.NET Core: Developing Cross-Platform Web Apps with ASP.NET Core – Workshop*PLUS*

Wael Kdouh - @waelkdouh

Senior Consultant

v3.0

Conditions and Terms of Use

Copyright and Trademarks

How to View This Presentation

- To switch to **Notes Page** view:
  - On the ribbon, click the **View** tab, and then click **Notes Page**
- To navigate through notes, use the Page Up and Page Down keys
  - Zoom in or zoom out, if required
- In the **Notes Page** view, you can:
  - Read any supporting text
    - Terminology List—a list of terms used in this course is provided in the Notes section.
  - Add notes to your copy of the presentation, if required
- Take the presentation files home with you

Module 8: Routing

Module Overview

Module 8: Routing

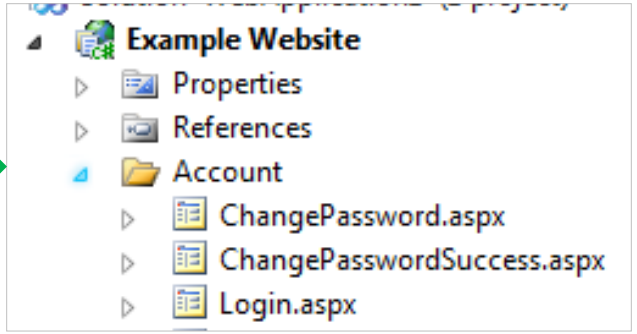
Section 1: Routing and URL Overview

Lesson: Routing and URL Overview

URL

- URL can represent physical files on disk in ASP, JSP, PHP, ASP.NET (without routing), etc.

http://example.com/Account/Login.aspx



- ASP.NET Model-View-Controller (MVC) maps URL to action methods of Controller classes

http://example.com/Account/Login

```
[Authorize]
public class AccountController : Controller
{
    [AllowAnonymous]
    public ActionResult Login(string returnUrl)
    {
        ViewBag.ReturnUrl = returnUrl;
        return View();
    }
```

URL Guidelines

- A domain name easy to remember and easy to spell
- Short URLs
- Easy-to-type URLs
- URLs that reflect the site structure
- Hackable URLs

http://blog.com/2009/4/6	Blog posts published on 4/6/2009
http://blog.com/2009/4	Blog posts published in April 2009
http://blog.com/2009	Blog posts published in 2009

- Persistent URLs:
  - URLs that do not change over time
  - Avoid URL breakage from caller sites

ASP.NET MVC Routing

- A route is a URL pattern mapped to a handler
- Handler can be a physical file or action method in a controller
- Route instance specifies:
  - URL pattern
  - Route handler
  - Route name (optional)

```
// Add MVC to the request pipeline.
app.UseMvc(routes =>
{
    routes.MapRoute(
        name: "default",
        template: "{controller=Home}/{action=Index}/{id?}");
});
```

- ASP.NET MVC Routing also constructs outgoing URLs corresponding to controller actions

Routing vs. URL Rewriting

Routing	URL Rewriting
Used for mapping a URL to a resource	Often used to map old URLs to a new set of URLs
Routing embodies resource-centric view; never rewrites URL	Rewrites URLs to correctly map to the resource
Routing helps generate URLs using the same routing rules	URL rewriting only applies to incoming requests
Performed at ASP.NET level	Besides ASP.NET, it can be implemented with Internet Server API (ISAPI) filters at Internet Information Services (IIS) level

`// Add MVC to the request pipeline.`
`app.UseMvc(routes =>`
`{`
`  routes.MapRoute(`
`  name: "default",`
`  template: "{controller=Home}/{action=Index}/{id?}");`
`});`
	`<RewriterConfig>`
	`  <Rules>`
	`  <!-- Rules for Product Lister -->`
	`  <RewriterRule>`
	`  <LookFor>~/Products/Beverages\.aspx</LookFor>`
	`  <SendTo>~/ListProductsByCategory.aspx?CategoryID=1</SendTo>`
	`  </RewriterRule>`
	`  <RewriterRule>`
	`  </Rules>`
	`</RewriterConfig>`

