**Introduction**

In September 2008 Scott Guthrie, the head of the [ASP](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx).NET team, announced in [a blog post](http://weblogs.asp.net/scottgu/archive/2008/09/28/jquery-and-microsoft.aspx) that Visual Studio would be shipping with the jQuery library. He writes:

*“jQuery is a lightweight open source [JavaScript library](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx" \t "_top)*

*[http://kona.kontera.com/javascript/lib/imgs/grey_loader.gif](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx" \t "_top)*

*(only 15kb in size) that in a relatively short span of time has become one of the most popular libraries on the web. A big part of the appeal of jQuery is that it allows you to elegantly (and efficiently) find and manipulate HTML elements with minimum lines of code … There is a huge ecosystem and community built up around JQuery. The jQuery library also works well on the same page with ASP.NET AJAX and the ASP.NET AJAX Control Toolkit.”*

With that, JQuery is officially embraced by ASP.NET.

A brief introduction of JQuery

jQuery is the star among the growing list of JavaScript libraries. A few of its characteristics are light-weight, cross-browser compatibility and simplicity. A common task that sometimes takes 10 lines of code with traditional JavaScript can be accomplished with jQuery in just one line of code. For example, if you want to dress up a table with an ID mytable with alternative color for every other row, you can simple do this in jQuery.

**Listing 1: jQuery code for making a zebra-style table**

[view source](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx#viewSource)[print](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx#printSource)[?](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx#about)

1.<script>

2.$(function() {

3.    $("table#mytable tr:nth-child(even)").addClass("even");

4.});

5.</script>

**The magic dollar sign ($) and a chain of operations**

In jQuery, the most powerful character / symbol is the dollar sign. A $() function normally returns a set of objects followed by a chain of operations. An example

[view source](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx#viewSource)[print](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx#printSource)[?](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx#about)

1.$("div.test").add("p.quote").html("a little test").fadeOut();

Think of it as a long sentence with punctuations. Indeed it is a chain of instructions to tell the browser to do the following:

1. Get a div with class name is test;
2. Insert a paragraph with class name is quote;
3. Add a little text to the paragraph;
4. Operate on the DIV using a predefined method called fadeout.

So there it is, the first two basics: $() and chainable.

**jQuery Selectors**

JQuery uses CSS selectors to single out one element or a group of elements, and normally we use a combination of them to target specific elements. For example:

$(‘p.note’) returns all <p> elements whose class name is note;

$(‘p#note’) returns the <p> element whose id is note;

$(‘p’) returns all <p> elements

To select a child or children, we use the right angle bracket (>), as in $(‘p>a’) (returns all of the hyper links within the <p> element);

To select element(s) with certain attributes, we use [], as in input[type=text] (returns all text input element);

To select a container of some other elements, we use has keyword, for example: $(‘p:has(a)’) (returns all <p> elements that contains an hyperlink);

jQuery also has a position-based selector for us to select elements by position, for example $(‘p:first’)

**Document.Ready()**

The most commonly used command in jQuery is Document.Ready(). It makes sure code is executed only when a page is fully loaded. We often place code blocks inside this Document.Ready() event. For example:

[view source](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx#viewSource)[print](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx#printSource)[?](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx#about)

1.$(document).ready(function(){

2. $("#buttonTest").click(function(event){

3.   alert("I am ready!");

4. });

5.});

So far, we have brushed upon a bit of the core jQuery library. The true power of jQuery lies in its speed and flexibility and extendibility, and the ever-growing however already immense collection of jQuery plugins that deal with tasks big and small. As of today, by the tally of jQuery.com, there are 1868 plug-ins, including 100 in AJAX, 123 in Animation and effects, 66 in data, 321 in user interface. After all, JQuery is designed to be small and nimble, providing only the core functionalities required in the most common scenarios, and make others available only when needed and serve in the form of a plug-in.

**ASP .NET and JQuery**

Long gone is the era when most computing was done on a desktop, when web pages were more or less like virtual bulletin board. Now the impatient and internet-saturated generation is insatiable with substance, dynamics and connectivity. They want rich content, dazzling visual and instant feedback. More than ever, web development has become a tight coordinated dance between server and client. The server does the heavy lifting in the background, processes requests, churns up data and passes it back to the requesting client; from which point the client computer takes over. Interactions now taken place between web and user would be taken care by client side script, and web server would only be involved when client initiates a new request for data operation. The introduction and adoption of AJAX (the technique of performing asynchronous requests through client scripts) is fueled by and fuels this trend.

