https://doi.org/10.48009/1_iis_2008_152-159

AJAX IN THE CLASSROOM

Thom Luce, Ohio University, luce@ohio.edu

ABSTRACT

The recent explosion of Web 2.0 applications has changed user's expectations regarding the web experience. Users now expect web pages to behave like desktop applications, reacting to mouse movements and individual key strokes and updating only small portions of the page at a time. While a number of technologies exist to create these rich internet applications one the most popular and widely used is AJAX. AJAX isn't a new technology but uses a number of existing technologies including JavaScript, XML and the XMLHttpRequest object to implement asynchronous browser-server communication and partial page updates. This paper explores some of the benefits of AJAX along with methods for implementing it and why it should be included in our curriculum.

Keywords: AJAX, Web 2.0, Curriculum, Job Market

INTRODUCTION

What AJAX is and where it came from

The popular computer press has been abuzz with talk of AJAX over the last few years, but should we be concerned about this in the classroom? AJAX isn't something new, it isn't a singular new technology but rather a collection of technologies that have been around for some time. The term "AJAX" was coined by Jesse James Garrett of Adaptive Path LLC as a catch phrase Asynchronous JavaScript and XML. The underlying technologies used for AJAX include 1) JavaScript; 2) the XMLHttpRequest object first introduced by Microsoft in Internet Explorer 5 and later incorporated into Mozilla based browsers; 3) XML; 4) the Document Object Model (DOM) and 5) Cascading Style Sheets for standards based presentation (1). While these technologies aren't new, their use by companies like Google (Google Suggest and Google Maps) and Amazon (A9.com) and Yahoo (Flickr) [1, 2] brought them (or what can be done with them) to the public's attention in late 2005 and 2006. Gartner's 2006 report on the "Emerging Technologies Hype Cycle" rated AJAX as "high impact and capable of reaching maturity in less than two years" [3].

The promise of AJAX

Traditional web page technology involves requesting a page from the server which sends a response back to the browser where the entire web page is refreshed – something the Open Ajax Alliance refers to as the "click, wait and refresh" cycle [4]. AJAX technologies allow developers to produce a better user experience, one that has web applications behaving more like a desktop application [5, 6, 9]. This is accomplished by asynchronous communications (using the XMLHttpRequest object) and incremental updates to the web page using the browser's document object model (DOM [4]). As a result, web pages typically update faster than in the traditional approach and they don't flicker because the whole page isn't replaced [7, 8].

According to JackBe [8] benefits of AJAX include less waiting (a web page is updated only as needed rather than all at once), faster user task performance (many things can be done on the client if it is designed properly), a familiar user interface (can look and act more like traditional windows applications) and lower bandwidth requirements (don't have to send the whole page back and forth every time). White [12] suggests the potential benefits of AJAX include 1) less wasted time waiting for data transmission; 2) less time required to complete a task; 3) less total bandwidth consumed. White goes on to say that users may benefit from fewer steps required to complete a task, the use of a familiar interface and improved responsiveness of the application.

Scott Dietzen, the CTO of Zimbra believes that "there's no other way to deliver a richly interactive experience on the Web [10]." Asaravala reports that "another benefit of AJAX … is that it makes deployment of zero-footprint software possible [11]." Asaravala goes on to say that AJAX can help automate the rollout of bug fixes and software updates to users because the software resides on the server, not on each desktop.

AJAX is one of the critical technologies in Web 2.0 [13] and Gartner's 2007 "Hype Cycle for Emerging Technologies" rates Web 2.0 as one of the technologies to watch because of its ability to transform business processes and models [14]. Gartner also predicts that "despite some difficulties, 2008 will be the year that Web 2.0 enters Type B (mainstream) enterprises [15]."

How it Works

In the now classic web page cycle a request is sent to a server when a user enters a URL in the browser's address bar or clicks a button and all activity halts as the browser waits for the server to respond (synchronous processing). When an AJAX enabled browser sends a request to the server it doesn't have to wait (asynchronous processing) but can continue to interact with the user. The user may not even be aware that a server request was made and may not know that a response was received because the browser is able to update selected parts of the page without doing a complete page refresh.