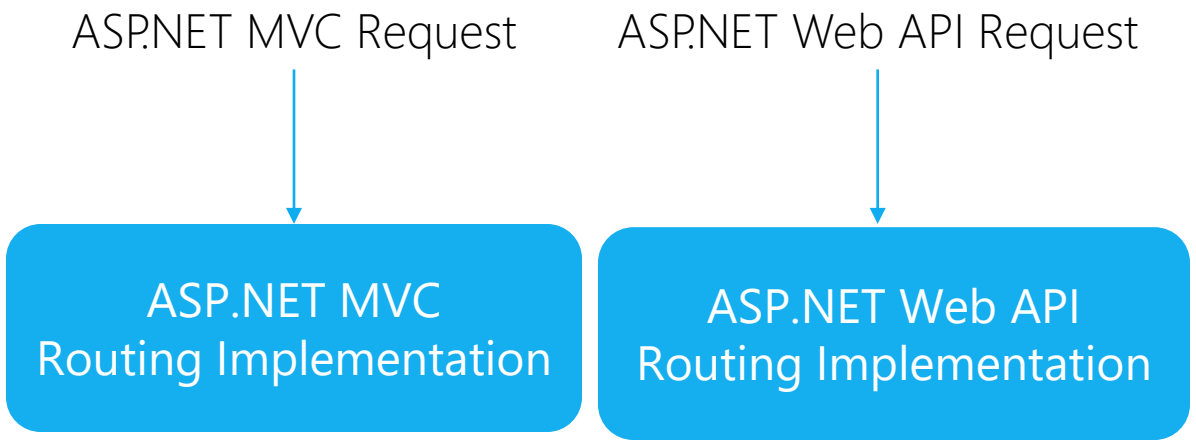
Module 8: Routing

Section 2: Routing Fundamentals

Lesson: Routing Fundamentals

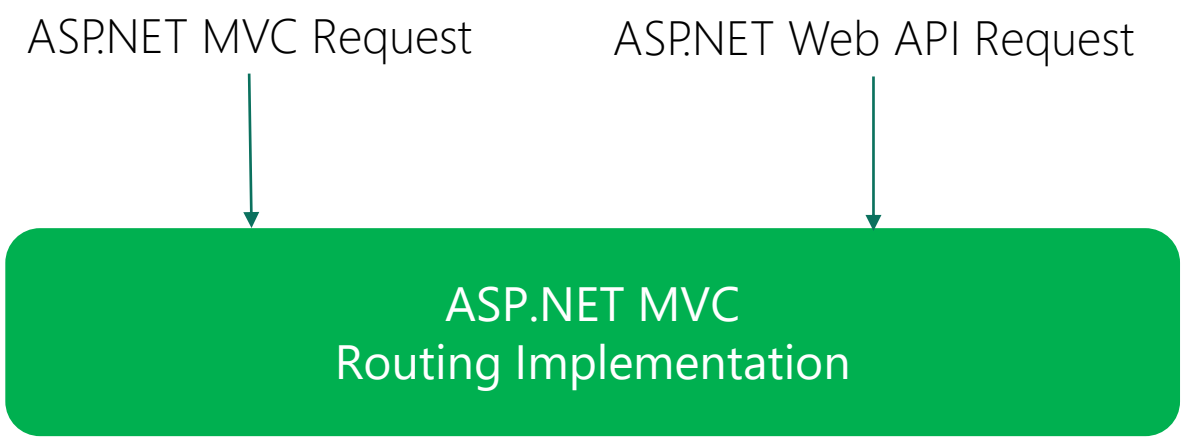
Request Routing

ASP.NET 4



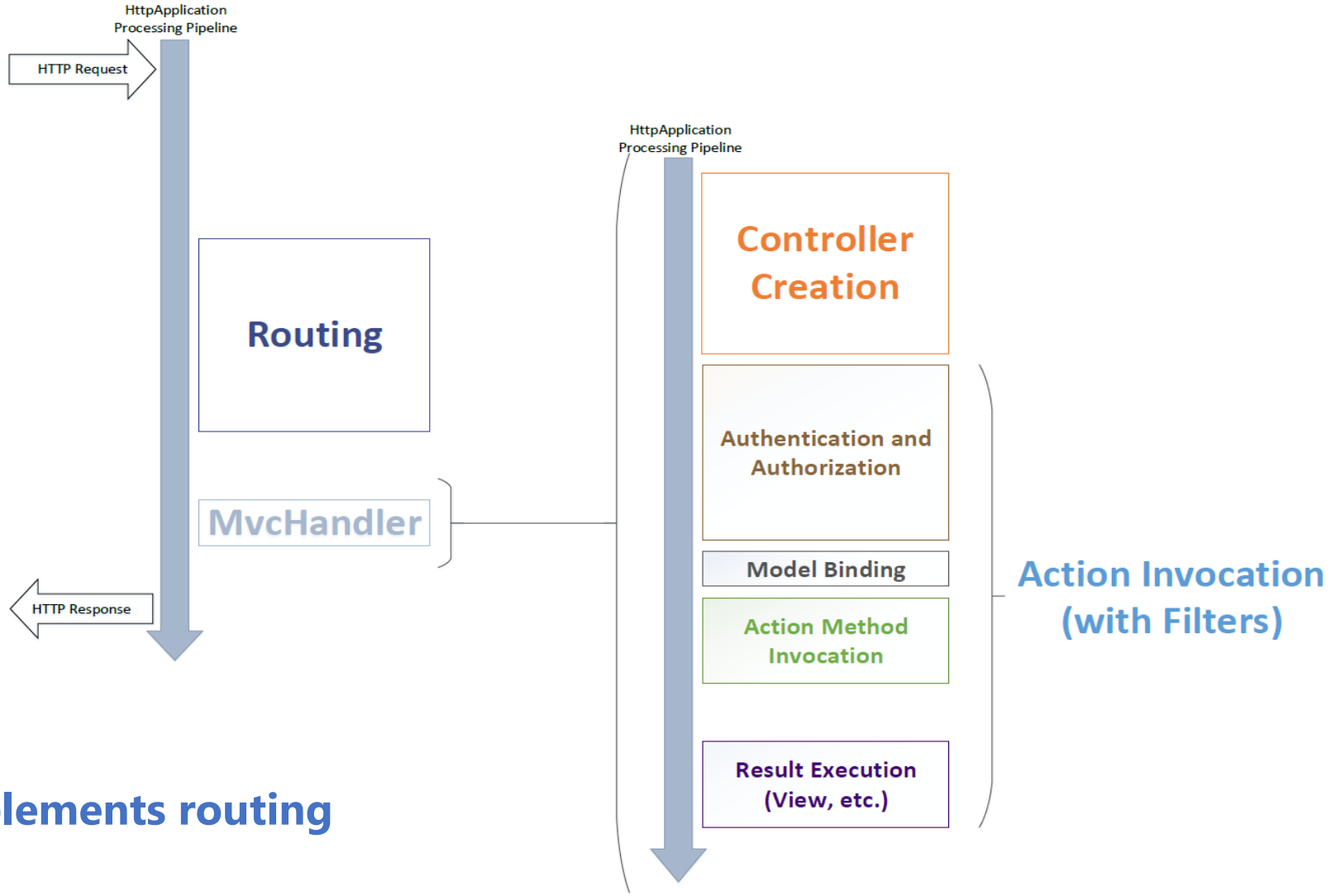
- Similar behavior
- Separate implementations
- Developed by two different teams in Microsoft

ASP.NET Core



- Same implementation and behavior
- Share the same framework
- Rewritten from ground-up

