Creating Clean and Maintainable View Code

Gill Cleeren
ARCHITECT

@gillcleeren www.snowball.be

Overview

- Advanced built-in tag helpers
- Creating custom tag helpers
- Async view components
- Localizing the application

Advanced Built-in Tag Helpers

Tag Helpers enable server-side C# code to participate in creating and rendering HTML elements in Razor files

Tag Helpers

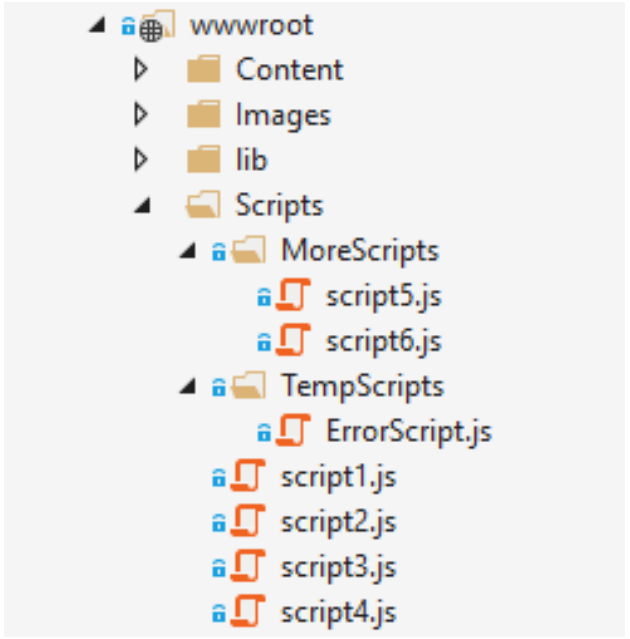
Built-in Tag Helpers

- Form tag helper
- Input tag helper
- Label tag helper
- asp-controller
- asp-action
- asp-route
- asp-antiforgery
- ...

```
<form asp-action="Checkout" method="post"
  asp-antiforgery="true" role="form">
  ...
</form>
```

Using Tag Helpers

JavaScript Tag Helper



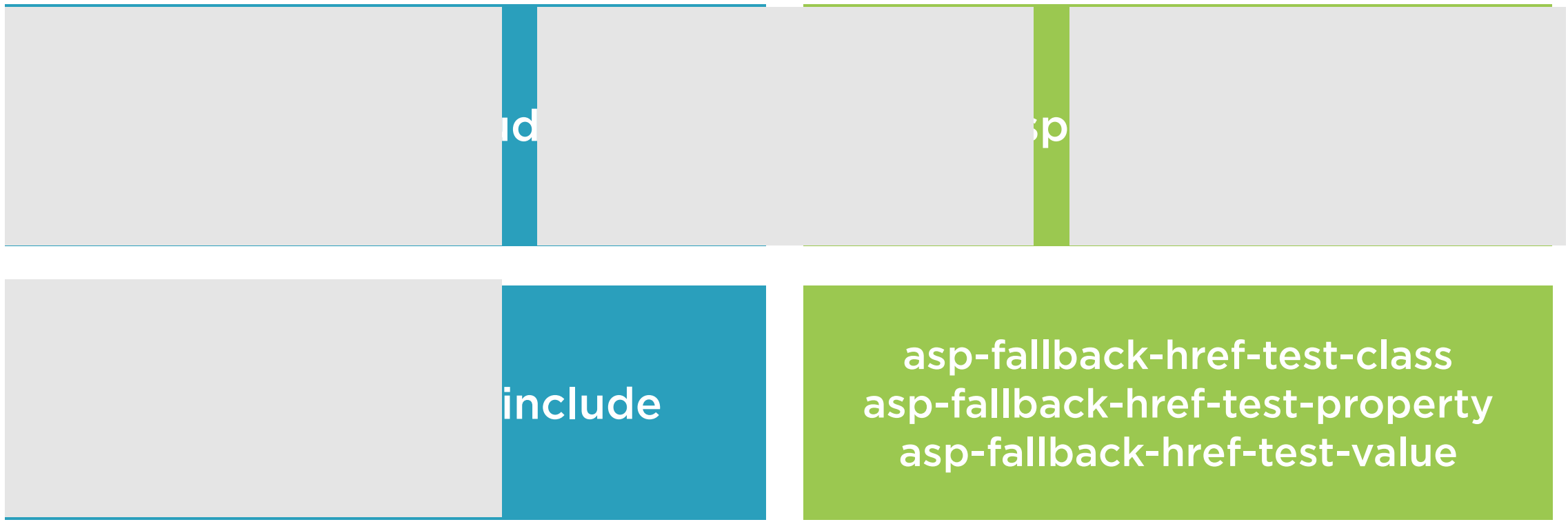

```
<script
        asp-src-include="Scripts/**/*.js"
        asp-src-exclude="Scripts/TempScripts/*.js">
</script>
```

