**What's New in ASP.NET 4 and Visual Web Developer**

**.NET Framework 4**

[Other Versions](javascript:;)

http://i.msdn.microsoft.com/Areas/Epx/Content/Images/ImageSprite.png

* [Visual Studio 2008](http://msdn.microsoft.com/en-us/library/s57a598e(d=printer,v=vs.90).aspx)
* [Visual Studio 2005](http://msdn.microsoft.com/en-us/library/s57a598e(d=printer,v=vs.80).aspx)

The .NET Framework version 4 includes enhancements for ASP.NET 4 in targeted areas. Visual Studio 2010 and Microsoft Visual Web Developer Express also include enhancements and new features for improved Web development. This document provides an overview of many of the new features that are included in the upcoming release.

This topic contains the following sections:

* [ASP.NET Core Services](http://msdn.microsoft.com/en-us/library/s57a598e(d=printer).aspx#aspnet_core_services)
* [ASP.NET Web Forms](http://msdn.microsoft.com/en-us/library/s57a598e(d=printer).aspx#aspnet_web_forms)
* [ASP.NET MVC](http://msdn.microsoft.com/en-us/library/s57a598e(d=printer).aspx#aspnet_mvc)
* [Dynamic Data](http://msdn.microsoft.com/en-us/library/s57a598e(d=printer).aspx#dynamic_data)
* [ASP.NET Chart Control](http://msdn.microsoft.com/en-us/library/s57a598e(d=printer).aspx#aspnet_chart_control)
* [Visual Web Developer Enhancements](http://msdn.microsoft.com/en-us/library/s57a598e(d=printer).aspx#visual_web_developer_enhancements)
* [Web Application Deployment with Visual Studio 2010](http://msdn.microsoft.com/en-us/library/s57a598e(d=printer).aspx#web_application_deployment_with_visual_studio_2010)
* [Enhancements to ASP.NET Multi-Targeting](http://msdn.microsoft.com/en-us/library/s57a598e(d=printer).aspx#enhancements_to_aspnet_multitargeting)

[ASP.NET Core Services](javascript:void(0))

ASP.NET 4 introduces many features that improve core ASP.NET services such as output caching and session state storage.

**Web.config File Refactoring**

The Web.config file that contains configuration information for a Web application has grown considerably over the past few releases of the .NET Framework as new features have been added. In .the .NET Framework 4, the major configuration elements have been moved to the machine.config file, and applications now inherit these settings. This allows the Web.config file in ASP.NET 4 applications to be empty or to specify only which version of the framework the application is targeting, as shown in the following example:

<?xml version="1.0"?>

<configuration>

<system.web>

<compilation targetFramework="4.0" />

</system.web>

</configuration>

**Extensible Output Caching**

Since the time that ASP.NET 1.0 was released, output caching has enabled developers to store the generated output of pages, controls, and HTTP responses in memory. On subsequent Web requests, ASP.NET can serve content more quickly by retrieving the generated output from memory instead of regenerating the output from scratch. However, this approach has a limitation — generated content always has to be stored in memory. On servers that experience heavy traffic, the memory requirements for output caching can compete with memory requirements for other parts of a Web application.

ASP.NET 4 adds extensibility to output caching that enables you to configure one or more custom output-cache providers. Output-cache providers can use any storage mechanism to persist HTML content. These storage options can include local or remote disks, cloud storage, and distributed cache engines.

Output-cache provider extensibility in ASP.NET 4 lets you design more aggressive and more intelligent output-caching strategies for Web sites. For example, you can create an output-cache provider that caches the "Top 10" pages of a site in memory, while caching pages that get lower traffic on disk. Alternatively, you can cache every vary-by combination for a rendered page, but use a distributed cache so that the memory consumption is offloaded from front-end Web servers.

You create a custom output-cache provider as a class that derives from the [OutputCacheProvider](http://msdn.microsoft.com/en-us/library/system.web.caching.outputcacheprovider.aspx) type. You can then configure the provider in the Web.config file by using the new providers subsection of the outputCache element

For more information and for examples that show how to configure the output cache, see [outputCache Element for caching (ASP.NET Settings Schema)](http://msdn.microsoft.com/en-us/library/ms228124.aspx). For more information about the classes that support caching, see the documentation for the [OutputCache](http://msdn.microsoft.com/en-us/library/system.web.caching.outputcache.aspx) and [OutputCacheProvider](http://msdn.microsoft.com/en-us/library/system.web.caching.outputcacheprovider.aspx) classes.

By default, in ASP.NET 4, all HTTP responses, rendered pages, and controls use the in-memory output cache. The defaultProvider attribute for ASP.NET is AspNetInternalProvider. You can change the default output-cache provider used for a Web application by specifying a different provider name for defaultProvider attribute.

In addition, you can select different output-cache providers for individual control and for individual requests and programmatically specify which provider to use. For more information, see the [HttpApplication.GetOutputCacheProviderName(HttpContext)](http://msdn.microsoft.com/en-us/library/system.web.httpapplication.getoutputcacheprovidername.aspx) method. The easiest way to choose a different output-cache provider for different Web user controls is to do so declaratively by using the new providerName attribute in a page or control directive, as shown in the following example:

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_0ed06265-1055-46d7-8261-176115b01c06');)

<%@ OutputCache Duration="60" VaryByParam="None"

providerName="DiskCache" %>

**Auto-Start Web Applications**

Some Web applications must load large amounts of data or must perform expensive initialization processing before serving the first request. In earlier versions of ASP.NET, for these situations you had to devise custom approaches to "wake up" an ASP.NET application and then run initialization code during the Application\_Load method in the Global.asax file.

To address this scenario, a new auto-start feature is available when ASP.NET 4 runs on IIS 7.5 on Windows Server 2008 R2. The feature provides a controlled approach for starting up an application pool, initializing an ASP.NET application, and then accepting HTTP requests. It lets you perform expensive application initialization prior to processing the first HTTP request.

For more information about the auto-start feature, see the [What's New for ASP.NET 4 White Paper](http://go.microsoft.com/fwlink/?LinkId=147805).

**Permanently Redirecting a Page**

Content in Web applications is often moved over the lifetime of the application. This can lead to links to be out of date, such as the links that are returned by search engines.

In ASP.NET, developers have traditionally handled requests to old URLs by using the [Redirect](http://msdn.microsoft.com/en-us/library/system.web.httpresponse.redirect.aspx) method to forward a request to the new URL. However, the [Redirect](http://msdn.microsoft.com/en-us/library/system.web.httpresponse.redirect.aspx) method issues an HTTP 302 (Found) response (which is used for a temporary redirect). This results in an extra HTTP round trip.

