ISSN:2320-0790

An International Journal of Advanced Computer Technology

AJAX: COMING TO AN APPLICATION NEAR YOU

Ms. Swapnali H. Bhosale
Department of Computer Engineering,
Padmbhushan Vasantdada Patil College of Engineering, Sion, Mumbai-09
Email: swapna.care@gmail.com

Abstract: Today's rich Web applications use a mix of Java Script and asynchronous communication with the application server. This mechanism is also known as Ajax: Asynchronous JavaScript and XML. The intent of Ajax is to exchange small pieces of data between the browser and the application server, and in doing so, use partial page refresh instead of reloading the entire Web page.
In recent years, information system based on browse/server architecture (namely B/S architecture) received more favor by enterprises. Ajax technology consists of five parts. They are HTML (Hyper Text Markup Language), JavaScript, DHTML (Dynamic Hyper Text Markup Language), DOM (Document Object Model) and XML (Extensible Markup Language). With the help of cooperation and collaboration of these technologies, they can optimize the conventional enterprise information system by using an asynchronous way. Meanwhile, a quickly-responded and smoother user interface was provided. Enterprise information system with Ajax can be operated in a more efficient way, which means even use the current hardware, it can provide more load capacity, be more stable and serve more clients in parallel. In this paper: we present two kinds of information system models, one use conventional B/S architecture and the other use Ajax enhanced B/S architecture.

Keywords: Asynchronous, AJAX(Asynchronus Javascript And XML), XML (Extensible Markup Language) , javascript., XMLHttpRequest(Extensible Markup Language Hyper Text Transfer Protocol), ASP.net(Active Server Page. Network). [3]

I. INTRODUCTION

AJAX stands for Asynchronous JavaScript and XML. This technology was introduced first by Microsoft back in 1999, and had been known as DHTML / JavaScript web application with remote calls. AJAX is not a new programming language, but a new technique for creating better, faster, and more interactive web applications.[1]

With AJAX, a JavaScript can communicate directly with the server, with the XMLHttpRequest object. With this object, a JavaScript can trade data with a web server, without reloading the page.
AJAX uses asynchronous data transfer (HTTP requests) between the browser and the web server, allowing web pages to request small bits of information from the server instead of whole pages.

II. TECHNOLOGIES USED IN AJAX

The term AJAX has come to represent a broad group of web technologies that can be used to implement a web application that communicates with a server in the background, without interfering with the current state of the page. To implement AJAX the following technologies are required:

- XHTML(Extensible Hiper Text Markup Language) and CSS(Casceding Style Sheet) for presentation
- the Document Object Model for dynamic display of and interaction with data
- XML(Extensible Markup Language) and XSLT(Extensible Stylish Language Transformation) for the interchange, and manipulation and display, of data, respectively
- the XMLHttpRequest(Extensible Markup Language Hiper Text Transfer Protocol) object for asynchronous communication.[4]

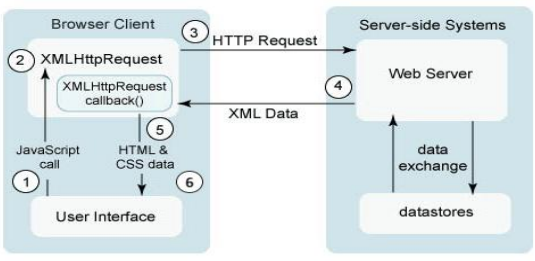

Figure1: Asynchronous JavaScript and XML[4]

III. COMPARING AJAX MODEL WITH TRADITIONAL MODEL

The classic web application model works like this: Most user actions in the interface trigger an HTTP request back to a web server. The server does some processing — retrieving data, crunching numbers, talking to various legacy systems — and then returns an HTML page to the client. It's a model adapted from the Web's original use as a hypertext medium, but as fans of The Elements of User Experience know, what makes the Web good for hypertext doesn't necessarily make it good for software applications.
This approach makes a lot of technical sense, but it doesn't make for a great user experience. While the server is doing its thing, what's the user doing? That's right, waiting. And at every step in a task, the user waits some more.
fans of The Elements of User Experience know, what makes the Web good for hypertext doesn't necessarily make it good for software applications.

This approach makes a lot of technical sense, but it doesn't make for a great user experience. While the server is doing its thing, what's the user doing? That's right, waiting. And at every step in a task, the user waits some more.

