enabled

setAdminProperty
public void setAdminProperty ( required propertyName, required propertyValue )
Sets Migration Wizard or Setup Wizard status.
Output: suppressed
Parameters:
  propertyName: any, required, propertyName - [illegible] and properties are:
  [illegible]
  propertyValue: any, required, propertyValue - Value for the migration or setup flag. Specify yes or no.

[illegible]
```

Figure 3

ing to the hint: emptystring is not a hexadecimal value. We should probably throw an exception instead (another thing we can document using metadata later). But we have another problem, too. When we make changes to the code, we may have to update the inputRange to let consumers know we are supporting more colors. What if changing our inputRange was all we had to do in order to make our validation correct? Let's revise the method definition by inserting the following between line 14 & 15:

```
<cfset var _metadata = simplifyMetadata (getMetaData("#this#")) />
<cfset var _range =
metadata.functions.convertColorToHex.parameters.colorName.inputRange />
<cfif not listfindnocase(arguments.colorName, _range)>
    <cfthrow type="valueOutOfRangeException" >
</cfif>
```

Now when we change the inputRange attribute on the colorName argument, the validation rule changes its behavior too. Metadata driving function – pretty cool. The performance-minded among you are asking what this coolness costs? Hardly anything, in fact. We already have an instance of the component loaded into memory (there are no static classes in CF so the only way to call our function is from an instance), and the getMetaData call takes so little time that it will usually come out as 0ms in your trace. If you build a function like simplifyMetadata, running that will cost you, as it will recurse through the metadata to simplify it. For speed, you could just rely on the getMetaData call alone.

Now we have the potential for our function to throw a valueOutOfRangeException if someone passes an unacceptable value. However, our client doesn't know that without reading the code. We can include another metadata attribute – throws:

```
<cffunction
  name="convertColorToHex"
  return="string"
  access="public"
  throws="valueOutOfRangeException"
  hint=  "Accepts an English color name and returns
         the corresponding six character hexadecimal."
>
```

Now our client has documentation for the possible exception and can determine how to deal with the event that it is thrown. Listing 3 brings all of the topics discussed together.

Conclusion

This article assumes that you standardize on a set of clearly defined and meaningful extra metadata attributes that are used consistently across your project(s) by all members of your team. Without consistency, your API consumers won't be able to rely on the metadata. "Was that attribute named inputRange or allowedValues?" Code templates/wizards are a good approach to such standardization. Once you have standardized, you may want to alter the component browser that ships with CF to expose your new standard attributes there, as it makes extensive use of getMetadata to extract useful information from CFCs. This is not hard, as Adobe has been kind enough to leave the code unencrypted.

If you haven't checked it out, open up a browser and navigate to cf_root/wwwroot/CFIDE/componentutils/componentdoc.cfm. As shown in Figure 3, you will see three panes: the upper left pane is a list of all CFC packages (directories containing CFCs); the lower left pane is a list of the components living inside of the package selected in the upper left pane; and the right pane is a description of the CFC itself.

I hope that you have found this discussion of component metadata interesting. I encourage you to experiment with this. There is a lot that you can do programmatically in this area. I have, for example, enabled rudimentary Interfaces by defining an implements attribute on CFCs. Overall, the biggest benefit to the practice of extending metadata will accrue to larger teams and those interested in generating metrics, documentation, or other code (such as test stubs) on the basis of the metadata.

About the Author

Over the past decade, Chip Temm moved from North America to Europe and on to Africa where his company anthroLogik solutions provided analysis and development services to non-governmental organizations across seven time-zones. He is currently in Washington, D.C. , where "remote development" means working from home and "wildlife" means raccoon.

chip@anthrologik.net

Listing 1

```
<cfscript>
  /******************
  Simple function to recurse cfc metadata and
  return the metadata with all nested arrays
  converted to structs.
  ******************/
  function metadataSimplify(in_metadata){//returns struct
      var _return_metadata = duplicate(arguments.in_metadata);
      var _key='';
      for(_key in _return_metadata){
          if(isArray(_return_metadata[_key])){
              _return_metadata[_key]=
convertArrayOfStructs(_return_metadata[_key]);
          }else if(isStruct(_return_metadata[_key])){//drill down
              _return_metadata[_key] =
metadataSimplify(_return_metadata[_key]);
          }
      }
      return _return_metadata;
  }

  function convertArrayOfStructs(inputArray){//returns struct
      var _length = arrayLen(arguments.inputArray);
      var _i=1;
      var _outputStruct = structNew();
      var _elementStruct ='';
      if(_length){
          for(_i=1;_i lte length;_i=_i+1){
              _elementStruct = arguments.inputArray[i];
              //set outputStruct key to the Name key of the nested
struct
              //ensure struct cleaned of nested arrays before return-
ing
              _outputStruct[_elementStruct.name] =
metadataSimplify(_elementStruct);
          }
      }
      return _outputStruct;

  }
</cfscript>
```

Listing 2

```
<cffunction
name="convertColorToHex"
return="string"
access="public"
hint=  "Accepts an English color name and returns
    the corresponding six character hexadecimal."
>
<cfargument
    name="colorName"
    type="string"
    inputrange="Red,Green,Blue"
    hint="The color to convert"
/>
<cfset var _colorStruct = structNew() />
<cfscript>
    _ colorStruct.Red = "FF0000";
    _ colorStruct.Green = "00FF00";
    _ colorStruct.Blue = "0000FF";
</cfscript>
<cfreturn _colorStruct[arguments.colorName] />
</cffunction
```

Listing 3

