ASP.NET MVC Beta Release Notes

This document describes changes that have been made to the ASP.NET MVC framework since previous releases for the current Beta release. It also describes changes that you must make in existing MVC applications to run with the new release.

[What’s New 2](#_Toc210044483)

[MVC Futures 2](#_Toc210044484)

[Beta Installer and the GAC 2](#_Toc210044485)

[Changes Made Between Preview 3 and Beta 3](#_Toc210044486)

[New Simple Membership Features in the Default Project Template 3](#_Toc210044487)

[New Filter Types for Authorization and Exception Handling 3](#_Toc210044488)

[New Output Cache Filter 4](#_Toc210044489)

[Changes for ASP.NET AJAX 4](#_Toc210044490)

[Namespaces in Routes 4](#_Toc210044491)

[New Interface for Enhanced Testability of TempData 5](#_Toc210044492)

[ActionInvoker Extensibility Improvements 5](#_Toc210044493)

[ViewDataDictionary 6](#_Toc210044494)

[ViewEngine Improvements 6](#_Toc210044495)

[Helper Improvements 6](#_Toc210044496)

[Controller and Filter Improvements 7](#_Toc210044497)

[Known Issues and Breaking Changes 9](#_Toc210044498)

[Upgrading from CodePlex Preview 3 to ASP.NET MVC Beta 10](#_Toc210044499)

[Changes Made Between CodePlex Preview 5 and Beta 11](#_Toc210044500)

[Changes 11](#_Toc210044501)

[Bug Fixes 13](#_Toc210044502)

[Upgrading from CodePlex Preview 5 to Beta 13](#_Toc210044503)

# What’s New

ASP.NET MVC Beta is an officially supported Beta release of the ASP.NET MVC framework. As always, feel free to provide feedback on the ASP.NET MVC forums (<http://forums.asp.net/1146.aspx>).

* **Note**: Before you run the ASP.NET MVC Beta installer, uninstall any earlier previews of ASP.NET MVC that you have installed. In addition, close all instances of Visual Studio 2008.

This section provides information about the new and changed features in this release.

# MVC Futures

Several preview releases contained an extra assembly named MvcFutures.dll. This assembly contains code that the feature team might want to include in future versions of ASP.NET MVC. This code has not undergone rigorous testing and might not meet the high quality bar that the team sets for production releases.

We will not be releasing this assembly in the RTM version of ASP.NET MVC. Therefore, in the ASP.NET MVC Beta , we’ve removed this assembly from the project template, because the Beta is intended to be as similar to the RTM release as possible. Some of the features in this assembly might make be integrated into the core framework in a future version of ASP.NET MVC.

The assembly remains available as a separate download. If you have created code that relies on this assembly, you can download it from <http://codeplex.com/aspnet>. Note that this code is not supported—use it at your own risk.

# Beta Installer and the GAC

The Beta installer installs the ASP.NET MVC assemblies (System.Web.Mvc.dll, System.Web.Routing.dll, and System.Web.Abstractions.dll) into the GAC. In previous previews, these were not installed into the GAC. Because of this change, the default project templates do not automatically copy the assembly into the Bin directory of your application.

The installer also copies the assemblies to the following folder:

%ProgramFiles%\Microsoft ASP.NET\ASP.NET MVC Beta

Even though the installer installs the assemblies to the GAC by default, you can copy the assemblies to an application's Bin folder. This lets you deploy your application to a hosting provider that supports ASP.NET 3.5, but that has not installed ASP.NET MVC Beta. This means that ASP.NET 3.5 SP1 is not required for an ASP.NET MVC application.

# Changes Made Between Preview 3 and Beta

ASP.NET MVC Preview 3 was the last official release of ASP.NET MVC that was posted on the <http://asp.net/> Web site. There have been subsequent interim releases on the CodePlex site at <http://codeplex.com/mvc>.

This section describes changes that were made between the Preview 3 release and the current Beta release. If you’ve been using the latest releases on CodePlex, you can skip to the next section, which covers just the changes that have been made since CodePlex Preview 5.

## New Simple Membership Features in the Default Project Template

Most Web applications require some form of user authentication. This release includes an AccountController class in the default project template that handles the following user actions:

* Login
* Logout
* Register
* Change password

The AccountController class provides an example of how to interact with the Membership API from an ASP.NET MVC controller.

The Web.config file in the default project template includes settings for the MembershipProvider class. This means that ASP.NET will automatically create a SQL Server Express Edition membership database when membership features are first accessed. You can transfer the SQL Server Express Edition database to SQL Server database, or you can modify the connection string in the Web.config file to point to an existing database that contains membership tables.