ASP.NET 4 adds a [RedirectPermanent](http://msdn.microsoft.com/en-us/library/system.web.httpresponse.redirectpermanent.aspx) helper method that makes it easy to issue HTTP 301 (Moved Permanently) responses, as in the following example:

RedirectPermanent("/newpath/foroldcontent.aspx");

Search engines and other user agents that recognize permanent redirects will store the new URL that is associated with the content, which eliminates the unnecessary round trip made by the browser for temporary redirects.

**Session State Compression**

By default, ASP.NET provides two options for storing session state across a Web farm. The first option is a session state provider that invokes an out-of-process session state server. The second option is a session state provider that stores data in a Microsoft SQL Server database.

Because both options store state information outside a Web application's worker process, session state has to be serialized before it is sent to remote storage. If a large amount of data is saved in session state, the size of the serialized data can become very large.

ASP.NET 4 introduces a new compression option for both kinds of out-of-process session state providers. By using this option, applications that have spare CPU cycles on Web servers can achieve substantial reductions in the size of serialized session state data.

You can set this option using the new compressionEnabled attribute of the [sessionState](http://msdn.microsoft.com/en-us/library/h6bb9cz9.aspx) element in the configuration file. When the compressionEnabled configuration option is set to true, ASP.NET compresses (and decompresses) serialized session state by using the .NET Framework [GZipStream](http://msdn.microsoft.com/en-us/library/system.io.compression.gzipstream.aspx) class.

**Expanding the Range of Allowable URLs**

ASP.NET 4 introduces new options for expanding the size of application URLs. Previous versions of ASP.NET constrained URL path lengths to 260 characters, based on the NTFS file-path limit. In ASP.NET 4, you have the option to increase (or decrease) this limit as appropriate for your applications, using two new attributes of the httpRuntime configuration element. The following example shows these new attributes.

<httpRuntime maxRequestPathLength="260" maxQueryStringLength="2048" />

ASP.NET 4 also enables you to configure the characters that are used by the URL character check. When ASP.NET finds an invalid character in the path portion of a URL, it rejects the request and issues an HTTP 400 (Bad request) status code. In previous versions of ASP.NET, the URL character checks were limited to a fixed set of characters. In ASP.NET 4, you can customize the set of valid characters using the new requestPathInvalidChars attribute of the httpRuntime configuration element, as shown in the following example:

<httpRuntime requestPathInvalidChars="&lt;,&gt;,\*,%,&amp;,:,\,?" />

(In the string that is assigned to requestPathInvalidChars by default, the less than (<), greater than (>), and ampersand (&) characters are encoded, because the Web.config file is an XML file.) By default, the requestPathInvalidChars attribute defines eight characters as invalid.

|  |
| --- |
| **NoteNote** |
| ASP.NET 4 always rejects URL paths that contain characters in the ASCII range of 0x00 to 0x1F, because those are invalid URL characters as defined in RFC 2396 of the IETF (see <http://www.ietf.org/rfc/rfc2396.txt> on the IETF Web site). On versions of Windows Server that run IIS 6 or higher, the http.sys protocol device driver automatically rejects URLs that contain these characters. |

[Extensible Request Validation](javascript:void(0))

ASP.NET request validation searches incoming HTTP request data for strings that are typically used in cross-site scripting (XSS) attacks. If potential XSS strings are found, request validation flags the suspect string and returns an error. The built-in request validation returns an error only when it finds the most common strings that are used in XSS attacks. Previous attempts to make the XSS validation more aggressive resulted in too many false positives. However, you might want request validation that is more aggressive. Conversely, you might want to intentionally relax XSS checks for specific pages or for specific types of requests.

In ASP.NET 4, the request validation feature has been made extensible so that you can use custom request-validation logic. To extend request validation, you create a class that derives from the new [System.Web.Util.RequestValidator](http://msdn.microsoft.com/en-us/library/system.web.util.requestvalidator.aspx) class, and you configure the application to use the custom type (in the httpRuntime section of the Web.config file). For more information, see [System.Web.Util.RequestValidator](http://msdn.microsoft.com/en-us/library/system.web.util.requestvalidator.aspx).

**Object Caching and Object Caching Extensibility**

Since its first release, ASP.NET has included a powerful in-memory object cache ([System.Web.Caching.Cache](http://msdn.microsoft.com/en-us/library/system.web.caching.cache.aspx)). The cache implementation has been so popular that it has been used in non-Web applications. However, it is awkward for a Windows Forms or WPF application to include a reference to System.Web.dll just to be able to use the ASP.NET object cache. To make caching available for all applications, the .NET Framework 4 introduces a new assembly, a new namespace, some base types, and a concrete caching implementation. The new System.Runtime.Caching.dll assembly contains a new caching API in the [System.Runtime.Caching](http://msdn.microsoft.com/en-us/library/system.runtime.caching.aspx) namespace. The namespace contains two core sets of classes:

* Abstract types that provide the foundation for building any type of custom cache implementation.
* A concrete in-memory object cache implementation (the [MemoryCache](http://msdn.microsoft.com/en-us/library/system.runtime.caching.memorycache.aspx) class).

The new [MemoryCache](http://msdn.microsoft.com/en-us/library/system.runtime.caching.memorycache.aspx) class is modeled closely on the ASP.NET cache, and it shares much of the internal cache engine logic with ASP.NET. Although the public caching APIs in the [System.Runtime.Caching](http://msdn.microsoft.com/en-us/library/system.runtime.caching.aspx) namespace have been updated to support development of custom caches, if you have used the ASP.NET Cache object, you will find familiar concepts in the new APIs.

**Extensible HTML, URL, and HTTP Header Encoding**

In ASP.NET 4, you can create custom encoding routines for the following common text-encoding tasks:

* HTML encoding.
* URL encoding.
* HTML attribute encoding.
* Encoding outbound HTTP headers.

You can create a custom encoder by deriving from the new [System.Web.Util.HttpEncoder](http://msdn.microsoft.com/en-us/library/system.web.util.httpencoder.aspx) type and then configuring ASP.NET to use the custom type in the httpRuntime section of the Web.config file, as shown in the following example:

<httpRuntime encoderType="Samples.MyCustomEncoder, Samples" />

After a custom encoder has been configured, ASP.NET automatically calls the custom encoding implementation whenever public encoding methods of the [HttpUtility](http://msdn.microsoft.com/en-us/library/system.web.httputility.aspx) or [HttpServerUtility](http://msdn.microsoft.com/en-us/library/system.web.httpserverutility.aspx) classes are called. This lets one part of a Web development team create a custom encoder that implements aggressive character encoding, while the rest of the Web development team continues to use the public ASP.NET encoding APIs. By centrally configuring a custom encoder in the httpRuntime element, you are guaranteed that all text-encoding calls from the public ASP.NET encoding APIs are routed through the custom encoder.

