XML Data Islands and Persistence in ASP.NET*

By Rahul Guha

The Challenge

Sometimes it's advantageous to separate data representation from the data itself, creating "data islands." For example, you can separate the user interface (display elements) in a XSLT file and transform the XML file (stream) through that XSLT file, which builds the HTML that is understood by browsers. A single page can access multiple data islands (such as an XML file, database source, and so forth) that each require a different XSLT file. The beauty of this design is that if you need to change any aspect of the data representation, you simply change a specific XSLT without touching the logic or main code block. This is a clean and easy way to separate presentation from data handling.

Once you have data represented in the page from multiple sources of data though separate XSLTs, you may need to persist the XML data blocks (islands), make changes to the data, and save the changes in one process when the user submits the page.

This article illustrates using persistent data islands.

The Solution

The Microsoft .NET* Framework includes the **XML Control** that specifically supports data representation. Use **DOMDocument** (Document Object Model) to persist the data islands in the client and make changes. Submit the updated XML stream to the server where it is saved to the database.

XML Control

XML Control is a part of the **System.Web.UI.Control** namespace. These are the members used in this solution described in this article:

- **DocumentContent:** Feeds the XML Control with a well-formed XML string
- **Document:** Feeds the XML Control with a DOMDocument (loaded from XML string / file / stream).
- **DocumentSource:** Feeds the XML Control from a source path
- **TransformSource:** Path for the XSLT file that will be used for transforming the XML doc to HTML.

Architecture Overview

This example uses one **Container page** that includes two XML Web server controls. The controls are attached to separate XML strings, which are created from datasets filled from two separate database calls. The database calls can be to any type of data source.

The XML streams are persisted in the client page in the form of hidden variables so that they can be accessed from the client-side code.

Two separate XSLT files are attached to the XML controls, which are independent of each other and can incorporate any user interface elements.

The client-side code (Internet Explorer*-specific) does the following:

1. Load the XML data island in DOMDocument (using the Microsoft XML parser)
2. Handle the events for various client elements
3. Update the DOM
4. Save the DOM back into the hidden variable
5. Submit the page, which sends the hidden form variables to the server where they are accessed and used to update the database in a batch process.

Container Page

The Container page includes two XML controls, one Submit button, and two hidden variables. The XML controls are fed through XML strings created using the **GetXml()** method of the dataset that is created while the container page is loaded.

The XSLT files render the two XML controls to display the list of data elements and data along with checkboxes against each of the data elements. An event handler is attached (in the XSLT) to the checkbox click event, which updates a specific node in the XML string with data. It also updates an element called **Dirty** to 1, which indicates that the element has been changed. When the page is submitted, the system can process only those elements that are changed instead of making a database call for every element.

