# return View() vs return RedirectToAction() vs return Redirect() vs return RedirectToRoute()

There are different ways for returning/rendering a view in MVC Razor. Many developers got confused when to use return View(), return RedirectToAction(), return Redirect() and return RedirectToRoute(). In this article, I would like to explain the difference among "return View()" and "return RedirectToAction()", "return Redirect()" and "return RedirectToRoute()".

**return View()**

This tells MVC to generate HTML to be displayed for the specified view and sends it to the browser. This acts like as Server.Transfer() in Asp.Net WebForm.

1. public ActionResult Index()
2. {
3. return View();
4. }
6. [HttpPost]
7. public ActionResult Index(string Name)
8. {
9. ViewBag.Message = "Hi, Dot Net Tricks";
10. //Like Server.Transfer() in Asp.Net WebForm
11. return View("MyIndex");
12. }
14. public ActionResult MyIndex()
15. {
16. ViewBag.Msg = ViewBag.Message; // Assigned value : "Hi, Dot Net Tricks"
17. return View("MyIndex");
18. }

What happens if we call the action method directly like return MyIndex(). It is simply a method call which returns a rendered view that is specified in MyIndex() action method.

1. public ActionResult Index()
2. {
3. return View();
4. }
6. [HttpPost]
7. public ActionResult Index(string Name)
8. {
9. ViewBag.Message = "Hi, Dot Net Tricks";
10. //Like Server.Transfer() in Asp.Net WebForm
11. return MyIndex();
12. }
14. public ActionResult MyIndex()
15. {
16. ViewBag.Msg = ViewBag.Message; // Assigned value : "Hi, Dot Net Tricks"
17. return View("MyIndex");
18. }

**return RedirectToAction()**

This tells MVC to redirect to specified action instead of rendering HTML. In this case, browser receives the redirect notification and make a new request for the specified action. This acts like as Response.Redirect() in Asp.Net WebForm.

Moreover, RedirectToAction construct a redirect url to a specific action/controller in your application and use the route table to generate the correct URL. RedirectToAction cause the browser to receive a 302 redirect within your application and gives you an easier way to work with your route table.

1. public ActionResult Index()
2. {
3. return View();
4. }
6. [HttpPost]
7. public ActionResult Index(string Name)
8. {
9. ViewBag.Message = "Hi, Dot Net Tricks";
10. //Like Response.Redirect() in Asp.Net WebForm
11. return RedirectToAction("MyIndex");
12. }
14. public ActionResult MyIndex()
15. {
16. ViewBag.Msg = ViewBag.Message; // Assigned value : Null
17. return View("MyIndex");
18. }

**return Redirect()**

This tells MVC to redirect to specified URL instead of rendering HTML. In this case, browser receives the redirect notification and make a new request for the specified URL. This also acts like as Response.Redirect() in Asp.Net WebForm. In this case, you have to specify the full URL to redirect.

Moreover, Redirect also cause the browser to receive a 302 redirect within your application, but you have to construct the URLs yourself.

1. public ActionResult Index()
2. {
3. return View();
4. }
6. [HttpPost]
7. public ActionResult Index(string Name)
8. {
9. ViewBag.Message = "Hi, Dot Net Tricks";
10. //Like Response.Redirect() in Asp.Net WebForm
11. return Redirect("Home/MyIndex");
12. }
14. public ActionResult MyIndex()
15. {
16. ViewBag.Msg = ViewBag.Message; // Assigned value : Null
17. return View("MyIndex");
18. }

**return RedirectToRoute()**

This tells MVC to look up the specifies route into the Route table that is is defined in global.asax and then redirect to that controller/action defined in that route. This also make a new request like RedirectToAction().

**Defined Route**

1. public static void RegisterRoutes(RouteCollection routes)
2. {
3. routes.MapRoute(
4. "MyRoute", // Route name
5. "Account/", // URL
6. new { controller = "Account", action = "Login"} // Parameter defaults
7. );
9. routes.MapRoute(
10. "Default", // Route name
11. "{controller}/{action}/{id}", // URL with parameters
12. new { controller = "Home", action = "MyIndex", id = UrlParameter.Optional } // Parameter defaults
13. );
14. }

**HomeController**

2. public ActionResult Index()
3. {
4. return View();
5. }
7. [HttpPost]
8. public ActionResult Index(string Name)
9. {
10. return RedirectToRoute("MyRoute");
11. }

**AccountController**

2. public ActionResult Login()
3. {

return View();

1. }

**Note**

1. Return View doesn't make a new requests, it just renders the view without changing URLs in the browser's address bar.
2. Return RedirectToAction makes a new requests and URL in the browser's address bar is updated with the generated URL by MVC.
3. Return Redirect also makes a new requests and URL in the browser's address bar is updated, but you have to specify the full URL to redirect
4. Between RedirectToAction and Redirect, best practice is to use RedirectToAction for anything dealing with your application actions/controllers. If you use Redirect and provide the URL, you'll need to modify those URLs manually when you change the route table.
5. RedirectToRoute redirects to a specific route defined in the Route table.