Routing

.NET

ASP.NET controllers use Routing middleware to match the URLs of incoming requests and map them to actions. Route templates are defined in configuration files or in attributes decorating the action methods, describe how URL paths are matched to actions, and are used to generate URLs for links. Queries can either be conventionally routed or routed based on attributes.

https://docs.microsoft.com/en-us/aspnet/core/mvc/controllers/routing?view=aspnetcore-5.0

Controllers

https://docs.microsoft.com/en-us/aspnet/core/mvc/controllers/actions?view=aspnetcore-5.0

A ***Controller*** is a class used to define and group a set of ***Action*** methods. ***Controllers*** group similar ***Actions*** together. Controllers allow routing, caching, and authorization to be applied collectively.

A ***Controller*** is responsible for the initial processing of a request and instantiation of a ***Model***.

To be classified as a ***Controller***, at least one of these conditions is true:

- The class inherits the ***Controller*** base class.
- The class inherits from a class whose name is suffixed with ***Controller***.
- The [Controller] attribute is applied to the class.

Controller classes reside in the project's root-level ***Controllers*** directory and inherit from ***Controller*** or ***ControllerBase*** classes.

```
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Logging;
using MvcProjectStarter.Models;

namespace MvcProjectStarter.Controllers
{
    public class HomeController : Controller
    {
        private readonly ILogger<HomeController>

        public HomeController(ILogger<HomeContro
        {
```

Action Methods

https://docs.microsoft.com/en-us/aspnet/core/mvc/controllers/actions?view=aspnetcore-5.0#defining-actions

- An ***Action*** method is a method in a ***Controller*** that handles requests.
- All public methods in a ***Controller*** (except those with the [NonAction] attribute) are ***Actions***.
- Parameters on ***Actions*** are bound to request data and are validated using ***ModelBinding***.
- ***Model validation*** occurs for everything that's ***Model-Bound***.
- The ModelState.IsValid method indicates whether ***ModelBinding*** and ***validation*** succeeded.
- ***Action*** methods should contain logic for mapping a request to a business concern.
- Business concerns should typically be represented as services that the ***Controller*** accesses through ***Dependency Injection***.
- ***Actions*** can return anything, but usually return an IActionResult or Task<IActionResult> (for async methods).

```
namespace MvcProjectStarter.Controllers
{
    public class SongsController : Controller
    {
        private readonly MvcSongContext _context;

        public SongsController(MvcSongContext context)
        {
            _context = context;
        }

        // GET: Songs
        public async Task<IActionResult> Index()
        {
            return View(await _context.Song.ToListAsync());
        }

        // GET: Songs/Details/5
        public async Task<IActionResult> Details(int? id)
        {
            if (id == null)
            {
                return NotFound();
            }

            var song = await _context.Song
                .FirstOrDefaultAsync(m => m.id == id);
            if (song == null)
            {
                return NotFound();
            }

            return View(song);
        }
```