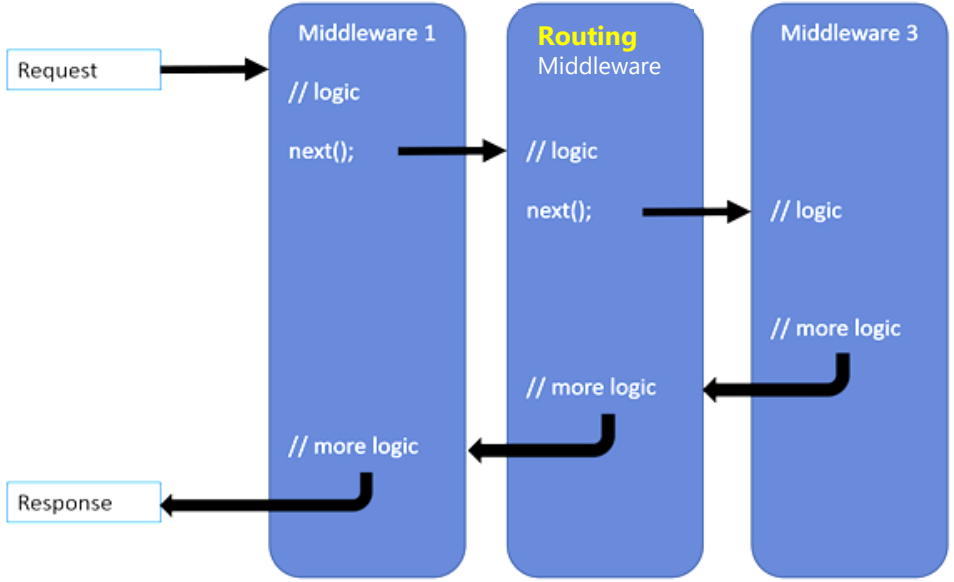
ASP.NET Core MVC: Routing in HTTP Application Processing Pipeline



ASP.NET Core MVC implements routing through middleware.

ASP.NET Core MVC Routing Pipeline [Middleware]

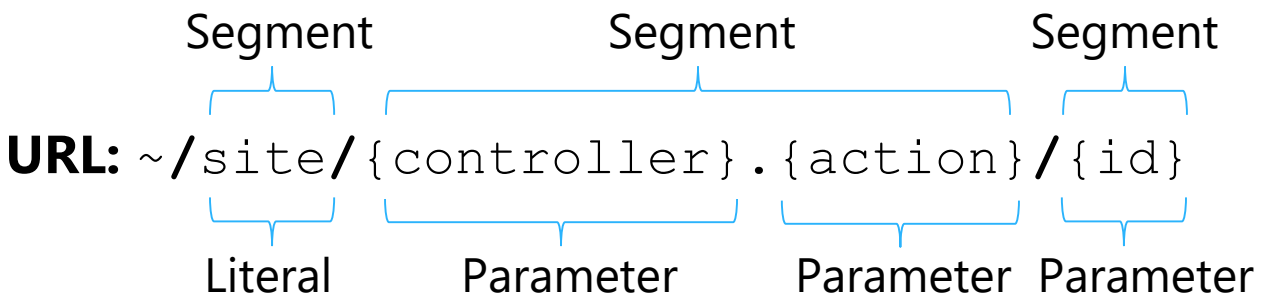
- Routing middleware tries to match the request with routes in route collection.
- If one of the routes matches the request, it looks for the handler for the route.
- The RouteAsync method of the handler is called. A flag called **IsHandled** is set to **true** to mark successful handling of request.
- If IsHandled is set to **false**, it means the route was not able to handle the request, and that the next route should be tried.



