Send Email Code Refractor

Thursday, December 7, 2023 11:05 PM

service that requires an instance of SMTP setting class,
its' going to create a class on demand
and it's going to load data from the app settings file from the given particular section
and populate instance of this class with the data from this particular section.

```
builder.Services.AddSingleton<IEmailService, EmailService>();
```

1. builder.Services: builder typically refers to an instance of WebApplicationBuilder or HostBuilder in the context of ASP.NET C ore. The Services property provides access to the service container where you configure services for dependency injection.
2. AddSingleton: This method is used to register a service in the dependency injection container. In this case, it's registering a singleton service.
3. IEmailService: This is the service type or interface that you want to register. In other words, when a component requests an instance of IEmailService, the dependency injection container will provide the instance that was registered.
4. EmailService: This is the implementation type that will be used when an instance of IEmailService is requested. The container will create and manage a single instance of EmailService throughout the lifetime of the application because AddSingleton is u sed.

To put it simply, this line of code is telling the ASP.NET Core dependency injection system:

- "Whenever someone requests an instance of IEmailService, provide them with a single, shared instance of the EmailService clas s."

The use of AddSingleton is appropriate when you want a single instance of the service to be shared across the entire applicat ion. It can be beneficial for services that are stateless and can be reused without any issues. However, it's important to no te that singletons should be thread-safe because they are shared across multiple parts of the application.

Create SMPT app setting to store SMTP Server credentials

```
{
  "DetailedErrors": true,
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },

  "ConnectionStrings": {
    "Default": "Data Source=MTNEPAL-DEEPAK\\SQLEXPRESS03;Initial Catalog=AppUsers;Integrated Security=True;Encrypt=False;Trust Server Certificate=True"
  },
  "SMTP": {
    "host": "smtp-relay.brevo.com",
    "port": 587,
    "username": "shresthadeepak61@gmail.com",
    "password": "4BHXN2VMAPRO3DcT"
  }
}
```

Create "Services" folder and create "**EmailService.cs**" and extract Interface from it i.e. "**IEmailService.cs**"

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore.Storage.ValueConversion.Internal;
using System.Net.Mail;
using System.Net;
using Microsoft.Extensions.Options;
using WebApp.Settings;

namespace WebApp.Services
{
    public class EmailService : IEmailService
    {
        private readonly IOptions<SMTPSettings> _smtpOptions;

        public EmailService(IOptions<SMTPSettings> smtpOptions)
        {
            _smtpOptions = smtpOptions;
        }

        public async Task SendAsync(string from, string to, string subject, string body)
        {
            var message = new MailMessage(from, to, subject, body);

            using (var emailClient = new SmtpClient(_smtpOptions.Value.host, _smtpOptions.Value.port))
            {
                emailClient.Credentials = new NetworkCredential(_smtpOptions.Value.username, _smtpOptions.Value.password);

                await emailClient.SendMailAsync(message);

            }
        }
    }
}
```

IEmailService.cs

```
namespace WebApp.Services
{
    public interface IEmailService
    {
        Task SendAsync(string from, string to, string subject, string body);
    }
}
```

Create "Settings' folder and create "SMPTSettings.cs"

```
namespace WebApp.Settings
{
    public class SMTPSettings
    {
        public string host { get; set; } = string.Empty;
        public int port { get; set; }
        public string username { get; set; } = string.Empty;
        public string password { get; set; } = string.Empty;

    }
}
```

Inject the service for SMTP Settings and IEmailServices.cs

//service that requires an instance of SMTP setting class,
//its' going to create a class on demand
//and it's going to load data from the app settings file from the given particular section
//and populate instance of this class with the data from this particular section.

```
builder.Services.Configure<SMTPSettings>(builder.Configuration.GetSection("SMTP"));
builder.Services.AddSingleton<IEmailService, EmailService>();
```