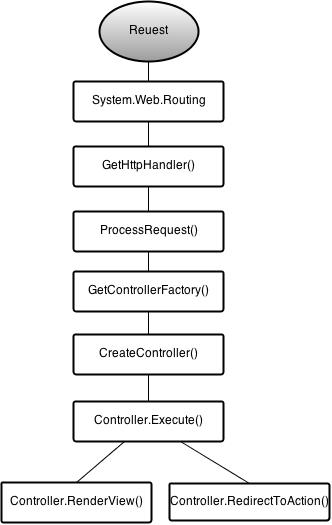
ASP.NET Routing

<http://msdn.microsoft.com/en-us/library/cc668201(v=vs.100).aspx#adding_constraints_to_routes>

**.NET Framework 4**

[Other Versions](javascript:;)



37 out of 46 rated this helpful - [Rate this topic](http://msdn.microsoft.com/en-us/library/cc668201(v=vs.100).aspx#feedback)

ASP.NET routing enables you to use URLs that do not have to map to specific files in a Web site. Because the URL does not have to map to a file, you can use URLs that are descriptive of the user's action and therefore are more easily understood by users.

The ASP.NET MVC framework and ASP.NET Dynamic Data extend routing to provide features that are used only in MVC applications and in Dynamic Data applications. For more information about MVC, see [ASP.NET MVC 3](http://go.microsoft.com/fwlink/?LinkId=216782). For more information about Dynamic Data, see [ASP.NET Dynamic Data Content Map](http://msdn.microsoft.com/en-us/library/cc488545(v=vs.100).aspx).

In an ASP.NET application that does not use routing, an incoming request for a URL typically maps to a physical file that handles the request, such as an .aspx file. For example, a request for http://server/application/Products.aspx?id=4 maps to a file that is named Products.aspx that contains code and markup for rendering a response to the browser. The Web page uses the query string value of id=4 to determine what type of content to display.

In ASP.NET routing, you can define URL patterns that map to request-handler files, but that do not necessarily include the names of those files in the URL. In addition, you can include placeholders in a URL pattern so that variable data can be passed to the request handler without requiring a query string.

For example, in the request for http://server/application/Products/show/beverages, the routing parser can pass the values Products, show, and beveragesto the page handler. In this example, if the route is defined by using the URL pattern server/application/{area}/{action}/{category}, the page handler would receive a dictionary collection in which the value associated with the key area is Products, the value for the key action is show, and the value for the key category isbeverages. In a request that is not managed by URL routing, the /Products/show/beverages fragment would be interpreted as the path of a file in the application.

This topic contains the following sections:

* [Routes](http://msdn.microsoft.com/en-us/library/cc668201(v=vs.100).aspx#routes)
* [URL Patterns](http://msdn.microsoft.com/en-us/library/cc668201(v=vs.100).aspx#url_patterns)
* [Adding Routes to a Web Forms Application](http://msdn.microsoft.com/en-us/library/cc668201(v=vs.100).aspx#adding_routes_to_a_web_forms_application)
* [Adding Routes to an MVC Application](http://msdn.microsoft.com/en-us/library/cc668201(v=vs.100).aspx#adding_routes_to_an_mvc_application)
* [Setting Default Values for URL Parameters](http://msdn.microsoft.com/en-us/library/cc668201(v=vs.100).aspx#setting_default_values_for_url_parameters)
* [Handling a Variable Number of Segments in a URL Pattern](http://msdn.microsoft.com/en-us/library/cc668201(v=vs.100).aspx#handling_a_variable_number_of_segments_in_a_url_pattern)
* [Adding Constraints to Routes](http://msdn.microsoft.com/en-us/library/cc668201(v=vs.100).aspx#adding_constraints_to_routes)
* [Scenarios When Routing Is Not Applied](http://msdn.microsoft.com/en-us/library/cc668201(v=vs.100).aspx#scenarios_when_routing_is_not_applied)
* [How URLs Are Matched to Routes](http://msdn.microsoft.com/en-us/library/cc668201(v=vs.100).aspx#how_urls_are_matched_to_routes)
* [Creating URLs from Routes](http://msdn.microsoft.com/en-us/library/cc668201(v=vs.100).aspx#creating_urls_from_routes)
* [Accessing URL Parameters in a Routed Page](http://msdn.microsoft.com/en-us/library/cc668201(v=vs.100).aspx#accessing_url_parameters_in_a_routed_page)
* [Configuration Settings for Routing](http://msdn.microsoft.com/en-us/library/cc668201(v=vs.100).aspx#configuration_settings_for_routing)
* [ASP.NET Routing and Security](http://msdn.microsoft.com/en-us/library/cc668201(v=vs.100).aspx#aspnet_routing_and_security)
* [ASP.NET Web Forms and Route Security](http://msdn.microsoft.com/en-us/library/cc668201(v=vs.100).aspx#aspnet_webforms_and_route_security)
* [ASP.NET MVC and Route Security](http://msdn.microsoft.com/en-us/library/cc668201(v=vs.100).aspx#aspnet_mvc_and_route_security)
* [Class Reference](http://msdn.microsoft.com/en-us/library/cc668201(v=vs.100).aspx#class_reference)
* [ASP.NET Routing versus URL Rewriting](http://msdn.microsoft.com/en-us/library/cc668201(v=vs.100).aspx#aspnet_routing_versus_url_rewriting)

