Chandara
LEARN THIS & LEARN THAT

March 6, 2012

Comet Programming: the Hidden IFrame Technique

1 Comment

As covered in Comet Programming: Using Ajax to Simulate Server Push (http://www.webreference.com/programming/javascript/rg28/), Comet is a Web application model that enables Web servers to send data to the client without having to explicitly request it. Hence, the Comet model provides an alternate mechanism to classical polling to update page content and/or data. In that article, we learned how to use XMLHttpRequest long polling to refresh page components and keep cached data in synch with the server. Another Comet strategy, sometimes referred to as the "Forever Frame" technique, is to use a hidden IFrame. As with XMLHttpRequest long polling, the "Forever Frame" technique also relies on browser-native technologies, rather than on proprietary plugins or other special technologies.

IFrames Technique Overview

IFrame stands for Inline Frame. It allows a Web page to embed one HTML document inside another HTML element. As you can imagine, it's a simple way to create a "mashup", whereby the page combines data from several sources into a single integrated document:

	view plain (http://www.webreference.com/programming/javascript/rg30/index.html#) | print (http://www.webreference.com/programming/javascript/rg30/index.html#) | ? (http://www.webreference.com/programming/javascript/rg30/index.html#)
1	<html>
2	<head>

	view plain (http://www.webreference.com/programming/javascript/rg30/index.html#) | print (http://www.webreference.com/programming/javascript/rg30/index.html#) | ? (http://www.webreference.com/programming/javascript/rg30/index.html#)
3	<title>IFrames Example</title>
4	<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
5	</head>
6	<body bgcolor="#FFFFFF" id=body>
7	IFrames Example
8	The content below comes from the website http://www.robgravelle.com
9	<iframe src="http://www.robgravelle.com/" height="200"&gt (http://www.robgravelle.com/" height="200"&gt);
10	Your browsers does not support IFrames.
11	</iframe>
12	</body>
13	</html>

Normally, data delivered in HTTP responses is sent in one piece. The length of the data is indicated by the`Content-Length` header field. With chunked encoding, the data is broken up into a series of blocks of data and transmitted in one or more "chunks" so that a server may start sending data before it knows the final size of the content that it's sending. Note that the size of the blocks may or may not be the same:

	view plain (http://www.webreference.com/programming/javascript/rg30/index.html#) | print (http://www.webreference.com/programming/javascript/rg30/index.html#) | ? (http://www.webreference.com/programming/javascript/rg30/index.html#)
1	HTTP/1.1 200 OK
2	Content-Type: text/plain
3	Transfer-Encoding: chunked
4	23
5	This is the data in the first chunk
6	1A

	view plain (http://www.webreference.com/programming/javascript/rg30/index.html#) | print (http://www.webreference.com/programming/javascript/rg30/index.html#) | ? (http://www.webreference.com/programming/javascript/rg30/index.html#)
7	and this is the second one
8	0

In Forever Frame Streaming, a series of JavaScript commands is sent to a hidden IFrame as a chunked block. As events occur, the IFrame is gradually filled with script tags, containing JavaScript to be executed in the browser. Because browsers render HTML pages incrementally, each script tag is executed as it is received.

Two benefits of the IFrame method are that it's fairly easy to implement and it works in every common browser. On the cons side, there is no way to handle errors reliably nor is it possible to track the state of the request calling process. Therefore, if you want to track the progress of the request, you'd best stick with the XMLHttpRequest method.

The CD Sales Page Revisited

You'll remember this example from the Comet Programming: Using Ajax to Simulate Server Push (http://www.webreference.com/programming/javascript/rg28/) article. We'll modify it in order to highlight the differences between the Ajax and IFrame techniques.

Minimal JavaScript Code

Many developers make the mistake of using Ajax to call the PHP script from the Web page. This is unnecessary and blurs the line between the two techniques. My view is that if you're going to go through the trouble of employing Ajax, you might as well forego the IFrame. One of the main benefits of the Hidden IFrame technique in my view is its simplicity and light use of client-side scripting. In fact, we only need one small JavaScript function:

	view plain (http://www.webreference.com/programming/javascript/rg30/index.html#) | print (http://www.webreference.com/programming/javascript/rg30/index.html#) | ? (http://www.webreference.com/programming/javascript/rg30/index.html#)

	view plain (http://www.webreference.com/programming/javascript/rg30/index.html#) | print (http://www.webreference.com/programming/javascript/rg30/index.html#) | ? (http://www.webreference.com/programming/javascript/rg30/index.html#)

```
1  <script type="text/javascript">
2      function updateCount(c) {
3          document.getElementById('CD Count').innerHTML = c;
4      }
5  </script>
```

The `updateCount()` function refreshes the CD count within the "CD Count" <SPAN> element. The innerHTML property is used because some browsers do not support innerText.