While AJAX is now the unifying technique across browsers (IE or Firefox), platforms (PC or MAC) and languages (C# or Java or PhP), it did not launch onto this popularity train until just a few years ago, when Google showcased its power with an array of applications. So with Google maps, Gmail, Google news, AJAX becomes the gold messenger of server and client communication. As if overnight, more than 100 [AJAX libraries](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx) sprung up to simplify and smooth this process.

jQuery is one of the more effective and light-weighted ones. ASP .NET team has come up with its own AJAX and JavaScript libraries in its unifying ambition to convert every programmer (be it C# or VB or something else) into a Microsoft faithful. However, its JavaScript library is considerably bulky, bandwidth-costly and poses a steep learning curve.

On the other hand, ASP .NET has always been predominantly a server side technology, web services and code behind page methods have always been the founding stones of web applications. The development of ASP .NET AJAX helps developers to easily make service calls from client side script.

Let’s take a look at how we can use jQuery to consume ASP .[NET web](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx" \t "_top) services and page methods.

**Consuming ASP .NET web services using jQuery**

From the conception to now, web services has gone a long way. Web services use XML as the default data exchange format and SOAP as its protocol. However it has long been dogged by complaints of complexity and lack of open standards. XML as its default message format often feels cumbersome and bandwidth-heavy.

However, with AJAX, JSON overtook XML as a more efficient alternative. It retains all of the advantages claimed by XML, such as being readable and writable for humans, and easy to parse and generate. JSON, though completely language independent, borrows a lot of conventions from languages such as C, C++, C#, Java, JavaScript, Perl, Python, etc. This makes JSON an instant hit among programmers.

Tailored to accommodate this trend, data returned from ASP .NET web script services are by default in JSON format.

**JSON serialized web service**

The following is a dummy ASP .NET web service. Please note that this service is adorned with the ScriptService attribute that makes it available to JavaScript clients.

**Listing 2: A Dummy web service**

[view source](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx#viewSource)[print](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx#printSource)[?](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx#about)

01.[WebService(Namespace = "<http://tempuri.org/>")]

02.[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1\_1)]

03.[System.Web.Script.Services.ScriptService]

04.public class dummyWebservice : System.Web.Services.WebService

05.{

06.  [WebMethod()]

07.  public string HelloToYou(string name)

08.  {

09.      return "Hello " + name;

10.  }

11.  [WebMethod()]

12.  public string sayHello()

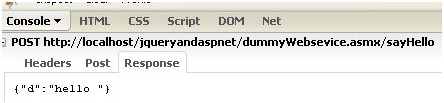
13.  {

14.      return "hello ";

15.  }

16.}

If we call the method sayHello and then use Firebug for Firefox to check the server response, it would look like this:



A call (GET or POST) to ASP .NET JSON-serialized web services must meet two criteria:

* It must carry a HTTP Content-Type header with its value set to application/json
* It must be called via the HTTP POST verb or else the request would be rejected. To submit a GET request, we just need to submit empty data string.

**Consuming a web service using ASP .NET AJAX**

It is easy to call web services with the native ASP .NET AJAX, which would automatically take care all of the gritty details. It takes two steps:

First, add a ServiceReference to the ScriptManager control in the web form, as such:

[view source](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx#viewSource)[print](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx#printSource)[?](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx#about)

1.<asp:ScriptManager ID="\_scriptManager" runat="server">

2.  <Services>

3.    <asp:ServiceReference Path="dummywebservice.asmx" />

4.  </Services>

5.</asp:ScriptManager>

Second, call any web methods defined in the web service by passing the necessary parameters and a [callback](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx) function. The callback will be invoked once the data is returned from server. The following is a complete example of how we call a web service using ASP .NET AJAX.

