This page is specific to

Microsoft Visual Studio 2010/.NET Framework 4

**Ajax Overview**

When you build an Ajax application, you provide your users with a richer and more responsive user experience. You can use Ajax features to enhance server-based ASP.NET Web Forms applications by using server-side ASP.NET controls, such as the [UpdatePanel](http://msdn.microsoft.com/en-us/library/system.web.ui.updatepanel.aspx) control.

|  |
| --- |
| **Description: NoteNote** |
| If you want to create Ajax client-side interactivity without using server controls, use jQuery. In Visual Studio, a jQuery library is included in the application when you create a project by using the **ASP.NET Web Site** project template, **ASP.NET Web Application** project template, and the **ASP.NET MVC 2 Web Application** project template. When you develop using jQuery, Visual Studio provides IntelliSense, and the Microsoft Content Delivery Network (CDN) provides downloadable versions of the jQuery library for efficient deployment. For more information, see the [jQuery Web site](http://www.jquery.com/). |

This topic focuses on enhancing server-based ASP.NET Web Forms applications.

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifWhy Use Microsoft Ajax Features?

Web Forms applications that use Ajax features offer the following features:

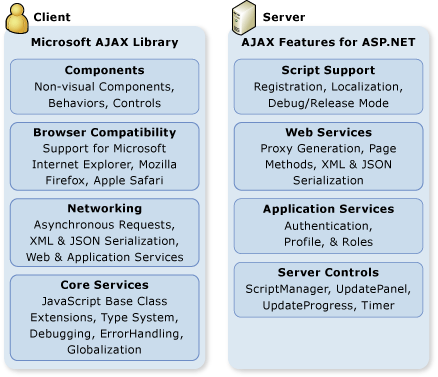
* Familiar interactive UI elements such as progress indicators, tooltips, and pop-up windows.
* Improved efficiency for Web Forms application, because significant parts of a Web page's processing can be performed in the browser.
* Partial-page updates that refresh only the parts of the Web page that have changed.
* Client integration with ASP.NET application services for forms authentication, roles, and user profiles.
* Auto-generated proxy classes that simplify calling Web service methods from client script.
* The ability to customize server controls to include client capabilities.
* Support for the most popular browsers, which includes Microsoft Internet Explorer, Mozilla Firefox, and Apple Safari.

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifArchitecture of Microsoft Ajax Applications

A Microsoft Ajax Web application consists of either a client-only solution or a client and server solution. A client-only solution uses Microsoft Ajax Library but does not use any ASP.NET server controls. For instance, an HTML can include **script** elements that reference the Microsoft Ajax Library .js files. The Microsoft Ajax Library allows Ajax applications to perform all processing on the client. A client and server solution consists of using both the Microsoft Ajax Library and ASP.NET server controls.

The following illustration shows the functionality that is included in the client-script libraries and server components that are included with .NET Framework 4.

**Microsoft Ajax client and server architecture**



The illustration shows the functionality of the client-based Microsoft Ajax Library, which includes support for creating client components, browser compatibility, and networking and core services. The illustration also shows the functionality of server-based Microsoft Ajax features, which include script support, Web services, application services, and server controls. The following sections describe the illustration in more detail.

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifMicrosoft Ajax Client Architecture

The client architecture includes libraries for component support, browser compatibility, networking, and core services.

**Components**

Client components enable rich behaviors in the browser without postbacks. Components fall into three categories:

* Components, which are non-visual objects that encapsulate code.
* Behaviors, which extend the behavior of existing DOM elements.
* Controls, which represent a new DOM element that has custom behavior.

The type of component that you use depends on the type of client behavior you want. For example, a watermark for an existing text box can be created by using a behavior that is attached to the text box. For more information, see [Creating Client Components and Controls](http://msdn.microsoft.com/en-us/library/bb386565.aspx).

**Browser Compatibility**

The browser compatibility layer provides Microsoft Ajax scripting compatibility for the most frequently used browsers (including Microsoft Internet Explorer, Mozilla Firefox, and Apple Safari). This enables you to write the same script regardless of which supported browser you are targeting. For more information, see [Browser Security Settings for Ajax Enabled ASP.NET Pages](http://msdn.microsoft.com/en-us/library/bb470452.aspx).

**Networking**

The networking layer handles communication between script in the browser and Web-based services and applications. It also manages asynchronous remote method calls. In many scenarios, such as partial-page updates that use the [UpdatePanel](http://msdn.microsoft.com/en-us/library/system.web.ui.updatepanel.aspx) control, the networking layer is used automatically and does not require that you write any code.