[Routes](javascript:void(0))

A *route* is a URL pattern that is mapped to a handler. The handler can be a physical file, such as an .aspx file in a Web Forms application. A handler can also be a class that processes the request, such as a controller in an MVC application. To define a route, you create an instance of the [Route](http://msdn.microsoft.com/en-us/library/system.web.routing.route(v=vs.100).aspx) class by specifying the URL pattern, the handler, and optionally a name for the route.

You add the route to the application by adding the [Route](http://msdn.microsoft.com/en-us/library/system.web.routing.route(v=vs.100).aspx) object to the static [Routes](http://msdn.microsoft.com/en-us/library/system.web.routing.routetable.routes(v=vs.100).aspx) property of the [RouteTable](http://msdn.microsoft.com/en-us/library/system.web.routing.routetable(v=vs.100).aspx) class. The [Routes](http://msdn.microsoft.com/en-us/library/system.web.routing.routetable.routes(v=vs.100).aspx) property is a [RouteCollection](http://msdn.microsoft.com/en-us/library/system.web.routing.routecollection(v=vs.100).aspx)object that stores all the routes for the application.

You typically do not have to write code to add routes in an MVC application. Visual Studio project templates for MVC include preconfigured URL routes. These are defined in the **MvcApplication** class, which is defined in the Global.asax file.

[URL Patterns](javascript:void(0))

A URL pattern can contain literal values and variable placeholders (referred to as *URL parameters*). The literals and placeholders are located in *segments* of the URL which are delimited by the slash (/) character.

When a request is made, the URL is parsed into segments and placeholders, and the variable values are provided to the request handler. This process is similar to the way the data in query strings is parsed and passed to the request handler. In both cases variable information is included in the URL and passed to the handler in the form of key-value pairs. For query strings both the keys and the values are in the URL. For routes, the keys are the placeholder names defined in the URL pattern, and only the values are in the URL.

In a URL pattern, you define placeholders by enclosing them in braces ( { and } ). You can define more than one placeholder in a segment, but they must be separated by a literal value. For example, {language}-{country}/{action} is a valid route pattern. However, {language}{country}/{action} is not a valid pattern, because there is no literal value or delimiter between the placeholders. Therefore, routing cannot determine where to separate the value for the languageplaceholder from the value for the country placeholder.

The following table shows valid route patterns and examples of URL requests that match the patterns.

|  |  |
| --- | --- |
| **Route definition** | **Example of matching URL** |
| {controller}/{action}/{id} | /Products/show/beverages |
| {table}/Details.aspx | /Products/Details.aspx |
| blog/{action}/{entry} | /blog/show/123 |
| {reporttype}/{year}/{month}/{day} | /sales/2008/1/5 |
| {locale}/{action} | /US/show |
| {language}-{country}/{action} | /en-US/show |

Typical URL Patterns in MVC Applications

URL patterns for routes in MVC applications typically include {controller} and {action} placeholders.

When a request is received, it is routed to the [UrlRoutingModule](http://msdn.microsoft.com/en-us/library/system.web.routing.urlroutingmodule(v=vs.100).aspx) object and then to the [MvcHandler](http://msdn.microsoft.com/en-us/library/system.web.mvc.mvchandler(v=vs.100).aspx) HTTP handler. The [MvcHandler](http://msdn.microsoft.com/en-us/library/system.web.mvc.mvchandler(v=vs.100).aspx) HTTP handler determines which controller to invoke by adding the suffix "Controller" to the controller value in the URL to determine the type name of the controller that will handle the request. The action value in the URL determines which action method to call.