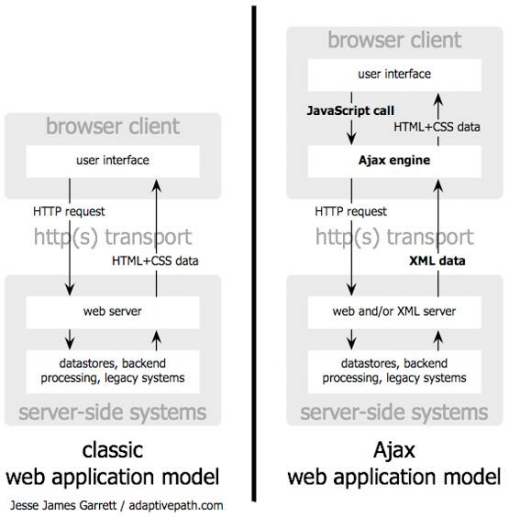

Figure 2: The traditional model for web applications (left) compared to the Ajax model (right).

Instead of loading a webpage, at the start of the session, the browser loads an Ajax engine — written in JavaScript and usually tucked away in a hidden frame. This engine is responsible for both rendering the interface the user sees and communicating with the server on the user's behalf. The Ajax engine allows the user's interaction with the application to happen asynchronously — independent of communication with the server. So the user is never staring at a blank browser window and an hourglass icon, waiting around for the server to do something.

Every user action that normally would generate an HTTP request takes the form of a JavaScript call to the Ajax engine instead. Any response to a user action that doesn't require a trip back to the server — such as simple data validation, editing data in memory, and even some navigation — the engine handles on its own. If the engine needs something from the server in order to respond — if it's submitting data for processing, loading additional interface code, or retrieving new data — the engine makes those requests asynchronously, usually using XML, without stalling a user's interaction with the application.

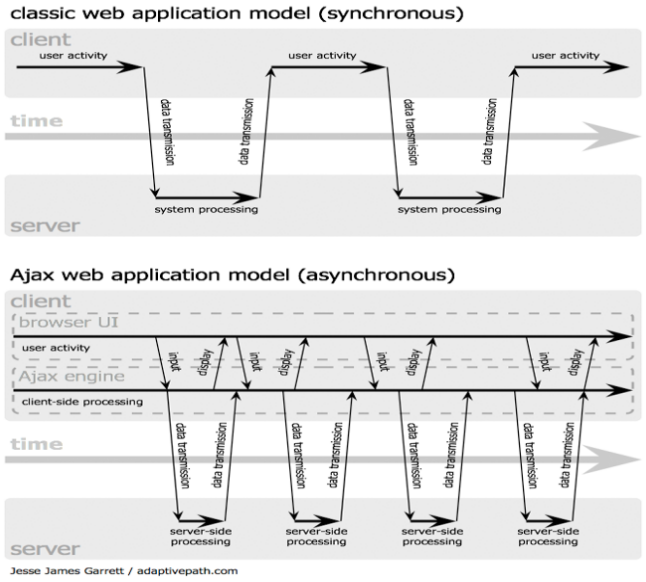

Figure 3: The synchronous interaction pattern of a traditional web application (top) compared with the asynchronous pattern of an Ajax application (bottom)[5]

IV. AJAX IS DIFFERENT TECHNOLOGY

An Ajax application eliminates the start-stop-start-stop nature of interaction on the Web by introducing an intermediary — an Ajax engine — between the user and the server. It seems like adding a layer to the application would make it less responsive, but the opposite is true.

V. USERS OF AJAX

Google is making a huge investment in developing the Ajax approach. All of the major products Google has introduced over the last year — Orkut, Gmail, the latest beta version of Google Groups, Google Suggest, and Google Maps — are Ajax applications.

VI. BROWSER SUPPORTING AJAX

Apple Safari 1.2 and above

- Microsoft Internet Explorer 5.0 and above
- Mozilla/Mozilla Firefox 1.0 and above
- Netscape 7.1 and above
- Konqueror
- Opera 7.6 and above
- Opera mobile browser 8.0 and above

Most web page with the received results without refreshing the whole page. By creators opinion, this should have improved customers experience, making HTTP pages look and feel very similar to Windows applications.

Because the core implementation of this technology was based on internet browser functionality, the usability was very limited at that time. But several years later, the technology has been reborn with new browsers support and massive implementation by such giants as Google, Amazon.com, eBay, etc.

Today, its known as AJAX, and considered as a natural part of any dynamic web page with advanced user experience.

VII. SOLUTION DESCRIPTION

The suggested approach provides a very simple, yet effective implementation of the AJAX functionality. It's very easy to maintain and change, does not require any special skills from developers, and, from our best knowledge, is cross-browser compatible.

