Optimizing the Discoverability of Pages Using the Routing Engine

Gill Cleeren
ARCHITECT

@gillcleeren www.snowball.be

Overview

An overview of routing

Attribute-based routing

Areas

Outgoing links

An Overview of Routing

Routing in ASP.NET Core MVC

http://www.bethanyspieshop.com/Home/Index

Controller

Action

{Controller}/{Action}

```
http://www.bethanyspieshop.com/Home : no match

http://www.bethanyspieshop.com/Home/index/pies : no match
```

Routing

Ls

Outgoing URLs

```
app.UseMvc(routes =>
{
    routes.MapRoute(
      name: "default",
      template:
"{controller}/{action}");
});
```

◀ Routing configuration

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

◀ Passing defaults

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

◀ Defaults inline

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

◀ Using constraints and optional

```
routes.MapRoute(
  name: "categoryfilter",
  template:
"Shop/{action}/{category?}",
  defaults: new { Controller
= "Pie", action = "List" });
```

◀ Static segments

Attribute-based Routing

```
public class HomeController : Controller
{
    [Route("Home")]
    public IActionResult Index()
    {
        return View();
    }

}
```

A Simple Example

Match for /Home

```
public class HomeController : Controller
{
    [Route("")]
    [Route("Home")]
    [Route("Home/Index")]
    public IActionResult Index()
    {
        return View();
    }

}
```

Defining Multiple Attribute-based routes

Match for

- /
- /Home
- /Home/Index

```
public class HomeController : Controller
{
    [Route("[controller]/Details/{id}")]
    public IActionResult Details(int id)
    {
        var pie = _pieRepository.GetPieById(id);
        if (pie == null)
            return NotFound();
        return View(pie);
    }
}
```

Using a Token in the Route

Will match

- /Pie/Details/3

```
[Route("[controller]/Details/{id:int}")]
public IActionResult Details(int id)
{
    var pie = _pieRepository.GetPieById(id);
    if (pie == null)
        return NotFound();
    return View(pie);
}
```

Using Constraints

```
[HttpGet("/pies")]
public IActionResult ListPies()

{ ... }
```

Using Http[verb] attributes

```
[HttpPost("/pies")]
public IActionResult EditPies(...)

{ ... }
```

Using Http[verb] attributes

Defining the Route on the Controller

```
[Route("pies")]
public class PiesController : Controller
{
    [HttpGet]
    public IActionResult ListPies() { ... }


    [HttpGet("{id}")]
    public IActionResult GetPie(int id) { ... }

}
```

More Than One Route Attribute

```
[Route("pies")]
[Route("[controller]")]
public class PiesController : Controller
{

    [HttpGet("List")]
    [HttpGet("Overview")]
    public IActionResult ListPies() { ... }

}
```

Combining attribute and convention-based routing

- Works 100%
- Attribute-based will overrule

Demo

Optimizing the routes in the application

Areas

Areas in ASP.NET Core MVC

m

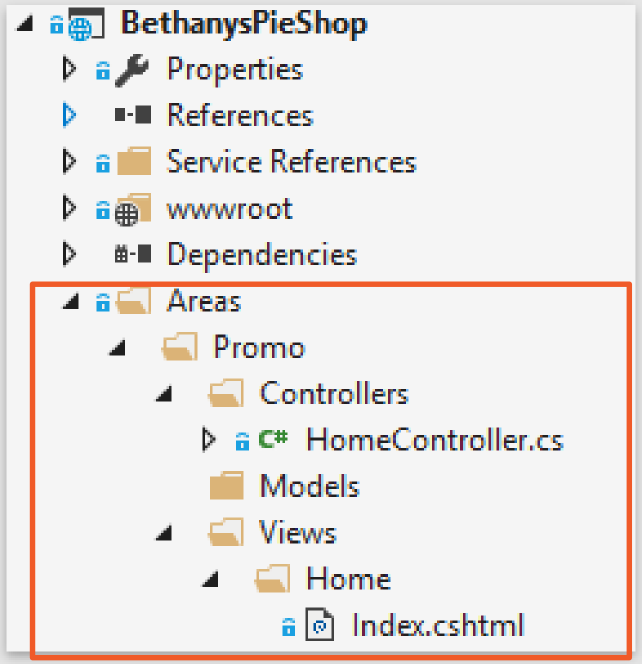
uc

c

meter

“Areas” folder

Areas in ASP.NET Core MVC



```
BethanysPieShop
  Properties
  References
  Service References
  wwwroot
  Dependencies
  Areas
    Promo
      Controllers
        HomeController.cs
      Models
      Views
        Home
          Index.cshtml
```

```
app.UseMvc(routes =>
{
    routes.MapRoute(
        name: "areas",
        template:
        "{area:exists}/{controller=Home}/{action=Index}");

}
```

Defining the Route

```
[Area("Promo")]
public class HomeController : Controller
{ ... }
```

Attributing the Controller

Demo

Adding the Promo area to the site

Managing Outgoing Links

Routing

Ls

Outgoing URLs

Generating a URL to Another Action

```
public class PieController : Controller
{

    public IActionResult Index()
    {
        // Generates /Pie/SomeAction
        var url = Url.Action("SomeAction");
        return Content($“Please go to {url}.");
    }

    public IActionResult SomeAction()
    {
        return View();
    }
  }
```

```
Controller="Controller"
Action="Index"
```

◀ Current request

```
Controller="PieController"
Action="SomeAction"
```

◀ New values

```
{controller}/{action}/{id}
```

◀ Template

```
/Pie/SomeAction
```

◀ Result

```
<h3>
   <a asp-controller="Pie" asp-action="List">
      @pie.Name</a>
</h3>
```

Urls in Views

Attribute-based Generation

```
public class PieController : Controller
{
    [HttpGet("")]
    public IActionResult Index()
    {

        // Generates /foo/bar/SomeAction
        var url = Url.Action(“SomeAction");
        return Content($“Please go to {url}.");
    }

    [HttpGet(“foo/bar/SomeAction")]
    public IActionResult Destination()
    {
        return View();
    }
}
```

Action-name

```
public class PieController : Controller
{
    public IActionResult Index()
    {

        // Generates /Pie/Details/1
        var url =
            Url.Action("Details", “Pie", new { id = 1 });
        return Content(url);
    }

}
```

URLs in Action Results

```
[HttpPost]
public async Task<IActionResult> Logout()
{
    await _signInManager.SignOutAsync();
    return RedirectToAction("Index", "Home");

}
```

Demo

Working with outgoing URLs

Summary

Routing engine is powerful concept

Attribute-based routing gives fine-grained control over routes

Areas make large sites easier to manage and navigate

Routing also covers outgoing links

Up next:
Testing our code with unit tests