**Listing 3: Calling a web service using ASP .NET AJAX**

[view source](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx#viewSource)[print](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx#printSource)[?](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx#about)

01.<%@ [Page Language](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx)="C#" %>

02.<%@ Import Namespace="System.Web.Services" %>

03.<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

04."<http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd>">

05.<html  >

06.<head id="Head1" runat="server">

07.<title>ASP.NET AJAX Web Services: Web Service [Sample Page](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx)</title>

08. <script type="text/javascript"  src="<http://ajax.googleapis.com/ajax/libs/jQuery/1.2.6/jQuery.min.js>">

09.</script>

10.  <script type="text/javascript">

11.      function OnSayHelloClick() {

12.          var txtName = $get("name");

13.          dummyWebservice.HelloToYou(txtName.value, SayHello);

14.      }

15.      function SayHello(result) {

16.          alert(result);

17.      }

18.  </script>

19.</head>

20.<body>

21.   <form id="form1" runat="server">

22.    <asp:ScriptManager ID="\_scriptManager" runat="server">

23.      <Services>

24.        <asp:ServiceReference Path="dummyWebsevice.asmx" />

25.      </Services>

26.    </asp:ScriptManager>

27.    <h1> ASP.NET AJAX Web Services: Web Service Sample Page </h1>

28.     Enter your name:

29.        <input id="name" />

30.        <br />

31.        <input id="sayHelloButton" value="Say Hello"

32.               type="button" onclick="OnSayHelloClick();" />

33.    </form>

34.</body>

35.</html>

However as we have mentioned before, ASP .NET AJAX is rather heavy-handed and carries hefty performance penalties. In comparison, jQuery is superior in its promise of “less code do more”.

However how do call ASP .NET web services using jQuery?

**Consuming a web service using jQuery**

The answer: use the jQuery command ajax with the following syntax:

[view source](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx#viewSource)[print](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx#printSource)[?](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx#about)

1.$.ajax (options)

It looks deceivingly simple. However, you can stuff a lot of specifics in this umbrella parameter options, such as the required ones: url of the web service (url), request content type (contentType), response data type (dataType), callback function for success (success); and the optional ones: callback function in case of failure (error), a timeout for the AJAX request in milliseconds (timeout), etc.

For example, we can call a specific web method in a web service as such.

**Listing 4: JQuery .ajax command**

[view source](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx#viewSource)[print](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx#printSource)[?](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx#about)

1.  $.ajax({

2.  type: "POST",

3.  contentType: "application/json; charset=utf-8",

4.  url: "WebService.asmx/WebMethodName",

5.  data: "{}",

6.  dataType: "json"

7.});

Two things are worth noting in the above JQuery AJAX call. First, we must specify the contentType’s value as application/json; charset=utf-8, and the dataType as json; second, to make a GET request, we leave the data value as empty.

So put it together, the following code demonstrates how we would use JQuery to call the web service we created above.

**Listing 5: Calling a web service using jQuery**

[view source](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx#viewSource)[print](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx#printSource)[?](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx#about)

01.<%@ Page Language="C#" %>

02.<%@ Import Namespace="System.Web.Services" %>

03.<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

04."<http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd>">

05.<html  >

06.<head id="Head1" runat="server">

07.<title>ASP.NET AJAX Web Services: Web Service Sample Page</title>

08. <script type="text/javascript"  src="<http://ajax.googleapis.com/ajax/libs/jQuery/1.2.6/jQuery.min.js>">

09.</script>

10.  <script type="text/javascript">

11.      $(document).ready(function() {

12.         $("#sayHelloButton").click(function(event){

13.             $.ajax({

14.                 type: "POST",

15.                 url: "dummyWebsevice.asmx/HelloToYou",

16.                 data: "{'name': '" + $('#name').val() + "'}",

17.                 contentType: "application/json; charset=utf-8",

18.                 dataType: "json",

19.                 success: function(msg) {

20.                     AjaxSucceeded(msg);

21.                 },

22.                 error: AjaxFailed

23.             });

24.         });

25.     });

26.          function AjaxSucceeded(result) {

27.              alert(result.d);

28.          }

29.          function AjaxFailed(result) {

30.              alert(result.status + ' ' + result.statusText);

31.          }

32.  </script>

33.</head>

34.<body>

35.    <form id="form1" runat="server">

36.     <h1> Calling ASP.NET AJAX Web Services with jQuery </h1>

37.     Enter your name:

38.        <input id="name" />

39.        <br />

40.        <input id="sayHelloButton" value="Say Hello"

41.               type="button"  />

42.    </form>

43.</body>

44.</html>

**Calling an ASP .NET page method**

In terms of AJAX calls, a page method and a web service are almost interchangeable. Page methods, as the name suggests, is page dependant. Rather than creating a stand-alone web service, you can simple code a server-side method for data fetching and processing and call it from your client-side script.