Basically, a regular AJAX-like implementation includes two main components: a client HTML page with JavaScript code making an AJAX call and receiving a response, and a remote page that can accept a request and respond with the required information. The JavaScript code on the client page is responsible for instantiating an XmlHttp object, then providing this object with a callback method which will be responsible for processing the received information, and finally, sending a request to the remote page via the XmlHttp object. All this is done by the JavaScript code.

Our approach is intended for use in ASP.NET applications, and considers the following possible scenarios:

- AJAX calls may be performed on different ASP.NET pages of the web application to different remote pages;
- A remote page URL may contain dynamically calculated parameters, and it may be more convenient to build a URL string in the code-behind of the ASP.NET page;
- A remote page may respond with a complex data requiring parsing before updating an HTML page, that once again may be done in the code-behind of the ASP.NET page;
- A remote page may be either an external third party page, or the web applications own page or service.
- All these considerations are illustrated by the diagram below:

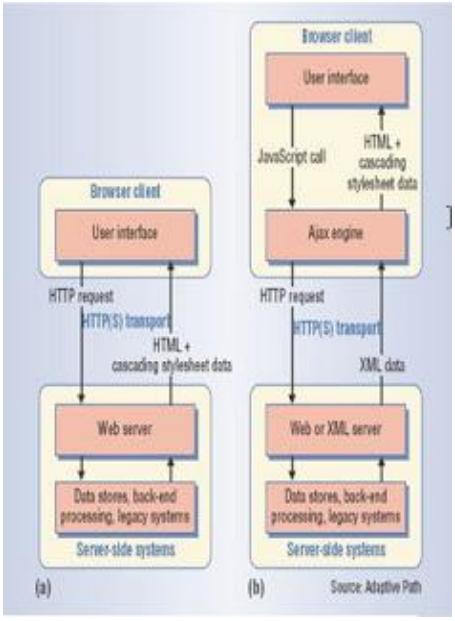

Figure 4: In (a) a traditional Web application, user actions trigger an HTTP request to a Web server, which processes the request and returns an HTML page to the client. Additional requests lock up the application until the system updates the page. (b) Ajax applications create a JavaScript-based engine that runs on the browser. The engine intercepts user inputs, displays requested material, and handles many interactions on the client side. If the engine needs more data, it requests material from the server in the background, while the user continues to interact with the application.

VIII. IMPLEMENTATION

A. Basic AJAX JavaScript methods

The JavaScript methods are divided into two parts: calling page specific JavaScript methods and AJAX JavaScript methods common for all the calling pages. Specific methods include a callback method as well, as it is responsible for updating the page content. Common AJAX methods are responsible for instantiating an XmlHttp object and sending an asynchronous request to the remote page.
Getting an XmlHttp object differs depending on the type of the browser. It is distinguished in to two basic types: a Microsoft browser which is one of the IE families, and a Mozilla browser which is one of Mozilla Firefox, Netscape, or Safari.

```
function GetXmlHttpObject(handler)
{
    var objXmlHttp = null;
    if (!window.XMLHttpRequest)
    {
```

```
        // Microsoft
        objXmlHttp = GetMSXmlHttp();
        if (objXmlHttp != null)
        {
             objXmlHttp.onreadystatechange
= handler;
        }
    }
    else
    {
        // Mozilla | Netscape | Safari
             objXmlHttp       =       new
XMLHttpRequest();
        if (objXmlHttp != null)
        {
            objXmlHttp.onload = handler;
            objXmlHttp.onerror = handler;
        }
    }
    return objXmlHttp;
}
function GetMSXmlHttp()
{
    var xmlHttp = null;
            var          clsids          =
["Msxml2.XMLHTTP.6.0","Msxml2.XMLHTTP.5.0
",

"Msxml2.XMLHTTP.4.0","Msxml2.XMLHTTP.3.0"
,
                "Msxml2.XMLHTTP.2.6","Mi
crosoft.XMLHTTP.1.0",
                "Microsoft.XMLHTTP.1","M
icrosoft.XMLHTTP"];
      for(var   i=0;   i<clsids.length   &&
xmlHttp == null; i++) {
                    xmlHttp             =
CreateXmlHttp(clsids[i]);
    }
    return xmlHttp;
}
function CreateXmlHttp(clsid) {
    var xmlHttp = null;
    try {
             xmlHttp        =        new
ActiveXObject(clsid);
        lastclsid = clsid;
        return xmlHttp;
    }
    catch(e) {}
}
```

