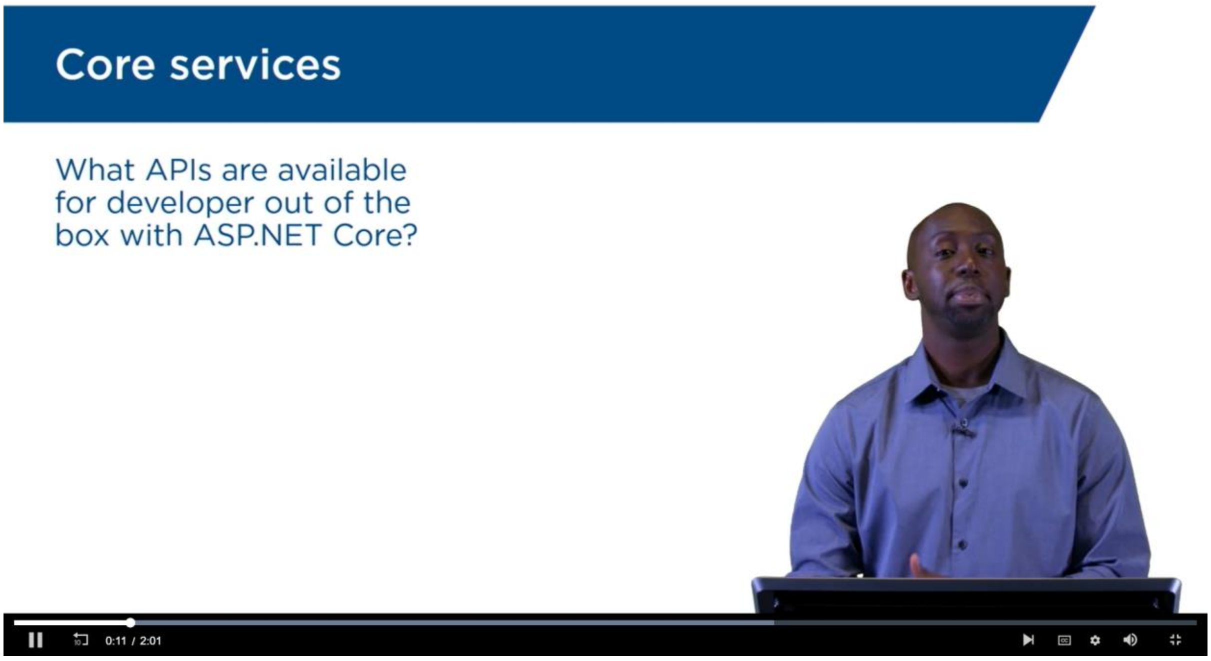
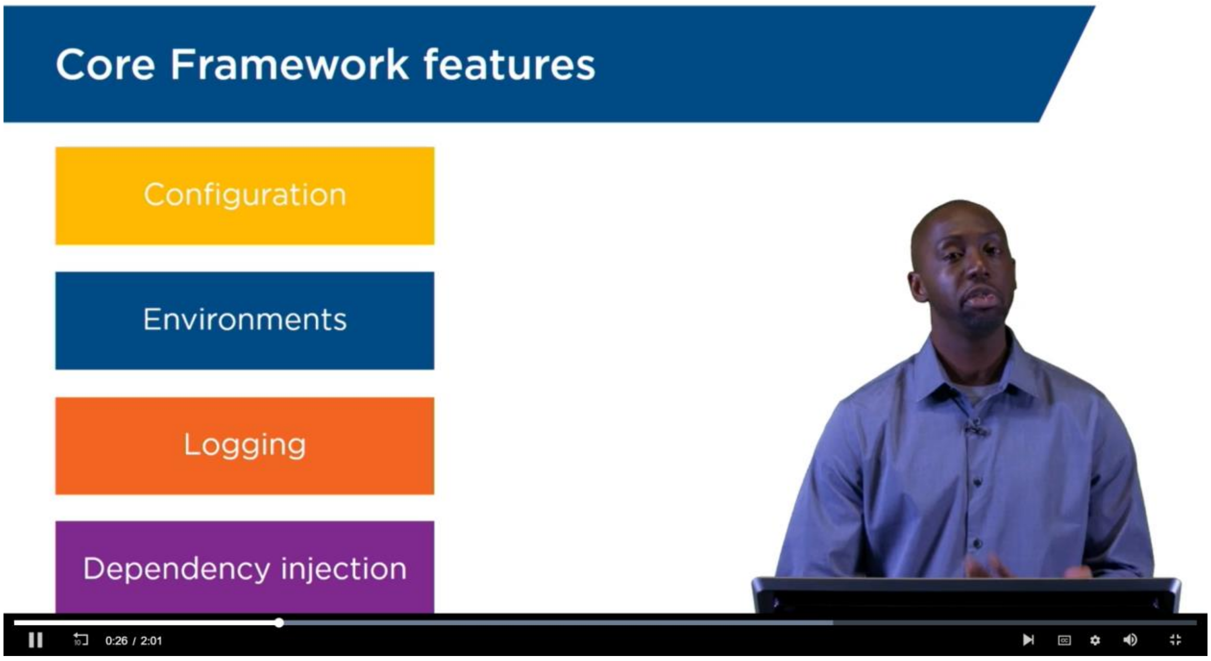


Core services

What APIs are available for developer out of the box with ASP.NET Core?



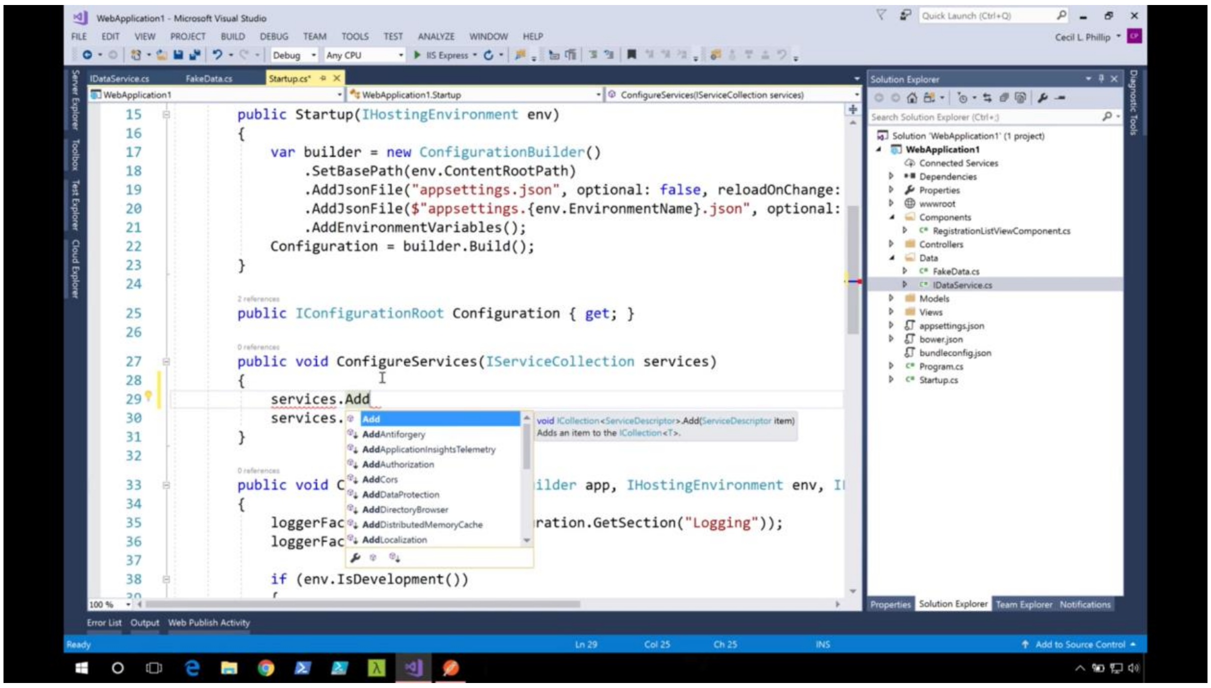
Core Framework features

Configuration

Environments

Logging

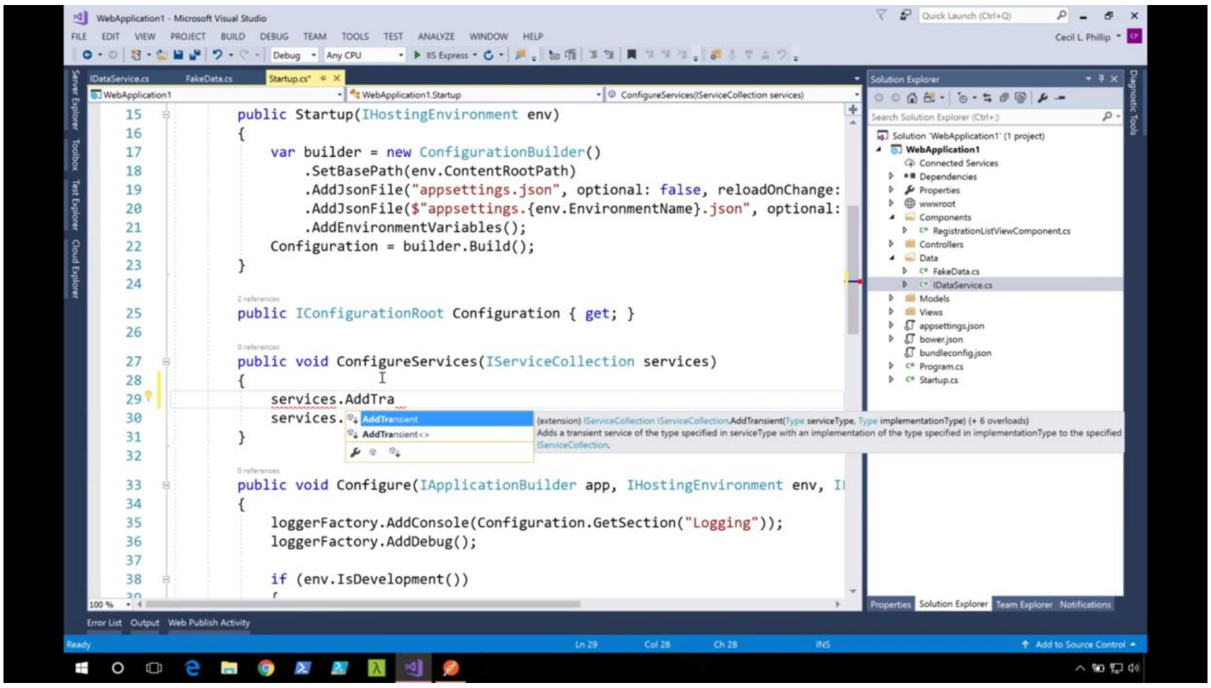
Dependency injection



```
        public Startup(IHostingEnvironment env)
        {
            var builder = new ConfigurationBuilder()
                .SetBasePath(env.ContentRootPath)
                .AddJsonFile("appsettings.json", optional: false, reloadOnChange:
                .AddJsonFile($"appsettings.{env.EnvironmentName}.json", optional:
                .AddEnvironmentVariables();
            Configuration = builder.Build();
        }

        2 references
        public IConfigurationRoot Configuration { get; }

        0 references
        public void ConfigureServices(IServiceCollection services)
        {
            services.Add
            services.
        }

        0 references
        public void C                         ilder app, IHostingEnvironment env, I
        {
            loggerFac                         ration.GetSection("Logging"));
            loggerFac

            if (env.IsDevelopment())
```

```
Add
AddAntiforgery
AddApplicationInsightsTelemetry
AddAuthorization
AddCors
AddDataProtection
AddDirectoryBrowser
AddDistributedMemoryCache
AddLocalization
```

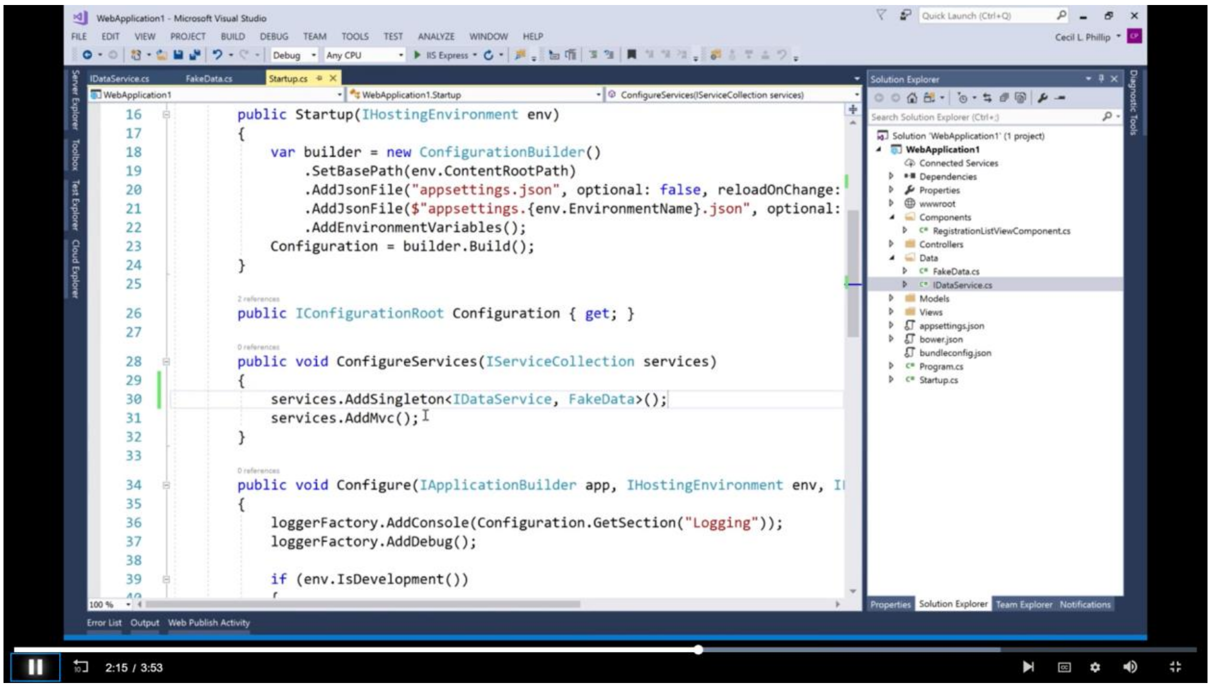
void ICollection<ServiceDescriptor>.Add(ServiceDescriptor item)
Adds an item to the ICollection<T>.



```
        public Startup(IHostingEnvironment env)
        {
            var builder = new ConfigurationBuilder()
                .SetBasePath(env.ContentRootPath)
                .AddJsonFile("appsettings.json", optional: false, reloadOnChange:
                .AddJsonFile($"appsettings.{env.EnvironmentName}.json", optional:
                .AddEnvironmentVariables();
            Configuration = builder.Build();
        }

        2 references
        public IConfigurationRoot Configuration { get; }

        0 references
        public void ConfigureServices(IServiceCollection services)
        {
            services.AddTra
            services.
        }

        0 references
        public void Configure(IApplicationBuilder app, IHostingEnvironment env, I
        {
            loggerFactory.AddConsole(Configuration.GetSection("Logging"));
            loggerFactory.AddDebug();

            if (env.IsDevelopment())
```

