iJET EXPLORE International Journal of Engineering & Technology
CENTRE OF PROFESSIONAL RESEARCH PUBLICATIONS
ISSN 2049-3444

International Journal of Engineering and Technology Volume 2 No. 7, July, 2012

Ajax Enabled Web Application Model with Comet Programming

Rajendra Kachhwaha[1], Priyadarshi Patni[2]
[1]Department of I.T., Faculty of E & T, Jodhpur National University, Jodhpur
[2] Department of Computer Science, Lachoo Memorial College of Science & Technology, Jodhpur

ABSTRACT

For any web application the connectivity with the server is always a crucial issue. Correctly accessing the data and availability to user at the faster rate is the major issue. In classical approach, stateless request and response make refreshing of complete page. The AJAX engine improves the situation from complete reference to server to need base access. Ajax can help in making a new model, in which the single-page web interface is composed of individual components that can be updated/ replaced independently. In this paper AJAX enabled web application model with comet programming is introduced and tested. Two approaches for long polling are considered namely, using UpdatePanel and using JSON. The experimental results are compared using Firebug tool provided by Mozilla Firefox.

Keywords: *Ajax, Reverse Ajax, Web traffic reduction, Web page auto refreshing, Long polling, JSON*

1. INTRODUCTION

The use of web applications is increasing more and more over the last few years, which includes business promotions, online working of any organization, social connectivity with the people and many more. One of the key differentiators for web applications is the connectivity – how they connected to the server over a network and receive and send data over network connections. In its early age, web applications were having static content delivery mechanism over a stateless request and response architecture of HTTP. Any piece of information that the client needs has to be requested explicitly by the client and the server sends the requested data. To display any piece of real time data, the web page is reloaded completely. In this basic model, there is synchronization between user activity and the data transmission. The client becomes idle and doses nothing except waiting for the response. Ajax can overcome this drawback by providing a new model, in which auto refreshing of any web page is performed with a specific polling period. Comet programming or Reverse Ajax starts a new era of web application with rich interactive experience. Comet programming is a web application model in which a long-held HTTP request allows a web server to push data to a browser, without the browser explicitly requesting it.

2. MODELS OF WEB APPLICATION

A Web application can be categorized into two models:

- A Classical Web Application Model
- An Ajax Enabled Web Application Model with auto refreshing
- An Ajax Enabled Web Application Model with comet programming

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.

An Ajax enabled web 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. 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.

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 does not 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 is 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.

Comet programming provides a model in which a long-held HTTP request allows a web server to push data to a browser, without the browser explicitly requesting it.

Comet applications attempt to eliminate the limitations of the page-by-page web model and traditional polling by offering real-time interaction, using a persistent HTTP connection between the server and the client. Since browsers and proxies are not designed with server events in mind, several techniques to achieve this have been developed.

Specific methods of implementing Comet fall into two major categories: Streaming & Long Polling.

Streaming: An application using streaming opens a single persistent connection from the client browser to the server for all comet events. These events are incrementally handled and interpreted on the client side every time the server sends a new event, with neither side closing the connection. Hidden iframe and XMLHttpRequest are used for accomplishing streaming Comet.

Long Polling: Long polling is a variation of the traditional polling technique and allows emulation of an information push from a server to a client. With long polling, the client requests information from the server in a similar way to a normal poll. However, if the server does not have any information available for the client, instead of sending an empty response, the server holds the request and waits for some information to be available. Once the information becomes available (or after a suitable timeout), a complete response is sent to the client. The client will normally then immediately re-request information from the server, so that the server will almost always have an available waiting request that it can use to deliver data in response to an event.

3. IMPLEMENTATION

To show the advantages (In terms of data downloaded & time taken in downloading & displaying data) of Ajax enabled web application over classic web application, few web pages were made.

A web page named as UpdateInfo.aspx is provided to Alumni members to update their contact information. The home page named as Index.aspx, shows a number with each batch for all pass out batches. This number shows count value of those alumni members who updated their information successfully.

The table below shows the technique names with their respective web page name used.