The code for the GetMSXmlHttp method can be simplified considering that user do not have to refer MSXML5 as it has been designed only for Office applications. Correspondingly, the simplified revision of the GetMSXmlHttp method may look as follows:

```
function GetMSXmlHttp() {
    var xmlHttp = null;
    var clsids = ["Msxml2.XMLHTTP.6.0",
                  "Msxml2.XMLHTTP.4.0",
                  "Msxml2.XMLHTTP.3.0"];
      for(var   i=0;   i<clsids.length   &&
xmlHttp == null; i++) {
                    xmlHttp             =
CreateXmlHttp(clsids[i]);
    }
    return xmlHttp;
}
```

GetXmlHttpObject methods accept a handler parameter which is a name of the callback method that should be defined in the page-specific code.An XmlHttp object, can be send an asynchronous request.

```
Function
SendXmlHttpRequest(xmlhttp, url) {
    xmlhttp.open('GET',        url,
true);
xmlhttp.send(null);
}
```

GET HTTP method can be use to a given URL, but this can be easily changed by changing the JS code above.

B. Page-specific methods

Methods need to perform a call to the remote page. In order to do this, user need to pass the callback method name to the `GetXmlHttpObject` method and then pass the URL string to the `SendXmlHttpRequest` method.

```
var xmlHttp;
function ExecuteCall(url)
{
    try
    {
        xmlHttp                          =
GetXmlHttpObject(CallbackMethod);
        SendXmlHttpRequest(xmlHttp, url);
    }
    catch(e){}
}
//CallbackMethod will fire when the state
//has changed, i.e. data is received back
function CallbackMethod()
{
    try
    {
         //readyState  of  4  or  'complete'
represents
```

```
        //that data has been returned
        if (xmlHttp.readyState == 4 ||
            xmlHttp.readyState          ==
'complete')
        {
                  var       response       =
xmlHttp.responseText;
            if (response.length > 0)
            {
                //update page

document.getElementById("elementId").inne
rHTML

          = response;
            }
        }
    }
    catch(e){}
}
```

The `CallbackMethod` is responsible for updating the page content. It simply updates the inner HTML of the given HTTP element. But in real time, it can be much more complex.

The last question regarding the calling page implementation is how this call's the `ExecuteCall` JS method. Well, it depends on what the page is doing. In some cases, the `ExecuteCall` method can be called when the JS event is fired. But if that is not the case, then register the method as a startup script for the page using the corresponding C# code in the page's code-behind.

```
Page.RegisterStartupScript("ajaxMethod",
   String.Format("<script>ExecuteCall('{0
}');</script>", url));
```

This code can be added either in the `Page_Prerender` or `Page_Load` method of the ASP.NET code-behind file.

REMOTE PAGE

Let's find out what a remote page could look like. If this is an ASP.NETpage. If user remove all the code from the *.aspx* file: it won't affect the behavior of the page in any way.

For example, take a public web service that converts temperature values in Celsius to Fahrenheit and vice versa. The service is available here. If user add this URL as a web reference to this project, Visual Studio will generate a proxy class with the name `com.developerdays.ITempConverterservice` in current namespace.Remote ASP.NET page, lets name it *getTemp.aspx*, will accept a query string parameter with the name temp which should contain an integer value of a temperature in Celsius to convert. So the target URL to the remote page will look like this: *http://localhost/getTemp.aspx?temp=25*. And the code-behind for this page is shown below:

```
private   void   Page_Load(object   sender,
EventArgs e)
{
    Response.Clear();
             string          temp          =
Request.QueryString["temp"];
    if (temp != null)
    {
        try
        {
            int tempC = int.Parse(temp);
                 string       tempF       =
getTempF(tempC);
            Response.Write(tempF);
        }
        catch
        {
        }
    }
    Response.End();
}
private string getTempF(int tempC)
{

com.developerdays.ITempConverterservice
          svc             =             new
ITempConverterservice();
    int tempF = svc.CtoF(tempC);
    return tempF.ToString();
}
```

According to our convention, we can now build a URL string for the remote page that we will pass to `theRegisterStartupScript` method in the example above, like this:

```
int tempC = 25;
string                 url                 =
String.Format("http://localhost/" +
```

```
            "getTemp.aspx?temp={0}",
tempC);
```

Using the approach with an intermediate ASP.NET page, calling in its turn a remote service, allows simplifying response processing, especially if it requires parsing. In simple cases when the response contains just text, we can pass the remote service URL directly to the JS `ExecuteCall` method.

