

ARF Usage Examples

Sample controls and XSL

Prepared By Vincent Rothwell



Table of Contents

Introduction to the samples	3
Debugging with ARF	3
Viewing the XML	3
Viewing debug statements	3
Creating an A to Z	4
Introduction	4
Sample control declaration	4
Sample XSL	4
AtoZ.xsl	4
AtoZ-Alternate.xsl	5
XSL selector control	6
Introduction	6
Sample control declaration	6
Sample XSL	6
Displaying a calendar	7
Introduction	7
Sample Control Declaration	7
Sample XSL	8
calendarTable.xsl	8
Query String Visibility Panel	10
Introduction	10
Sample control declaration	10
Sample XSL	10
Query String Filter	11
Introduction	11
Sample control declaration	11
Sample XSL	11
Creating a page with tabs	12
Introduction	12
Sample control declaration	12
Sample XSL	13
TabStrip.xsl	13

Introduction to the samples

Each sample in this document shows a way of using particular controls from ARF. It provides an introduction to the control, a sample definition you can use in your layouts and some XSL to get you started.

Once you have successfully followed the 'ARF Quick Start Guide' you will be in a position to try these samples. The samples will need to be added to a layout in your site and for some you may need to create additional columns. The XSL can be copied into a new XSL file, which should be placed in the XSL Library.

Debugging with ARF

Viewing the XML

Generally you will need to see the XML produced by the controls so that you can create the appropriate XSL. ARF provides you with two ways to achieve this.

Firstly you can just change the `Xsl-XslName` attribute to be '**xml.xml**'. This will use the XSL file installed by ARF to display the XML.

Secondly (and more easily) you can add a query string parameter to tell ARF to display the XML for all the ARF based controls on the page. Simply add **?arf-xml=true** to the URL and providing you have the appropriate permissions ARF will display the XML for you.

Viewing debug statements

ARF writes any problems and potentially useful information out to the debug console. By providing two downloads for release and debug you get more information from the debug version. It is recommended to use the debug version whilst in development and use the release version on your live site. If you have a problem on your live site you can always deploy the debug build to see more information.

You can view this output in two ways.

1. Turn on page trace in web.config and use trace.axd to see the statements. This has the added benefit of only showing the debug statements for your request. However it can miss some if they are not written to the page trace, particularly when they are before or after `BeginRequest()`.
2. A more useful approach is to use DebugView from TechNet. This will display all statements for the server on which it is run, but you do need access to the server.

Using these approaches it is generally easy to see what the problem is and how to solve it.

Creating an A to Z

Introduction

Creating an A to Z for a site is quite a common practice, but it's not always clear as to which links should appear on the page. Also using the title is not always appropriate as the begging letter is not always relevant to the page.

This is a simple solution allowing the page author to decide if it should appear within the A to Z and under which category or letter.

To use this you need to add two columns to your base page content type and add the field controls to the page. For the example below to work they should be called AtoZTitle1 and AtoZTitle2, thus providing the author with the ability to have the same page appear in two different places in the A to Z.

Sample control declaration

```
<ARF:SiteQuery runat="server" ServerTemplate="850" Xsl-XslName="AtoZ.xsl"
    Webs="<Webs Scope='Recursive'/">
    AdditionalViewFields="<FieldRef Name='AtoZTitle1'/"><FieldRef
Name='AtoZTitle2'/">
    Query="<Where><Or><IsNotNull><FieldRef
Name='AtoZTitle1'/"></IsNotNull><IsNotNull><FieldRef
Name='AtoZTitle2'/"></IsNotNull></Or></Where>" />
```