**Performance Monitoring for Individual Applications in a Single Worker Process**

In order to increase the number of Web sites that can be hosted on a single server, many hosters run multiple ASP.NET applications in a single worker process. However, if multiple applications use a single shared worker process, it is difficult for server administrators to identify an individual application that is experiencing problems.

ASP.NET 4 leverages new resource-monitoring functionality introduced by the CLR. To enable this functionality, you can add the following XML configuration snippet to the aspnet.config configuration file.

<?xml version="1.0" encoding="UTF-8" ?>

<configuration>

<runtime>

<appDomainResourceMonitoring enabled="true"/>

</runtime>

</configuration>

(The aspnet.config file is in the directory where the .NET Framework is installed. It is not the application Web.config file.) When the appDomainResourceMonitoring feature has been enabled, two new performance counters are available in the "ASP.NET Applications" performance category: % Managed Processor Time and Managed Memory Used. Both of these performance counters use the new CLR application-domain resource management feature to track estimated CPU time and managed memory utilization of individual ASP.NET applications. As a result, with ASP.NET 4, administrators now have a more granular view into the resource consumption of individual applications running in a single worker process.

**jQuery Included with Web Forms and MVC**

The Visual Studio templates for both Web Forms and MVC include the open-source jQuery library. When you create a new web site or project, a Scripts folder is created that contains following files:

* jQuery-1.4.1.js – The human-readable, unminified version of the jQuery library. (Minification is the practice of removing unnecessary characters from code to reduce its size and thereby improve load and execution times.)
* jQuery-14.1.min.js – The minified version of the jQuery library.
* jQuery-1.4.1-vsdoc.js – The IntelliSense documentation file for the jQuery library.

Include the unminified version of jQuery while you are developing an application. Include the minified version of jQuery for production applications.

**Content Delivery Network Support**

The Microsoft Ajax Content Delivery Network (CDN) enables you to easily add ASP.NET Ajax and jQuery scripts to your Web applications. For example, you can start using the jQuery library simply by adding a script element to your page that points to Ajax.microsoft.com, as shown in the following example:

<script src="http://ajax.microsoft.com/ajax/jquery/jquery-1.4.2.js" type="text/javascript"></script>

By taking advantage of the Microsoft Ajax CDN, you can significantly improve the performance of your Ajax applications. The contents of the Microsoft Ajax CDN are cached on servers that are located around the world. In addition, the Microsoft Ajax CDN enables browsers to reuse cached JavaScript files for Web sites that are located in different domains. The Microsoft Ajax CDN supports SSL (https://) in case you need to serve a Web page using the Secure Sockets Layer.

The ASP.NET [ScriptManager](http://msdn.microsoft.com/en-us/library/system.web.ui.scriptmanager.aspx) control supports the Microsoft Ajax CDN. Set the [EnableCdn](http://msdn.microsoft.com/en-us/library/system.web.ui.scriptmanager.enablecdn.aspx) property as shown in the following example to retrieve all ASP.NET framework JavaScript files from the CDN:

<asp:ScriptManager ID="sm1" EnableCdn="true" runat="server" />

When the [EnableCdn](http://msdn.microsoft.com/en-us/library/system.web.ui.scriptmanager.enablecdn.aspx) property is set to true, the ASP.NET framework will retrieve all ASP.NET framework JavaScript files from the CDN, including all JavaScript files that are used for validation and for the [UpdatePanel](http://msdn.microsoft.com/en-us/library/system.web.ui.updatepanel.aspx) control. This can have a dramatic impact on the performance of your Web application.

You can set the CDN path for your own JavaScript files by using the [WebResourceAttribute](http://msdn.microsoft.com/en-us/library/system.web.ui.webresourceattribute.aspx) attribute. The new [CdnPath](http://msdn.microsoft.com/en-us/library/system.web.ui.webresourceattribute.cdnpath.aspx) property specifies the path to the CDN used when you set the [EnableCdn](http://msdn.microsoft.com/en-us/library/system.web.ui.scriptmanager.enablecdn.aspx) property to true, as shown in the following example:

[assembly: WebResource("Example.js", "application/x-javascript", CdnPath ="http://contoso.com/app/site/Example.js")]

For more information about the Microsoft Ajax CDN, see [Microsoft Ajax Content Delivery Network](http://www.asp.net/ajaxlibrary/CDN.ashx) on the ASP.NET Web site.

**ScriptManager Explicit Scripts**

If you used the ASP.NET [ScriptManager](http://msdn.microsoft.com/en-us/library/system.web.ui.scriptmanager.aspx) in earlier versions of ASP.NET, your Web application had to load the entire ASP.NET Ajax Library. By taking advantage of the new [AjaxFrameworkMode](http://msdn.microsoft.com/en-us/library/system.web.ui.scriptmanager.ajaxframeworkmode.aspx) property, you can control exactly which components of the ASP.NET Ajax Library are loaded. For more information, see the [AjaxFrameworkMode](http://msdn.microsoft.com/en-us/library/system.web.ui.scriptmanager.ajaxframeworkmode.aspx) property.

[ASP.NET Web Forms](javascript:void(0))

Web Forms has been a core feature in ASP.NET since the release of ASP.NET 1.0. Many enhancements have been in this area for ASP.NET 4, such as the following:

* The ability to set meta tags.
* More control over view state.
* Support for recently introduced browsers and devices.
* Easier ways to work with browser capabilities.
* Support for using ASP.NET routing with Web Forms.
* More control over generated IDs.
* The ability to persist selected rows in data controls.
* More control over rendered HTML in the FormView and ListView controls.
* Filtering support for data source controls.
* Enhanced support for Web standards and accessibility.
* Project template changes.

**Setting Meta Tags with the Page.MetaKeywords and Page.MetaDescription Properties**

Two properties have been added to the [Page](http://msdn.microsoft.com/en-us/library/system.web.ui.page.aspx) class: [MetaKeywords](http://msdn.microsoft.com/en-us/library/system.web.ui.page.metakeywords.aspx) and [MetaDescription](http://msdn.microsoft.com/en-us/library/system.web.ui.page.metadescription.aspx). These two properties represent corresponding meta tags in the HTML rendered for a page, as shown in the following example:

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

<title>Untitled Page</title>

<meta name="keywords" content="keyword1, keyword2' />

<meta name="description" content="Description of my page" />

</head>

These two properties work like the [Title](http://msdn.microsoft.com/en-us/library/system.web.ui.page.title.aspx) property does, and they can be set in the [@ Page](http://msdn.microsoft.com/en-us/library/ydy4x04a.aspx) directive. For more information, see [Page.MetaKeywords](http://msdn.microsoft.com/en-us/library/system.web.ui.page.metakeywords.aspx) and [Page.MetaDescription](http://msdn.microsoft.com/en-us/library/system.web.ui.page.metadescription.aspx).