The Code

Container.aspx

```
<%@ Page language="c#" Codebehind="container.aspx.cs" AutoEventWireup="false"
Inherits="XmlDemo.container" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" >
<HTML>
	<HEAD>
		<title>XML Demo</title>
		<meta content="Microsoft Visual Studio 7.0" name="GENERATOR">
		<meta content="C#" name="CODE_LANGUAGE">
		<meta content="JavaScript" name="vs_defaultClientScript">
		<meta content="http://schemas.microsoft.com/Intellisense/ie5"
name="vs_targetSchema">
		<LINK href="style.css" type="text/css" rel="stylesheet">


<!-- ************************************************************************
THIS BLOCK OF CLIENT-SIDE CODE IS USED FOR LOADING THE XML DATA ISLAND IN THE DOM, AS WELL
AS UPDATING IT AND SAVING itBACK TO THE HIDDEN VARIABLES.
************************************************************************ -->

<script language="javascript" id="clientEventHandlersJS">
<!--

var xmlDoc = new ActiveXObject("Microsoft.XMLDOM");
var xmlDocTopic = new ActiveXObject("Microsoft.XMLDOM");

function HandleEvent (ID)
{
	if (window.event.srcElement.checked)
	{
		updateXML(ID,"1");
	}
	else
	{
		updateXML(ID,"0");
	}
}

function updateXML(ID, SubValue)
{
	xmlDoc.loadXML (document.all["xmlData"].value);
	var lMatch;
	lMatch = "//Table[ID=" + String.fromCharCode(34) +  ID +String.fromCharCode(34) +"]";
	if (xmlDoc.selectNodes(lMatch).length > 0)
	{
		curNode = xmlDoc.selectSingleNode (lMatch);
		lMatch ="NodeData"
		curNode.selectSingleNode(lMatch).text = SubValue ;
		lMatch ="Dirty"
		curNode.selectSingleNode(lMatch).text = "1";
	}
	document.all["xmlData"].value = xmlDoc.xml;
```

```
}

function HandleEventforTopic (ID)
{
	if (window.event.srcElement.checked)
	{
		updateXMLforTopic(TopicID,"1");

	}
	else
	{
		updateXMLforTopic(TopicID,"0");
	}
}



function updateXMLforTopic(ID, SubValue)
{
	xmlDocTopic.loadXML (document.all["xmlTopicData"].value);
	var lMatch;
	lMatch = "//Table[ID=" + String.fromCharCode(34) +  ID +String.fromCharCode(34) +"]";
	if (xmlDocTopic.selectNodes(lMatch).length > 0)
	{
		curNode = xmlDocTopic.selectSingleNode (lMatch);
		lMatch ="NodeData"
		curNode.selectSingleNode(lMatch).text = SubValue ;
		lMatch ="Dirty"
		curNode.selectSingleNode(lMatch).text = "1";
	}
	document.all["xmlTopicData"].value = xmlDocTopic.xml;
}

//-->
		</script>
	</HEAD>
<body >
<form id="container" method="post" runat="server">

<table width="790" border="0" cellspacing="0" cellpadding="0">
<tr>
	<td width="164" valign="top">
	</td>
	<td width="626" valign="top">
	<div class="DocumentBody">
	<table class="TableStyleBorder">
<TR>
<TD align="right">
<asp:button id="btnSave" runat="server" Width="62px" CssClass="stButton"
Text="Save"></asp:button></TD>
</TR>
<tr>
<td align="left">
<div id="oDiv" style="OVERFLOW: auto; WIDTH: 100%; " runat="server">



<asp:xml id="xmlTopic" runat="server" TransformSource="xsl1.xslt"></asp:xml>
<asp:xml id="xmlSub" runat="server" TransformSource="xsl2.xsl"></asp:xml>
</div>
</td>
</tr>
</table>

<INPUT id="xmlData" type="hidden" name="xmlData" runat="server">
<INPUT id="xmlTopicData" type="hidden" name="xmlTopicData" runat="server">
</FORM>

</body>
</HTML>
```

Container.aspx.cs

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;
using System.Xml;
namespace XmlDemo
{
	public class container: System.Web.UI.Page
	{
		protected System.Web.UI.HtmlControls.HtmlGenericControl oDiv;
		protected System.Web.UI.WebControls.Button btnSave;
		protected System.Web.UI.WebControls.Xml xmlSub;
		protected System.Web.UI.HtmlControls.HtmlInputHidden xmlData;
		protected System.Web.UI.WebControls.Xml xmlTopic;
		protected System.Web.UI.HtmlControls.HtmlInputHidden xmlTopicData;

		private void Page_Load(object sender, System.EventArgs e)
		{
			Initialize();
			LoadXML();
		}

		private void Initialize()
		{
			// put initialization code here ... read querystring etc.
		}
		private void LoadXML()
		{
			try
			{
				DataSet ds = new DataSet ();
				DataSet dsTopics = new DataSet ();