The Old Way

Consider the simple web application using Microsoft's Northwind Database shown in Figure 1.

<insert Figure 1 here>

The first dropdown in this example is filled with a list of Product Categories at the server. The second dropdown initially contains only the "Please select…" item. When the user selects a Product Category the browser sends the category ID to a request handler on the server (request handlers are similar to a web page but without a user interface) which finds the corresponding products in the database, formats the results as XML and sends it back to the browser. The browser reads the list of products, converts them to dropdown list options and adds them to the second dropdown list and changes the text of the dropdown to "Select a product" to let the user know that the data is ready (that is the only indication to the user that the operation is complete). The process is repeated when the user selects a product. In this case the product ID is sent to another request handler which looks up the product information, formats it as XML and returns it to the browser where it is displayed. All of this happens with no general post back and no flickering of the page after the initial page load.

This application was written in C# using the ASP.NET 2.0 framework. The first dropdown list is bound to an ASP.NET DataSource using standard .NET programming techniques. The remainder of the client side application is written in JavaScript and uses the XMLHttpRequest object to communicate with server side request handlers.

The first thing the browser must do is create an instance of the XMLHttpRequest object. All major browsers currently support this object but they do so in slightly different ways. Safari, Opera, Firefox and Internet Explorer 7 support XMLHttpRequest directly while earlier versions of Internet Explorer use one of several ActiveX objects. Figure 2 is a code snippet that should create an instance of the XMLHttpRequest object on most browsers.

<insert Figure 2 here>

To use the XMLHttpRequest object, client side events are set to call JavaScript functions. Figure 3 shows a .NET DropDownList set to call the client side method getProducts() when the user selects a product from the dropdown.

<insert Figure 3 here>

Code to process the drop down list's onchange event is shown in Figure 4 with comments explaining how the XMLHttpRequest object works:

<insert Figure 4 here>

The code shown in Figure 4 is fairly typical of browser – server communication with

XMLHttpRequest. The method used to communicate is generally GET or POST and the web site must be in the same domain as the original page. Synchronous communication is possible by changing the final argument to false. XMLHttpRequest can read data returned from the server as either plain text or XML. The following snippet shows some of the XML returned by getProducts.ashx.

```
<?xml version="1.0" encoding="utf-8" ?>
<Products>
   <Product>
      <name>Gustaf's Knäckebröd</name>
      <pid>22</pid>
   </Product>
   <Product>
      <name>Tunnbröd</name>
      <pid>23</pid>
   </Product>
</Products>
```

Processing this XML data requires DOM navigation of the file. Figure 5 shows code used to process the information returned by getProducts.ashx. Once again comments have been added to the code to explain the processing.

<insert Figure 5 here>

As shown in Figure 5, implementing AJAX features using the basic components, JavaScript, XML, etc., takes a fair amount of JavaScript programming and knowledge of both XML processing and the browsers Document Object Model. Fortunately a number of organizations have come forward to help with this problem.

Frameworks

In an interview with Application Development Trends, Ray Valdes, an analyst with Gartner, identified four "levels of AJAX adoption [16]." According to Valdes, the first level is the "snippet level" with small amounts of AJAX code added to existing applications (similar to the code shown earlier in this paper). Valdes identifies the second level as the "widget level." He says that this level is characterized by the creation of new interface elements, not just modifying existing elements. One example of this is the AJAX Control Toolkit from CodePlex [17]. This toolkit provides new AJAX driven interface elements for ASP.NET including a CollapsiblePanel, an Accordion control, AutoCompletion and many more.

The third level identified by Valdes is the "framework level" where he says you "throw away your existing code and re-implement the app [15]." The fourth and final level mentioned by Valdes is the "enhanced framework" level that combines both client-side and server-side elements.