Sample XSL

AtoZ.xsl

```
<xsl:stylesheet xmlns:x=http://www.w3.org/2001/XMLSchema version="1.0"
xmlns:str="http://exslt.org/strings" xmlns:c="http://exslt.org/common"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
exclude-result-prefixes="str c x xsl">

    <xsl:output omit-xml-declaration="yes" />

    <xsl:key name="availableTitles" match="//rows/row/AtoZTitle1[. != ''] |
//rows/row/AtoZTitle2[. != '']" use="."/>

    <xsl:template match="/">
        <xsl:apply-templates select="rows/row/AtoZTitle1 | rows/row/AtoZTitle2">
            <xsl:sort select="." />
        </xsl:apply-templates>
    </xsl:template>

    <xsl:template match="AtoZTitle1 | AtoZTitle2">
        <xsl:if test="generate-id(..) = generate-id(key('availableTitles', ..)[1]/..)">
            <div class="atoz">
                <span>
                    <xsl:value-of select="."/>
                </span>
            </div>
        </xsl:if>
        <xsl:apply-templates select=".."/>
    </xsl:template>

    <xsl:template match="row">
        <a href="{substring-after(FileRef, '#')}" class="atoz">
```

```

        <xsl:value-of select="Title"/>
    </a>
    <br/>
</xsl:template>
</xsl:stylesheet>

```

AtoZ-Alternate.xsl

```

<xsl:stylesheet xmlns:x="http://www.w3.org/2001/XMLSchema" version="1.0"
xmlns:str="http://exslt.org/strings" xmlns:c="http://exslt.org/common"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
exclude-result-prefixes="str c x xsl">

    <xsl:output omit-xml-declaration="yes" />

    <xsl:variable name="alphabet">ABCDEFGHIJKLMNOPQRSTUVWXYZ</xsl:variable>

    <xsl:key name="matchedTitles" match="//rows/row/AtoZTitle1[. != ''] |
//rows/row/AtoZTitle2[. != '']" use="substring(., 1, 1)"/>

    <xsl:template match="/">
        <xsl:apply-templates select="rows"/>
    </xsl:template>

    <xsl:template match="rows">
        <xsl:call-template name="recurseAlphabet">
            <xsl:with-param name="currentPosition" select="number(1)"/>
        </xsl:call-template>
    </xsl:template>

    <xsl:template name="recurseAlphabet">
        <xsl:param name="currentPosition"/>
        <xsl:call-template name="letter">
            <xsl:with-param name="currentLetter">
                <xsl:value-of select="substring($alphabet,
$currentPosition, 1)"/>
            </xsl:with-param>
        </xsl:call-template>
        <xsl:if test="$currentPosition != 26">
            <xsl:call-template name="recurseAlphabet">
                <xsl:with-param name="currentPosition" select="$currentPosition + 1"/>
            </xsl:call-template>
        </xsl:if>
    </xsl:template>

    <xsl:template name="letter">
        <xsl:param name="currentLetter"/>
        <xsl:variable name="titles" select="key('matchedTitles',
$currentLetter)"/>

        <xsl:if test="count($titles) > 0">
            <h2 class="atoz">
                <xsl:value-of select="$currentLetter"/>
            </h2>
            <xsl:apply-templates select="$titles">
                <xsl:sort select="."/>
            </xsl:apply-templates>
        </xsl:if>
    </xsl:template>

```

```

<xsl:template match="AtoZTitle1 | AtoZTitle2">
  <li>
    <a href="{substring-after(../FileRef, '#')}">
      <xsl:value-of select=".."/>
      <br/>
    </a>
    <br/>
  </li>
</xsl:template>
</xsl:stylesheet>

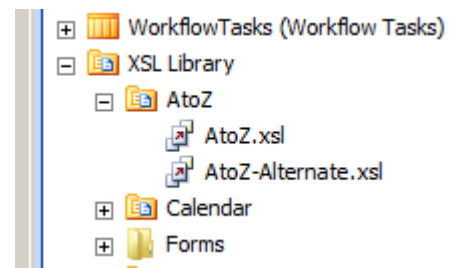
```

XSL selector control

Introduction

Following on from the A to Z example is the XSL selector example. This shows how you can allow the page author to choose which transformation to use for any of the ARF controls.

