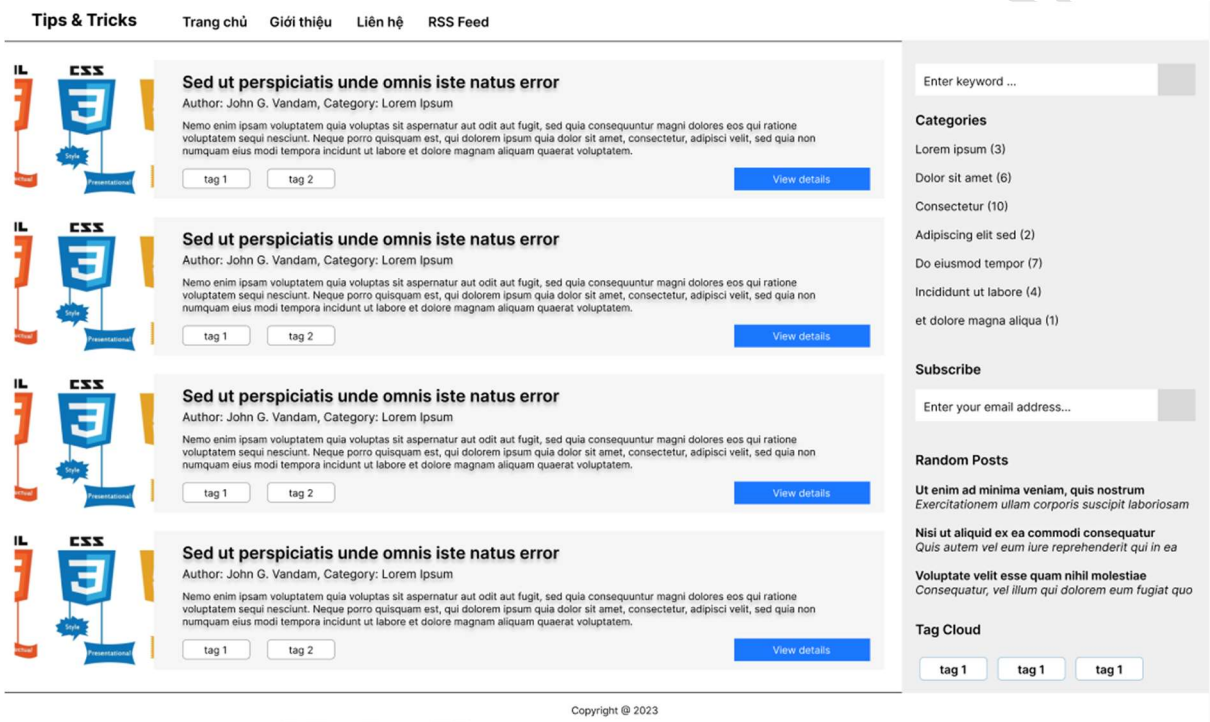
LAB 2. TAT-BLOG: CHỨC NĂNG CƠ BẢN CHO ĐỘC GIẢ

Thời lượng: 4 tiết

A. Mục tiêu

Bài thực hành này yêu cầu sinh viên xây dựng từ đầu đến cuối các chức năng cơ bản dành cho độc giả của một trang blog đơn giản. Giao diện của ứng dụng được phác thảo như hình sau.



Sau khi hoàn thành bài thực hành này, sinh viên cần nắm vững:

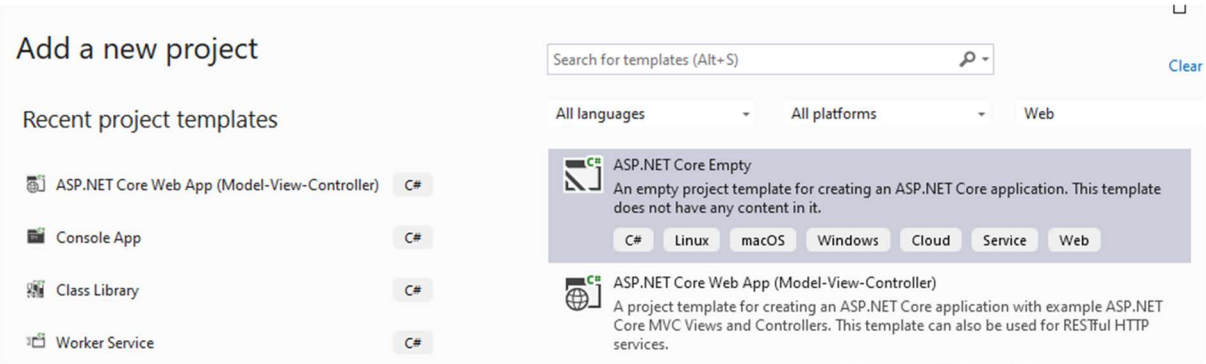
- Cách tổ chức mã nguồn một dự án web được xây dựng theo MVC Framework.
- Cách cài đặt các gói thư viện dùng cho phía client (client-side libraries).
- Cách tạo view, partial view, xây dựng layout chung cho ứng dụng.
- Cách tạo controller, action và truyền dữ liệu từ controller sang view.
- Cách tạo và sử dụng view components.
- Cách render dữ liệu trong view, partial view, view component.
- Cách đăng ký các dịch vụ cần thiết với DI Container.
- Cách định nghĩa các endpoint, route template và friendly URL.
- Kết hợp việc đọc dữ liệu từ CSDL và tạo kết quả trả về cho người dùng.

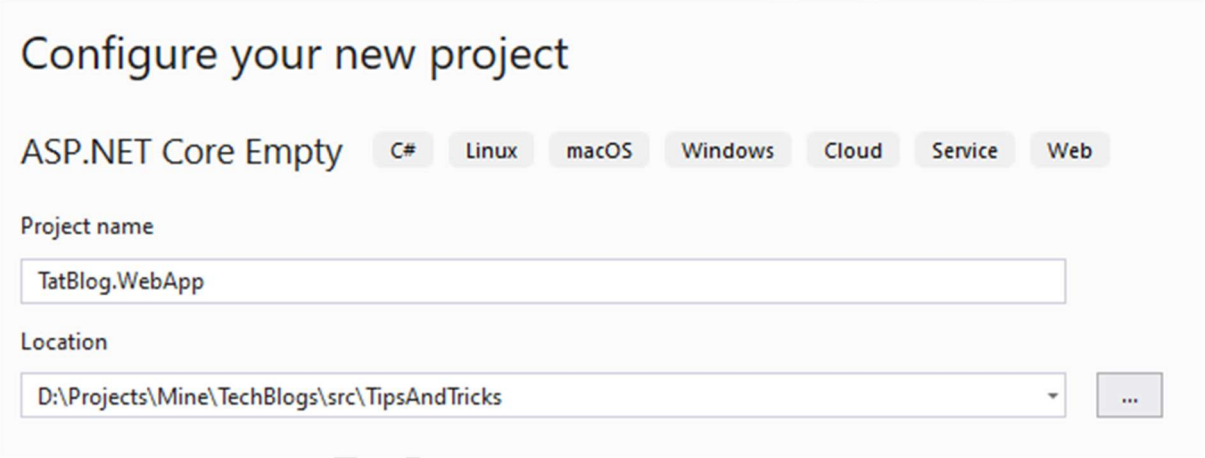
Yêu cầu: Sinh viên tự làm phần "B. Hướng dẫn thực hành" ở nhà và nộp lên hệ thống LMS. Tại phòng Lab, sinh viên làm phần "C. Bài tập thực hành" dựa trên dự án đã hoàn thành ở phần B.