There are a growing number of AJAX tool packages currently on the market. These packages simplify the coding of snippets, provide widgets and a framework, and in most cases support for an enhanced framework. Both InfoWorld [18] and DevX [19] recently reviewed a number of major widget and framework package vendors including Backbase, Bindows, Helmi Technologies, ICESoft Technologies, JackBe, Nexaweb and Tibco General Interface. In addition, Microsoft recently released Visual Studio 2008 [20] which includes enhanced framework support for AJAX.

The New Way - Visual Studio 2008

Microsoft's AJAX framework contains both client and server-side support. On the client-side are a number of components for behavior and control, support for browser compatibility, networking support for asynchronous requests, XML, web services and more along with various JavaScript extensions [21]. The server-side components add additional script support, wide support for web services and application services including authentication along with four server controls [21].

The server controls should be of special interest to instructors teaching lower-level courses who want to introduce AJAX concepts to students who aren't ready to deal with all the low level details of AJAX. The server controls consist of a ScriptManager, an UpdatePanel, an UpdateProgress control and a Timer control.

To create AJAX enhanced web pages using the ASP.NET server controls the user need only drop a ScriptManager on the page along with one or more UpdatePanels and then populate the UpdatePanels with any controls that should operate asynchronously. Any events that would normally cause a post back

will instead be processed by the ScriptManager and controls in the UpdatePanel will be updated without refreshing the entire page. The UpdateProgress control may be used to display a message or graphic when an update takes an extended period of time to complete. The Timer control may be used to force events in the UpdatePanel to occur on a regular basis, e.g. updating the time or invoking an AdRotator.

Figure 6 shows an application similar to Figure 1 except this time the drop down lists and the results are inside an UpdatePanel and there is NO user written client-side code and no server-side code.

<insert Figure 6 here>

If this were a normal web page then changes in either of the drop down lists would cause a server post back and the entire page would be refreshed with the results. Because these controls are contained in an UpdatePanel the post backs are handled asynchronously by the ScriptManager and only small portions of the page are updated.

Microsoft's AJAX framework allows for much more elaborate interactions than shown here, but this is all that is necessary to get started and should allow even beginning students to incorporate AJAX features in their web applications.

Why our students need to know

AJAX is one of the fundamental technology sets underlying Web 2.0. As previously mentioned, Gartner believes that AJAX [3] and Web 2.0 [14] will be widely used in mainstream enterprises, the future employers of our students, within a few years. A number of authors [4, 13, 19, 22] talk about AJAX and Web 2.0 technologies' role in creating front ends to Service-oriented Architecture (SOA) back-end systems. Gartner [14] considers SOA to be a transformational technology that is well along the way to becoming mainstream – yet another thing future employers will want our students to know.

As early as 2006 an article published in the ABA Journal [23] listed Web 2.0 and AJAX in a top-ten list of technologies in the legal field. Computerworld [24] lists programming and application development as one of the hottest skills of 2008 because companies are continuing to develop Web 2.0 applications using AJAX, .NET and PHP. Rothberg [25] agrees and lists AJAX and JavaScript on eWeek's pick of ten programming languages in wide use and with high marketability.

CONCLUSION

Rich internet applications, made popular by Web 2.0, are changing user's expectations of the web experience. Advances in the use and implementation of AJAX and other Web 2.0 technologies have made these technologies more accessible to students at all levels of education. As the acceptance of these technologies continues to grow in main-stream enterprises the need for developers and programmers experienced with the technologies also grows. The information systems curriculum of today should consider these technologies when preparing students for tomorrow.