```
<cftry>
<cfset myUtilityClassInstance = createObject("component","UtilityCla
ss") />
<cfset metadata = simplifyMetadata(getMetadata('#myUtilityClassInsta
nce#'));
<cfset hex = myUtilityClassInstance.convertColorToHex(valueSentFromT
extInput) />
<cfcatch type="valueOutOfRangeException">
<cfoutput>
  You have entered a color I don't know. Try one of
  #metaData.functions.convertColorToHex.parameters.colorName.
inputRange#
</cfoutput>
</cfcatch>
<cfcatch>
  Something bad happened.
</cfcatch>
</cftry>
```

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

CFDJ Advertiser Index

ADVERTISER	URL	PHONE	PAGE
Adobe	www.adobe.com/products/coldfusion/flex2		2
AJAXWorld East	www.ajaxseminar.com	201-802-3020	41
AJAXWorld Bootcamp	www.AJAXbootcamp.com	201-802-3000	49
CFDynamics	www.cfdynamics.com	866-233-9626	3
CFDJ	http://coldfusion.sys-con.com/	800-303-5282	39
CommunityMX	www.communitymx.com/trial		6
Cynergy Systems	www.cynergysystems.com		26-27
EdgeWebHosting	edgewebhosting.net	1-866-334-3932	4
Hal Helms	halhelms.com		45
HostMySite	www.hostmysite.com	877-215-4678	11, 52
Hotbanana	hotbana.com/cfdj	866-296-1803	45
Intergral	www.fusiondebug.com		23
Paperthin	www.paperthin.com	1-800-940-3087	37
TeraTech	www.teratech.com		51
VitalStream	www.vitalstream.com/tools/mxdj.asp	800-254-7554	14-15

Before and After with an AJAX Rich Internet Application GUI

A Webmaster's view

By Nate Weisiger

TapeTrader.com is a ColdFusion-powered, live music sharing community with over 50,000 members and a vast database cataloging hundreds of thousands of hours of live music recordings. The TapeTrader.com community connects fans together so that they can catalog, share and trade their recordings.

In addition to my day job at StubHub and my personal project running (ColdFusion powered) GuitarsAndMore.com, I contribute some engineering time to the TapeTrader.com community. You get the idea… I'm a live music fan!…and kind of a nerd.

When TapeTrader.com wanted to update its live recording catalog user interface, we turned to AJAX and TIBCO's General Interface software to provide a feature rich GUI that looks and feels more like a desktop application than a standard Web application. Using the General Interface AJAX libraries and visual AJAX tools, which TIBCO Software recently provided for free, made the job quick, painless and fun.

This article discusses how we transformed our existing CFML/HTML pages into XML services optimized to work with the TIBCO General Interface (GI) AJAX libraries.

Before and After

The old HTML pages generated with CFML were good, but not great. Some members have huge show lists and paging through the results could be tedious, not to mention the extra load put on the database and Web servers (and don't get me started about maintenance).

It was decided, instead, to create a system that could transfer the entire result set to the browser using XML, which made the process of exploring a catalog of recordings a client-side process. By running the Catalog Explorer application on the client, we can deliver a much richer and friendlier user experience. As shown in Figures 1 and 2, this application provides the ability to explore a member's list, add selections to your "favorites," and see a recording's details, all within a single screen. The legacy HTML application required many separate pages/pop-ups and cumbersome server calls to facilitate that same interaction.

Architectural Choices

Running the JavaScript application in the browser prevents the need for page refreshes for each interaction. It also allows us to send query result sets from the database to the browser in a single request as opposed to the constant querying and subset pagination that is required with the standard HTML page interface (see Figure 3). This not only reduces the number of hits to the server, but also improves the user experience by eliminating the latency of HTTP requests and server-side processing time.

The General Interface libraries map well with these concepts. They provide the rich user interface controls we need, as well as an "XML data cache" object and easy-to-use data-binding mechanisms to bind our XML to our GUI (see Figure 4). It was relatively easy to modify our current CFML routines that were previously used to process, generate, and send XML files to be consumed by the client-side GI user interface. Because these new ColdFusion scripts are pure data services, the XML feeds can be used in other applications or given to users to implement as they wish.

Creating Our AJAX Rich Internet Application

GI's visual development environment (GI Builder) makes it fast and easy to create a feature-rich user interface. GI Builder provides many out-of-the-box GUI components that you simply drag and drop together to create any user interface you like. Styles are set using the component properties or by using CSS. GI Builder's event editor makes it simple to bind JavaScript functions to user interface components, and the JavaScript editor provides menu-based references to the more than 100 Java Script class objects that make up the underlying GI Framework. Asynchronous communication classes, data-caching objects and many other goodies are packed right into the GI Builder environment. GI Builder provides wizards for connecting to XML, SOAP, JSON and CSV data services, as well as nifty debugging and logging capabilities which are very useful when creating JavaScript applications. If necessary, GI Builder can connect to JSON and other HTTP service types via GI APIs. You can find a lot of great documentation, tutorials and chat boards at the TIBCO Website (http://www.tibco.com/devnet/index.html).

The source code for the GI project demonstrated in this article can be downloaded from the online version of this article at http://coldfusion.sys-con.com. You can view the source files with any editor, but it is best to use GI's visual editor as GI Builder is optimized for GUI file types. You can download GI Builder for free at http://www.tibco.com/devnet/index.html.

Let's start by creating our ColdFusion data services, and then we'll create the UI and bind it to the GI AJAX application.

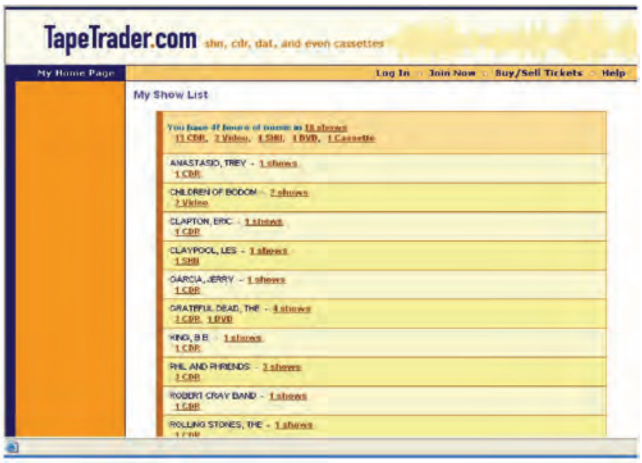
Server-Side Code

Listing 1 shows the original ColdFusion code which generated the HTML for the Catalog Explorer. A couple of queries are executed against the database, the results are looped through with some conditional logic, and HTML is generated and sent to the browser. Pretty standard ColdFusion stuff.

Recall that in GI applications, HTML is not needed. The UI is generated by the client-side GI framework in the browser. To create our new application, we can strip the code in Listing 1 down to its very basic elements that provide the needed data as a simple data service. To do this, we need to define how the XML response from the server should be structured such that the client-side GI app can consume it.

GI's Common Data Format (CDF)

While GI can work with any XML, SOAP, JSON or CSV source, its GUI components are set up to bind to what it calls



TapeTrader.com shn, cdr, dat, and even cassettes

My Home Page | Log In | Join Now | Buy/Sell Tickets | Help

My Show List

You have 47 hours of music in 18 shows
11 CDR, 2 Video, 1 SHN, 1 DVD, 1 Cassette

ANASTASIO, TREY - 1 shows
1 CDR

CHILDREN OF BODOM - 2 shows
2 Video

CLAPTON, ERIC - 1 shows
1 CDR

CLAYPOOL, LES - 1 shows
1 SHN

GARCIA, JERRY - 1 shows
1 CDR

GRATEFUL DEAD, THE - 4 shows
3 CDR, 1 DVD

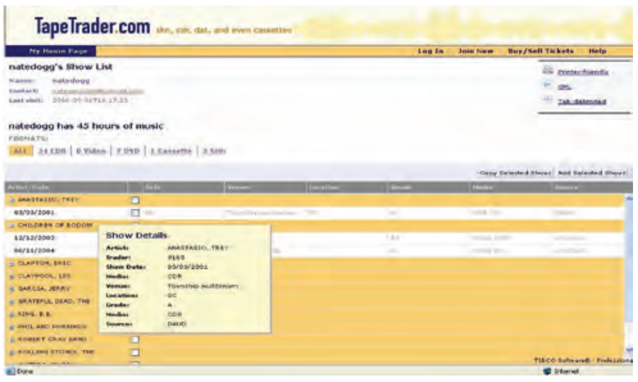
KING, B B - 1 shows
1 CDR

PHIL AND PHRIENDS - 1 shows
1 CDR

ROBERT CRAY BAND - 1 shows
1 CDR

ROLLING STONES, THE - 1 shows

Figure 1 Before: Static HTML Pages



TapeTrader.com

My Home Page | Log In | Join Now | Buy/Sell Tickets | Help

natedogg's Show List
Name: natedogg

natedogg has 45 hours of music
FORMATS:
ALL | 14 CDR | 0 Video | 2 DVD | 1 Cassette | 3 SHN

Show Details
Artist: ANASTASIO, TREY
Show Date: 03/03/2001
Media: CDR
Venue: Township Auditorium
Grade: A
Media: CDR
Source: DAUD

Figure 2 After: AJAX Rich Internet Application

Common Data Format (CDF). CDF is a simple XML schema for describing data where each unit of data is bounded by a <record> node. <record> nodes can contain other <record> nodes to describe hierarchical relationships. The <record> node is used as a pre-set delimiter to which GI's GUI objects can automatically bind. The <record> node thing is handy, but one has to wonder a bit why any XML wouldn't work if you could just specify <artist> as a delimiter and iterate on <artist>… but anyway, that's how GI works.

Luckily, if you do happen to have a data source that comes from a service that's not in the CDF format, GI Builder provides both visual tools and APIs that let you transform these XML structures into and from GI's CDF format. This allows you to preserve the semantics of the original XML while allowing GI's CDF to simplify client-side development. Tip: For large data sets, be sure to use the .compile() method for XML to XML transformations. This will use a faster XSLT processor that is much more efficient than JavaScript.

For simplicity, our application uses ColdFusion to generate the data feed in the CDF format. Listing 2 shows what we need our CFML to generate for client-side consumption. (Listings 2-4 can be downloaded from the online version of this article at http://coldfusion.sys-con.com)

Listing 3 shows the CFML service used to generate the CDF

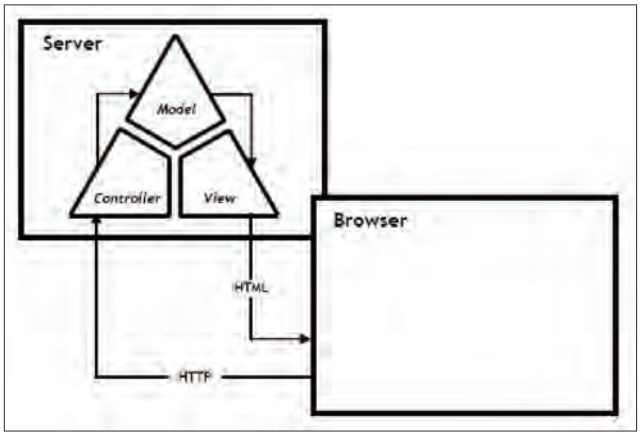


Figure 3 HTML page view architecture

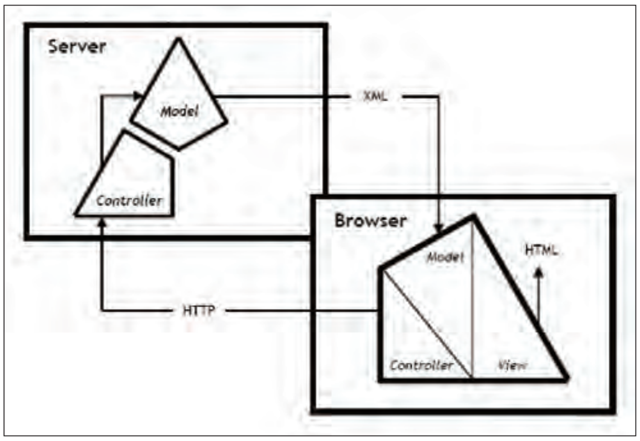


Figure 4 AJAX RIA and XML Service architecture

Property	Value	Explanation
XML ID	source_xml	This is the unique ID of the XML data as persisted in the cache by the code in listing 4. This setting creates the relationship between the GI control and the data object in cache.

Table 1 For the Matrix control named mtxShowList

Property	Value	Explanation
Path	venue	This is the xml path to the data for the "venue", in this case the attribute named "venue" on the CDF record. Repeat similar xml path bindings for the other columns in the data grid.

Table 2 For the Matrix.Column control named Venue

data feed. Note that we reused the query from the original source code, but added the extra detail needed for this data feed. We are getting nearly all the data we need in a single query operation, then generating a hierarchical GI's CDF data feed as output instead of generating HTML. This method of development allows the ColdFusion code to be much simpler and cleaner, and since we're not sending HTML requests back and forth, we significantly decrease bandwidth, server and database utilization. Once the data is sent to the client, the trader has all the data necessary to browse the entire collection of another trader.

Note the inclusion of the prepareGiOutput() function, which adds an XML header tag, and the root <data> node required by GI's CDF schema, and removes extra, unnecessary white space (speed is everything, after all).

Connecting the Client Application to the Data Service

Listing 4 shows the JavaScript code that is part of the client application. This code uses GI's jsx3.net.Request class to asynchronously call the ColdFusion XML service we created earlier as a GET operation. Any input the service needs (in this case, trader_id) should be passed as URL variables. Upon receiving the response XML from the service, the data is put into cache and the data grid component updates its view (a pattern consistent with the standard Model-View-Controller architectures).

GI's data grid class is called a Matrix. The Matrix control can render as a grid, a list, or a tree with a variety of cell edit mask types. jsx3.gui.Matrix also implements the jsx3.xml.CDF interfaces and can contain columns of the type jsx3.gui.Matrix.Column. Like all GI GUI components, the Matrix component in GI Builder is configured through the properties editor or through JavaScript APIs. Tables 1 and 2 show the core properties needed to establish the binding between the Matrix data grid component and the CDF data in the cache.

Other Matrix and Matrix Column properties enable you to configure the sub-categorization of data, tune the pagination and expand/collapse display of data, and stylize the look and feel of the control (see Figure 5).

A Note about Security

Security seems to be one of the development commu-

Adobe Solution Partner 2006

INTERGRAL information solutions

<cf_essentials>

Indispensable CFMX tools.

FusionReactor

CFMX / J2EE Server, database and application monitoring

from **$249**

FusionDebug

Interactive Debugging for ColdFusion MX

from **$99***

fusion **www.fusion-reactor.com**

* Using GOT2DEBUG discount coupon. Offer ends October 31st.

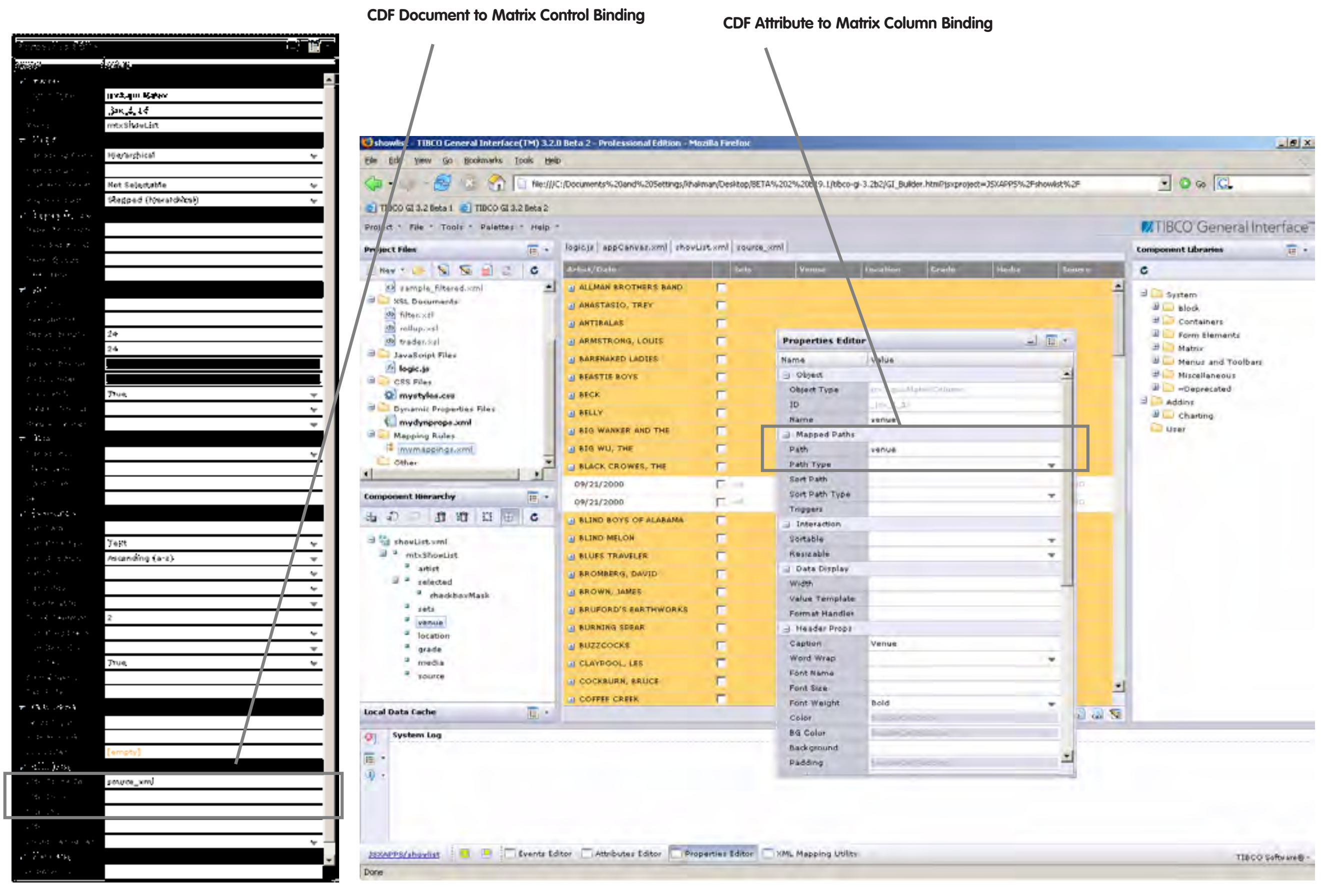


Figure 5 Matrix and Matrix Column controls being configured in TIBCO GI Builder

nity's concerns these days regarding AJAX. One of the nice things about the approach outlined earlier is that, from a security point of view, it all works within the existing Browser/Server security infrastructure with which, from the last decade of Web application development, we're all familiar. We could authenticate and control access to the .cfm processes that return XML the same way we would for any other HTML page. Further, by using XML as opposed to JSON, the client-side application is less subject to code-insertion risks since the data stream never needs to be evaluated on the client. Since XML is also piped through the XHR object in the browser by GI, it benefits from the XHR's restriction to talk to data services only from the same server as the containing page.

This restricts cross-site scripting and (unless you want cross-site scripting) is a good thing. For solutions like mash-ups, when using the XHR or GI, usually one will set up a gateway to the remote data domain in the form of a server proxy. This server proxy also gives an added place to implement security policies, and to log and monitor traffic between your primary security context and other security contexts. Most security concerns popping up today are from people making the same kinds of mistakes that we made seven or eight years ago in relation to CGI processes and the like. But this topic could be the subject of another paper, so let's just stop here for now.

Conclusion

AJAX not only enables a richer end-user experience, it also promotes the creation of data services that can be reused by multiple applications. This article explored how the AJAX Rich Internet Application toolkit TIBCO General Interface can be used to rapidly create rich interactive GUIs that could be used in conjunction with CFML pages that output XML rather than HTML.

Resources

- TIBCO Developer network site provides TIBCO General Interface AJAX Rich Internet Application Toolkit downloads, video tutorials, documentation, sample applications and discussion forums: http://developer.tibco.com
- Source code for the GI application used in this article can be found here: at the onlie version at this article at http://coldfusion.sys-con.com.

About the Author

Nate Weisiger is a Production Support Engineer and Web Developer for StubHub. Inc and is the owner/operator of GuitarsAndMore.com (Guitars and More, LLC). In the rare case that he gets bored, he develops and supports TapeTrader.com

nweisiger@stubhub.com

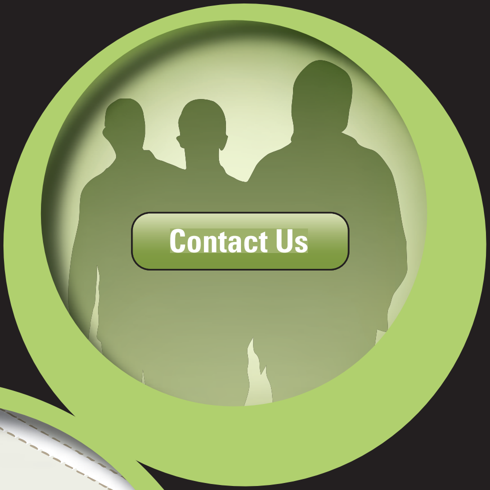
Listing 1

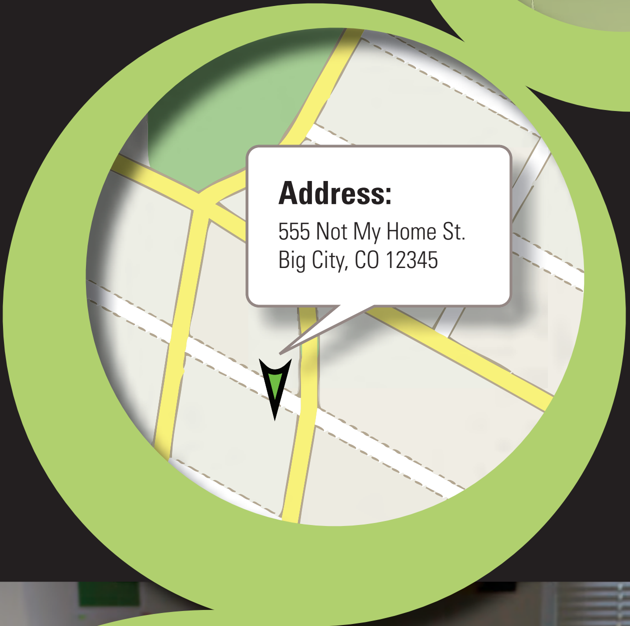
```
<!--- Retrieve the new show id --->
<cfquery datasource="#request.default_dsn#" username="ttnew"
password="secretphrase" name="SearchInfo">
   SELECT count(show_id) as mainshowCount, artist_name, artist.artist_id
   FROM show, artist
   WHERE show.artist_id = artist.artist_id
   AND user_id = #thisUserId#
   GROUP BY artist_name, artist.artist_id
   ORDER BY artist_name
</cfquery>

<!--- Retrieve the media counts--->
<cfquery datasource="#request.default_dsn#" username="ttnew"
password="secretphrase" name="MediaInfo">
   SELECT count(show_id) as mediashowCount, media_name, media.media_id

   FROM show, media
   WHERE show.media_id = media.media_id
   AND user_id = #thisUserId#
   GROUP BY media_name, media.media_id
   ORDER BY mediashowCount DESC
</cfquery>

