# Chapter 3. Controllers and Routing

# 1 Controllers and Action Methods in ASP.NET MVC Applications

The ASP.NET MVC framework maps URLs to classes that are referred to as controllers. Controllers process incoming requests, handle user input and interactions, and execute appropriate application logic. A controller class typically calls a separate view component to generate the HTML markup for the request.

The base class for all controllers is the [ControllerBase](http://msdn.microsoft.com/en-us/library/system.web.mvc.controllerbase(v=vs.98).aspx) class, which provides general MVC handling. The [Controller](http://msdn.microsoft.com/en-us/library/system.web.mvc.controller(v=vs.98).aspx) class inherits from ControllerBase and is the default implement of a controller. The Controller class is responsible for the following processing stages:

* Locating the appropriate action method to call and validating that it can be called.
* Getting the values to use as the action method's arguments.
* Handling all errors that might occur during the execution of the action method.
* Providing the default [WebFormViewEngine](http://msdn.microsoft.com/en-us/library/system.web.mvc.webformviewengine(v=vs.98).aspx) class for rendering ASP.NET page types (views).

|  |
| --- |
| **NoteNote:** |
| To help secure access to controllers and action methods, you can use the [AuthorizeAttribute](http://msdn.microsoft.com/en-us/library/system.web.mvc.authorizeattribute(v=vs.98).aspx) class. |

All controller classes must be named by using the "Controller" suffix. The following example shows the sample controller class, which is named HomeController. This controller class contains action methods that render view pages.

C#

[VB](http://msdn.microsoft.com/en-us/library/dd410269(v=vs.98).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-1)

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_7b32f27f-f4c0-4486-b983-64e0bc895f23');" \o "Copy to clipboard.)

[HandleError]

public class HomeController : Controller

{

public ActionResult Index()

{

ViewData["Message"] = "Welcome to ASP.NET MVC!";

return View();

}

public ActionResult About()

{

return View();

}

}

[Action Methods](javascript:void(0))

In ASP.NET applications that do not use the MVC framework, user interaction is organized around pages, and around raising and handling events from the page and from controls on the page. In contrast, user interaction with ASP.NET MVC applications is organized around controllers and action methods. The controller defines action methods. Controllers can include as many action methods as needed.

Action methods typically have a one-to-one mapping with user interactions. Examples of user interactions include entering a URL into the browser, clicking a link, and submitting a form. Each of these user interactions causes a request to be sent to the server. In each case, the URL of the request includes information that the MVC framework uses to invoke an action method.

When a user enters a URL into the browser, the MVC application uses routing rules that are defined in the Global.asax file to parse the URL and to determine the path of the controller. The controller then determines the appropriate action method to handle the request. By default, the URL of a request is treated as a sub-path that includes the controller name followed by the action name. For example, if a user enters the URL http://contoso.com/MyWebSite/Products/Categories, the sub-path is /Products/Categories. The default routing rule treats "Products" as the prefix name of the controller, which must end with "Controller" (such as ProductsController). It treats "Categories" as the name of the action. Therefore, the routing rule invokes the Categories method of the Products controller in order to process the request. If the URL ends with /Products/Detail/5, the default routing rule treats "Detail" as the name of the action, and the Detail method of the Products controller is invoked to process the request. By default, the value "5" in the URL will be passed to the Detail method as a parameter.

The following example shows a controller class that has a HelloWorld action method.

C#

[VB](http://msdn.microsoft.com/en-us/library/dd410269(v=vs.98).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-2)

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_ae670525-dfea-40ae-a769-222fc4a451fb');)

public class MyController : Controller

{

public ActionResult HelloWorld()

{

ViewData["Message"] = "Hello World!";

return View();

}

}

[ActionResult Return Type](javascript:void(0))

Most action methods return an instance of a class that derives from [ActionResult](http://msdn.microsoft.com/en-us/library/system.web.mvc.actionresult(v=vs.98).aspx). The ActionResult class is the base for all action results. However, there are different action result types, depending on the task that the action method is performing. For example, the most common action is to call the [View](http://msdn.microsoft.com/en-us/library/system.web.mvc.controller.view(v=vs.98).aspx) method. The View method returns an instance of the [ViewResult](http://msdn.microsoft.com/en-us/library/system.web.mvc.viewresult(v=vs.98).aspx) class, which is derived from ActionResult.