To do this you can create a folder in the XSL library which contains the XSL from which the author can select. You then place the `XslSelector` control on the page, give it an ID and specify that ID on your XML control using the `Xsl-XslSelectorID` property.



Sample control declaration

```

<ARF:XslSelector runat="server" ID="idAtoZSelector" XslPath="/Xsl Library/AtoZ/">
<ARF:SiteQuery runat="server" ServerTemplate="850" Xsl-XslName="AtoZ/AtoZ.xsl"
  Xsl-XslSelectorID="idAtoZSelector" Webs="<Webs Scope='Recursive'/">
  AdditionalViewFields="<FieldRef Name='AtoZTitle1' /><FieldRef
Name='AtoZTitle2' />"
  Query="<Where><Or><IsNotNull><FieldRef Name='AtoZTitle1'
/></IsNotNull><IsNotNull><FieldRef Name='AtoZTitle2' /></IsNotNull></Or></Where>"/>

```

Sample XSL

This example uses the XSL from the A to Z example.

Displaying a calendar

Introduction

ARF allows you to access the events stored in a calendar. It allows you to navigate from month to month and render the events using XSLT.

The XSLT also demonstrates some more advanced grouping techniques to enable the display of the days. This uses XSL keys to index the nodes in the XML, providing faster and more direct access to the relevant nodes.

Sample Control Declaration

```
<ARF:CalendarView runat="server" CalendarUrl="Calendar" ViewName="Calendar"  
WebUrl="/" Xsl-XslName="calendarTable.xsl" WeekStartsOn="Sunday"  
ShowWeekends="True"/>
```

Sample XSL

calendarTable.xsl

```

<xsl:stylesheet xmlns:x="http://www.w3.org/2001/XMLSchema" version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform" exclude-result-prefixes="xsl x">

  <xsl:output omit-xml-declaration="yes" method="xml" indent="yes"/>
  <xsl:key name="events" match="calendar/rows/row" use="concat(substring(EventDate, 1, 4),
substring(EventDate, 6, 2), substring(EventDate, 9, 2))"/>

  <xsl:key name="eventDays" match="calendar/rows/row/eventDays/day" use="@OrderedDate"/>

  <xsl:key name="weeks" match="calendar/month/day" use="@DayOfWeekNum"/>

  <xsl:param name="ShowWeekends"/>

  <xsl:template match="/">
    <div style="width:600px;text-align:center">
      <a>
        <xsl:attribute name="href"?m=<xsl:value-of select="calendar/@PrevMonth"
/>&amp;y=<xsl:value-of select="calendar/@PrevYear" /></xsl:attribute>&lt;&lt;
        </a>&#160;&#160;

        <xsl:value-of select="calendar/@MonthName" />&#160;<xsl:value-of
select="calendar/@Year"/>&#160;&#160;

      <a>
        <xsl:attribute name="href"?m=<xsl:value-of select="calendar/@NextMonth"
/>&amp;y=<xsl:value-of select="calendar/@NextYear" /></xsl:attribute>&gt;&gt;
      </a>
    </div>

    <table class="arfCalendar">
      <tr>
        <th><xsl:value-of select="key('weeks', 1) [1]/@DayOfWeek"/></th>
        <th><xsl:value-of select="key('weeks', 2) [1]/@DayOfWeek"/></th>
        <th><xsl:value-of select="key('weeks', 3) [1]/@DayOfWeek"/></th>
        <th><xsl:value-of select="key('weeks', 4) [1]/@DayOfWeek"/></th>
        <th><xsl:value-of select="key('weeks', 5) [1]/@DayOfWeek"/></th>
        <xsl:if test="$ShowWeekends = 'True'">
          <th><xsl:value-of select="key('weeks', 6) [1]/@DayOfWeek"/></th>
          <th><xsl:value-of select="key('weeks', 0) [1]/@DayOfWeek"/></th>
        </xsl:if>
      </tr>
      <tr valign="top">
        <td><xsl:apply-templates select="key('weeks', 1)"/></td>
        <td><xsl:apply-templates select="key('weeks', 2)"/></td>
        <td><xsl:apply-templates select="key('weeks', 3)"/></td>
        <td><xsl:apply-templates select="key('weeks', 4)"/></td>
        <td><xsl:apply-templates select="key('weeks', 5)"/></td>
        <xsl:if test="$ShowWeekends = 'True'">
          <td><xsl:apply-templates select="key('weeks', 6)"/></td>
          <td><xsl:apply-templates select="key('weeks', 0)"/></td>
        </xsl:if>
      </tr>
    </table>
  </xsl:template>

