Cross site AJAX HttpRequest problems and how to create a Proxy handler

Posted At : 19 July 2010 15:36 | Posted By : Shaun McCran
Related Categories: JQuery, Coldfusion, Json, AJAX

Most of us are familiar with the AJAX post() and getJSON() methods of remotely calling and passing around data, I use them with abundance when building applications. One thing I had not considered until recently is that all my AJAX Http requests are usually internal to that application, which also means they are on the same server and domain.

I recently jumped into a project where the application spanned 24 domains, and had been developed to use a core component library to help code re use. The problem with this arose when you are on one domain (www.domain1.com) and you want to make a request to a different domain (www.domain2.com). You encounter something called the 'same-Origin' policy.

This article deals with how to create a proxy to handle cross site AJAX http Requests.

The 'same-origin' Policy (http://en.wikipedia.org/wiki/Same_origin_policy) dictates that only scripts running on the same server can access each others methods and properties. So it basically blocks AJAX requests to 'foreign' domains.

There is no such restriction to server side http requests though, so instead of having AJAX make the request, we create a CFC proxy, that typically uses cfhttp to go get the data we want, and return it to our AJAX function.

In the code I was using to do this I am using the serialize JQuery function to suck up all the form fields and spit them at my 'foreign.cfc' (names have been changed).

I'm also passing in the name of the method that the proxy should call, and the URL of the remote url, so that it is basically a generic Proxy object.

```
<InvalidTag language="javascript">
  $(document).ready(function() {

      $('input').change(function() {

xy.cfc?method=send&remoteMethod=remoteMethodName&url=http://www.otherDomain/foreig

          $("#result").text(data);

              });
          })
      });
  </script>

<form name="searchForm" id="searchForm">
      Form fields.....
  </form>
```

Because of this I don't actually know what the arguments going into the proxy cfc are.

```
end" access="remote" output="true" returntype="query" hint="proxy handler to forei

 name="remoteMethod" type="string" required="false" hint="name of the remote metho
ument name="url" type="string" required="false" hint="url of the remote method to
            <cfset var response = "">

are generated dynamically using jquery.serialise so we don't know what they are.

    <cfhttp url="#arguments.url#" method="get">
  <cfhttpparam type="url" name="method" value="#arguments.remoteMethod#">
      <cfloop collection="#arguments#" item="item">
<cfhttpparam type="url" name="#item#" value="#arguments[item]#">
          </cfloop>
      </cfhttp>

    <cfset response = trim(cfhttp.fileContent)>
        <cfoutput>#response#</cfoutput>
        <!--- <cfreturn response> --->
      </cffunction>
```

In this way we have a re usable proxy that will hand off remote requests and pass back the data to the calling AJAX function.

Browser specific fixes

Most of the modern browsers have a browser specific work around available to them. John Resig has an interesting (slightly older) article here (http://ejohn.org/blog/cross-site-xmlhttprequest/) about how to use headers to allow cross site access to and from a domain.

Creating a browser specific fix seems like a lot of hard work, and introducing multiple blocks of browser dependant code to fix the same issue smacks of the old CSS days, and I can't really recommend it, when there are far more elegant solutions available.

Access Denied in Internet Explorer

This will also fix an erroneously reported Internet Explorer error. If Internet Explorer reports that "Access is denied" to the JQuery.js library this is in fact incorrect.

It is reporting the http status code error from the cross site http Request and mistakenly reporting that the parent object has created the access denied message, when it is in fact only the httpRequest encountering the 'same-origin' policy.

Another alternative is to use JSON-P

Not sure what JSON-P is? Ray Camden has written a great article explaining it on the O'Reilly site: http://insideria.com/2009/03/what-in-the-heck-is-jsonp-and.html

The basic premise is that you can dynamically create script blocks, and point them at whatever domain you want, in this way you can create cross site AJAX requests.

Personally I don't think this is as elegant a solution as a Proxy handler, also building a Proxy allows you to alter the data in a server side request, and supply clean data in the format you actually need, rather than introducing additional client side processing.