

# Delphi and AJAX

by Marco Cantù

Asynchronous JavaScript XML is not really a brand new web technology, but its usage is rapidly changing the way users perceive web applications. If the browser is becoming a platform, can Delphi fit into this new model? The flexibility of WebBroker and the strengths of the database connectivity make Delphi a very good choice for an AJAX-based server.

## What is AJAX?

Until a few months ago, I used to associate the term AJAX with one of the most famous Dutch football teams. Over the last few months, however, I've kept seeing more and more astonishing web sites, based on standard technologies (HTML, CSS, JavaScript), mostly browser independent, featuring rich client applications! If you want to see an overall description of AJAX and the underlying technologies, refer to the key references at <http://www.adaptivepath.com/publications/essays/archives/000385.php>.

The AJAX term was invented in February 2006, but certainly it was Google adoption of the technology that has been critical to its spreading. Google is not the only company using it, though. There are countless sites featuring it, including the short list of AJAX-based sites below. Be sure to visit these sites to get a feeling of the power of this technology:

Google Suggests (<http://www.google.com/webhp?complete=1&hl=en>)

Gmail (<http://gmail.google.com/>, feel free to email me on [marco.cantu@gmail.com](mailto:marco.cantu@gmail.com) if you want an invitation)

Google Maps (<http://maps.google.com/>)

Protopage (<http://www.protopage.com/>)

Press-display (<http://www.pressdisplay.com/pressdisplay/viewer.aspx>)

Ta-da Lists (<http://www.tadalist.com/>)

Microsoft's experimental AJAX site at start.com (<http://www.start.com/>). Notice in particular how you can drag and drop the portions of the page to rearrange them.

## AJAX in Delphi

Needless to say I've started playing with AJAX myself. I've found a very rich engine for query processing (also very easy to use) at Ajax Toolbox (<http://www.ajaxtoolbox.com/>) and I've done some experiments with my newsgroup front-end (beta) web site at <http://dev.newswhat.com>. If you go to a thread you can open individual messages, fetching them one by one without refreshing the page. A similar example is in my blog site, if you follow the link to the Comments List ([http://blog.marcocantu.com/blog\\_alltalkback.html](http://blog.marcocantu.com/blog_alltalkback.html)) in the main menu. You'll see a list of talkback messages and can open each of them directly in the page. These examples use AJAX on top of Delphi-driven applications but the architecture of these applications is too complex to describe here.

This is why I've also built a simple Delphi AJAX demo (for a Birds-of-a-Feather session I did at BorCon 2005) that I'm going to discuss in detail here. For the development of this project I've built a Delphi CGI WebBroker application (to keep things simple) and used two different JavaScript libraries: [www.ajaxtoolbox.com](http://www.ajaxtoolbox.com) for the AJAX communication and Google's own JavaScript XML, XPath, and XSLT engine at [sourceforge.net/projects/goog-ajaxslt](http://sourceforge.net/projects/goog-ajaxslt) for XML processing.

The goal of this demo is to display database data dynamically and combine master-detail data over web pages without sending too much data to the client upfront.

## The Web Page

The demo has only one web page, used to display different types of data. First of all, it includes a set of JavaScript files:

---

```
<script language="JavaScript1.2"
  src="/include/AjaxMdDemo.js">&#160;</script>
<script language="JavaScript1.2"
  src="/include/AjaxRequest.js">&#160;</script>
```

```
<script src="/include/misc.js" type="text/javascript"/>
<script src="/include/dom.js" type="text/javascript"/>
<script src="/include/xpath.js" type="text/javascript"/>

<script src="/include/xslt.js" type="text/javascript"/>
```

---

The first is a custom file I've written, everything else comes from the libraries mentioned above. The web page has a menu with the AJAX-based commands exposed by the custom JavaScript files. This is the menu:

---

```
<div class="navigation-item">
  <a href="javascript:AjaxMdAllHtml()">
    All Customers (HTML)
  </a>
</div>
<div class="navigation-item">
  <a href="javascript:AjaxMdAllXml()">
    All Customers (XML)
  </a>
</div>
<div class="navigation-item">
  <a href="javascript:AjaxMdCustList()">
    Customers List (XML, M/D)
  </a>
</div>
```

---

Finally, the page has a *div* section that is going to host the dynamic content. At the beginning (in the default.html file) it looks like this:

---

```
<div id="content-main">
  <h2>Home</h2>
  <p>Home page content goes here</p>
</div>
```

---

## The AJAX call

What happens when a user clicks on one of the menu items? The corresponding JavaScript function gets executed. This is the first one:

---

```
function AjaxMdAllHtml()
{
  AjaxRequest.get({
    'url': '/code/ajaxmasterdetail.exe/htmlall',
    'onSuccess': function(req) {
      document.getElementById('content-main').innerHTML =
        '<h1> Data </h1>' + req.responseText;
    }
  })
}
```

---

We use an `AjaxRequest` data structure with two settings: the URL we want to call and the function to execute if the data is successfully retrieved. The effect is to copy the result of the HTTP request (`req.responseText`) into the main *div* block. The call is sent to the CGI application, written in Delphi with the WebBroker framework. This application has an *htmlall* action associated with a `DataSetTableProduced` mapped to a `ClientDataSet` component hosting some data:

---

```
Actions = <
  item
    Name = 'wmHtmlAll'
    PathInfo = '/htmlall'
    Producer = DataSetTableProducer1
  end
```

```
object DataSetTableProducer1: TDataSetTableProducer
  Columns = <...>
  DataSet = cdsCustomer
end
object cdsCustomer: TClientDataSet
  FileName = '..\Data\customer.xml'
end
```

---

As we return an HTML snippet, we can display it right away.