```

```

<xsl:template name="spacer">
  <xsl:param name="end"/>
  <xsl:param name="current"/>
  <xsl:if test="number($current) < number($end)">
    <div>&#160;</div>
    <xsl:call-template name="spacer">
      <xsl:with-param name="end" select="$end"/>
      <xsl:with-param name="current" select="$current + 1"/>
    </xsl:call-template>
  </xsl:if>
</xsl:template>

<xsl:template match="row/eventDays/day">
  <xsl:if test="(position()-1) < number(..../@Index)">
    <xsl:call-template name="spacer">
      <xsl:with-param name="end" select="number(..../@Index)"/>
      <xsl:with-param name="current" select="0"/>
    </xsl:call-template>
  </xsl:if>
  <div>
    <xsl:value-of select="..../Title"/><br/>
  </div>
</xsl:template>

<xsl:template match="calendar/month/day">
  <div class="arfDayHeader">
    <xsl:if test="/calendar/@Month != @Month">
      <xsl:attribute name="class">
        arfDayHeader arfDayHeaderOther
      </xsl:attribute>
    </xsl:if>
    <xsl:value-of select="@Day"/>
  </div>
  <div class="arfDayDetails">
    <xsl:apply-templates select="key('eventDays', @OrderedDate)"/>
    <xsl:if test="count(key('eventDays', @OrderedDate)) = 0">
      &#160;
    </xsl:if>
  </div>
</xsl:template>
</xsl:stylesheet>

```

Query String Visibility Panel

Introduction

The visibility panel allows you to show or hide page content based upon a query string.

Sample control declaration

```
<ARF:QSVisibilityPanel runat="server" QueryStringParameter="Show" Visible="False"
ShowWhenParameterPresent="True">
```

```
<p>This content is only visible when the query string 'Show' is present.</p>
```

```
</ARF:QSVisibilityPanel>
```

```
<ARF:QSVisibilityPanel runat="server" QueryStringParameter="Show" Visible="True"
ShowWhenParameterPresent="False">
```

```
<p>This content is only visible when the query string 'Show' is NOT present.</p>
```

```
</ARF:QSVisibilityPanel>
```

```
<ARF:QSVisibilityPanel runat="server" QueryStringParameter="Show" Visible="false"
QueryStringValue="True">
```

```
<p>This content is only visible when the query string 'Show' is equal to
"True".</p>
```

```
</ARF:QSVisibilityPanel>
```

Sample XSL

The visibility panel does not use XSL.

Query String Filter

Introduction

The Query String Filter control allows you to filter a Site Query based upon a query string in the URL. The example below shows the Site Query using a filter to show pages based upon the content type specified in the query string.

Sample control declaration

```
<ARF:QueryStringFilter runat="server" id="idQueryFilter" FilterReplaceString="[QS]"  
IncludeWhereClause="True" QueryStringParameters="ContentType"/>
```

```
<ARF:SiteQuery runat="server" ServerTemplate="850" Query="[QS]"  
QueryFilter="idQueryFilter" Xsl-XslName="default.xsl" Webs="<Webs  
Scope='SiteCollection' />" />
```

Sample XSL

The query string filter does not use XSL.

Creating a page with tabs

Introduction

This sample shows using the `QSVisibilityPanel` to create a tabbed display. The tabs are created by the initial `XmlProvider` control, which is just creating the links to switch tabs. One thing to note here is the `Xsl-XslParameters` property, which is passing a custom parameter to the XSL transformation.