REFERENCES

1. Garrett, Jesse James, "Ajax: A New Approach to Web Applications," 2005, available from http://www.adaptivepath.com/ideas/essays/archives/000385.php [accessed 3/26/2008].
2. Powazek, Derek, "Ajax, Ajax Everywhere," 2005, available from http://www.powazek.com/2005/05/000520.html [accessed on 3/26/2008].
3. Gartner, Inc., "Gartner's 2006 Emerging Technology Hype Cycle Highlights Key Technology Themes," 2006, available from http://www.gartner.com/it/page.jsp?id=495475 [accessed on 3/26/2008].
4. Open Ajax Alliance, "When Does Ajax Make Business Sense," 2008, available from http://www.openajax.org/whitepapers/When%20Does%20Ajax%20Make%20Business%20Sense.php [accessed 3/26/2008].
5. Open Ajax Alliance, "Next-Generation Applications Using Ajax and OpenAjax", 2008, available from http://www.openajax.org/whitepapers/Next-Generation%20Applications%20Using%20Ajax%20and%20OpenAjax.php [accessed 3/26/2008].
6. MacVittie, Lori, "The Impact of AJAX on the Network," available from http://www.f5.com/pdf/white-papers/ajax-wp.pdf [accessed 3/26/2008].

7. "The characteristics of Ajax applications," available from http://www.openajax.org/member/wiki/images/8/89/NexawebAjaxCharacteristics.pdf [accessed on 3/26/2008].
8. Prothero, Jerrold, "Ajax Usability Benefits and Best Practices," available from http://www.jackbe.com/downloads/JackBe_White_Paper___Ajax_Best_Practices.pdf [accessed on 3/26/2008].
9. Anglin, Todd, "The Ajax Papers Part III: Why, When and What," 2007, available from http://blogs.telerik.com/toddanglin/Posts/07-04-16/The_Ajax_Papers_Part_III.aspx?ReturnURL=%2FToddAnglin%2FPosts.aspx [accessed on 3/26/2008].
10. Krill, Paul, "AJAX benefits, issues cited by Zimbra exe," 2007, available from http://www.infoworld.com/article/07/12/03/ajax-benefits-issues-cited-Zimbra-exec_1.html [accessed on 3/26/2008].
11. Assaravala, Amit, "Putting AJAX to work," 2005, available from http://www.infoworld.com/article/05/10/17/42FEajaxcase_1.html [accessed on 3/26/2008].
12. White, Alexel, "Measuring the Benefits of Ajax," 2008, available from http://www.developer.com/java/other/article.php/3554271 [accessed on 3/26/2008].
13. OpenAjax Alliance, "Introducing Ajax and OpenAjax," 2008, available from http://www.openajax.org/whitepapers/Introducing%20Ajax%20and%20OpenAjax.php [accessed on 3/26/2008]
14. Gartner Inc., "Hype Cycle for Emerging Technologies, 2007, " 2007, available from http://www.gartner.com/DisplayDocument?doc_cd=149712&format=html [accessed on 3/26/2008].
15. Gartner Inc., "Predicts 2008: Web Technologies Continue to Drive Business Innovation," 2007, available from http://www.gartner.com/DisplayDocument?ref=g_search&id=564307&subref=simplesearch [accessed on 3/26/2008].
16. "Snippets, widgets and other levels of AJAX development," available from http://www.adtmag.com/print.aspx?id=17953 (accessed on 3/26/2008].
17. "AJAX Control Toolkit," available from http://www.codeplex.com/AtlasControlToolkit [accessed on 3/26/2008].
18. Wayner, Peter, "Top AJAX tools deliver rich GUI goodness, " available from http://www.infoworld.com/article/06/11/27/48FEajax_1.html [accessed on 3/26/2008].
19. Kunene, Glen, "AJAX, the Enterprise, and SOA – A Look Into the Future," available from http://www.devx.com/AJAXRoundup/Article/33210 [accessed on 3/26/2008].
20. Visual Studio 2008 web site at http://msdn2.microsoft.com/en-us/vs2008/default.aspx [accessed on 3/26/2008].
21. "ASP.NET AJAX Overview," available at http://www.asp.net/AJAX/Documentation/Live/overview/default.aspx [accessed on 3/26/2008].
22. Taft, Darryl, "Prepare to Merge," *eWeek, pp 29-34* 23 June 2007.
23. Krause, Jason, "Top Ten in Tech," ABA Journal, pp 40-47, December 2006.
24. Hoffman, Thomas, "8 Hottest skills for '08," Computerwrold, p 29, Vol 42(1), January 1, 2008.
25. Rothbert, Deborah, "10 Programming Languages You Should Learn Right Now," available from http://www.eweek.com/c/a/Careers/10-Programming_Languages_You-Should-Learn-Right-Now/ [accessed 3/26/2008].

