# Ajax

**AJAX stands for Asynchronous JavaScript And XML.**

**AJAX is a type of programming made popular in 2005 by Google (with Google Suggest).**

**AJAX is not a new programming language, but a new way to use existing standards.**

**With AJAX you can create better, faster, and more user-friendly web applications.**

**AJAX is based on JavaScript and HTTP requests.**

## AJAX = Asynchronous JavaScript and XML

AJAX is not a new programming language, but a technique for creating better, faster, and more interactive web applications.

With AJAX, your JavaScript can communicate directly with the server, using the JavaScript **XMLHttpRequest** object. With this object, your 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.

The AJAX technique makes Internet applications smaller, faster and more user-friendly.

 AJAX is a browser technology independent of web server software.

## AJAX is Based on Web Standards

AJAX is based on the following web standards:

- **JavaScript**
- **XML**
- **HTML**
- **CSS**

The web standards used in AJAX are well defined, and supported by all major browsers. AJAX applications are browser and platform independent.

## AJAX is About Better Internet Applications

Web applications have many benefits over desktop applications; they can reach a larger audience, they are easier to install and support, and easier to develop.

However, Internet applications are not always as "rich" and user-friendly as traditional desktop applications.

With AJAX, Internet applications can be made richer and more user-friendly.

## AJAX Uses HTTP Requests

In traditional JavaScript coding, if you want to get any information from a database or a file on the server, or send user information to a server, you will have to make an HTML form and GET or POST data to the server. The user will have to click the "Submit" button to send/get the information, wait for the server to respond, then a new page will load with the results.

Because the server returns a new page each time the user submits input, traditional web applications can run slowly and tend to be less user-friendly.

With AJAX, your JavaScript communicates directly with the server, through the JavaScript **XMLHttpRequest** object

With an HTTP request, a web page can make a request to, and get a response from a web server - without reloading the page. The user will stay on the same page, and he or she will not notice that scripts request pages, or send data to a server in the background.

## The XMLHttpRequest Object

**By using the XMLHttpRequest object, a web developer can update a page with data from the server after the page has loaded!**

AJAX was made popular in 2005 by Google (with Google Suggest).

Google Suggest is using the XMLHttpRequest object to create a very dynamic web interface: When you start typing in Google's search box, a JavaScript sends the letters off to a server and the server returns a list of suggestions.

The XMLHttpRequest object is supported in Internet Explorer 5.0+, Safari 1.2, Mozilla 1.0 / Firefox, Opera 8+, and Netscape 7.

## AJAX - Browser Support

The keystone of AJAX is the XMLHttpRequest object.

Different browsers use different methods to create the XMLHttpRequest object.

Internet Explorer uses an **ActiveXObject**, while other browsers uses the built-in JavaScript object called **XMLHttpRequest**.

To create this object, and deal with different browsers, we are going to use a "try and catch" statement. You can read more about the try and catch statement in our JavaScript tutorial.

Let's update our "testAjax.htm" file with the JavaScript that creates the XMLHttpRequest object:

```
<html>
<body>
<script type="text/javascript">
function ajaxFunction()
  {
  var xmlHttp;
  try
    {
    // Firefox, Opera 8.0+, Safari
    xmlHttp=new XMLHttpRequest();
    }
  catch (e)
```

```
    {
    // Internet Explorer
    try
      {
      xmlHttp=new ActiveXObject("Msxml2.XMLHTTP");
      }
    catch (e)
      {
      try
        {
        xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
        }
      catch (e)
        {
        alert("Your browser does not support AJAX!");
        return false;
        }
      }
    }
  }
</script>
<form name="myForm">
Name: <input type="text" name="username" />
Time: <input type="text" name="time" />
</form>
</body>
</html>
```

Example explained: First create a variable xmlHttp to hold the XMLHttpRequest object.

Then try to create the object with XMLHttp=new XMLHttpRequest(). This is for the Firefox, Opera, and Safari browsers. If that fails, try xmlHttp=new ActiveXObject("Msxml2.XMLHTTP") which is for Internet Explorer 6.0+, if that also fails, try xmlHttp=new ActiveXObject("Microsoft.XMLHTTP") which is for Internet Explorer 5.5+

If none of the three methods work, the user has a very outdated browser, and he or she will get an alert stating that the browser doesn't support AJAX.

**Note:** The browser-specific code above is long and quite complex. However, this is the code you can use every time you need to create an XMLHttpRequest object, so you can just copy and paste it whenever you need it. The code above is compatible with all the popular browsers: Internet Explorer, Opera, Firefox, and Safari.