<cfoutput>

<cfif thisUserId EQ session.user.user_id>
   <span class="headline">My Show List</span><br><br>
<cfelse>
   <span class="headline">
   <a href="##" onClick="popupWindow('#request.root_dir#/members/
popup_trader_details.cfm?user_id=#thisUserId#','traderdetails','menu
bar=no,scrollbars=yes,width=475,height=450,top=70');"><span class="h
eadline">#checkForUser.username#</span></a><span class="headline">'s
List</span><br><br>
</cfif>

<table border="0" cellspacing="0" cellpadding="0" width="553"
align="center">
   <tr>
      <td bgcolor="##CC6600" rowspan="700"><img src="#request.clear#"
width="8" height="1" border="0"></td>
      <td bgcolor="##CC6600"><img src="#request.clear#" width="526"
height="1" border="0"></td>
      <td bgcolor="##CC6600" rowspan="700"><img src="#request.clear#"
width="1" height="1" border="0"></td>
   </tr>
   <tr>
      <td bgcolor="##FFE198">
         <br><span class="forminstruct">
         &nbsp;&nbsp;<cfif thisUserId EQ session.user.user_id>You
have<cfelse>#checkForUser.username# has</cfif><b>#NumberFormat(numb
erOfHours,"999,999,999")#</b> hours of music in <cfif getShowInfo.
recordCount><a href="#request.root_dir#/members/show_list_detail.cfm?u
id=#thisUserId#">#NumberFormat(getShowInfo.recordcount,"999,999,999")#
shows</a><cfelse>0 Hours</cfif></span><br>&nbsp;&nbsp;&nbsp;&nbsp;
         <cfloop query="MediaInfo">
            <a href="#request.root_dir#/members/show_list_detail.cfm
?uid=#thisUserId#&mid=#mediaInfo.media_id#">#mediaInfo.mediashowCount#
#mediaInfo.media_name#</a><cfif mediaInfo.currentrow NEQ mediaInfo.
recordcount>,&nbsp;&nbsp;</cfif>
         </cfloop><br>&nbsp;
      </td>
   </tr>
   <tr>
      <td bgcolor="##CC6600"><img src="#request.clear#" height="1"
width="1" border="0"></td>
   </tr>
   <cfif searchInfo.recordCount>
      <cfloop query="SearchInfo">
         <!--- Get Show Count For Each Artist --->
         <cfquery datasource="#request.default_dsn#" username="ttnew"
password="secretphrase" name="showCount">
            SELECT media_name, media.media_id, count(show_id) as
showCount
            FROM show, media
            WHERE show.media_id = media.media_id
            AND user_id = #thisUserId#
            AND artist_id = #searchInfo.artist_id#
            GROUP BY media.media_name, media.media_id
         </cfquery>
         <cfset RowColor = "FDF9CE">
         <cfif searchInfo.currentrow MOD 2 EQ 0>
            <cfset RowColor = "FBF49C">
         </cfif>
         <cfset thisArtist = searchInfo.artist_id>
         <tr>
            <td bgcolor="#RowColor#" nowrap><img src="#request.clear#"
height="5" width="1" border="0"><br>&nbsp;&nbsp;&nbsp;#searchInfo.art-
ist_name#&nbsp;&nbsp;-&nbsp;&nbsp;<a href="#request.root_dir#/members/
show_list_detail.cfm?uid=#thisUserId#&aid=#thisArtist#">#searchInfo.
mainshowCount# shows</a><br>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<cfloop
query="showCount"><a href="#request.root_dir#/members/show_list_de-
tail.cfm?uid=#thisUserId#&aid=#thisArtist#&mid=#showCount.media_
id#">#showCount# #media_name#</a><cfif showCount.currentrow NEQ show-
Count.recordcount>,&nbsp;&nbsp;</cfif></cfloop><br><img src="#request.
clear#" height="5" width="1" border="0"></td>
         </tr>
         <tr>
            <td bgcolor="##CC6600"><img src="#request.clear#"
height="1" width="1" border="0"></td>
         </tr>
      </cfloop>
   <cfelse>
      <tr>
         <td bgcolor="##FBF49C" align="center">
            <br>
            <cfif thisUserId EQ session.user.user_id>
               <span class="errorred">You have no shows in your
list.</span><br><br>
               <a href="#request.root_dir#/members/show_addedit.
cfm">Add A Show</a><br><br><br><br><br>
            <cfelse>
               <span class="errorred">#checkForUser.username# has no
shows listed.</span><br><br><br><br><br><br><br>
            </cfif>
         </td>
      </tr>
      <tr>
         <td bgcolor="##CC6600"><img src="#request.clear#" width="1"
height="1" border="0"></td>
      </tr>
   </cfif>
</table>
</cfoutput>
```

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

Scary Question.

Exactly who is developing your next app?





Your App Starts Here.

We are the leaders in RIA development services.

INCREDIBLE APPLICATIONS

PASSIONATE USERS

PROVEN SUCCESS

CYNERGY

Unlock your potential
with the help of industry leaders in
Rich Internet Application development.
10 Years. 1400+ Customers.
Your app starts here.

CYNERGYSYSTEMS.COM

Adobe | Solution PARTNER

Content Management with ColdFusion

How ColdFusion can be used to create a CMS covering every phase of the CMS lifecycle

By Harry Klein

As the Web develops, the need for easy and efficient Web site and portal management increases.

Developing an easy-to-use future-proof content management system (CMS) from scratch isn't easy. The basic functionalities can be assembled in a reasonable timeframe, meaning that you will be able to edit the structure and the content of the Web site. An enterprise CMS with complex features like scheduled content publishing, content replication, cluster support, sophisticated workflows, roles, and other advanced features can take several years.

In this article I'll explain the purpose of a CMS and its benefits. Then I'll describe the content lifecycle and how ColdFusion can be used in every phase. In the last section I'll provide links to helpful (mostly Open Source) ColdFusion projects that can be integrated into a CMS.

What a Content Management System Is and Why You Should Use One

A content management system delivers functionalities to empower non-technical staff to create new pages on a site and produce and modify existing content.

After the set-up and configuration process, which in many cases still has to be done by technical staff, a product manager can update the information on a new product release without any programming skills, HR managers can publish new job offers, and the PR manager can put press releases on the Web site. Web site templates and structured content classes assure that the company's corporate Web site design will always be maintained.

A flexible CMS allows the comprehensive control and man-

agement of digital content by integrating different content types (e.g., text, images, Flash, video, sound, etc.).

One of the most important features of an up-to-date CMS is a comprehensive security system including rights, roles, and groups (see Figure 1). So each Web site team member can participate in the publishing process to the degree desired, permissions such as to create, edit, change, delete, or publish content have to be easily assigned. If there are more than 10 editors you'll find it helpful to assign bundles of rights to roles and assign these roles to the editors. Typical roles are "author" "editor," "translator," and "editor-in-chief." If several sites must be maintained like an intranet, a corporate Web site, and different country Web sites, groups will help to assign the editors to the sites they're entitled to work on. For maximum flexibility it should be possible to assign arbitrary combinations of groups and roles to an editor: a person can be an author for the international Web site and an editor with more permissions for the country Web site.

Workflows help to define processes, e.g., new or changed content has to be reviewed before publishing. The CMS should inform the editor-in-chief via e-mail or an integrated notification system when an article has been finished and has to be approved. Powerful workflow systems can publish an approved article automatically or the article will revert back to the author with comments about changes.

Content management systems help companies efficiently create and maintain intranets, portals, and other Web sites. They structure and speed up the publishing process and guarantee compliance with corporate design. Last but not least, a CMS reduces Web site publishing costs in a significant way, considering that employees can easily provide and publish content and avoid the expense of Web agencies making simple content changes.

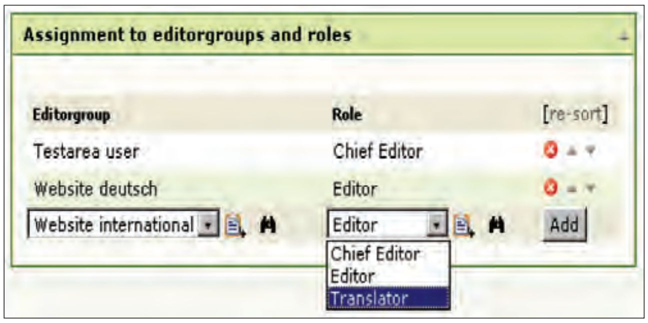
Phases of the Content Management Lifecycle

Most CMS experts describe the four phases of content management (see Figure 2) as:

1. Content creation, editing, integration of external content
2. Quality assurance and approval
3. Publishing and content delivery
4. Versioning and archiving

The first phase includes content creation, editing existing content, and integrating external content through syndication (e.g., sports or stock exchange news). It also means aggregating content, categorizing it (for target groups, personalization, etc.), and defining the metadata for optimized content retrieval. To achieve a high acceptance by editors, content creation has to be simple and accurate: functions like WYSIWYG, preview, libraries, form-based-entry, and direct edit help integrate the content fast and comfortably.

To avoid typos and unwanted information on the Web site, workflows can help create the approval process (the second phase). Sophisticated workflow systems can set up a simple four-eye principle that can be started manually, as well as predefined actions and complex processes that start automatically or periodically. In a workflow several tasks can be approved and forwarded to the next step or rejected to the author. Content repositories or libraries store the content managed and should deliver functions for editors to enable easy retrieval and reuse



Assignment to editorgroups and roles

Editorgroup	Role	[re-sort]
Testarea user	Chief Editor	
Website deutsch	Editor	
Website international	Editor	Add

Figure 1 Assign users to editor groups and roles

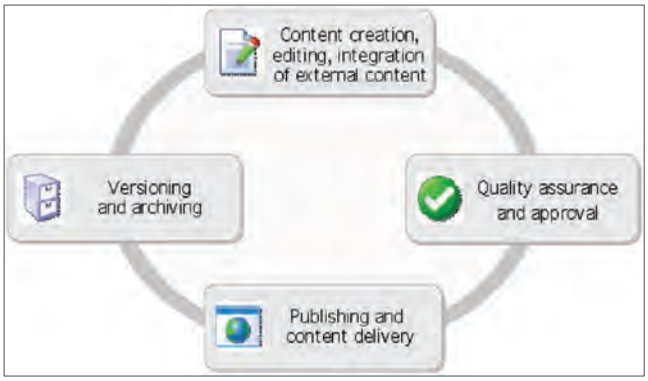
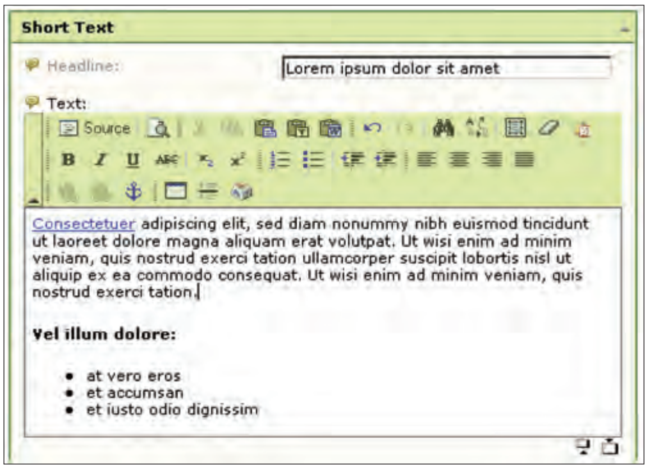

Content creation, editing, integration of external content
Quality assurance and approval
Publishing and content delivery
Versioning and archiving

Figure 2 Phases of the content management lifecycle



Consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Ut wisi enim ad minim veniam, quis nostrud exerci tation.

Vel illum dolore:

- at vero eros
- et accumsan
- et iusto odio dignissim

Figure 3 FCKeditor

of captured content. The API defined in the Java specification JSR170 helps access content in a vendor- or implementation-neutral fashion. It contains several content services such as the versioning and query service.

Publishing and content delivery constitute the third phase of the content lifecycle. Publishing is the preparation of content for different target groups and channels (e.g., HTML, print, PDA output), generating files, and distributing them to different servers. Content delivery means serving static, dynamic, or hybrid pages to Web site visitors. Static pages provide highest performance but only dynamic pages allow personalized content. The choice of publishing static or dynamic pages should depend on the content, not the CMS technology.

Versioning and archiving is the fourth and last step in the content lifecycle. Changes to published content have to be saved and should be restorable with minimal effort. If content has been deleted by mistake or typos have been overlooked, rollback functions should reactivate former versions of the content in a few clicks.

Over seven major revisions ColdFusion has empowered many developers to rapidly build powerful Internet and Web applications. High-performance and scalable CMS software can and has already been built using it. In the next sections I'll show you how ColdFusion can be used to create a CMS covering every phase of the CMS lifecycle.

Using ColdFusion in the Content Creation Phase

Acquisition

ColdFusion has built-in tags for HTTP support like CFHTTP, CFHeader, and CFContent.

CFHTTP can be used to retrieve data from another page or site. You can also have ColdFusion automatically save the files to the server when it gets the information back from the URL.

The conversion and import of external data (like Microsoft Office data or XML data) is possible through COM objects. For all kinds of COM questions cfComet is a good resource.

Dave Raggett's HTML TIDY is a free utility to fix HTML mistakes automatically and tidy up sloppy editing into nicely laid-out markup. Tidy can also be configured for HTML code saved from Microsoft Word.

Greg Steward wrote the ColdFusion function makexHTML-Valid, which takes a string and URL as arguments and uses jTidy to parse and validate the XHTML.

Aggregation

Incoming syndication feeds (like RSS or Atom feeds) can be retrieved using CFHTTP. The ColdFusion XMLParse function can be used to parse the document. This is also a validating parser, so it will choke if you pass invalid XML.

The CFXML tag is used to create an XML document that will later be converted to a string and written to a file. This lets you create an RSS feed for your site.

ColdFusion has built-in support for Web Services and provides a function (CreateObject) and a tag (CFINVOKE) for Web Service consumption. It's easy to integrate external data using Web Services even if they're developed in other languages.

Authoring

Some CMS Web-based text editors are very feature-rich and almost content management tools in themselves. Others are essentially plug-ins that act as form fields. Some are free and some are extremely expensive. The FCKeditor (see Figure 3) is one of the most popular and with good reason: it's free Open Source (LGPL, Closed Distribution Licence), easy to customize, and it works very well. On the server side, FCKeditor offers a complete integration pack for ColdFusion.

XStandard is another standards-compliant WYSIWYG plug-in editor for desktop applications and browser-based content management systems. The editor generates clean XHTML Strict or 1.1, uses CSS for formatting, and ensures the clean separation of content from presentation. Ben Nadel programmed a ColdFusion version of the XStandard Web Service.

CFFM is a cfml-based media asset repository manager intended for integration into existing Web site management solutions. It's for uploading, creating, renaming, deleting, and editing of files and directories.

The ASF File Explorer is a file manager to browse, add, and

"Over seven major revisions, ColdFusion has empowered many developers to rapidly build powerful Internet and Web applications"

remove directories and files from a specific server location. It was built using CFForms and Flash remoting.

Using ColdFusion in the Quality Assurance and Approval Phase

Workflow

An appealing Workflow GUI could easily be built using Flash or Flex. Workflow messaging, e-mail, or SMS notifications can be realized with CFMAIL and the SMS Event Gateway.

Spellchecker

Content quality can be ensured by including a spellchecker module. Some WYSIWYG editors like FCKeditor come with integrated spellcheckers like ieSpell and Speller Pages.

An interesting article on the *ColdFusion Developer's Journa*l Web site, explaining how to use the Open Sourced Jazzy Spell Checker with ColdFusion MX can be found at http://cfdj.sys-con.com/read/42120.htm.

Localization

rbMan is a cfmx app used for managing Java resource bundles. It can be integrated into other apps or run as a standalone Web-based editor. The beauty of it being Web-based is that your translators can work on a live app from anywhere around the world without having to install any third-party software.

Reporting and Charting

Powerful integrated business reporting capabilities were added to ColdFusion MX 7. The Flex product family includes Flex Charting 2 (a set of charting components).

Using ColdFusion in the Publishing and Content Delivery Phase

Publishing and Distribution

CMS pages can be distributed to the live server using several protocols. CFFTP or the CFSFTP CFC let you transfer files using ftp or the secure sftp protocol.

Syndication

The CFXML tag provides the functionality to create RDF and RSS syndicated news feeds for your or your customer's Web site. This tag creates an XML document that can be converted to a string and written to a file.

You can make any of your ColdFusion components available to any system over the Internet by publishing it as a Web Service. ColdFusion makes this extremely easy to do. In fact, you only have to add one attribute (access="remote") to any of your component's methods to turn it into a Web Service.

Search and Locate

ColdFusion includes the Verity search engine that lets you index and search documents, databases, and existing Verity collections.

Using ColdFusion in the Versioning and Archiving Phase

Storage

JDBC enables ColdFusion MX 7 to interact with a variety of database management systems.

Reactor is an object-relational modeling tool that generates database abstractions on-the-fly as needed, creating an object-oriented database abstraction layer. Objects are regenerated as your database or configuration file changes.

Mark Mandel's Transfer framework automates the repetitive tasks of creating the SQL and custom CFCs that are often required when developing a ColdFusion application. Through a central configuration file Transfer knows how to generate objects and manage them and their relationships back to the database.

A good way to handle SQL trees and hierarchies is the nested set model. Barney Boisvert's TreeManager CFC manages a tree of records in a database table using this model.

Versioning

Rick Osborne's Java SVN browser is a nice example of how you can connect to a Subversion repository. Rick has also created CFDiff. It's intended to provide similar functionality to that provided by the classic Unix diff utility.

Conclusion

This article has been an introductory guide to Content Management with ColdFusion. I've walked through all the phases of the CMS lifecycle and showed you how ColdFusion and Open Source software based on ColdFusion can help you to develop a basic CMS. This was not meant to be an extensive overview of all facets of a CMS, but should provide you with some ideas and links to interesting ColdFusion projects.

Resources

http://cfdj.sys-con.com/read/42120.htm - Spellchecker article
http://cfopen.org/projects/cffm - ColdFusion File Manager
http://cfregex.com/cfcomet - CFComet Web site
http://code.google.com/p/cfdiff - CFDiff project
http://gregs.tcias.co.uk/cold_fusion/cfmx_and_jtidy.cfm - jTidy blog article
http://rickosborne.org/blog/?p=86 – SVN browser
http://trac.reactorframework.com/reactor - Reactor framework
http://transfer.riaforge.org – Transfer framework
http://www.asfusion.com/projects/fileexplorer - AS Fusion File Explorer
http://www.barneyb.com/blog/archives/000532.jsp - Tree Manager
http://www.bennadel.com/blog/79-XStandard-ColdFusion-Web-Services-Solution.htm - XStandard Web Service
http://www.fckeditor.net - FCKeditor
http://xstandard.com - XStandard Editor

About the Author

Harry Klein is co-founder and CTO at CONTENS Software GmbH, a leading supplier of enterprise content management software. He is a Certified Advanced ColdFusion developer and a Microsoft MSCE.

klein@contens.de

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/

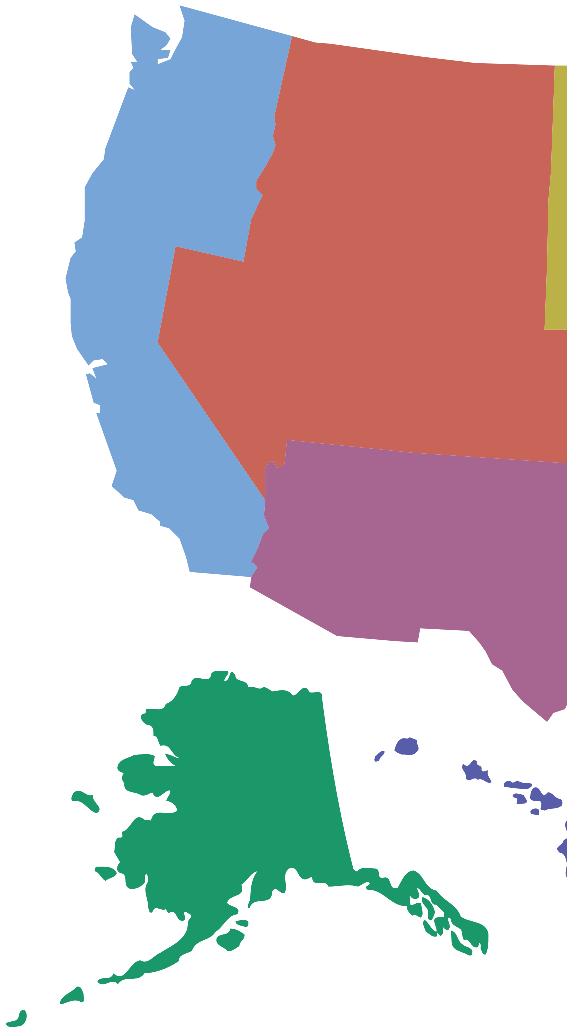
Tennessee
Nashville, TN CFUG
http://www.ncfug.com

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



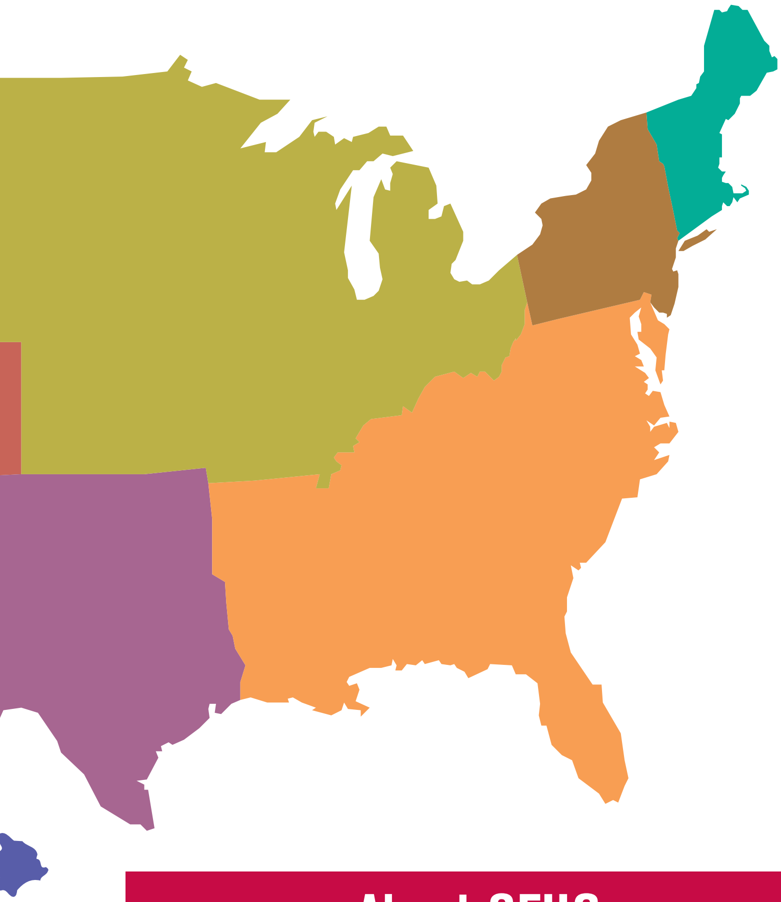
Utah
Salt Lake City, UT CFUG
www.slcfug.org

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

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

User Groups

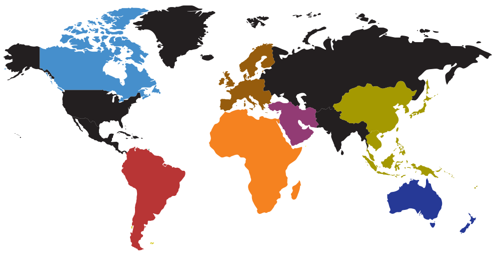
http://www.macromedia.com/cfusion/usergroups



About CFUGs

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

INTERNATIONAL



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

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

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

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

Canada
Kingston, ON CFUG
www.kcfug.org

Canada
Toronto, ON CFUG
www.cfugtoronto.org

Germany
Central Europe CFUG
www.cfug.de

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

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

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

Scotland
Scottish CFUG
www.scottishcfug.com

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

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

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

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

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

Turkey
Turkey CFUG
www.cftr.net

United Kingdom
UK CFUG
www.ukcfug.org

An Introduction to Intent Driven Design

A practical approach to specifying projects

By Peter Bell

Often the hardest part of developing an application is getting agreement on what exactly it should do.

Intent Driven Design (IDD) is an approach that simplifies and standardizes the process of getting detailed technical requirements from non-technical business users so you can develop more complete and consistent requirements in less time.

The Importance of Intent

One of the most frustrating elements of application development is having to continually change the way an application works because of conflicting requirements from different stakeholders. Most such problems come from the lack of a clear focus for a site. You have to agree, in writing, with all of the stakeholders – why you're building the site, who will use it, and what their motivations will be – before you can even start to discuss how the site should work. If you don't, you are guaranteed weeks of wrangling as different stakeholders disagree on fundamental design decisions or priorities because of the lack of a shared vision for the site – often after much of the application has been coded.

The goal of IDD is to provide a standard but lightweight approach to capturing all of the key information that you need to specify a site. For a simple site, you might be able to go through the entire process in a couple of hours. For a more complex project, you might be discussing the issues raised for a number of weeks. Either way, you should end up with a well-structured set of somewhat traceable requirements that you can provide to your development team, and use to turn the high-level business objectives into the appropriate code to achieve those agreed objectives.

No Silver Bullet

In his classic book "The Mythical Man Month," Frederick P. Brooks wrote an entire chapter about an important distinction between "accidental difficulties" and "essential difficulties." For example, memory management/garbage collection is accidentally difficult – it is a problem that can be solved with a smarter programming language (Java or C# instead of C++). However, building a flexible object model for a real world application is actually difficult – there is no "silver bullet" (methodology, IDE,

code generator, etc.) that can make this fundamentally difficult problem simple.

So, IDD is not a silver bullet. It doesn't make it easy to specify complex systems or to handle conflicting requirements. However, it does allow you to clearly and consistently document why you're building the system and what each group of users is expecting from it. It also allows you to "start at the beginning" with a formal process that lets you lock down the objectives which drive the tasks, which drive the screens and actions, which, in turn, drive the data model and the scripts required to implement the requirements. In short, it tries to simplify the process of capturing requirements in a way that doesn't require a degree in computer science to understand!

How Big Is Too Big?

It is often better to build a small project to meet a limited, but valuable, business intent than to try to put your entire business online, delaying the benefits of your supplier extranet or online store, while trying to decide exactly what format your online job application form should take or how to track vacation days online. I have found that if I have more than three to five major intents for a given phase of a project, unless they are being handled by independent teams, I'm probably going to end up with problems.

Too many intents make it hard to keep the focus required for developing a great site, and they increase the likelihood that different intents will conflict, leading to a "nobody likes it, but nobody hates it" kind of compromise design. Overly ambitious projects are also much more likely to ship late or even to fail to ship at all.

This doesn't mean that you can't specify complex applications using Intent Driven Design. It does mean that you might want to break down large, complex projects into smaller projects, and run through the IDD process for each phase. My general rule-of-thumb is that, if the coding (excluding specification, testing, revisions and deployment) for a phase takes more than a couple of weeks, the phase is too big. With application generators becoming more common, there are very few collections of 10-to-20-use cases that take more than a couple of weeks to implement (assuming two-to-five-use cases per programmer over the two weeks, and a maximum team size of five developers before you need a more robust methodology).

These estimates exclude "rocket science," where you're trying to solve fundamentally hard technical problems (such as seamlessly integrating Hibernate with ColdFusion). They also exclude "lab experiments" where issues like performance or load are critical and you may need to build a couple of prototypes before you find the right approach. Such projects can take orders-of-magnitude more developer time to complete, but can still benefit from the IDD approach.

Business Intent

The first step in IDD is to write a paragraph describing the business intent. Why bother building the site? What exactly will it do for your organization? Writing down and agreeing on the business intent upfront makes many of the later decisions easier by creating a shared focus. If the intent is to increase your online sales, the job board and the HR benefits pages are not essential to that intent so they can be pushed off to phase 2.

Roles

What people (and computer systems) are going to use the site? From a requirements-gathering perspective, it is usually best to break down your audience by the collections of tasks they will want to perform (as opposed to by demographics, psychographics or other such approaches). A given user will often perform multiple roles, so you may well have users that are both customers (for buying things online) and administrators (for managing site content), or are both sales people (for accessing the sales reporting system) and employees (for accessing HR benefits online). Roles in a typical project may also include one or more classes of administrators, such as data entry, content editors, approvers and reviewers, and may also include various reporting-based roles.

Roles are not always performed by a person. To capture all of the tasks that the system needs to support, you might also have roles for scheduled tasks, imports, exports or any web services and other interfaces that you provide.

Role Intent

Once you've identified all of the roles, the next step is to document the key reasons why each role would be performed. By documenting why a customer would choose to order online, or what business information a manager is looking for from an online report, you will be able to make better design decisions regarding trade offs such as balancing ease-of-use with flexibility. Also, if there is no good reason for someone to use your website, they probably won't use it. You will only achieve your objectives from your site to the extent that you help your users get what they want out of it. Don't worry if not every role has a meaningful intent, but do go through the process as you'll capture distinctions that will help when you're working on the detailed design.

Compare your role intents with your overall business intents to make sure they are consistent. If you want to increase revenues and your customers want to place orders online 24/7, supporting that intent will meet their needs and yours. However, if your customers want to be able to find the lowest cost supplier, that is probably not an intent you want to support within your site as it is unlikely to help you increase your sales.

For a medium-sized e-commerce system, you can probably document the business intent, roles and role intents on just two or three sheets of paper. Indeed, with a small team (one to five developers), if you have much more than three sheets of intents, you will probably find it hard to deliver a system that can achieve them in a timely manner.

Essential Tasks

Once you have a list of roles and their associated intents, the next step is to figure out what functionality your system must provide to help achieve your business intent by helping your users achieve their goals.

A task is just another name for what most programmers would call a "use case." We've found that by calling it a task, our non-technical clients grasp the concept more easily. A task is a series of screens and actions that allow a user to achieve a valued outcome. Examples might be "order products online," "check order status" or "compose email newsletter." The specification for each task should include any alternate paths (along with their screens and actions) for different choices and for handling error conditions.

Now, instead of having the usual "wouldn't it be cool if the site could . . ." discussions, you can come up with a much more focused list of tasks by asking what exact tasks the site must support to allow each of the roles to achieve their intents, and in a way that helps the organization get what it wants from the site. I tend to focus on essential tasks (as opposed to "nice to haves") as I prefer to deliver the absolute minimum amount of software required to meet a useful set of business and role intents. If you want, you can add "nice to have" tasks depending on the time and budget constraints on your projects, but classify your tasks into essential, useful and optional so that you can deliver the most important functionality first.