JavaScript Tag Helper

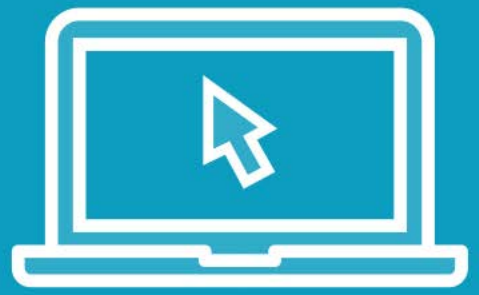
CSS Tag Helper

```
<link asp-href-include="/lib/bootstrap/dist/**/.min.css"
    asp-fallback-href="lib/bootstrap/dist/css/bootstrap.min.css"
    asp-fallback-test-class="hidden“
    asp-fallback-test-property="visibility"
    asp-fallback-test-value="hidden" />
```

CSS Tag Helper



Demo



Using the JavaScript and CSS tag helpers

```
<img class="img-responsive"
alt="Bethany's Pie Shop'"
src="~/Images/bethanylogo.png"
asp-append-version="true" />
```

◀ Force download of image

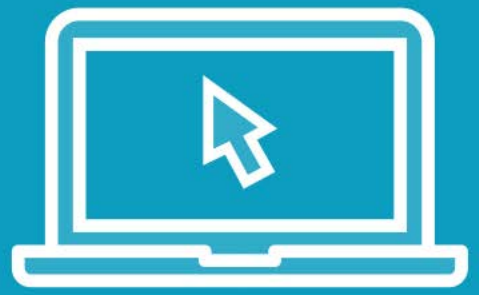
```
<img class="img-responsive"
alt="Bethany's Pie Shop'"
src="/Images/bethanylogo.png?v
=RxwwTsdaEj_HASHFSDA9Yk-
enpNQuclHL-Uxef9CUjM" />
```

◀ Generated image tag

Environment Tag Helper

```
<environment names="Development">
    <link rel="stylesheet" href="~/Content/site.css" />
    <script src="~/lib/bootstrap/dist/js/bootstrap.js"></script>
</environment>

<environment names="Staging,Production">

    <link rel="stylesheet" href="~/Content/site.min.css"
        asp-append-version="true" />

    <script src="~/lib/bootstrap/dist/js/bootstrap.min.js"></script>

</environment>
```

Demo



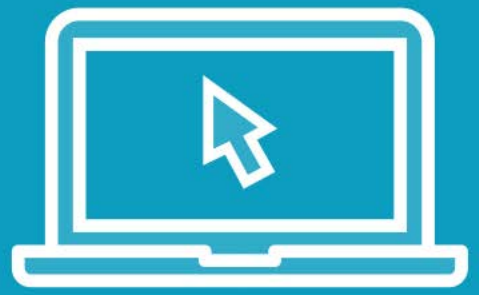
Using other tag helpers

Creating Custom Tag Helpers

```
public class EmailTagHelper: TagHelper
{
    public override void Process(TagHelperContext context,
        TagHelperOutput output)
    { ... }
}
```