Table 1. Implemented Techniques with Web Page Name

S.No.	Technique Name	Web Page Name
1.	Classic Web Application Model	Index1.aspx
2.	Ajax Enabled Web Application Model with Auto Refreshing	Index2.aspx
3.	Ajax Enabled Web Application Model with Comet Programming (Using UpdatePanel)	Index3.aspx
4.	Ajax Enabled Web Application Model with Comet Programming (Using JSON)	Index4.aspx

3.1 Using Classic Web Application Model

In a classic web application model, the web page, Index1.aspx is reloaded completed with each request response process initiated by a client. The user requests the information by the user interface (ex. browser). After receiving this request, the server processes it and makes a response for it. This response is sent back to the client side. If there is a successful update operation performed by the any client using UpdateInfo.aspx web page, the client currently viewing home page (Index1.aspx) doesn't get the updated information. To get the updated information, client needs to reload the web page. Following timing diagram shows the activity of client.

3.2 Using Ajax Enabled Web Application Model with Auto Refreshing

Now Consider Index2.aspx web page refreshing just the parts that change makes a lot more sense than refreshing the entire page (Index1.aspx) for a few minor changes. Auto Refresh offers a significant benefit of less work for client. A specific polling period is set so that it should spread out the requests more evenly. Polling period of 15 seconds is used in the experiment. If there is a successful update operation performed by any client using UpdateInfo.aspx web page, the client currently viewing home page (Index2.aspx) doesn't need to reload the web page because the updated information will be displayed to after polling period. Following timing diagram shows the activity of client.

3.3 Using Ajax Enabled Web Application Model with Comet Programming

In previous technique (Index2.aspx), the user will see updated information with a maximum delay of 15 seconds. The Comet programming technique (Index3.aspx & Index4.aspx) is used when a periodic refresh delay is too big, and we cannot decrease this time because we will break down our server with lots of data refresh requests. Here, comet programming is used for long polling. Following timing diagram shows the activity of client.