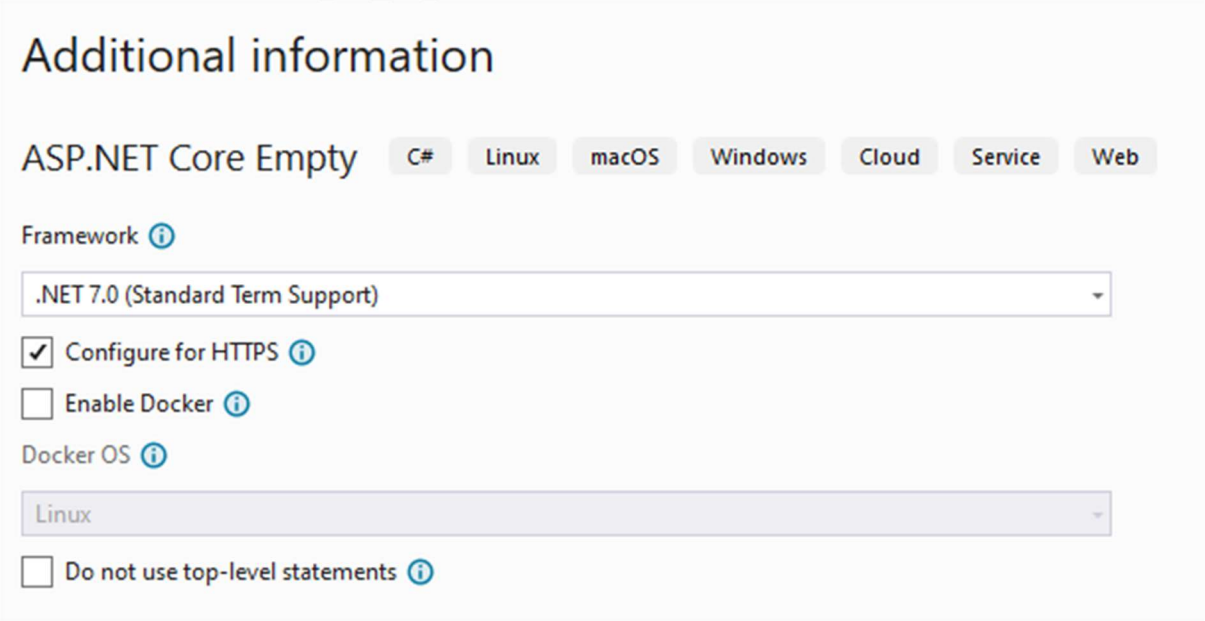
B. Hướng dẫn thực hành

1. Khởi tạo dự án Web

Trong VS 2022 (sinh viên có thể sử dụng VS Code), tạo một dự án mới theo template **ASP.NET Core Empty**, đặt tên là **TatBlog.WebApp**.



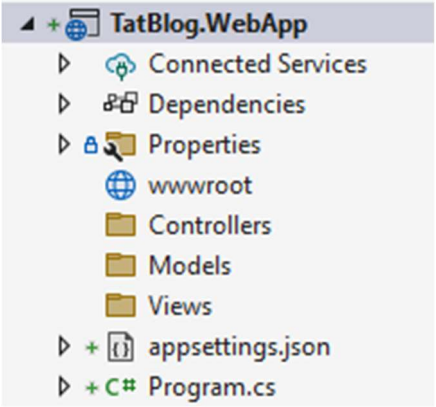


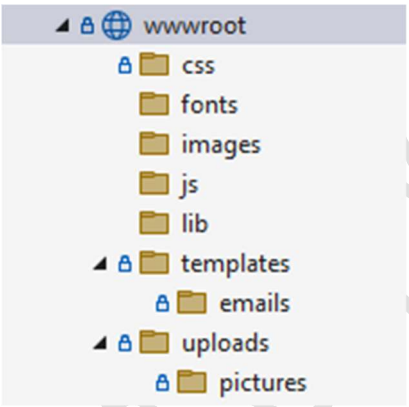


Nhấp phải chuột vào dự án vừa tạo, chọn **Set as startup project**.

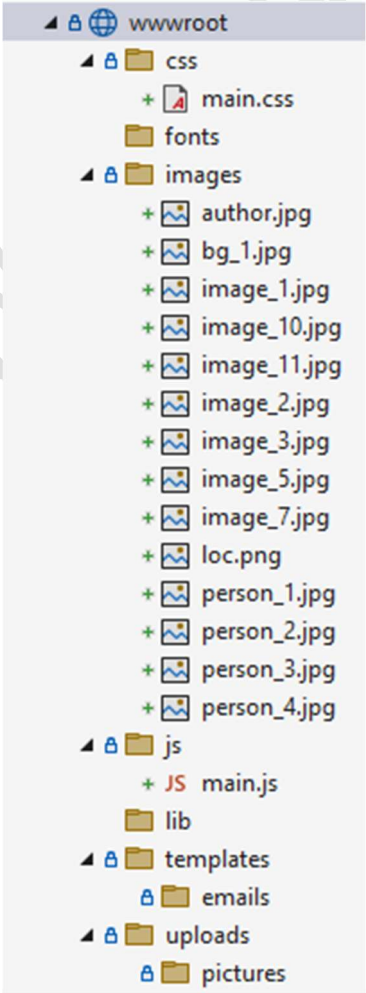
Sử dụng chức năng **Add Reference** để **TatBlog.WebApp** tham chiếu đến 3 dự án **TatBlog.Core, TatBlog.Data, TatBlog.Services**.

Trong dự án **TatBlog.WebApp,** tạo các thêm các thư mục **wwwroot, Controllers, Models, Views**. Trong thư mục **wwwroot**, tạo các thư mục như hình sau:



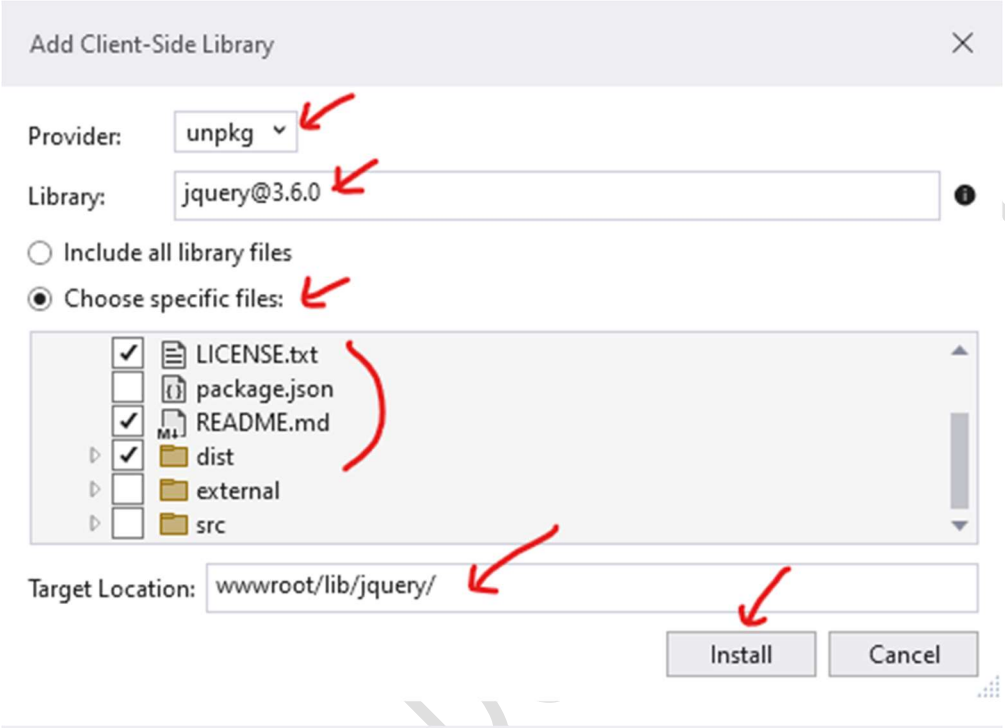


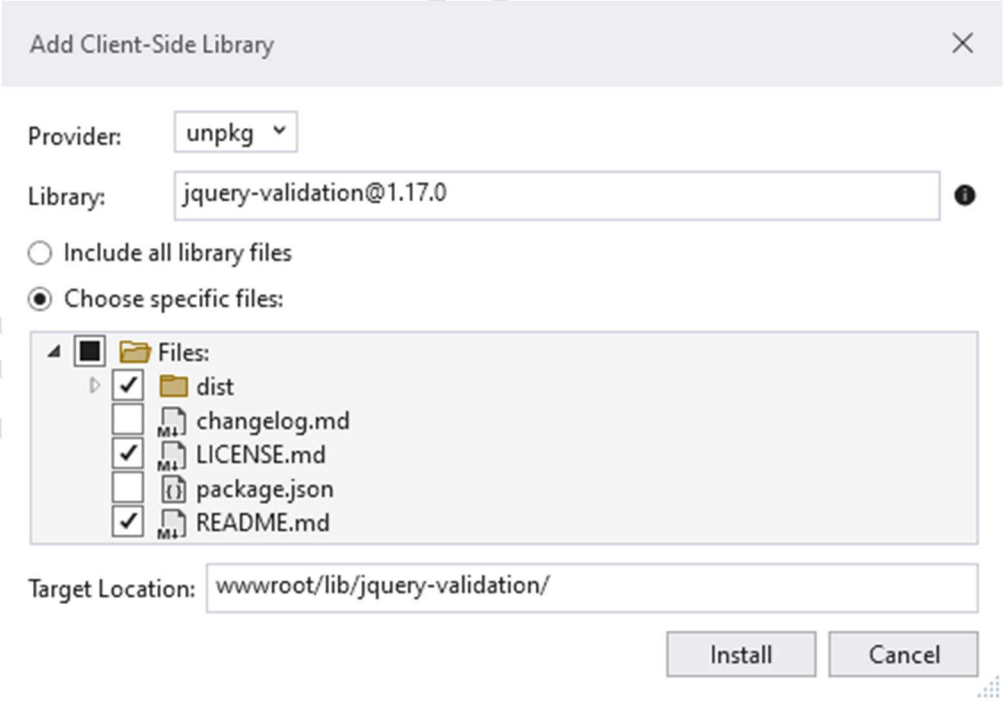
Trong thư mục **css**, tạo tập tin **main.css**. Trong thư mục **js**, tạo tập tin **main.js**. Tìm và thêm một vài hình ảnh vào thư mục **images** để làm hình nền, hình minh họa cho các bài viết trong blog.

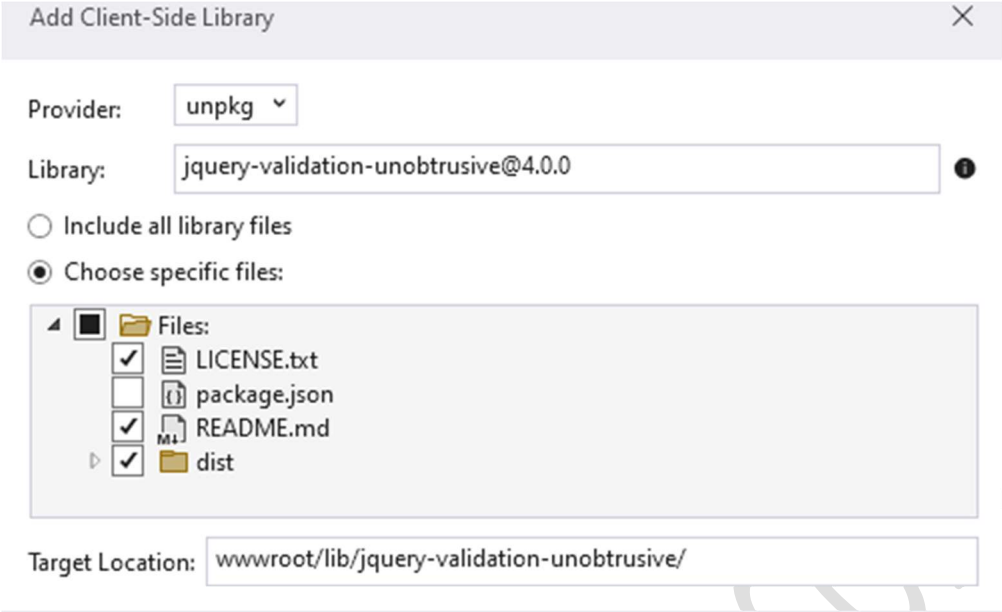


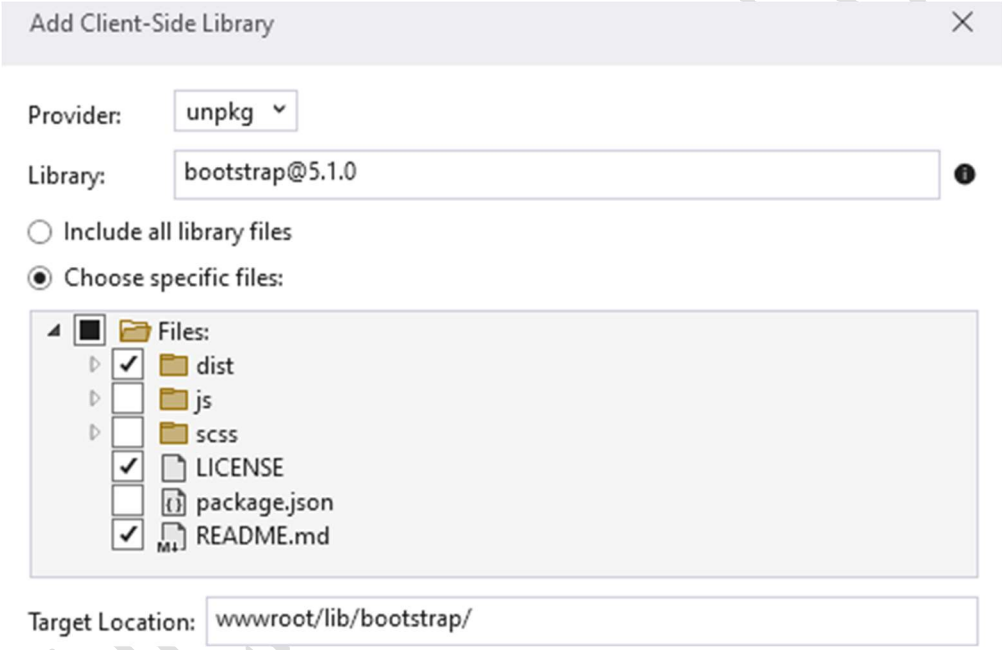
2. Cài đặt các thư viện Client-Side

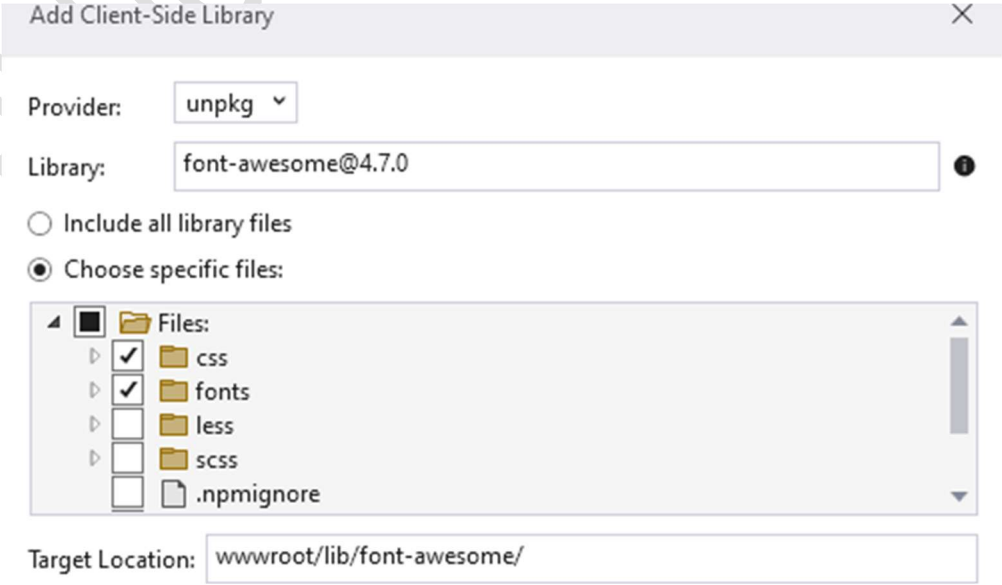
Nhấp phải chuột vào dự án **TatBlog.WebApp**, chọn **Add > Client-Side Library**. Lần lượt cài đặt các thư viện **jQuery, jQuery Validation, jQuery Validation Unobtrusive, Bootstrap, FontAwesome** theo các bước hướng dẫn bên dưới.