## Returning XML

To make things a little more interesting, we can have the application return XML for browser-based processing. In this case we have to write some code for the Delphi application action:

---

```
procedure TWebModule4.WebModule4waXmlAllAction(
  Sender: TObject; Request: TWebRequest;
  Response: TWebResponse; var Handled: Boolean);
begin
  cdsCustomer.Open;
  Response.Content := cdsCustomer.XMLData;
end;
```

---

As this code is very simple you might think the JavaScript needed to fetch and process it becomes very complex, but this is not true, thanks to the Google libraries. What I've done is to add a textarea (that would be hidden in a real application) to the web page with the XSLT code used to transform the XML data. This is how you can invoke the entire process:

---

```
function AjaxMdAllXml()
{
  AjaxRequest.get({
    'url': '/code/ajaxmasterdetail.exe/xmlall',
    'onSuccess':function(req) {
      var xml = xmlParse(req.responseText);
      var xslt = xmlParse(el('xslt').value);
      var html = xsltProcess(xml, xslt);
      el('content-main').innerHTML = html;
    }
  })
}
```

---

The code loads the response in an XML parser, the XSLT from the textarea in another one, does the XSLT transformation (`xsltProcess`) and displays the result. Here I've omitted the details of the XSLT code that you can find inside the *xslt* textarea of the default.html file in the source code.

## More XML Processing

The net effect of this XSLT-based processing is not different from the previous one. However, in the first case any change in the resulting HTML requires you to recompile the Delphi application, while in the second case you can completely change or fine tune the user interface by modifying the XSLT information on the fly.

However, if you go towards the use of XML you can do more. The following JavaScript function takes a slightly different approach: it fetches the same XML data, moves it to a local cache (another textarea control, called *xml\_data*), and processes it with a different XSLT (*xslt\_list*) to display only a list of companies:

---

```
function AjaxMdCustList()
{
  AjaxRequest.get({
    'url': '/code/ajaxmasterdetail.exe/xmlall',
    'onSuccess':function(req) {
      el('xml_data').value = req.responseText;
      var xml = xmlParse(req.responseText);
      var xslt = xmlParse(el('xslt_list').value);
      var html = xsltProcess(xml, xslt);
      el('content-main').innerHTML = html;
    }
  })
}
```

---

Now the XSLT generates a list of company names linked to a further action:

```
<div>
  <a>
    <xsl:attribute name="href">
      javascript:AjaxMdShowCustomer(
        <xsl:value-of select="@CustNo"/>);
    </xsl:attribute>
    <xsl:value-of select="@Company"/>
  </a>
</div>
```

## Master/Details

The last of my JavaScript functions takes as parameter the id of the customer we want to display. The function does two separate things. First it fetches some more data about the customer from the local XML cache, using XPath. Secondly, it requests the details of the master table to the server (htmldetail action):

```
function AjaxMdShowCustomer(id)
{
  var xmlDoc = xmlParse(el('xml_data').value);
  el('master').innerHTML = '<h2>' +
    xpathEval('/DATAPACKET/ROWDATA/ROW[@CustNo=' + id + ']/@Company',
      new ExprContext(xmlDoc)).stringValue() +
    '</h2>' +
    '<p>' +
    xpathEval('/DATAPACKET/ROWDATA/ROW[@CustNo=' + id + ']/@City',
      new ExprContext(xmlDoc)).stringValue() +
    '</p>';

  var params = new Object();
  params["id"] = id;

  AjaxRequest.get({
    'url': '/code/ajaxmasterdetail.exe/htmldetail',
    'parameters': params,
    'onSuccess': function(req) {
      el('detail').innerHTML = '<h3> Detail </h3>' + req.responseText;
    }
  })
}
```

In this case the Delphi application returns the ready-to-use HTML, with the following code:

```
procedure TWebModule4.WebModule4waHtmlDetailAction(
  Sender: TObject; Request: TWebRequest;
  Response: TWebResponse; var Handled: Boolean);
begin
  cdsOrders.Filter :=
    'CustNo = ' + Request.QueryFields.Values['id'];
  cdsOrders.Filtered := True;
  Response.Content :=
    DataSetTableProducer2.Content;
end;
```

## Demo Sites

As setting up this demo on your own machine might take quite some time, I've also created a Delphi/AJAXdemo site at the domain <http://ajax.marcocantu.com>. Feel free to have a look there to see this application in action and to download its complete source code.

## Wrapping Up

I'm pretty sure AJAX and similar technologies will be big in the next few years. They will be big in the ASP.NET realm, but my impression is that using AJAX reduces the need for ASP.NET and brings the simpler and much more flexible (also in terms of deployment) WebBroker back to the stage. I wonder if it is time for Delphi to resume the Internet Express technologies and turn it into an AJAX engine. I'll certainly do some more experiments with it and let you know. Stay tuned to the demo site above and to my blog, <http://blob.marcocantu.com>, for updates.