**Enabling View State for Individual Controls**

A new property has been added to the [Control](http://msdn.microsoft.com/en-us/library/system.web.ui.control.aspx) class: [ViewStateMode](http://msdn.microsoft.com/en-us/library/system.web.ui.control.viewstatemode.aspx). You can use this property to disable view state for all controls on a page except those for which you explicitly enable view state.

View state data is included in a page's HTML and increases the amount of time it takes to send a page to the client and post it back. Storing more view state than is necessary can cause significant decrease in performance. In earlier versions of ASP.NET, you could reduce the impact of view state on a page's performance by disabling view state for specific controls. But sometimes it is easier to enable view state for a few controls that need it instead of disabling it for many that do not need it.

For more information, see [Control.ViewStateMode](http://msdn.microsoft.com/en-us/library/system.web.ui.control.viewstatemode.aspx).

**Support for Recently Introduced Browsers and Devices**

ASP.NET includes a feature that is named browser capabilities that lets you determine the capabilities of the user's browser. Browser capabilities are represented by the [HttpBrowserCapabilities](http://msdn.microsoft.com/en-us/library/system.web.httpbrowsercapabilities.aspx) object which is stored in the [HttpRequest.Browser](http://msdn.microsoft.com/en-us/library/system.web.httprequest.browser.aspx) property. Information about a particular browser's capabilities is defined by a browser definition file. In ASP.NET 4, these browser definition files have been updated to contain information about recently introduced browsers and devices such as Google Chrome, Research in Motion BlackBerry smart phones, and Apple iPhone. Existing browser definition files have also been updated. For more information, see [How to: Upgrade an ASP.NET Web Application to ASP.NET 4](http://msdn.microsoft.com/en-us/library/dd483478.aspx) and [ASP.NET Web Server Controls and Browser Capabilities](http://msdn.microsoft.com/en-us/library/x3k2ssx2.aspx).

The browser definition files that are included with ASP.NET 4 are shown in the following list:

•blackberry.browser

•chrome.browser

•Default.browser

•firefox.browser

•gateway.browser

•generic.browser

•ie.browser

•iemobile.browser

•iphone.browser

•opera.browser

•safari.browser

**A New Way to Define Browser Capabilities**

ASP.NET 4 includes a new feature referred to as browser capabilities providers. As the name suggests, this lets you build a provider that in turn lets you write custom code to determine browser capabilities.

In ASP.NET version 3.5 Service Pack 1, you define browser capabilities in an XML file. This file resides in a machine-level folder or an application-level folder. Most developers do not need to customize these files, but for those who do, the provider approach can be easier than dealing with complex XML syntax. The provider approach makes it possible to simplify the process by implementing a common browser definition syntax, or a database that contains up-to-date browser definitions, or even a Web service for such a database.

For more information about the new browser capabilities provider, see the [What's New for ASP.NET 4 White Paper](http://go.microsoft.com/fwlink/?LinkId=147805).

**Routing in ASP.NET 4**

ASP.NET 4 adds built-in support for routing with Web Forms. Routing is a feature that was introduced with ASP.NET 3.5 SP1 and lets you configure an application to use URLs that are meaningful to users and to search engines because they do not have to specify physical file names. This can make your site more user-friendly and your site content more discoverable by search engines.

For example, the URL for a page that displays product categories in your application might look like the following example:

http://website/products.aspx?categoryid=12

By using routing, you can use the following URL to render the same information:

http://website/products/software

The second URL lets the user know what to expect and can result in significantly improved rankings in search engine results.

The new features include the following:

* The [PageRouteHandler](http://msdn.microsoft.com/en-us/library/system.web.routing.pageroutehandler.aspx) class is a simple HTTP handler that you use when you define routes. You no longer have to write a custom route handler.
* The [HttpRequest.RequestContext](http://msdn.microsoft.com/en-us/library/system.web.httprequest.requestcontext.aspx) and [Page.RouteData](http://msdn.microsoft.com/en-us/library/system.web.ui.page.routedata.aspx) properties make it easier to access information that is passed in URL parameters.
  + The RouteUrl expression provides a simple way to create a routed URL in markup.
  + The RouteValue expression provides a simple way to extract URL parameter values in markup.
* The [RouteParameter](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.routeparameter.aspx) class makes it easier to pass URL parameter values to a query for a data source control (similar to [FormParameter](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.formparameter.aspx)).
* You no longer have to change the Web.config file to enable routing.

For more information about routing, see the following topics:

* [ASP.NET Routing](http://msdn.microsoft.com/en-us/library/cc668201.aspx)
* [Walkthrough: Using ASP.NET Routing in a Web Forms Application](http://msdn.microsoft.com/en-us/library/dd329551.aspx)
* [How to: Define Routes for Web Forms Applications](http://msdn.microsoft.com/en-us/library/cc668177.aspx)
* [How to: Construct URLs from Routes](http://msdn.microsoft.com/en-us/library/cc668176.aspx)
* [How to: Access URL Parameters in a Routed Page](http://msdn.microsoft.com/en-us/library/dd535620.aspx)

**Setting Client IDs**

The new [ClientIDMode](http://msdn.microsoft.com/en-us/library/system.web.ui.control.clientidmode.aspx) property makes it easier to write client script that references HTML elements rendered for server controls. Increasing use of Microsoft Ajax makes the need to do this more common. For example, you may have a data control that renders a long list of products with prices and you want to use client script to make a Web service call and update individual prices in the list as they change without refreshing the entire page.

Typically you get a reference to an HTML element in client script by using the document.GetElementById method. You pass to this method the value of the id attribute of the HTML element you want to reference. In the case of elements that are rendered for ASP.NET server controls earlier versions of ASP.NET could make this difficult or impossible. You were not always able to predict what id values ASP.NET would generate, or ASP.NET could generate very long id values. The problem was especially difficult for data controls that would generate multiple rows for a single instance of the control in your markup.

