**Creating an Action (C#)**

Learn how to add a new action to an ASP.NET MVC controller. Learn about the requirements for a method to be an action.

The goal of this tutorial is to explain how you can create a new controller action. You learn about the requirements of an action method. You also learn how to prevent a method from being exposed as an action.

**Adding an Action to a Controller**

You add a new action to a controller by adding a new method to the controller. For example, the controller in Listing 1 contains an action named Index() and an action named SayHello(). Both methods are exposed as actions.

**Listing 1 - Controllers\HomeController.cs**

C#Copy

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace MvcApplication1.Controllers

{

[HandleError]

public class HomeController : Controller

{

public ActionResult Index()

{

return View();

}

public string SayHello()

{

return "Hello!";

}

}

}

In order to be exposed to the universe as an action, a method must meet certain requirements:

* The method must be public.
* The method cannot be a static method.
* The method cannot be an extension method.
* The method cannot be a constructor, getter, or setter.
* The method cannot have open generic types.
* The method is not a method of the controller base class.
* The method cannot contain **ref** or **out** parameters.

Notice that there are no restrictions on the return type of a controller action. A controller action can return a string, a DateTime, an instance of the Random class, or void. The ASP.NET MVC framework will convert any return type that is not an action result into a string and render the string to the browser.

When you add any method that does not violate these requirements to a controller, the method is exposed as a controller action. Be careful here. A controller action can be invoked by anyone connected to the Internet. Do not, for example, create a DeleteMyWebsite() controller action.

**Preventing a Public Method from Being Invoked**

If you need to create a public method in a controller class and you don't want to expose the method as a controller action then you can prevent the method from being invoked by using the [NonAction] attribute. For example, the controller in Listing 2 contains a public method named CompanySecrets() that is decorated with the [NonAction] attribute.

**Listing 2 - Controllers\WorkController.cs**

C#Copy

using System.Web.Mvc;

namespace MvcApplication1.Controllers

{

public class WorkController : Controller

{

[NonAction]

public string CompanySecrets()

{

return "This information is secret.";

}

}

}

If you attempt to invoke the CompanySecrets() controller action by typing /Work/CompanySecrets into the address bar of your browser then you'll get the error message in Figure 1.

**Figure 01**: Invoking a NonAction method([Click to view full-size image](https://docs.microsoft.com/en-us/aspnet/mvc/overview/older-versions-1/controllers-and-routing/creating-an-action-cs/_static/image2.png))

[Previous](https://docs.microsoft.com/en-us/aspnet/mvc/overview/older-versions-1/controllers-and-routing/creating-a-controller-cs) [Next](https://docs.microsoft.com/en-us/aspnet/mvc/overview/older-versions-1/controllers-and-routing/asp-net-mvc-routing-overview-vb)

**Recommended content**

**[Controller.View Method (System.Web.Mvc)](https://docs.microsoft.com/en-us/dotnet/api/system.web.mvc.controller.view)**

Creates a ViewResult object using the view name, master-page name, and model that renders a view.

**[Controllers and Routing](https://docs.microsoft.com/en-us/aspnet/mvc/overview/older-versions-1/controllers-and-routing/)**

In this tutorial set, you learn about ASP.NET routing, which maps browser requests to ASP.NET MVC controller actions.

**[Creating a Controller (C#)](https://docs.microsoft.com/en-us/aspnet/mvc/overview/older-versions-1/controllers-and-routing/creating-a-controller-cs)**

In this tutorial, Stephen Walther demonstrates how you can add a controller to an ASP.NET MVC application.

**[ASP.NET MVC Controller Overview (C#)](https://docs.microsoft.com/en-us/aspnet/mvc/overview/older-versions-1/controllers-and-routing/aspnet-mvc-controllers-overview-cs)**

In this tutorial, Stephen Walther introduces you to ASP.NET MVC controllers. You learn how to create new controllers and return different types of action res...

**[ASP.NET MVC Routing Overview (C#)](https://docs.microsoft.com/en-us/aspnet/mvc/overview/older-versions-1/controllers-and-routing/asp-net-mvc-routing-overview-cs)**

In this tutorial, Stephen Walther shows how the ASP.NET MVC framework maps browser requests to controller actions.

**[Creating Custom Routes (C#)](https://docs.microsoft.com/en-us/aspnet/mvc/overview/older-versions-1/controllers-and-routing/creating-custom-routes-cs)**

Learn how to add custom routes to an ASP.NET MVC application. In this tutorial, you learn how to modify the default route table in the Global.asax file.

**[Creating a Route Constraint (C#)](https://docs.microsoft.com/en-us/aspnet/mvc/overview/older-versions-1/controllers-and-routing/creating-a-route-constraint-cs)**

In this tutorial, Stephen Walther demonstrates how you can control how browser requests match routes by creating route constraints with regular expressions.

**[ViewResult Class (System.Web.Mvc)](https://docs.microsoft.com/en-us/dotnet/api/system.web.mvc.viewresult)**

Represents a class that is used to render a view by using an IView instance that is returned by an IViewEngine object.

Show less