URL Parameter Value Mapping

URL Pattern: `{first}/{second}/{third}`

URL	URL Parameter Values
~/Products/Show/123	first = “Products”; second = “Show” third = “123”
~/electronics/pcs/baz	first = “electronics”; second = “pcs” third = “baz”
~/a.b/b-c	first = “a.b”; second = “b-c” third = “”



URL Patterns

Route Definition	Example of Matching URL
{controller}/{action}/{id}	~/Products/show/beverages
{table}/Details.aspx	~/Products/Details.aspx
blog/{action}/{entry}	~/blog/show/123
{reporttype}/{year}/{month}/{day}	~/sales/2008/1/5
{locale}/{action}	~/US/show
{language}-{country}/{action}	~/en-US/show
{controller}.{action}.{id}	~/Products.Show.123

Route Constraints

- Constraints allow you to apply a regular expression to URL segments to restrict request matching

```
routes.MapRoute("blog", "{locale}/{year}/{month}/{day}",
    new { controller = "Blog", action = "Index" },
    new
    {
        locale = "[a-z]{2}-[A-Z]{2}",
        year = @"\d{4}",
        month = @"\d{2}",
        day = @"\d{2}"
    });
```

Example URL	Match/No-Match?
~/en-US/08	No match
~/en-US/08/05/25	No match
~/en-GB/2008/05/25	Match
~/fr-FR/2012/04/2	No match
~/fr-FR/2012/04/02	Match

Multiple URL Parameters in a Segment

- Route URL may have multiple parameters in a segment
- Parameters cannot be adjacent to avoid ambiguity

Route URL	Request URL	Route Data Result
{filename}.{ext}	~/Foo.xml.aspx	filename="Foo.xml" ext="aspx"
My{title}-{cat}	~/MyHouse-dwelling	title="House" cat="dwelling"
{foo}xyz{bar}	~/xyzxyzxyzblah	foo="xyzxyz" bar="blah"
{title}{artist}	-	-
{Filename}{ext}	-	-

Demo: URL Patterns

Module 8: Routing

Section 3: ASP.NET MVC Routing Techniques

Lesson: Routing and MVC

Route Configuration

Startup.cs:

```
// Add MVC to the request pipeline.
app.UseMvc(routes =>
{
    routes.MapRoute(
        name: "default",
        template: "{controller=Home}/{action=Index}/{id?}");
});
```

```
// Add MVC with default route to the request pipeline.
app.UseMvcWithDefaultRoute();
```

Route Mapping to Controller Actions

URL Pattern: `{controller}/{action}/{id}`

URL: `~/albums/display/123`

```
public class AlbumsController : Controller
    {
        public ActionResult Display(int id)
        {
            // Do something
            return View();
        }
    }
```

Optional and Default Parameters

```
// Add MVC to the request pipeline.
app.UseMvc(routes =>
{
    routes.MapRoute(
        name: "default",
        template: "{controller=Home}/{action=Index}/{id?}");
});
```

Route URL Pattern	Defaults	Examples of Matching URLs
{controller}/{action}/{id}	new {id = UrlParameter.Optional}	/albums/display/123 /albums/display
{controller}/{action}/{id}	new { controller="home", action="index", id = UrlParameter.Optional }	/albums/display/123 /albums/display /albums /

Named Routes

- Always use route names to avoid ambiguities during route generation

```
@Html.RouteLink(linkText: "Test Route", routeName: "Test",
routeValues: new { controller = "Test", action = "Index", id = "123" })

@Html.RouteLink(linkText: "Default Route", routeName: "Default",
routeValues: new { controller = "Home", action = "Index", id = "123" })
```