The networking layer also provides support for accessing server-based forms authentication, role information, and profile information in client script. This support is also available to Web applications that are not created by using ASP.NET, as long as the application has access to the Microsoft Ajax Library.

**Core Services**

The Ajax client-script libraries in ASP.NET consist of JavaScript (.js) files that provide features for object-oriented development. The object-oriented features included in the Microsoft Ajax client-script libraries enable a high level of consistency and modularity in client scripting. The following core services are part of the client architecture:

* Object-oriented extensions to JavaScript, such as classes, namespaces, event handling, inheritance, data types, and object serialization.
* A base class library, which includes components such as string builders and extended error handling.
* Support for JavaScript libraries that are either embedded in an assembly or are provided as standalone JavaScript (.js) files. Embedding JavaScript libraries in an assembly can make it easier to deploy applications and can help solve versioning issues.
* For more information, see [Creating Custom Client Script by Using the Microsoft Ajax Library](http://msdn.microsoft.com/en-us/library/bb386453.aspx).

**Debugging and Error Handling**

The core services include the [Sys.Debug](http://msdn.microsoft.com/en-us/library/bb397422.aspx) class, which provides methods for displaying objects in readable form at the end of a Web page. The class also shows trace messages, enables you to use assertions, and lets you break into the debugger. An extended [Error](http://msdn.microsoft.com/en-us/library/t9zk6eay.aspx) object API provides helpful exception details with support for release and debug modes. For more information, see [Debugging and Tracing Ajax Applications Overview](http://msdn.microsoft.com/en-us/library/bb398817.aspx).

**Globalization**

The Ajax server and client architecture in ASP.NET provides a model for localizing and globalizing client script. This enables you to design applications that use a single code base to provide UI for many locales (languages and cultures). For example, the Ajax architecture enables JavaScript code to format **Date** or **Number** objects automatically according to culture settings of the user's browser, without requiring a postback to the server. For more information, see [Walkthrough: Globalizing a Date by Using Client Script](http://msdn.microsoft.com/en-us/library/bb386581.aspx).

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifAjax Server Architecture

The server pieces that support Ajax development consist of ASP.NET Web server controls and components that manage the UI and flow of an application. The server pieces also manage serialization, validation, and control extensibility. There are also ASP.NET Web services that enable you to access ASP.NET application services for forms authentication, roles, and user profiles.

**Script Support**

Ajax features in ASP.NET are commonly implemented by using client script libraries that perform processing strictly on the client. You can also implement Ajax features by using server controls that support scripts sent from the server to the client.

You can also create custom client script for your ASP.NET applications. In that case, you can also use Ajax features to manage your custom script as static .js files (on disk) or as .js files embedded as resources in an assembly.

Ajax features include a model for release and debug modes. Release mode provides error checking and exception handling that is optimized for performance, with minimized script size. Debug mode provides more robust debugging features, such as type and argument checking. ASP.NET runs the debug versions when the application is in debug mode. This enables you to throw exceptions in debug scripts while minimizing the size of release code.

Script support for Ajax in ASP.NET is used to provide two important features:

* The Microsoft Ajax Library, which is a type system and a set of JavaScript extensions that provide namespaces, inheritance, interfaces, enumerations, reflection, and additional features. For more information see [Creating Custom Client Script by Using the Microsoft Ajax Library](http://msdn.microsoft.com/en-us/library/bb386453.aspx).
* Partial-page rendering, which updates regions of the page by using an asynchronous postback. For more information, see [Partial-Page Rendering Overview](http://msdn.microsoft.com/en-us/library/bb386573.aspx).

**Localization**

The Microsoft Ajax architecture builds on the foundation of the ASP.NET 2.0 localization model. It provides additional support for localized .js files that are embedded in an assembly or that are provided on disk. ASP.NET can serve localized client scripts and resources automatically for specific languages and regions.

For more information, see the following topics:

* [Localizing Resources for Component Libraries Overview](http://msdn.microsoft.com/en-us/library/bb398937.aspx)
* [Walkthrough: Embedding Localized Resources for a JavaScript File](http://msdn.microsoft.com/en-us/library/bb398868.aspx)

**Web Services**

With Ajax functionality in an ASP.NET Web page, you can use client script to call both ASP.NET Web services (.asmx) and Windows Communication Foundation (WCF) services (.svc). The required script references are automatically added to the page, and they in turn automatically generate the Web service proxy classes that you use from client script to call the Web service.