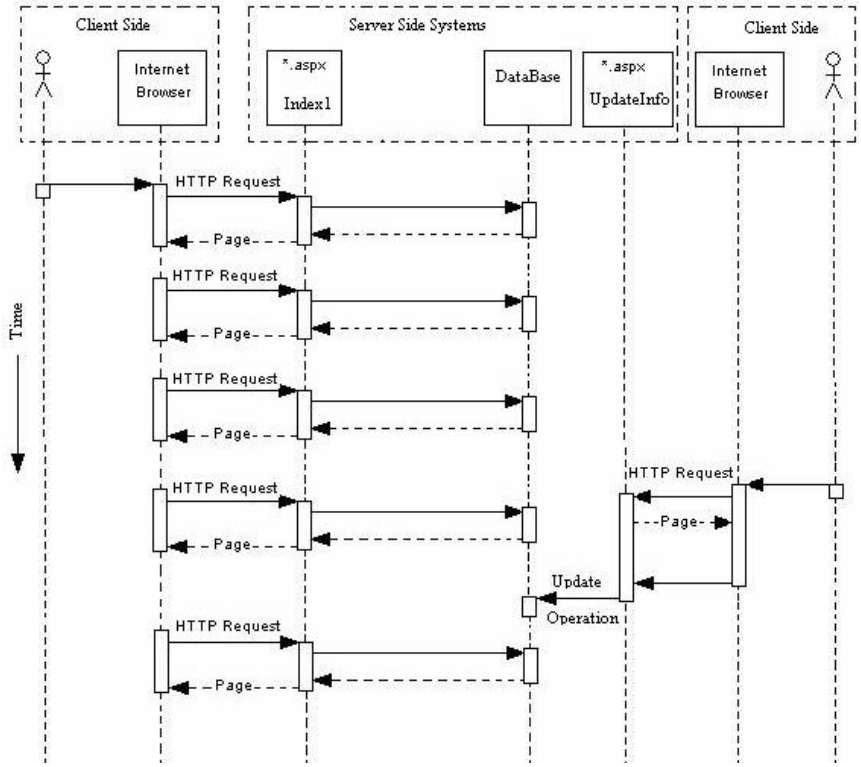

Fig 1. Client activity in Index1.aspx

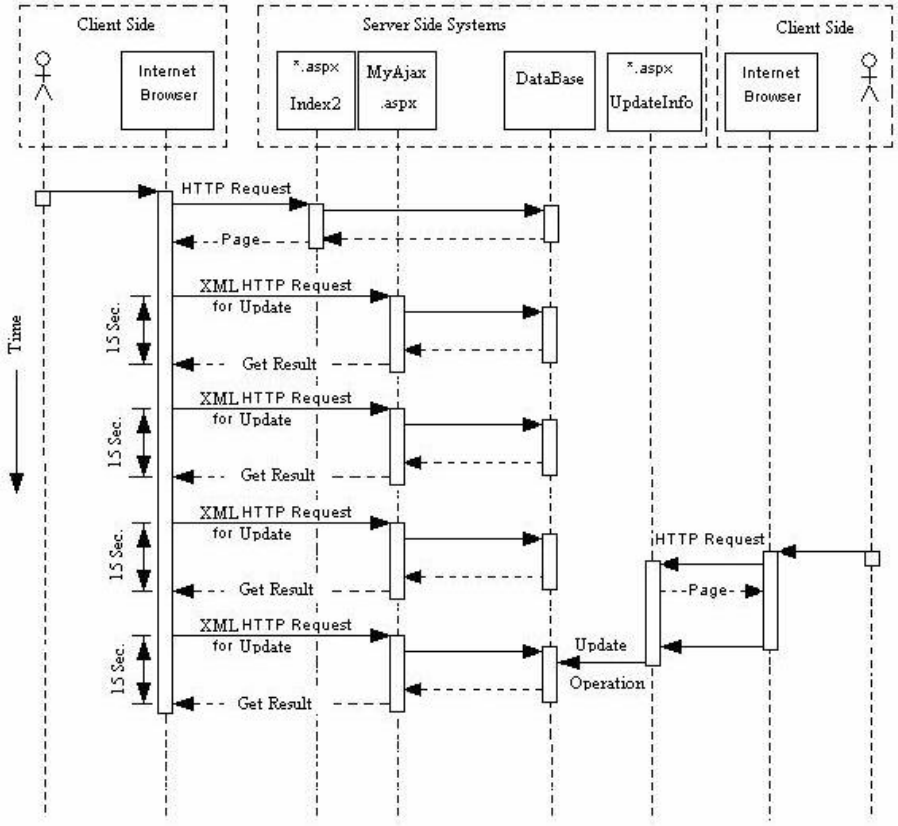

Fig 2. Client activity in Index2.aspx

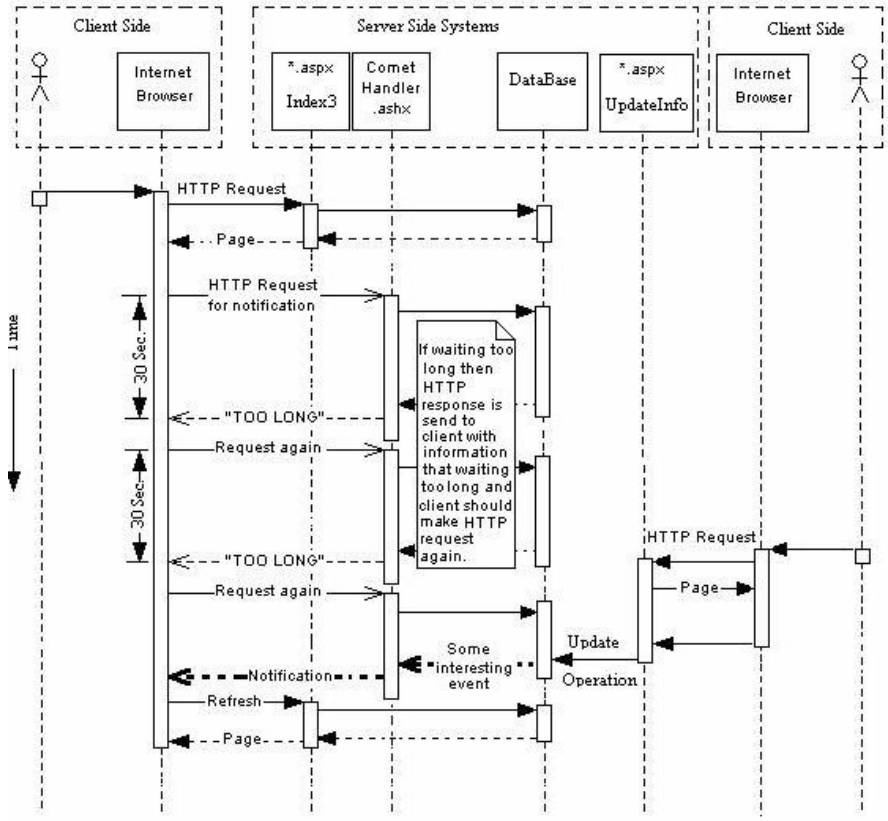

Fig 3. Client activity in Index3.aspx & Index4.aspx

Long polling is implemented in two different ways:

- Long Polling Using UpdatePanel
- Long Polling Using JSON

3.3.1 Long Polling Using UpdatePanel

The Asp.net UpdatePanel makes AJAX trivially easy for anyone to implement, even without knowledge of what's actually going on behind AJAX. Unfortunately, there is a lack of transparency regarding the mechanism of the client/server exchanges. Index3.aspx is implemented using UpdatePanel. Whenever it found a database update signal (notification), an asynchronous request is made to change the displaying information. To display the information, lot of data received (Approx 1555 bytes). It is acceptable for infrequently used functionality but a potential deal breaker in heavy use. Following figure shows the data received from the server in response to the client update request.

```
621|updatePanel|UpdatePanel1|

                    <div>
        <table class="body_text_home" cellspacing="0" rules="all" border="1" id="GridView1" style="width:170px
;border-collapse:collapse;">
                <tr>
                        <th scope="col">Item</th>
                </tr><tr>
                        <td>2006 (RU)       :       7</td>
                </tr><tr>
                        <td>2007 (RU)       :       2</td>
                </tr><tr>
                        <td>2008 (RU)       :       3</td>
                </tr><tr>
                        <td>2009 (RU)       :       4</td>
                </tr><tr>
                        <td>2010 (RTU)     :      4</td>
                </tr><tr>
                        <td>2011 (RTU)     :      3</td>
                </tr>
        </table>
</div>

                    |604|hiddenField|__VIEWSTATE|/wEPDwULLTEzMDcxMjU2MzMPZBYCAgMPZBYEAgUPFgIeCW1
ubmVyaHRtbAUIMjk2IGlzZWNkAgkPZBYCZg9kFgICAQ88KuANAgAPFgQeC18hDCF0YUJvdW5kZnd4LXyFJdGVtQ291bnQCBmQMFCs
AADYCHgRUeXB1GSsCHgROYW11BQRJdGVtHglEYXRhRml1bGQFASEWAmYPZBY0AgEPZBYCZg8PFgIeBFRleHQFFjIwMDYgKFJVKSA
gICAgIDogICAgIDdkZAICD2QWAmYPDxYCHwYFFjIwMDcgKFJVKSAgICAgIDogICAgIDJkZAIDD2QWAmYPDxYCHwYFFjIwMDggKFJ
VKSAgICAgIDogICAgIDNkZAIED2QWAmYPDxYCHwYFFjIwMDkgKFJVKSAgICAgIDogICAgIDRkZAIFD2QWAmYPDxYCHwYFFTIwMTA
gKFJUVSkgICAgOiAgICAgNGRkAgYPZBYCZg8PFgIfBgUWMjAxMSAoUlRVKSAgICA6ICAgICAzZGQCBw8PFgIeB1Zpc2liBGVoZGQYAQUJR3JpZFZpZXcxDzwrAAkBCAIBZDr1Nr8tNbk7
|0|asyncPostBackControlIDs|||0|postBackControlIDs|||13|updatePanelIDs||tUpdatePanel1|0|childUpdatePanelIDs
|||12|panelsToRefreshIDs||UpdatePanel1|2|asyncPostBackTimeout||90|11|formAction||Index3.aspx|58|pageTitle
||Technic III: AJAX Enabled Web Model(With Server Initiated)|
```


Fig 4. Server Response in UpdatePanel

3.3.2 Long Polling Using JSON

Instead of posting back and then receiving html content to completely replace our UpdatePanel's content, we can use a web method to request only the information that we are interested in using JSON (Java Script Object Notation). JSON is perfectly suited for light weight communication between client and server. Index4.aspx is implemented using JSON. With this, we have completely eliminated the Http Post data that was present in the UpdatePanel's request and reduced the response down to just the data we are interested in requesting. Following is the data received (92 Bytes) from the server in response to the client update request.


```
*18*7*2006 (RU)*22*2*2007 (RU)*20*3*2008 (RU)*21*3*2009 (RU)*19*4*2010 (RTU)*23*3*2011 (RTU)
```


Fig 5. Server Response in JSON

4. RESULTS

Before comparing, we see the page refresh dependency with use of notification. This is shown in the following table.

