MICROSOFT ARCHITECT
By Joydip Kanjilal, Columnist, InfoWorld
MAR 30, 2020 3:00 AM PDT

How to send emails in ASP.NET Core

Take advantage of the open source MailKit library to send emails in ASP.NET Core easily

You will often have the need to send emails through your application. You can take advantage of the MailKit NuGet package to send emails in ASP.NET Core. MailKit is an open source mail client library that can be used in .NET or .NET Core applications running on Windows, Linux, or Mac systems. This article presents a discussion of how we can use the MailKit NuGet package to send emails in ASP.NET Core.

To work with the code examples provided in this article, you should have Visual Studio 2019 installed in your system. If you don't already have a copy, you can download Visual Studio 2019 here.

[Also on InfoWorld: Developing for Windows 10X and the Surface Neo]

Create an ASP.NET Core API project

First off, let's create an ASP.NET Core project in Visual Studio. Assuming Visual Studio 2019 is installed in your system, follow the steps outlined below to create a new ASP.NET Core project in Visual Studio.

1. Launch the Visual Studio IDE.
2. Click on "Create new project."

3. In the “Create new project” window, select “ASP.NET Core Web Application” from the list of templates displayed.
4. Click Next.
5. In the “Configure your new project” window shown next, specify the name and location for the new project.
6. Click Create.
7. In the “Create New ASP.Net Core Web Application” window, select .NET Core as the runtime and ASP.NET Core 2.2 (or later) from the drop-down list at the top. I’ll be using ASP.NET Core 3.0 here.
8. Select “API” as the project template to create a new ASP.NET Core API application.
9. Ensure that the check boxes “Enable Docker Support” and “Configure for HTTPS” are unchecked as we won’t be using those features here.
10. Ensure that Authentication is set as “No Authentication” as we won’t be using authentication either.
11. Click Create.

This will create a new ASP.NET Core API project in Visual Studio. Select the Controllers solution folder in the Solution Explorer window and click “Add -> Controller...” to create a new controller named DefaultController. We’ll use this project in the subsequent sections of this article.

Install the MailKit NuGet package

To work with MailKit, you should install the MailKit package from NuGet. You can do this either via the NuGet package manager inside the Visual Studio 2019 IDE, or by executing the following command at the NuGet package manager console:

```
Install-Package NETCore.MailKit
```

You will also need to add references to the following namespaces in your code:

```
using MailKit.Net.Smtp;
using MimeKit;
```

Specify email configuration metadata in ASP.NET Core

The following code snippet shows how you can specify the email configuration details in the appsettings.json file.

```
"NotificationMetadata": {
    "Sender": "sender_email@gmail.com",
    "SmtpServer": "smtp.gmail.com",
    "Reciever": "receiver_email@yahoo.com",
    "Port": 465,
    "Username": "sender_email_user@gmail.com",
    "Password": "specify your password here"
  }
```

To read the email configuration data, we will take advantage of the following class.

```
public class NotificationMetadata
    {
        public string Sender { get; set; }
        public string Reciever { get; set; }
        public string SmtpServer { get; set; }
        public int Port { get; set; }
        public string UserName { get; set; }
        public string Password { get; set; }
    }
```

Here's how you can read the email configuration data from the appsettings.json file into an instance of NotificationMetadata class.

```
public void ConfigureServices(IServiceCollection services)
{
     var notificationMetadata =
     Configuration.GetSection("NotificationMetadata").
     Get<NotificationMetadata>();
     services.AddSingleton(notificationMetadata);
     services.AddControllers();
}
```

Create an instance of the EmailMessage class in ASP.NET Core

Create a new class named EmailMessage with the following code:

```
public class EmailMessage
    {
        public MailboxAddress Sender { get; set; }
        public MailboxAddress Reciever { get; set; }
        public string Subject { get; set; }
        public string Content { get; set; }
    }
```

Create an instance of the MimeMessage class in ASP.NET Core

The following method illustrates how you can create a MimeMessage instance from an instance of our custom class EmailMessage.

```
private MimeMessage CreateMimeMessageFromEmailMessage(EmailMessage message)
{
     var mimeMessage = new MimeMessage();
     mimeMessage.From.Add(message.Sender);
     mimeMessage.To.Add(message.Reciever);
     mimeMessage.Subject = message.Subject;
     mimeMessage.Body = new TextPart(MimeKit.Text.TextFormat.Text)
     { Text = message.Content };
     return mimeMessage;
}
```

Send emails synchronously using MailKit in ASP.NET Core

To send out an email, we need to take advantage of the SmtpClient class pertaining to the MailKit.Net.Smtp namespace. The following code snippet illustrates how this can be done.

```
using (SmtpClient smtpClient = new SmtpClient())
{
  smtpClient.Connect(_notificationMetadata.SmtpServer,
  _notificationMetadata.Port, true);
  smtpClient.Authenticate(_notificationMetadata.UserName,
  _notificationMetadata.Password);
  smtpClient.Send(mimeMessage);
  smtpClient.Disconnect(true);
}
```