				// put your data access code here
				// Populate the datasets
				ds = cls.GetSubscriptionbyUser (UserID);
				dsTopics = cls.GetTopics (UserID);
				// feed the xml controls
				xmlTopic.DocumentContent = dsTopics.GetXml ();
				xmlSub.DocumentContent = ds.GetXml ();

				// persist the xml data in client
				xmlData.Value = ds.GetXml ();
				xmlTopicData.Value = dsTopics.GetXml ();

				// clean it up
				ds = null;
				dsTopics = null;
				cls = null;
			}
			catch (Exception e)
			{
				Response.Write (e.ToString());
			}

		}
		#region Web Form Designer generated code
		override protected void OnInit(EventArgs e)
		{
			InitializeComponent();
			base.OnInit(e);
		}

		private void InitializeComponent()
		{
		this.btnSave.Click += new System.EventHandler(this.btnSave_Click);
```

```
                    this.Load += new System.EventHandler(this.Page_Load);

                }
                #endregion

                private void btnSave_Click(object sender, System.EventArgs e)
                {

                // Load the edited xml string into DomDocuments
                string xmlSTR = "<root>" + Request.Form ["xmlData"].ToString() + "</root>";
                XmlDocument xmlDoc = new XmlDocument();
                xmlDoc.LoadXml (xmlSTR);
                xmlSTR = "<root>" + Request.Form ["xmlTopicData"].ToString() + "</root>";
                XmlDocument xmlDocTopic = new XmlDocument();
                xmlDocTopic.LoadXml (xmlSTR);

// ********************************************************************
INPUT YOUR CODE THAT WILL READ EACH OF THE DATA ISLANDS AND WRITE THEM BACK TO THE DATABASE.
*********************************************************************//


                cls.UpdateSubscription (xmlDoc);
                cls.UpdateTopicSubscription (xmlDocTopic);
                XmlDocument doc = new XmlDocument();

                // Refresh the page with new data
                doc = xmlSub.Document;
                xmlData.Value = doc.OuterXml;
                doc = xmlTopic.Document ;
                xmlTopicData.Value = doc.OuterXml ;
                LoadXML();
                }
        }
}
```

Function to Update the Database from the Submitted XML Data

```
public bool UpdateSubscription(string idsid, XmlDocument xmldoc)
{
        string lMatch;
        string strNodeData;
        bool Err;
        Err = false;
        lMatch ="//Table";
        foreach (XmlNode nd in xmldoc.SelectNodes(lMatch))
        {
                lMatch = "Dirty";
        if (nd.SelectSingleNode (lMatch).InnerText.ToString()   == "1")
        {
        lMatch = "ID";
        string strID = nd.SelectSingleNode (lMatch).InnerText .ToString();
        // extract data for this id
        lMatch = "MyNode";
        strNodeData = nd.SelectSingleNode (lMatch).InnerText .ToString();
        // make database call
        DataSet ds = new DataSet();
        ds = WriteData (strID,strNodeData);

        // handle error
        if (ds.Tables[0].Rows[0][0].ToString () != "ERROR")
        {
                ds = null;
                Err=false;
        }
        else
        {
                ds = null;
                Err=true;
        }

        }
        else
```

```
        {
                Err = false;
        }
        }
        if (!Err)
        {
                return true;
        }
        else
        {
                return false;
        }


        }
```

XML for Hidden Variable XMLData (rendered in XML control xmlsub)