For example, a URL that includes the URL path /Products is mapped to a controller named ProductsController. The value in the *action* parameter is the name of the action method that is called. A URL that includes the URL path /Products/show would result in a call to the Show method of the ProductsController class.

The following table shows the default URL patterns, and it shows examples of URL requests that are handled by the default routes.

|  |  |
| --- | --- |
| **Default URL pattern** | **Examples of matching URL** |
| {controller}/{action}/{id} | http://server/application/Products/show/beverages |
| {resource}.axd/{\*pathInfo} | http://server/application/WebResource.axd?d=... |

The route with the pattern {resource}.axd/{\*pathInfo} is included to prevent requests for the Web resource files such as WebResource.axd or ScriptResource.axd from being passed to a controller.

For IIS 7.0, no file-name extension is needed. For IIS 6.0, you must add the .mvc file-name extension to the URL pattern, as in the following example:

{controller}.mvc/{action}/{id}

[Adding Routes to a Web Forms Application](javascript:void(0))

In a Web Forms application, you create routes by using the [MapPageRoute(String, String, String)](http://msdn.microsoft.com/en-us/library/dd992955(v=vs.100).aspx) method of the [RouteCollection](http://msdn.microsoft.com/en-us/library/system.web.routing.routecollection(v=vs.100).aspx) class. The [MapPageRoute](http://msdn.microsoft.com/en-us/library/system.web.routing.routecollection.mappageroute(v=vs.100).aspx) method creates a [Route](http://msdn.microsoft.com/en-us/library/system.web.routing.route(v=vs.100).aspx) object and adds it to the [RouteCollection](http://msdn.microsoft.com/en-us/library/system.web.routing.routecollection(v=vs.100).aspx) object. You specify properties for the [Route](http://msdn.microsoft.com/en-us/library/system.web.routing.route(v=vs.100).aspx) object in parameters that you pass to the [MapPageRoute](http://msdn.microsoft.com/en-us/library/system.web.routing.routecollection.mappageroute(v=vs.100).aspx)method.

Typically, you add routes in a method that is called from the handler for the **Application\_Start** event in the Global.asax file. This approach makes sure that the routes are available when the application starts. It also enables you to call the method directly when you unit-test the application. If you want to call a method directly when you unit-test the application, the method that registers the routes must be static (**Shared** in Visual Basic) and must have a [RouteCollection](http://msdn.microsoft.com/en-us/library/system.web.routing.routecollection(v=vs.100).aspx) parameter.

The following example shows code from a Global.asax file that adds a [Route](http://msdn.microsoft.com/en-us/library/system.web.routing.route(v=vs.100).aspx) object that defines two URL parameters named action and categoryName. URLs that have the specified pattern are directed to the physical page named Categories.aspx.

C#

[**VB**](http://msdn.microsoft.com/en-us/library/cc668201(v=vs.100).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-2)

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

{

RegisterRoutes(RouteTable.Routes);

}

public static void RegisterRoutes(RouteCollection routes)

{

routes.MapPageRoute("",

"Category/{action}/{categoryName}",

"~/categoriespage.aspx");

}

[Adding Routes to an MVC Application](javascript:void(0))

If you adopt the MVC convention of implementing controllers by creating classes that derive from the [ControllerBase](http://msdn.microsoft.com/en-us/library/system.web.mvc.controllerbase(v=vs.100).aspx) class and giving them names that end with "Controller", you do not need to manually add routes in an MVC application. The preconfigured routes will invoke the action methods that you implement in the controller classes.

If you want to add custom routes in an MVC application, you use the [MapRoute(RouteCollection, String, String)](http://msdn.microsoft.com/en-us/library/dd470521(v=vs.100).aspx) method instead of the [MapPageRoute(String, String, String)](http://msdn.microsoft.com/en-us/library/dd992955(v=vs.100).aspx) method.

The following example shows the code that creates default MVC routes in the Global.asax file, as defined in the Visual Studio project template for MVC applications.