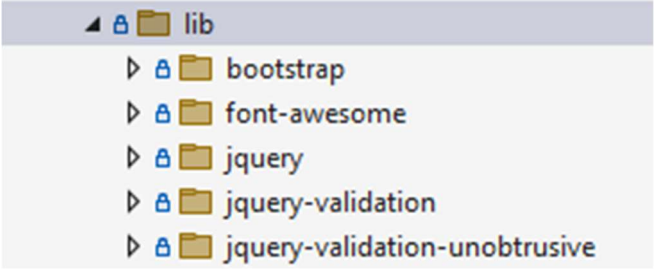



Add Client-Side Library
Provider:
unpkg
Library:
jquery-validation-unobtrusive@4.0.0
Include all library files
Choose specific files:
Files:
LICENSE.txt
package.json
README.md
dist
Target Location:
wwwroot/lib/jquery-validation-unobtrusive/


Add Client-Side Library
Provider:
unpkg
Library:
bootstrap@5.1.0
Include all library files
Choose specific files:
Files:
dist
js
scss
LICENSE
package.json
README.md
Target Location:
wwwroot/lib/bootstrap/


Add Client-Side Library
Provider:
unpkg
Library:
font-awesome@4.7.0
Include all library files
Choose specific files:
Files:
css
fonts
less
scss
.npmignore
Target Location:
wwwroot/lib/font-awesome/

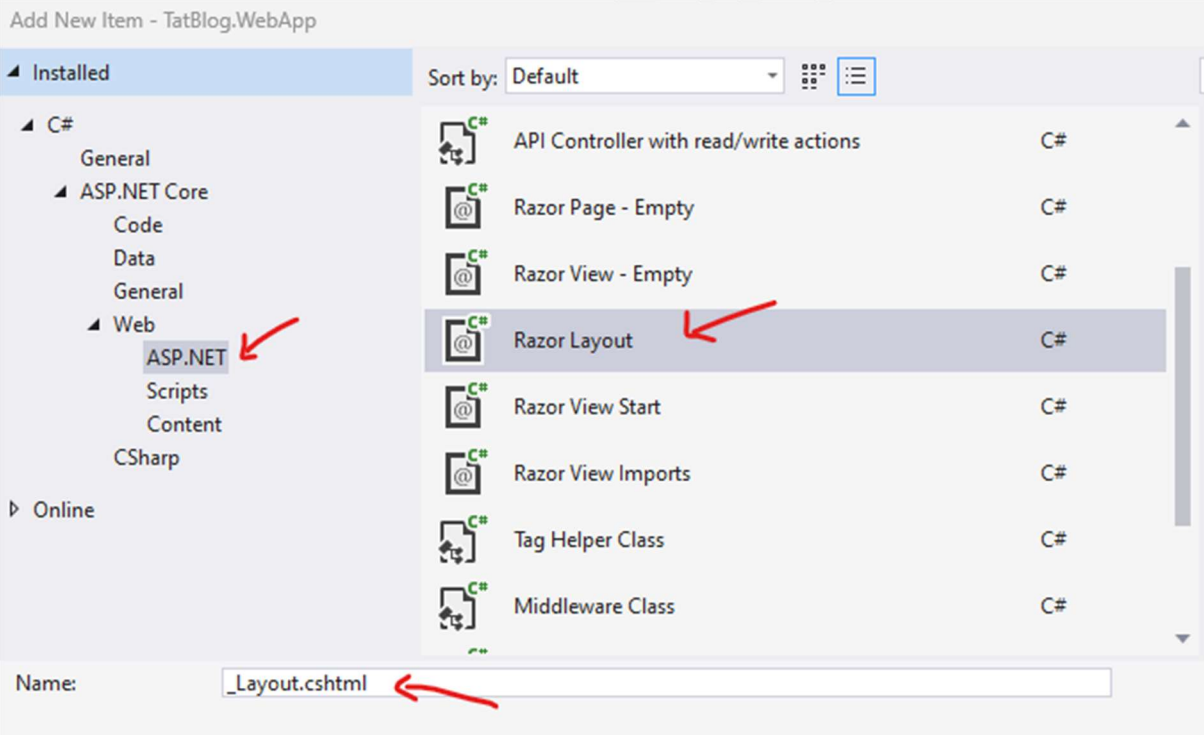
Cấu trúc thư mục **wwwroot > lib** sau khi cài đặt các thư viện:



3. Tạo bố cục (layout) chung cho trang blog

Trong thư mục **TatBlog.WebApp > Views**, tạo một thư mục mới, đặt tên là **Shared**. Thư mục này sẽ chứa các thành phần (partial view) và mẫu (template) giao diện dùng chung cho cả ứng dụng.

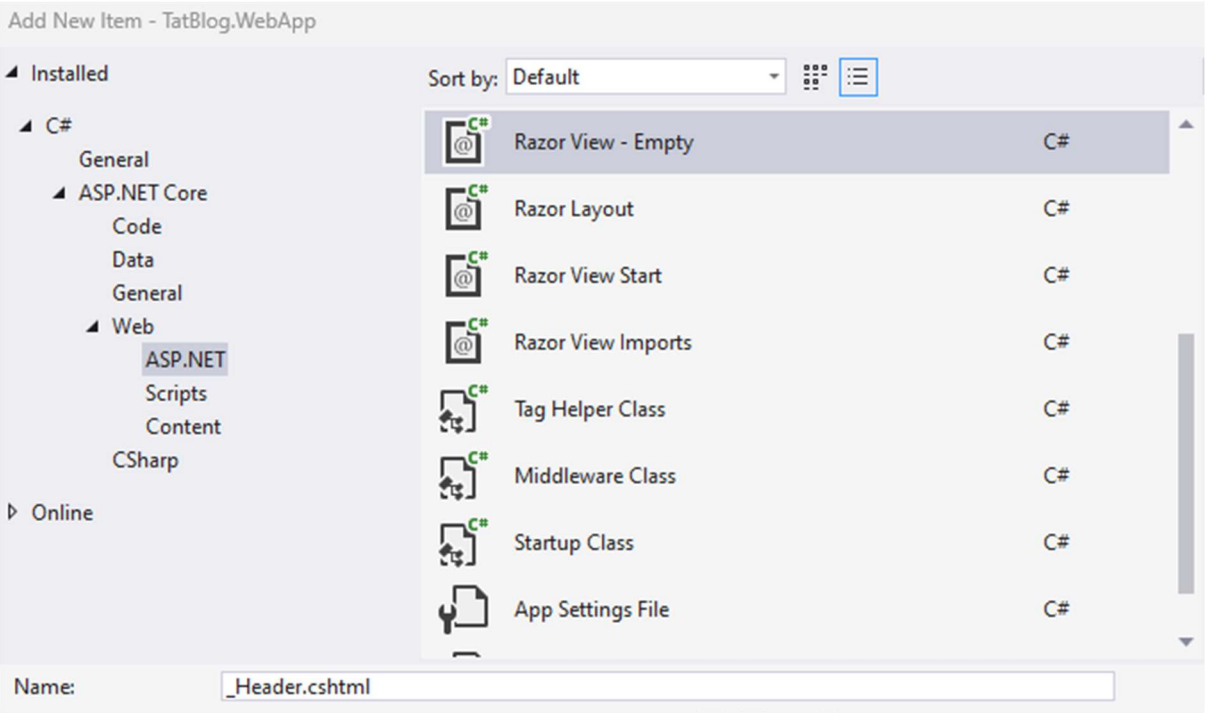
Nhấp phải chuột vào thư mục **Shared** vừa tạo, chọn **Add > New Item**. Trong cửa sổ **Add New Item**, chọn **C# > ASP.NET Core > Web > ASP.NET > Razor Layout**. Đặt tên cho tập tin là **_Layout.cshtml** rồi nhấn nút **Add** để thêm.