The default template also includes copies of Microsoft ASP.NET AJAX script files.

## New Filter Types for Authorization and Exception Handling

This release includes two new filter types: authorization filters and exception filters. Although action filters continue to be the most common filter type, we wanted to provide filters that could run before any and all action filters, no matter what the scope was for the filters. This can help prevent scenarios, such as where an OutputCache filter runs before an Authorization filter, and thus circumvents authorization.

The following features have been added or changed to support the new filter types:

* New IAuthorizationFilter and IExceptionFilter interfaces. Authorization filters are guaranteed to run before any action filters. Every exception filter is executed, even if a filter indicates that it has handled the request. This is useful for logging and handling exceptions.
* AuthorizeAttribute class. This is the default concrete implementation of IAuthorizationFilter. It is used to secure action methods.
* HandleErrorAttribute class. This is the default concrete implementation of IExceptionFilter. It is used to handle exceptions and to specify a view to render when an exception occurs.
* New FilterAttribute base class. This is a new base class that is useful for any filter attribute.

## New Output Cache Filter

OutputCacheAttribute is a new action filter that is used to cache the output of an action method by using the built-in ASP.NET output cache. The functionality supplied by this attribute is similar to the functionality that is provided by the OutputCache directive in ASP.NET Web pages.

## Changes for ASP.NET AJAX

The following AJAX helpers were added. They make use of a new AjaxOptions class that is used to specify options for asynchronous operations.

ActionLink*.* This renders an anchor tag to an action method. When the link is clicked, the action method is invoked asynchronously. A typical use for this helper it to read information from the response and update a DOM element by using the AjaxOptions.UpdateTargetId property.

BeginForm*.* This renders an HTML form that is submitted asynchronously. A typical use for this helper is to submit a form and then, as with ActionLink, read the response and update a DOM element by using the AjaxOptions.UpdateTargetId property.

## Namespaces in Routes

In earlier previews of ASP.NET MVC, the framework scanned all the types in an assembly in order to find controller implementations. This occasionally caused exceptions to be thrown when the framework reflected over a type that inherits from another type for which no assembly is loaded.

In this release, we provide a means to specify which namespaces to examine when the framework tries to load a controller type by using the DefaultNamespaces property of ControllerBuilder. The following example shows how to add namespaces for resolving controllers.

void Application\_Start(object sender, EventArgs e) {

 ControllerBuilder.Current.DefaultNamespaces.Add("MyApp.Controllers");

 ControllerBuilder.Current.DefaultNamespaces.Add("MyApp.Blog.Controllers");

 ControllerBuilder.Current.DefaultNamespaces.Add("ThirdPartyApp.Controllers");

 // ...

}

You can also specify namespaces for individual routes. (In this release you cannot do so by using the MapRoute extension method.) The following example shows how to specify namespaces for a route.

var dataTokens = new RouteValueDictionary();

dataTokens.Add("namespaces", new HashSet<string>(

 new string[] {

 "MyApp.Blog.Controllers",

 "MyApp.Forum.Controllers",

 "MyApp.Controllers"

 }));

routes.Add(

 new Route("ns/{controller}/{action}/{id}", new MvcRouteHandler()) {

 Defaults = new RouteValueDictionary(new {

 action = "Index",

 id = (string)null

 }),

 DataTokens = dataTokens

});

## New Interface for Enhanced Testability of TempData

In this release we improved testability by allowing you to use session cookies rather than session state. A new ITempDataProvider interface has been added. By default, controllers use the SessionStateTempDataProvider which uses session state, but it is now possible to implement other providers. An example of a cookie-based ITempDataProvider is included in the MVC Futures project.

## ActionInvoker Extensibility Improvements

New virtual methods have been added that allow you to extend the action-method invoker in advanced scenarios. The new methods are listed in the following table.

|  |  |
| --- | --- |
| **Method** | **Notes** |
| GetFiltersForActionMethod | Returns all filters (authorization, action, and exception filters) for the action method. |
| InvokeActionResultWithFilters | Calls ExecuteResulton the ActionResult object that is returned from the action method, along with all the result filters that are applied to the action method. |
| InvokeAuthorizationFilters | Invokes the authorization filters that are applied to the action method. |
| InvokeExceptionFilters | Invokes the exception filters that are applied to the action method. |

## ViewDataDictionary

We made a minor change to the ViewDataDictionary class to change its indexer to be a normal indexer. We also added an Eval method that evaluates the model. The Eval method contains an overload that accepts a format string.