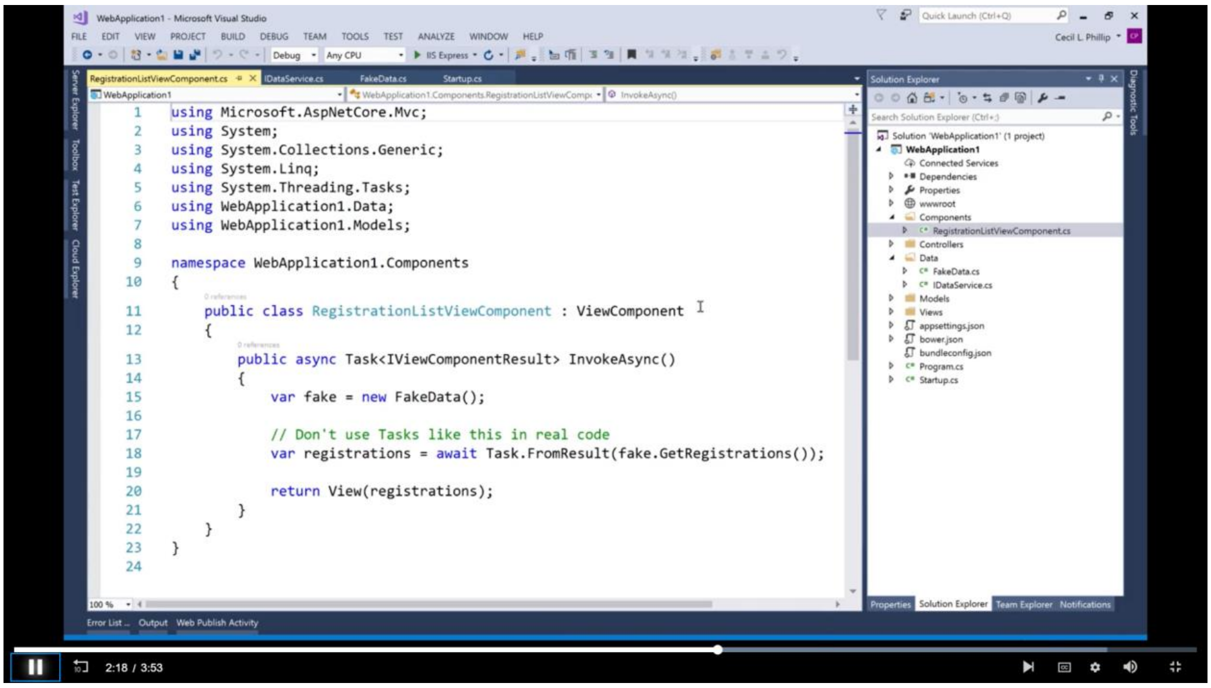
```
AddTransient
AddTransient<>
```

(extension) IServiceCollection IServiceCollection.AddTransient(Type serviceType, Type implementationType) (+ 6 overloads)
Adds a transient service of the type specified in serviceType with an implementation of the type specified in implementationType to the specified IServiceCollection.

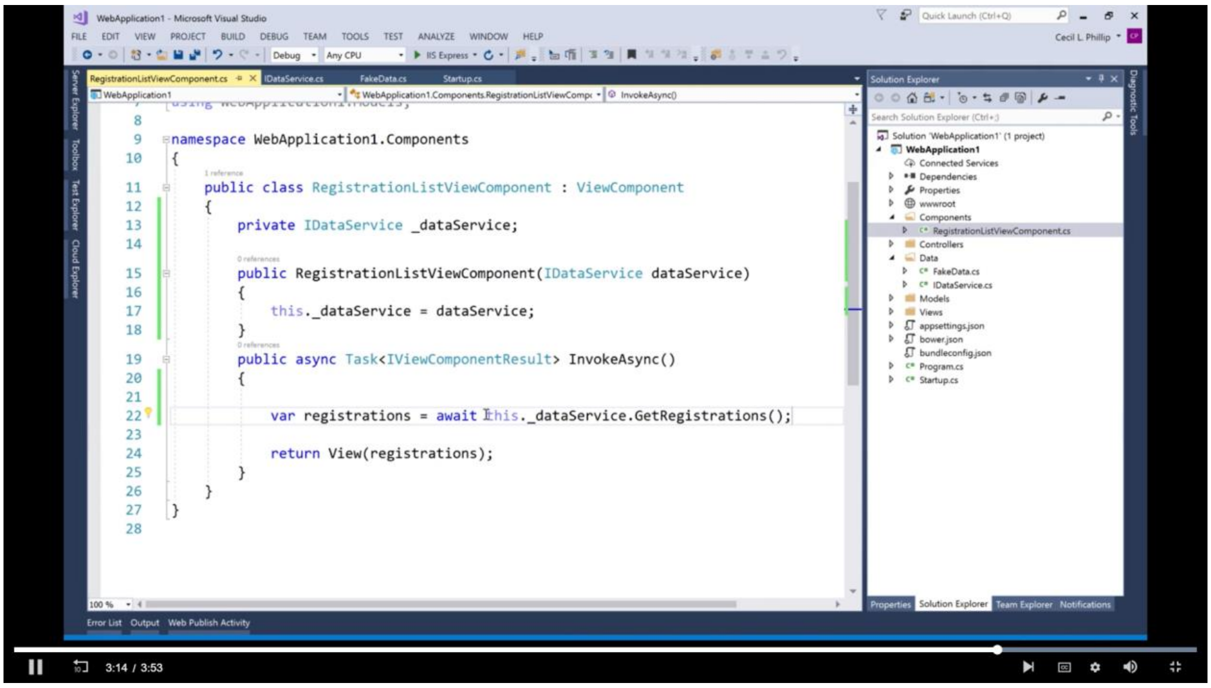


```
        public Startup(IHostingEnvironment env)
        {
            var builder = new ConfigurationBuilder()
                .SetBasePath(env.ContentRootPath)
                .AddJsonFile("appsettings.json", optional: false, reloadOnChange:
                .AddJsonFile($"appsettings.{env.EnvironmentName}.json", optional:
                .AddEnvironmentVariables();
            Configuration = builder.Build();
        }

        2 references
        public IConfigurationRoot Configuration { get; }

        0 references
        public void ConfigureServices(IServiceCollection services)
        {
            services.AddSingleton<IDataService, FakeData>();
            services.AddMvc();
        }

        0 references
        public void Configure(IApplicationBuilder app, IHostingEnvironment env, I
        {
            loggerFactory.AddConsole(Configuration.GetSection("Logging"));
            loggerFactory.AddDebug();

            if (env.IsDevelopment())
```

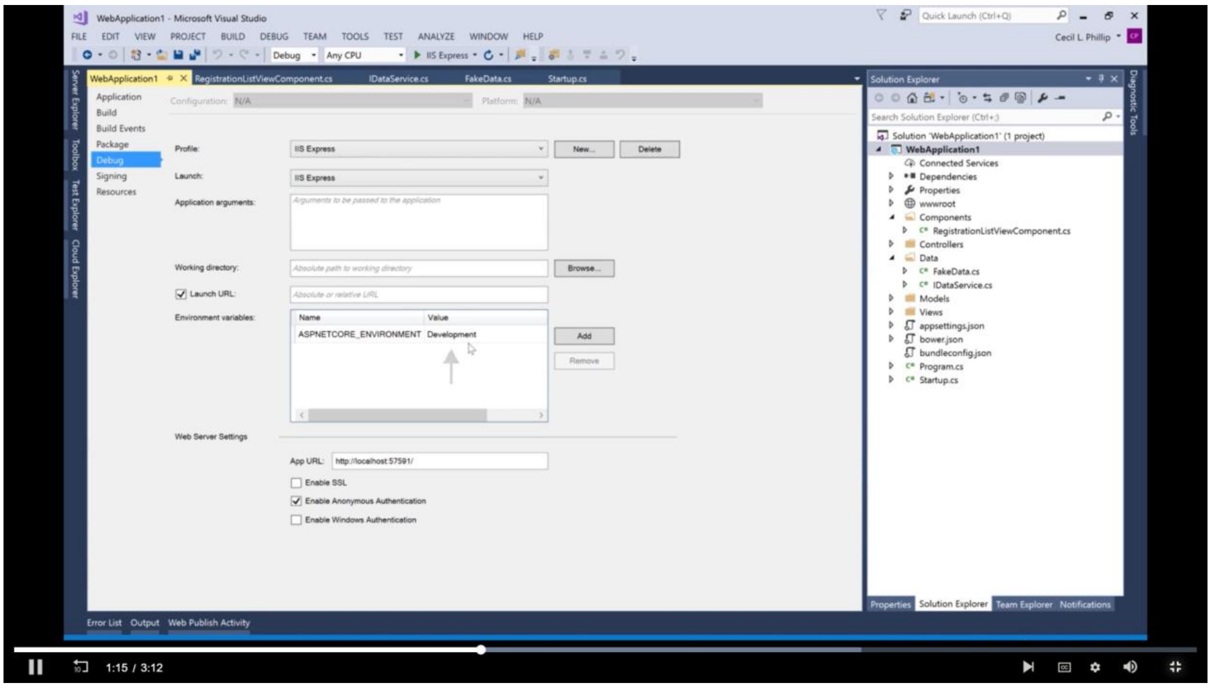
I want to change this code



```
using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using WebApplication1.Data;
using WebApplication1.Models;

namespace WebApplication1.Components
{
    0 references
    public class RegistrationListViewComponent : ViewComponent
    {
        0 references
        public async Task<IViewComponentResult> InvokeAsync()
        {
            var fake = new FakeData();

            // Don't use Tasks like this in real code
            var registrations = await Task.FromResult(fake.GetRegistrations());

            return View(registrations);
        }
    }
}
```



```
namespace WebApplication1.Components
{
    public class RegistrationListViewComponent : ViewComponent
    {
        private IDataService _dataService;

        public RegistrationListViewComponent(IDataService dataService)
        {
            this._dataService = dataService;
        }
        public async Task<IViewComponentResult> InvokeAsync()
        {

            var registrations = await this._dataService.GetRegistrations();

            return View(registrations);
        }
    }
}
```



Configuration: N/A Platform: N/A

Profile: IIS Express

Launch: IIS Express

Application arguments: Arguments to be passed to the application

Working directory: Absolute path to working directory

[✓] Launch URL: Absolute or relative URL

Environment variables:

Name	Value
ASPNETCORE_ENVIRONMENT	Development

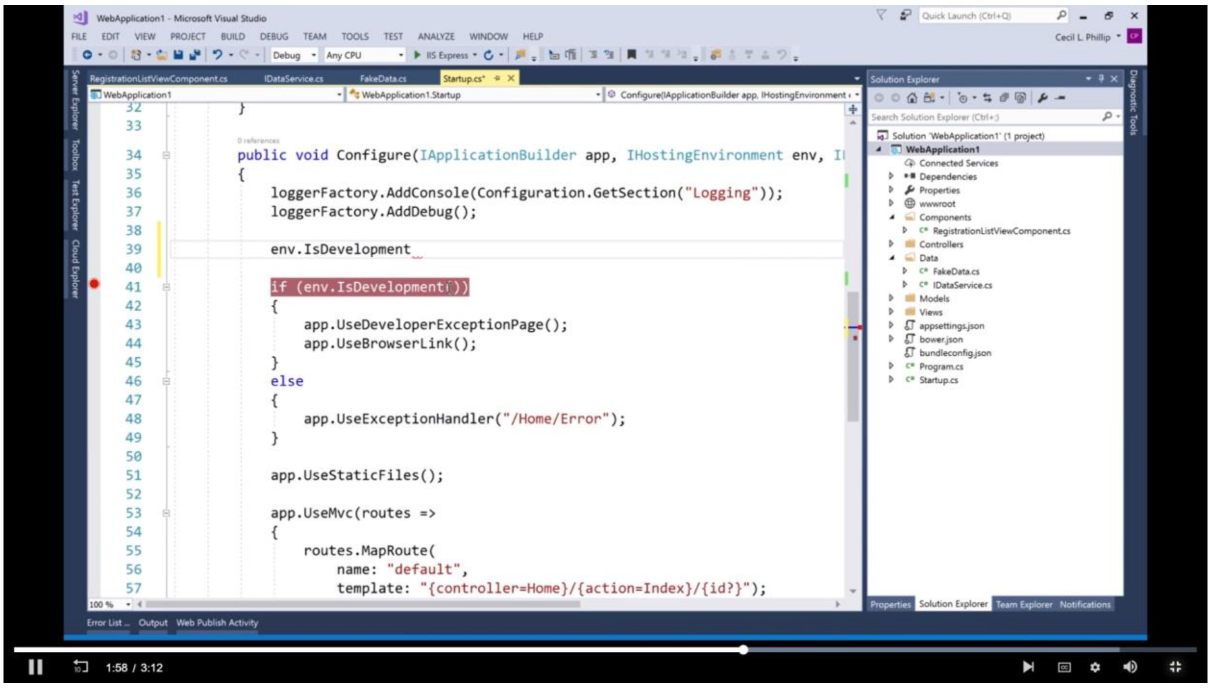
Web Server Settings

App URL: http://localhost:57591/

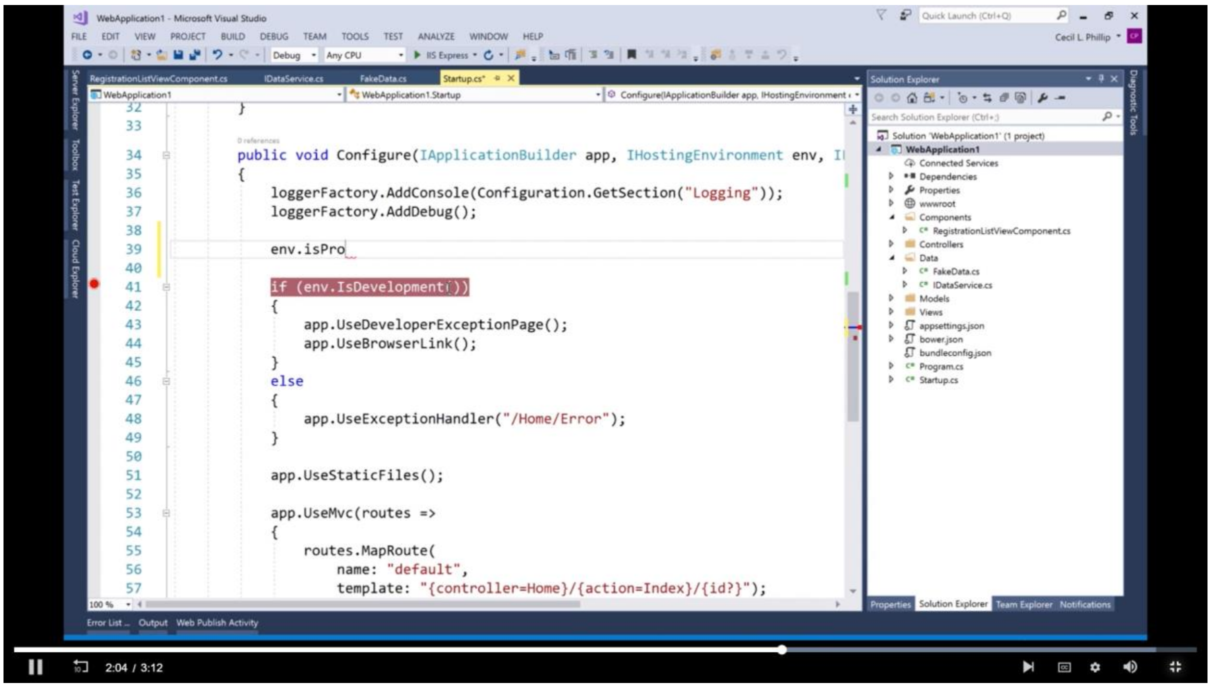
[ ] Enable SSL

[✓] Enable Anonymous Authentication

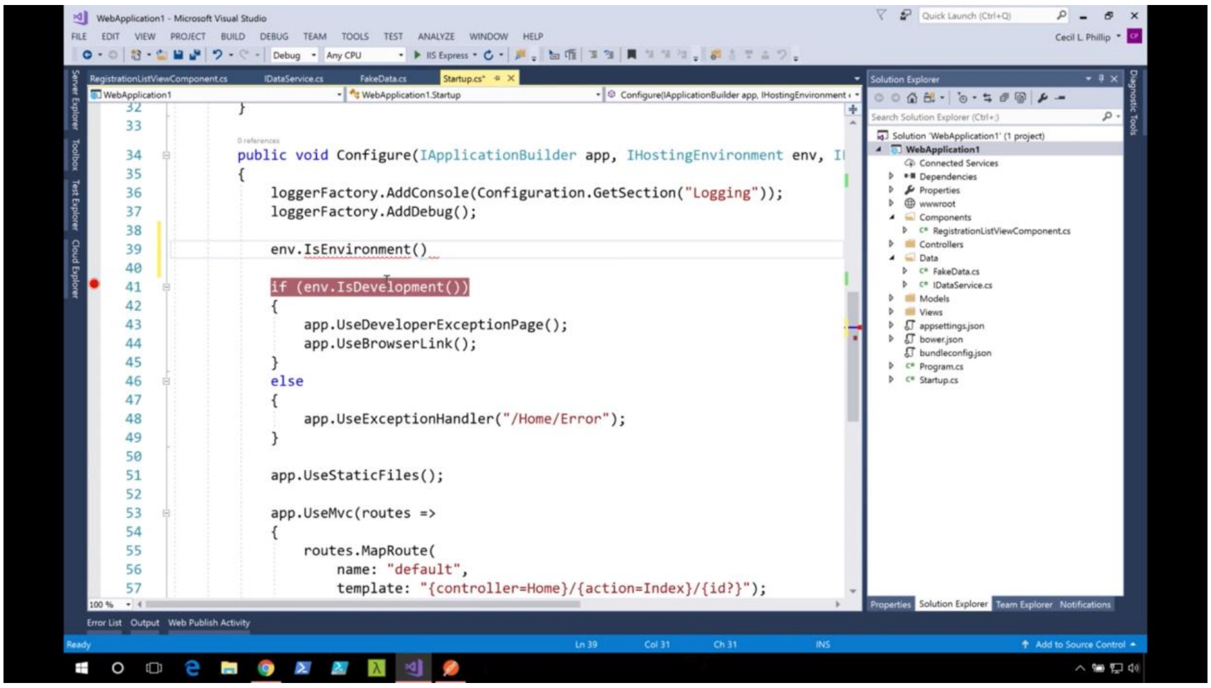
[ ] Enable Windows Authentication



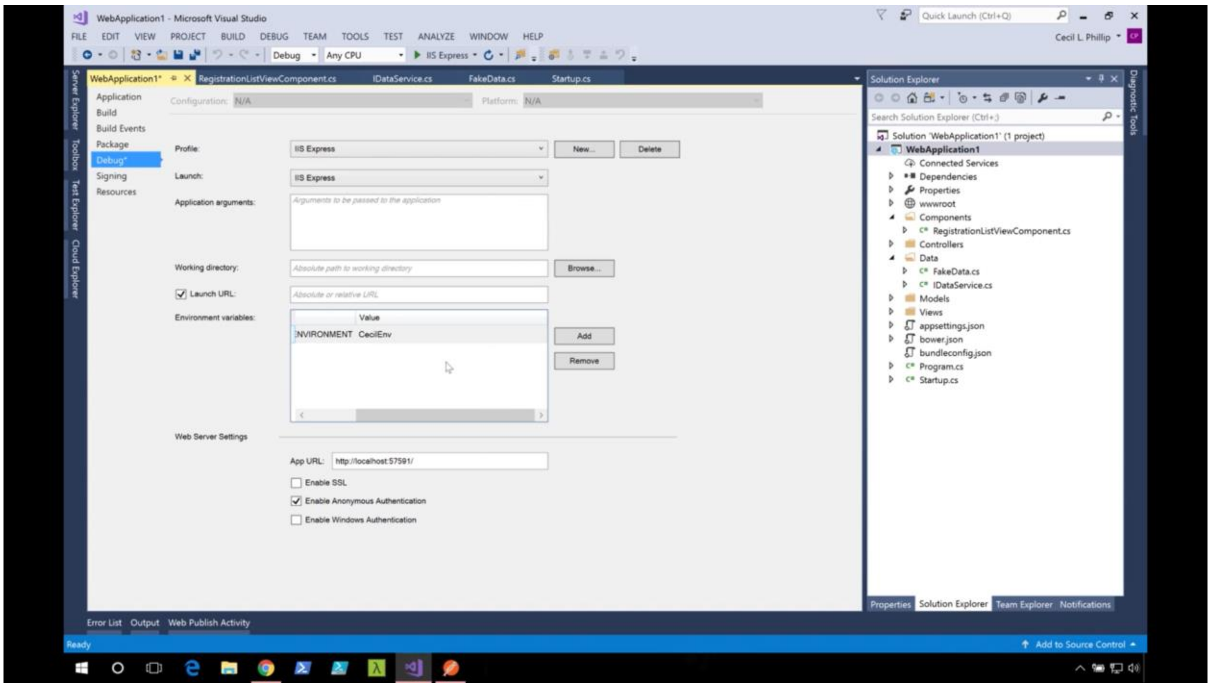
```
        }

        0 references
        public void Configure(IApplicationBuilder app, IHostingEnvironment env, I
        {
            loggerFactory.AddConsole(Configuration.GetSection("Logging"));
            loggerFactory.AddDebug();

            env.IsDevelopment

            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
                app.UseBrowserLink();
            }
            else
            {
                app.UseExceptionHandler("/Home/Error");
            }

            app.UseStaticFiles();

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



```
        }

        0 references
        public void Configure(IApplicationBuilder app, IHostingEnvironment env, I
        {
            loggerFactory.AddConsole(Configuration.GetSection("Logging"));
            loggerFactory.AddDebug();

            env.isPro

            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
                app.UseBrowserLink();
            }
            else
            {
                app.UseExceptionHandler("/Home/Error");
            }

            app.UseStaticFiles();