Here is how.

**A dummy page method**

The following is the code sample of a dummy page method that takes no parameters

**Listing 6: A dummy hello world page method**

[view source](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx#viewSource)[print](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx#printSource)[?](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx#about)

1.[WebMethod()]

2.public static string sayHello()

3.{

4.    return "hello ";

5.}

Please note that page methods must be declared as static, meaning a page method is completely self-contained, and no page controls are accessible through the page method. For example, if you have a textbox on the web form, there is no way the page method can get or set its value. Consequently any changes with regards to the controls have no affect on the page method. It is stateless and it does not carry any of the view-state data typically carries around by an asp .NET page.

**Calling a page method from jQuery**

We use the same jQuery command $.ajax to call a page method, as shown in the following example.

[view source](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx#viewSource)[print](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx#printSource)[?](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx#about)

01.<script type="text/javascript">

02.      $(document).ready(function() {

03.          $.ajax({

04.              type: "POST",

05.              url: "pagemethod.aspx/sayHello",

06.              contentType: "application/json; charset=utf-8",

07.              data: "{}",

08.              dataType: "json",

09.              success: AjaxSucceeded,

10.              error: AjaxFailed

11.          });

12.    });

13.      function AjaxSucceeded(result) {

14.          alert(result.d);

15.      }

16.      function AjaxFailed(result) {

17.          alert(result.status + ' ' + result.statusText);

18.      }

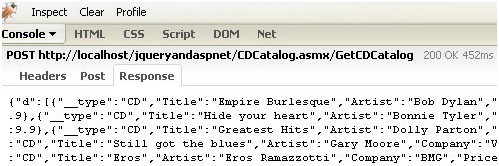
19.  </script>

**Client Templating**

In ASP .NET, [server side](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx) controls can be bound to a data source. Those controls are essentially presentation templates. Many web applications have more or less abandoned the server-centric [programming model](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx) of ASP .NET. As a result, we have to retake up the tasks of doing our own plumbing, of iterating through data and stuffing it in the right slot.

So for example, if we want to populate a table with data obtained from a web service, without a proper template, we have manually iterate through records and weave all of the data into a long string, peppered with events.

Let’s say we have a web service that returns a CD catalog, as such:



To display the data in a table, we might do the following

**Listing 7: Data presentation without a template**

[view source](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx#viewSource)[print](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx#printSource)[?](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx#about)

01.function BuildTable(msg) {

02.      var table = '<table><thead>

03.        <tr><th>Artist</th>

04.           <th>Company</th>

05.           <th>Title</th>

06.                   <th>Price</th></thead>

07.        <tbody>';

08.      for (var cd in msg) {

09.          var row = '<tr>';

10.          row += '<td>' + msg[cd].Artist + '</td>';

11.          row += '<td>' + msg[cd].Company + '</td>';

12.          row += '<td>' + msg[cd].Title + '</td>';

13.          row += '<td>' + msg[cd].Price + '</td>';

14.          row += '</tr>';

15.          table += row;

16.      }

17.      table += '</tbody></table>';

18.      $('#Container').html(table);

19.  }

**How to use jTemplate**

A much better alternative is that we use a template engine. jQuery has a few template plugins, the most widely accepted is jTemplate. jTemplate is a JQuery template engine that works with AJAX and JSON data. We can use Jtemplate in the following steps:

First, download the JQuery JavaScript file and reference it in your web page:

[view source](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx#viewSource)[print](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx#printSource)[?](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx#about)

1.<scrip src="scripts/jQuery-jtemplates.min.js" type="text/javascript"></script>

Second, define your template:

[view source](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx#viewSource)[print](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx#printSource)[?](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx#about)