You can create action methods that return an object of any type, such as a string, an integer, or a Boolean value. These return types are wrapped in an appropriate ActionResult type before they are rendered to the response stream.

The following table shows the built-in action result types and the action helper methods that return them.

|  |  |  |
| --- | --- | --- |
| **Action Result** | **Helper Method** | **Description** |
| [ViewResult](http://msdn.microsoft.com/en-us/library/system.web.mvc.viewresult(v=vs.98).aspx) | [View](http://msdn.microsoft.com/en-us/library/system.web.mvc.controller.view(v=vs.98).aspx) | Renders a view as a Web page. |
| [PartialViewResult](http://msdn.microsoft.com/en-us/library/system.web.mvc.partialviewresult(v=vs.98).aspx) | [PartialView](http://msdn.microsoft.com/en-us/library/system.web.mvc.controller.partialview(v=vs.98).aspx) | Renders a partial view, which defines a section of a view that can be rendered inside another view. |
| [RedirectResult](http://msdn.microsoft.com/en-us/library/system.web.mvc.redirectresult(v=vs.98).aspx) | [Redirect](http://msdn.microsoft.com/en-us/library/system.web.mvc.controller.redirect(v=vs.98).aspx) | Redirects to another action method by using its URL. |
| [RedirectToRouteResult](http://msdn.microsoft.com/en-us/library/system.web.mvc.redirecttorouteresult(v=vs.98).aspx) | [RedirectToAction](http://msdn.microsoft.com/en-us/library/system.web.mvc.controller.redirecttoaction(v=vs.98).aspx)  [RedirectToRoute](http://msdn.microsoft.com/en-us/library/system.web.mvc.controller.redirecttoroute(v=vs.98).aspx) | Redirects to another action method. |
| [ContentResult](http://msdn.microsoft.com/en-us/library/system.web.mvc.contentresult(v=vs.98).aspx) | [Content](http://msdn.microsoft.com/en-us/library/system.web.mvc.controller.content(v=vs.98).aspx) | Returns a user-defined content type. |
| [JsonResult](http://msdn.microsoft.com/en-us/library/system.web.mvc.jsonresult(v=vs.98).aspx) | [Json](http://msdn.microsoft.com/en-us/library/system.web.mvc.controller.json(v=vs.98).aspx) | Returns a serialized JSON object. |
| [JavaScriptResult](http://msdn.microsoft.com/en-us/library/system.web.mvc.javascriptresult(v=vs.98).aspx) | [JavaScript](http://msdn.microsoft.com/en-us/library/system.web.mvc.controller.javascript(v=vs.98).aspx) | Returns a script that can be executed on the client. |
| [FileResult](http://msdn.microsoft.com/en-us/library/system.web.mvc.fileresult(v=vs.98).aspx) | [File](http://msdn.microsoft.com/en-us/library/system.web.mvc.controller.file(v=vs.98).aspx) | Returns binary output to write to the response. |
| [EmptyResult](http://msdn.microsoft.com/en-us/library/system.web.mvc.emptyresult(v=vs.98).aspx) | (None) | Represents a return value that is used if the action method must return a null result (void). |

[Marking Public Methods as Non-Action Methods](javascript:void(0))

By default, the MVC framework treats all public methods of a controller class as action methods. If your controller class contains a public method and you do not want it to be an action method, you must mark that method with the [NonActionAttribute](http://msdn.microsoft.com/en-us/library/system.web.mvc.nonactionattribute(v=vs.98).aspx) attribute.

The following example shows a method that is marked with the NonAction attribute.

C#

[VB](http://msdn.microsoft.com/en-us/library/dd410269(v=vs.98).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-3)

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_b0513f22-9e3a-4f16-bd13-b3be62f5a37f');)

[NonAction]

private void DoSomething()

{

// Method logic.