Integrated PHP Code

Another departure from the Ajax Comet page is that all the required functionality is contained within the same file. PHP, like ASP and JSP is ideal for inserting within the HTML markup. By integrating code in one file, the results of actions that take place on the server are then available to the rest of the page or can be included in place using the echo() function. Whereas the PHP and HTML were housed in separate files last time in order to clearly distinguish server-side components from client-side, the CometHiddenIFrame.php file contains all the necessary code. This was not done merely for the sake of convenience; but contributes to setting the execution order of the code. For example, at the top of the page, there is a short PHP script to send the headers to the browser. It is necessary to do this before sending any HTML to the browser, as by that time, it is too late. In fact, trying to send headers after the page content has already begun to download will result in a "Headers Already Sent" PHP error.

The `set_time_limit()` function is called with a value of `0` to turn off the maximum script execution time (which is 30 seconds by default). The headers that we are sending prevent browser caching – always important when sending dynamic content. Finally the `flush()` function is called to make sure that the header information is sent to the client:

	view plain (http://www.webreference.com/programming/javascript/rg30/index.html#) | print (http://www.webreference.com/programming/javascript/rg30/index.html#) | ? (http://www.webreference.com/programming/javascript/rg30/index.html#)

	view plain (http://www.webreference.com/programming/javascript/rg30/index.html#) | print (http://www.webreference.com/programming/javascript/rg30/index.html#) | ? (http://www.webreference.com/programming/javascript/rg30/index.html#)
1	<?php
2	set_time_limit(0);
3	header("Cache-Control: no-cache, must-revalidate");
4	header("Expires: Mon, 26 Jul 1997 05:00:00 GMT");
5	flush();
6	?>

The rest of the PHP code is contained in a script at the end of the page. It must be placed there so that the page has a chance to load, as the `do while` loop will prevent the rest of the page from displaying. The random order generation has also been moved from the `getCdCount()` function so that we can call `updateStock()` without a delay when the `Purchase` button is clicked. In fact, the `getCdCount()` function is now deprecated, and can be removed:

1	$num = $_GET['num'];
2	if ($num == "")
3	{
4	//start the update service
5	srand();
6	do
7	{
8	$newOrder = rand(1, 3);
9	$sleeptime = rand(2, 10);
10	sleep($sleeptime);
11	} while(updateStock($newOrder) > 0);
12	}
13	else

```
14 {
15     updateStock((int)$num);
16 }
```

The `updateStock()` function has been altered to return the remaining CD inventory. The loop in the main script body exits once the number falls to zero. Rather than print the number, the IFrames version echoes JavaScript code to call our `updateCount()` function in the main page, passing in the CD `$count`. Again, `flush()` is used to make sure that the string is not held in the buffer:

view plain (http://www.webreference.com/programming/javascript/rg30/2.html#) | print (http://www.webreference.com/programming/javascript/rg30/2.html#) | ? (http://www.webreference.com/programming/javascript/rg30/2.html#)

```
1  $cd_stock = (“CdCount.txt”);
2  function updateStock($num)
3  {
4      global $cd_stock;
5      $count = file($cd_stock);
6      $count = (int)$count[0];
7      $count = $count – $num;
8      if ($count < 0) $count = 0;
9      $fp = fopen($cd_stock , “w”);
10     fputs($fp , “$count”);
11     fclose($fp);
12     echo “<script type=’text/javascript’>parent.updateCount(‘” . $count . “’)</script>\n”;
13     flush();
14     return $count;
15 }
```

HTML Markup

Without an XMLHttpRequest object, we need to submit the form's contents the old-fashioned way, using a`Submit` button.
The `frmCdSales` form's `action` is set to the same file so that submitting the form will not unload the page. Equally important in this regard, is that the `target` must be set to the `name` of the hidden IFrame because it's the one that executes the scripts. The method is set to `GET` so that the `txtQty` textfield will be sent in the Request query string, as before. Its name has been changed to `num`, to match the property name that the PHP script is expecting.

There are any number of ways to stop the IFrame from displaying. Our solution is to set its `height` property to `0`. Here is all of the HTML markup:

	view plain (http://www.webreference.com/programming/javascript/rg30/2.html#) | print (http://www.webreference.com/programming/javascript/rg30/2.html#) | ? (http://www.webreference.com/programming/javascript/rg30/2.html#)
1	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd"&gt (http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd"&gt);
2	<html xmlns="http://www.w3.org/1999/xhtml"&gt (http://www.w3.org/1999/xhtml"&gt);
3	<head>
4	<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
5	<title>CD Store</title>
6	</head>
7	<body bgcolor="#FFFFFF" id=body>
8	CD Sales Page
9	<form id="frmCdSales" name="frmCdSales" action="CometHiddenIFrame.php" method="GET" target="hidden">
10	
11	<p>Quantity:
12	<input type="text" id="txtQty" name="num" size="5" maxlength="3">

	view plain (http://www.webreference.com/programming/javascript/rg30/2.html#) | print (http://www.webreference.com/programming/javascript/rg30/2.html#) | ? (http://www.webreference.com/programming/javascript/rg30/2.html#)
13	<input type="submit" id="btnSubmit" value="Purchase" name="btnSubmit">
14	<img src="rob_synch2.gif" width="122" height="122" align="middle" border="1" hspace="5">
15	
16	</form>
17	
18	There are 100 CDs left.
19	
20	
21	
22	
23	</body>

Hidden IFrame Gotchas

Although this technique is as straightforward and non-technical a way to implement Comet streaming in a Web page as you're likely to find, there are some caveats that you should be aware of.