Cập nhật lại mã nguồn trong tập tin vừa tạo như hình dưới đây.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>@ViewData["PageTitle"] - Tips & Tricks Blog</title>

    <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
    <link rel="stylesheet" href="~/lib/font-awesome/css/font-awesome.min.css" />
    <link rel="stylesheet" href="~/css/main.css" asp-append-version="true" />

    @await RenderSectionAsync("styles", required: false)
</head>
<body>
    <!-- Thanh tiều đề, logo, menu -->
    <partial name="_Header" />

    <div class="container-fluid">
        <div class="row">
            <div class="col-9">
                <!-- Phần nội dung chính -->
                @RenderBody()
            </div>
            <div class="col-3 border-start">
                <!-- Danh sách chuyên mục, tag cloud, ... -->
                <partial name="_SideBar"/>
            </div>
        </div>
    </div>

    <footer class="border-top footer text-muted">
        <div class="container-fluid text-center">
            &copy; 2023 - Tips & Tricks Blog
        </div>
    </footer>

    <script src="~/lib/jquery/dist/jquery.min.js"></script>
    <script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
    <script src="~/js/main.js" asp-append-version="true"></script>

    @await RenderSectionAsync("scripts", required: false)
</body>
</html>
```

Tập tin layout ở trên có sử dụng 2 partial view là **_Header** và **_SiderBar**. Để tạo partial view, nhấp phải chuột vào thư mục **Shared**, chọn **Add > New Item > Razor View - Empty**. Đặt tên tập tin là **_Header.cshtml**.



Cập nhật mã nguồn trong tập tin **_Header** vừa tạo để tạo thanh tiêu đề và menu cho ứng dụng.

```
<header>
    <nav class="navbar navbar-expand-sm
         navbar-toggleable-sm navbar-light
         bg-white border-bottom box-shadow">
        <div class="container-fluid">
            <a class="navbar-brand"
               asp-area=""
               asp-controller="Blog"
               asp-action="Index">
                Tips & Tricks
            </a>

            <button class="navbar-toggler"
                    type="button"
                    data-bs-toggle="collapse"
                    data-bs-target=".navbar-collapse"
                    aria-controls="navbarSupportedContent"
                    aria-expanded="false"
                    aria-label="Toggle navigation">
                <span class="navbar-toggler-icon"></span>
            </button>

            <div class="navbar-collapse collapse
                 d-sm-inline-flex
                 justify-content-between">
                <ul class="navbar-nav flex-grow-1">
                    <li class="nav-item">
```

```
29                                  <a class="nav-link text-dark"
30                                     asp-area=""
31                                     asp-controller="Blog"
32                                     asp-action="Index">
33                                      Trang chủ
34                                  </a>
35                              </li>
36                              <li class="nav-item">
37                                  <a class="nav-link text-dark"
38                                     asp-area=""
39                                     asp-controller="Blog"
40                                     asp-action="Contact"
41                                     title="Xem thông tin liên lạc và góp ý">
42                                      Liên hệ
43                                  </a>
44                              </li>
45                              <li class="nav-item">
46                                  <a class="nav-link text-dark"
47                                     asp-area=""
48                                     asp-controller="Blog"
49                                     asp-action="About"
50                                     title="Thông tin về TAT Blog">
51                                      Giới thiệu
52                                  </a>
53                              </li>
54                              <li class="nav-item">
55                                  <a class="nav-link text-dark"
56                                     asp-area=""
57                                     asp-controller="Blog"
58                                     asp-action="Rss"
59                                     title="Tải danh sách bài viết mới">
60                                      RSS Feed
61                                  </a>
62                              </li>
63                          </ul>
64                      </div>
65                  </div>
66              </nav>
67          </header>
```

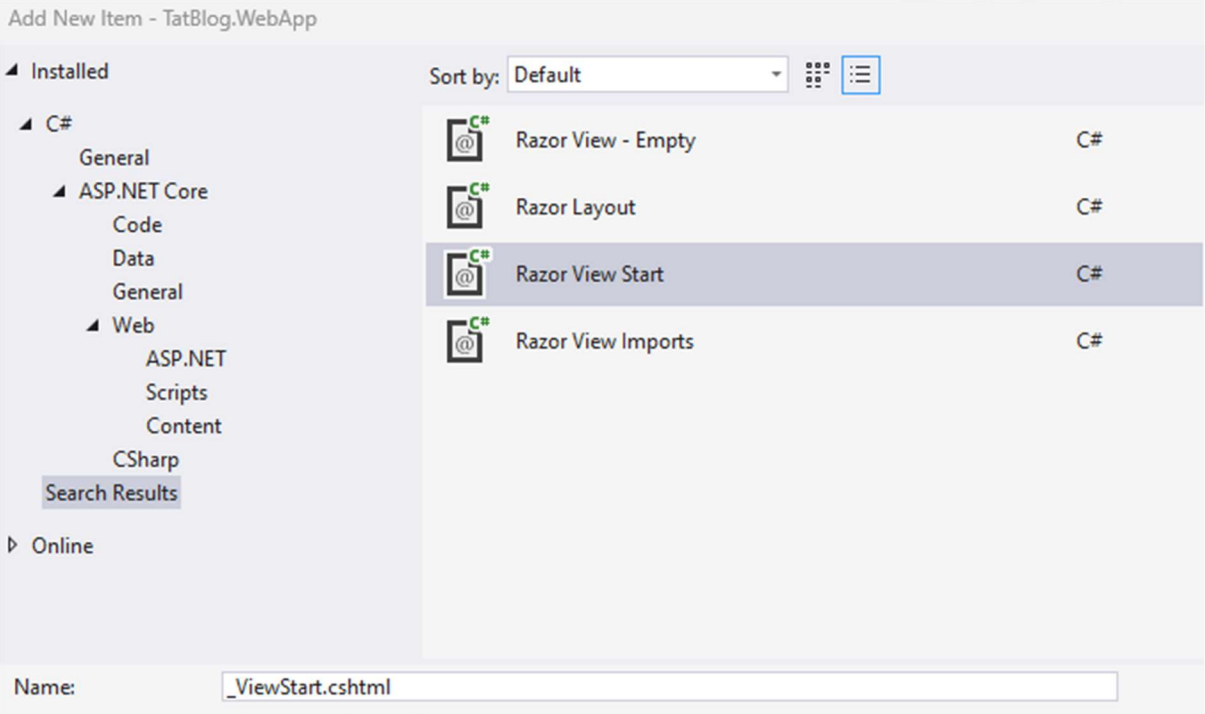
Tương tự, tạo partial view **_SideBar** với nội dung như sau.