Once you have the business intent, roles, role intents and an essential task list, you should have a pretty good sense of the problems that you are solving (both for the organization and for the various classes of site users) and the high level functionality that the site will provide. From there, you can progress to the detailed specification phase.

Fleshing Out the Specs

You can fully specify the requirements for any task by describing the screens with which the user will interact and the actions that each screen supports (as long as you describe screens and actions sufficiently broadly and deeply). In practice, this tends to be a really straightforward way to gather comprehensive requirements quickly from non-technical users.

Screens

Ask the users to describe all of the screens they'd need for each task. To get full coverage, you also have to consider "screens" that are not web pages. If an email or SMS needs to be sent out, that can be considered as a screen, and if you have a web service or, perhaps, an RSS feed, that also has to be described as a screen to determine what information it displays and what functionality (if any) it provides.

Actions

Most screens support one or more actions. An action is a button or link that allows you to do something (submit a form, view a different screen, buy some products, etc.). Typically actions will comprise one or more steps, which usually involve tasks such as getting or modifying data, sending emails, reading or writing files and/or making conditional choices. For example, the "save" action on a "contact us" form would validate the form data, and display a thank you screen. If the data were invalid, it would redisplay the form along with any validation errors returned.

If you start with the landing screen, get a list of all of its actions, and then repeat for every screen that every action can take you to, you'll eventually get a complete list of screens and actions for that task.

You also need to consider non-standard actions such as receiving email messages or XML requests, or even scheduled requests from a scheduled task, to get complete coverage of the project requirements.

Filling in the Details

Once you've got a good set of screens and actions, you can then start to fill in the details. For each screen you can clarify the fields to display, filter, order by, etc. For each action you can get more detailed information about all of the steps and any business rules required.

MVC Modeling

The great thing about this approach is that it flows perfectly into a Model-View-Controller style architecture – whether you are using a community framework like Fusebox, Mach-II or Model Glue, or whether you are rolling your own.

Every screen corresponds to a view, every action corresponds to a controller method (an event in Model Glue or Mach-II, a fuseaction in Fusebox) and most of the steps that make up an action describe method calls against your model to do things like save an article, get a list of users or get information on a product to display. This makes it extremely easy to flow from the detailed specification into developing the code, and very easy to relate each event, method call and view to the task it supports and hence the role(s) and intent(s) that it is meant to support.

Technical Design

Once you have completed the IDD process, you should be ready for the technical design phase. In that phase you'll consider issues like the appropriate architecture to use, what framework (if any) to implement and how you are going to handle everything from transaction management and error logging, to user authentication and application security.

Once you have that locked, you should be ready to code an application that comes pretty close to meeting your users' expectations first time round – especially if you prototype all of the screens and actions as part of the specification process. You will also have information on all of the fields being imported, exported, listed, viewed, filtered, searched and ordered by and for each object (product, user, article, etc.), so you should find it

"A task is a series of screens and actions that allows a user to achieve a valued outcome."

fairly straightforward to develop a database schema if you don't already have one to work from.

Conclusion

Intent Driven Design is not a complete requirements-gathering process for large, well-funded projects. It doesn't include any explicit discussion of non-functional requirements (performance, latency, load, environment, etc.). It also doesn't include marketing strategy, market research or formal usability testing. Those are usually outside of the budget for smaller organizations (although they can be added to the process quite easily if required).

However, it is a proven, quick and effective methodology for capturing a useful set of functional requirements in a way that easily translates into well-architected working code, and that allows everyone on the project to quickly understand what they're coding and why it matters. .

About the Author

Peter Bell is CEO/CTO of SystemsForge (http://www.systemsforge.com) and helps Web designers to increase their profits and build a residual income by generating custom web applications - in minutes, not months. An experienced entrepreneur and software architect with fifteen years of business experience, he lectures and blogs extensively on application generation and ColdFusion design patterns (http://www.pbell.com).

pbell@systemsforge.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

© Copyright 2005 PaperThin, Inc. All rights reserved.

The RIA Team Concept

Maximize developer efficiency

By Andrew Powell

One thing that constantly bugs me on projects is when I am asked to work on the user interface. I can do CSS, but it is, admittedly, not my strongest suit. I can do some graphics work, but not my strongest suit.

My time, and the client's money, is best spent on me maximizing ColdFusion's potential. I am not the greatest at Flash. I can create AS classes to do things like Flash Remoting, etc., but I am not so good at the interface part of Flash. I am aware of my client-side shortcomings and am perfectly comfortable with this.

There are some client-side developers, such as my Flash developer friends, who are more than happy to keep their focus on the client-side and not have anything to do with the server-side of the application. Their time and, again, the client's money, is best spent when they are working on the client-side of the project. These guys are the ones who kick out killer interfaces and deliver some of the coolest things we've seen with Flash and AJAX. All they want from the server side is the data and they'll take it from there, thank you very much.

Some Flash developers and I both understand the value of at least understanding each other's focus though. I can look at their ActionScript and understand what they are doing. They can look at my CFCs and understand what I'm doing. We may not be able to create the same code, but we can understand what it is saying and offer suggestions if something is awry. This is where the server-side developer and the client-side developer can complement each other. We don't have to be able to do each other's job; we just need to have a general understanding of what is going on to be able to fine-tune our piece of the project to fit with the rest of the team.This meeting of the minds is just what "The RIA Team Concept" is built around. A strong RIA Team consists of two to three members: a server-side developer, a client-side developer, and an optional team leader who can coordinate the two sides, i.e., a project manager with some development skills. Sometimes the server-side or client-side developer can fill the team leader role, but not always.

The RIA Team Concept is not radical, but practical. Traditional Web developers have historically been able to code on the server-side and client-side. There are very few developers who are strong at both aspects of Web development. Web developers are traditionally stronger on one side of the coin or the other, but are often called upon to work on both aspects of a project. The RIA Team Concept allows developers to focus their skills where they are the strongest, thereby maximizing value from their billable time.These RIA Teams can efficiently complete projects and maximize value to the client because they are, before anything else, confident in the skills of the other members of the team.

They know that, just as they are able to take care of their piece of the application, their teammates can do just as good a job on their piece. This kind of trust and confidence among team members leads to better communication and better applications because each member of the team knows what the other is looking for from each other. These are non-technical skills that every project team needs to survive and thrive.The benefits of the RIA Team Concept for the client are clear: a better product end-to-end, a faster turn-around, and a much better ROI than previously available. Allowing developers to specialize in an area ensures that they focus on making their piece of the application as solid and efficient as it can be. Each piece of the application is completed faster because developers are allowed to focus their time on their segment of the application. The ROI is increased because the application can be delivered faster and start generating value for the client much sooner than before. The RIA Team Concept is one that can maximize your developers' efficiency and make them feel more like a true team that depends on each other. This provides many benefits for a project because development time is decreased, quality is increased, and value is able to be generated by the project at a quicker rate. These are benefits any project manager, or company, would be glad to enjoy.

About the Author

Andrew Powell is a senior consultant for Universal Mind (www.universalmind.com). He specializes in development, incident resolutions, and performance tuning for ColdFusion, Mach-II, Spry, and Flex. Andrew has been developing Web applications since 1998 using ASP, ASP.NET, and ColdFusion. He holds a BBA in Computer Information Systems, Finance, and Management from Mercer University. You can read his blog at www.infoaccelerator.net .

andrewp@univeralmind.com

A ColdFusion Service Management System Streamlines Operations

Improves Service to Deaf and Hard of Hearing

By Nader Ayoub

In today's global economy, service providers are under increasing pressure to maximize margins and streamline operational efficiency to remain competitive.

In order to do this, providers are turning to new technologies, advanced collaboration with partners and creative service strategies to ensure a well-managed workforce. They are also creating virtual workforces where resources may work from home or on the road over a disparate geographic region. To this end, service providers are seeking to reduce or all together eliminate the brick and mortar office, and replace it with a loose coalition of people with diverse skill sets. As this shift continues in the service industry, so do their requirements. For example, they are now required to handle the entire service order management process by continually updating their offerings to stay competitive against others in the same industry; i.e., doing more with less.

The Catholic Community Services (CCS) serves the Deaf and Hard of Hearing community in Arizona and New Mexico by providing sign language interpreting, vocational assistance, counseling and other services. CCS and its agencies in Phoenix, Tucson and New Mexico, have provided interpreter referral services over 25 years. In recent times, the three agencies determined that they had outgrown their custom software, which was inflexible in dealing with changes in the market, and which had been modified locally in each city, resulting in incompatible platforms. CCS needed, not only a way to operate their backend office across three locations, but also needed new ways to increase revenue and service offerings. In order to update their service technology, they elected to go with a web-based solution to handle service collaboration, fulfillment and administrative activities.

Best Solution

After a thorough pilot development and evaluation, it was decided to use ColdFusion to develop the CCS service management solution which automates all aspects of personnel scheduling, time recording, billing and invoicing. Overall, the solution comprises five parts:

1. ***Service Order Collaboration***: Scheduling and collaboration capabilities are at the heart of the system. By providing greater visibility into workforce assets, administrators can deploy skilled resources by reviewing details on the number and type of associates needed for a given day and time, while keeping overtime parameters and budget objectives in the mix. The web-based platform identifies the best-matched resource, notifying and receiving confirmation from all participants.
2. ***Service Fulfillment***: Once the proper resource has been identified and assigned, the system automatically tracks each job until completion.
3. ***Service Settlement***: The system automates pay rules and ensures compliance with government regulations and contracts. It tracks service claims, ledger activity, financial adjustments, customer invoices and statements, and payments.
4. ***Administrative Platform***: The system includes a centralized, web-based interface for managing services, service regions, resources, accounts, service calendars, organizational hierarchies, consumers, pay rates, terms and conditions, as well as others.
5. ***Reporting & Analysis***: This allows organizations to track key performance indicators (KPIs) and make informed decisions for establishing best practices to ensure increased productivity at all service levels.

To summarize, the CCS solution enables management to automate and streamline all aspects of the field service operations, allowing providers to offer additional services with greater accessibility to ensure growth and fitness for service. The solution's architecture allows virtually unlimited numbers of people to interact, conduct business and receive services through a collaborative environment.

Why ColdFusion

ColdFusion was a clear winner for a number of reasons, chief among them its ability for Rapid Application Development (RAD) and prototyping. The goal in developing a workforce management solution was not to focus on the technology, but

REGISTER TODAY AND $AVE!

Rich Internet Applications: AJAX, Flash, Web 2.0 and Beyond...

www.AjaxWorldExpo.com

AjaxWorld™ East Conference & Expo

NEW YORK CITY

THE ROOSEVELT HOTEL LOCATED AT MADISON & 45th

SYS-CON Events is proud to announce the AjaxWorld East Conference 2007!

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 unique, timely conference – especially the web designers and developers building those experiences, and those who manage them.

BEING HELD MARCH 19 - 21, 2007!

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 in learning about a wide range of RIA topics that can help them achieve business value.

instead on the business problem and ensuing solution. ColdFusion offered the best support for developing an extensive service system encompassing over 3,000 web pages, accessible from various devices, while providing a means to expand daily, embracing new technologies, enhanced capabilities and additional uses.

ColdFusion let us centralize database access, and handle requests for over 1,000 SQL statements and debug rapidly. In order to create a large-scale, web-based system, we constructed a ColdFusion framework and scaled it to handle the entire application. By leveraging reusable components and the readily extensible language itself, we were able to create and change literally thousands of pages rapidly, giving us a better opportunity to adapt the technology to the business, as opposed to other way around. We were able to change and migrate to many Web servers and database engines without third-party plug-ins, and with few revisions as the solution evolved. Additionally, it provided out-of-the-box advantages over many other products on the market. In other words, we focused on solving the personnel

resource scheduling problem, independent of the technology.

ColdFusion held the shortest learning curve of any language in the web space, moreover, with its rapid development capabilities, the real cost in software development was not in the setup, but in the cost of adjusting the software to meet new and burgeoning business requirements.

We managed to displace the initial software costs by selecting a hosting partner. By using CFDynamics, we were able to rapidly set up a dedicated environment for ColdFusion and SQL Server with support, data backups and 24/7 monitoring-- allowing us to focus on core development in an environment accessible from anywhere. Between ColdFusion and CFDynamics hosting, we were able to focus on the business because they took care of the technology. In essence, Adobe's ColdFusion became a vehicle to deliver a world-class, large-scale, multi-organization, multi-user and feature-rich environment in a relatively short period of time.

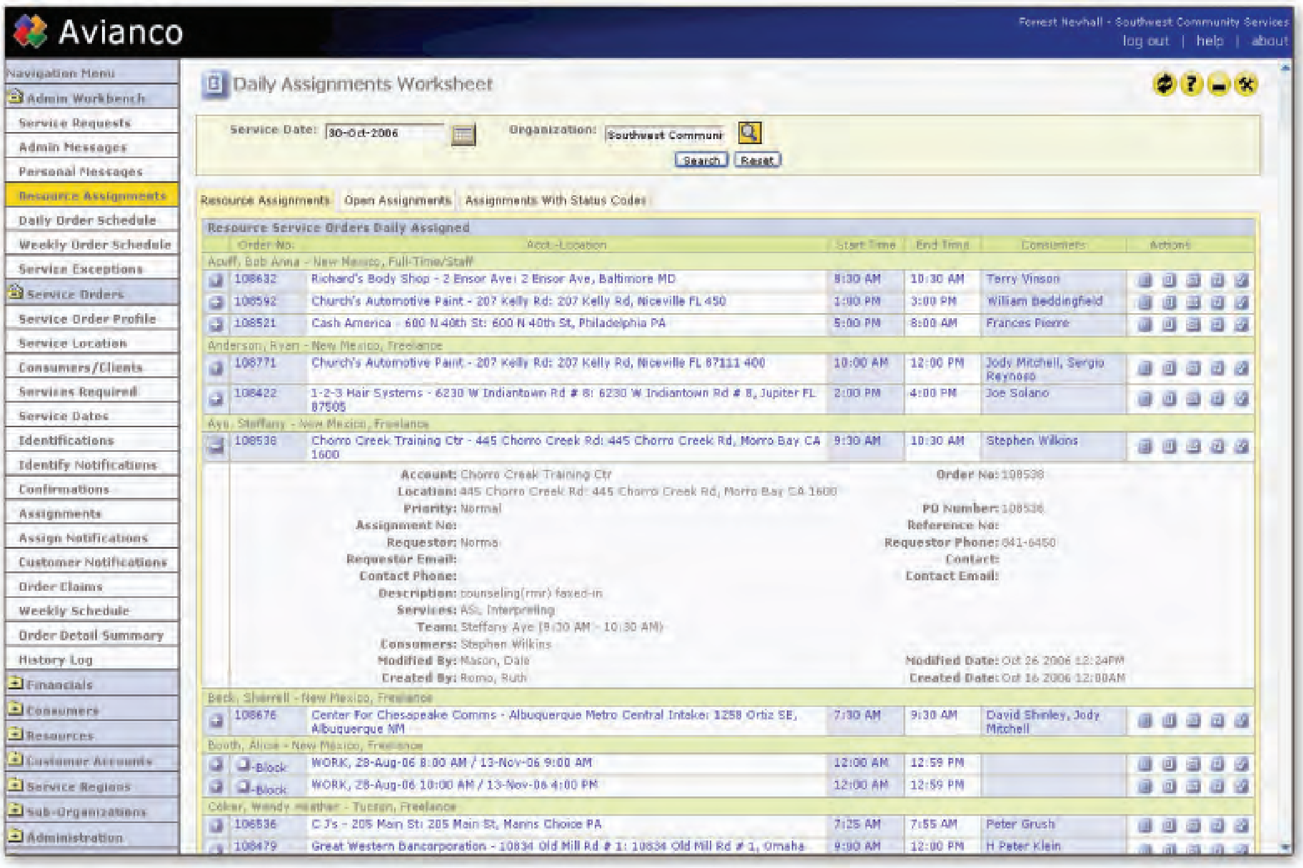
A Win-Win Program

The Catholic Community Services (CCS) witnessed immediate and real results after implementing the ColdFusion system: improved operational efficiency, reduction in time for billing and payroll processing, and the reduction of data entry errors through elimination of redundant data entry. The organization is now better equipped to provide the highest quality service possible, while reducing its labor costs. They are able to field more service requests and accelerate service fulfillment. They can coordinate various contract personnel in remote locations, match resources with the job requirements, resolve conflicts, collect billing information, and track a service order until payment completion. In general, the added benefits include:

- Improved personnel scheduling and matching
- Reduction in travel costs
- Elimination of data entry errors
- More satisfied workers
- Faster invoice processing and payments
- Reduced call load with customer self-service
- Increased visibility for better decision making

The solution allows administrators to schedule service personnel across three geographic locations encompassing statewide services in Arizona and New Mexico including rural areas, keep track of work performed, expenses, and billing, while coordinating over 400 resources and service personnel. Among those who benefit from the solution are:

- ***Administrators***: From one location, administrators can handle all the complexities of a service order (from creation to assignment), resource confirmation, claims of services provided, final billing and invoicing. Billing statements reflect current account status and reduce collection activity for unpaid invoices in previous billing cycles. Avianco accelerates these operations as administrators can execute all order-related activities from a single screen. Administrators gain better visibility into daily operations by retrieving real-time data on employees' time, schedule, vacation, and overtime, and contractors' availability and preferences. Accordingly, they now have the ability to dynamically swap resource schedules to meet changing needs.
- ***Interpreter Resources***: The system offers a new level of personnel participation and self-management in the scheduling process. Interpreters get a convenient and simple way to see their future schedule online, without having to call in or receive daily paged or emailed schedules. By directly accessing information and reporting their time, interpreters feel as though they are an integral part of the solution -- contributing to the overall success of the organization. Accessing the system in the field via a wireless device (PDA, tablet PC, notebook, even cell phone), interpreters can retrieve work orders, obtain directions to their next site, and instantly submit hours and close orders from the field. With the online solution, CCS has significantly increased interpreters' satisfaction and retention.
- ***Customers (the groups or businesses that provide payments for the clients)***: By introducing an online self-service ability,


Avianco

Forrest Nevhall - Southwest Community Services
log out | help | about

Navigation Menu
- Admin Workbench
  - Service Requests
  - Admin Messages
  - Personal Messages
  - Resource Assignments
  - Daily Order Schedule
  - Weekly Order Schedule
  - Service Exceptions
- Service Orders
  - Service Order Profile
  - Service Location
  - Consumers/Clients
  - Services Required
  - Service Dates
  - Identifications
  - Identify Notifications
  - Confirmations
  - Assignments
  - Assign Notifications
  - Customer Notifications
  - Order Claims
  - Weekly Schedule
  - Order Detail Summary
  - History Log
- Financials
- Consumers
- Resources
- Customer Accounts
- Service Regions
- Sub-Organizations
- Administration

Daily Assignments Worksheet

Service Date: 30-Oct-2006 Organization: Southwest Communi

Search Reset

Resource Assignments | Open Assignments | Assignments With Status Codes

Resource Service Orders Daily Assigned

Order No:	Acct.-Location	Start Time	End Time	Consumers	Actions
Acuff, Bob Anna - New Mexico, Full-Time/Staff					
108632	Richard's Body Shop - 2 Ensor Ave: 2 Ensor Ave, Baltimore MD	9:30 AM	10:30 AM	Terry Vinson	
108592	Church's Automotive Paint - 207 Kelly Rd: 207 Kelly Rd, Niceville FL 450	1:00 PM	3:00 PM	William Beddingfield	
108521	Cash America - 600 N 40th St: 600 N 40th St, Philadelphia PA	5:00 PM	8:00 AM	Frances Pierre	
Anderson, Ryan - New Mexico, Freelance					
108771	Church's Automotive Paint - 207 Kelly Rd: 207 Kelly Rd, Niceville FL 87111 400	10:00 AM	12:00 PM	Jody Mitchell, Sergio Reynoso	
108422	1-2-3 Hair Systems - 6230 W Indiantown Rd # 8: 6230 W Indiantown Rd # 8, Jupiter FL 87505	2:00 PM	4:00 PM	Joe Solano	
Aye, Steffany - New Mexico, Freelance					
108538	Chorro Creek Training Ctr - 445 Chorro Creek Rd: 445 Chorro Creek Rd, Morro Bay CA 1600	9:30 AM	10:30 AM	Stephen Wilkins	

Account: Chorro Creek Training Ctr
Location: 445 Chorro Creek Rd: 445 Chorro Creek Rd, Morro Bay CA 1600
Priority: Normal
Assignment No:
Requestor: Norma
Requestor Email:
Contact Phone:
Description: counseling(rmr) faxed-in
Services: ASL Interpreting
Team: Steffany Aye (9:30 AM - 10:30 AM)
Consumers: Stephen Wilkins
Modified By: Mason, Dale
Created By: Romo, Ruth

Order No: 108538
PO Number: 108538
Reference No:
Requestor Phone: 841-6450
Contact:
Contact Email:
Modified Date: Oct 26 2006 12:24PM
Created Date: Oct 16 2006 12:00AM

Order No:	Acct.-Location	Start Time	End Time	Consumers	Actions
Beck, Sherrell - New Mexico, Freelance					
108676	Center For Chesapeake Comms - Albuquerque Metro Central Intake: 1258 Ortiz SE, Albuquerque NM	7:30 AM	9:30 AM	David Shinley, Jody Mitchell	
Booth, Alice - New Mexico, Freelance					
-Block	WORK, 28-Aug-06 8:00 AM / 13-Nov-06 9:00 AM	12:00 AM	12:59 PM		
-Block	WORK, 28-Aug-06 10:00 AM / 13-Nov-06 4:00 PM	12:00 AM	12:59 PM		
Coker, Wendy Heather - Tucson, Freelance					
106536	C J's - 205 Main St: 205 Main St, Manns Choice PA	7:25 AM	7:55 AM	Peter Grush	
108479	Great Western Bancorporation - 10834 Old Mill Rd # 1: 10834 Old Mill Rd # 1, Omaha	9:00 AM	12:00 PM	H Peter Klein	

customers can create their own service requests at any time. Many customers welcome the speed and convenience of creating requests and checking status without administrator intervention. Of course, CCS acknowledges that not all of its customers are comfortable solely using the online self-service feature. However, the system has significantly reduced staff workload so they are free to handle new accounts over the phone.

- ***Deaf and Hard of Hearing Clients (the ultimate recipient of services)***: Giving the clients online access is a significant byproduct of the system. Clients could check on schedules and retrieve documents prior to an interpreter's visit.

The system also allows CCS to handle all assignments beyond interpreters, where other departments, such as Client Services are able to handle staff scheduling and coordination. Finally, the software is instrumental in setting up new services previously unavailable, such as Support Service Providers and Community Outreach programs.

What's Next

By leveraging ColdFusion, we are continuing to explore additional services and options for automation. One such service offered by the system is video interpreting through the internet, IP-message relaying, and text messaging via ColdFusion's gateway capabilities. Soon we will include an online payment option, allowing customers to review invoices online and post payments to their accounts. We are also exploring other ways ColdFusion's technology can benefit CCS and the deaf community at large.

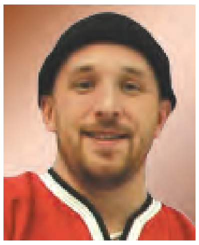
About the Author

Nader Ayoub is president and founder of Avianco.

ayoub@avianco.com

An introduction to AJAX and Taconite

Yes, it works



By Tom Schreck and Jason Doyle

AJAX stands for Asynchronous JavaScript and XML. The core of AJAX is the use of the XMLHttpRequest() object to communicate with server-side scripts. It can send and receive information in XML, HTML, and plain text.

The "Asynchronous" part means the client can communicate with the server without page refreshes. This allows you to update portions of a page based upon user events and is what allows developers to create Rich Internet Applications (RIA) using existing technologies.

Introducing Taconite

Taconite is a framework that simplifies the creation of AJAX-enabled Web applications. It's a very lightweight framework that automates the tedious tasks related to AJAX development, such as the creation and management of the XMLHttpRequest object and the creation of dynamic content. Taconite can be used with all modern Web browsers (Firefox, Safari, IE, Opera, and Konqueror, to name a few) and can be used with any server-side technology, including Java EE, .NET, PHP, ColdFusion or any language that lets you return XHTML.

The XMLHttpRequest object is not officially part of any Web standard, including W3C standards. Hence, the W3C created the Document Object Model Load and Save specification. Taconite is based on the concepts behind the W3C's Document Object Model Load and Save specification.

The heart of Taconite is a parser that converts normal (X)HTML code into a series of JavaScript commands that dynamically create the content on the browser. This parser allows the developer to write content in a way that is natural – as (X)HTML. You no longer have to crowd your pages with a slew of document.createElement and document.appendChild commands to dynamically create new content. The resulting JavaScript is evaluated by the browser's JavaScript engine to produce the content specified by the (X)HTML. All you need to do is make sure that the content being generated is valid XHTML. That means single tags must be self-closing, attributes need to be in quotes, special characters need to be escaped and attribute minimization must be avoided (i.e., nowrap should be nowrap="nowrap").

The Taconite custom parser is implemented as a set of JSP custom tags that can be used in any Java servlet container, or as a client-side JavaScript library, meaning it can be used in conjunction with virtually any server-side technology.

Taconite focuses on removing the tedious task of writing large amounts of JavaScript to dynamically update content on a Web page. Unlike some frameworks that try to invent a new way of sending request data to the server, Taconite relies on the tried-and-true name/value pairs either embedded within the query string of a GET request, or tucked safely into the body of a POST. The client-side library even automates the creation of the name/value query string. All you need to worry about is telling Taconite what values should be sent to the server, and it takes care of the rest.

Since the JavaScript is created by the parser based on the HTML you specify, you don't have to worry about dealing with browser JavaScript incompatibilities. The parser takes care of them all for you. All you need to do is specify the dynamic content as HTML, and the parser does the rest.

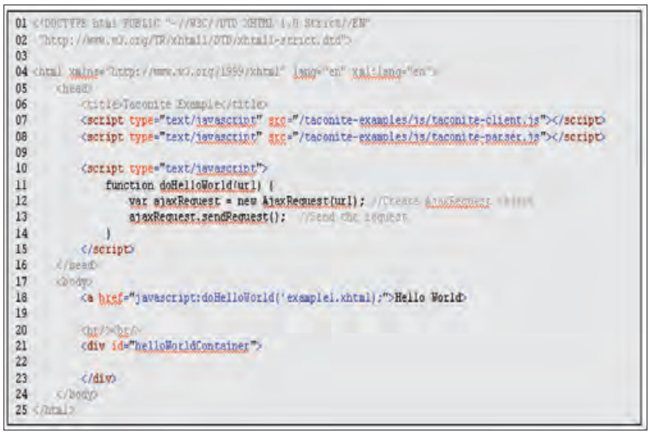
InnerHTML Avoided

Many of the currently available AJAX frameworks rely on the innerHTML property to dynamically create and update content on the Web page. Taconite completely avoids the use of innerHTML.

The main problem with innerHTML is that it's non-standard. It is not specified by any W3C recommendation, although it is supported by most of today's browsers. Another problem with innerHTML is that it really only applies to HTML documents, not XHTML documents, which is what the industry is moving to. As browsers move to better, more strict support for XHTML documents, they may not support innerHTML due to its non-standard nature.

Coming as a surprise to absolutely nobody, the innerHTML property has its own browser-to-browser quirks. For example, IE won't allow the innerHTML of a <table> to be appended. So, if you want to append one row to a table using innerHTML, you'll have to reproduce the entire contents of the table tag, with the one new row added at the end. That's just a waste of processing time and bandwidth.

Using the innerHTML property doesn't lend much control