C#

[**VB**](http://msdn.microsoft.com/en-us/library/cc668201(v=vs.100).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-3)

public class MvcApplication : System.Web.HttpApplication

{

public static void RegisterRoutes(RouteCollection routes)

{

routes.IgnoreRoute("{resource}.axd/{\*pathInfo}");

routes.MapRoute(

"Default", // Route name

"{controller}/{action}/{id}", // URL with parameters

new { controller = "Home", action = "Index", id = "" } // Parameter defaults

);

}

protected void Application\_Start()

{

RegisterRoutes(RouteTable.Routes);

}

}

[Setting Default Values for URL Parameters](javascript:void(0))

When you define a route, you can assign a default value for a parameter. The default value is used if a value for that parameter is not included in the URL. You set default values for a route by assigning a dictionary object to the [Defaults](http://msdn.microsoft.com/en-us/library/system.web.routing.route.defaults(v=vs.100).aspx) property of the [Route](http://msdn.microsoft.com/en-us/library/system.web.routing.route(v=vs.100).aspx) class. The following example shows how to add a route that has default values, by using the [MapPageRoute(String, String, String, Boolean, RouteValueDictionary)](http://msdn.microsoft.com/en-us/library/dd784594(v=vs.100).aspx) method.

C#

[**VB**](http://msdn.microsoft.com/en-us/library/cc668201(v=vs.100).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-4)

void Application\_Start(object sender, EventArgs e)

{

RegisterRoutes(RouteTable.Routes);

}

public static void RegisterRoutes(RouteCollection routes)

{

routes.MapPageRoute("",

"Category/{action}/{categoryName}",

"~/categoriespage.aspx",

true,

new RouteValueDictionary

{{"categoryName", "food"}, {"action", "show"}});

}

When ASP.NET routing handles a URL request, the route definition shown in the example (with default values of food for categoryName and show for action) produces the results that are listed in the following table.

|  |  |
| --- | --- |
| **URL** | **Parameter values** |
| /Category | *action* = "show" (default value)  *categoryName* = "food" (default value) |
| /Category/add | *action* = "add"  *categoryName* = "food" (default value) |
| /Category/add/beverages | *action* = "add"  *categoryName*= "beverages" |

For MVC applications, overloads of the [RouteCollectionExtensions.MapRoute](http://msdn.microsoft.com/en-us/library/system.web.mvc.routecollectionextensions.maproute(v=vs.100).aspx) method, such as [MapRoute(RouteCollection, String, String, Object, Object)](http://msdn.microsoft.com/en-us/library/dd460520(v=vs.100).aspx), enable you to specify defaults.

[Handling a Variable Number of Segments in a URL Pattern](javascript:void(0))

Sometimes you have to handle URL requests that contain a variable number of URL segments. When you define a route, you can specify that if a URL has more segments than there are in the pattern, the extra segments are considered to be part of the last segment. To handle additional segments in this manner you mark the last parameter with an asterisk (\*). This is referred to as a *catch-all* parameter. A route with a catch-all parameter will also match URLs that do not contain any values for the last parameter. The following example shows a route pattern that matches an unknown number of segments.

query/{queryname}/{\*queryvalues}

When ASP.NET routing handles a URL request, the route definition shown in the example produces the results that are listed in the following table.

|  |  |
| --- | --- |
| **URL** | **Parameter values** |
| /query/select/bikes/onsale | *queryname* = "select"  *queryvalues* = "bikes/onsale" |
| /query/select/bikes | *queryname* = "select"  *queryvalues* = "bikes" |
| /query/select | *queryname* = "select"  *queryvalues* = Empty string |

[Adding Constraints to Routes](javascript:void(0))

In addition to matching a URL request to a route definition by the number of parameters in the URL, you can specify that values in the parameters meet certain constraints. If a URL contains values that are outside the constraints for a route, that route is not used to handle the request. You add constraints to make sure that the URL parameters contain values that will work in your application.

Constraints are defined by using regular expressions or by using objects that implement the [IRouteConstraint](http://msdn.microsoft.com/en-us/library/system.web.routing.irouteconstraint(v=vs.100).aspx) interface. When you add the route definition to the[Routes](http://msdn.microsoft.com/en-us/library/system.web.routing.routetable.routes(v=vs.100).aspx) collection, you add constraints by creating a [RouteValueDictionary](http://msdn.microsoft.com/en-us/library/system.web.routing.routevaluedictionary(v=vs.100).aspx) object that contains the verification test. The key in the dictionary identifies the parameter that the constraint applies to. The value in the dictionary can be either a string that represents a regular expression or an object that implements the[IRouteConstraint](http://msdn.microsoft.com/en-us/library/system.web.routing.irouteconstraint(v=vs.100).aspx) interface.