```
1   <!-- Form tìm kiếm -->
2   <h1>
3       Tìm kiếm bài viết
4   </h1>
5
6   <!-- Các chuyên mục/chủ đề -->
7   <h1>
8       Các chủ đề
9   </h1>
10
11  <!-- Top bài viết nổi bật -->
12  <h1>
13      Bài viết nổi bật
14  </h1>
15
16  <!-- Đăng ký nhận tin mới -->
17  <h1>
18      Đăng ký nhận tin mới
19  </h1>
20
21  <!-- Tag cloud -->
22  <h1>
23      Tag cloud
24  </h1>
```

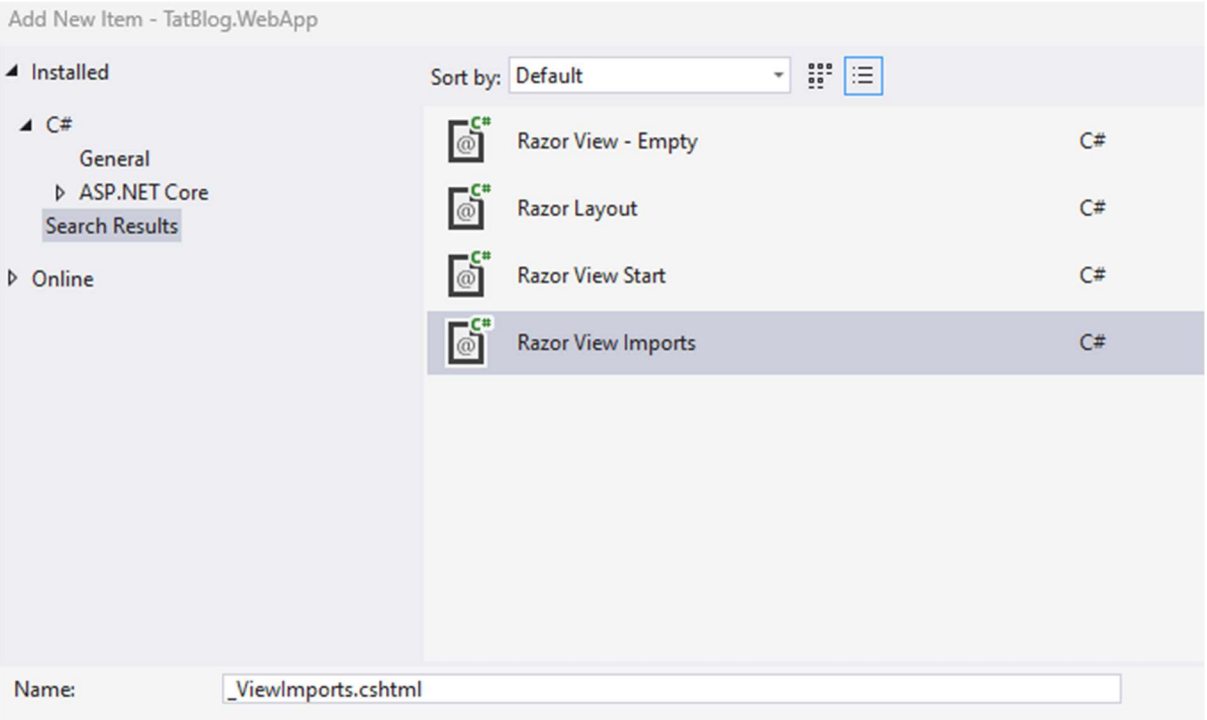
Trong thư mục **Shared**, tạo thêm một partial view, đặt tên là **_ValidationScriptsPartial.cshtml**. Sau đó, nhập đoạn mã dưới đây.

```
1 <script
2     src="~/lib/jquery-validation/dist/jquery.validate.min.js">
3 </script>
4 <script
5     src="~/lib/jquery-validation-unobtrusive/dist/jquery.validate.unobtrusive.min.js">
6 </script>
```

Nhấp phải chuột vào thư mục **Views**, chọn **Add > New Item > Razor View Start**. Đặt tên tập tin là **_ViewStart.cshtml** rồi nhấn nút **Add**.



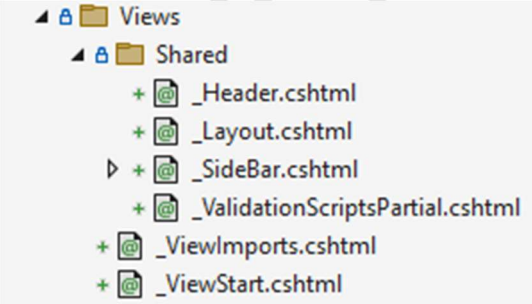
Thực hiện các bước tương tự để tạo tập tin **_ViewImports.cshtml**.



Cập nhật lại mã nguồn trong tập tin **_ViewImports.cshtml** vừa tạo.

```
1  @using TatBlog.Core.Collections
2  @using TatBlog.Core.Contracts
3  @using TatBlog.Core.DTO
4  @using TatBlog.Core.Entities
5
6  @addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
```

Sau bước này, thư mục **Views** có cấu trúc như sau:



4. Tạo controller và action

Nhấp phải chuột vào thư mục **Controllers** trong dự án **TatBlog.WebApp**, chọn **Add > Class**. Đặt tên cho class là **BlogController**. Nhấn nút **Add**.

Cài đặt lớp **BlogController** và các phương thức **Index, About, Contact, Rss** như sau:

```
using Microsoft.AspNetCore.Mvc;

namespace TatBlog.WebApp.Controllers
{
    0 references | 0 changes | 0 authors, 0 changes
    public class BlogController : Controller
    {
        0 references | 0 changes | 0 authors, 0 changes
        public IActionResult Index()
        {
            ViewBag.CurrentTime = DateTime.Now.ToString("HH:mm:ss");

            return View();
        }

        0 references | 0 changes | 0 authors, 0 changes
        public IActionResult About()
            => View();

        0 references | 0 changes | 0 authors, 0 changes
        public IActionResult Contact()
            => View();

        0 references | 0 changes | 0 authors, 0 changes
        public IActionResult Rss()
            => Content("Nội dung sẽ được cập nhật");
    }
}

```

5. Tạo view để trình bày kết quả trả về từ các action

Trong thư mục **TatBlog.WebApp > Views**, tạo thêm thư mục **Blog**. Sau đó, trong thư mục **Blog**, tạo các view và lần lượt đặt tên là **Index.cshtml, About.cshtml, Contact.cshtml**. Dưới đây là mã nguồn của các tập tin tương ứng.

Index.cshtml

```
@{
    ViewData["PageTitle"] = "Trang chủ";
}

<h1>
    Đây là trang chủ
</h1>

<p>
    Bây giờ là: @ViewBag.CurrentTime
</p>
```

About.cshtml

```
@{
    ViewData["PageTitle"] = "Giới thiệu";
}

<h1>
    Đây là trang giới thiệu
</h1>
```

Contact.cshtml

```
@{
    ViewData["PageTitle"] = "Liên hệ";
}

<h1>
    Đây là trang liên hệ
</h1>
```

6. Đăng ký các dịch vụ và cấu hình request pipeline

Mở tập tin **Program**, cập nhật lại mã nguồn như hình dưới đây.

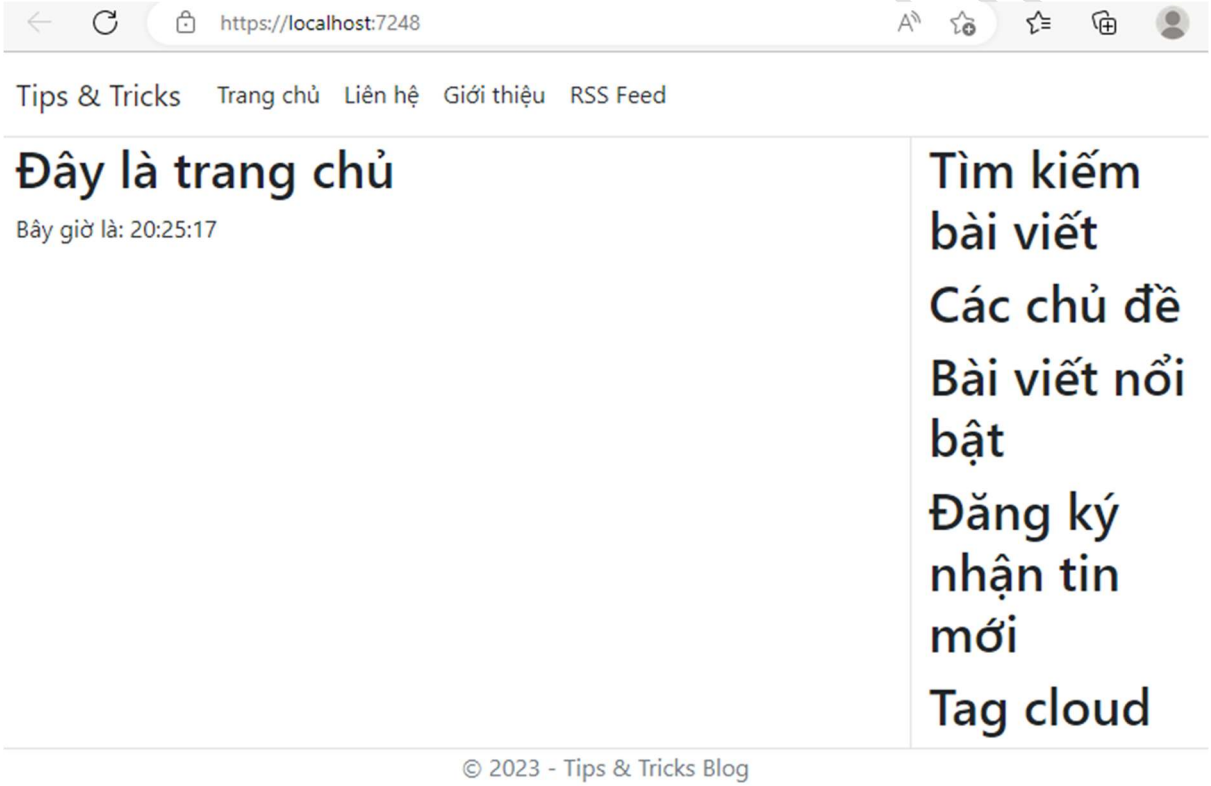
```
var builder = WebApplication.CreateBuilder(args);
{
    // Thêm các dịch vụ được yêu cầu bởi MVC Framework
    builder.Services.AddControllersWithViews();
}