```
01 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
02  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
03
04 <html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
05     <head>
06         <title>Taconite Example</title>
07         <script type="text/javascript" src="/taconite-examples/js/taconite-client.js"></script>
08         <script type="text/javascript" src="/taconite-examples/js/taconite-parser.js"></script>
09
10         <script type="text/javascript">
11             function doHelloWorld(url) {
12                 var ajaxRequest = new AjaxRequest(url); //Create AjaxRequest object
13                 ajaxRequest.sendRequest();  //Send the request
14             }
15         </script>
16     </head>
17     <body>
18         <a href="javascript:doHelloWorld('example1.xhtml);">Hello World>
19
20         <br/><br/>
21         <div id="helloWorldContainer">
22
23         </div>
24     </body>
25 </html>
```

Figure 1: Hello World!

when it comes to placing the new content. For instance, if you want to insert a table row between two existing table rows using the innerHTML property, you would have to reproduce the entire table. Taconite gives you the flexibility to specify where content should be placed. In fact, this concept is straight out of the W3C DOM Load and Save specification.

Figure 1 shows the client HTML page. Note that there is only one link on the page (line 18). Clicking the link sends an AJAX request to the server, and the server's response is appended to the page.

Lines 7 and 8 reference Taconite JavaScript files. The taconite-client.js file is always required when Taconite is used. This file defines the AjaxRequest object, which is the heart of the Taconite client-side library. The taconite-parser.js file references the JavaScript-based Taconite parser. It is only required when you wish to perform the XHMTL-to-DOM parsing on the browser rather than on the server.

Lines 11-14 define the doHelloWorld function. Thanks to the Taconite framework, it's only two lines long. The first line creates an instance of the AjaxRequest object. The AjaxRequest object, as described by its name, encapsulates an AJAX request. Each instance of an AjaxRequest object creates its own instance of the XMLHttpRequest object, so you can create as many as you like without worrying about instances of XMLHttpRequest overwriting one another.

Line 18 defines the link that initiates the AJAX request when clicked. The link calls the doHelloWorld function, passing to it the URL of the AJAX service to be called.

The AjaxRequest object is created by passing a single argument to the object constructor. The argument is the URL to which the AJAX request will be sent.

The second line of the doHelloWorld function sends the request to the server. The AjaxRequest object defines a sendRe-

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!

Visit us today at:
hotbanana.com/webinar
or call 1-866-296-1803
to schedule a
live 1-on-1 demo.

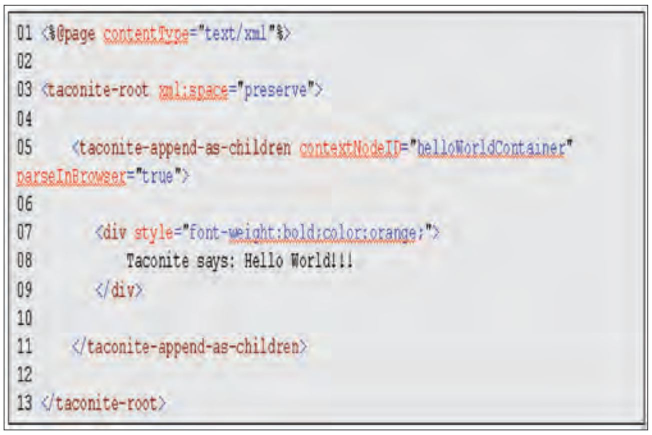
hotbanana®

"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.