## AJAX - More About the XMLHttpRequest Object

Before sending data to the server, we have to explain three important properties of the XMLHttpRequest object.

### The onreadystatechange Property

After a request to the server, we need a function that can receive the data that is returned by the server.

The onreadystatechange property stores the function that will process the response from a server. The following code defines an empty function and sets the onreadystatechange property at the same time:

```
xmlHttp.onreadystatechange=function()
```

```
  {
  // We are going to write some code here
  }
```

### The readyState Property

The readyState property holds the status of the server's response. Each time the readyState changes, the onreadystatechange function will be executed.

Here are the possible values for the readyState property:

| State | Description |
|---|---|
| 0 | The request is not initialized |
| 1 | The request has been set up |
| 2 | The request has been sent |
| 3 | The request is in process |
| 4 | The request is complete |

We are going to add an If statement to the onreadystatechange function to test if our response is complete (this means that we can get our data):

```
xmlHttp.onreadystatechange=function()
  {
  if(xmlHttp.readyState==4)
    {
    // Get the data from the server's response
    }
  }
```

### The responseText Property

The data sent back from the server can be retrieved with the responseText property.

In our code, we will set the value of our "time" input field equal to responseText:

```
xmlHttp.onreadystatechange=function()
  {
  if(xmlHttp.readyState==4)
    {
    document.myForm.time.value=xmlHttp.responseText;
    }
  }
```

## AJAX - Sending a Request to the Server

To send off a request to the server, we use the open() method and the send() method.

The open() method takes three arguments. The first argument defines which method to use when sending the request (GET or POST). The second argument specifies the URL of the server-side script. The third

argument specifies that the request should be handled asynchronously. The send() method sends the request off to the server. If we assume that the HTML and ASP file are in the same directory, the code would be:

```
xmlHttp.open("GET","time.asp",true);
xmlHttp.send(null);
```

Now we must decide when the AJAX function should be executed. We will let the function run "behind the scenes" when the user types something in the username text field:

```
<form name="myForm">
Name: <input type="text"
onkeyup="ajaxFunction();" name="username" />
Time: <input type="text" name="time" />
</form>
```

Our updated AJAX-ready "testAjax.htm" file now looks like this:

```
<html>
<body>
<script type="text/javascript">
function ajaxFunction()
  {
  var xmlHttp;
  try
    {
    // Firefox, Opera 8.0+, Safari
    xmlHttp=new XMLHttpRequest();
    }
  catch (e)
    {
    // Internet Explorer
    try
      {
      xmlHttp=new ActiveXObject("Msxml2.XMLHTTP");
      }
    catch (e)
      {
      try
        {
        xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
        }
      catch (e)
        {
        alert("Your browser does not support AJAX!");
        return false;
        }
      }
    }
    xmlHttp.onreadystatechange=function()
      {
      if(xmlHttp.readyState==4)
        {
        document.myForm.time.value=xmlHttp.responseText;
        }
      }
    xmlHttp.open("GET","time.asp",true);
```

```
    xmlHttp.send(null);
  }
</script>
<form name="myForm">
Name: <input type="text"
onkeyup="ajaxFunction();" name="username" />
Time: <input type="text" name="time" />
</form>
</body>
</html>
```

**AJAX - The Server-Side ASP Script**

Now we are going to create the script that displays the current server time.

The responseText property (explained in the previous chapter) will store the data returned from the server. Here we want to send back the current time. The code in "time.asp" looks like this:

```
<%
response.expires=-1
response.write(time)
%>
```

**Note:** The Expires property sets how long (in minutes) a page will be cached on a browser before it expires. If a user returns to the same page before it expires, the cached version is displayed. Response.Expires=-1 indicates that the page will never be cached.

# http://www.w3schools.com/ajax/ajax_serverscript.asp

# Microsoft Office SharePoint 2007 and ASP.NET 2.0 AJAX 1.0 Web Part

## Introduction

This article describes how to build your first SharePoint 2007 Web part which supports ASP.NET 2.0 AJAX 1.0.

