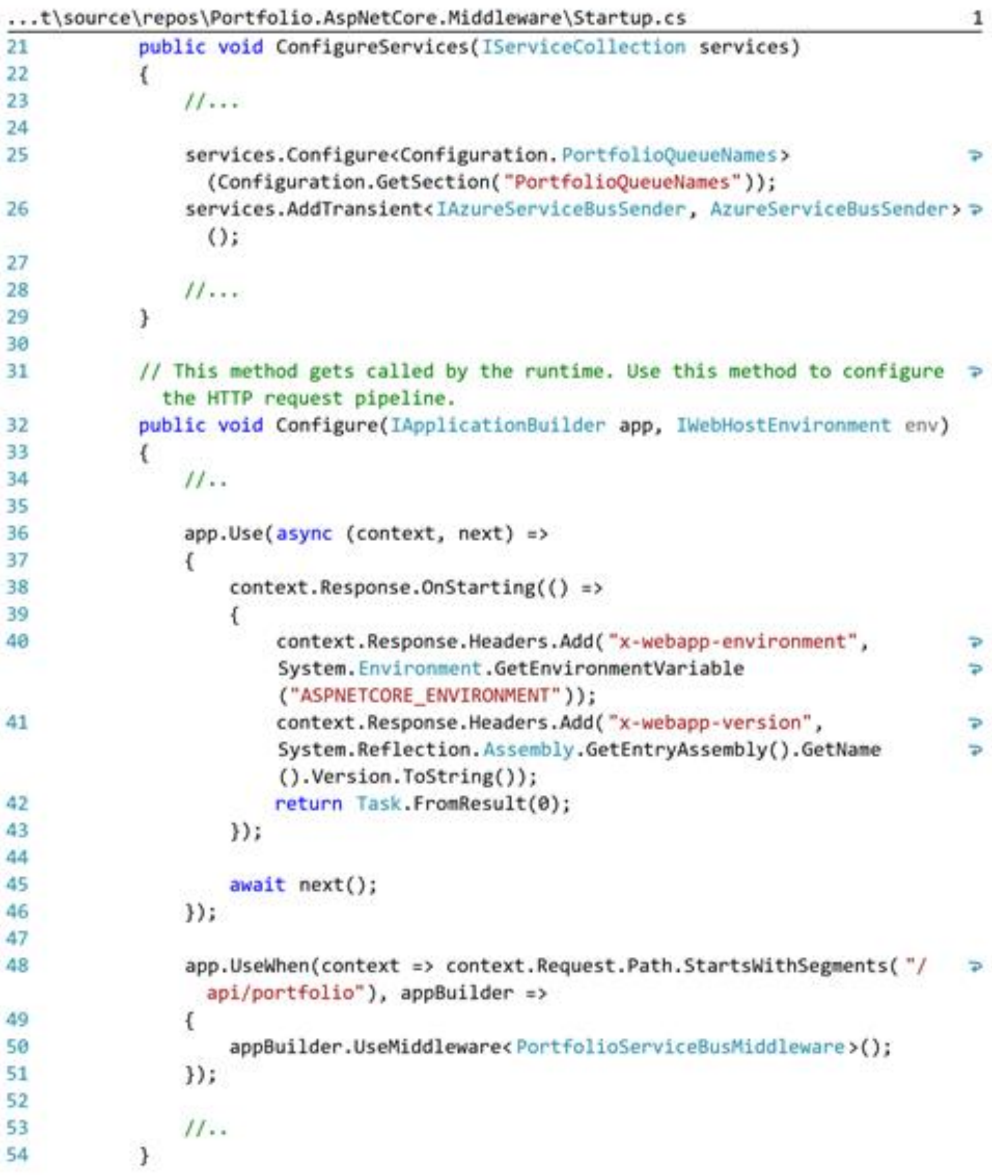
Example 1 - ASP.NET Core - Platform/Middleware