ASP.NET 4 adds two new algorithms for generating id attributes. These algorithms can generate id attributes that are easier to work with in client script because they are more predictable and that are easier to work with because they are simpler. For more information about how to use the new algorithms, see the following topics:

* [ASP.NET Web Server Control Identification](http://msdn.microsoft.com/en-us/library/1d04y8ss.aspx)
* [Walkthrough: Making Data-Bound Controls Easier to Access from JavaScript](http://msdn.microsoft.com/en-us/library/dd381611.aspx)
* [Walkthrough: Making Controls Located in Web User Controls Easier to Access from JavaScript](http://msdn.microsoft.com/en-us/library/dd483476.aspx)
* [How to: Access Controls from JavaScript by ID](http://msdn.microsoft.com/en-us/library/dd410598.aspx)

**Persisting Row Selection in Data Controls**

The [GridView](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.gridview.aspx) and [ListView](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.listview.aspx) controls enable users to select a row. In previous versions of ASP.NET, row selection was based on the row index on the page. For example, if you select the third item on page 1 and then move to page 2, the third item on page 2 is selected. In most cases, is more desirable not to select any rows on page 2. ASP.NET 4 supports Persisted Selection, a new feature that was initially supported only in Dynamic Data projects in the .NET Framework 3.5 SP1. When this feature is enabled, the selected item is based on the row data key. This means that if you select the third row on page 1 and move to page 2, nothing is selected on page 2. When you move back to page 1, the third row is still selected. This is a much more natural behavior than the behavior in earlier versions of ASP.NET. Persisted selection is now supported for the [GridView](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.gridview.aspx) and [ListView](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.listview.aspx) controls in all projects. You can enable this feature in the [GridView](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.gridview.aspx) control, for example, by setting the [EnablePersistedSelection](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.gridview.enablepersistedselection.aspx) property, as shown in the following example:

<asp:GridView id="GridView2" runat="server" PersistedSelection="true">

</asp:GridView>

**FormView Control Enhancements**

The [FormView](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.formview.aspx) control is enhanced to make it easier to style the content of the control with CSS. In previous versions of ASP.NET, the [FormView](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.formview.aspx) control rendered it contents using an item template. This made styling more difficult in the markup because unexpected table row and table cell tags were rendered by the control. The [FormView](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.formview.aspx) control supports [RenderOuterTable](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.formview.renderoutertable.aspx), a property in ASP.NET 4. When this property is set to false, as show in the following example, the table tags are not rendered. This makes it easier to apply CSS style to the contents of the control.

<asp:FormView ID="FormView1" runat="server" RenderTable="false">

For more information, see [FormView Web Server Control Overview](http://msdn.microsoft.com/en-us/library/ms227992.aspx).

**ListView Control Enhancements**

The ListView control, which was introduced in ASP.NET 3.5, has all the functionality of the GridView control while giving you complete control over the output. This control has been made easier to use in ASP.NET 4. The earlier version of the control required that you specify a layout template that contained a server control with a known ID. The following markup shows a typical example of how to use the ListView control in ASP.NET 3.5.

<asp:ListView ID="ListView1" runat="server">

<LayoutTemplate>

<asp:PlaceHolder ID="ItemPlaceHolder" runat="server"></asp:PlaceHolder>

</LayoutTemplate>

<ItemTemplate>

<% Eval("LastName")%>

</ItemTemplate>

</asp:ListView>

In ASP.NET 4, the ListView control does not require a layout template. The markup shown in the previous example can be replaced with the following markup:

<asp:ListView ID="ListView1" runat="server">

<ItemTemplate>

<% Eval("LastName")%>

</ItemTemplate>

</asp:ListView>

For more information, see [ListView Web Server Control Overview](http://msdn.microsoft.com/en-us/library/bb398790.aspx).

**Filtering Data with the QueryExtender Control**

A very common task for developers who create data-driven Web pages is to filter data. This traditionally has been performed by building Where clauses in data source controls. This approach can be complicated, and in some cases the Where syntax does not let you take advantage of the full functionality of the underlying database.

To make filtering easier, a new [QueryExtender](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.queryextender.aspx) control has been added in ASP.NET 4. This control can be added to [EntityDataSource](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.entitydatasource.aspx) or [LinqDataSource](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.linqdatasource.aspx) controls in order to filter the data returned by these controls.

The [QueryExtender](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.queryextender.aspx) control supports a variety of filter options. The following lists [QueryExtender](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.queryextender.aspx) filter options.

|  |  |
| --- | --- |
| **Term** | **Definition** |
| [SearchExpression](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.expressions.searchexpression.aspx) | Searches a field or fields for string values and compares them to a specified string value. |
| [RangeExpression](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.expressions.rangeexpression.aspx) | Searches a field or fields for values in a range specified by a pair of values. |
| [PropertyExpression](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.expressions.propertyexpression.aspx) | Compares a specified value to a property value in a field. If the expression evaluates to true, the data that is being examined is returned. |
| [OrderByExpression](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.expressions.orderbyexpression.aspx) | Sorts data by a specified column and sort direction. |
| [CustomExpression](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.expressions.customexpression.aspx) | Calls a function that defines custom filter in the page. |

For more information, see [QueryExtender](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.queryextender.aspx)[QueryExtender Web Server Control Overview](http://msdn.microsoft.com/en-us/library/dd537671.aspx).

**Enhanced Support for Web Standards and Accessibility**

Earlier versions of ASP.NET controls sometimes render markup that does not conform to HTML, XHTML, or accessibility standards. ASP.NET 4 eliminates most of these exceptions.

For details about how the HTML that is rendered by each control meets accessibility standards, see [ASP.NET Controls and Accessibility](http://msdn.microsoft.com/en-us/library/ms227996.aspx).

**CSS for Controls that Can be Disabled**

In ASP.NET 3.5, when a control is disabled (see [WebControl.Enabled](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.webcontrol.enabled.aspx)), a disabled attribute is added to the rendered HTML element. For example, the following markup creates a [Label](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.label.aspx) control that is disabled:

<asp:Label id="Label1" runat="server"

  Text="Test" Enabled="false" />

In ASP.NET 3.5, the previous control settings generate the following HTML:

<span id="Label1" disabled="disabled">Test</span>

In HTML 4.01, the disabled attribute is not considered valid on span elements. It is valid only on input elements because it specifies that they cannot be accessed. On display-only elements such as span elements, browsers typically support rendering for a disabled appearance, but a Web page that relies on this non-standard behavior is not robust according to accessibility standards.

For display-only elements, you should use CSS to indicate a disabled visual appearance. Therefore, by default ASP.NET 4 generates the following HTML for the control settings shown previously:

<span id="Label1" class="aspNetDisabled">Test</span>

You can change the value of the class attribute that is rendered by default when a control is disabled by setting the [DisabledCssClass](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.webcontrol.disabledcssclass.aspx) property.

**CSS for Validation Controls**

In ASP.NET 3.5, validation controls render a default color of red as an inline style. For example, the following markup creates a [RequiredFieldValidator](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.requiredfieldvalidator.aspx) control:

<asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"

  ErrorMessage="Required Field" ControlToValidate="RadioButtonList1" />

ASP.NET 3.5 renders the following HTML for the validator control:

<span id="RequiredFieldValidator1"

  style="color:Red;visibility:hidden;">RequiredFieldValidator</span>

By default, ASP.NET 4 does not render an inline style to set the color to red. An inline style is used only to hide or show the validator, as shown in the following example:

<span id="RequiredFieldValidator1"

  style"visibility:hidden;">RequiredFieldValidator</span>

Therefore, ASP.NET 4 does not automatically show error messages in red. For information about how to use CSS to specify a visual style for a validation control, see [Validating User Input in ASP.NET Web Pages](http://msdn.microsoft.com/en-us/library/7kh55542.aspx).