- Performance improvement for routing engine

Demo: Constraint in MVC

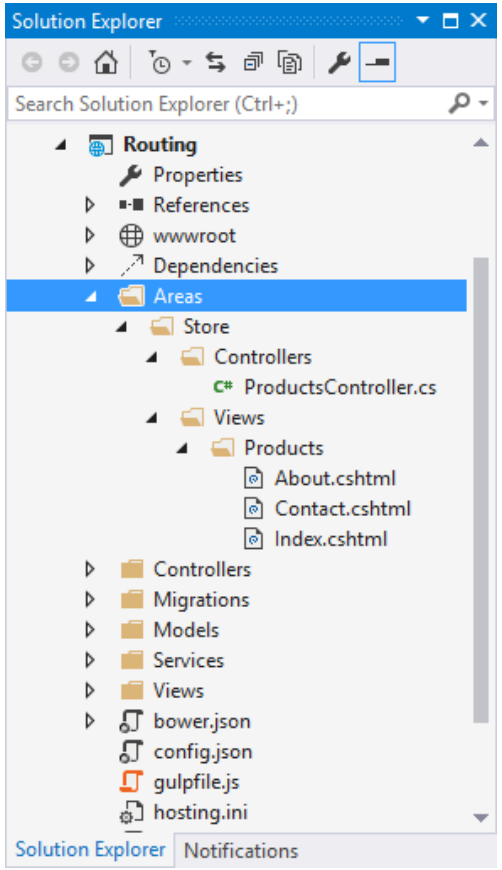
Module 8: Routing

Section 3: ASP.NET MVC Routing Techniques

Lesson: Areas

Areas

- MVC areas separate a large MVC application into smaller functional groups
  - For example, a large e-commerce site is divided into areas for storefront, product reviews, user accounts, etc.
- Area guidelines:
  - *Areas* directory must exist as project child directory
  - *Areas* contains subdirectory for each area
  - Controllers should be located at: */Areas/[area]/Controllers/[controller].cs*
  - Views should be located at: */Areas/[area]/Views/[controller]/[action].cshtml*



Area Registration and Linking

- Area Registration

```
app.UseMvc(routes =>
{
    routes.MapRoute(
        name: "AreaRoute",
        template: "{area=Store}/{controller=Products}/{action=Index}/{id?}"
    );

    routes.MapRoute(
        name: "default",
        template: "{controller=Home}/{action=Index}/{id?}");
});
```

```
namespace DemoApp.Areas.Store.Controllers
{
    [Area ("Store")]
    0 references
    public class ProductsController : Controller
    {
        0 references
        public IActionResult Index()
        {
            return Content("it works!");
        }
```

- Area Linking in Views

```
@Html.ActionLink("See Products Home Page", "Index", "Home", new { area = "Products" }, null)
```

```
@Html.ActionLink("Go to Home Page", "Index", "Home", new { area = "" }, null)
```

Areas

```
[Area("Admin")]
public class MenuController : Controller
{
    // eg: /admin/menu/login
    public ActionResult Login() { ... }
    // eg: /admin/menu/show-options
    [Route("show-options")]
    public ActionResult Options() { ... }
    // eg: /stats
    [Route("~/stats")]
    public ActionResult Stats() { ... }
}
```

Module 8: Routing

Section 3: ASP.NET MVC Routing Techniques

Lesson: Attribute Routing

Attribute Routing is the recommended approach in ASP.NET Core MVC

Combination of conventional and attribute routing is allowed

Convention-Based Routing vs. Attribute Routing

Convention-based Routing

```
routes.MapRoute(
    name: "ProductPage",
    url: "{productId}/{productTitle}",
    defaults: new { controller = "Products", action = "Show" },
    constraints: new { productId = "\\d+" }
);
```

Attribute Routing

```
[Route("{productId:int}/{productTitle}")]
public IActionResult Show(int productId) { ... }
```

Routing co-defined with implementation.

Optional and Default Parameters

```
public class BooksController : Controller
{
    // eg: /books, /books/1430210079
    [Route("books/{isbn?}")]
    public IActionResult View(string isbn)
    {
        if (!String.IsNullOrEmpty(isbn))
        {
            return View("OneBook", GetBook(isbn));
        }
        return View("AllBooks", GetBooks());
    }

    // eg: /books/lang, /books/lang/en, /books/lang/he
    [Route("books/lang/{lang=en}")]
    public IActionResult ViewByLanguage(string lang)
    {
        return View("OneBook", GetBooksByLanguage(lang));
    }
}
```

Common Route Prefix

```
[Route ("reviews")]
public class ReviewsController : Controller
{
    // eg.: /reviews
    public IActionResult Index() { ... }
    // eg.: /reviews/5
    [Route("{reviewId}")]
    public IActionResult Show(int reviewId) { ... }
    // eg.: /reviews/5/edit
    [Route("{reviewId}/edit")]
    public IActionResult Edit(int reviewId) { ... }
    // eg.: /spotlight-review
    [Route("~/spotlight-review")]
    public IActionResult ShowSpotlight() { ... }
}
```