Summary

The following example leverages several ASP.NET Core capabilities including adding custom middleware to the request pipeline, conditional middleware for specific routes, built-in dependency injection, and Options pattern providing strongly-typed binding via configuration section.



```
...t\source\repos\Portfolio.AspNetCore.Middleware\Startup.cs                1
        public void ConfigureServices(IServiceCollection services)
        {
            //...

            services.Configure<Configuration.PortfolioQueueNames>
              (Configuration.GetSection("PortfolioQueueNames"));
            services.AddTransient<IAzureServiceBusSender, AzureServiceBusSender>
              ();

            //...
        }

        // This method gets called by the runtime. Use this method to configure
          the HTTP request pipeline.
        public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
        {
            //..

            app.Use(async (context, next) =>
            {
                context.Response.OnStarting(() =>
                {
                    context.Response.Headers.Add("x-webapp-environment",
                    System.Environment.GetEnvironmentVariable
                    ("ASPNETCORE_ENVIRONMENT"));
                    context.Response.Headers.Add("x-webapp-version",
                    System.Reflection.Assembly.GetEntryAssembly().GetName
                    ().Version.ToString());
                    return Task.FromResult(0);
                });

                await next();
            });

            app.UseWhen(context => context.Request.Path.StartsWithSegments("/
              api/portfolio"), appBuilder =>
            {
                appBuilder.UseMiddleware<PortfolioServiceBusMiddleware>();
            });

            //..
        }
```

....AspNetCore.Middleware\Services\IAzureServiceBusSender.cs 1

```
    public interface IAzureServiceBusSender
    {
        Task SendMessageAsync(string content, string queueName);
    }
```

...o.AspNetCore.Middleware\Services\AzureServiceBusSender.cs 1

```
    public class AzureServiceBusSender : IAzureServiceBusSender
    {
        private readonly string _connectionString;

        public AzureServiceBusSender(IConfiguration configuration)
        {
            _connectionString = configuration.GetConnectionString
              ("AzureServiceBus");
        }

        public async Task SendMessageAsync(string content, string queueName)
        {
            var queueClient = new QueueClient(_connectionString, queueName);

            Message message = new Message();
            message.Body = Encoding.UTF8.GetBytes(content);

            await queueClient.SendAsync(message);
        }
    }
```

```csharp
using Microsoft.AspNetCore.Http;
using Microsoft.Extensions.Options;
using Portfolio.AspNetCore.Middleware.Services;
using System;
using System.Buffers;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Portfolio.AspNetCore.Middleware
{
    public class PortfolioServiceBusMiddleware
    {
        private readonly RequestDelegate _next;

        public PortfolioServiceBusMiddleware(RequestDelegate next)
        {
            _next = next;
        }

        public async Task InvokeAsync(HttpContext httpContext, IAzureServiceBusSender sender, IOptions<Configuration.PortfolioQueueNames> portfolioConfig)
        {
            string targetQueueName = null;

            if (httpContext.Request.Method == System.Net.Http.HttpMethod.Post.Method)
            {
                targetQueueName = portfolioConfig.Value.PortfolioCreated;
            }
            else if (httpContext.Request.Method == System.Net.Http.HttpMethod.Put.Method)
            {
                targetQueueName = portfolioConfig.Value.PortfolioUpdated;
            }
            else if (httpContext.Request.Method == System.Net.Http.HttpMethod.Delete.Method)
            {
                targetQueueName = portfolioConfig.Value.PortfolioDeleted;
            }

            if (!String.IsNullOrWhiteSpace(targetQueueName))
            {
                httpContext.Request.EnableBuffering();
                var originalPosition = httpContext.Request.Body.Position;

                using (var reader = new StreamReader(
                    httpContext.Request.Body,
                    encoding: Encoding.UTF8,
                    detectEncodingFromByteOrderMarks: false,
                    bufferSize: 1024,
                    leaveOpen: true))
                {
                    var body = await reader.ReadToEndAsync();
                    await sender.SendMessageAsync(body, targetQueueName);
                    httpContext.Request.Body.Position = originalPosition;
                }
            }

            await _next(httpContext);
        }
    }

}

```

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

namespace Portfolio.AspNetCore.Middleware.Configuration
{
    public class PortfolioQueueNames
    {
        public string PortfolioCreated { get; set; }
        public string PortfolioUpdated { get; set; }
        public string PortfolioDeleted { get; set; }
    }
}

```