The Ajax engine has multiple responsibilities (Figure 2)

Figure 5:

1. *Detecting user interactions. The engine detects and reacts to user interactions as they take place. For example, if the user hovers the mouse over a specific area, the Ajax engine recognizes the action and triggers an HTTP request*.
2. *Submitting HTTP request to the server. When the pre-defined user interaction takes place, the Ajax engine submits a request to the Web server asynchronously*.
3. *Handling HTTP response returned by the server. The engine handles markup returned by the Web or application server. For example, if the response is XML, the Ajax engine applies the XSL style sheet to it*
4. *Performing partial page refresh. The engine makes the necessary changes to the Document Object Model (DOM), which is the internal representation of the HTML document, thus updating the rendered page. For example, the engine can display*

IX. ADVANTAGES

- Using traditional methods, that content would have to be reloaded on every request. However, using AJAX, a web application can request only the content that needs to be updated, thus drastically reducing bandwidth usage and load time.
- The use of asynchronous requests allows the client's Web browser UI to be more interactive and to respond quickly to inputs, and sections of pages can also be reloaded individually. Users may perceive the application to be faster or more responsive, even if the application has not changed on the server side.
- The use of AJAX can reduce connections to the server, since scripts and style sheets only have to be requested once
- State can be maintained throughout a Web site. JavaScript variables will persist because the main container page need not be reloaded.

X. DRAWBACKS

- AJAX interfaces are substantially harder to develop properly than static pages.
- Pages dynamically created using successive AJAX requests do not automatically register themselves with the browser's history engine, so clicking the browser's "back" button may not return the user to an earlier state of the , but may instead return them to the last full page visited before it.
- Dynamic web page updates also make it difficult for a user to bookmark a particular state of the application. Solutions to this problem exist, many of which use the URL fragment identifier to keep track of, and allow users to return to, the application in a given state.
- Any user whose browser does not support JavaScript or XMLHttpRequest, or simply has this functionality disabled, will not be able to properly use pages which depend on AJAX. Similarly, devices such as mobile phones, PDAs, and screen readers may not have support for the required technologies. Screen readers that are able to use AJAX may still not be able to properly read the dynamically generated content.
- AJAX opens up another attack vector for malicious code that web developers might not fully test for.

WHAT NEXT:

Some sources say the recent attention to Ajax has also brought attention to rich Web applications, which will help vendors using other development approaches, Garrett said. According to Norbye, better browsers, tools, and network performance will improve Ajax's capabilities in the future. Ajax could find various uses. For example, vendors could use it to build Web based versions of desktop applications. This way, companies could make software widely available to employees via a network and thus avoid spending the time and money required

to install applications on every computer. Ajax also could be useful for the growing number of Web applications for mobile devices. However, predicted Root, while Ajax may prove interesting to developers now, they may turn to versions of Flash and other technologies in the future because, for example, Flash supports audio, video, advanced vector graphics, and other capabilities that Ajax can't offer. Because they find it useful, companies will create more Ajax-based applications in the near future, predicted Kevin Lynch, Macromedia's chief software architect."We're now entering a period of experimentation," said Adaptive Path's Garrett. "A lot of people in the past six months became aware of the possibilities that Ajax opens up for them. Developers are pushing at the boundaries of what they can do with it." Ajax will do well as long as it is competitive with other approaches. For example, Google's Taylor said, his company will use Ajax as long as it likes what the technology offers. He explained, "We will use whatever technology platform provides the richest user experience possible."

X. CONCLUSION

The AJAX technique makes Internet applications smaller, faster and more user-friendly. AJAX is a technology, which breaks the paradigm of page reload and saves lot of bandwidth. It can send and retrieve the data without reloading the web page, meaning, that gone are the days where for each data retrieval and we needed to reload the complete page.

Therefore AJAX technology is using in all web browsers, hence we can conclude that this technology will stand in the first of all technologies.

XI. REFERENCES

[1] http://www.techterms.com/definition/ajax
[2] www.ajaxtechnology.com
[3] http://www.slideshare.net/Zia_Rehman/ajax-technology
[4] http://sureshjain.wordpress.com/category/ajax/
[5] www.adaptivepath.com/ideas/ajax-new-approach-web-applications
[6] www.wrox.com
[7] www.howstuffworks.com
[8] http://www.tutorialspoint.com/ajax/ajax_browser_support.htm
[9] http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications
[10] AJAX by Nicholas C. Zakas, Jcremy Mcpeak, Joe Fawcett