# Gửi Mail với ASP.Net Core

Nếu muốn gửi email từ localhost trong quá trình phát triển ứng dụng ASP.NET, có thể sử dụng một máy chủ SMTP cục bộ như **Papercut SMTP** hoặc **MailDev**, hoặc sử dụng một dịch vụ SMTP giả lập như **Mailtrap** để bắt và kiểm tra email mà không thực sự gửi chúng qua Internet.

**Papercut SMTP** là một công cụ đơn giản để bắt email gửi từ ứng dụng trong quá trình phát triển.

**Lưu ý:** khi sử dụng **Papercut SMTP để làm máy chủ email, các email được gửi sẽ không được gửi đến email, mà chúng sẽ được hiển thị trong Papercut SMTP.**

Cài đặt **Papercut.Smtp.Setup.exe**

* **Bước 1:** Tải và cài đặt [Papercut SMTP](https://github.com/ChangemakerStudios/Papercut-SMTP).
* **Bước 2:** Cấu hình ứng dụng của bạn để sử dụng Papercut SMTP. Mặc định, Papercut SMTP chạy trên localhost với cổng 25.

**Cấu hình trong appsettings.json:**

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*",

"Smtp": {

"Host": "localhost",

"Server": "localhost",

"FromAddress": "sender@gmail.com",

"Port": 25,

"EnableSsl": false,

"UserName": "",

"Password": ""

}

}

Vì Papercut SMTP không yêu cầu SSL hay xác thực, nên có thể để trống **UserName** và **Password**.

Tại Folder Helper, tạo class mới: MailService.cs

Helper/MailService.cs

using System.Net.Mail;

using System.Net;

namespace ShopManager.Helper

{

public class EmailService

{

private readonly IConfiguration \_configuration;

public EmailService(IConfiguration configuration)

{

\_configuration = configuration;

}

public void SendEmail(string toEmail, string subject, string body)

{

var smtpConfig = \_configuration.GetSection("Smtp");

var smtpClient = new SmtpClient(smtpConfig["Host"])

{

Port = int.Parse(smtpConfig["Port"]),

Credentials = new NetworkCredential(smtpConfig["UserName"], smtpConfig["Password"]),

EnableSsl = bool.Parse(smtpConfig["EnableSsl"])

};

var mailMessage = new MailMessage

{

From = new MailAddress(smtpConfig["UserName"]),

Subject = subject,

Body = body,

IsBodyHtml = true,

};

mailMessage.To.Add(toEmail);

smtpClient.Send(mailMessage);

}

}

}

Sử dụng

namespace ShopManager.Helper

{

public class HomeController : Controller

{

private readonly EmailService \_emailService;

public HomeController(EmailService emailService)

{

\_emailService = emailService;

}

public IActionResult SendEmail()

{

\_emailService.SendEmail("recipient@example.com", "Chủ đề email", "Nội dung email");

return View();

}

}

}

Program.cs

using Microsoft.AspNetCore.Authentication.Cookies;

using ShopManager.Helper;

var builder = WebApplication.CreateBuilder(args);

// Đăng ký các dịch vụ

builder.Services.AddTransient<EmailService>();

// Add services to the container.

builder.Services.AddControllersWithViews();

// Use Cookie

builder.Services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme)

.AddCookie(option =>

{

option.LoginPath = "/Customer/SignIn";

//truy cập trang có yêu cầu đăng nhập

//mà người dùng chưa đăng nhập

//thì sẽ tự động chuyển sang đường dẫn được khai báo trong LoginPath

option.AccessDeniedPath = "/AccessDenied";

//Khi người dùng truy cập trang

// mà người dùng đó không được quyền truy cập

// thì sẽ tự động chuyển sang đường dẫn được khai báo trong AccessDeniedPath

});

//Cho phép Field Null trong Form

builder.Services.AddControllersWithViews(options => options.SuppressImplicitRequiredAttributeForNonNullableReferenceTypes = true);