**CSS for the Hidden Fields Div Element**

ASP.NET uses hidden fields to store state information such as view state and control state. These hidden fields are contained by a div element. In ASP.NET 3.5, this div element does not have a class attribute or an id attribute. Therefore, CSS rules that affect all div elements could unintentionally cause this div to be visible. To avoid this problem, ASP.NET 4 renders the div element for hidden fields with a CSS class that you can use to differentiate the hidden fields div from others. The new class value is shown in the following example:

<div class="aspNetHidden">

**CSS for the Table, Image, and ImageButton Controls**

By default, in ASP.NET 3.5, some controls set the border attribute of rendered HTML to zero (0). The following example shows HTML that is generated by the [Table](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.table.aspx) control in ASP.NET 3.5:

<table id="Table2" border="0">

The [Image](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.image.aspx) control and the [ImageButton](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.imagebutton.aspx) control also do this. Because this is not necessary and provides visual formatting information that should be provided by using CSS, the attribute is not generated in ASP.NET 4.

**CSS for the UpdatePanel and UpdateProgress Controls**

In ASP.NET 3.5, the [UpdatePanel](http://msdn.microsoft.com/en-us/library/system.web.ui.updatepanel.aspx) and [UpdateProgress](http://msdn.microsoft.com/en-us/library/system.web.ui.updateprogress.aspx) controls do not support expando attributes. This makes it impossible to set a CSS class on the HTML elements that they render. In ASP.NET 4 these controls have been changed to accept expando attributes, as shown in the following example:

<asp:UpdatePanel runat="server" class="myStyle">

</asp:UpdatePanel>

The following HTML is rendered for this markup:

<div id="ctl00\_MainContent\_UpdatePanel1" class="myStyle">

</div>

**Eliminating Unnecessary Outer Tables**

In ASP.NET 3.5, the HTML that is rendered for the following controls is wrapped in a table element whose purpose is to apply inline styles to the entire control:

* [FormView](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.formview.aspx)
* [Login](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.login.aspx)
* [PasswordRecovery](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.passwordrecovery.aspx)
* [ChangePassword](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.changepassword.aspx)

If you use templates to customize the appearance of these controls, you can specify CSS styles in the markup that you provide in the templates. In that case, no extra outer table is required. In ASP.NET 4, you can prevent the table from being rendered by setting the new [RenderOuterTable](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.formview.renderoutertable.aspx) property to false.

**Layout Templates for Wizard Controls**

In ASP.NET 3.5, the [Wizard](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.wizard.aspx) and [CreateUserWizard](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.createuserwizard.aspx) controls generate an HTML table element that is used for visual formatting. In ASP.NET 4 you can use a LayoutTemplate element to specify the layout. If you do this, the HTML table element is not generated. In the template, you create placeholder controls to indicate where items should be dynamically inserted into the control. (This is similar to how the template model for the [ListView](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.listview.aspx) control works.) For more information, see the [Wizard.LayoutTemplate](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.wizard.layouttemplate.aspx) property.

**New HTML Formatting Options for the CheckBoxList and RadioButtonList Controls**

ASP.NET 3.5 uses HTML table elements to format the output for the [CheckBoxList](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.checkboxlist.aspx) and [RadioButtonList](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.radiobuttonlist.aspx) controls. To provide an alternative that does not use tables for visual formatting, ASP.NET 4 adds two new options to the [RepeatLayout](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.repeatlayout.aspx) enumeration:

* [UnorderedList](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.repeatlayout.aspx). This option causes the HTML output to be formatted by using ul and li elements instead of a table.
* [OrderedList](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.repeatlayout.aspx). This option causes the HTML output to be formatted by using ol and li elements instead of a table.

For examples of HTML that is rendered for the new options, see the [RepeatLayout](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.repeatlayout.aspx) enumeration.

**Header and Footer Elements for the Table Control**

In ASP.NET 3.5, the [Table](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.table.aspx) control can be configured to render thead and tfoot elements by setting the [TableSection](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.tablerow.tablesection.aspx) property of the [TableHeaderRow](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.tableheaderrow.aspx) class and the [TableFooterRow](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.tablefooterrow.aspx) class. In ASP.NET 4 these properties are set to the appropriate values by default.

**CSS and ARIA Support for the Menu Control**

In ASP.NET 3.5, the [Menu](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.menu.aspx) control uses HTML table elements for visual formatting, and in some configurations it is not keyboard-accessible. ASP.NET 4 addresses these problems and improves accessibility in the following ways:

* The generated HTML is structured as an unordered list (ul and li elements).
* CSS is used for visual formatting.
* The menu behaves in accordance with ARIA standards for keyboard access. You can use arrow keys to navigate menu items. (For information about ARIA, see [Accessibility in Visual Studio and ASP.NET](http://msdn.microsoft.com/en-us/library/ms228004.aspx).)
* ARIA role and property attributes are added to the generated HTML. (Attributes are added by using JavaScript instead of included in the HTML, to avoid generating HTML that would cause markup validation errors.)

Styles for the [Menu](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.menu.aspx) control are rendered in a style block at the top of the page, instead of inline with the rendered HTML elements. If you want to use a separate CSS file so that you can modify the menu styles, you can set the [Menu](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.menu.aspx) control's new [IncludeStyleBlock](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.menu.includestyleblock.aspx) property to false, in which case the style block is not generated.

**Valid XHTML for the HtmlForm Control**

In ASP.NET 3.5, the [HtmlForm](http://msdn.microsoft.com/en-us/library/system.web.ui.htmlcontrols.htmlform.aspx) control (which is created implicitly by the <form runat="server"> tag) renders an HTML form element that has both name and id attributes. The name attribute is deprecated in XHTML 1.1. Therefore, this control does not render the name attribute in ASP.NET 4.

**Maintaining Backward Compatibility in Control Rendering**

An existing ASP.NET Web site might have code in it that assumes that controls are rendering HTML the way they do in ASP.NET 3.5. To avoid causing backward compatibility problems when you upgrade the site to ASP.NET 4, you can have ASP.NET continue to generate HTML the way it does in ASP.NET 3.5 after you upgrade the site. To do so, you can set the controlRenderingCompatibilityVersion attribute of the pages element to "3.5" in the Web.config file of an ASP.NET 4 Web site, as shown in the following example:

<system.web>

  <pages controlRenderingCompatibilityVersion="3.5"/>

</system.web>

If this setting is omitted, the default value is the same as the version of ASP.NET that the Web site targets. (For information about multi-targeting in ASP.NET, see [.NET Framework Multi-Targeting for ASP.NET Web Projects](http://msdn.microsoft.com/en-us/library/bb398791.aspx).)