var app = builder.Build();
{
    // Cấu hình HTTP Request pipeline

    // Thêm middleware để hiển thị thông báo lỗi
    if (app.Environment.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }
    else
    {
        app.UseExceptionHandler("/Blog/Error");

        // Thêm middleware cho việc áp dụng HSTS (thêm header
        // Strict-Transport-Security vào HTTP Response).
        app.UseHsts();
    }

    // Thêm middleware để chuyển hướng HTTP sang HTTPS
    app.UseHttpsRedirection();

    // Thêm middleware phục vụ các yêu cầu liên quan
    // tới các tập tin nội dung tĩnh như hình ảnh, css, ...
    app.UseStaticFiles();

    // Thêm middleware lựa chọn endpoint phù hợp nhất
    // để xử lý một HTTP request.
    app.UseRouting();
```

NOT COPY

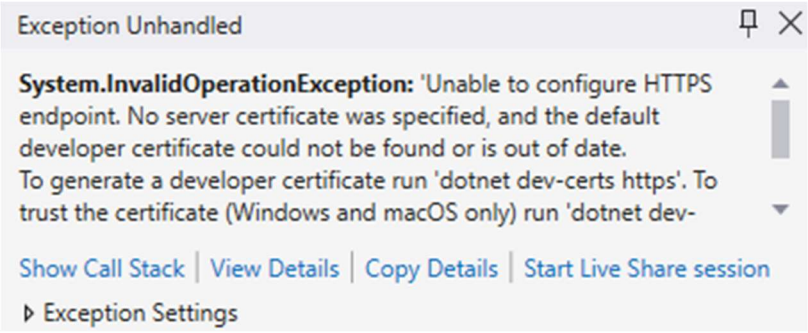
```
35
36      // Định nghĩa route template, route constraint cho các
37      // endpoints kết hợp với các action trong các controller.
38      app.MapControllerRoute(
39          name: "default",
40          pattern: "{controller=Blog}/{action=Index}/{id?}");
41  }
42
43  app.Run();
44
```

Nhấn tổ hợp phím **Ctrl + Shift + B** hoặc chọn menu **Build > Build Solution** để biên dịch và kiểm tra lỗi. Nếu có lỗi, dò lại các bước trước để kiểm tra và sửa lỗi.

Nhấn phím **F5** hoặc nút **Start Debugging** để chạy ứng dụng Web. Nếu không có lỗi, trình duyệt hiển thị kết quả tương tự như hình sau.



Trường hợp khi chạy chương trình gặp phải lỗi trong hình dưới, nhấp phải chuột vào dự án **TatBlog.WebApp**, chọn **Open Terminal**.



Sau đó chạy 2 lệnh: **dotnet dev-certs https --clean** và **dotnet dev-certs https --trust**

```
Developer PowerShell
+ Developer PowerShell
**********************************************************************
** Visual Studio 2022 Developer PowerShell v17.4.5
** Copyright (c) 2022 Microsoft Corporation
**********************************************************************
PS D:\Projects\Mine\TechBlogs\src\TipsAndTricks\TatBlog.WebApp> dotnet dev-certs https --clean
Cleaning HTTPS development certificates from the machine. A prompt might get displayed to confi
HTTPS development certificates successfully removed from the machine.
PS D:\Projects\Mine\TechBlogs\src\TipsAndTricks\TatBlog.WebApp> dotnet dev-certs https --trust
Trusting the HTTPS development certificate was requested. A confirmation prompt will be display
Successfully created and trusted a new HTTPS certificate.
PS D:\Projects\Mine\TechBlogs\src\TipsAndTricks\TatBlog.WebApp>
```

Click chuột vào các menu **Trang chủ, Liên hệ, Giới thiệu, RSS Feed** để xem kết quả.

8. Cấu hình chuỗi kết nối CSDL và cập nhật lớp DbContext, Program

Trong dự án **TatBlog.WebApp**, mở tập tin **appsettings.json**, bổ sung thêm chuỗi kết nối cơ sở dữ liệu (phù hợp với CSDL bạn đang dùng).

```
1  {
2      "Logging": {
3          "LogLevel": {
4              "Default": "Information",
5              "Microsoft.AspNetCore": "Warning"
6          }
7      },
8      "AllowedHosts": "*",
9      "ConnectionStrings": {
10         "DefaultConnection": "put-your-connection-string-here"
11     }
12 }
13
```

Cập nhật lại mã nguồn trong lớp **BlogDbContext**. Phương thức khởi tạo của lớp này nhận vào một đối tượng **DbContextOptions** chứa những thông tin cần thiết để khởi tạo đối tượng **DbContext**.

```
using Microsoft.EntityFrameworkCore;
using TatBlog.Core.Entities;
using TatBlog.Data.Mappings;

namespace TatBlog.Data.Contexts;

10 references | 0 changes | 0 authors, 0 changes
public class BlogDbContext : DbContext
{
    1 reference | 0 changes | 0 authors, 0 changes
    public DbSet<Author> Authors { get; set; }

    1 reference | 0 changes | 0 authors, 0 changes
    public DbSet<Category> Categories { get; set; }

    2 references | 0 changes | 0 authors, 0 changes
    public DbSet<Post> Posts { get; set; }

    1 reference | 0 changes | 0 authors, 0 changes
    public DbSet<Tag> Tags { get; set; }

    0 references | 0 changes | 0 authors, 0 changes
    public BlogDbContext(DbContextOptions<BlogDbContext> options)
        : base(options)
    {
    }

    0 references | 0 changes | 0 authors, 0 changes
    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.ApplyConfigurationsFromAssembly(
            typeof(CategoryMap).Assembly);
    }
}
```

Trong tập tin **Program.cs**, bổ sung mã để đăng ký các lớp thực hiện việc truy xuất dữ liệu với DI Container.

```
using Microsoft.EntityFrameworkCore;
using TatBlog.Data.Contexts;
using TatBlog.Data.Seeders;
using TatBlog.Services.Blogs;

var builder = WebApplication.CreateBuilder(args);
{
    // Thêm các dịch vụ được yêu cầu bởi MVC Framework
    builder.Services.AddControllersWithViews();

    // Đăng ký các dịch vụ với DI Container
    builder.Services.AddDbContext<BlogDbContext>(options =>
        options.UseSqlServer(
            builder.Configuration.GetConnectionString("DefaultConnection")));

    builder.Services.AddScoped<IBlogRepository, BlogRepository>();
    builder.Services.AddScoped<IDataSeeder, DataSeeder>();
}
```

Cũng trong tập tin **Program.cs**, thêm đoạn mã dưới đây để đảm bảo dữ liệu mẫu được thêm vào CSDL trước khi ứng dụng Web khởi chạy.