## ViewEngine Improvements

The following improvements have been made to the ViewEngine class:

* **Added global registration of view engines**. It is no longer the controller’s responsibility to determine which view engine to use to render a view (though the controller can still select a view engine if needed). Instead, when a controller wants to render a view, it returns a ViewResult object (via the View method).

By default, the ViewResult instance calls into the AutoViewEngine to find a view. The AutoViewEngine iterates through each IViewEngine instance in the ViewEngines.Engines static collection and queries each instance (by calling the FindView or FindPartialView methods) to determine whether it can render the specified view. The first engine that can render the view returns a ViewEngineResult instance that contains an IView instance that actually performs the rendering.

* **Changed the IViewEngine interface to add the FindPartial method**. View engines now support the concept of rendering partial views. This is used in tandem with the Html.RenderPartial helper method.

## Helper Improvements

The following improvements have been made to helpers:

* **Added support for rendering partial views**.You have several options for passing view data to the partial view. The partial view can be rendered by a different view engine than the containing view. Usage is <% Html.RenderPartial(...); %>. This method does not return a string, but instead renders to the TextWriter instance of the underlying response.
* **Added a parameter to specify a default option label for DropDownList controls**. Specifying a value for the optionLabel argument to the DropDownList method will render the first option element with its value attribute equal to an empty string, but with its inner text as the value that you provide. This fulfills a common scenario where the first option in a drop-down list control is not a valid value, but exists just to provide instructions.
* **Converted all the AjaxHelper and HtmlHelper methods into extension methods of their classes**. The AjaxHelper and HtmlHelper methodswere also moved to new namespaces. This makes it easy to completely replace the AJAX helper and HTML helper methods with your own code implementations.

**Note**: If you are upgrading an existing Preview 3 project and do not see these methods, import the System.Web.Mvc.Ajax and System.Web.Mvc.Html namespaces into your application by adding the following highlighted lines to the Web.config file:

<namespaces>

 <add namespace="System.Web.Mvc"/>

 <add namespace="System.Web.Mvc.Ajax"/>

 <add namespace="System.Web.Mvc.Html"/>

 <add namespace="Microsoft.Web.Mvc"/>

 <add namespace="System.Web.Routing"/>

 <add namespace="System.Linq"/>

 <add namespace="System.Collections.Generic"/>

</namespaces>

* **Added RadioButton and TextArea helper methods and made overall improvements to other helpers**. In some cases, overloads have been added or removed to avoid overload ambiguity. (See next point.)
* **Removed helper method overloads to avoid ambiguity**. For example, in order to reduce overload ambiguity, some overloads were removed and the value parameter was changed from object to string for several helper methods. This avoids scenarios such as one in which you call Html.Hidden("name", 123) and expect "123" to be the value, but instead the overload is invoked that accepts an object that represents the HTML attributes (because 123 is an object).

## Controller and Filter Improvements

The following improvements have been made to controllers and filters:

* **Added array,** **IEnumerable,** **ICollection, and** **IList support for action method parameters.** Data from elements with the same name can be aggregated into an array that is passed to the action method. For example, suppose you have three form elements (such as text boxes) named “color”, each with a different color name. When the form is posted to an action, if the action method has a parameter named “color” that is a string array, the invoker will populate the array with the posted color names.
* **Removed the ActionMethod property from action filter context objects.** This removes coupling to reflection in our public API as much as possible.
* **Added support for custom model binders**. Custom binders allow you to define complex types as parameters to an action method. To use this feature, mark the complex type or the parameter declaration with [ModelBinder(...)].
* **Added an IActionInvoker interface.** The Controller class no longer has a dependency on the concrete ControllerActionInvoker class. It now references the IActionInvoker interface, which the ControllerActionInvoker implements. This leaves open the possibility of different implementations of the invoker in the future.
* **Added an UpdateModel method to the Controller class.** The method can be used to update the property values of a model object based on either the values posted in the HTTP request or on route data. When you attempt to update the model, errors that occur when the post values are converted to the property type are recorded in the new ModelStateDictionary instance, which stores information about the model.
* **Changed** **HandleErrorAttribute so that it does not handle exceptions when HttpContext.IsCustomErrorEnabled is false.** This allows exceptions to propagate to the standard ASP.NET error page (the “Yellow Screen of Death”) during development, which provides you with more information than an error page that is intended for users. For more information, see Eilon Lipton’s blog post "[HttpContext.IsCustomErrorEnabled - One of ASP.NET's hidden gems](http://weblogs.asp.net/leftslipper/archive/2008/07/13/httpcontext-iscustomerrorenabled-one-of-asp-net-s-hidden-gems.aspx)".
* **Added a new AcceptVerbsAttribute** class.Thisattribute allows you to specify which HTTP verbs an action method will handle. If the current request’s HTTP method does not match the list of verbs that are specified in the attribute, from the perspective of the action invoker, the method does not exist. This allows you to have two action methods of the same name, as long as they respond to different HTTP verbs. This attribute inherits from the abstract ActionMethodSelectorAttribute class, which can be used to control how action methods are selected in order to respond to a request.
* **Added a new ActionName attribute.** Thisattribute allows you to assign an action name to a method. By default, if a method does not have this attribute applied, the method name itself serves as the action name.