[Project Template Changes](javascript:void(0))

In earlier versions of ASP.NET, when you use Visual Studio to create a new Web Site project or Web Application project, the resulting projects contain only a Default.aspx page, a default Web.config file, and the App\_Data folder. Visual Studio also supports an Empty Web Site project type, which contains no files at all. The result is that for the beginner, there is very little guidance on how to build a production Web application. Therefore, ASP.NET 4 introduces three new templates, one for an empty Web application project, and one each for a Web Application and Web Site project:

* Empty Web Application and Web Site Project Templates. These are similar to the Empty Web Site layout from earlier versions of ASP.NET, except they contain a Web.config file that specifies the targeted version of the .NET Framework.
* Web Application and Web Site Project Templates. These include a number of files that were not created in earlier versions. The additional files provide basic membership functionality, a master page and content pages that use it, AJAX and CSS files. The intention of these changes to the project templates is to provide guidance on how to start building a new Web application.,

For more information, see [Visual Studio Templates for Web Projects](http://msdn.microsoft.com/en-us/library/ee377605.aspx).

[ASP.NET MVC](javascript:void(0))

ASP.NET MVC helps Web developers build compelling standards-based Web sites that are easy to maintain because it decreases the dependency among application layers by using the Model-View-Controller (MVC) pattern. MVC provides complete control over the page markup. It also improves testability by inherently supporting Test Driven Development (TDD).

Web sites created using ASP.NET MVC have a modular architecture. This allows members of a team to work independently on the various modules and can be used to improve collaboration. For example, developers can work on the model and controller layers (data and logic), while the designer work on the view (presentation).

For tutorials, walkthroughs, conceptual content, code samples, and a complete API reference, see [ASP.NET MVC 2](http://msdn.microsoft.com/en-us/library/dd394709.aspx).

[Dynamic Data](javascript:void(0))

Dynamic Data was introduced in the .NET Framework 3.5 SP1 release in mid-2008. This feature provides many enhancements for creating data-driven applications, such as the following:

* A RAD experience for quickly building a data-driven Web site.
* Automatic validation that is based on constraints defined in the data model.
* The ability to easily change the markup that is generated for fields in the GridView and DetailsView controls by using field templates that are part of your Dynamic Data project.

For ASP.NET 4, Dynamic Data has been enhanced to give developers even more power for quickly building data-driven Web sites. For more information, see [ASP.NET Dynamic Data Content Map](http://msdn.microsoft.com/en-us/library/cc488545.aspx).

**Enabling Dynamic Data for Individual Data-Bound Controls in Existing Web Applications**

You can use Dynamic Data features in existing ASP.NET Web applications that do not use scaffolding by enabling Dynamic Data for individual data-bound controls. Dynamic Data provides the presentation and data layer support for rendering these controls. When you enable Dynamic Data for data-bound controls, you get the following benefits:

* Setting default values for data fields. Dynamic Data enables you to provide default values at run time for fields in a data control.
* Interacting with the database without creating and registering a data model.
* Automatically validating the data that is entered by the user without writing any code.

For more information, see [Walkthrough: Enabling Dynamic Data in ASP.NET Data-Bound Controls](http://msdn.microsoft.com/en-us/library/cc488549.aspx).

**Declarative DynamicDataManager Control Syntax**

The [DynamicDataManager](http://msdn.microsoft.com/en-us/library/system.web.dynamicdata.dynamicdatamanager.aspx) control has been enhanced so that you can configure it declaratively, as with most controls in ASP.NET, instead of only in code. The markup for the [DynamicDataManager](http://msdn.microsoft.com/en-us/library/system.web.dynamicdata.dynamicdatamanager.aspx) control resembles the following example:

<asp:DynamicDataManager ID="DynamicDataManager1" runat="server"

AutoLoadForeignKeys="true">

<DataControls>

<asp:DataControlReference ControlID="GridView1" />

</DataControls>

</asp:DynamicDataManager>

<asp:GridView id="GridView1" runat="server"

</asp:GridView>

This markup enables Dynamic Data behavior for the GridView1 control that is referenced in the DataControls section of the [DynamicDataManager](http://msdn.microsoft.com/en-us/library/system.web.dynamicdata.dynamicdatamanager.aspx) control.

**Entity Templates**

Entity templates offer a new way to customize the layout of data without requiring you to create a custom page. Page templates use the [FormView](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.formview.aspx) control instead of the [DetailsView](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.detailsview.aspx) control, as was used in page templates in earlier versions of Dynamic Data. Page templates also use the [DynamicEntity](http://msdn.microsoft.com/en-us/library/system.web.dynamicdata.dynamicentity.aspx) control to render Entity templates. This gives you more control over the markup that is rendered by Dynamic Data.

For more information about entity templates, see [ASP.NET 4 and Visual Studio 2010 Web Development Overview](http://go.microsoft.com/fwlink/?LinkId=147805) (.pdf format) on the ASP.NET Web site.

**New Field Templates for URLs and E-mail Addresses**

ASP.NET 4 introduces two new built-in field templates, EmailAddress.ascx and Url.ascx. These templates are used for fields that are marked as EmailAddress or Url using the [DataTypeAttribute](http://msdn.microsoft.com/en-us/library/system.componentmodel.dataannotations.datatypeattribute.aspx) attribute. For EmailAddress objects, the field is displayed as a hyperlink that is created by using the mailto: protocol. When users click the link, it opens the user's e-mail client and creates a skeleton message. Objects typed as Url are displayed as ordinary hyperlinks.

The following example shows how to mark fields.

[DataType(DataType.EmailAddress)]

public object HomeEmail { get; set; }

[DataType(DataType.Url)]

public object Website { get; set; }

**Creating Links with the DynamicHyperLink Control**

Dynamic Data uses the new routing feature that was added in the .NET Framework 3.5 SP1 to control the URLs that users see when they access the Web site. The new DynamicHyperLink control makes it easy to build links to pages in a Dynamic Data site.

For information, see [How to: Create Table Action Links in Dynamic Data](http://msdn.microsoft.com/en-us/library/dd465348.aspx)

**Support for Inheritance in the Data Model**

Both the ADO.NET Entity Framework and LINQ to SQL support inheritance in their data models. An example of this might be a database that has an InsurancePolicy table. It might also contain CarPolicy and HousePolicy tables that have the same fields as InsurancePolicy and then add more fields. Dynamic Data has been modified to understand inherited objects in the data model and to support scaffolding for the inherited tables.

