Sql server, .net and c# video tutorial

Free C#, .Net and Sql server video tutorial for beginners and intermediate programmers.

Attribute Routing in ASP.NET Core MVC

Suggested Videos

Part 30 - _ViewStart.cshtml in ASP.NET Core MVC | Text | Slides
Part 31 - _ViewImports.cshtml in ASP.NET Core MVC | Text | Slides
Part 32 - Routing in ASP.NET Core MVC | Text | Slides

In this video we will discuss **Attribute Routing in ASP.NET Core MVC**

Consider the following code in Configure() method of Startup.cs file. Notice we are using UseMvc() method without passing the default route template as a parameter.

```
public void Configure(IApplicationBuilder app, IHostingEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }

    app.UseStaticFiles();

    app.UseMvc();

    //app.UseMvc(routes =>
    //{
    //    routes.MapRoute("default", "{controller=Home}/{action=Index}/{id?}");
    //});
}
```

This means, at the moment our application does not have any routes configured and when we navigate to any of the following URLs we see 404 errors.

http://localhost:1234
http://localhost:1234/home
http://localhost:1234/home/index

Attribute Routing Example

With attribute routing, we use the Route attribute to define our routes. We could apply the Route attribute on the Controller or on the Controller Action Methods.

Consider the example below.

```
public class HomeController : Controller
{
```

```
    [Route("")]
    [Route("Home")]
    [Route("Home/Index")]
    public ViewResult Index()
    {
        return View();
    }
}
```

The Route() attribute is specified 3 times on the Index() action method. With each instance of the Route() attribute we specified a different route template. With these 3 route templates in place, the Index() action method of the HomeController will be executed for any of the following 3 URL paths.

/
/Home
/Home/Index

Attribute Routing Parameters

With conventional routing we can specify route parameter as part of the route template. We can do the same with attribute routing as well. Consider the example below.

```
public class HomeController : Controller
{
    private IEmployeeRepository _employeeRepository;

    public HomeController(IEmployeeRepository employeeRepository)
    {
        _employeeRepository = employeeRepository;
    }

    [Route("Home/Details/{id}")]
    public ViewResult Details(int id)
    {
        HomeDetailsViewModel homeDetailsViewModel = new HomeDetailsViewModel()
        {
            Employee = _employeeRepository.GetEmployee(id),
            PageTitle = "Employee Details"
        };

        return View(homeDetailsViewModel);
    }
}
```

Details() action method has id parameter. This parameter specifies the id of the employee whose details we want to view. Notice in the route template we specified id parameter. So the URL (/Home/Details/1) will execute the Details(int id) action method and maps the value "1" to the "id" parameter of the Details(int id). This is done by a process called **Model binding**. We will discuss model binding in our upcoming videos.

Attribute Route Optional Parameters

At the moment, the Details(int id) action method of the HomeController is executed, only if we have a value for the "id" route parameter in the URL(/Home/Details/1). If the id value is not in the URL, we get 404. For example, URI /Home/Details will not execute the Details(int id) action method. Instead 404 error is displayed.

To make the route parameter **"id"** optional, simply include a **"?"** at the end.

```
public class HomeController : Controller
{
    private IEmployeeRepository _employeeRepository;

    public HomeController(IEmployeeRepository employeeRepository)
    {
        _employeeRepository = employeeRepository;
```

JavaScript Tutorial
Charts Tutorial
LINQ
LINQ to SQL
LINQ to XML
Entity Framework
WCF
ASP.NET Web Services
Dot Net Basics
C#
SQL Server
ADO.NET
ASP.NET
GridView
ASP.NET MVC
Visual Studio Tips and Tricks
Dot Net Interview Questions

Slides
Entity Framework
WCF
ASP.NET Web Services
Dot Net Basics
C#
SQL Server
ADO.NET
ASP.NET
GridView
ASP.NET MVC
Visual Studio Tips and Tricks

Java Video Tutorials
Part 1 : Video | Text | Slides
Part 2 : Video | Text | Slides
Part 3 : Video | Text | Slides

```
    }

    // The ? makes id route parameter optional. To make it required remove ?
    [Route("Home/Details/{id?}")]
    // ? makes id method parameter nullable
    public ViewResult Details(int? id)
    {
        HomeDetailsViewModel homeDetailsViewModel = new HomeDetailsViewModel()
        {
            // If "id" is null use 1, else use the value passed from the route
            Employee = _employeeRepository.GetEmployee(id ?? 1),
            PageTitle = "Employee Details"
        };

        return View(homeDetailsViewModel);
    }
}
```

Controller and Action Method Names

With attribute routing the controller name and action method names play no role in which action is selected. Consider the example below.

```
public class WelcomeController : Controller
{
    [Route("")]
    [Route("Home")]
    [Route("Home/Index")]
    public ViewResult Welcome()
    {
        return View();
    }
}
```

Since we have specified the route template directly on the action method, Welcome() action in the WelcomeController is executed for all the following 3 URL paths.

/
/Home
/Home/Index

Attribute Routes are Hierarchical

The Route() attribute can be applied on the Controller class as well on the individual actions. To make attribute routing less repetitive, route attributes on the controller are combined with route attributes on the individual action methods.

