عنوان: **معرفی کتابخانه Postal برای ASP.NET MVC**
نویسنده: آرمین ضیاء
تاریخ: ۱۳۹۲/۱۰/۲۰ ۰:۱۵
آدرس: www.dotnettips.info
گروه‌ها: Tips, Razor, ASP.Net MVC, MVC, Email, SMTP

Postal کتابخانه ای برای تولید و ارسال ایمیل توسط نماهای ASP.NET MVC است. برای شروع این کتابخانه را به پروژه خود اضافه کنید. پنجره **Package Manager Console** را باز کرده و فرمان زیر را اجرا کنید.

```
PM> Install-Package Postal
```

شروع به کار با Postal

نحوه استفاده از Postal در کنترلرهای خود را در کد زیر مشاهده می‌کنید.

```
using Postal;

public class HomeController : Controller
{
  public ActionResult Index()
  {
      dynamic email = new Email("Example");
      email.To = "webninja@example.com";
      email.FunnyLink = DB.GetRandomLolcatLink();
      email.Send();
      return View();
  }
}
```

Postal نمای ایمیل را در مسیر Views\Emails\Example.cshtml جستجو می‌کند.

```
To: @ViewBag.To
From: lolcats@website.com
Subject: Important Message

Hello,
You wanted important web links right?
Check out this: @ViewBag.FunnyLink

<3
```

پیکربندی SMTP

Postal ایمیل‌ها را توسط **SmtpClient** ارسال می‌کند که در فریم ورک دات نت موجود است. تنظیمات SMTP را می‌توانید در فایل web.config خود پیکربندی کنید. برای اطلاعات بیشتر به MSDN Documentation مراجعه کنید.

```
<configuration>
  ...
  <system.net>
    <mailSettings>
      <smtp deliveryMethod="network">
        <network host="example.org" port="25" defaultCredentials="true"/>
      </smtp>
    </mailSettings>
  </system.net>
  ...
</configuration>
```

ایمیل‌های Strongly-typed

همه خوششان نمی‌آید از آبجکت‌های دینامیک استفاده کنند. علاوه بر آن آبجکت‌های دینامیک مشکلاتی هم دارند. مثلا قابلیت

IntelliSense و یا Compile-time error را نخواهید داشت.
قدم اول - کلاسی تعریف کنید که از Email ارث بری می‌کند.

```
namespace App.Models
{
  public class ExampleEmail : Email
  {
    public string To { get; set; }
    public string Message { get; set; }
  }
}
```

قدم دوم - از این کلاس استفاده کنید!

```
public void Send()
{
  var email = new ExampleEmail
  {
    To = "hello@world.com",
    Message = "Strong typed message"
  };
  email.Send();
}
```

قدم سوم - نمایی ایجاد کنید که از مدل شما استفاده می‌کند. نام نما، بر اساس نام کلاس مدل انتخاب شده است. بنابراین مثلا *ExampleEmail* نمایی با نام *Example.cshtml* لازم دارد.

```
@model App.Models.ExampleEmail
To: @Model.To
From: postal@example.com
Subject: Example

Hello,
@Model.Message
Thanks!
```

آزمون‌های واحد (Unit Testing)

هنگام تست کردن کدهایی که با Postal کار می‌کنند، یکی از کارهایی که می‌خواهید انجام دهید حصول اطمینان از ارسال شدن ایمیل‌ها است. البته در بدنه تست‌ها نمی‌خواهیم هیچ ایمیلی ارسال شود.
Postal یک قرارداد بنام **IEmailService** و یک پیاده سازی پیش فرض از آن بنام **EmailService** ارائه می‌کند، که در واقع ایمیل‌ها را ارسال هم می‌کند. با در نظر گرفتن این پیش فرض که شما از یک IoC Container استفاده می‌کنید (مانند StructureMap, Ninject)، آن را طوری پیکربندی کنید تا یک نمونه از IEmailService به کنترلرها تزریق کند. سپس از این سرویس برای ارسال آبجکت‌های ایمیل‌ها استفاده کنید (بجای فراخوانی متد Email.Send()).

```
public class ExampleController : Controller
{
    public ExampleController(IEmailService emailService)
    {
        this.emailService = emailService;
    }

    readonly IEmailService emailService;

    public ActionResult Index()
    {
        dynamic email = new Email("Example");
        // ...
        emailService.Send(email);
        return View();
    }
}
```

این کنترلر را با ساختن یک Mock از اینترفیس IEmailService تست کنید. یک مثال با استفاده از FakeItEasy را در زیر مشاهده می‌کنید.

```
[Test]
public void ItSendsEmail()
{
    var emailService = A.Fake<IEmailService>();
    var controller = new ExampleController(emailService);
    controller.Index();
    A.CallTo(() => emailService.Send(A<Email>._))
     .MustHaveHappened();
}
```

ایمیل‌های ساده و HTML

Postal ارسال ایمیل‌های ساده (plain text) و HTML را بسیار ساده می‌کند.

قدم اول - نمای اصلی را بسازید. این نما headerها را خواهد داشت و نماهای مورد نیاز را هم رفرنس می‌کند. مسیر نما ~\Views\Emails\Example.cshtml است.

```
To: test@test.com
From: example@test.com
Subject: Fancy email
Views: Text, Html
```

قدوم دوم - نمای تکست را ایجاد کنید. به قوانین نامگذاری دقت کنید، Example.cshtml به Example.Text.cshtml تغییر یافته. مسیر فایل Views\Emails\Example.Text.cshtml است.

```
Content-Type: text/plain; charset=utf-8

Hello @ViewBag.PersonName,
This is a message
```

دقت داشته باشید که تنها یک Content-Type باید تعریف کنید.