IFrames were never meant to serve as a control for handling streaming. As such, this approach has been criticized by some as being a bit of a "hack". While it is true that there are better alternatives emerging, it can still be a suitable solution in smaller-scale applications. One reason that today's example is not a great candidate for large websites is that PHP does not scale well in terms of concurrent connections. PHP scripts are run in a separate instance for each client request, making for a potentially resource intensive implementation. For high volume utilization, languages like Python and Java Servlets are better choices, as they are designed to handle hundreds or thousands of simultaneous connections.

Another problem is that IFrames are not terribly fast-loading elements, to say the least. As the following chart indicates, IFrames just might be the slowest-loading element there is!

Element Download Times Chart (http://www.webreference.com/programming/javascript/rg30/element_download_times.gif)

There is a JavaScript issue in that IFrame's parent's onload doesn't fire until IFrame and all of its components are downloaded. In streaming Comet, that could be never! One possible workaround is to set the IFrame's `src` in JavaScript:

	view plain (http://www.webreference.com/programming/javascript/rg30/2.html#) | print (http://www.webreference.com/programming/javascript/rg30/2.html#) | ? (http://www.webreference.com/programming/javascript/rg30/2.html#)
1	
2	<script type="text/javascript">
3	document.getElementById('iframe1').src="url";
4	</script>

IFrames share the connection pool with the parent; that greatly reduces the number of other data transactions you can perform on the page. Internet Explorer 7 has a per-server connection limit of 2, whereas in IE 8 the number has been increased to 6.

Comet Streaming Technologies

There are many tools available to help you perform Comet streaming. Here is a sampling of them taken from theajaxpatterns.org (http://ajaxpatterns.org/HTTP_Streaming) site:

- **APE**: Ajax Push Engine (APE) is a complete packaged solution, designed to push real-time data in a lightweight, highly scalable and modular way, only using JavaScript for the client side. Their server supports patterns such as XHR long-polling (cross browser), forever frame and many more.
- **LivePage**: Included with Nevow, a Python-based Web application framework.
- **Jotspot Live**: A live, multi-user, wiki environment which uses HTTP Streaming to update message content.
- **Realtime on Rails**: A real-time chat application that uses Service Streaming on Firefox and Periodic Refresh on other browsers.
- **Lightstreamer**: A commercial "Push Engine" for performing large-scale HTTP Streaming.
- **Pushlets**: A Java servlet library based on HTTP Streaming, and supports either Page or Service Streaming.
- **CogStream**: Another commercial HTTP streaming solution. The Web Server runs in WinTel and Linux environments.
- **Caplin Liberator**: A free version of a commercial Comet server from Caplin Systems. Liberator can handle many thousands of concurrent users with high update rate and low message latency. Liberator is a standalone server that runs on Linux.

- **Kaazing Enterprise Comet**: Provides the industry's most productive and advanced environment for creating real-time Web applications that extend SOA's event and message delivery to the browser, allowing browsers to participate in the server-side message bus. All applications developed with Enterprise Comet are deployed to standard Java EE containers with no browser plugins.
- **RMDS2Web Server**: Migratory Data Systems RMDS2Web Server is currently the industry's most scalable real-time Web streaming solution able to serve 500,000 concurrent users with low latency from $2,000 hardware.

Additional References

- Zeitoun.net: How to implement COMET with PHP (http://www.zeitoun.net/articles/comet_and_php/start)
- Http Streaming Comet in Safari (http://stackoverflow.com/questions/169258/is-http-streaming-comet-possible-in-safari)

Here's the source for the PHP (http://www.webreference.com/programming/javascript/rg30/CometHiddenIFrameFiles.zip) file that we worked on today

Running the Demo

I used the Abyss Web Server for testing, but you can use any server that supports PHP.

To view the IFrames Streaming demo, launch your server and enter the URL to the CometHiddenIFrame.php page in the address bar; it's "http://localhost/caching/IFrames/CometHiddenIFrame.php&#8221 (http://localhost/caching/IFrames/CometHiddenIFrame.php&#8221); in my browser. You should see the number of CDs gradually decrement in steps of 1 to 3 as imaginary shoppers snatch them up. Entering a number in the `Quantity` textbox and clicking the `Purchase` button should update the number immediately.
That concludes our look at Comet and the Hidden IFrame technique. It's an easy way to add Comet streaming to small-scale Web apps.

Reference : http://www.webreference.com/programming/javascript/rg30/index.html (http://www.webreference.com/programming/javascript/rg30/index.html)

Posted by chandara's blog in HTML-CSS-JS-Ajax, JS

One thought on “Comet Programming: the Hidden IFrame Technique”

1. Pingback: *nice one*