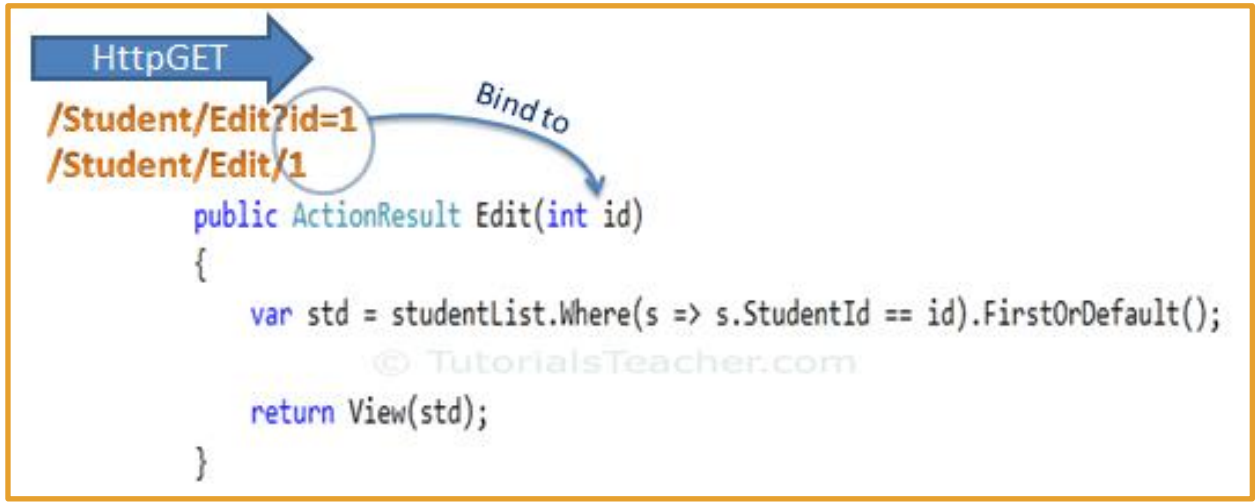
Model Binding

https://docs.microsoft.com/en-us/aspnet/core/mvc/models/model-binding?view=aspnetcore-5.0

Controllers and ***Action*** methods work with data that comes from HTTP requests. (Ex. ***POST***ed form fields provide values for the properties of the ***model***.)

Writing code to retrieve each of these values and convert them from strings to .NET ***types*** would be tedious and error-prone. ***ModelBinding*** automates this process.

The ***ModelBinding*** system:
- Retrieves data from various sources such as route data, form fields, and query strings.
- Provides the data to ***Controllers*** in ***Action*** method parameters and public ***Properties***.
- Converts ***string*** data to .NET types.
- Updates Properties of complex types.



Model Binding

https://docs.microsoft.com/en-us/aspnet/core/mvc/models/model-binding?view=aspnetcore-5.0

In this example, ***ModelBinding*** goes through the following steps for the request at the bottom.

1. The routing system selects the correct ***action*** method.
2. It needs the first parameter of GetByID (id) and looks through the HTTP request.
3. It finds id = "2" in the route data.
4. The system converts string "2" into integer 2.
5. It finds the next parameter of GetByID(dogsOnly).
6. The system finds "DogsOnly=true" in the query string. Name matching is not case-sensitive.
7. The system converts the string "true" to a boolean true.

Suppose you have the following action method:

C#

```
[HttpGet("{id}")]
public ActionResult<Pet> GetById(int id, bool dogsOnly)
```

And the app receives a request with this URL:

```
http://contoso.com/api/pets/2?DogsOnly=true
```

Alternate Controller Helper Methods

https://docs.microsoft.com/en-us/aspnet/core/mvc/controllers/actions?view=aspnetcore-5.0#controller-helper-methods

---

The ***Controller*** provides access to three categories of helper methods. These helper methods help return the appropriate status code or route to predefined pages.

an empty response body	a non-empty response body with a predefined content type	a non-empty response body formatted in a content type negotiated with the client
HTTP Status Code (ex. BadRequest(), NotFound(), and Ok();)	View() which uses a ***Model*** to render HTML. (ex. Return View(Customer);)	This category is known as **Content Negotiation**. ***Content negotiation*** applies whenever an ***action*** returns an ObjectResult type or something other than an IActionResult. (Ex. BadRequest(), CreatedAtRoute();, and Ok();)
Redirect - returns a redirect to an action or destination (Redirect(), LocalRedirect(), RedirectToAction(), or RedirectToRoute();).	**Formatted Response** - JSON or a similar data exchange format to represent an object, (ex. Json(customer);)	

Conventional Routing

https://docs.microsoft.com/en-us/aspnet/core/mvc/controllers/routing?view=aspnetcore-5.0#cr

---

Program.cs typically has code similar to the following when using conventional routing.

Inside the call to UseEndpoints(), ***.MapControllerRoute()*** is used to create a route. This single route is named "***default***". /Home/Index/<args> being the default route used when a request arrives to the base URL.

```
app.UseEndpoints(endpoints =>
{
    endpoints.MapControllerRoute(
        name: "default",
        pattern: "{controller=Home}/{action=Index}/{id?}");
});
```

Conventional Routing

https://docs.microsoft.com/en-us/aspnet/core/mvc/controllers/routing?view=aspnetcore-5.0#set-up-conventional-route
https://docs.microsoft.com/en-us/aspnet/core/mvc/controllers/routing?view=aspnetcore-5.0#multiple-conventional-routes