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



```
        }

        0 references
        public void Configure(IApplicationBuilder app, IHostingEnvironment env, I
        {
            loggerFactory.AddConsole(Configuration.GetSection("Logging"));
            loggerFactory.AddDebug();

            env.IsEnvironment()

            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
                app.UseBrowserLink();
            }
            else
            {
                app.UseExceptionHandler("/Home/Error");
            }

            app.UseStaticFiles();

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



Configuration: N/A Platform: N/A

Profile: IIS Express

Launch: IIS Express

Application arguments: Arguments to be passed to the application

Working directory: Absolute path to working directory

[✓] Launch URL: Absolute or relative URL

Environment variables:

	Value
NVIRONMENT	CecilEnv

Web Server Settings

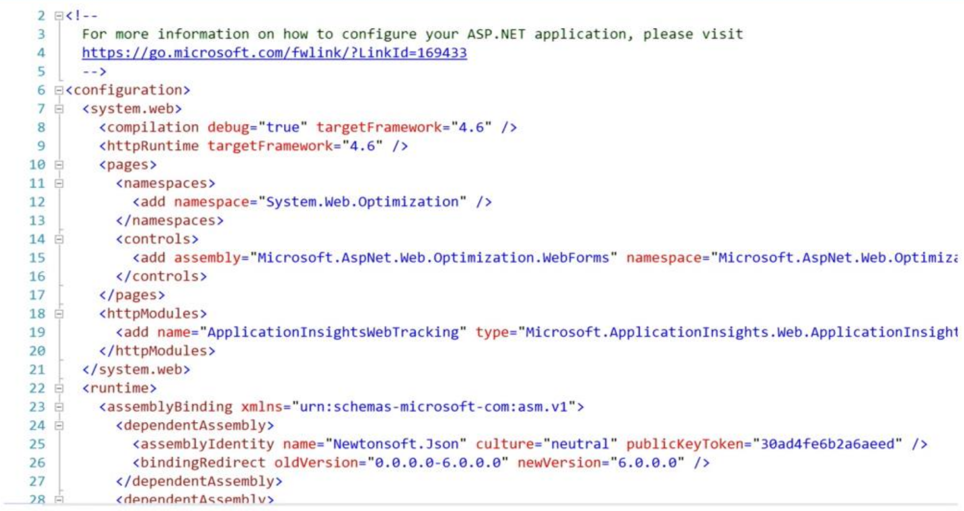
App URL: http://localhost:57591/

[ ] Enable SSL

[✓] Enable Anonymous Authentication

[ ] Enable Windows Authentication

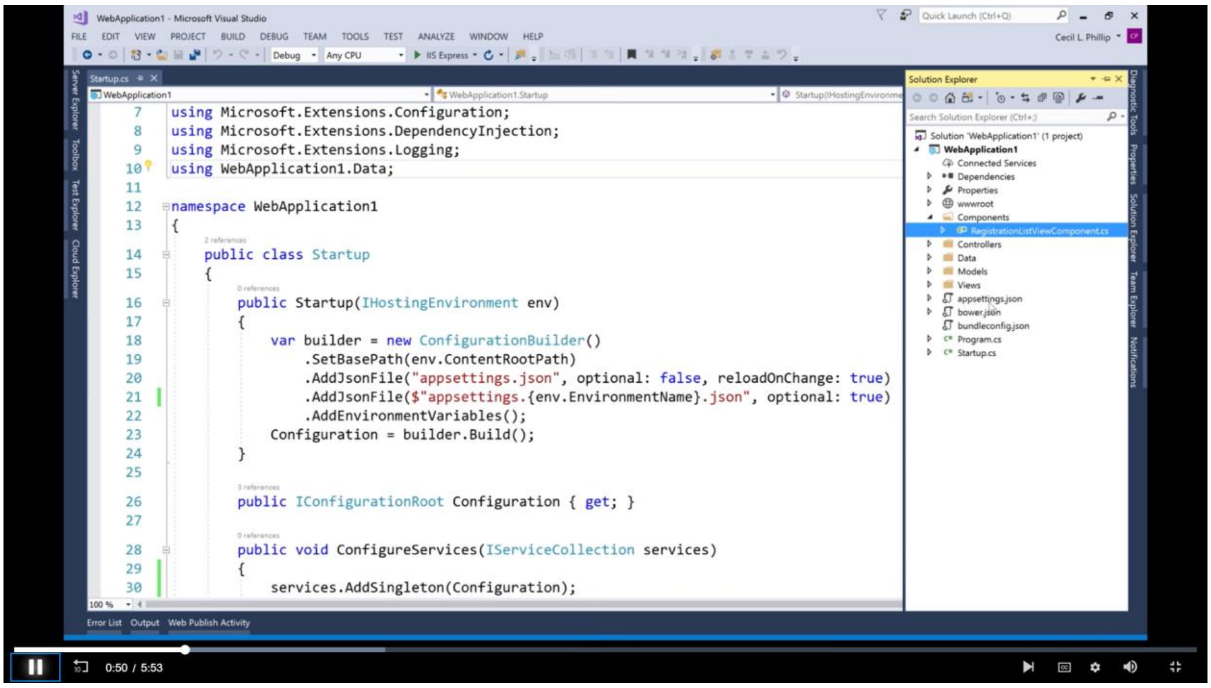
Traditional .NET configuration files



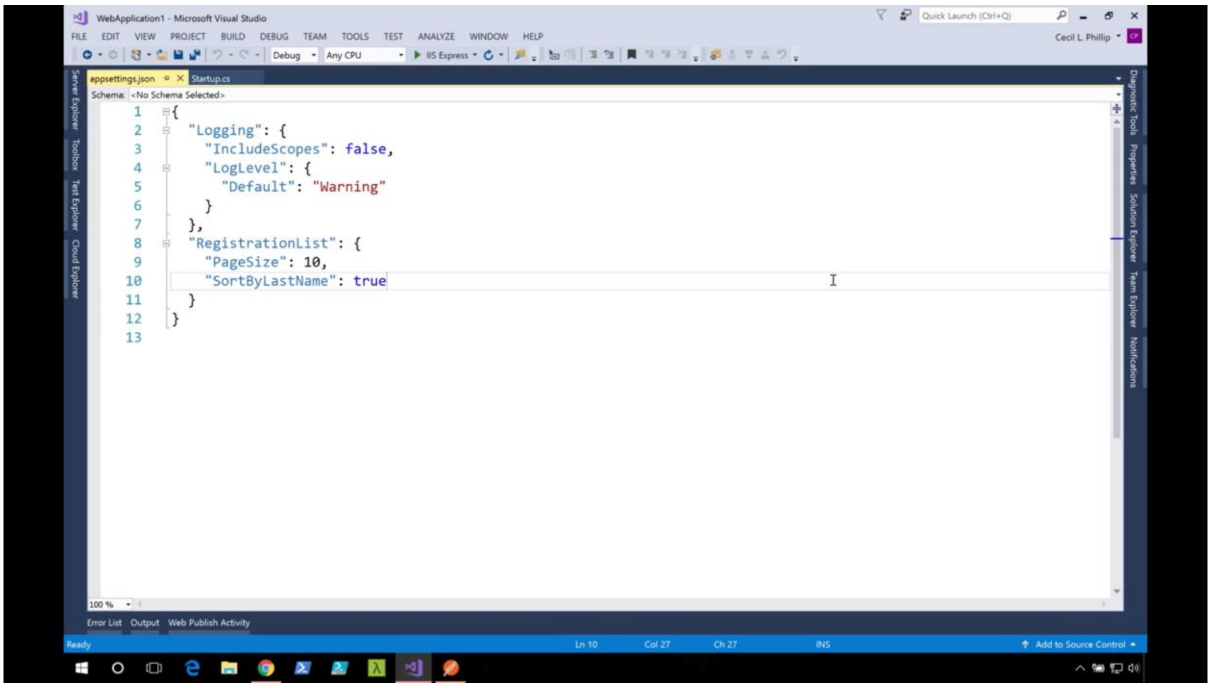
```
<!--
  For more information on how to configure your ASP.NET application, please visit
  https://go.microsoft.com/fwlink/?LinkId=169433
  -->
<configuration>
  <system.web>
    <compilation debug="true" targetFramework="4.6" />
    <httpRuntime targetFramework="4.6" />
    <pages>
      <namespaces>
        <add namespace="System.Web.Optimization" />
      </namespaces>
      <controls>
        <add assembly="Microsoft.AspNet.Web.Optimization.WebForms" namespace="Microsoft.AspNet.Web.Optimiza
      </controls>
    </pages>
    <httpModules>
      <add name="ApplicationInsightsWebTracking" type="Microsoft.ApplicationInsights.Web.ApplicationInsight
    </httpModules>
  </system.web>
  <runtime>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
      <dependentAssembly>
        <assemblyIdentity name="Newtonsoft.Json" culture="neutral" publicKeyToken="30ad4fe6b2a6aeed" />
        <bindingRedirect oldVersion="0.0.0.0-6.0.0.0" newVersion="6.0.0.0" />
      </dependentAssembly>
      <dependentAssembly>
```

New configuration APIs

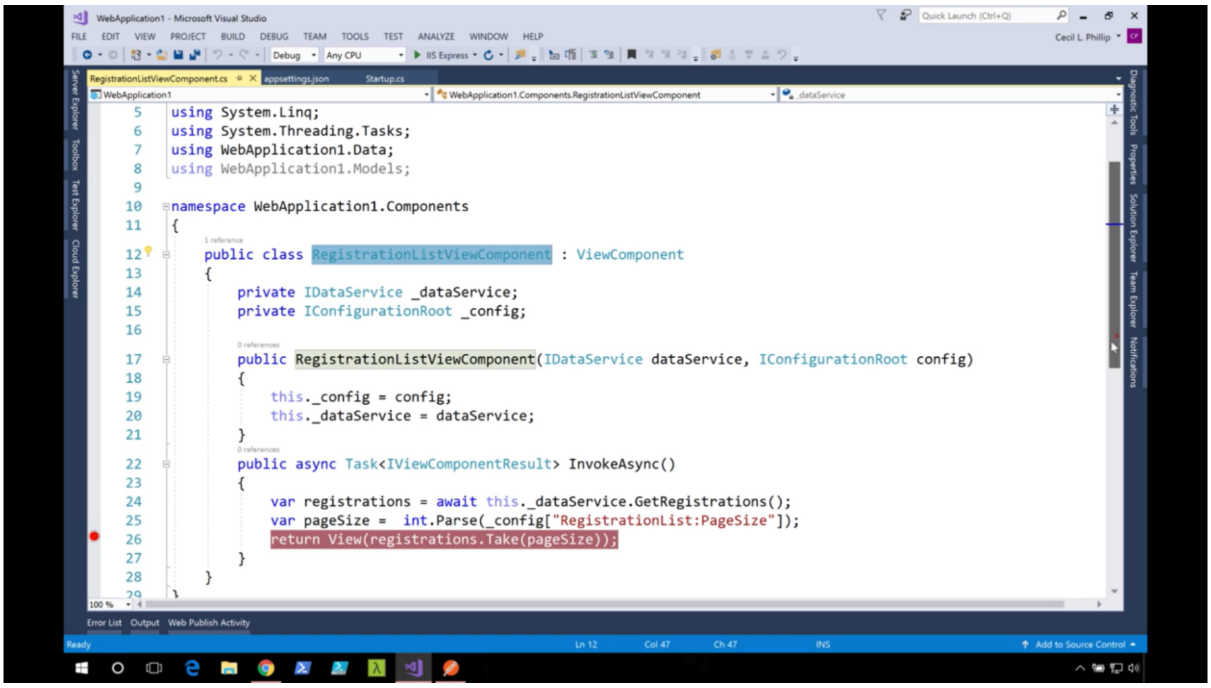
- Use whatever format you want
- Use multiple configuration sources
- Easily extended
- Supports strongly-typed configuration



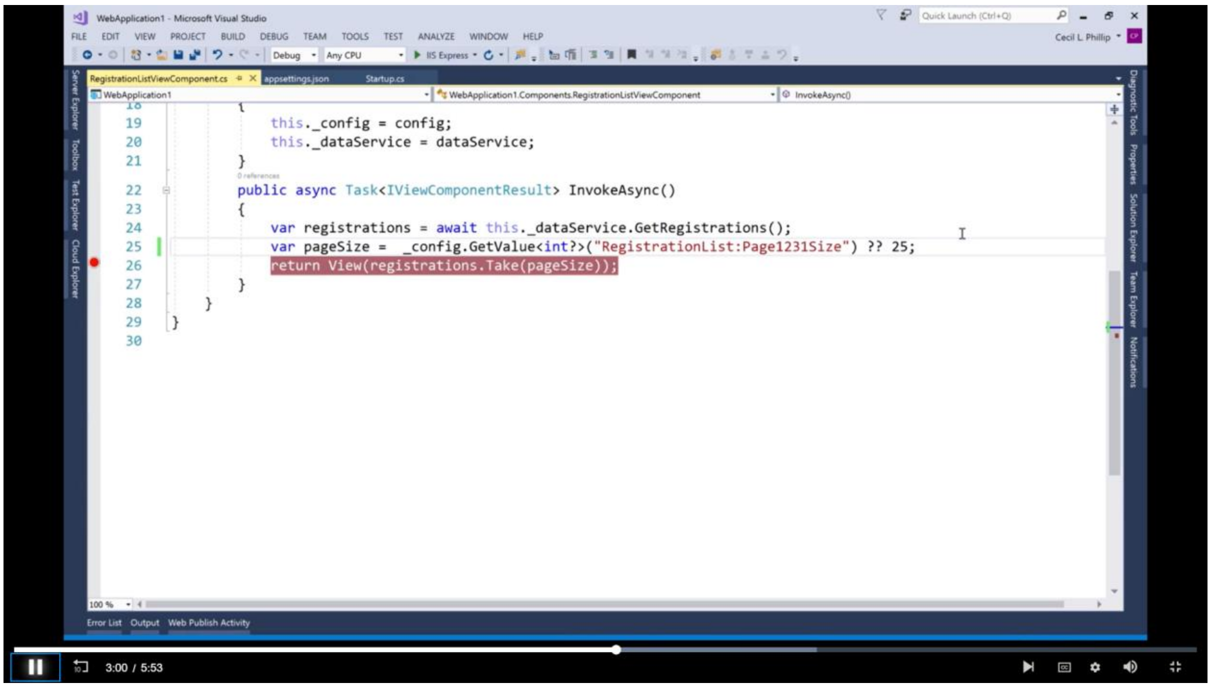
```csharp
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Logging;
using WebApplication1.Data;

namespace WebApplication1
{
    public class Startup
    {
        public Startup(IHostingEnvironment env)
        {
            var builder = new ConfigurationBuilder()
                .SetBasePath(env.ContentRootPath)
                .AddJsonFile("appsettings.json", optional: false, reloadOnChange: true)
                .AddJsonFile($"appsettings.{env.EnvironmentName}.json", optional: true)
                .AddEnvironmentVariables();
            Configuration = builder.Build();
        }

        public IConfigurationRoot Configuration { get; }

        public void ConfigureServices(IServiceCollection services)
        {
            services.AddSingleton(Configuration);
```



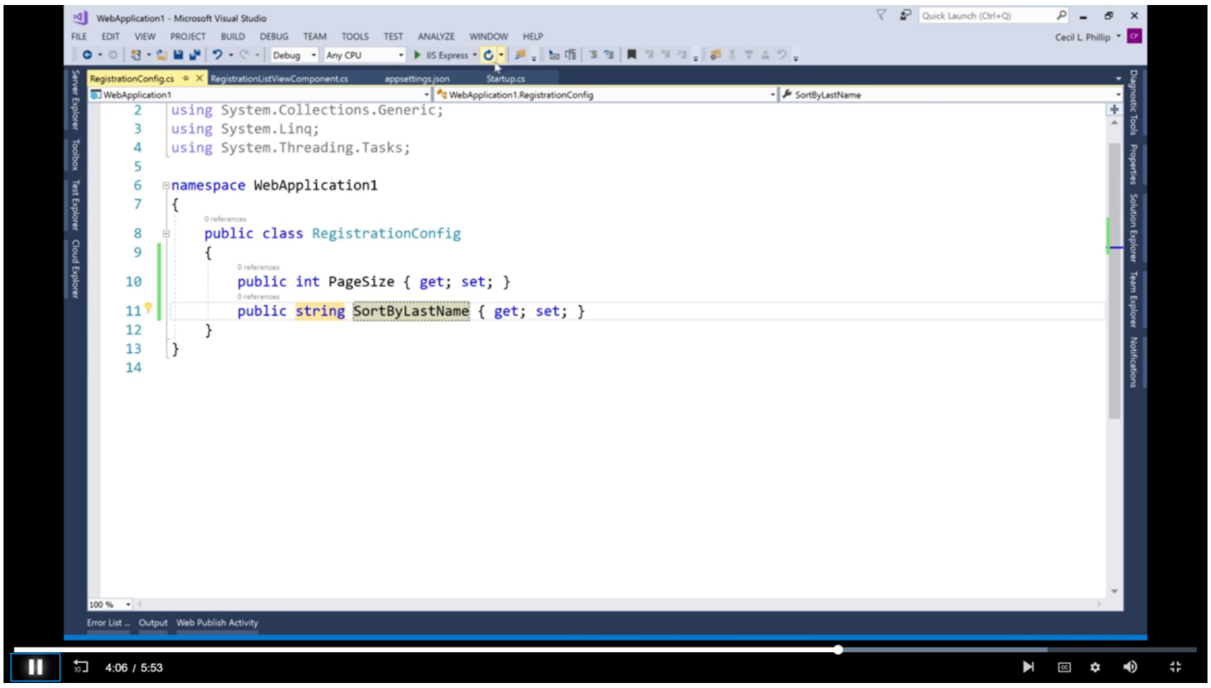
```json
{
  "Logging": {
    "IncludeScopes": false,
    "LogLevel": {
      "Default": "Warning"
    }
  },
  "RegistrationList": {
    "PageSize": 10,
    "SortByLastName": true
  }
}
```



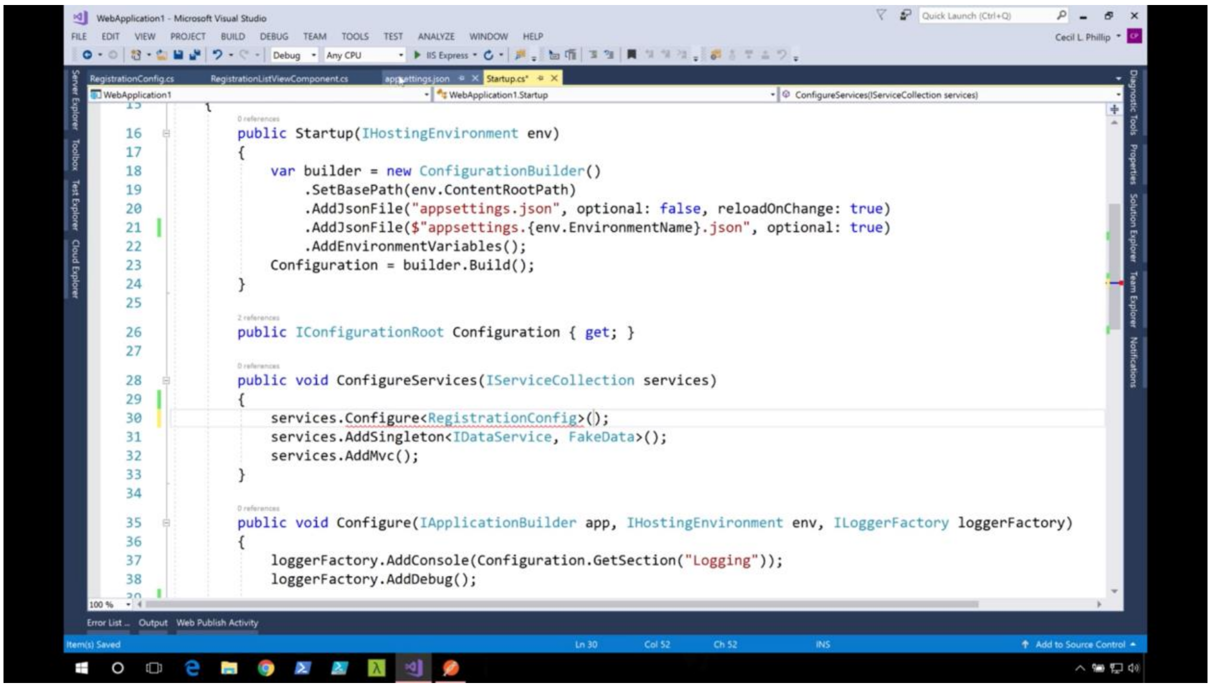
```csharp
using System.Linq;
using System.Threading.Tasks;
using WebApplication1.Data;
using WebApplication1.Models;