// Use Session

builder.Services.AddDistributedMemoryCache();

builder.Services.AddSession(options =>

{

options.IdleTimeout = TimeSpan.FromDays(10);

options.Cookie.HttpOnly = true;

options.Cookie.IsEssential = true;

});

var app = builder.Build();

// Configure the HTTP request pipeline.

if (!app.Environment.IsDevelopment())

{

app.UseExceptionHandler("/Home/Error");

// The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.

app.UseHsts();

}

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();

app.UseAuthentication();

app.UseAuthorization();

app.UseSession();

app.MapControllerRoute(

name: "admin",

pattern: "{area:exists}/{controller=ProductAdmin}/{action=Index}/{id?}");

app.MapControllerRoute(

name: "default",

pattern: "{controller=Home}/{action=Index}/{id?}");

app.Run();

# Gửi Email với template HTML

Cài đặt thư viện **RazorLight**

dotnet add package RazorLight

hoặc

Tool -> Nuget Package Manager -> Manage Nuget Package for Solution

tìm **RazorLight**  để cài đặt

Tạo một template HTML

wwwroot, tạo folder **Template**, tạo file **EmailTemplate.htm**

**wwwroot/Template/EmailTemplate.htm**

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

    <title></title>

</head>

<body>

    <img style="background:black;" src="https://www.aspsnippets.com/assets/img/logo\_ns.png" /><br /><br />

    <div style="border-top:3px solid #61028D">&nbsp;</div>

    <span style="font-family:Arial; font-size:10pt">

        Hello <b>{UserName}</b>,<br /><br />

        A new article has been published on ASPSnippets.<br /><br/>

        <a style="color:#61028D" href="{Url}">{Title}</a><br />

        {Description}

        <br /><br />

        Thanks<br />

        ASPSnippets

    </span>

</body>

</html>

Tại Folder Helper, tạo Class mới: **MailHelper.cs**

**Helper/MailHelper.cs**

using System.IO;

using System.Net;

using System.Net.Mail;

using RazorLight;

namespace ShopManager.Helper

{

public class MailHelper

{

private IWebHostEnvironment Environment { get; set; }

public IConfiguration Configuration { get; set; }

private readonly IConfiguration \_configuration;

private readonly IRazorLightEngine \_razorLightEngine;

public MailHelper(IConfiguration \_configuration, IWebHostEnvironment environment, IConfiguration configuration, IConfiguration iConfiguration)

{

Configuration = \_configuration;

Environment = environment;

// Tạo RazorLightEngine

\_razorLightEngine = new RazorLightEngineBuilder()

.UseFileSystemProject(Path.Combine(Directory.GetCurrentDirectory(), "Views"))

.UseMemoryCachingProvider()

.Build();

}

public string PopulateBody(string userName, string title, string url, string description)

{

//Lấy Template HTML

string body = string.Empty;

string path = Path.Combine(this.Environment.WebRootPath, "Template\\EmailTemplate.htm");

using (StreamReader reader = new StreamReader(path))

{

body = reader.ReadToEnd();

}

body = body.Replace("{UserName}", userName);

body = body.Replace("{Title}", title);

body = body.Replace("{Url}", url);

body = body.Replace("{Description}", description);

return body;

}

public void SendHtmlFormattedEmail(string recepientEmail, string subject, string body)

{

string host = this.Configuration.GetValue<string>("Smtp:Server");

int port = this.Configuration.GetValue<int>("Smtp:Port");

string fromAddress = this.Configuration.GetValue<string>("Smtp:FromAddress");

string userName = this.Configuration.GetValue<string>("Smtp:UserName");

string password = this.Configuration.GetValue<string>("Smtp:Password");

using (MailMessage mm = new MailMessage(fromAddress, recepientEmail))

{

mm.Subject = subject;

mm.Body = body;

mm.IsBodyHtml = true;

using (SmtpClient smtp = new SmtpClient())

{

smtp.Host = host;

smtp.EnableSsl = false;

NetworkCredential networkCred = new NetworkCredential(userName, password);

smtp.UseDefaultCredentials = true;

smtp.Credentials = networkCred;

smtp.Port = port;

smtp.Send(mm);

}

}

}

}

}

**Program.cs**

using Microsoft.AspNetCore.Authentication.Cookies;

using ShopManager.Helper;

var builder = WebApplication.CreateBuilder(args);

// Đăng ký các dịch vụ

builder.Services.AddTransient<MailHelper>();

// Add services to the container.

builder.Services.AddControllersWithViews();

// Use Cookie

builder.Services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme)

...