Table 2. Web Page Name with Page Refresh Dependency

Web Page Name	Page Refresh Dependency	Use of Notification
Index1.aspx	On User Event	No
Index2.aspx	Automatic (After 15 Seconds)	No
Index3.aspx	Automatic (After 30 Seconds)	Yes
Index4.aspx	Automatic (After 30 Seconds)	Yes

4.1 Experimental Work Done

Now we perform a comparison of implemented techniques in terms of data downloaded and time taken to download that data, either the database is successfully updated or not. For this, we first run the application and perform an Alumni member's contact information update operation using UpdateInfo.aspx web page. Then we use an Index1.aspx web page to show the working of classic web application model. Now we perform one more information update operation using UpdateInfo.aspx web page. This time, Index2.aspx web page is used to show the working of AJAX enabled web application with auto refresh. Now we perform one more information update operation using UpdateInfo.aspx web page. This time, Index3.aspx web page is used to show the working of AJAX enabled web application with comet programming (using UpdatePanel). Now we perform one more information update operation using UpdateInfo.aspx web page. This time, Index4.aspx web page is used to show the working of AJAX enabled web application comet programming (using JSON). Now we examine the effect of technology trends, when we see the size of downloaded data (in bytes) and time taken to download (in ms) in above four cases. The experimental results, as shown in following table, are compared using Firebug: a tool provided by Mozila Firfox, has a variation in the range of 0.020 ms to 0.090 ms in time taken parameter values shown on the respective web pages.

Table 3. Comparison of Implemented Techniques

Web Page Name	Database Updated	Result Noticed	Polling Interval (in ms)	Downloaded /Time Taken (byte / ms)	
				For Notification	For Data
Index1.aspx	No	I Time	No Polling	00	3468/ 8670
		II Time	No Polling	00	3468/ 6890
		III Time	No Polling	00	3468/ 7230
	Yes	I Time	No Polling	00	3468/10210
		II Time	No Polling	00	3468/ 8750
		III Time	No Polling	00	3468/ 9280
Index2.aspx	No	I Time	15000	00	92/ 2450
		II Time	15000	00	92/ 885
		III Time	15000	00	92/ 627
	Yes	I Time	15000	00	92/ 223
		II Time	15000	00	92/ 176
		III Time	15000	00	92/ 129
Index3.aspx	No	I Time	30000	1	00
		II Time	30000	1	00
		III Time	30000	1	00
	Yes	I Time	30000	1/ 112	1555/ 1725
		II Time	30000	1/ 126	1555/ 1256
		III Time	30000	1/ 114	1555/ 989
Index4.aspx	No	I Time	30000	1	00
		II Time	30000	1	00
		III Time	30000	1	00
	Yes	I Time	30000	1/ 114	92/ 412
		II Time	30000	1/ 109	92/ 219
		III Time	30000	1/ 107	92/ 98

Note: Time to load the pages may vary depending on the Internet Connection.

5. CONCLUSION

Web Applications are becoming increasingly complex. Correctly accessing the data and availability to user at the faster rate is the major issue. Ajax provides user-friendlier environment like any window application runs on a local system, over a web application with reducing server traffic and network bandwidth with time reduction in server response time.

We have experimental results show that data transferred in third case is .9 byte/ms at I time and 1.57 bytes/ms at III time. In fourth case it varies from .22 to .93. These two cases show better data transfer rate which means time reduction in server response time.

REFERENCES

[1] Dr. Shahram Khosravi, "ASP.NET AJAX Programmers Reference with ASP.NET 2.0 or ASP.NET 3.0", Published by Wiley Publishing, Inc, ISBN: 978-0-470-10998-4, Chapter 1, pp 2-3.

[2] Ryan Asleson and Nathaniel T. Schutta, "Foundations of AJAX", ISBN: 1-59059-582-3, Chapter 2, pp 23-28, Chapter 8, pp 220-222.

[3] Matthew MacDonald, "Beginning ASP.NET 3.5 in C# 2008: From Novice to Professional", Second Edition, ISBN: 978-1-59059-891-7, Chapter 25, pp 855-858.

[4] Glenn Johnson and Tony Northrup, "Microsoft .Net Framework 2.0 Web Based Client Development", Self Paced Training, Published by Microsoft Press, ISBN: 978-0-7356-2334-7, Chapter 1, pp 3-6.

[5] Jesse James Garrett, "Ajax: A new approach to web applications", http://www.adaptivepath.com/ideas/essays/archives/000385.php.

[6] CoperNick, “HTTP Push from SQL Server Comet SQL”, http://www.codeproject.com/Articles/63568/HTTP-Push-from-SQL-Server-Comet-SQL.

[7] Sandeep Malik, \Implement a real-time server push in Ajax applications using socket-based RIA technologies", http://www.ibm.com/developerworks/web/library/wa-aj-socket/