namespace WebApplication1.Components
{
    public class RegistrationListViewComponent : ViewComponent
    {
        private IDataService _dataService;
        private IConfigurationRoot _config;

        public RegistrationListViewComponent(IDataService dataService, IConfigurationRoot config)
        {
            this._config = config;
            this._dataService = dataService;
        }
        public async Task<IViewComponentResult> InvokeAsync()
        {
            var registrations = await this._dataService.GetRegistrations();
            var pageSize =  int.Parse(_config["RegistrationList:PageSize"]);
            return View(registrations.Take(pageSize));
        }
    }
}
```


WebApplication1 - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ANALYZE WINDOW HELP
Quick Launch (Ctrl+Q)
Cecil L. Phillip
RegistrationListViewComponent.cs
appsettings.json
Startup.cs
WebApplication1
WebApplication1.Components.RegistrationListViewComponent
InvokeAsync()
100 %
Error List
Output
Web Publish Activity
3:00 / 5:53

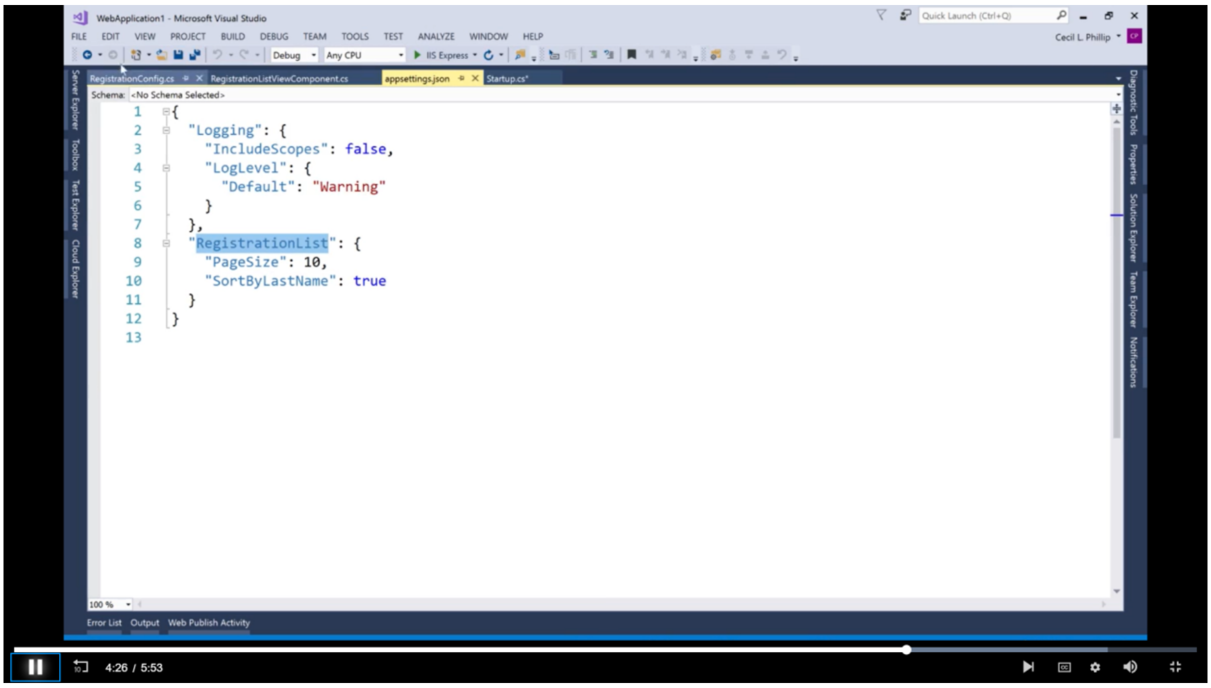
```csharp
        {
            this._config = config;
            this._dataService = dataService;
        }
        public async Task<IViewComponentResult> InvokeAsync()
        {
            var registrations = await this._dataService.GetRegistrations();
            var pageSize =  _config.GetValue<int?>("RegistrationList:Page1231Size") ?? 25;
            return View(registrations.Take(pageSize));
        }
    }
}
```


WebApplication1 - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ANALYZE WINDOW HELP
RegistrationConfig.cs
RegistrationListViewComponent.cs
appsettings.json
Startup.cs
WebApplication1.RegistrationConfig
SortByLastName
4:06 / 5:53

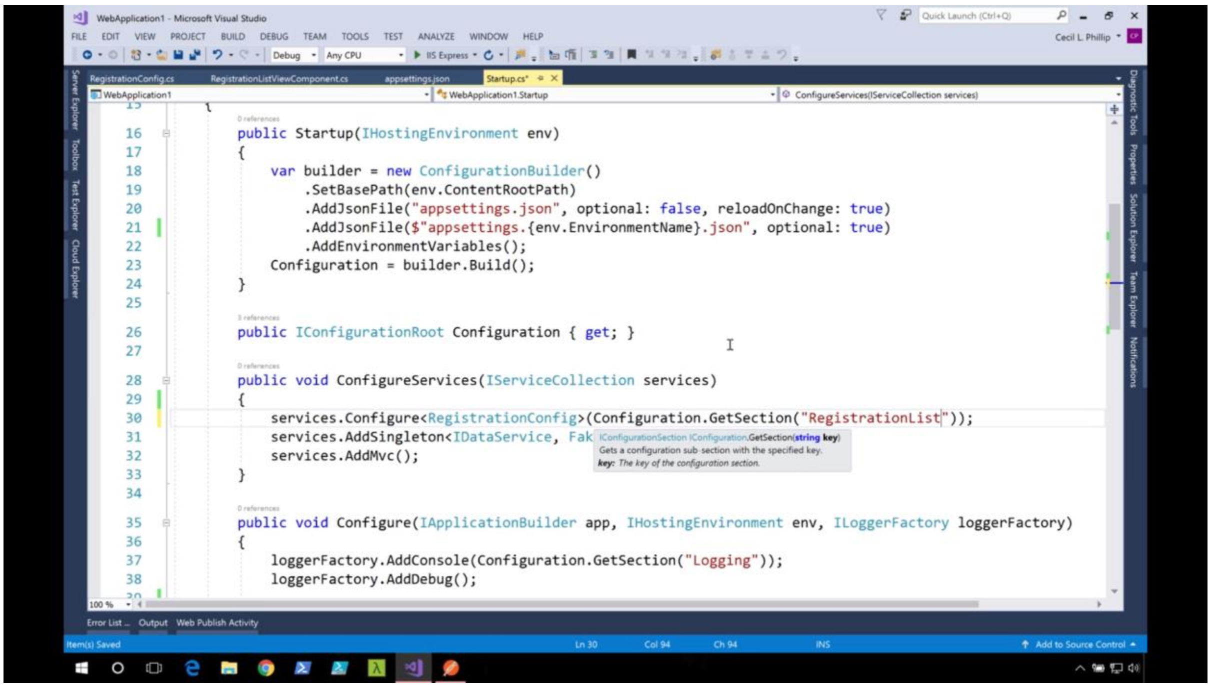
```
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace WebApplication1
{
    public class RegistrationConfig
    {
        public int PageSize { get; set; }
        public string SortByLastName { get; set; }
    }
}
```



```
        public Startup(IHostingEnvironment env)
        {
            var builder = new ConfigurationBuilder()
                .SetBasePath(env.ContentRootPath)
                .AddJsonFile("appsettings.json", optional: false, reloadOnChange: true)
                .AddJsonFile($"appsettings.{env.EnvironmentName}.json", optional: true)
                .AddEnvironmentVariables();
            Configuration = builder.Build();
        }

        public IConfigurationRoot Configuration { get; }

        public void ConfigureServices(IServiceCollection services)
        {
            services.Configure<RegistrationConfig>();
            services.AddSingleton<IDataService, FakeData>();
            services.AddMvc();
        }

        public void Configure(IApplicationBuilder app, IHostingEnvironment env, ILoggerFactory loggerFactory)
        {
            loggerFactory.AddConsole(Configuration.GetSection("Logging"));
            loggerFactory.AddDebug();
```

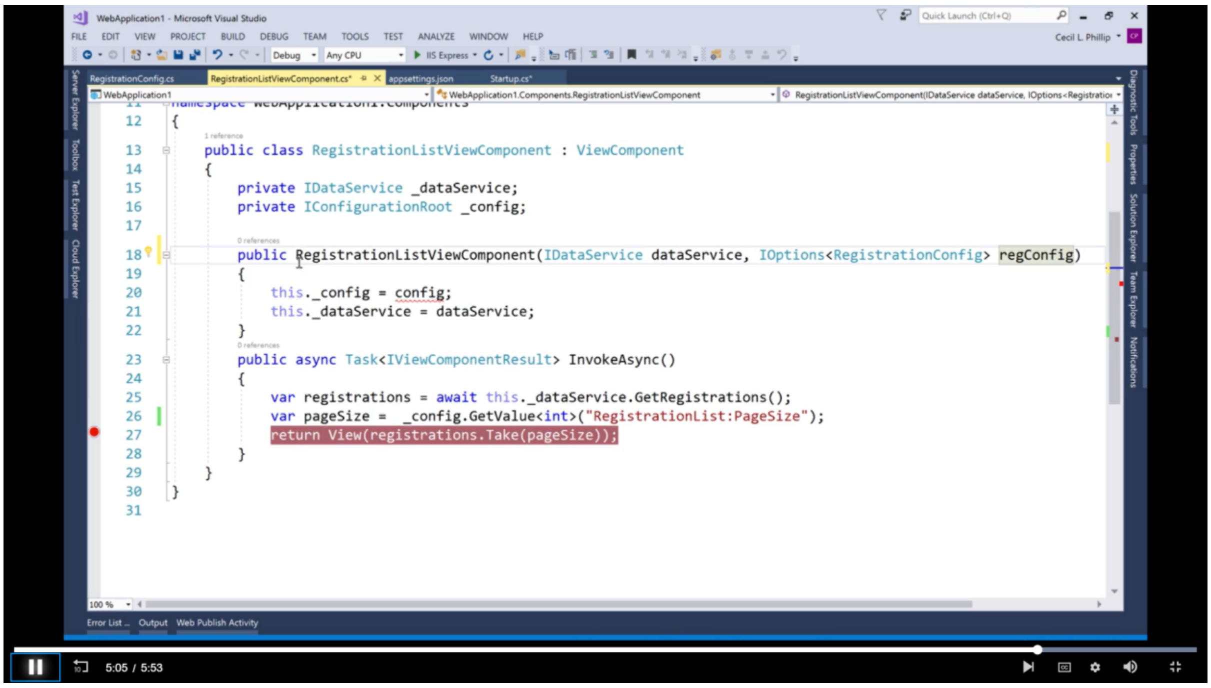

WebApplication1 - Microsoft Visual Studio
RegistrationConfig.cs
RegistrationListViewComponent.cs
appsettings.json
Startup.cs*
Schema: <No Schema Selected>
4:26 / 5:53

```json
{
  "Logging": {
    "IncludeScopes": false,
    "LogLevel": {
      "Default": "Warning"
    }
  },
  "RegistrationList": {
    "PageSize": 10,
    "SortByLastName": true
  }
}
```

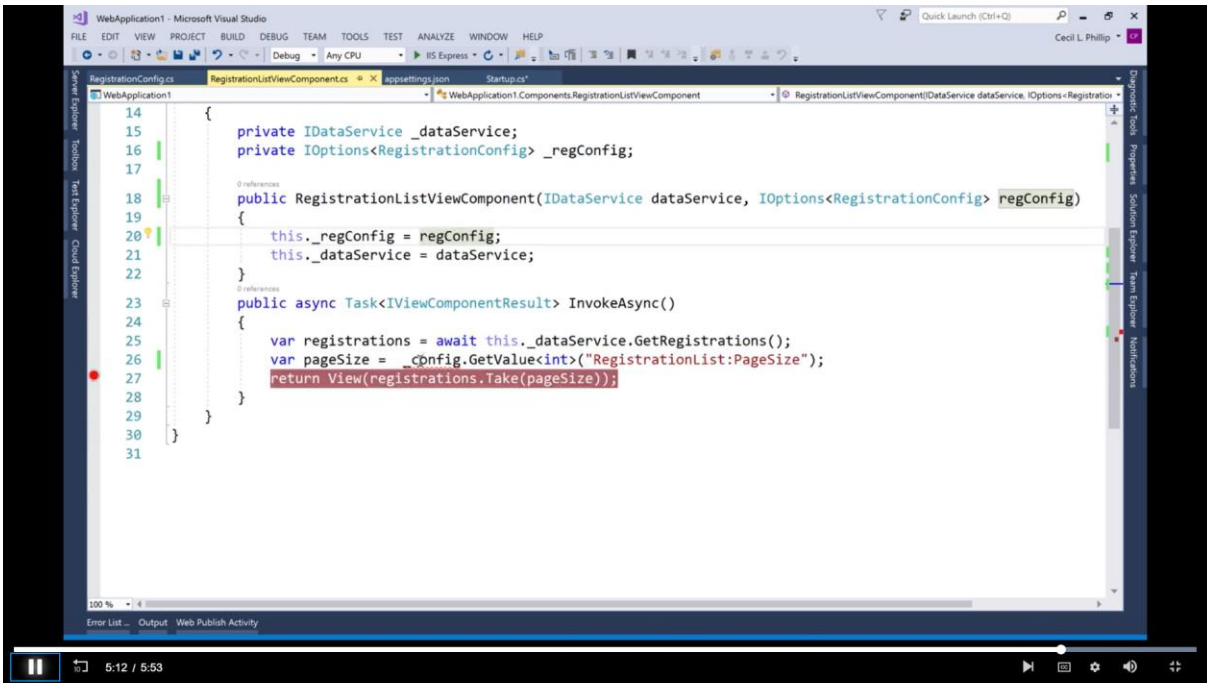

WebApplication1 - Microsoft Visual Studio
Startup.cs*
WebApplication1.Startup
ConfigureServices(IServiceCollection services)
Item(s) Saved
Ln 30
Col 94
Ch 94
INS
Add to Source Control