FIGURES

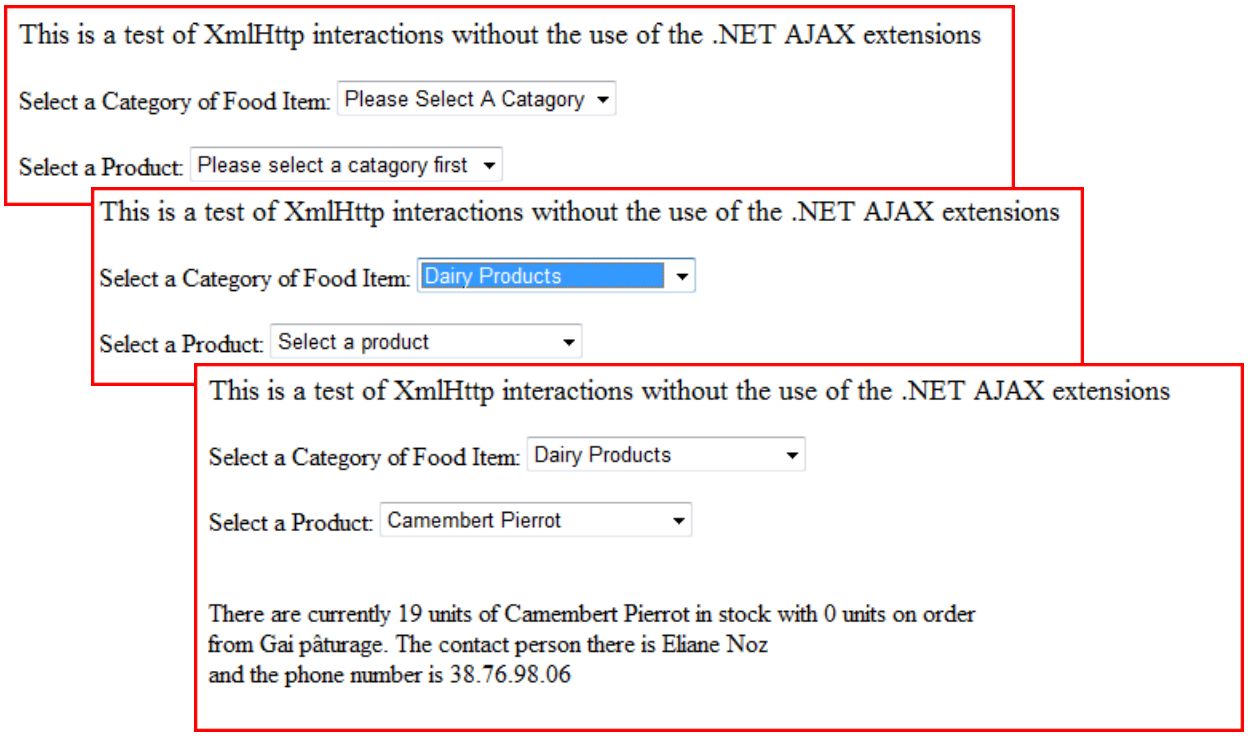

Figure 1. A simple AJAX enabled application.

```
function CreateXmlHttpRequestObject()
{
    var xmlObj;
    if (window.ActiveXObject)
    {
        try  // older browsers use different ActiveX objects
        {
           xmlObj = new ActiveXObject("Microsoft.XMLHTTP");
        } catch (e)
        {
           xmlObj = new ActiveXObject("Msxml2.XMLHTTP");
        }
    }
    else
        xmlObj = new XMLHttpRequest();

    return xmlObj;
}
```

