```
protected void Application_Start()
        {
            AreaRegistration.RegisterAllAreas();
            FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
            RouteConfig.RegisterRoutes(RouteTable.Routes);
            BundleConfig.RegisterBundles(BundleTable.Bundles);
        }
```

```
public static void RegisterRoutes(RouteCollection routes)
        {
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

            routes.MapMvcAttributeRoutes();
```

If there's any overlap between a traditional route and an attribute route, the first route registered wins

```
            routes.MapRoute(
                name: "Default",
                url: "{controller}/{action}/{id}",
                defaults: new { controller = "Home", action = "Index", id =
UrlParameter.Optional }
            );
        }
```

routes.MapMvcAttributeRoutes();

Kontrolerot ne se zema vo predvid

http://localhost:48447/da

```
public class HomeController : Controller

        public ActionResult Index()
        {
            return View();
        }
        [Route("da")]
        public ActionResult About()
        {
            ViewBag.Message = "Your application description page.";

            return View();
        }
```

[Route("home/{action}")]
public class HomeController : Controller
{.....

When you specify a route attribute at the action level, you're overriding anything specified at the controller level.

http://localhost:48447/da/1?num=2 na ova ke odgovara

```
[Route("da/{id}")]
        public ActionResult About(int id,int num)
        {
```

When an attribute route matches and an action method runs, the route parameters from the route are used by model binding to populate values for any method parameters with the same name. {id} vo id

When you define a route on the controller class, you can use a special route parameter named action
[Route("home/{action}")] It has the same effect
as your putting separate routes on each action and typing in the action name statically

Optional
Route atribute

```
  [Route("da/{id?}")]
        public ActionResult About(int ?id)
traditional
id = UrlParameter.Optional
```

Route Debugger, simply use NuGet to install it via the following command in the Package Manager Console window in Visual Studio: Install-Package RouteDebugger.