Creating Custom Tag Helpers

Demo



Creating a progress tag helper

Creating a shorthand tag helper

Changing the content around the tag

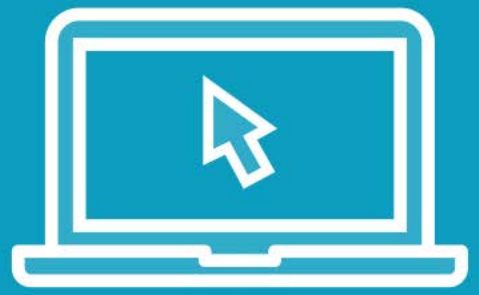
```
@if (SignInManager.IsSignedIn(User))
{
    <li>
        <a asp-controller="PieGift"
           asp-action="Index">Send a pie</a></li>
}
```

This Works...

```
<li condition="SignInManager.IsSignedIn(User)">
    <a asp-controller="PieGift" asp-action="Index">
        Send a pie
    </a>
</li>
```

But This Is Better!

Demo

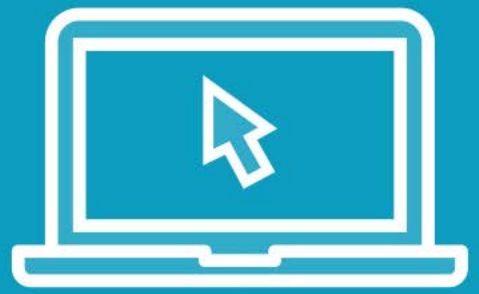


Creating a conditional tag helper

```
@tagHelperPrefix "custom:"

<custom:div progress-value="33">
</custom:div>
```