```
01 <%@page contentType="text/xml"%>
02
03 <taconite-root xml:space="preserve">
04
05     <taconite-append-as-children contextNodeID="helloWorldContainer"
parseInBrowser="true">
06
07         <div style="font-weight:bold;color:orange;">
08             Taconite says: Hello World!!!
09         </div>
10
11     </taconite-append-as-children>
12
13 </taconite-root>
```

Figure 2: Writing the Server Response

quest method, which is called here to send the request. The rest is defined on the server, which we'll look at next.

Line 1 in Figure 2 is a directive indicating that the Content-Type of the page should be returned as text/xml. All Taconite responses must be returned with a Content-Type of text/xml, otherwise, the XMLHttpRequest object won't be able to parse it as an XML document. This file can be reproduced by any server-side technology; just set Content-Type to be text/xml.

Line 3 has the taconite-root XML tag, which is the root tag of all Taconite responses. All Taconite responses must have the taconite-root tag as the root tag. The presence of this tag ensures that the response is well-formed XML. Note that the taconite-root tag has an attribute of xml:space="preserve." This is to ensure that Internet Explorer handles whitespace correctly.

Line 5 opens the taconite-append-as-children tag. This tag says that the child elements of this tag will be appended as children to the context node. The context node is specified by the contextNodeID attribute. The value of this attribute must correspond to some element on the XHTML page. Remember the empty div element on Line 23 of the XHTML page, which had an ID attribute value of helloWorldContainer? This is where the message will be appended, because the contextNodeID attribute of the taconite-append-as-children tag has a value of helloWorldContainer.

The second attribute on the taconite-append-as-children tag is parseInBrowser, which has a value of true (when not using the JSP tags, parseInBrowser must always be true). This tells the AjaxRequest object that it needs to parse the response into JavaScript before running the JavaScript to produce the "Hello World" message.

Finally, the "Hello World!" message is defined on Lines 7-9. Notice how the message is simply defined as valid XHTML. In fact, the div tag even has an embedded style that makes the message font orange in color! This is the beauty of Taconite: you simply specify the new or updated content as regular ol' XHTML, and Taconite does the rest, without resorting to using the pesky innerHTML property.

Using Taconite with ColdFusion

AJAX really improves your user's experience. Your Web page appears to load faster because your application is responding to your user and only loading the necessary content. Entire page reloads are no longer necessary. How you approach designing your page changes. You become focused on how to respond to your user's actions. Your code becomes more precise and more manageable.

Incorporating Taconite and AJAX into your ColdFusion development is seamless. Typically, you do the following:

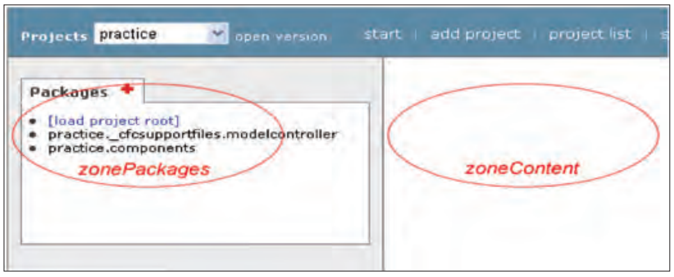
1. ***Determine the point in your app where your user will kick off an AJAX request***. Typically, this is a link or a button that, when clicked, calls a JavaScript function that starts the AJAX request process. This step may pass data to the JavaScript function that will be sent to the server for processing.
2. ***Determine what area(s) on the page will be modified/updated on the client***. An AJAX response can change the content in several locations on the client.
3. ***Determine what server-side file should be called to process the AJAX request***.
4. ***Determine what business logic needs to happen inside the server-side file using the parameters passed from the AJAX request***.
5. ***Construct the response(s) you want to send back to the client page***.

The beauty of AJAX, particularly Taconite, is in the ability to return multiple content blobs to your client. All that is needed is for the server-side file to know the IDs (contextNodeIDs) of the containers that will be updated. You can incorporate the IDs that will be updated in the AJAX request for a truly dynamic experience, or have the contextNodeIDs hard coded in the server file.

Below are examples of how I incorporated the steps above inside cfcPowerTools (www.cfcpowertools.com). cfcPowerTools is a Web-based application used to batch generate ColdFusion Components (CFCs) from database tables.

1. ***Determine the point in your app where your user will kick off an AJAX request***. The end user creates a project used to organize CFCs. The UI has a drop down for selecting a project. AJAX is used to load all of the Packages within a project and to clear out the main content area. The Project dropdown passes the projectID to the JavaScript function that initiates a Taconite call.