For example, you can map a method named ViewName to respond to the action View. This attribute and the AcceptVerbs attribute can be used together, as shown in the following example:

[AcceptVerbs("GET")]

public ActionResult MyAction(...);

[ActionName("MyAction"), AcceptVerb("POST")]

public ActionResult MyAction\_FormSubmit(...);

[AcceptVerbs("GET")]

public ActionResult OtherActionOverloaded(string id);

[AcceptVerbs("POST")]

public ActionResult OtherActionOverloaded(string id, ...);

A GET request for /controller/MyAction is handled by the first method, while a POST request for /controller/MyAction is handled by the second method.

Likewise, even though the code contains two methods named OtherActionOverloaded, which is normally not allowed for actions, a GET request for /controller/OtherActionOverloaded only matches one of those methods (due to the AcceptVerbsAttribute) and no exception is thrown. A POST request for the same URL matches the second OtherActionOverloaded method.

## Known Issues and Breaking Changes

### Controllers

The abstract Controller class now derives from ControllerBase, which contains properties such as ViewData and TempData. Additionally, the ControllerContext.Controller property is a ControllerBase reference rather than an IController reference. This change was made so that filters can access ViewData and TempData instances without casting the ControllerContext.Controller property.

To support this change, the protected Controller.Execute(controllerContext) method was removed. If you have a custom controller that overrides the Execute(controllerContext) method, change it to override the ExecuteCore method instead.

Controllers that require additional initialization steps to perform tasks such as setting the thread culture or assigning a custom provider for TempData can do so by overriding the Initialize(requestContext) method. The implementation in the base class ensures that the ControllerContext property will be set.

### View Engines

The IViewEngine interface has been changed. Its responsibility is now to find views, not to render them. This change in responsibility results in breaking changes in the API.

### Helper Methods

Some overloads for some helper methods have been removed. This might cause errors in your existing views when you upgrade to this version of ASP.NET MVC. Also, the namespace for the ASP.NET AJAX helper methods has been changed. For more information, see the [Helper Improvements](#_Helper_Improvements) section

## Upgrading from CodePlex Preview 3 to ASP.NET MVC Beta

To migrate an application from the Preview 3 release to the current Beta release, make the following changes in your applications.

* Update the references to the following assemblies to point to the new Beta versions of the assemblies:

System.Web.Abstractions.dll

System.Web.Routing.dll

System.Web.Mvc.dll

By default, these assemblies are located in the following folder:

%ProgramFiles%\Microsoft ASP.NET\ASP.NET MVC Beta

* Because the version numbers of the System.Web.Abstractions and System.Web.Routing assemblies have been changed back to 3.5.0.0 (we now include exact binary copies of these assemblies released in ASP.NET 3.5 SP1), you must update version information in the Web.config file. In the Web.config file, search for the following string:
1. System.Web.Routing, Version=0.0.0.0

Replace it with the following string:

1. System.Web.Routing, Version=3.5.0.0

Search for the following string:

1. System.Web.Abstractions, Version=0.0.0.0

Replace it with the following string:

1. System.Web.Abstractions, Version=3.5.0.0
* Due to changes in the action filter APIs, if you wrote any custom action filters, you might need to update your code to match the new signatures.
* In the Web.config namespaces section, add the following namespace entries:

<add namespace="System.Web.Mvc.Ajax"/>

<add namespace="System.Web.Mvc.Html"/>

* After you have made these changes, compile your application and resolve any compilation errors. Most of the errors should be the result of one of the breaking changes listed above.
* To take advantage of changes to the templates, copy the template changes into your own project.

# Changes Made Between CodePlex Preview 5 and Beta

Even though the last official release of ASP.NET MVC was ASP.NET MVC Preview 3, many developers downloaded and used the interim CodePlex releases. This section describes changes and bug fixes that have been made between CodePlex Preview 5 and Beta.