}

[Action Method Parameters](javascript:void(0))

By default, the values for action method parameters are retrieved from the request's data collection. The data collection includes name/values pairs for form data, query string values, and cookie values.

The controller class locates the action method and determines any parameter values for the action method, based on the [RouteData](http://msdn.microsoft.com/en-us/library/system.web.routing.routedata(v=vs.98).aspx) instance and based on the form data. If the parameter value cannot be parsed, and if the type of the parameter is a reference type or a nullable value type, null is passed as the parameter value. Otherwise, an exception is thrown.

There are several ways to access URL parameter values in the action methods of controller classes. The [Controller](http://msdn.microsoft.com/en-us/library/system.web.mvc.controller(v=vs.98).aspx) class exposes [Request](http://msdn.microsoft.com/en-us/library/system.web.mvc.controller.request(v=vs.98).aspx) and [Response](http://msdn.microsoft.com/en-us/library/system.web.mvc.controller.response(v=vs.98).aspx) properties that can be accessed in an action method. These properties have the same semantics as the [HttpRequest](http://msdn.microsoft.com/en-us/library/system.web.httprequest(v=vs.98).aspx) and [HttpResponse](http://msdn.microsoft.com/en-us/library/system.web.httpresponse(v=vs.98).aspx) objects that are already a part of ASP.NET. However, the Request and Response objects of the Controller class accept objects that implement the [HttpRequestBase](http://msdn.microsoft.com/en-us/library/system.web.httprequestbase(v=vs.98).aspx) and [HttpResponseBase](http://msdn.microsoft.com/en-us/library/system.web.httpresponsebase(v=vs.98).aspx) abstract classes instead of being sealed classes. These base classes make it easy to create mock objects, which in turn makes it easy to create unit tests for controller classes.

The following example shows how to use the Request object to retrieve a query-string value named id.

C#

[VB](http://msdn.microsoft.com/en-us/library/dd410269(v=vs.98).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-4)

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_18663db7-8ea5-4168-98ba-fdf72cefd2e8');)

public void Detail()

{

int id = Convert.ToInt32(Request["id"]);

}

[Automatically Mapping Action-Method Parameters](javascript:void(0))

The ASP.NET MVC framework can automatically map URL parameter values to parameter values for action methods. By default, if an action method takes a parameter, the MVC framework examines incoming request data and determines whether the request contains an HTTP request value with the same name. If so, the request value is automatically passed to the action method.

The following example shows a variation of the previous example. In this variation, the id parameter is assumed to map to a request value that is also named id. Because of this automatic mapping, the action method does not have to include code to get a parameter value from the request, and the parameter value is therefore easier to use.

C#

[VB](http://msdn.microsoft.com/en-us/library/dd410269(v=vs.98).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-5)

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_0a3bf446-23e6-4132-a535-ccf9c53294a5');)

public ResultAction Detail(int id)

{

ViewData["DetailInfo"] = id;

return View();

}

You can also embed parameter values as part of the URL instead of as query-string values. For example, instead of using the URL with a query string such as /Products/Detail?id=3, you can use a URL like /Products/Detail/3. The default route-mapping rule has the format /{controller}/{action}/{id}. If there is a URL sub-path after the controller and action names in the URL, it is treated as a parameter named id, and is automatically passed to the action method as a parameter value.

The MVC framework also supports optional arguments for action methods. Optional parameters in the MVC framework are handled by using nullable-type arguments for controller action methods. For example, if a method can take a date as part of the query string but you want to the default to be today's date if the query string parameter is missing, you can use code like that in the following example:

C#

[VB](http://msdn.microsoft.com/en-us/library/dd410269(v=vs.98).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-6)

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_04277772-f0aa-4915-9d31-05fd9ac776fa');)

public ActionResult ShowArticles(DateTime? date)

{

if(!date.HasValue)

{

date = DateTime.Now;

}

// ...

}

If the request includes a value for the date parameter, that value is passed to the ShowArticles method. If the request does not include a value for this parameter, the argument is null, and the controller can take whatever actions are required in order to handle the missing parameter.