If you provide a string, routing treats the string as a regular expression and checks whether the parameter value is valid by calling the [IsMatch](http://msdn.microsoft.com/en-us/library/system.text.regularexpressions.regex.ismatch(v=vs.100).aspx) method of the[Regex](http://msdn.microsoft.com/en-us/library/system.text.regularexpressions.regex(v=vs.100).aspx) class. The regular expression is always treated as case-insensitive. For more information, see [.NET Framework Regular Expressions](http://msdn.microsoft.com/en-us/library/hs600312(v=vs.100).aspx).

If you provide an [IRouteConstraint](http://msdn.microsoft.com/en-us/library/system.web.routing.irouteconstraint(v=vs.100).aspx) object, ASP.NET routing checks whether the parameter value is valid by calling the [Match](http://msdn.microsoft.com/en-us/library/system.web.routing.irouteconstraint.match(v=vs.100).aspx) method of the [IRouteConstraint](http://msdn.microsoft.com/en-us/library/system.web.routing.irouteconstraint(v=vs.100).aspx)object. The [Match](http://msdn.microsoft.com/en-us/library/system.web.routing.irouteconstraint.match(v=vs.100).aspx) method returns a Boolean value that indicates whether the parameter value is valid.

The following example shows how to use the [MapPageRoute](http://msdn.microsoft.com/en-us/library/system.web.routing.routecollection.mappageroute(v=vs.100).aspx) method to create a route that has constraints that limit what values can be included in the *locale* and*year* parameters. (In an MVC application, you would use the [MapRoute](http://msdn.microsoft.com/en-us/library/system.web.mvc.routecollectionextensions.maproute(v=vs.100).aspx) method.)

C#

[**VB**](http://msdn.microsoft.com/en-us/library/cc668201(v=vs.100).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-5)

public static void RegisterRoutes(RouteCollection routes)

{

routes.MapPageRoute("",

"Category/{action}/{categoryName}",

"~/categoriespage.aspx",

true,

new RouteValueDictionary

{{"categoryName", "food"}, {"action", "show"}},

new RouteValueDictionary

{{"locale", "[a-z]{2}-[a-z]{2}"},{"year", @"\d{4}"}}

);

}

When routing handles a URL request, the route definition shown in the previous example produces the results that are listed in the following table.

|  |  |
| --- | --- |
| **URL** | **Result** |
| /US | No match. Both *locale* and *year* are required. |
| /US/08 | No match. The constraint on *year* requires 4 digits. |
| /US/2008 | *locale* = "US"  *year* = "2008" |

[Scenarios When Routing Is Not Applied](javascript:void(0))

Under some circumstances, ASP.NET routing does not handle a request even when is enabled for the Web site. This section describes some scenarios in which routing does not handle the request.

A Physical File is Found that Matches the URL Pattern

By default, routing does not handle requests that map to an existing physical file on the Web server. For example, a request forhttp://server/application/Products/Beverages/Coffee.aspx is not handled by routing if a physical file exists at Products/Beverages/Coffee.aspx. Routing does not handle the request even if it matches a defined pattern, such as {controller}/{action}/{id}.

If you want routing to handle all requests, even requests that point to files, you can override the default behavior by setting the [RouteExistingFiles](http://msdn.microsoft.com/en-us/library/system.web.routing.routecollection.routeexistingfiles(v=vs.100).aspx) property of the[RouteCollection](http://msdn.microsoft.com/en-us/library/system.web.routing.routecollection(v=vs.100).aspx) object to **true**. When you set this value to **true**, all requests that match a defined pattern are handled by routing.