Explicit Usage of a Tag Helper

Demo



Using tagHelperPrefix

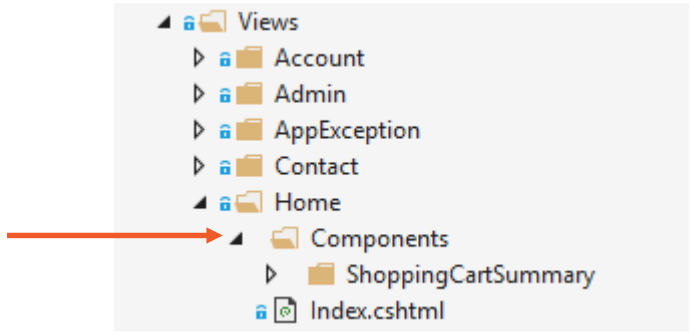
Async View Components

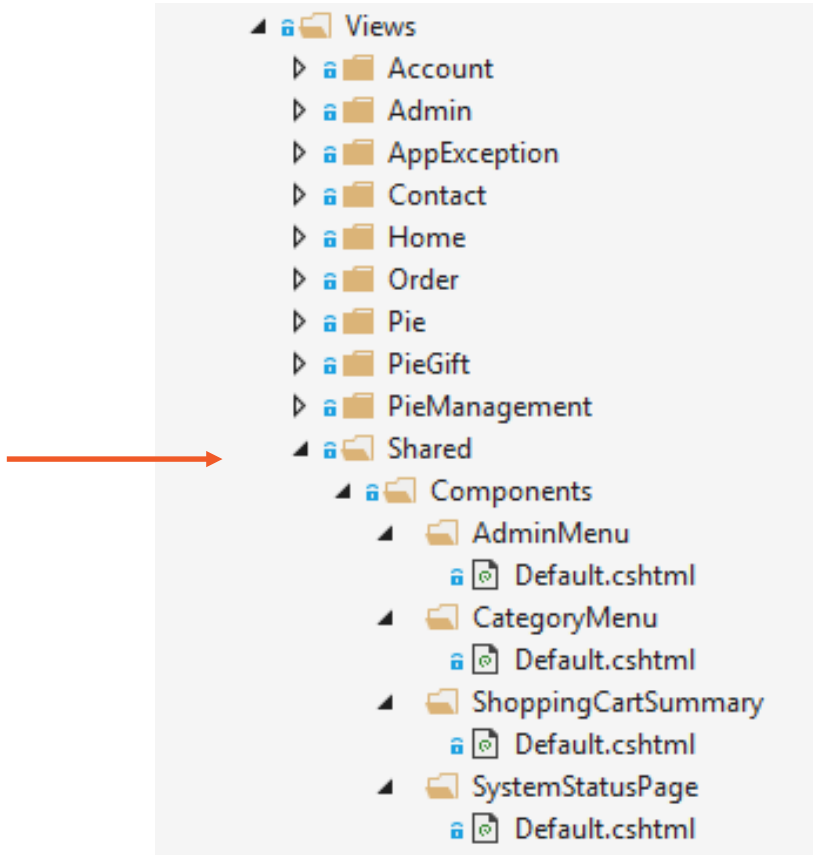
Recap: View Components

Creating a View Component

```
public class ShoppingCartSummary : ViewComponent
{
    public IViewComponentResult Invoke()
    {
        return View(model);
    }

}
```

View Location

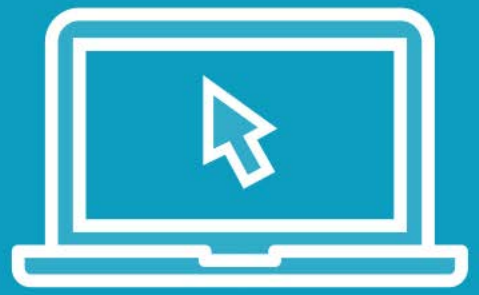




Async View Components

```
public class AsyncComponent : ViewComponent
{
    public async Task<IViewComponentResult> InvokeAsync()
    { ... }
}
```

Demo



Creating an async view component

A question from Bethany

- I want to go international with my store so I want to support multiple languages. Can you support this?