Consider the example below

```
public class HomeController : Controller
{
    private IEmployeeRepository _employeeRepository;

    public HomeController(IEmployeeRepository employeeRepository)
    {
        _employeeRepository = employeeRepository;
    }

    [Route("")]
    [Route("Home")]
    [Route("Home/Index")]
    public ViewResult Index()
    {
        var model = _employeeRepository.GetAllEmployees();
        return View(model);
    }

    [Route("Home/Details/{id?}")]
    public ViewResult Details(int? id)
    {
        HomeDetailsViewModel homeDetailsViewModel = new HomeDetailsViewModel()
```

```
        {
            Employee = _employeeRepository.GetEmployee(id ?? 1),
            PageTitle = "Employee Details"
        };

        return View(homeDetailsViewModel);
    }
}
```

Index() action method of the HomeController is executed for the following 3 URL paths

/Home
/Home/Index

Details(int? id) action method of the HomeController is executed for the following 2 URL paths

/Home/Details
/Home/Details/2

As you can see there is lot of repetition. To make these routes less repetitive, apply the Route() attribute on the HomeController class as shown below.

```
[Route("Home")]
public class HomeController : Controller
{
    private IEmployeeRepository _employeeRepository;

    public HomeController(IEmployeeRepository employeeRepository)
    {
        _employeeRepository = employeeRepository;
    }

    [Route("")]
    [Route("Index")]
    public ViewResult Index()
    {
        var model = _employeeRepository.GetAllEmployees();
        return View(model);
    }

    [Route("Details/{id?}")]
    public ViewResult Details(int? id)
    {
        HomeDetailsViewModel homeDetailsViewModel = new HomeDetailsViewModel()
        {
            Employee = _employeeRepository.GetEmployee(id ?? 1),
            PageTitle = "Employee Details"
        };

        return View(homeDetailsViewModel);
    }
}
```

The Route template applied on the controller is prepended to the route template applied on the action. However, when we navigate to the root URL (http://localhost:1234), the Index() action method of the HomeController will not be executed. We instead see 404. To address this, include route template that begins with **"/"** on the Index() action method as shown below.

```
[Route("/")]
[Route("")]
[Route("Index")]
public ViewResult Index()
{
    var model = _employeeRepository.GetAllEmployees();
    return View(model);
}
```

One very important point to keep in mind is, the controller route template is not

combined with action method route template if the route template on the action method begins with **/** or **~/**

Tokens in Attribute Routing

Attribute routes support token replacement by enclosing a token in square-braces ([ ]). The tokens [controller] and [action] are replaced with the values of the controller name and action name where the route is defined.

Consider this example.

```
[Route("[controller]")]
public class DepartmentsController : Controller
{
    [Route("[action]")]
    public string List()
    {
        return "List() of DepartmentsController";
    }

    [Route("[action]")]
    public string Details()
    {
        return "Details() of DepartmentsController";
    }
}
```

With the controller and action tokens in place, the URL /Departments/List executes the List() action in the DepartmentsController. Similarly the URL /Departments/Details executes Details() action in the DepartmentsController.

This is a very powerful technique because, later if we rename the controller or action name we do not have to change our route templates. The application just works with the new controller and action names.

To make the List() action the default action for the DepartmentsController, you can still include the Route("") attribute with an empty string as shown below.

```
[Route("[controller]")]
public class DepartmentsController : Controller
{
    [Route("[action]")]
    [Route("")] // Makes List(), the default action
    public string List()
    {
        return "List() of DepartmentsController";
    }

    [Route("[action]")]
    public string Details()
    {
        return "Details() of DepartmentsController";
    }
}
```

Instead of including the [action] token on every action method in a controller, we can apply it just once on the controller as shown below.

```
[Route("[controller]/[action]")]
public class DepartmentsController : Controller
{
    public string List()
    {
        return "List() of DepartmentsController";
    }

    public string Details()
    {
        return "Details() of DepartmentsController";
    }
```

```
}
```

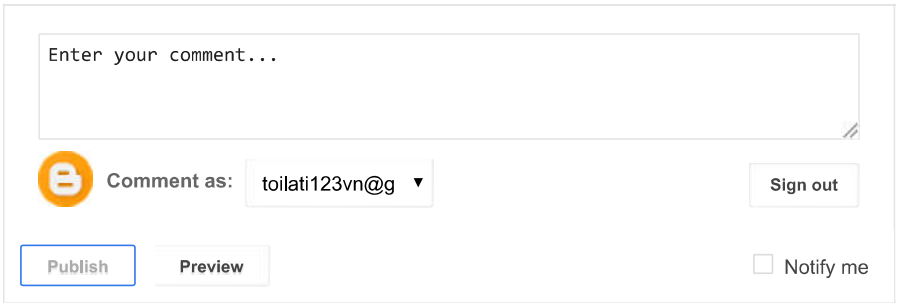
Conventional Routing vs Attribute Routing

With attribute routing, routes are placed next to the action methods that will actually use them. Attribute routes offer a bit more flexibility than conventional routes. However, in general, conventional routes are used for controllers that serve HTML pages, and attribute routes for controllers that serve REST APIs. However, there is nothing stopping us from mixing conventional routing with attribute routing in a single application to get a bit more flexibility with routes.

No comments:

Post a Comment

If you like this website, please share with your friends on facebook and Google+ and recommend us on google using the g+1 button on the top right hand corner.



Links to this post

Create a Link

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)