//Cho phép Field Null trong Form

builder.Services.AddControllersWithViews(options => options.SuppressImplicitRequiredAttributeForNonNullableReferenceTypes = true);

// Use Session

builder.Services.AddDistributedMemoryCache();

builder.Services.AddSession(options =>

{

options.IdleTimeout = TimeSpan.FromDays(10);

options.Cookie.HttpOnly = true;

options.Cookie.IsEssential = true;

});

var app = builder.Build();

// Configure the HTTP request pipeline.

if (!app.Environment.IsDevelopment())

{

app.UseExceptionHandler("/Home/Error");

// The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.

app.UseHsts();

}

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();

app.UseAuthentication();

app.UseAuthorization();

app.UseSession();

app.MapControllerRoute(

name: "admin",

pattern: "{area:exists}/{controller=ProductAdmin}/{action=Index}/{id?}");

app.MapControllerRoute(

name: "default",

pattern: "{controller=Home}/{action=Index}/{id?}");

app.Run();

**Sử dụng**

namespace ShopManager.Controllers

{

public class CustomerController : Controller

{

private readonly MailHelper \_mailHelper;

public CustomerController(MailHelper mailHelper)

{

\_mailHelper = mailHelper;

}

public IActionResult SendMail()

{

string emailReceived = "testmail@gmail.com";

string userName = "Duc Huy";

string title = "Tự học ASP.net";

string url = "https://www.google.com";

string description = "Hướng dẫn gửi email với Template HTML";

string body = \_mailHelper.PopulateBody(userName, title, url, description);

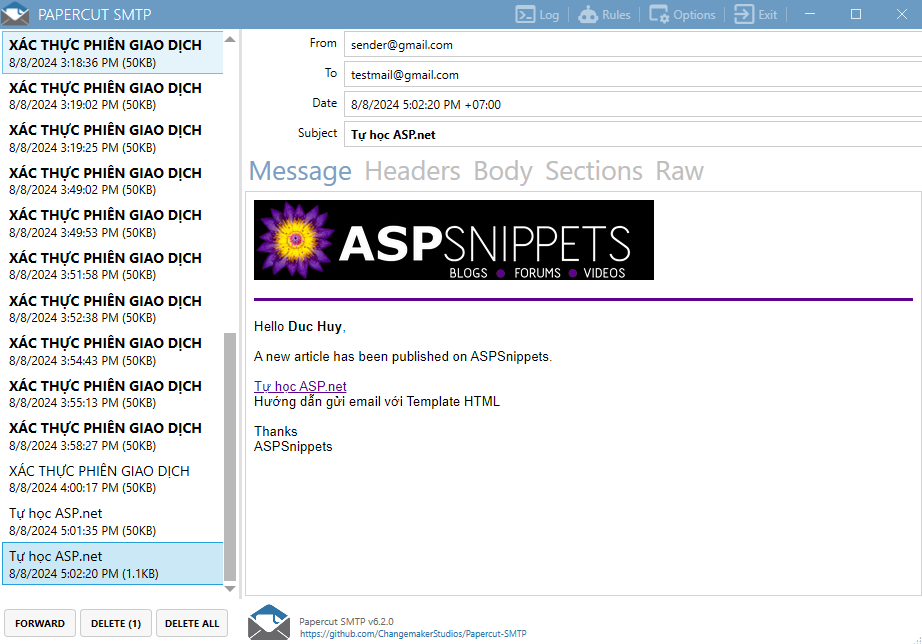
\_mailHelper.SendHtmlFormattedEmail(emailReceived, title, body);

return View();

}

}

}



**---THE END---**