قدم سوم - نمای HTML را ایجاد کنید (باز هم فقط با یک Content-Type). مسیر فایل ~\Views\Emails\Example.Html.cshtml است.

```
Content-Type: text/html; charset=utf-8

<html>
  <body>
    <p>Hello @ViewBag.PersonName,</p>
    <p>This is a message</p>
  </body>
</html>
```

ضمیمه ها

برای افزودن ضمائم خود به ایمیل ها، متد Attach را فراخوانی کنید.

```
dynamic email = new Email("Example");
email.Attach(new Attachment("c:\\attachment.txt"));
email.Send();
```

جاسازی تصاویر در ایمیل ها

Postal یک HTML Helper دارد که امکان جاسازی (embedding) تصاویر در ایمیل‌ها را فراهم می‌کند. دیگر نیازی نیست به یک URL خارجی اشاره کنید.

ابتدا مطمئن شوید که فایل web.config شما فضای نام Postal را اضافه کرده است. این کار دسترسی به HTML Helper مذکور در نمای‌های ایمیل را ممکن می‌سازد.

```
<configuration>
  <system.web.webPages.razor>
    <pages pageBaseType="System.Web.Mvc.WebViewPage">
      <namespaces>
        <add namespace="Postal" />
      </namespaces>
    </pages>
  </system.web.webPages.razor>
</configuration>
```

متد **EmbedImage** تصویر مورد نظر را در ایمیل شما جاسازی می‌کند و توسط یک تگ <img/> آن را رفرنس می‌کند.

```
To: john@example.org
From: app@example.org
Subject: Image

@Html.EmbedImage("~/content/postal.jpg")
```

Postal سعی می‌کند تا نام فایل تصویر را، بر اساس مسیر تقریبی ریشه اپلیکیشن شما تعیین کند.

Postal بیرون از ASP.NET

Postal می‌تواند نماهای ایمیل‌ها را بیرون از فضای ASP.NET رندر کند. مثلا در یک اپلیکیشن کنسول یا یک سرویس ویندوز.
این امر توسط یک View Engine سفارشی میسر می‌شود. تنها نماهای Razor پشتیبانی می‌شوند. نمونه کدی را در زیر مشاهده می‌کنید.

```
using Postal;

class Program
{
    static void Main(string[] args)
    {
        // Get the path to the directory containing views
        var viewsPath = Path.GetFullPath(@"..\..\Views");

        var engines = new ViewEngineCollection();
        engines.Add(new FileSystemRazorViewEngine(viewsPath));

        var service = new EmailService(engines);

        dynamic email = new Email("Test");
        // Will look for Test.cshtml or Test.vbhtml in Views directory.
        email.Message = "Hello, non-asp.net world!";
        service.Send(email);
    }
}
```

محدودیت ها: نمی توانید برای نمای ایمیل هایتان از Layoutها استفاده کنید. همچنین در نماهای خود تنها از مدل‌ها (Models) می‌توانید استفاده کنید، و نه ViewBag.

Email Headers: برای در بر داشتن نام، در آدرس ایمیل از فرمت زیر استفاده کنید.

```
To: John Smith <john@example.org>
```

Multiple Values: برخی از headerها می‌توانند چند مقدار داشته باشند. مثلا Bcc و CC. اینگونه مقادیر را می‌توانید به دو روش در نمای خود تعریف کنید:
جدا کردن مقادیر با کاما:

```
Bcc: john@smith.com, harry@green.com
Subject: Example
```

```
etc
```

و یا تکرار header:

```
Bcc: john@smith.com
Bcc: harry@green.com
Subject: Example

etc
```

ساختن ایمیل بدون ارسال آن

لازم نیست برای ارسال ایمیل هایتان به Postal تکیه کنید. در عوض می‌توانید یک آبجکت از نوع System.Net.Mail.MailMessage تولید کنید و به هر نحوی که می‌خواهید آن را پردازش کنید. مثلا شاید بخواهید بجای ارسال ایمیل ها، آنها را به یک صف پیام مثل MSMQ انتقال دهید یا بعدا توسط سرویس دیگری ارسال شوند. این آبجکت MailMessage تمامی Header ها، محتوای اصلی ایمیل و ضمائم را در بر خواهد گرفت.

کلاس EmailService در Postal متدی با نام CreateMailMessage فراهم می‌کند.

```
public class ExampleController : Controller
{
    public ExampleController(IEmailService emailService)
    {
        this.emailService = emailService;
    }

    readonly IEmailService emailService;

    public ActionResult Index()
    {
        dynamic email = new Email("Example");
        // ...

        var message = emailService.CreateMailMessage(email);
        CustomProcessMailMessage(message);

        return View();
    }
}
```

در این پست با امکانات اصلی کتابخانه Postal آشنا شدید و دیدید که به سادگی می‌توانید ایمیل‌های Razor بسازید. برای اطلاعات بیشتر لطفا به سایت پروژه Postal مراجعه کنید.

نظرات خوانندگان

نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۴/۰۲/۲۴ ۱۲:۴۹

صفحه مستندات این کتابخانه

دستورات جدید نیوگت به شرح زیر هست:

```
Install-Package Postal.Mvc4
```

```
PM> Install-Package Postal.Mvc5
```

نمونه‌های مشابه دیگر