Inline Constraints

Constraint	Description	Example Template
alpha	Matches uppercase or lowercase Latin alphabet characters (a-z, A-Z)	"Product/{ProductName:alpha}"
int	Matches a Signed 32-bit integer value	"Product/{ProductId:int}"
long	Matches a Signed 64-bit integer value	"Product/{ProductId:long}"
minlength	Matches a string with a minimum length	"Product/{ProductName:minlength(10)}"
regex	Matches a regular expression	"Product/{productId:regex(^\\d{4}$)}"

Route Constraints

```
// eg: /users/5
[Route("users/{id:int}"]
public ActionResult GetUserById(int id) { ... }
// eg: users/ken
[Route("users/{name}"]
public ActionResult GetUserByName(string name) { ... }

// eg: /users/5 but not /users/10000000000 because it is larger than int.MaxValue, and not /users/0 because
of the min(1) constraint.
[Route("users/{id:int:min(1)}")]
public ActionResult GetUserById(int id) { ... }
// eg: /greetings/bye and /greetings because of the Optional modifier,
// but not /greetings/see-you-tomorrow because of the maxlength(3) constraint.
[Route("greetings/{message:maxlength(3)?}")]
public ActionResult Greet(string message) { ... }
```

Demo: Routing

Module 8: Routing

Section 3: ASP.NET MVC Routing Techniques

Lesson: Routing Techniques

Catch-All Parameter

- Catch-All parameter allows for route to match arbitrary number of segments
- Catch-All parameter in URL Pattern:

```
"{controller}/{action}/{id}/{*ExtraParam}"
```

Example

- URL: `~/Home/Index/1234/523/89`
- RouteDebugger Output:

Matched Route: {controller}/{action}/{id}/{*ExtraParam}

Route Data		Data Tokens	
Key	**Value**	**Key**	**Value**
controller	Home		
action	Index		
id	1234		
ExtraParam	523/89		

Ambient Route Values

- Ambient values are the route values of previous request, which are re-used in the context of the current request
  - For example, Controller and action values for the second action link is shown as follows:

```
@Html.ActionLink("Page 2", "List",
new { controller = "Tasks",
action = "List", page = 2 })
```

```
tasks/list/2
```

```
@Html.ActionLink("Page 3", "List",
new { page = 3 })
```

```
tasks/list/3
```