Here is the complete code of the Get action method of our DefaultController class for your convenience.

```
public string Get()
{
EmailMessage message = new EmailMessage();
message.Sender = new MailboxAddress("Self", _notificationMetadata.Sender);
message.Reciever = new MailboxAddress("Self", _notificationMetadata.Reciever);
message.Subject = "Welcome";
message.Content = "Hello World!";
var mimeMessage = CreateEmailMessage(message);
using (SmtpClient smtpClient = new SmtpClient())
 {
    smtpClient.Connect(_notificationMetadata.SmtpServer,
    _notificationMetadata.Port, true);
    smtpClient.Authenticate(_notificationMetadata.UserName,
    _notificationMetadata.Password);
    smtpClient.Send(mimeMessage);
    smtpClient.Disconnect(true);
  }
 return "Email sent successfully";
}
```

[Also on InfoWorld: 27 essential tips for Git and GitHub users]

Send emails asynchronously using MailKit in ASP.NET Core

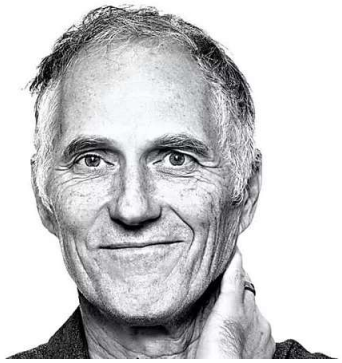
The following code snippet illustrates an asynchronous version of the code we just wrote to send emails synchronously.

```
using (SmtpClient smtpClient = new SmtpClient())
 {
      await smtpClient.ConnectAsync(_notificationMetadata.SmtpServer,
      _notificationMetadata.Port, true);
      await smtpClient.AuthenticateAsync(_notificationMetadata.UserName,
      _notificationMetadata.Password);
      await smtpClient.SendAsync(mimeMessage);
      await smtpClient.DisconnectAsync(true);
 }
```

Finally, note that MailKit also allows you to send emails using templates and even emails that have attachments. I will demonstrate the additional features of MailKit in a future article here.

Joydip Kanjilal is a Microsoft MVP in ASP.Net, as well as a speaker and author of several books and articles. He has more than 20 years of experience in IT including more than 16 years in Microsoft .Net and related technologies.

Follow



Tim O'Reilly: the golden age of the programmer is over

The decline of Heroku

Using Pulumi 3.0 to manage Azure infrastructure

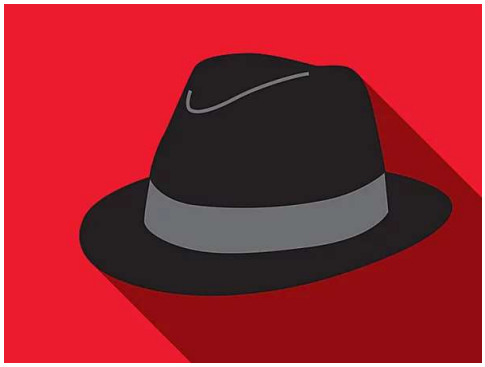
When to use Task.WaitAll vs. Task.WhenAll in .NET

6 ways Alibaba Cloud challenges AWS, Azure, and GCP

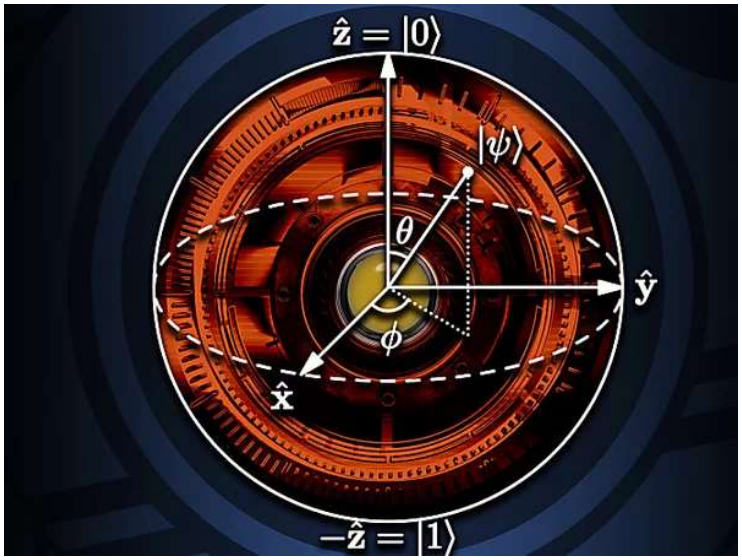
Google's Logica language addresses SQL's flaws

Edge computing archetypes are emerging,

3 enterprise AI success stories



Red Hat OpenShift ramps up security and



Amazon Braket: Get started with quantum computing

7 best practices for remote development teams