Routing Is Explicitly Disabled for a URL Pattern

You can also specify that routing should not handle certain URL requests. You prevent routing from handling certain requests by defining a route and specifying that the [StopRoutingHandler](http://msdn.microsoft.com/en-us/library/system.web.routing.stoproutinghandler(v=vs.100).aspx) class should be used to handle that pattern. When a request is handled by a [StopRoutingHandler](http://msdn.microsoft.com/en-us/library/system.web.routing.stoproutinghandler(v=vs.100).aspx) object, the [StopRoutingHandler](http://msdn.microsoft.com/en-us/library/system.web.routing.stoproutinghandler(v=vs.100).aspx)object blocks any additional processing of the request as a route. Instead, the request is processed as an ASP.NET page, Web service, or other ASP.NET endpoint. You can use the [RouteCollection.Ignore](http://msdn.microsoft.com/en-us/library/system.web.routing.routecollection.ignore(v=vs.100).aspx) method (or [RouteCollectionExtensions.IgnoreRoute](http://msdn.microsoft.com/en-us/library/system.web.mvc.routecollectionextensions.ignoreroute(v=vs.100).aspx) for MVC applications) to create routes that use the[StopRoutingHandler](http://msdn.microsoft.com/en-us/library/system.web.routing.stoproutinghandler(v=vs.100).aspx) class. The following example shows how to prevent routing from handling requests for the WebResource.axd file.

C#

[**VB**](http://msdn.microsoft.com/en-us/library/cc668201(v=vs.100).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-6)

public static void RegisterRoutes(RouteCollection routes)

{

routes.Ignore("{resource}.axd/{\*pathInfo}");

}

[How URLs Are Matched to Routes](javascript:void(0))

When routing handles URL requests, it tries to match the URL of the request to a route. Matching a URL request to a route depends on all the following conditions:

* The route patterns that you have defined or the default route patterns, if any, that are included in your project type.
* The order in which you added them to the [Routes](http://msdn.microsoft.com/en-us/library/system.web.routing.routetable.routes(v=vs.100).aspx) collection.
* Any default values that you have provided for a route.
* Any constraints that you have provided for a route.
* Whether you have defined routing to handle requests that match a physical file.

To avoid having the wrong handler handle a request, you must consider all these conditions when you define routes. The order in which [Route](http://msdn.microsoft.com/en-us/library/system.web.routing.route(v=vs.100).aspx) objects appear in the [Routes](http://msdn.microsoft.com/en-us/library/system.web.routing.routetable.routes(v=vs.100).aspx) collection is significant. Route matching is tried from the first route to the last route in the collection. When a match occurs, no more routes are evaluated. In general, add routes to the [Routes](http://msdn.microsoft.com/en-us/library/system.web.routing.routetable.routes(v=vs.100).aspx) property in order from the most specific route definitions to least specific ones.

For example, suppose that you add routes with the following patterns:

* Route 1 is set to {controller}/{action}/{id}
* Route 2 is set to products/show/{id}

Route 2 will never handle a request because Route 1 is evaluated first, and it will always match requests that could also work for Route 2. A request forhttp://server/application/products/show/bikes seems to match Route 2 more closely, but it is handled by Route 1 with the following values:

* controller is products.
* action is show.
* id is bikes.

Default values are used if a parameter is missing from the request. Therefore, they can cause a route to match a request that you did not expect. For example, suppose that you add routes with the following patterns:

* Route 1: {report}/{year}/{month}, with default values for year and month.
* Route 2: {report}/{year}, with a default value for year.

Route 2 will never handle a request. Route 1 might be intended for a monthly report, and Route 2 might be intended for an annual report. However, the default values in Route 1 mean that it will match any request that could also work for Route 2.

You can avoid ambiguity in the patterns by including constants, such as annual/{report}/{year} and monthly/{report}/{year}/{month}.

If a URL does not match any [Route](http://msdn.microsoft.com/en-us/library/system.web.routing.route(v=vs.100).aspx) object that is defined in the [RouteTable](http://msdn.microsoft.com/en-us/library/system.web.routing.routetable(v=vs.100).aspx) collection, ASP.NET routing does not process the request. Instead, processing is passed to an ASP.NET page, Web service, or other ASP.NET endpoint.

[Creating URLs from Routes](javascript:void(0))

If you want to create hyperlinks to pages in your site, you can use the URL patterns to programmatically create URLs that correspond to the routes. When you change the patterns, URLs matching the new patterns will be generated automatically. For information about how to generate URLs in code or in markup, see [How to: Construct URLs from Routes](http://msdn.microsoft.com/en-us/library/cc668176(v=vs.100).aspx).

[Accessing URL Parameters in a Routed Page](javascript:void(0))

In the handler for a routed page request you can access the values that are passed in the URL placeholders by using code or markup. For more information, see[How to: Access URL Parameters in a Routed Page](http://msdn.microsoft.com/en-us/library/dd535620(v=vs.100).aspx).

For MVC applications, the MVC framework automatically handles the values passed in URL placeholders. For more information, see [Passing Data in an ASP.NET MVC Application](http://msdn.microsoft.com/en-us/library/dd394711(v=vs.100).aspx).