You can also access ASP.NET Web services without using Microsoft Ajax server controls (for example, if you are using a different Web development environment). To do so, in the page, you can manually include references to the Microsoft Ajax Library, to script files, and to the Web service itself. At run time, ASP.NET generates the proxy classes that you can use to call the services. For more information, see [ASP.NET Web Services in Ajax](http://msdn.microsoft.com/en-us/library/bb398785.aspx).

**Application Services**

Application services in ASP.NET are built-in Web services that are based on ASP.NET forms authentication, roles, and user profiles. These services can be called by client script in an Ajax-enabled Web page, by a Windows client application, or by a WCF-compatible client. For information and examples, see. [ASP.NET Web Services](http://msdn.microsoft.com/en-us/library/t745kdsh.aspx).

**Server Controls**

Ajax server controls consist of server and client code that integrate to produce rich client behavior. When you add an Ajax-enabled control to an ASP.NET Web page, the page automatically sends supporting client script to the browser for Ajax functionality. You can provide additional client code to customize the functionality of a control, but this is not required.

The following list describes the most frequently used Ajax server controls.

|  |  |
| --- | --- |
| Control | Description |
| [ScriptManager](http://msdn.microsoft.com/en-us/library/system.web.ui.scriptmanager.aspx) | Manages script resources for client components, partial-page rendering, localization, globalization, and custom user scripts. The [ScriptManager](http://msdn.microsoft.com/en-us/library/system.web.ui.scriptmanager.aspx) control is required in order to use the [UpdatePanel](http://msdn.microsoft.com/en-us/library/system.web.ui.updatepanel.aspx), [UpdateProgress](http://msdn.microsoft.com/en-us/library/system.web.ui.updateprogress.aspx), and [Timer](http://msdn.microsoft.com/en-us/library/system.web.ui.timer.aspx) controls. However, the [ScriptManager](http://msdn.microsoft.com/en-us/library/system.web.ui.scriptmanager.aspx) control is not required when creating a client-only solution. |
| [UpdatePanel](http://msdn.microsoft.com/en-us/library/system.web.ui.updatepanel.aspx) | Enables you to refresh selected parts of the page, instead of refreshing the whole page by using a synchronous postback. |
| [UpdateProgress](http://msdn.microsoft.com/en-us/library/system.web.ui.updateprogress.aspx) | Provides status information about partial-page updates in [UpdatePanel](http://msdn.microsoft.com/en-us/library/system.web.ui.updatepanel.aspx) controls. |
| [Timer](http://msdn.microsoft.com/en-us/library/system.web.ui.timer.aspx) | Performs postbacks at defined intervals. You can use the [Timer](http://msdn.microsoft.com/en-us/library/system.web.ui.timer.aspx) control to post the whole page, or use it together with the [UpdatePanel](http://msdn.microsoft.com/en-us/library/system.web.ui.updatepanel.aspx) control to perform partial-page updates at a defined interval. |

You can also create custom ASP.NET server controls that include Ajax client behaviors. Custom controls that enhance the capabilities of other ASP.NET Web controls are referred to as *extender* *control*s. For more information, see [Adding Ajax Functionality to ASP.NET Controls](http://msdn.microsoft.com/en-us/library/bb398782.aspx) and [Microsoft Ajax Extender Controls Overview](http://msdn.microsoft.com/en-us/library/bb470384.aspx).

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifAjax Control Toolkit

The [Ajax Control Toolkit](http://go.microsoft.com/fwlink/?LinkId=187212) contains controls that you can use to build highly responsive and interactive Ajax-enabled Web applications. These controls do not require knowledge of JavaScript or Ajax. They are designed using concepts that are familiar to ASP.NET Web Forms application developers. Using the Ajax Control Toolkit, you can build Ajax-enabled ASP.NET Web Forms applications and ASP.NET MVC Web applications by dragging the controls from the Visual Studio **Toolbox** onto a page. The Ajax Control Toolkit is an open-source project that is part of the CodePlex Foundation. For more information, see the [Ajax Control Toolkit](http://go.microsoft.com/fwlink/?LinkId=187212) on the CodePlex Web site.

|  |
| --- |
| **Description: NoteNote** |
| If you want to create Ajax client-side interactivity without using server controls, use jQuery. For more information, see the [jQuery Web site](http://www.jquery.com/). |