For more information, see [Walkthrough: Mapping Table-per-Hierarchy Inheritance in Dynamic Data](http://msdn.microsoft.com/en-us/library/dd793152.aspx).

**Support for Many-to-Many Relationships (Entity Framework Only)**

The Entity Framework has rich support for many-to-many relationships between tables, which is implemented by exposing the relationship as a collection on an Entity object. New field templates (ManyToMany.ascx and ManyToMany\_Edit.ascx) have been added to provide support for displaying and editing data that is involved in many-to-many relationships.

For more information, see [Working with Many-to-Many Data Relationships in Dynamic Data](http://msdn.microsoft.com/en-us/library/dd742359.aspx).

**New Attributes to Control Display and Support Enumerations**

The [DisplayAttribute](http://msdn.microsoft.com/en-us/library/system.componentmodel.dataannotations.displayattribute.aspx) has been added to give you additional control over how fields are displayed. The [DisplayNameAttribute](http://msdn.microsoft.com/en-us/library/system.componentmodel.displaynameattribute.aspx) attribute in earlier versions of Dynamic Data enabled you to change the name that is used as a caption for a field. The new DisplayAttribute class lets you specify more options for displaying a field, such as the order in which a field is displayed and whether a field will be used as a filter. The attribute also provides independent control of the name that is used for the labels in a GridView control, the name that is used in a DetailsView control, the help text for the field, and the watermark used for the field (if the field accepts text input).

The EnumDataTypeAttribute class has been added to let you map fields to enumerations. When you apply this attribute to a field, you specify an enumeration type. Dynamic Data uses the new Enumeration.ascx field template to create UI for displaying and editing enumeration values. The template maps the values from the database to the names in the enumeration.

**Enhanced Support for Filters**

Dynamic Data 1.0 had built-in filters for Boolean columns and foreign-key columns. The filters did not let you specify the order in which they were displayed. The new DisplayAttribute attribute addresses this by giving you control over whether a column appears as a filter and in what order it will be displayed.

An additional enhancement is that filtering support has been rewritten to use the new [QueryExtender](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.queryextender.aspx) feature of Web Forms. This lets you create filters without requiring knowledge of the data source control that the filters will be used with. Along with these extensions, filters have also been turned into template controls, which lets you add new ones. Finally, the DisplayAttribute class mentioned earlier allows the default filter to be overridden, in the same way that UIHint allows the default field template for a column to be overridden.

For more information, see [Walkthrough: Filtering Rows in Tables That Have a Parent-Child Relationship](http://msdn.microsoft.com/en-us/library/dd985039.aspx) and [QueryableFilterRepeater](http://msdn.microsoft.com/en-us/library/system.web.dynamicdata.queryablefilterrepeater.aspx).

[ASP.NET Chart Control](javascript:void(0))

The ASP.NET chart server control enables you to create ASP.NET pages applications that have simple, intuitive charts for complex statistical or financial analysis. The chart control supports the following features:

* Data series, chart areas, axes, legends, labels, titles, and more.
* Data binding.
* Data manipulation, such as copying, splitting, merging, alignment, grouping, sorting, searching, and filtering.
* Statistical formulas and financial formulas.
* Advanced chart appearance, such as 3-D, anti-aliasing, lighting, and perspective.
* Events and customizations.
* Interactivity and Microsoft Ajax.
* Support for the Ajax Content Delivery Network (CDN), which provides an optimized way for you to add Microsoft Ajax Library and jQuery scripts to your Web applications.

For more information, see [Chart Web Server Control Overview](http://msdn.microsoft.com/en-us/library/ee410579.aspx).

[Visual Web Developer Enhancements](javascript:void(0))

The following sections provide information about enhancements and new features in Visual Studio 2010 and Visual Web Developer Express.

The Web page designer in Visual Studio 2010 has been enhanced for better CSS compatibility, includes additional support for HTML and ASP.NET markup snippets, and features a redesigned version of IntelliSense for JScript.

**Improved CSS Compatibility**

The Visual Web Developer designer in Visual Studio 2010 has been updated to improve CSS 2.1 standards compliance. The designer better preserves HTML source code and is more robust than in previous versions of Visual Studio.

**HTML and JavaScript Snippets**

In the HTML editor, IntelliSense auto-completes tag names. The IntelliSense Snippets feature auto-completes whole tags and more. In Visual Studio 2010, IntelliSense snippets are supported for JScript, alongside C# and Visual Basic, which were supported in earlier versions of Visual Studio.

Visual Studio 2010 includes over 200 snippets that help you auto-complete common ASP.NET and HTML tags, including required attributes (such as runat="server") and common attributes specific to a tag (such as ID, DataSourceID, ControlToValidate, and Text).

You can download additional snippets, or you can write your own snippets that encapsulate the blocks of markup that you or your team use for common tasks. For more information on HTML snippets, see [Walkthrough: Using HTML Snippets](http://msdn.microsoft.com/en-us/library/dd985242.aspx).

**JavaScript IntelliSense Enhancements**

In Visual 2010, JScript IntelliSense has been redesigned to provide an even richer editing experience. IntelliSense now recognizes objects that have been dynamically generated by methods such as registerNamespace and by similar techniques used by other JavaScript frameworks. Performance has been improved to analyze large libraries of script and to display IntelliSense with little or no processing delay. Compatibility has been significantly increased to support almost all third-party libraries and to support diverse coding styles. Documentation comments are now parsed as you type and are immediately leveraged by IntelliSense.

**Web Application Deployment with Visual Studio 2010**

For Web application projects, Visual Studio now provides tools that work with the IIS Web Deployment Tool (Web Deploy) to automate many processes that had to be done manually in earlier versions of ASP.NET. For example, the following tasks can now be automated:

* Creating an IIS application on the destination computer and configuring IIS settings.
* Copying files to the destination computer.
* Changing Web.config settings that must be different in the destination environment.
* Propagating changes to data or data structures in SQL Server databases that are used by the Web application.

For more information about Web application deployment, see [ASP.NET Deployment Content Map](http://msdn.microsoft.com/en-us/library/bb386521.aspx).

[Enhancements to ASP.NET Multi-Targeting](javascript:void(0))

ASP.NET 4 adds new features to the multi-targeting feature to make it easier to work with projects that target earlier versions of the .NET Framework.

Multi-targeting was introduced in ASP.NET 3.5 to enable you to use the latest version of Visual Studio without having to upgrade existing Web sites or Web services to the latest version of the .NET Framework.

In Visual Studio 2008, when you work with a project targeted for an earlier version of the .NET Framework, most features of the development environment adapt to the targeted version. However, IntelliSense displays language features that are available in the current version, and property windows display properties available in the current version. In Visual Studio 2010, only language features and properties available in the targeted version of the .NET Framework are shown.