[Configuration Settings for Routing](javascript:void(0))

In ASP.NET, you can add the following configuration setting to applications in order to support routing:

xmlLang

<configuration>

<system.webServer>

<modules runAllManagedModulesForAllRequests="true">

<!-- more -->

</modules>

</system.webServer>

</configuration>

When **runAllManagedModulesForAllRequests** is true, a URL like http://mysite/myapp/home goes to ASP.NET, even though there is no .aspx, .mvc, or similar extension on the URL.

However, an update that was made to IIS 7 makes the **runAllManagedModulesForAllRequests** setting unnecessary and supports ASP.NET routing natively. (For information about the update, see the Microsoft Support article [An update is available that enables certain IIS 7.0 or IIS 7.5 handlers to handle requests whose URLs do not end with a period](http://support.microsoft.com/kb/980368).)

If your website is running on IIS 7 and if IIS has been updated, you do not need to set **runAllManagedModulesForAllRequests** to true. In fact, setting it to true is not recommended, because it adds unnecessary processing overhead to request. When this setting is true, all requests, including those for .htm, .jpg, and other static files, also go through the ASP.NET request pipeline.

The default setting for **runAllManagedModulesForAllRequests** is false. If the site's configuration file does not explicitly set**runAllManagedModulesForAllRequests** to true, when you then run the website on Windows 7 without SP1 installed, IIS 7 will not include the required update. As a consequence, routing will not work and you will see errors. If you have a problem where routing does not work, you can do the following:

* Update Windows 7 to SP1, which will add the update to IIS 7.
* Install the update that's described in the Microsoft Support article listed previously.
* Set **runAllManagedModulesForAllRequests** to true in that website's Web.config file. Note that this will add some overhead to requests.

[ASP.NET Routing and Security](javascript:void(0))

Authorization rules can be applied to only the route URL or to both the route URL and to the physical URL that it maps to. For example, authorization rules might specify that all users can access URLs that begin with Category but only administrators can access the Categories.aspx page. If the route URL patterncontoso.com/Category/{controller}/{action} maps to the physical URL contoso.com/Categoriespage.aspx, and you apply authorization rules only to the route URL, all users will be allowed access to Categoriespage.aspx when it is requested by using a route URL. However, only administrators will have access to it when it is requested by using a physical URL.

By default, authorization rules are applied to both route URL and physical URL. For more information, see the [PageRouteHandler.CheckPhysicalUrlAccess](http://msdn.microsoft.com/en-us/library/system.web.routing.pageroutehandler.checkphysicalurlaccess(v=vs.100).aspx) property.

ASP.NET Web Forms and Route Security

In an ASP.NET Web Forms application, you should not depend solely on route authorization rules to secure your site, because that might leave some physical URLs unprotected.

ASP.NET MVC and Route Security

You cannot use routing or web.config files to secure an MVC application. The only supported way to secure an MVC application is to apply the [AuthorizeAttribute](http://msdn.microsoft.com/en-us/library/system.web.mvc.authorizeattribute(v=vs.100).aspx)attribute to each controller and use the AllowAnonymousAttribute attribute on the login and register actions. For more information, see [Securing your ASP.NET MVC 4 App and the new AllowAnonymous Attribute](http://blogs.msdn.com/b/rickandy/archive/2012/03/23/securing-your-asp-net-mvc-4-app-and-the-new-allowanonymous-attribute.aspx) on Rick Anderson's blog. To secure an ASP.NET MVC 3 application, see [Securing your ASP.NET MVC 3 Application](http://blogs.msdn.com/b/rickandy/archive/2011/05/02/securing-your-asp-net-mvc-3-application.aspx), also on Rick Anderson's blog.