```csharp
        public Startup(IHostingEnvironment env)
        {
            var builder = new ConfigurationBuilder()
                .SetBasePath(env.ContentRootPath)
                .AddJsonFile("appsettings.json", optional: false, reloadOnChange: true)
                .AddJsonFile($"appsettings.{env.EnvironmentName}.json", optional: true)
                .AddEnvironmentVariables();
            Configuration = builder.Build();
        }

        public IConfigurationRoot Configuration { get; }

        public void ConfigureServices(IServiceCollection services)
        {
            services.Configure<RegistrationConfig>(Configuration.GetSection("RegistrationList"));
            services.AddSingleton<IDataService, Fak
            services.AddMvc();
        }

        public void Configure(IApplicationBuilder app, IHostingEnvironment env, ILoggerFactory loggerFactory)
        {
            loggerFactory.AddConsole(Configuration.GetSection("Logging"));
            loggerFactory.AddDebug();
```

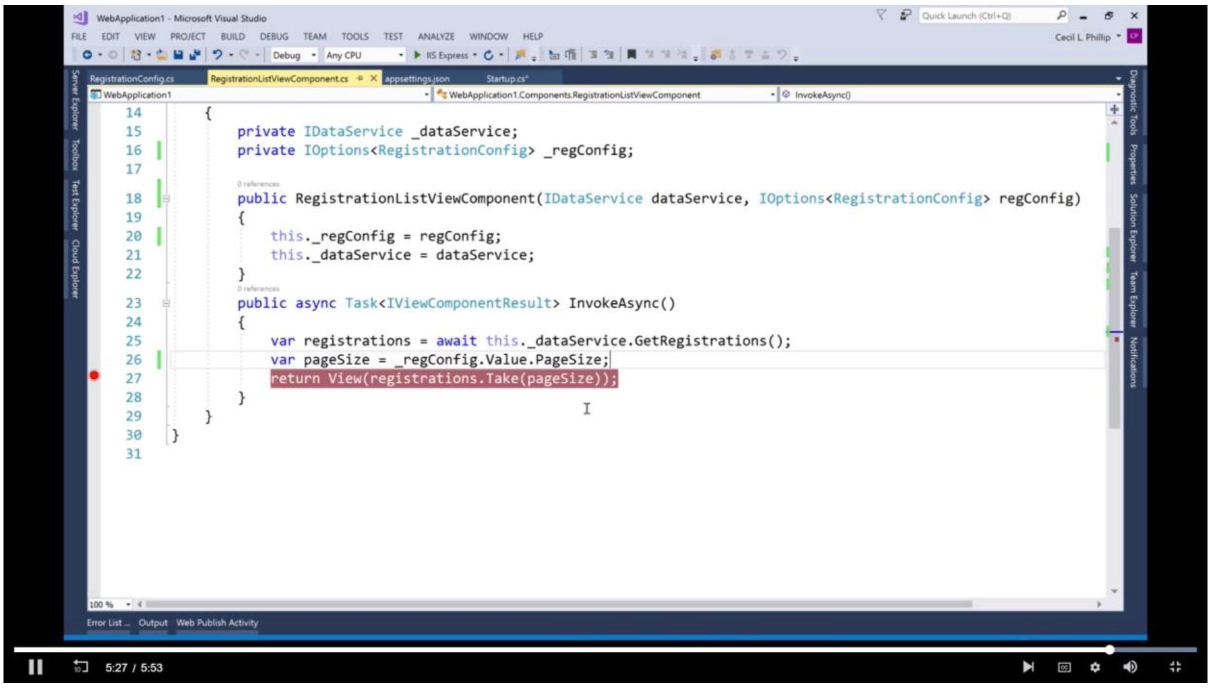
IConfigurationSection IConfiguration.GetSection(string key)
Gets a configuration sub-section with the specified key.
key: The key of the configuration section.


WebApplication1 - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ANALYZE WINDOW HELP
Debug
Any CPU
IIS Express
Quick Launch (Ctrl+Q)
Cecil L. Phillip
RegistrationConfig.cs
RegistrationListViewComponent.cs*
appsettings.json
Startup.cs*
WebApplication1
WebApplication1.Components.RegistrationListViewComponent
RegistrationListViewComponent(IDataService dataService, IOptions<Registratio
100 %
Error List ... Output Web Publish Activity
5:05 / 5:53

```
namespace WebApplication1.Components
{
    1 reference
    public class RegistrationListViewComponent : ViewComponent
    {
        private IDataService _dataService;
        private IConfigurationRoot _config;

        0 references
        public RegistrationListViewComponent(IDataService dataService, IOptions<RegistrationConfig> regConfig)
        {
            this._config = config;
            this._dataService = dataService;
        }
        0 references
        public async Task<IViewComponentResult> InvokeAsync()
        {
            var registrations = await this._dataService.GetRegistrations();
            var pageSize =  _config.GetValue<int>("RegistrationList:PageSize");
            return View(registrations.Take(pageSize));
        }
    }
}
```


WebApplication1 - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ANALYZE WINDOW HELP
Debug
Any CPU
IIS Express
Quick Launch (Ctrl+Q)
Cecil L. Phillip
RegistrationConfig.cs
RegistrationListViewComponent.cs
appsettings.json
Startup.cs*
WebApplication1
WebApplication1.Components.RegistrationListViewComponent
RegistrationListViewComponent(IDataService dataService, IOptions<Registratio
100 %
Error List ... Output Web Publish Activity
5:12 / 5:53

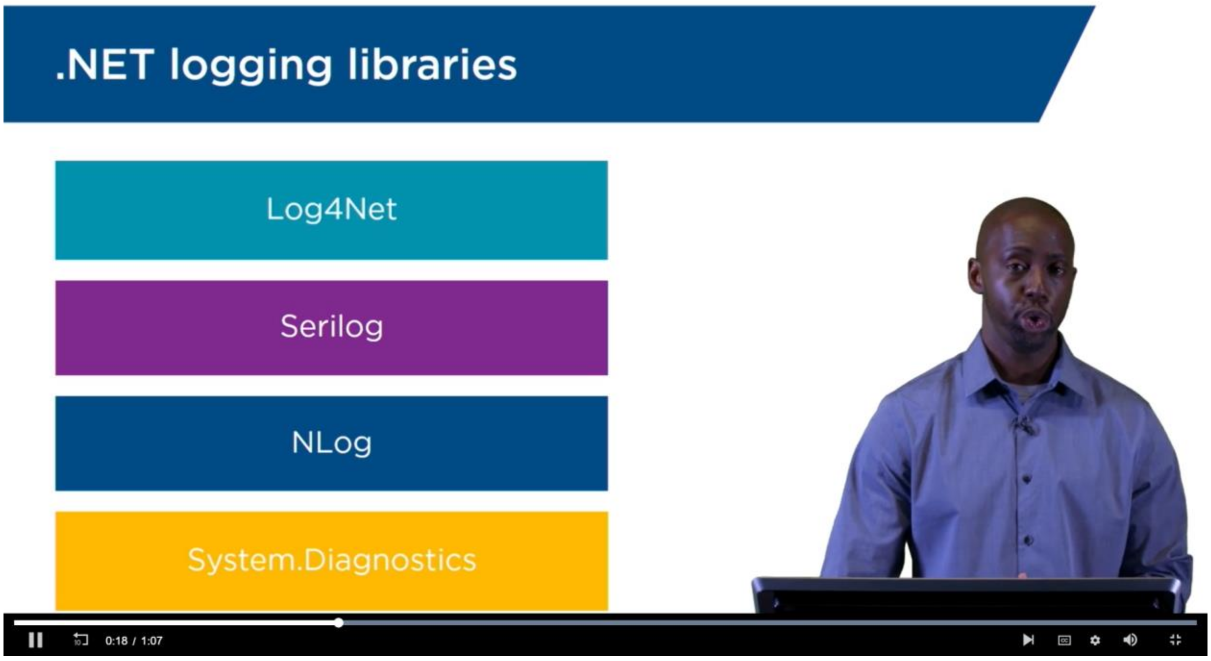
```
    {
        private IDataService _dataService;
        private IOptions<RegistrationConfig> _regConfig;

        0 references
        public RegistrationListViewComponent(IDataService dataService, IOptions<RegistrationConfig> regConfig)
        {
            this._regConfig = regConfig;
            this._dataService = dataService;
        }
        0 references
        public async Task<IViewComponentResult> InvokeAsync()
        {
            var registrations = await this._dataService.GetRegistrations();
            var pageSize =  _config.GetValue<int>("RegistrationList:PageSize");
            return View(registrations.Take(pageSize));
        }
    }
}
```


WebApplication1 - Microsoft Visual Studio
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ANALYZE WINDOW HELP
RegistrationConfig.cs
RegistrationListViewComponent.cs
appsettings.json
Startup.cs*
WebApplication1.Components.RegistrationListViewComponent
InvokeAsync()
5:27 / 5:53

```
    {
        private IDataService _dataService;
        private IOptions<RegistrationConfig> _regConfig;

        public RegistrationListViewComponent(IDataService dataService, IOptions<RegistrationConfig> regConfig)
        {
            this._regConfig = regConfig;
            this._dataService = dataService;
        }
        public async Task<IViewComponentResult> InvokeAsync()
        {
            var registrations = await this._dataService.GetRegistrations();
            var pageSize = _regConfig.Value.PageSize;
            return View(registrations.Take(pageSize));
        }
    }
}
```

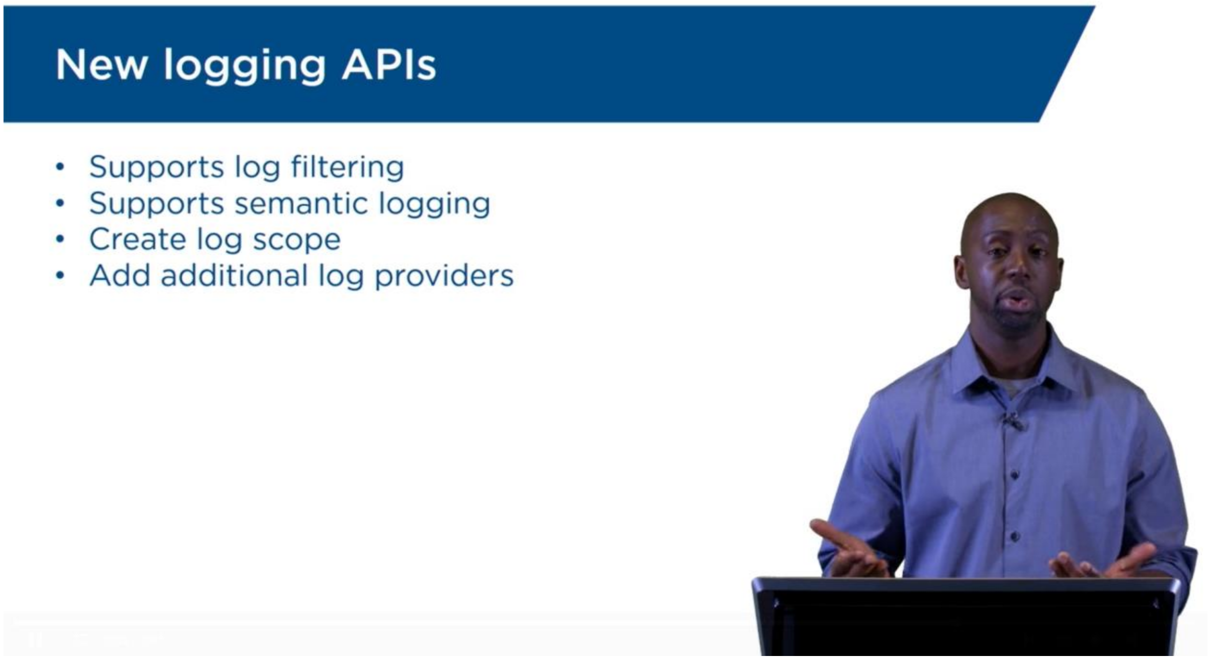
Logging support

Application logging is a fundamental component in understanding what happened in your software.



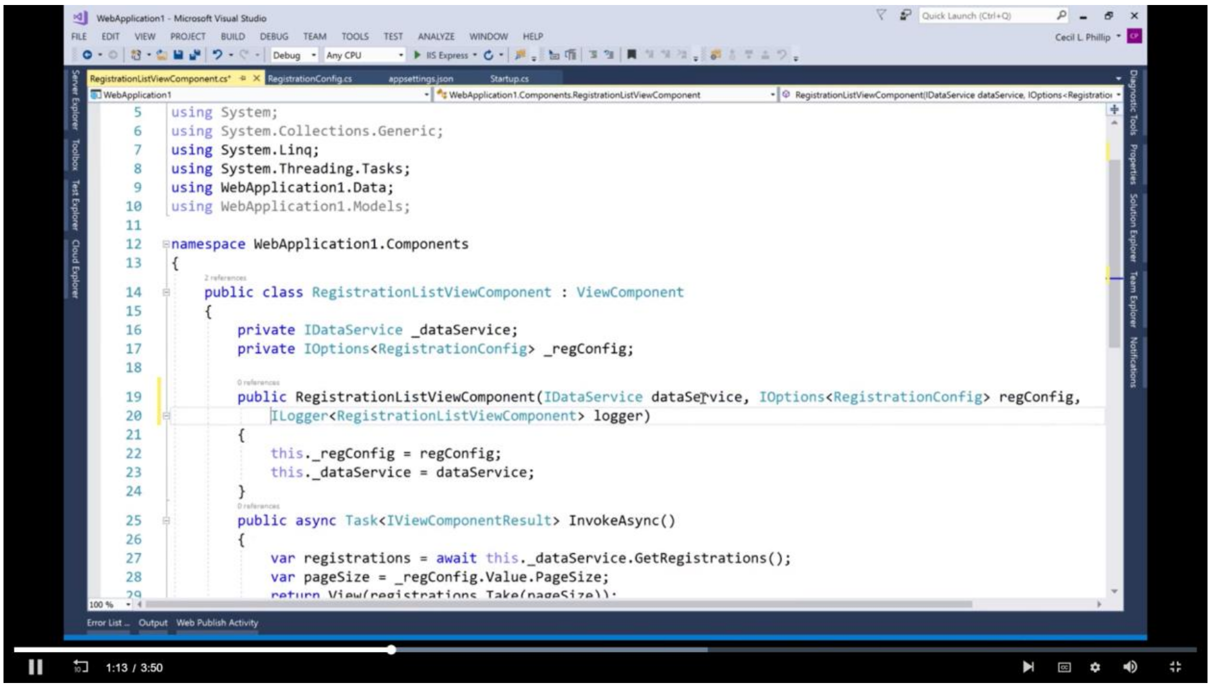
.NET logging libraries

- Log4Net
- Serilog
- NLog
- System.Diagnostics

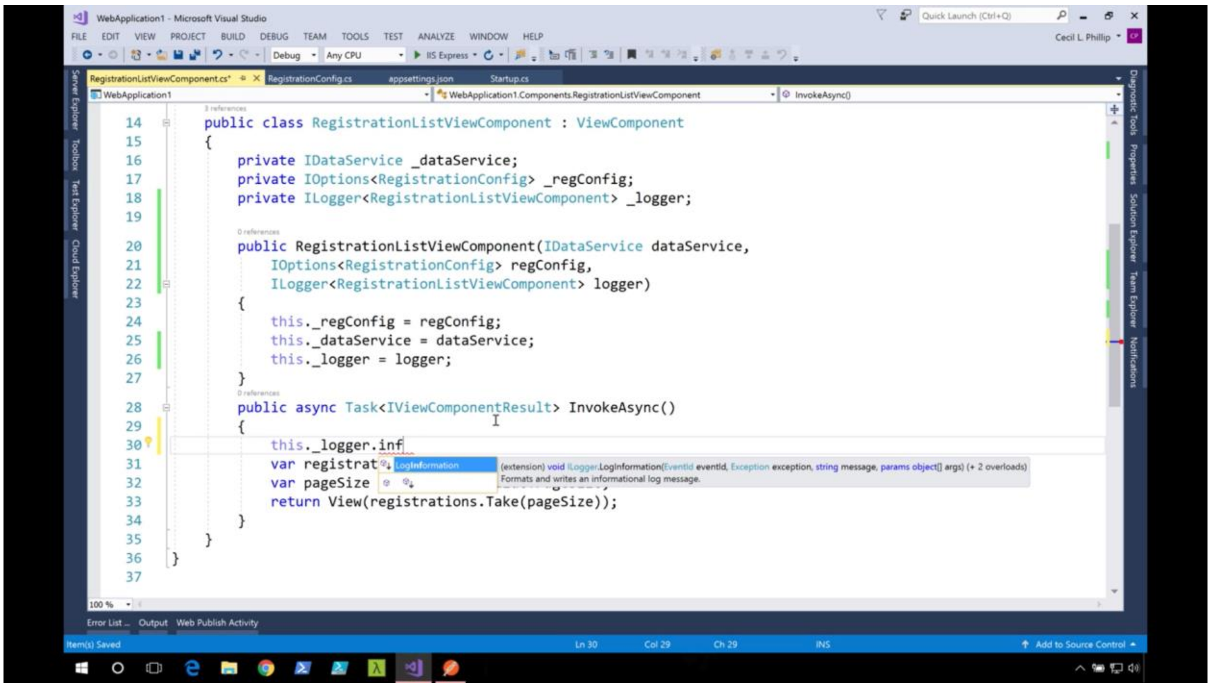


New logging APIs

- Supports log filtering
- Supports semantic logging
- Create log scope
- Add additional log providers



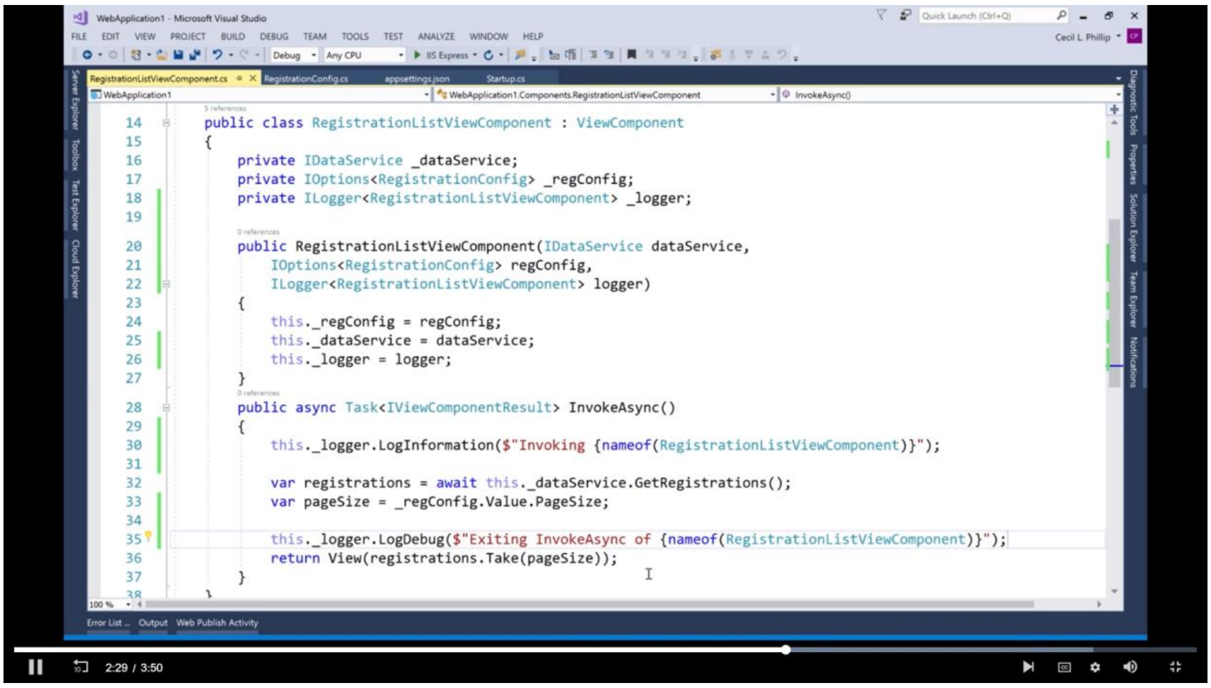
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using WebApplication1.Data;
using WebApplication1.Models;

namespace WebApplication1.Components
{
    2 references
    public class RegistrationListViewComponent : ViewComponent
    {
        private IDataService _dataService;
        private IOptions<RegistrationConfig> _regConfig;

        0 references
        public RegistrationListViewComponent(IDataService dataService, IOptions<RegistrationConfig> regConfig,
            ILogger<RegistrationListViewComponent> logger)
        {
            this._regConfig = regConfig;
            this._dataService = dataService;
        }
        0 references
        public async Task<IViewComponentResult> InvokeAsync()
        {
            var registrations = await this._dataService.GetRegistrations();
            var pageSize = _regConfig.Value.PageSize;
            return View(registrations.Take(pageSize));
```