```
<Data>
  <Table>
    <ID>35</ThreadID>
    <TopicID>0</TopicID>
    <Title>.Net Remoting!</ThreadTitle>
    <Topic>.Net Framework</TopicTitle>
    <Subscribed>1</Subscribed>
    <Dirty>0</Dirty>
  </Table>
  <Table>
    <ID>65</ThreadID>
    <TopicID>0</TopicID>
    <Title>COM Interop</ThreadTitle>
    <Topic>.Net Framework</TopicTitle>
    <Subscribed>1</Subscribed>
    <Dirty>0</Dirty>
  </Table>
  <Table>
    <ID>62</ThreadID>
    <TopicID>0</TopicID>
    <Title>Microsoft.ApplicationBlocks.Data </ThreadTitle>
    <Topic>.Net Framework</TopicTitle>
    <Subscribed>1</Subscribed>
    <Dirty>0</Dirty>
  </Table>
  <Table>
    <ID>66</ThreadID>
    <TopicID>0</TopicID>
    <Title>c# vs vb.Net</ThreadTitle>
    <Topic>vb.Net</TopicTitle>
    <Subscribed>1</Subscribed>
    <Dirty>0</Dirty>
  </Table>
  <Table>
    <ID>1</ThreadID>
    <TopicID>0</TopicID>
    <Title>Jobs in Moon</ThreadTitle>
    <Topic>.Net Jobs</TopicTitle>
    <Subscribed>2</Subscribed>
    <Dirty>0</Dirty>
  </Table>
  <Table>
    <ID>92</ThreadID>
    <TopicID>14</TopicID>
    <Title>XLM Data Island</ThreadTitle>
    <Topic>XML </TopicTitle>
    <Subscribed>0</Subscribed>
    <Dirty>0</Dirty>
  </Table>
  <Table>
    <ID>93</ThreadID>
    <TopicID>14</TopicID>
    <Title>Schema</ThreadTitle>
```

```
    <Topic>XML </TopicTitle>
    <Subscribed>0</Subscribed>
    <Dirty>0</Dirty>
  </Table>
</Data>
```

XML for Hidden Variable xmlTopic (rendered in xmlTopic)

```
<Data>
  <Table>
    <ID>0</TopicID>
    <Topic>.Net CLR</TopicTitle>
    <Subscribed>1</Subscribed>
    <Dirty>0</Dirty>
  </Table>
  <Table>
    <ID>1</TopicID>
    <Topic>c#</TopicTitle>
    <Subscribed>1</Subscribed>
    <Dirty>0</Dirty>
  </Table>
  <Table>
    <ID>2</TopicID>
    <Topic>ASP.Net</TopicTitle>
    <Subscribed>1</Subscribed>
    <Dirty>0</Dirty>
  </Table>
  <Table>
    <ID>3</TopicID>
    <Topic>VB.Net</TopicTitle>
    <Subscribed>1</Subscribed>
    <Dirty>0</Dirty>
  </Table>
  <Table>
    <ID>4</TopicID>
    <Topic>SOAP Web Services</TopicTitle>
    <Subscribed>1</Subscribed>
    <Dirty>0</Dirty>
  </Table>
  <Table>
    <ID>5</TopicID>
    <Topic>Data access techniques in .Net</TopicTitle>
    <Subscribed>0</Subscribed>
    <Dirty>0</Dirty>
  </Table>
  <Table>
    <ID>13</TopicID>
    <Topic>Training and Resources</TopicTitle>
    <Subscribed>0</Subscribed>
    <Dirty>0</Dirty>
  </Table>
  <Table>
    <ID>14</TopicID>
    <Topic>XML </TopicTitle>
    <Subscribed>0</Subscribed>
    <Dirty>0</Dirty>
  </Table>
</Data>
```

xsl1.XSLT (rendered with control xmlTopic)

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<table class="TableStyleBorder" cellpadding="5" cellspacing="0" width="500px">
<thead class="TableHeadStyle">
```

```
<tr>
<td>
	Posting Subject
</td>
<td>
	Subscribed ?
</td>
</tr>
</thead>

<tbody>

<xsl:for-each select="//Table">

<xsl:choose >
<xsl:when test="position() mod 2> 0">
	<tr bgcolor="lightblue">
	<td wrap="true">
		<span class="stLabel">
			<xsl:value-of select="Topic"/>
		</span>
	</td>
		<td align='center' >
				<input type="checkbox" language="javascript" >
				<xsl:attribute name="onclick">HandleEventforTopic
'<xsl:value-of select="ID"  />' </xsl:attribute>
					<xsl:choose>
					<xsl:when test= "Subscribed = 1">
						<xsl:attribute name="checked"></xsl:attribute>
					</xsl:when>
					</xsl:choose>
				</input>
		</td>
	</tr>
</xsl:when>
<xsl:otherwise>
	<tr>
	<td wrap="true">
		<span class="stLabel">
			<xsl:value-of select="Topic"/>
		</span>
	</td>
	<td align='center'>
			<input type="checkbox" language="javascript" >
			<xsl:attribute name="onclick">HandleEventforTopic  '<xsl:value-of
select="ID"  />' </xsl:attribute>
				<xsl:choose>
				<xsl:when test= "Subscribed = 1">
					<xsl:attribute name="checked"></xsl:attribute>
				</xsl:when>
				</xsl:choose>
			</input>
	</td>
	</tr>
</xsl:otherwise>
</xsl:choose>

</xsl:for-each>
</tbody>
</table>
</xsl:template>
  </xsl:stylesheet>
```