---

The route template (in program.cs) "{controller=Home}/{action=Index}/{id?}" matches a URL path like /Products/Details/5.

```
app.UseEndpoints(endpoints =>
{
    endpoints.MapControllerRoute(
        name: "default",
        pattern: "{controller=Home}/{action=Index}/{id?}");
});
```

The route template ***tokenizes***(extracts) the route values:

- Controller = Products,
- Action = Details,
- id = 5

This results in a match if the app has a ***Controller*** named ProductsController and an ***Action*** called Details. The ***id*** value is optional due to the ?.

```
app.UseEndpoints(endpoints =>
{
    endpoints.MapControllerRoute(name: "blog",
                pattern: "blog/{*article}",
                defaults: new { controller = "Blog", action = "Article" });
    endpoints.MapControllerRoute(name: "default",
                pattern: "{controller=Home}/{action=Index}/{id?}");
});
```

Attribute Routing – REST API's

https://docs.microsoft.com/en-us/aspnet/core/mvc/controllers/routing?view=aspnetcore-5.0#attribute-routing-for-rest-apis

---

RESTful APIs should use ***Attribute Routing*** to model the app's functionality as a set of resources where operations are represented by ***HTTP verbs***.

Attribute Routing uses sets of ***Attributes*** on each ***Controller Action*** to map ***Actions*** directly to route templates. The following program.cs code is typical for a ***RESTful API***.

In ASP.NET, .MapControllers() is called inside UseEndpoints() to map attribute routed controllers.

```
app.UseEndpoints(endpoints =>
{
    endpoints.MapControllers();
});
```

Attribute Routing – REST API's

https://docs.microsoft.com/en-us/aspnet/core/mvc/controllers/routing?view=aspnetcore-5.0#attribute-routing-for-rest-apis

---

HomeController matches a set of URLs similar to how the default ***conventional*** routing matches a ***Controller Action*** to {controller=Home}/{action=Index}/{id?}. ***Conventional Routing*** handles routes more succinctly, but ***Attribute Routing*** allows (and requires) precise control over which route templates apply to each ***Action***.

With ***Attribute Routing***, the ***Controller*** name and ***Action*** names no longer play a role in which ***Action*** is matched.

```
public class MyDemoController : Controller
{
    [Route("")]
    [Route("Home")]
    [Route("Home/Index")]
    [Route("Home/Index/{id?}")]
    public IActionResult MyIndex(int? id)
    {
        return ControllerContext.MyDisplayRouteInfo(id);
    }

    [Route("Home/About")]
    [Route("Home/About/{id?}")]
    public IActionResult MyAbout(int? id)
    {
        return ControllerContext.MyDisplayRouteInfo(id);
    }
}
```

Attribute Routing - HTTP Verb Templates

https://docs.microsoft.com/en-us/aspnet/core/mvc/controllers/routing?view=aspnetcore-5.0#http-verb-templates

ASP.NET Core provides these ***HTTP verb*** attributes: [HttpGet], [HttpPost], [HttpPut], [HttpDelete], [HttpHead], [HttpPatch].

The GetProduct() ***Action*** method includes the "{id}" template, therefore 'id' is appended to the "api/[controller]" attribute template above the ***Controller***, so GetProduct() template is "api/test2/{id}".

Therefore, GetProduct(string id) can match ***GET*** requests of the form:

- /api/test2/{any string}.

```
[Route("api/[controller]")]
[ApiController]
public class Test2Controller : ControllerBase
{
    [HttpGet]   // GET /api/test2
    public IActionResult ListProducts()
    {
        return ControllerContext.MyDisplayRouteInfo();
    }

    [HttpGet("{id}")]   // GET /api/test2/xyz
    public IActionResult GetProduct(string id)
    {
       return ControllerContext.MyDisplayRouteInfo(id);
    }

    [HttpGet("int/{id:int}")] // GET /api/test2/int/3
    public IActionResult GetIntProduct(int id)
    {
        return ControllerContext.MyDisplayRouteInfo(id);
    }

    [HttpGet("int2/{id}")]  // GET /api/test2/int2/3
    public IActionResult GetInt2Product(int id)
    {
        return ControllerContext.MyDisplayRouteInfo(id);
    }
}
```