## Software needed

- SharePoint Portal Server 2007 (installed with site collection created on port: 80)

- Visual Studio 2005
- Visual Studio 2005 Extensions for SharePoint 2007
- ASP.NET 2.0 AJAX 1.0

## Using the code

First you need to configure your SharePoint Portal to support AJAX. To do this, it is better to open a new AJAX web site in Visual Studio to pick a copy from configuration sections in *web.config*, or you can copy from here, so let's start:

1. Add the following part under: `<configSections>`

```
<sectionGroup name="system.web.extensions"
    type="System.Web.Configuration.SystemWebExtensionsS
ectionGroup,
    System.Web.Extensions, Version=1.0.61025.0,
Culture=neutral,
    PublicKeyToken=31bf3856ad364e35">

<sectionGroup name="scripting"
    type="System.Web.Configuration.ScriptingSectionGrou
p,
    System.Web.Extensions, Version=1.0.61025.0,
Culture=neutral,
    PublicKeyToken=31bf3856ad364e35">

      <section name="scriptResourceHandler"
      type="System.Web.Configuration.ScriptingScriptRes
ourceHandlerSection,
      System.Web.Extensions, Version=1.0.61025.0,
Culture=neutral,
      PublicKeyToken=31bf3856ad364e35"
requirePermission="false"
      allowDefinition="MachineToApplication"/>

<sectionGroup name="webServices"
    type="System.Web.Configuration.ScriptingWebServices
SectionGroup,
    System.Web.Extensions, Version=1.0.61025.0,
Culture=neutral,
    PublicKeyToken=31bf3856ad364e35">

      <section name="jsonSerialization"
```

```
      type="System.Web.Configuration.ScriptingJsonSeria
lizationSection,
      System.Web.Extensions, Version=1.0.61025.0,
Culture=neutral,
      PublicKeyToken=31bf3856ad364e35"
requirePermission="false"
      allowDefinition="Everywhere" />

      <section name="profileService"
      type="System.Web.Configuration.ScriptingProfileSe
rviceSection,
      System.Web.Extensions, Version=1.0.61025.0,
Culture=neutral,
      PublicKeyToken=31bf3856ad364e35"
requirePermission="false"
      allowDefinition="MachineToApplication" />


      <section name="authenticationService"
      type="System.Web.Configuration.ScriptingAuthentic
ationServiceSection,
      System.Web.Extensions, Version=1.0.61025.0,
Culture=neutral,
      PublicKeyToken=31bf3856ad364e35"
requirePermission="false"
      allowDefinition="MachineToApplication" />

</sectionGroup>
</sectionGroup>
</sectionGroup>
```

2. Add the following part under: <pages>

```
<controls>
       <add tagPrefix="asp" namespace="System.Web.UI"
    assembly="System.Web.Extensions,
Version=1.0.61025.0,
    Culture=neutral, PublicKeyToken=31bf3856ad364e35"
/>
</controls>
Add the following part under :
<compilation><assemblies>
<add assembly="System.Web.Extensions,
Version=1.0.61025.0,
```

```
   Culture=neutral, PublicKeyToken=31bf3856ad364e35" />
Add the following part under: <httpHandlers>
<add verb="*" path="*.asmx" validate="false"
   type="System.Web.Script.Services.ScriptHandlerFactor
y,
   System.Web.Extensions, Version=1.0.61025.0,
Culture=neutral,
   PublicKeyToken=31bf3856ad364e35"/>

<add verb="*" path="*_AppService.axd" validate="false"
   type="System.Web.Script.Services.ScriptHandlerFactor
y,
   System.Web.Extensions, Version=1.0.61025.0,
Culture=neutral,
   PublicKeyToken=31bf3856ad364e35"/>

<add verb="GET,HEAD" path="ScriptResource.axd"
   type="System.Web.Handlers.ScriptResourceHandler,
   System.Web.Extensions, Version=1.0.61025.0,
Culture=neutral,
    PublicKeyToken=31bf3856ad364e35" validate="false"/>
Add the following part under: <httpModules>
<add name="ScriptModule"
type="System.Web.Handlers.ScriptModule,
   System.Web.Extensions, Version=1.0.61025.0,
Culture=neutral,
   PublicKeyToken=31bf3856ad364e35"/>
```