xsl2.XSLT (rendered with control xmlsub)

```
<?xml version="1.0" ?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">

<table class="TableStyleBorder" cellpadding="5" cellspacing = "0" width="500px">
<thead class="TableHeadStyle">
<tr>
<td>Topic</td>
<td>Posting Title</td>
<td>Subscribed ?</td>
</tr>
</thead>

<tbody>

<xsl:for-each select="//Table">
<xsl:choose >
<xsl:when test="position() mod 2> 0">

	<tr bgcolor="lightblue">
	<td wrap="true">
			<span class="stLabel">
				<xsl:value-of select="Topic"/>
			</span>
	</td>

	<td wrap="true">
			<span class="stLabel">
				<xsl:value-of select="Title"/>
			</span>
	</td>

	<td align='center'>
	<input type="checkbox" language="javascript" >
		<xsl:attribute name="onclick">HandleEvent '<xsl:value-of select="ID"  />'
	</xsl:attribute>
			<xsl:choose>
			<xsl:when test= "Subscribed = 1">
				<xsl:attribute name="checked"></xsl:attribute>
			</xsl:when>
			</xsl:choose>
	</input>
	</td>
	</tr>
</xsl:when>
<xsl:otherwise >
	<tr >
	<td wrap="true">
			<span class="stLabel">
				<xsl:value-of select="Topic"/>
			</span>
	</td>

	<td wrap="true">
			<span class="stLabel">
				<xsl:value-of select="Title"/>
			</span>
	</td>

	<td align='center'>
```

```
            <input type="checkbox" language="javascript" >
            <xsl:attribute name="onclick">HandleEvent '<xsl:value-of select="ID"  />'
</xsl:attribute>
            <xsl:choose>
            <xsl:when test= "Subscribed = 1">
                  <xsl:attribute name="checked"></xsl:attribute>
            </xsl:when>
            </xsl:choose>
      </input>
      </td>
      </tr>

</xsl:otherwise>
</xsl:choose>
</xsl:for-each>
</tbody>
</table>
</xsl:template>
</xsl:stylesheet>
```

The Advantages

This solution offers the following advantages:

- Separates the data from the presentation code, which provides flexibility.
- Supports multiple data sources integrated into one page using the same or different presentation code.
- Saves trips to the server every time the user makes a change. Data access is handled in one shot when the page is submitted.
- The server processes only changed data.

About the Author

Rahul Guha is a senior Software Engineer in the Intel® e-Business Architecture Group, where he is responsible for architecting, designing, and developing Web-based applications. He is also a part of the .NET Taskforce team that creates application development guidelines for .NET in the Intel e-Business Group. He has more than nine years of experience in the computing industry, most of which comes from consulting with clients from Fortune 500 to startups. He joined Intel in February 1999.