```
//GetPackages
01 function GetPackages(projectID)
02{
```



```
03      var url = “getPackages.cfm”;
04      var packageRequest = new AjaxRequest(url);
05      packageRequest.setQueryString(‘projectID=’ + projectID);
06      //packageRequest.setEchoDebugInfo();
07      packageRequest.sendRequest();
08}
```

The GetPackages JavaScript function accepts the projectID that will be appended to the URL. The URL is defined in line 3. The AJAX request is created in line 4. The projectID is appended to the query string in line 5. You can turn debug on/off using line 6. Line 7 sends the request.

2. ***Determine which area(s) on the page will be modified/updated on the client***. By selecting a Project, the Packages tab is updated and the main content area (white part) is cleared out. Below are the containers on the client awaiting content to be AJAX’d. Notice the ID values of each <div>:

```
<div id=”zonePackages” class=”leftNavDivs”></div>
<div id=”zoneContent”></div>
```

3. ***Determine which server-side file should be called to process an AJAX request***. The GetPackages JavaScript function calls getPackages.cfm server file (line 3). This file is responsible for processing any business logic and assembling an AJAX response.
4. ***Determine which business logic needs to happen inside the server-side file using the parameters passed from the AJAX request***. Refer to Figure 3. Line 5 is the business logic. It simply sets a session variable using the passed projectID.
5. ***Construct the response(s) you want to send back to the client page***. Lines 8-12 construct the AJAX response. Line 8 replaces zonePackages with the contents of packageList.cfm. Line 12 clears out zoneContent.

Notice that the response is created within <cfsavecontent> tag (line 6). This allows you to better control the content being returned. The content is stored in variables.Content variable. Also, notice everything happens inside <cftry> block (line 4). This allows you to create an error message to be AJAX’d back to the client. Notice that inside <cfcatch> (line 14) another <cfsavecontent> (line 15) produces an error response (also named variables.Content). It returns content to an error div (<div id=”zoneError”/>).

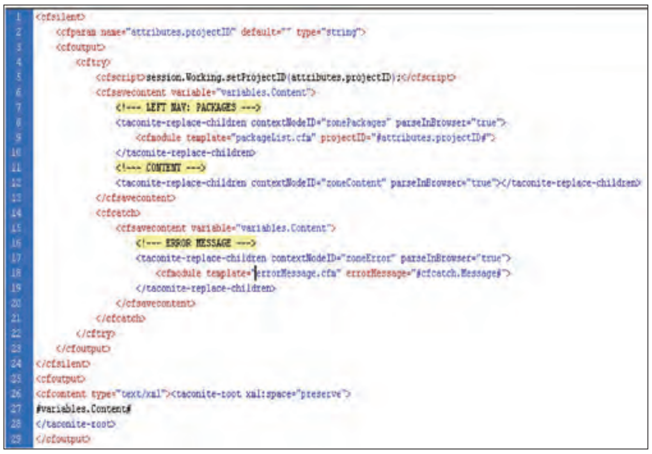
Regardless of whether an error happens, or the page successfully renders, content is returned to the client. Line 31 injects variables.Content into the Taconite response.

Notice the use of <cfcontent type=”text/xml”> (line 26).

AJAX Tips

Here are some tips you may want to consider in your development:

1. ***Custom debug functionality***. I found debugging Taconite to be a bit cumbersome. I’ve added a Taconite response to each page:



```
1  <cfsilent>
2      <cfparam name="attributes.projectID" default="" type="string">
3      <cfoutput>
4          <cftry>
5              <cfscript>session.Working.setProjectID(attributes.projectID);</cfscript>
6              <cfsavecontent variable="variables.Content">
7                  <!--- LEFT NAV: PACKAGES --->
8                  <taconite-replace-children contextNodeID="zonePackages" parseInBrowser="true">
9                      <cfmodule template="packageList.cfm" projectID="#attributes.projectID#">
10                 </taconite-replace-children>
11                 <!--- CONTENT --->
12                 <taconite-replace-children contextNodeID="zoneContent" parseInBrowser="true"></taconite-replace-children>
13             </cfsavecontent>
14             <cfcatch>
15                 <cfsavecontent variable="variables.Content">
16                     <!--- ERROR MESSAGE --->
17                     <taconite-replace-children contextNodeID="zoneError" parseInBrowser="true">
18                         <cfmodule template="errorMessage.cfm" errorMessage="#cfcatch.Message#">
19                     </taconite-replace-children>
20                 </cfsavecontent>
21             </cfcatch>
22         </cftry>
23     </cfoutput>
24 </cfsilent>
25 <cfoutput>
26 <cfcontent type="text/xml"><taconite-root xml:space="preserve">
27 #variables.Content#
28 </taconite-root>
29 </cfoutput>
```

Figure 3