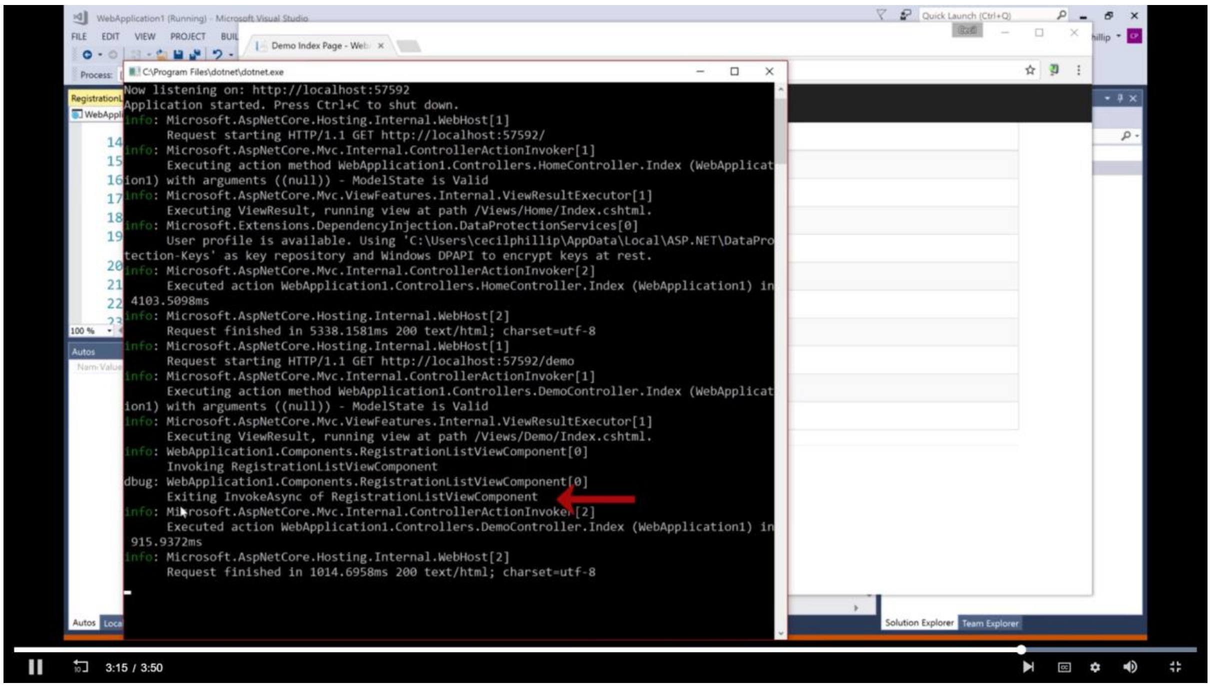
```
    3 references
    public class RegistrationListViewComponent : ViewComponent
    {
        private IDataService _dataService;
        private IOptions<RegistrationConfig> _regConfig;
        private ILogger<RegistrationListViewComponent> _logger;

        0 references
        public RegistrationListViewComponent(IDataService dataService,
            IOptions<RegistrationConfig> regConfig,
            ILogger<RegistrationListViewComponent> logger)
        {
            this._regConfig = regConfig;
            this._dataService = dataService;
            this._logger = logger;
        }
        0 references
        public async Task<IViewComponentResult> InvokeAsync()
        {
            this._logger.inf
            var registrat
            var pageSize
            return View(registrations.Take(pageSize));
        }
    }
}
```

LogInformation

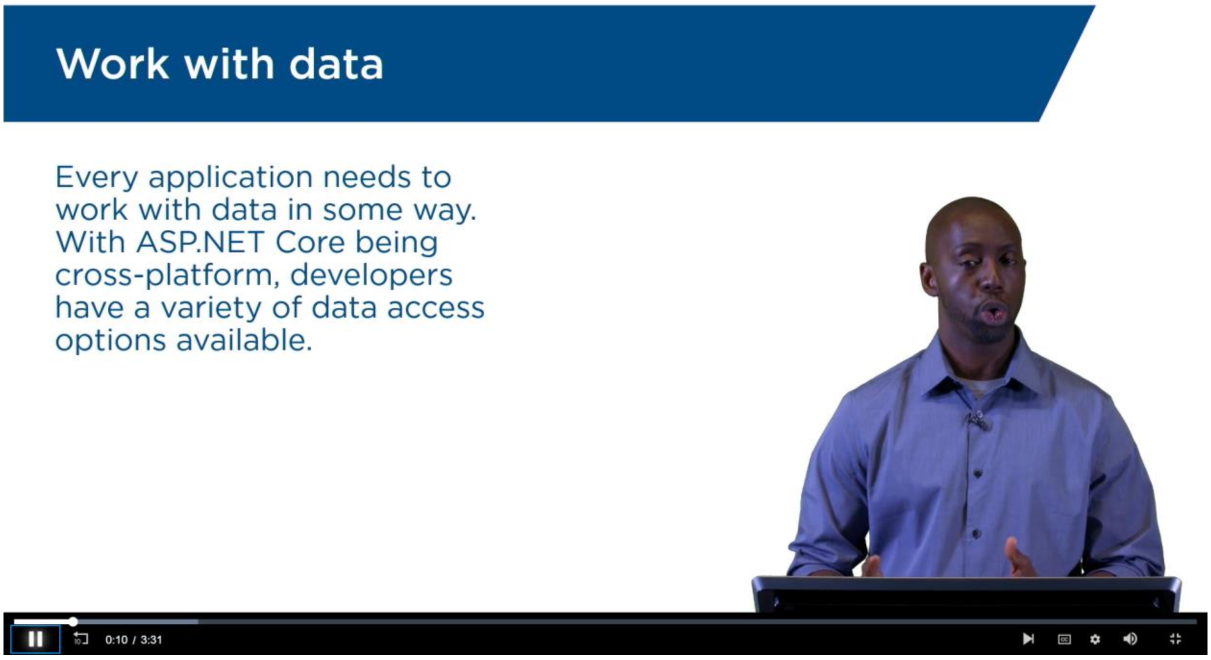
(extension) void ILogger.LogInformation(EventId eventId, Exception exception, string message, params object[] args) (+ 2 overloads)
Formats and writes an informational log message.


WebApplication1 - Microsoft Visual Studio
RegistrationListViewComponent.cs
RegistrationConfig.cs
appsettings.json
Startup.cs
WebApplication1.Components.RegistrationListViewComponent
InvokeAsync()
2:29 / 3:50

```csharp
public class RegistrationListViewComponent : ViewComponent
{
    private IDataService _dataService;
    private IOptions<RegistrationConfig> _regConfig;
    private ILogger<RegistrationListViewComponent> _logger;

    public RegistrationListViewComponent(IDataService dataService,
        IOptions<RegistrationConfig> regConfig,
        ILogger<RegistrationListViewComponent> logger)
    {
        this._regConfig = regConfig;
        this._dataService = dataService;
        this._logger = logger;
    }
    public async Task<IViewComponentResult> InvokeAsync()
    {
        this._logger.LogInformation($"Invoking {nameof(RegistrationListViewComponent)}");

        var registrations = await this._dataService.GetRegistrations();
        var pageSize = _regConfig.Value.PageSize;

        this._logger.LogDebug($"Exiting InvokeAsync of {nameof(RegistrationListViewComponent)}");
        return View(registrations.Take(pageSize));
    }
```