[Class Reference](javascript:void(0))

The following table lists the key server classes for ASP.NET routing.

|  |  |
| --- | --- |
| **Class** | **Description** |
| [Route](http://msdn.microsoft.com/en-us/library/system.web.routing.route(v=vs.100).aspx) | Represents a route in a Web Forms or MVC application. |
| [DynamicDataRoute](http://msdn.microsoft.com/en-us/library/system.web.dynamicdata.dynamicdataroute(v=vs.100).aspx) | Represents a route in a Dynamic Data application. |
| [RouteBase](http://msdn.microsoft.com/en-us/library/system.web.routing.routebase(v=vs.100).aspx) | Serves as the base class for all classes that represent an ASP.NET route. |
| [RouteTable](http://msdn.microsoft.com/en-us/library/system.web.routing.routetable(v=vs.100).aspx) | Stores the routes for an application. |
| [RouteCollection](http://msdn.microsoft.com/en-us/library/system.web.routing.routecollection(v=vs.100).aspx) | Provides methods that enable you to manage a collection of routes. |
| [RouteCollectionExtensions](http://msdn.microsoft.com/en-us/library/system.web.mvc.routecollectionextensions(v=vs.100).aspx) | Provides additional methods that enable you to manage a collection of routes in MVC applications. |
| [RouteData](http://msdn.microsoft.com/en-us/library/system.web.routing.routedata(v=vs.100).aspx) | Contains the values for a requested route. |
| [RequestContext](http://msdn.microsoft.com/en-us/library/system.web.routing.requestcontext(v=vs.100).aspx) | Contains information about the HTTP request that corresponds to a route. |
| [StopRoutingHandler](http://msdn.microsoft.com/en-us/library/system.web.routing.stoproutinghandler(v=vs.100).aspx) | Provides a way to specify that ASP.NET routing should not handle requests for a URL pattern. |
| [PageRouteHandler](http://msdn.microsoft.com/en-us/library/system.web.routing.pageroutehandler(v=vs.100).aspx) | Provides a way to define routes for Web Forms applications. |
| [RouteValueDictionary](http://msdn.microsoft.com/en-us/library/system.web.routing.routevaluedictionary(v=vs.100).aspx) | Provides a way to store route [Constraints](http://msdn.microsoft.com/en-us/library/system.web.routing.route.constraints(v=vs.100).aspx), [Defaults](http://msdn.microsoft.com/en-us/library/system.web.routing.route.defaults(v=vs.100).aspx), and [DataTokens](http://msdn.microsoft.com/en-us/library/system.web.routing.route.datatokens(v=vs.100).aspx) objects. |
| [VirtualPathData](http://msdn.microsoft.com/en-us/library/system.web.routing.virtualpathdata(v=vs.100).aspx) | Provides a way to generate URLs from route information. |

[ASP.NET Routing versus URL Rewriting](javascript:void(0))

ASP.NET routing differs from URL rewriting. URL rewriting processes incoming requests by actually changing the URL before it sends the request to the Web page. For example, an application that uses URL rewriting might change a URL from /Products/Widgets/ to /Products.aspx?id=4. Also, URL rewriting typically does not have an API for creating URLs that are based on your patterns. In URL rewriting, if you change a URL pattern, you must manually update all hyperlinks that contain the original URL.

With ASP.NET routing, the URL is not changed when an incoming request is handled, because routing can extract values from the URL. When you have to create a URL, you pass parameter values into a method that generates the URL for you. To change the URL pattern, you change it in one location, and all the links that you create in the application that are based on that pattern will automatically use the new pattern.

[See Also](javascript:void(0))

Tasks

[Walkthrough: Using ASP.NET Routing in a Web Forms Application](http://msdn.microsoft.com/en-us/library/dd329551(v=vs.100).aspx)

[How to: Construct URLs from Routes](http://msdn.microsoft.com/en-us/library/cc668176(v=vs.100).aspx)

[How to: Access URL Parameters in a Routed Page](http://msdn.microsoft.com/en-us/library/dd535620(v=vs.100).aspx)

[How to: Define Routes for Web Forms Applications](http://msdn.microsoft.com/en-us/library/cc668177(v=vs.100).aspx)

Reference

[Route](http://msdn.microsoft.com/en-us/library/system.web.routing.route(v=vs.100).aspx)

[PageRouteHandler](http://msdn.microsoft.com/en-us/library/system.web.routing.pageroutehandler(v=vs.100).aspx)

[MvcRouteHandler](http://msdn.microsoft.com/en-us/library/system.web.mvc.mvcroutehandler(v=vs.100).aspx)

[RouteData](http://msdn.microsoft.com/en-us/library/system.web.ui.page.routedata(v=vs.100).aspx)

Concepts

[.NET Framework 4 Migration Issues](http://msdn.microsoft.com/en-us/library/ee941656(v=vs.100).aspx)

[ASP.NET Infrastructure](http://msdn.microsoft.com/en-us/library/bb398900(v=vs.100).aspx)