```
<!--- SHOW DEBUG --->
<cfif request.ynDebug>
   <taconite-replace-children contextNodeID=”zoneDebug”
parseInBrowser=”true”>
   <cfmodule template=”debug.cfm”>
</taconite-replace-children>
</cfif>
```

I have a <div id=”zoneDebug”/> on the client page. It sits there waiting to receive debug information. If debugging is turned on, then I include a custom debug page whose contents are returned to the debug div.

2. ***AJAX requires the use of JavaScript***. So, you will need to be sure your users allow JavaScript functionality in their browsers.

Conclusion

There are many layers to AJAX, but using just the basic functionality can greatly improve the richness of your application. Incorporating AJAX using the Taconite library is very easy. Following the steps and examples above you can inject AJAX functionality into your applications.

References

http://taconite.sourceforge.net/

About the Authors

Tom Schreck is a Macromedia certified ColdFusion MX 6.1 developer. He has been working with ColdFusion since 1997. Check out www.cfcPowerTools.com for more information on cfcPowerTools.

Jason Doyle is a Web systems engineer at Thomson. He is currently focusing on Rich Internet Applications with the use of mash-ups in a service-oriented environment.

tkschreck@sbcglobal.net
jason.doyle@thomson.com

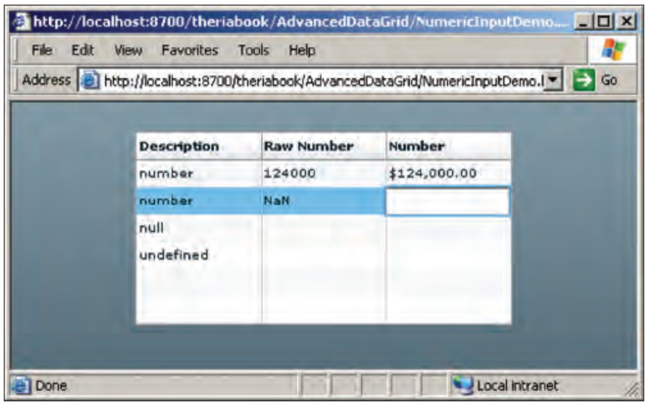
Adobe Flex 2: Advanced DataGrid

– CONTINUED FROM PAGE 13

Alpha keys will be blocked, but the completeness of the phone is up to the user. The control is an extension of mx.controls.TextInput and its text returns the "mask-free" data input, i.e., 61755551212, in our use case. The main controlling property of MaskedInput is inputMask, which can consist of any characters except:

- #: A single digit
- C: A letter (no digits allowed)
- c: Forces a letter to lowercase (no digits allowed)
- A or a: Allows any character.

Like for all other components in this series of articles, we have added a MaskedInput to theriabook.swc and registered it in the component manifest file – theriabook-manifest.xml. The only modifications we made to the original MaskedInput were changing the original packaging of MaskedInput to com.theriabook.controls and overriding the implementation of the TextInput data setter to accommodate its use in the DataGrid:



http://localhost:8700/theriabook/AdvancedDataGrid/NumericInputDemo...

http://localhost:8700/theriabook/AdvancedDataGrid/NumericInputDemo.l

Description	Raw Number	Number
number	124000	$124,000.00
number	NaN	
null		
undefined		

Figure 6: NumericInputDemo application

```
public override  function set data(data:Object):void {
 super.data = data;
 var dgListData:DataGridListData =  DataGridListData(listData);
 text = data[dgListData.dataField];
}
```

The complete code for com.theriabook.controls.MaskedInput as well of all other samples from this article are available at http://samples.faratasystems.com/AdvancedDataGrid/index.html .

The second mask that we present in this section is NumericInput. It doesn't control the insertion point so typing and pasting is not restricted. However, it blocks any typing or pasting that contains invalid characters. Listing 10 shows the first iteration of NumericInput code.

As you can see in the listing, we listen to TextEvent.TEXT_INPUT, which corresponds to a character or a sequence of characters (paste) entered by the user. Whatever has been entered comes as an event.text and we test it with a regular expression, trying to find illegal characters. The string literal that we used for RegExp reads as "anything but characters in the range 0-9 or comma or dot or minus." If an illegal character is found, we reject the typing or pasting by cancelling the default behavior of the event with:

```
event.preventDefault();
```

That's all it takes to bullet proof your input fields from undesired characters. Two more small patches before we leave the NumericInput, though.

In the course of marshalling the Java data across the wire, chances are your numeric data will come as Number.NaN, a direct counterpart of Java Double.NaN or Float.NaN. In general, unless you use special DTOs with embedded null indicators; this is the natural way to marshal numeric nulls. No one would appreciate the lettering NaN staring at the user instead of the empty cell. Following the established pattern, we are going to make our NumericInput cognizant of the value, then have it produce the appropriate text as required for the presentation. Hence the following addition to NumericInput code:

```
private var _value:*;
[Bindable("change")]
public function set value(v:*):void {
 _value = v;
   if ((isNaN(v)) || (v==null /*null or undefined*/)) {
    text="";
   } else {
    text = String(v as Number);
   }
}

public function get value():* {
 // Preserve NaN | null | undefined, when nothing has been entered
   if (((_value!=null )&& (String(_value)!="NaN")) || (text!="") ) {
    _value = Number(text.replace(/,/g,"")); // deformat first
   }
   return _value;
}

  public override  function set data(item:Object):void {
 if (listData && listData is DataGridListData) {
    var dgListData:DataGridListData = DataGridListData(listData);
    value = item[dgListData.dataField];
  }
  super.data = item;
}
```

In the code fragment above we intervened in the setter for data property for when NumericInput is embedded in the DataGrid as a renderer. There we modify the value and let the value, in turn, modify the text.

In the value getter we return the original content: null, undefined, or Number.NaN, provided the user has not entered any-

ALL ATTENDEES RECEIVE: Certificate of Completion PLUS Course Materials on DVD!*

AJAX ONE-DAY HANDS-ON INTENSE TRAINING!

AJAXWorld University's **"AJAX Developer Bootcamp"** is an intensive, one-day hands-on training program that will teach Web developers and designers how to build high-quality AJAX applications from beginning to end. The AJAX Developer Bootcamp is intended to be the premier AJAX instructional program presently available anywhere.

AjaxWorld University BOOTCAMP

www.AJAXBOOTCAMP.sys-con.com

Roosevelt Hotel **New York, NY** **January 22, 2007**	Roosevelt Hotel **New York, NY** **March 18, 2007**

What You Will Learn...

Overview of AJAX Technologies
- HTML vs. DHTML
- Network Concerns
- Asynchronous Conversations with Web servers
- The characteristics of high-quality AJAX applications
  - The Web page is the application
  - What the server provides
  - User interaction

Understanding AJAX through the basics of AJAX
- Asynchronous server communication
- Dynamic HTML,
- Javascript Design patterns
- User interface strategies for building elegant, highly addictive Web sites and applications
- The Essential AJAX Pieces
  - Javascript
  - Cascading Style Sheet (CSS)
  - Document Object Model (DOM)
  - XMLHttpRequestObject
- The AJAX Application with Javascript
- Using CSS
- Structuring the View Using the DOM
  - Applying Styles with Javascript
  - Communicating with the Web Server in the Background
  - Designing AJAX Applications
  - Design Patterns
- Introduction to AJAX Frameworks-
  - Dojo, script.aculo.us, Prototype
  - Over of framework capabilities
- Examples of frameworks in use
- Best Practices

Hand-On Development The Fundamentals:
Building the Framing for an Ajax Application
- Review the courseware code with the Instructor
- Begin building a working AJAX application and start applying technique and technologies as introduced in class
- Create the basic AJAX application by creating HTML, Javascript, and CSS files
- Learn Best Practices and Validation
- Learn and add script.aculo.us effects
- Learn and add the Dojo Framework

Adding Basic Ajax Capabilities to a Web Page:
Going Deep Into the AJAX User Experience
- Elements on the Rich Internet Experience
  - Interactivity
  - Robustness
  - Simplicity
  - Recognizable Metaphors
  - Preservation of the Browser Model Bookmarks/Back Button
- Background operations
- Building a AJAX Notification Framework
- Provenance and Relevance
- Rich Experience Support with Third-Party AJAX Client – Framework
- Using AJAX layouts, containers, and widgets
- Patterns for Animation and Highlighting
- User Productivity Techniques
- Tracking Outstanding Network Requests

Hands-On Development:
Expand the Application with more Advanced Ajax
- Review the courseware code with the Instructor
- Expand the Mural Application
- Add Features using Dojo
- Add specific Dojo Libraries to support Ajax widgets
- Work on server side communications in the background
- Create a tabbed layout
- Create a submission form to upload to the server, all without reloading the page
- Create an Ajax submission form that will take uploads on one tab.
- Created a form validation that ensures only the right information is submitted.
- More Stretch work for those who want to learn additional concepts

Advanced AJAX Concepts
- Review Ajax Concepts
- SOA and Mashups
- Current state of Ajax Frameworks
- Web 2.0 and the Global SOA
- Ajax Constraints
- Design Patterns
- Javascript Timers
- Ajax Programming Patterns
- Performance and Throttling

Hands-On Development:
Working with Advanced Ajax Capabilities
- Review the courseware code with the Instructor
- Work with the Accordion control
- Learn how to use the Tree control
- Explore Dojo's animation capabilities
- Explore how the debug output can be used in <div> elements
- Tour Dojo and RICO demos
- Experiment with new Dojo features from the Dojo demos source code and attempt to add them to various parts of the Mural application
- Overview of Future of AJAX and Rich Internet Applications

*ALL RELEVANT COURSE MATERIALS WILL BE OFFERED ON DVD AFTER THE EVENT

HURRY! REGISTER NOW FOR EARLY-BIRD DISCOUNT!

What Attendees Are Saying...

"The trainer was excellent. The material too!

"The hands-on, although long, was useful and educational!

"The instructor was good. He answered questions thoroughly!

"Well designed and organized. Good mix of lecture vs lots of hands-on!

SYS-CON EVENTS For more great events visit www.EVENTS.SYS-CON.com thing. Otherwise, we use a regular expression to convert the text to a Number, globally eliminating the spaces and the thousand separators first:

```
      _value = Number(text.replace(/,/g,""));
```

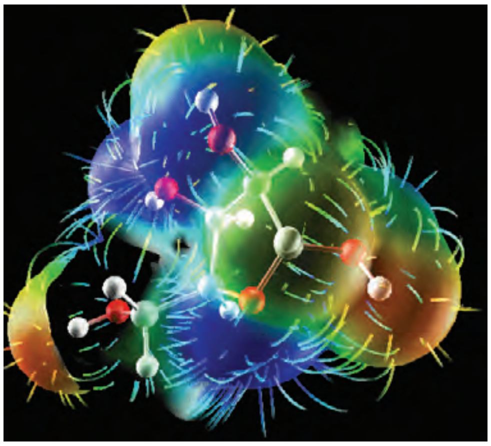
Strictly speaking, we should have operated with a locale-specific thousands separator character instead of ",". For reference, U.S. English-specific definitions of String constants decimalSeparator and thousandsSeparator (".” and ",") are located in the file Flex SDK/ 2/frameworks/locale/en_US/validator.properties. If you are delivering your application to Brazil, you could create a similar or smaller file in the Flex SDK/ 2/frameworks/locale/pr_BR folder, where you would redefine decimalSeparator as "," and thousandsSeparator as ".". Supposedly you would keep the original name – validators.properties. Then you would modify the compiler options for theriabook.swc to specify the pr_BR locale and rebuild NumericInput after adding the following code:

import mx.resources.ResourceBundle;

```
    import mx.resources.ResourceBundle;

 public class NumericInput extends TextInput {

  .   .   .   .

  [ResourceBundle("validators")]
  private static var rb:ResourceBundle;

  public static var decimalSeparator:String;
  public static var thousandsSeparator:String;

  // Load resources during class definition loading
  loadResources();

  private static function loadResources():void {
   decimalSeparator =  rb.getString("decimalSeparator");
   thousandsSeparator = rb.getString("thousandsSeparator");
  }
   }
```

Let's move on and present the testing application – NumericInputDemo. When you run it, try to type anything but digits, a comma and dot into the "Number" column; you won't be able to. Again, please make no mistake: NumericInput is a light-weight mask and it does not replace a need for validation. In particular, if you like regular expressions as much as we do, you may base the validation of the currency field on the mx.vaidators.RegExpValidator applying the regular expression which is the best match for your use case, for instance, something like:

```
^\$?([1-9]{1}[0-9]{0,2}(\,[0-9]{3})*(\.[0-9]{0,2})?|[1-9]{1}[0-
9]{0,}(\.[0-9]{0,2})?|0(\.[0-9]{0,2})?|(\.[0-9]{1,2})?)
```



Another feature to notice while running the demo app is how NaN, null, and undefined values are preserved in the absence of a meaningful input in the corresponding cells (see Figure 6).

The code of the testing application is shown in Listing 11, and the helper class NumberScope.as is presented in Listing 12.

Finally, Listing 13 depicts the complete code for NumericInput.

Summary

In this article we've continued our journey into not so obvious techniques of working with Adobe Flex DataGrid. In the final version of the article, we'll talk about data grids with dynamically computed item editors, data-driven programming and pivoted data grid.

About the Authors

Dr. Victor Rasputnis is a managing principal of Farata Systems. He's responsible for providing architectural design, implementation management and mentoring to companies migrating to XML Internet technologies. He holds a PhD in computer science from the Moscow Institute of Robotics.

Yakov Fain is a managing principal of Farata Systems. He's authored several Java books, dozens of technical articles. SYS-CON Books will be releasing his latest book, "Rich Internet Applications with Adobe Flex and Java: Secrets of the Masters" this Fall. Sun Microsystems has nominated and awarded Yakov with the title Java Champion. He leads the Princeton Java Users Group. Yakov teaches Java and Flex 2 at New York University. He is Adobe Certified Flex Instructor.

Anatole Tartakovsky is a managing principal of Farata Systems. He's responsible for creation of frameworks and reusable components. Anatole authored number of books and articles on AJAX, XML, Internet and client-server technologies. He holds an MS in mathematics.

vrasputnis@faratasystems.com

yfain@faratasystems.com

atartakovsky@faratasystems.com

www.frameworksconference.com

FRAMEWORKS 2007 CONFERENCE

Washington D.C.

February 1st & 2nd

NEW TOPICS GREAT SPEAKERS NETWORK LATEST TECHNOLOGIES

The Frameworks 2007 conference will be located in the Washington DC area, on February 2007. We will have lots of great speakers like last year. This year in addition to Fusebox and FLiP we will have sessions on other frameworks such as Mach-ii, Model Glue, Ruby on Rails, ColdSpring and Struts, and other methodologies such as XP and test-driven development.

We will be looking at frameworks for several web-based languages including ColdFusion, Java and .Net. There will be sessions for beginning and advanced developers, with lots of opportunities for learning from "foreign frameworks" and cross-pollination.

WHY SHOULD YOU COME TO THE FRAMEWORKS CONFERENCE?

“Frameworks is FOR developers, put on BY developers. So the information is useful and there is a ton of it... Are you into “Networking”, how would you like to talk to the people who created Fusebox, FLiP, or Fusedoc’s? Would you like to meet leaders in our industry using these tools? Well, you can, and that is why you should come."
Eric R. L.

"I'm coming to the conference because our core applications are far behind the Fusebox times–we're running on Fusebox version 2! So I'm interested in learning about other frameworks that are out there, as well as ways of migrating our existing apps smoothly."
Troy B.

"To stay current with the latest fusebox and frameworks conference."
Anne S.

TeraTech Programming
www.teratech.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