Figure 2. Creating an instance of the XmlHttpRequest Object

```
<asp:DropDownList ID="ddlCatagory" runat="server" DataSourceID="sdsCatagories"
DataTextField="CategoryName" DataValueField="CategoryID" onchange="getProducts();">
```

Figure 3. Linking a drop down list to JavaScript.

```
function getProducts()
{
 // the ID of the selected Category from the drop down list
 var cat = document.getElementById("ddlCatagory").value;
 if(cat != -1) // The value of the “Please Select a Category” item is -1
 {
     // open the XMLHttpRequest object to communicate with  with request handler
     // getPrducts.ashx passing the category ID as a query string (GET).
     // the final argumewnt, true, indicates asynchronous processing
     xmlHttpObj.open("GET","getProducts.ashx?catID=" + cat, true);
     // the next, create a call-back function for the XMLHttpRequest object
     xmlHttpObj.onreadystatechange = function()
     {
        // the XMLHttpRequest’s readyState property changes throughout
        // the life cycle of the call.  READYSTATE_COMPLETE (set in JavaScript with
        // var READYSTATE_COMPLETE = 4;) means the process has finished
        if (xmlHttpObj.readyState == READYSTATE_COMPLETE)
        {
           // The XMLHttpRequest’s status property returns the HTTP status.
           // The following line uses a constant that was set with
           // var = HTTPSTATUS_OK = 200;
           // A value of 200 means the request handler completed its task successfully
           if (xmlHttpObj.status == HTTPSTATUS_OK)
           {
           // update the dropdown list (code described below)
           }
        }
     } // end of function
     // Finally, execute the request
     xmlHttpObj.send("SomeDataToSend");
  }
  else
  {
     // no catagory selected so the product list is cleared here
  }
}
```

Figure 4. Responding to changes in the selected value of the drop down list.

```
// the first line returns the data as an XML document
// (to return the data as text, use responseText instead of responseXML
var xmlDoc = xmlHttpObj.responseXML;
// locate the product dropdown
var prodDDL = document.getElementById("ddlProducts");
// amd remove the old products
for(var j=0; j<prodDDL.length; j++)
{
   prodDDL.options[0] = null;
}
// create a new first option for the product dropdown
prodDDL[0]=new Option("Select a product", "-1");
// Use XPath to return all the  <Product> nodes
var productNodes = xmlDoc.selectNodes("//Products/Product");
// loop through the product nodes
for(var i=0; i<productNodes.length; i++)
   {
      // use XPath expressions to locate the name and pid nodes an get their values
      var productName = productNodes[i].selectSingleNode("name/text()").nodeValue;
      var productID = productNodes[i].selectSingleNode("pid/text()").nodeValue;
      // create a new Option for each product and add it to the dropdown list
      prodDDL[prodDDL.length]= new Option(productName, productID);
   }
// set the dropdown to the first item, “Select a product”
prodDDL.selectedIndex=0;
```

Figure 5. Client processing of XML data.

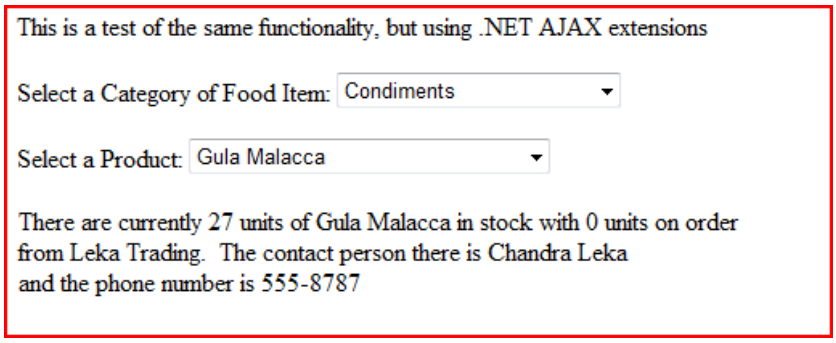

Figure 6. A simple AJAX enabled application using Microsoft’s ASP.NET AJAX extensions.