Localizing the Application

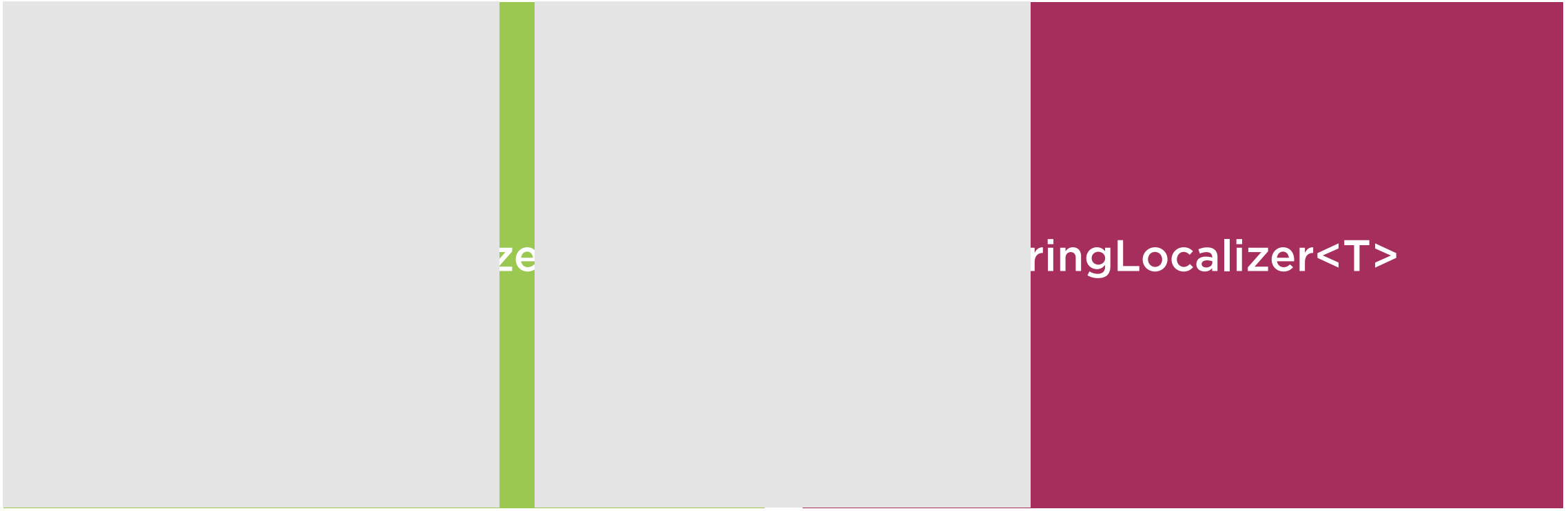
Similar to previous editions of ASP.NET

Based on *.resx

Using the resources has changed

- Strongly-typed accessing is gone

Localization in ASP.NET Core



IStringLocalizer

```
public class HomeController : Controller
{
    private readonly IStringLocalizer<HomeController>
        _stringLocalizer;

    public HomeController(
        IStringLocalizer<HomeController> stringLocalizer)
    { ... }

    ViewBag.PageTitle = _stringLocalizer["PageTitle"];

}
```

IStringLocalizer

stringLocalizer["PageTitle"]

?

Value from resx file

PageTitle

Required Configuration

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddLocalization(
        opts => { opts.ResourcesPath = "Resources"; });

    services.AddMvc()
        .AddViewLocalization(
            LanguageViewLocationExpanderFormat.Suffix,
            opts => { opts.ResourcesPath = "Resources"; });

}
```

Supporting Language Selection

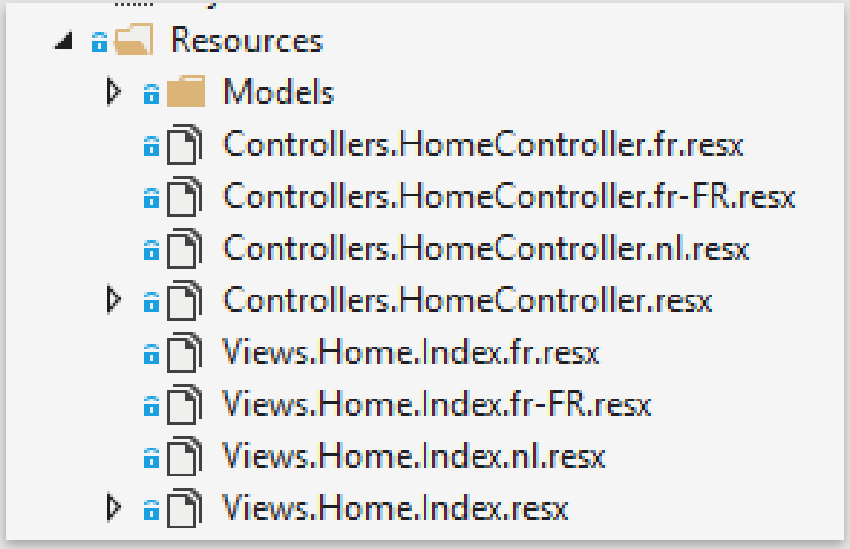
```
services.Configure<RequestLocalizationOptions>(
options =>
{
    var supportedCultures = new List<CultureInfo>
    {
        new CultureInfo("fr"),
        new CultureInfo("fr-FR"),
        new CultureInfo("nl"),
        new CultureInfo("nl-BE")

    };


    options.DefaultRequestCulture = new RequestCulture("en-US");

    options.SupportedCultures = supportedCultures;

    options.SupportedUICultures = supportedCultures;

});
```

Using the RequestLocalizationMiddleware

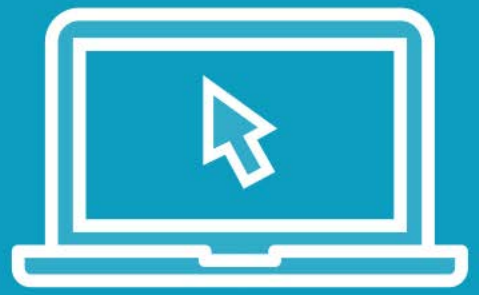
```
public void Configure()
{
    app.UseRequestLocalization(
        app.ApplicationServices.GetService
            <IOptions<RequestLocalizationOptions>>().Value);
}
```

Support for Localization

Structure of the Resources Folder



Demo



Supporting localization in the application

Summary

- Tag helpers wrap functionality
- Asynchronous view components allow calling asynchronous APIs
- New way of localizing our application

Up next:

Creating more discoverable pages