## Changes

* Changed the default validation messages to be more end-user friendly.
* Renamed CompositeViewEngine to AutoViewEngine.
* Added a Url property to Controller of type UrlHelper. This makes it convenient to generate routing-based URLs from within a controller.
* Added the ActionNameSelectorAttribute abstract base class, which serves as the base type for ActionNameAttribute. By inheriting from this base attribute class, you can create custom attributes that participate in action selection by name.
* Added a new ReleaseView method to IViewEngine that allows custom view engines to be notified when a view is done rendering. This is useful for cleanup or for view-pooling scenarios.
* Renamed the ControllerBuilder method DisposeController to ReleaseController to fit with the pattern that is established for view engines.
* Removed most of the methods on the HtmlHelper class, converting them to extension methods of the HtmlHelper class instead. These methods exist in a new namespace (System.Web.Mvc.Html). If you are migrating from Preview 5, you must add the following element to the namespaces section of the Web.config file:
1. <add namespace="System.Web.Mvc.Html"/>

This makes it possible for you to completely replace our helper methods with your own.

* Changed the default model binder (DefaultModelBinder) to handle complex types. The IModelBinder interface has also been changed to accept a single parameter of type ModelBindingContext.
* Added a new HttpVerbs enumeration that contains the most commonly used HTTP verbs (GET, POST, PUT, DELETE, HEAD). Also added a constructor overload to AcceptVerbsAttribute that accepts the enumeration. The enumerated values can be combined. For example, the following snippet shows an action method that can respond to both POST and PUT requests.
1. [AcceptVerbs(HttpVerbs.Post | HttpVerbs.Put)]
2. public ActionResult Update() {
3. }

Because it is possible to respond to HTTP verbs that are not included in the enumeration, the AcceptVerbsAttribute retains the constructor that accepts an array of strings as a parameter.

* Modified the RadioButton helper method to ensure that every overload accepts a value. Because radio buttons are used to specify a choice from a set of possible values, specifying a value for a radio button is necessary.
* Made modifications and fixes to the default project template. This includes moving script files to a new Scripts folder. The default template uses the ModelState class to report validation errors.
* Changed action-method selection. If two action methods match a request, but only one of those has an attribute that derives from ActionMethodSelectorAttribute that matches the request, that action is invoked. In earlier releases, this scenario resulted in an exception.

For example, the following two action methods are in the same controller:

1. public ActionResult Edit() {
2. //...
3. }
4. [AcceptVerbs(HttpVerbs.Post)]
5. public ActionResult Edit(FormCollection form) {
6. //...
7. }

In Preview 5, a POST request for the Edit action would cause an exception, because two methods match the request. In the Beta, precedence is given to the method that matches the current request via the AcceptVerb attribute. In this example, the first method will handle any non-POST requests for the Edit action.

* Added an overload for the ViewDataDictionary.Eval method that accepts a format string.
* Removed the ViewName property from the ViewContext class.
* Added an IValueProvider interface for value providers, along with a default implementation, DefaultValueProvider. Value providers supply values that are used by the model binders when binding to a model object. The UpdateModel method of the Controller class has been updated to allow you to specify a custom value provider.

## Bug Fixes

* Fixed a bug in which the ignore-routes setting (created by using the IgnoreRoute extension method) affected URL generation.
* Fixed a view engine caching bug when the application is not in debug mode (that is, when debug="false" is set in the Web.config file). This bug occurred if different action methods in different controllers had the same name. In that case, an action method could render the view for the wrong controller.
* Fixed a bug in OutputCacheAttribute in which cached authenticated content did not require authentication. Even though the content is cached, if it requires authentication, the user should be required to authenticate first before seeing the cached content.
* Fixed a bug in which RenderPartial does not work when tracing is turned on.
* Fixed a bug in the Html.TextArea helper method in which an overload was not looking in ViewData for its value when the provided value is null.
* Fixed the OutputCacheAttribute.CacheProfile property so that it works in Medium Trust.

## Upgrading from CodePlex Preview 5 to Beta

There are not many changes between ASP.NET MVC Preview 5 and Beta. However, you will need to make a few changes to your applications after you install the Beta release. Most of these changes are apparent when you try to compile your application by using the latest binaries, so we do not list every possible change. The compiler will guide you there. The following list describes some of the changes that you must make.

* Update the references to the following assemblies to point to the new Beta versions of the assemblies:

System.Web.Abstractions.dll

System.Web.Routing.dll

System.Web.Mvc.dll

By default, these assemblies are located in the following folder:

%ProgramFiles%\Microsoft ASP.NET\ASP.NET MVC Beta

* In the Web.config namespaces section, add the following namespace entry if it is not there already:
1. <add namespace="System.Web.Mvc.Html"/>
* The Form HTML helper was renamed to BeginForm.
* After you have made these changes, compile your application and resolve any compilation errors. Most of the errors will be the result of one of the breaking changes listed earlier.