Adding Email to MVC

This document will guide you through setting up and email service in a MVC templated project utilizing your personal email service as the smtp.

First, create a model for the email information. Add a new class in the Models folder of your project. This will give us a basic structure for passing email information through our application.

```
public class EmailModel
{
    [Required, Display(Name = "Name")]
    public string FromName { get; set; }
    [Required, Display(Name = "Email"), EmailAddress]
    public string FromEmail { get; set; }
    [Required]
    public string Subject { get; set; }
    [Required]
    public string Body { get; set; }
}
```

Next, let's set up the configuration files to supplement our code. In this process we will add a new configuration file for our private information.

NOTE: It is VERY important that we protect this information. Because this will be a plain English file, we must ensure that only our code will access it and also that it will not be pushed to source control.

Find your personal email provider's smtp information. We will plug it into a custom configuration file that we will reference in the code. This will keep us from having to hard code any personal information in the application. Here are some starting points for getting your information.

Google: https://support.google.com/a/answer/176600?hl=en

Yahoo: https://help.yahoo.com/kb/SLN4724.html

Outlook: https://www.outlook-apps.com/outlook-com-pop-settings/

More…: https://www.arclab.com/en/kb/email/list-of-smtp-and-pop3-servers-mailserver-list.html

NOTE: It is a good idea to get a login key from your preferred email provider by registering your application. It will be much safer than using your email password and will allow you to revoke it if it becomes publicly known!

Add a new configuration file to the root of your project. Name it “private.config”. Delete the default code and set it up like this.

```
<appSettings>
  <add key="username" value="sample@email.com"/>
  <add key="password" value="emailpassword"/>
  <add key="emailto" value="default@email.com"/>
  <add key="host" value="SMTPSERVICE"/>
  <add key="port" value="SUGGESTED PORT"/>
</appSettings>
```

We will then reference this file in the Web.config. This way the private file will not have to be a part of the project when it is stored in the version control.

Find the `<appSettings>` tag in your Web.config. Change it to `<appSettings file="private.config">`. This will allow the private setting to be seen in the Web.config during the compile and thereby allow us to use these settings in our C# code.

Add the following class in your “Identity.config” file. You will find it in the App_Start folder of your project.

```
public class PersonalEmail
{
    public async Task SendAsync(MailMessage message)
    {
        var GmailUsername = WebConfigurationManager.AppSettings["username"];
        var GmailPassword = WebConfigurationManager.AppSettings["password"];
        var host = WebConfigurationManager.AppSettings["host"];
        int port = Convert.ToInt32(WebConfigurationManager.AppSettings["port"]);


        using (var smtp = new SmtpClient()
        {
            Host = host,
            Port = port,
            EnableSsl = true,
            DeliveryMethod = SmtpDeliveryMethod.Network,
            UseDefaultCredentials = false,
            Credentials = new NetworkCredential(GmailUsername, GmailPassword)
        })
        {
            try
            {
                await smtp.SendMailAsync(message);
            }
            catch (Exception e)
            {
                Console.WriteLine(e.Message);
                await Task.FromResult(0);

            }
        };
    }
}
```

This class is the email service that will handle the actual sending of the email information over the internet through a built in .Net class called "`SmtpClient()`".

Now we need to call it from our Controller. We will format a user's information and pass it to the service for sending to our email.

Let's add a new ActionResult in the HomeController like so…

```
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<ActionResult> Contact(EmailModel model)
{
    if (ModelState.IsValid)
    {
        try
        {
             var body = "<p>Email From: <bold>{0}</bold>
                         ({1})</p><p>Message:</p><p>{2}</p>";
            var from = "MyPortfolio<antonio@raynor.com>";
            model.Body = "This is a message from your portfolio site.  The name and
                          the email of the contacting person is above.";

            var email = new MailMessage(from,
                        ConfigurationManager.AppSettings["emailto"])
            {
                Subject = "Portfolio Contact Email",
                Body = string.Format(body, model.FromName, model.FromEmail,
                                     model.Body),
                IsBodyHtml = true
            };

            var svc = new PersonalEmail();
            await svc.SendAsync(email);

            return RedirectToAction("Sent");
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
            await Task.FromResult(0);
        }
    }
    return View(model);
}
```

We'll modify the Contact View to become a form for your users to submit their information. Add something like this in the "Contact.cshtml" file.

```
@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()
    <h4>Send me a message.</h4>
    <hr />
    @Html.ValidationSummary("", new { @class = "text-danger" })
    <div class="form-group">
        @Html.LabelFor(m => m.FromName, new { @class = "col-md-2 control-label" })
        <div class="col-md-10">
            @Html.TextBoxFor(m => m.FromName, new { @class = "form-control" })
            @Html.ValidationMessageFor(m => m.FromName)
        </div>
    </div>
    <div class="form-group">
        @Html.LabelFor(m => m.FromEmail, new { @class = "col-md-2 control-label" })
        <div class="col-md-10">
            @Html.TextBoxFor(m => m.FromEmail, new { @class = "form-control", placeholder =
"example@email.com" })
            @Html.ValidationMessageFor(m => m.FromEmail)
        </div>
    </div>
    <div class="form-group">
        <div class="col-md-offset-2 col-md-10">
            <input type="submit" class="btn btn-primary" value="Send" />
        </div>
    </div>
}
```