Each tab is made visible dependant on the value of a query string parameter, in this example it is 'tab'. Each tab is displayed when the query string parameter is equal to the value specified by the `ParameterValue` property.

Each tab contains a `RichHtmlField` which allow the page author to edit the content of each tab. The `QSVisibilityPanel` will ensure all tabs are visible when the page is being edited.

Sample control declaration

```
<arf:XmlProvider runat="server" XmlString="<Test/>" Xsl-
XslName="tabs/TabStrip1.xsl" Xsl-XslParameters="qsName=tab"/>

<div class="arfTabContainer" style="height:150px">

    <ARF:QSVisibilityPanel runat="server" ID="tab1"
        QueryStringParameter="tab" ParameterValue="1"
        DefaultPanel="True">

        <PublishingWebControls:RichHtmlField FieldName="Html1" runat="server"
            InputFieldLabel="Tab One Content" />

    </ARF:QSVisibilityPanel>

    <ARF:QSVisibilityPanel runat="server" ID="tab2"
        QueryStringParameter="tab" ParameterValue="2">

        <PublishingWebControls:RichHtmlField FieldName="Html2" runat="server"
            InputFieldLabel="Tab Two Content" />

    </ARF:QSVisibilityPanel>

    <ARF:QSVisibilityPanel runat="server" ID="tab3"
        QueryStringParameter="tab" ParameterValue="3">

        <PublishingWebControls:RichHtmlField FieldName="Html3" runat="server"
            InputFieldLabel="Tab Three Content" />

    </ARF:QSVisibilityPanel>

    <ARF:QSVisibilityPanel runat="server" ID="tab4"
        QueryStringParameter="tab" ParameterValue="4">

        <PublishingWebControls:RichHtmlField FieldName="Html4" runat="server"
            InputFieldLabel="Tab Four Content" />

    </ARF:QSVisibilityPanel>

</div>
```

Sample XSL

TabStrip.xsl

```
<xsl:stylesheet xmlns:x="http://www.w3.org/2001/XMLSchema" version="1.0"
xmlns:arf="http://blog.thekid.me.uk/arf"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform" exclude-result-prefixes="xsl x
arf">
  <xsl:output omit-xml-declaration="yes" method="xml" indent="yes"/>
  <xsl:param name="qs-tab"/>
  <xsl:param name="qsName"/>
  <xsl:template match="/">
    <div class="arfTabs">
      <ul>
        <xsl:call-template name="tab"><xsl:with-param
name="number" select="'1'"/><xsl:with-param name="text"
select="'About'"/></xsl:call-template>
        <xsl:call-template name="tab"><xsl:with-param
name="number" select="'2'"/><xsl:with-param name="text"
select="'Uses'"/></xsl:call-template>
        <xsl:call-template name="tab"><xsl:with-param
name="number" select="'3'"/><xsl:with-param name="text"
select="'Parameters'"/></xsl:call-template>
        <xsl:call-template name="tab"><xsl:with-param
name="number" select="'4'"/><xsl:with-param name="text" select="'The
HTML'"/><xsl:with-param name="class" select="'end'"/></xsl:call-template>
      </ul>
    </div>
  </xsl:template>
  <xsl:template name="tab">
    <xsl:param name="number"/>
    <xsl:param name="text"/>
    <xsl:param name="class"/>
    <li class="{ $class }">
      <xsl:if test="$qs-tab=$number"><xsl:attribute
name="class">selected <xsl:value-of select="$class"/></xsl:attribute></xsl:if>
      <a>
        <xsl:attribute name="href"><xsl:value-of
select="arf:BuildQueryString(concat($qsName, '=', $number),
$qsName)"/></xsl:attribute>
        <xsl:value-of select="$text"/>
      </a>
    </li>
  </xsl:template>
</xsl:stylesheet>
```