01.<script type="text/html" id="TemplateResultsTable">

02.{#template MAIN}

03.<table  cellpadding="10" cellspacing="0">

04.  <tr>

05.    <th>Artist</th>

06.    <th>Company</th>

07.    <th>Title</th>

08.    <th>Price</th>

09.  </tr>

10.  {#foreach $T.d as CD}

11.    {#include ROW root=$T.CD}

12.  {#/for}

13.</table>

14.{#/template MAIN}

15.{#template ROW}

16.<tr class="{#cycle values=['','evenRow']}">

17.  <td>{$T.Artist}</td>

18.  <td>{$T.Company}</td>

19.  <td>{$T.Title}</td>

20.  <td>{$T.Price}</td>

21.</tr>

22.{#/template ROW}

23.</script>

Please note that we define the template in a script block, which can be accessed by its id; also,“jTemplates uses python like syntax and the $T is a reference to the data item passed to the template. Data can be passed in as an object and you can reference the object’s properties off this $T data item.” (Rick Strahl)

Third, call your web service in your script, designate the template and instantiate it with data.

[view source](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx#viewSource)[print](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx#printSource)[?](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx#about)

01.$(document).ready(function() {

02.          $.ajax({

03.              type: "POST",

04.              url: "CDCatalog.[asmx](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx)/GetCDCatalog",

05.              data: "{}",

06.              contentType: "application/json; charset=utf-8",

07.              dataType: "json",

08.              success: function(msg) {

09.                  //instantiate a template with data

10.                  ApplyTemplate(msg);

11.              }

12.          });

13.      });

14.function ApplyTemplate(msg) {

15.      $('#Container').setTemplate($("#TemplateResultsTable").html());

16.      $('#Container').processTemplate(msg);

17.}

**Figure 4: Using jTemplate to display a CD catalog**



**ASP .NET client templating engine**

In July 2008, the ASP .NET team also announced its own client template engine, as blogged by Bertrand Le Roy. It uses {{expression}} as place holder for data, it works with JavaScript and AJAX. The ASP .NET client template engine also allows live bindings (dynamic data refreshing and updates). For more information, please read [Using client templates, part 2: Live Bindings.](http://weblogs.asp.net/bleroy/archive/2008/09/02/using-client-templates-part-2-live-bindings.aspx)

The following is how we can populate an ASP .NET client template engine with the data we get from the above CD catalog web service.

**Listing 8: ASP .NET AJAX client templating engine**

[view source](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx#viewSource)[print](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx#printSource)[?](http://dotnetslackers.com/articles/ajax/Using-jQuery-with-ASP-NET.aspx#about)

01.<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "<http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd>">

02.<html xmlns="<http://www.w3.org/1999/xhtml>">

03.<head>

04.    <link href="template.css" rel="stylesheet" type="text/css" />

05.</head>

06.<body>

07.            <table>

08.             <thead>

09.                    <tr>

10.                    <th>Artist</th>

11.                    <th>Company</th>

12.                    <th>Title</th>

13.                    <th>Price</th>

14.                    </tr>

15.             </thead>

16.              <tbody id="cdCatalog">

17.                    <tr>

18.                    <td>{{Artist}}</td>

19.                    <td>{{Company}}</td>

20.                    <td>{{Title}}</td>

21.                    <td>{{Price}}</td>

22.                    </tr>

23.                </tbody>

24.            </table>

25.    <script type="text/javascript" src="Scripts/MicrosoftAjax.debug.js"></script>

26.    <script type="text/javascript" src="Scripts/MicrosoftAjaxTemplates.debug.js"></script>

27.    <script type="text/javascript" src="CDCatalog.asmx/js"></script>

28.    <script type="text/javascript">

29.        Sys.Application.add\_init(function() {

30.        var cdCatalog = $create(Sys.Preview.UI.DataView, {}, {}, {}, $get("cdCatalog"));

31.        CDCatalog.GetCDCatalog(function(results) {

32.        cdCatalog.set\_data(results);

33.            });

34.        });

35.        Sys.Application.initialize();

36.    </script>

37.</body>

38.</html>

As we can see, the ASP .NET client template engine is very simple and elegant. However, it is still at its early stage. The template is still lacking in flexibility and it still waits to be widely tested and accepted by the developers community.

**Summary**

With the marching forward of AJAX to the forefront of building rich interactive web sites, jQuery as the lightweight yet powerful JavaScript library also gains ever more prominence. In late 2008, Microsoft ASP .NET officially partners with jQuery to develop better and more efficient approaches to speed and smooth the communication between servers and client. Reflecting on the trend of harnessing the best of the web, this article paints in broad strokes some of the characteristics of jQuery and how we can integrate jQuery into ASP .NET