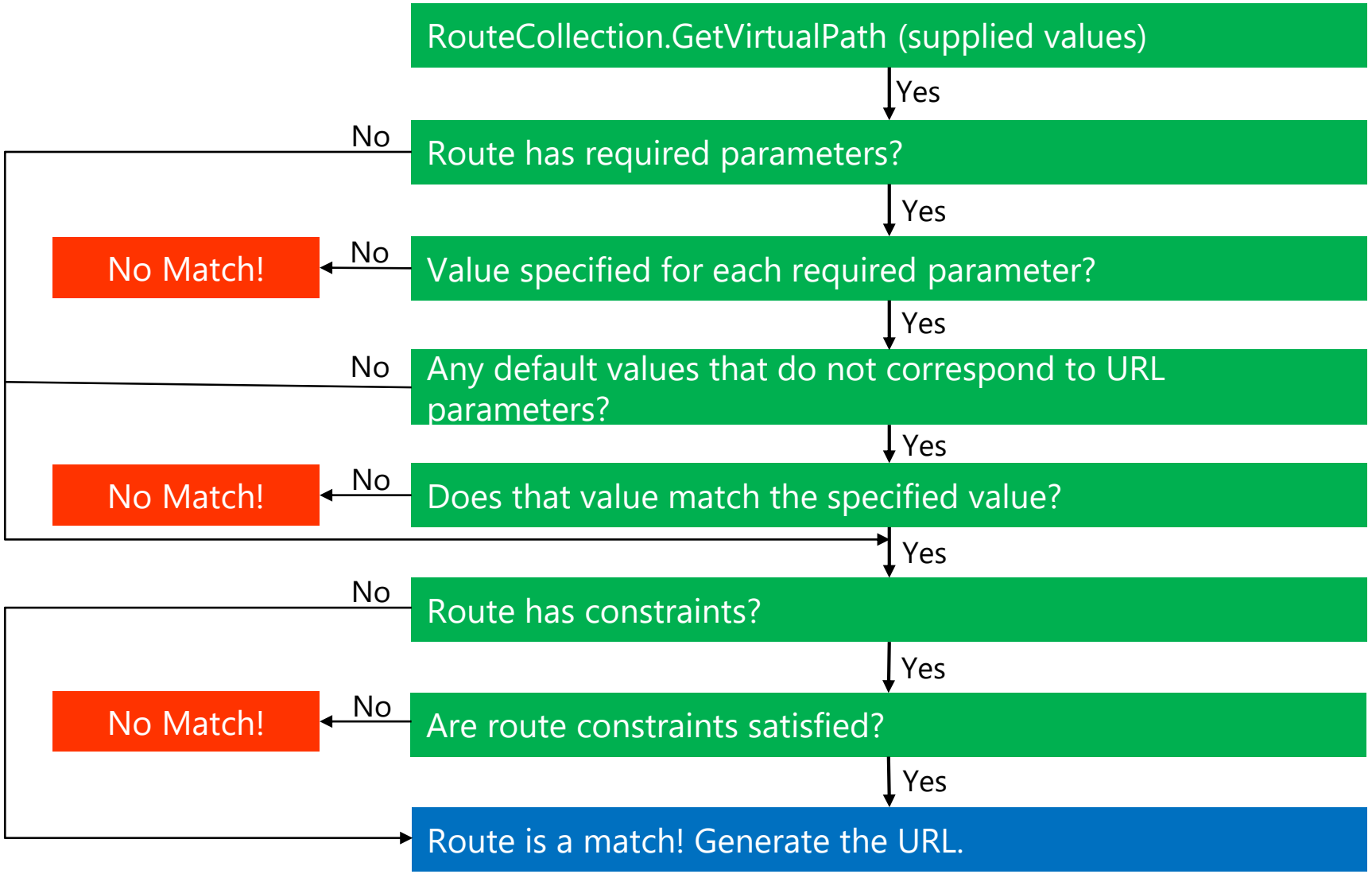
Overflow Parameters

- Overflow parameters are route values that are not specified in the route's definition
  - Appended to generated URL as query string parameters

```
routes.MapRoute("Reports", "reports/{year}/{month}/{day}", new { day = 1 });
```

Parameters	Resulting URL	Reason
year=2010, month=1, day=12	/reports/2010/1/12	Straightforward match
year=2010, month=1	/reports/2010/1/1	Default for day = 1
year=2010, month=1, day=12, category=64	/reports/2010/1/12?category=64	Overflow parameters go into query string
Year=2007	null	Required parameters not provided

URL Generation



IApplicationBuilder.UseStaticFiles( )

- Enables static file serving for the current request path from the current directory
- Added to the application pipeline **before** MVC

```
// Configure is called after ConfigureServices is called.
0 references
public void Configure(IApplicationBuilder app, IHostingEnvironment env, ILoggerFactory loggerFactory)
{
    // Add static files to the request pipeline.
    app.UseStaticFiles();

    // Add cookie-based authentication to the request pipeline.
    app.UseIdentity();

    // Add MVC with default route to the request pipeline.
    app.UseMvcWithDefaultRoute();
}
```

Startup.cs

Module Summary

- In this module, you learned about:
  - Usability guidelines for URLs
  - ASP.NET MVC Routing
  - Conventional Routing
  - Attribute Routing
  - MVC areas and their route registration
  - URL generation through routing rules
  - Request routing pipeline
  - Route debugging

Lab: Routing

Microsoft