3. At the end of *web.config* add the following part under: <configuration>

```
<system.web.extensions>
```

4. <scripting>

```
      <webServices>
      <!-- Uncomment this line to enable the
authentication service.
    Include requireSSL="true" if appropriate. -->
      <!--
       <authenticationService enabled="true" requireSSL
= "true|false"/>
      -->
      <!-- Uncomment these lines to enable the profile
service.
```

```
    To allow profile properties to be retrieved and
    modified in ASP.NET AJAX applications,
    you need to add each property name to the
readAccessProperties
    and writeAccessProperties attributes. -->
      <!--
      <profileService enabled="true"
                  readAccessProperties="propertyname1,p
ropertyname2"
                  writeAccessProperties="propertyname1,
propertyname2" />
      -->
      </webServices>
      <!--
      <scriptResourceHandler enableCompression="true"
                enableCaching="true" />
      -->
    </scripting>
  </system.web.extensions>
  <system.webServer>
    <validation
validateIntegratedModeConfiguration="false"/>
    <modules>
      <add name="ScriptModule"
preCondition="integratedMode"
    type="System.Web.Handlers.ScriptModule,
System.Web.Extensions,
    Version=1.0.61025.0, Culture=neutral,
    PublicKeyToken=31bf3856ad364e35"/>
    </modules>

   <handlers>
      <remove name="WebServiceHandlerFactory-
Integrated" />
      <add name="ScriptHandlerFactory" verb="*"
path="*.asmx"
    preCondition="integratedMode"
      type="System.Web.Script.Services.ScriptHandlerFac
tory,
    System.Web.Extensions, Version=1.0.61025.0,
Culture=neutral,
    PublicKeyToken=31bf3856ad364e35"/>
```

```
      <add name="ScriptHandlerFactoryAppServices"
    verb="*" path="*_AppService.axd"
preCondition="integratedMode"
    type="System.Web.Script.Services.ScriptHandlerFacto
ry,
    System.Web.Extensions, Version=1.0.61025.0,
Culture=neutral,
    PublicKeyToken=31bf3856ad364e35"/>

      <add name="ScriptResource"
preCondition="integratedMode"
    verb="GET,HEAD" path="ScriptResource.axd"
    type="System.Web.Handlers.ScriptResourceHandler,
    System.Web.Extensions, Version=1.0.61025.0,
Culture=neutral,
    PublicKeyToken=31bf3856ad364e35" />
    </handlers>
  </system.webServer>
```

5. Before closing *web.config* we should add the AJAX controls *dll* to SharePoint Safe Controls, so copy the following part under: `<SafeControls>`

```
<SafeControl Assembly="System.Web.Extensions,
Version=1.0.61025.0,
    Culture=neutral, PublicKeyToken=31bf3856ad364e35"
    Namespace="System.Web.UI" TypeName="*" Safe="True"
/>
```

6. It is time to include the AJAX script Manager to the master page. In my case, I've included the script manager control in the *default.master* located in the following path: *C:\Program Files\Common Files\Microsoft Shared\web server extensions\12\TEMPLATE\GLOBAL*

So, according to your portal template; locate the right master page file or you can open the master page from the SharePoint Designer under *_catalogs* folder. After you locate the master page file, open the file then put the following line inside the top of `<form>` tag.

```
<asp:ScriptManager runat="server" ID="ScriptManager1">
</asp:ScriptManager>
```

As shown below:

```
<form runat="server" onsubmit="return
_spFormOnSubmitWrapper();">
```

```
 <WebPartPages:SPWebPartManager id="m" runat="Server"
/>
 <asp:ScriptManager runat="server" ID="ScriptManager1">
 </asp:ScriptManager>
  <TABLE class="ms-main" CELLPADDING=0 CELLSPACING=0
BORDER=0
    WIDTH="100%" HEIGHT="100%">
```

7. Finally it is time to write our code. Open a new web part project from Visual Studio 2005, then add a reference of *System.Web.Extensions* to the project and write the following code to web part code file:

Note: There is a SharePoint Script included for changing the form action which may stop the form submission, so I included the `FixFormAction` method which will reset the form action again

```
using System;
using System.Runtime.InteropServices;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Serialization;
using Microsoft.SharePoint;
using Microsoft.SharePoint.WebControls;
using Microsoft.SharePoint.WebPartPages;

namespace AjaxWebPart
{
    [Guid("733ee261-6e34-49cf-ae29-e8aeb4df4563")]
    public class AjaxWebPart :
System.Web.UI.WebControls.WebParts.WebPart
    {
        public AjaxWebPart()
        {
            this.ExportMode = WebPartExportMode.All;
        }
        // ASP.NET Controls declaration
        private Label label;
        private Label label2;
        private Label label3;

        private TextBox textBox1;
        private TextBox textBox2;
```

```
        // ASP.NET AJAX Controls declaration
        protected UpdatePanel udatePanel;
        protected UpdateProgress updateProgress;

        protected override void CreateChildControls()
        {

            base.CreateChildControls();

            // Fix Form Action
            this.FixFormAction();

            udatePanel = new UpdatePanel();
            updateProgress = new UpdateProgress();

            udatePanel.ID = "_UpdatePanel";
            updateProgress.ID = "_UpdateProgress";

            //Create Update Progress Template
            string templateHTML =
            "<div><img alt=
            \"Loading...\"
src=\"/_layouts/images/loader.gif\"/>
                Loading...</div>";

            updateProgress.ProgressTemplate =
            new ProgressTemplate(templateHTML);

            updateProgress.AssociatedUpdatePanelID =
udatePanel.ClientID;

            udatePanel.UpdateMode =
UpdatePanelUpdateMode.Conditional;

            this.Controls.Add(udatePanel);

            this.label = new Label();
            this.label2 = new Label();
            this.label3 = new Label();

            this.label.Text = "Enter 1st Number: ";
            this.label2.Text = "Enter 2nd Number: ";
```

```
            this.textBox1 = new TextBox();
            this.textBox1.ID = "TextBox1";

            this.textBox2 = new TextBox();
            this.textBox2.ID = "TextBox2";

           //Adding Controls
            udatePanel.ContentTemplateContainer.Control
s.Add(this.label);
            udatePanel.ContentTemplateContainer.Control
s.Add
                        (this.textBox1);
            udatePanel.ContentTemplateContainer.Control
s.Add
                    (new LiteralControl("<br />"));

            udatePanel.ContentTemplateContainer.Control
s.Add(this.label2);
            udatePanel.ContentTemplateContainer.Control
s.Add
                            (this.textBox2);

            udatePanel.ContentTemplateContainer.Control
s.Add
                    (new LiteralControl("<br /><br
/>"));

            Button button = new Button();
            button.Text = "Calculate";

            button.Click += new
EventHandler(HandleButtonClick);
            udatePanel.ContentTemplateContainer.Control
s.Add(button);

            udatePanel.ContentTemplateContainer.Control
s.Add
                (new LiteralControl("  "));

            udatePanel.ContentTemplateContainer.Control
s.Add(this.label3);
```

```
            udatePanel.ContentTemplateContainer.Control
s.Add
                                (updateProgress);
        }

        private void HandleButtonClick(object sender,
EventArgs eventArgs)
        {
            //Just wait to see the progress loader
working
            System.Threading.Thread.Sleep(1000);

            this.label3.Text =
Convert.ToString(int.Parse(textBox1.Text)
            + int.Parse(textBox2.Text));

        }

        //Fixing Form Action
        private void FixFormAction ()
        {
            if (this.Page.Form != null)
            {
                string formOnSubmitAtt =
            this.Page.Form.Attributes["onsubmit"];
                if(formOnSubmitAtt == "return
_spFormOnSubmitWrapper();")
                {
                    this.Page.Form.Attributes["onsubmit
"] =
                    "_spFormOnSubmitWrapper();";
                }
            }
            ScriptManager.RegisterStartupScript
        (this, typeof(AjaxWebPart), "UpdatePanelFixup",
        "_spOriginalFormAction =
document.forms[0].action;
        _spSuppressFormOnSubmitWrapper=true;", true);
        }
    }

    //Class for Building progress tempales
    public class ProgressTemplate : ITemplate
```

```
        {
            private string template;

            public ProgressTemplate(string temp)
            {
                template = temp;
            }

            public void InstantiateIn(Control container)
            {
                LiteralControl ltr = new
    LiteralControl(this.template);
                container.Controls.Add(ltr);
            }
        }
    }
```

Then build the solution and deploy the web part from Visual Studio to your portal. As you see, it is a simple web part to calculate the summation of two integers.

**http://www.codeproject.com/spoint/MossAjaxWebPart.asp?df=100&forumid=417786&exp=0&select=2045182**

# Download  Ajax S/Ws :

1.Ajax Beta version (dll)

http://ajax.schwarz-interactive.de/CSharpSample/

2. Microsoft Asp .Net 2.0 Ajax Extensions

**http://asp.net/ajax/Default.aspx(Ajax Downloads)**

# Other Ajax Related links

****http://www.dotnet-friends.com/articles/web/asp/csharp/62167d87-d146-4948-8c02-cc41d31946fe.aspx**

**http://weblogs.asp.net/jan/archive/2007/02/26/new-version-of-smartpart-now-with-ajax-connections.aspx (Communication through smart part)**

**http://www.codeproject.com/Ajax/IntroAjaxASPNET.asp**

****http://en.wikipedia.org/wiki/Ajax_(programming)**

****http://www.w3schools.com/ajax/ajax_xmlfile.asp**

**http://dev2dev.bea.com/pub/a/2006/01/ajax-portal-1.html**

** **http://dotnet.org.za/zlatan/archive/2007/10/12/developing-ajax-web-parts-in-sharepoint-2007.aspx**

**http://www.codeproject.com/Ajax/IntroAjaxASPNET.asp**

****http://www.codeproject.com/useritems/Ajax_Chat.asp**

****http://sharepoint.microsoft.com/blogs/mike/Lists/Posts/Post.aspx?ID=3 (with the steps)**

**http://msdn2.microsoft.com/en-us/library/aa479042.aspx**