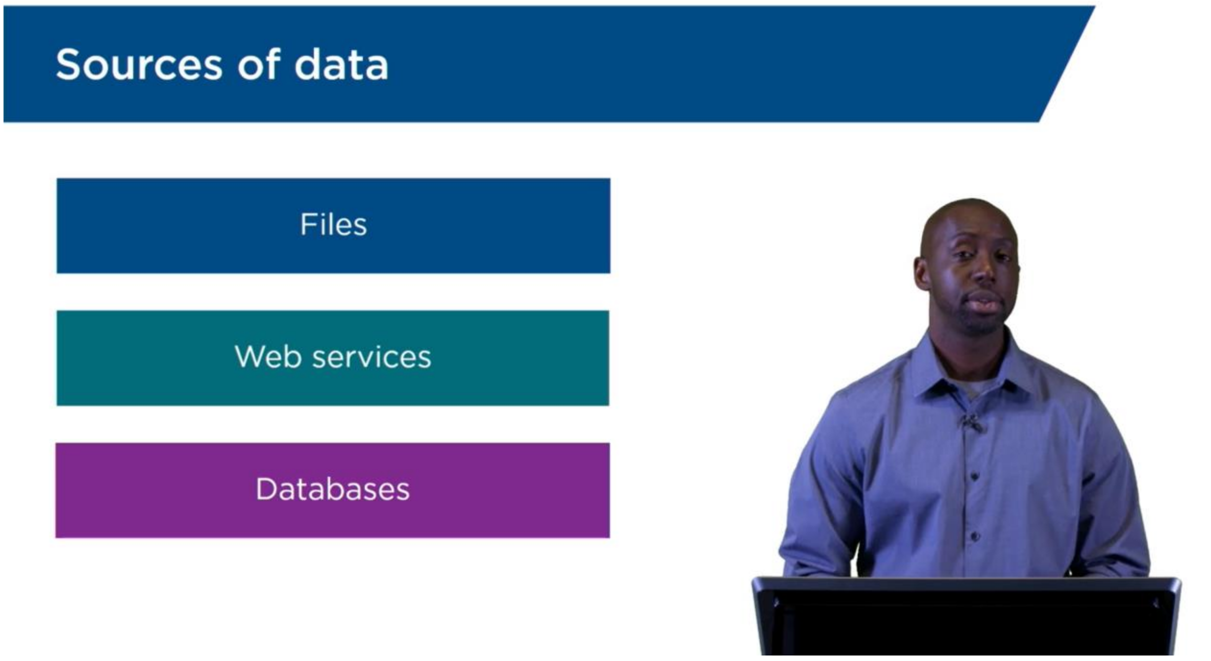

WebApplication1 (Running) - Microsoft Visual Studio
Demo Index Page - Web
C:\Program Files\dotnet\dotnet.exe
Solution Explorer
Team Explorer
3:15 / 3:50

```
Now listening on: http://localhost:57592
Application started. Press Ctrl+C to shut down.
info: Microsoft.AspNetCore.Hosting.Internal.WebHost[1]
      Request starting HTTP/1.1 GET http://localhost:57592/
info: Microsoft.AspNetCore.Mvc.Internal.ControllerActionInvoker[1]
      Executing action method WebApplication1.Controllers.HomeController.Index (WebApplicat
ion1) with arguments ((null)) - ModelState is Valid
info: Microsoft.AspNetCore.Mvc.ViewFeatures.Internal.ViewResultExecutor[1]
      Executing ViewResult, running view at path /Views/Home/Index.cshtml.
info: Microsoft.Extensions.DependencyInjection.DataProtectionServices[0]
      User profile is available. Using 'C:\Users\cecilphillip\AppData\Local\ASP.NET\DataPro
tection-Keys' as key repository and Windows DPAPI to encrypt keys at rest.
info: Microsoft.AspNetCore.Mvc.Internal.ControllerActionInvoker[2]
      Executed action WebApplication1.Controllers.HomeController.Index (WebApplication1) in
 4103.5098ms
info: Microsoft.AspNetCore.Hosting.Internal.WebHost[2]
      Request finished in 5338.1581ms 200 text/html; charset=utf-8
info: Microsoft.AspNetCore.Hosting.Internal.WebHost[1]
      Request starting HTTP/1.1 GET http://localhost:57592/demo
info: Microsoft.AspNetCore.Mvc.Internal.ControllerActionInvoker[1]
      Executing action method WebApplication1.Controllers.DemoController.Index (WebApplicat
ion1) with arguments ((null)) - ModelState is Valid
info: Microsoft.AspNetCore.Mvc.ViewFeatures.Internal.ViewResultExecutor[1]
      Executing ViewResult, running view at path /Views/Demo/Index.cshtml.
info: WebApplication1.Components.RegistrationListViewComponent[0]
      Invoking RegistrationListViewComponent
dbug: WebApplication1.Components.RegistrationListViewComponent[0]
      Exiting InvokeAsync of RegistrationListViewComponent
info: Microsoft.AspNetCore.Mvc.Internal.ControllerActionInvoker[2]
      Executed action WebApplication1.Controllers.DemoController.Index (WebApplication1) in
 915.9372ms
info: Microsoft.AspNetCore.Hosting.Internal.WebHost[2]
      Request finished in 1014.6958ms 200 text/html; charset=utf-8
```



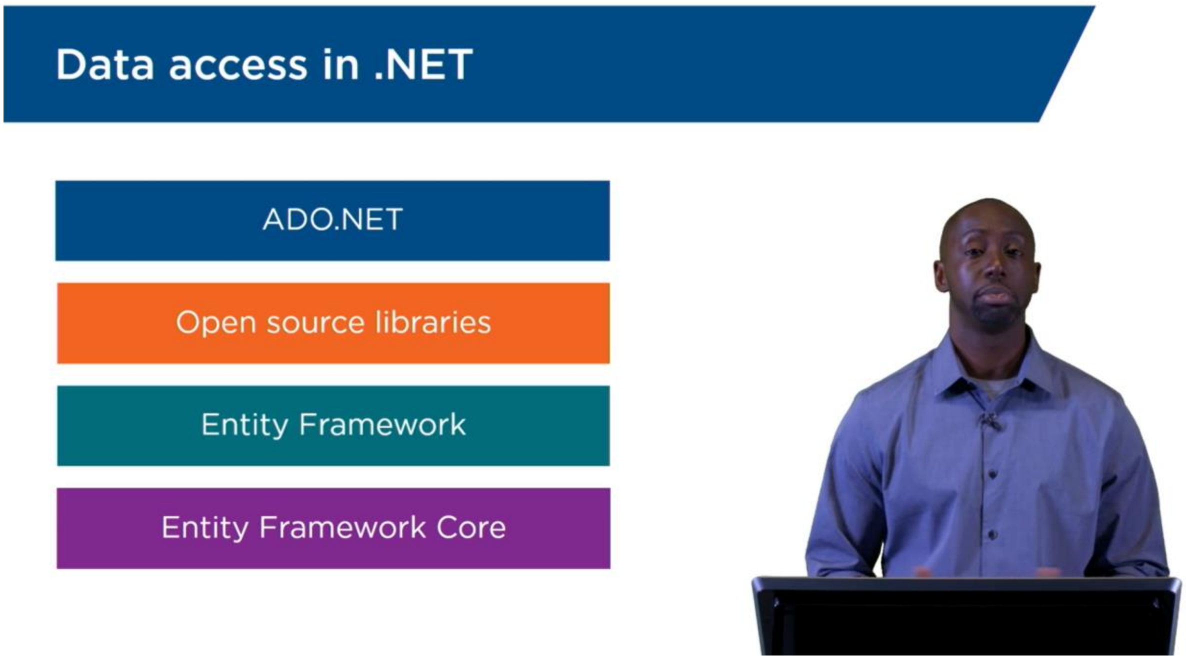
Work with data

Every application needs to work with data in some way. With ASP.NET Core being cross-platform, developers have a variety of data access options available.



Sources of data

- Files
- Web services
- Databases



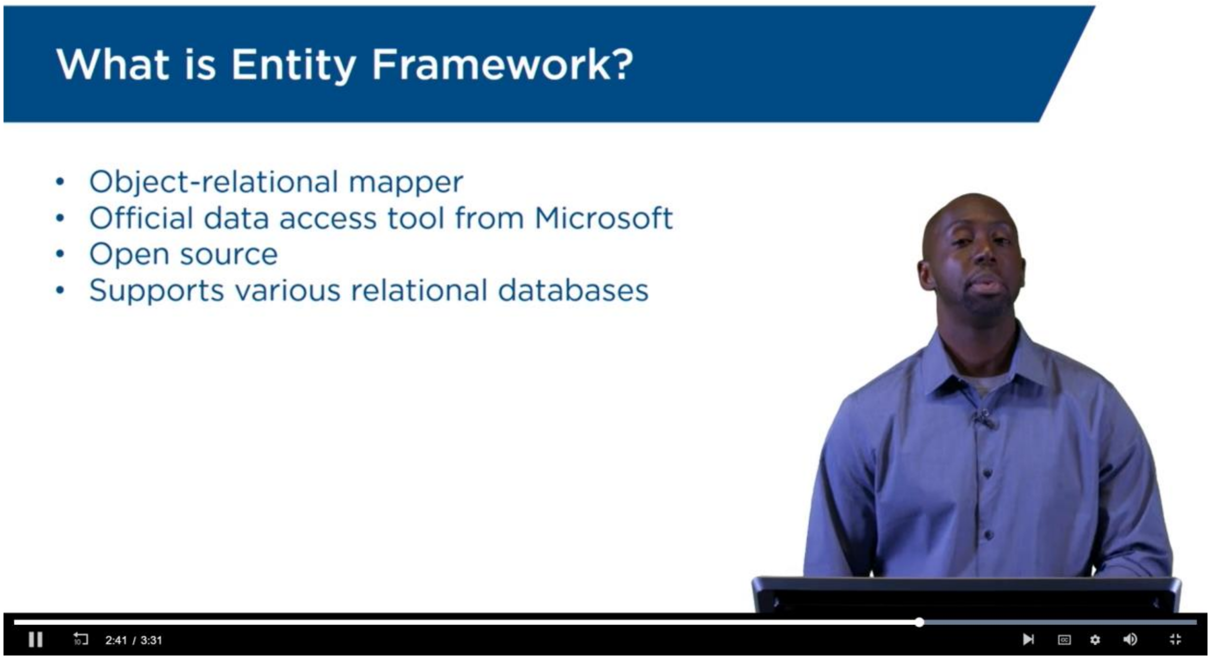
Data access in .NET

ADO.NET

Open source libraries

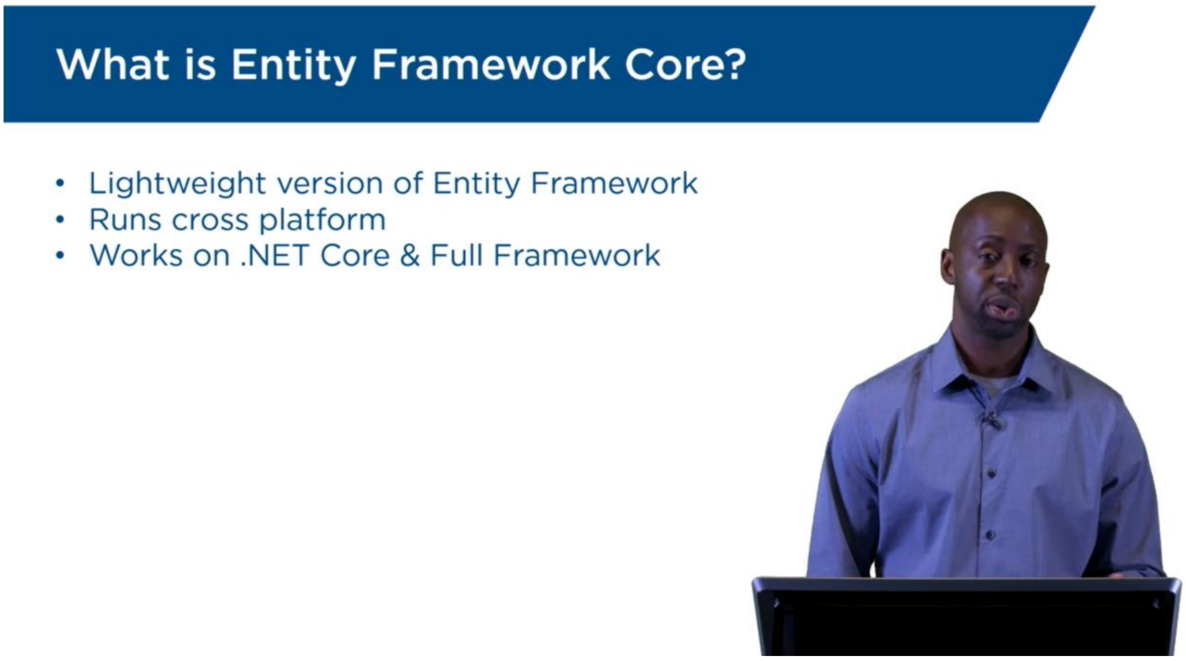
Entity Framework

Entity Framework Core



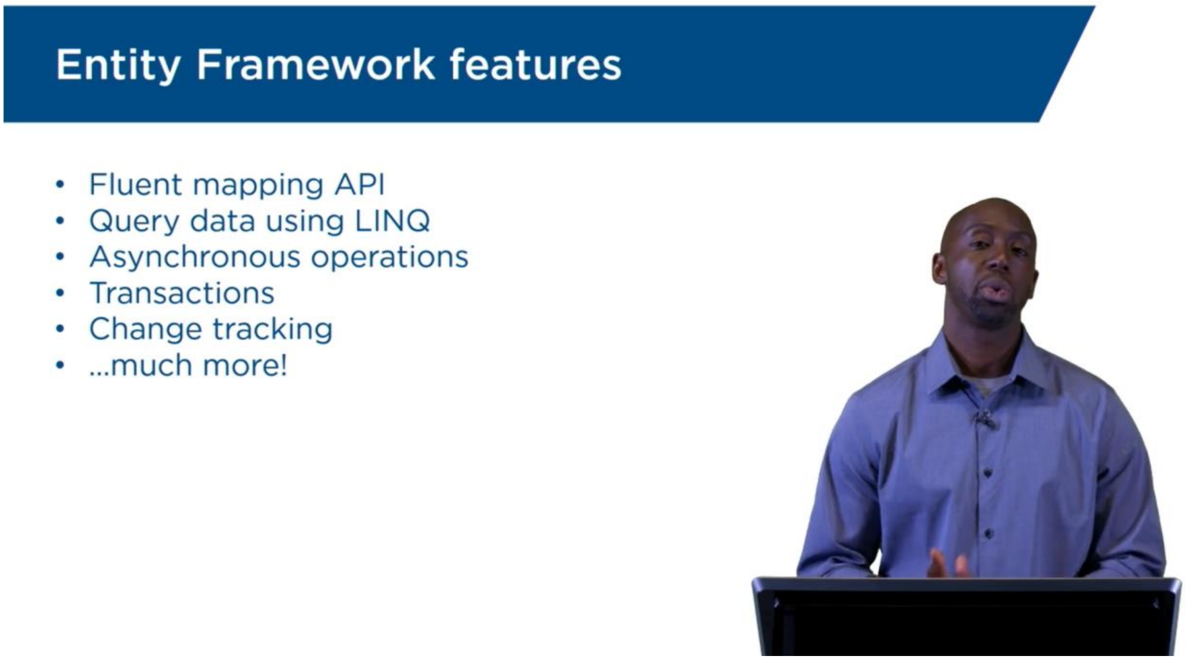
What is Entity Framework?

- Object-relational mapper
- Official data access tool from Microsoft
- Open source
- Supports various relational databases



What is Entity Framework Core?

- Lightweight version of Entity Framework
- Runs cross platform
- Works on .NET Core & Full Framework



Entity Framework features

- Fluent mapping API
- Query data using LINQ
- Asynchronous operations
- Transactions
- Change tracking
- ...much more!