```
70
71  // Thêm dữ liệu mẫu vào CSDL
72  using (var scope = app.Services.CreateScope())
73  {
74      var seeder = scope.ServiceProvider.GetRequiredService<IDataSeeder>();
75      seeder.Initialize();
76  }
77
78  app.Run();
```

9. Hiển thị danh sách bài viết mới nhất lên trang chủ

Phần này yêu cầu bạn phải hoàn thành **bài tập 1.s.** của **Lab 01**. Giả sử, ta có phương thức tìm kiếm và phân trang các bài viết, sắp xếp giảm dần theo ngày giờ đăng bài như sau:

```
    1 reference | 0 changes | 0 authors, 0 changes
278 public async Task<IPagedList<Post>> GetPagedPostsAsync(
279     PostQuery condition,
280     int pageNumber = 1,
281     int pageSize = 10,
282     CancellationToken cancellationToken = default)
283 {
284     return await FilterPosts(condition).ToPagedListAsync(
285         pageNumber, pageSize,
286         nameof(Post.PostedDate), "DESC",
287         cancellationToken);
288 }
```

Trong lớp **BlogController**, tạo biến thành viên và cài đặt phương thức khởi tạo

```
    1 reference | 0 changes | 0 authors, 0 changes
7   public class BlogController : Controller
8   {
9       private readonly IBlogRepository _blogRepository;
10
    0 references | 0 changes | 0 authors, 0 changes
11      public BlogController(IBlogRepository blogRepository)
12      {
13          _blogRepository = blogRepository;
14      }
```

Cập nhật lại mã nguồn trong **action Index** của **BlogController** như sau

```
16        // Action này xử lý HTTP request đến trang chủ của
17        // ứng dụng web hoặc tìm kiếm bài viết theo từ khóa
          0 references | 0 changes | 0 authors, 0 changes
18        public async Task<IActionResult> Index(
19            [FromQuery(Name = "p")] int pageNumber = 1,
20            [FromQuery(Name = "ps")] int pageSize = 10)
21        {
22            // Tạo đối tượng chứa các điều kiện truy vấn
23            var postQuery = new PostQuery()
24            {
25                // Chỉ lấy những bài viết có trạng thái Published
26                PublishedOnly = true
27            };
28
29            // Truy vấn các bài viết theo điều kiện đã tạo
30            var postsList = await _blogRepository
31                .GetPagedPostsAsync(postQuery, pageNumber, pageSize);
32
33            // Lưu lại điều kiện truy vấn để hiển thị trong View
34            ViewBag.PostQuery = postQuery;
35
36            // Truyền danh sách bài viết vào View để render ra HTML
37            return View(postsList);
38        }
```

Trong thư mục **Views > Blog**, tạo partial view **_TagList.cshtml**. Nhập đoạn mã dưới đây để hiển thị các từ khóa của bài viết.

```
1  @model IList<Tag>
2
3  @foreach (var item in Model)
4  {
5      <a asp-controller="Blog"
6         asp-action="Tag"
7         asp-route-slug="@item.UrlSlug"
8         title="Tag: @item.Name"
9         class="btn btn-sm btn-outline-secondary">
10         @item.Name
11     </a>
12 }
```

Tiếp tục tạo thêm partial view **_PostList.cshtml** để hiển thị thông tin tóm tắt một bài viết.

```
1  @model Post
2
3  @{
4      var imageUrl = string.IsNullOrWhiteSpace(Model.ImageUrl)
5          ? "/images/image_1.jpg"
6          : $"/{Model.ImageUrl}";
7  }
8
```

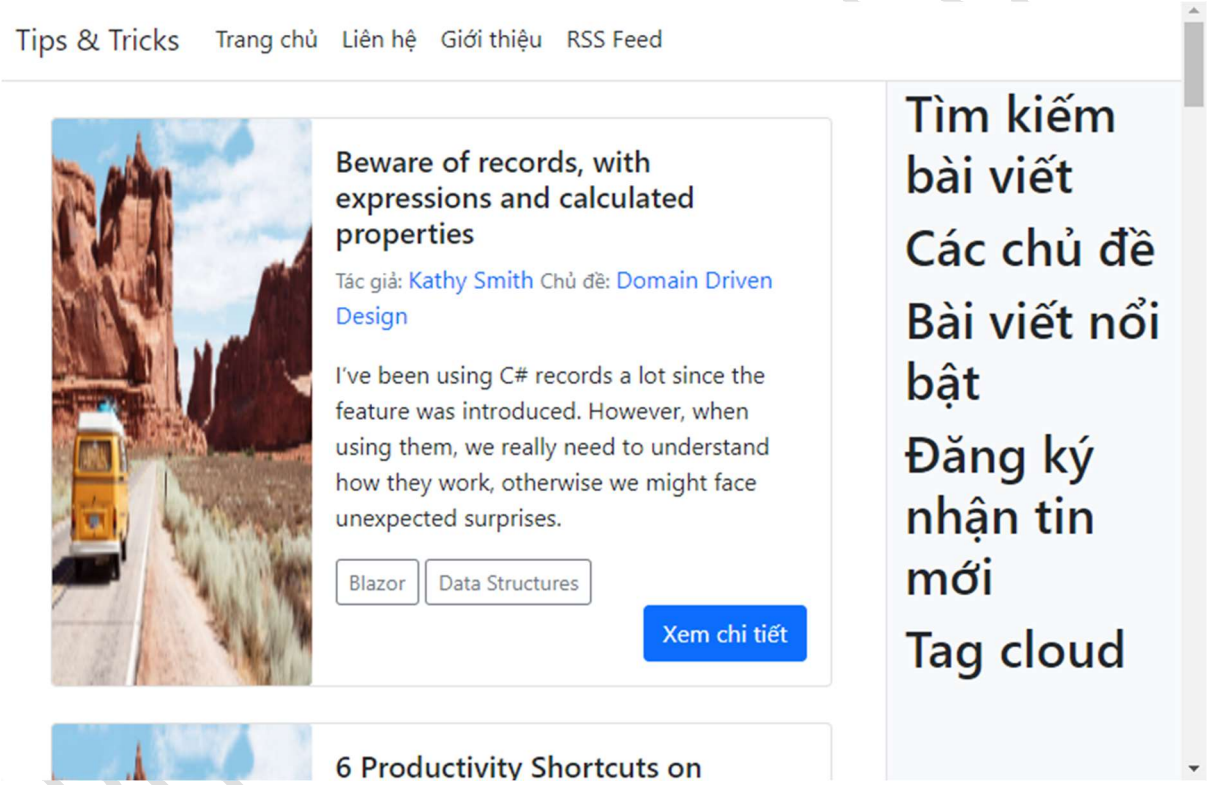
```
9  <article class="blog-entry mb-4">
10     <div class="card">
11         <div class="row g-0">
12             <div class="col-md-4">
13                 <img src="@imageUrl"
14                      class="card-img"
15                      alt="@Model.Title">
16             </div>
17             <div class="col-md-8">
18                 <div class="card-body">
19                     <h5 class="card-title">
20                         @Model.Title
21                     </h5>
22                     <p class="card-text">
23                         <small class="text-muted">
24                             Tác giả:
25                         </small>
26                         <span class="text-primary">
27                             @Model.Author.FullName
28                         </span>
29                         <small class="text-muted">
30                             Chủ đề:
31                         </small>
32                         <span class="text-primary">
33                             @Model.Category.Name
34                         </span>
35                     </p>
36                     <p class="card-text">
37                         @Html.Raw(Model.ShortDescription)
38                     </p>
39                     <div class="tag-list">
40                         <partial name="_TagList" model="@Model.Tags"/>
41                     </div>
44                     <div class="text-end">
45                         <a asp-controller="Blog"
46                            asp-action="Post"
47                            asp-route-year="@Model.PostedDate.Year"
48                            asp-route-month="@Model.PostedDate.Month"
49                            asp-route-day="@Model.PostedDate.Day"
50                            asp-route-slug="@Model.UrlSlug"
51                            class="btn btn-primary"
52                            title="@Model.Title">
53                             Xem chi tiết
54                         </a>
55                     </div>
56                 </div>
57             </div>
58         </div>
59     </div>
60 </article>
```

Cập nhật mã nguồn trong tập tin **Views > Blog > Index.cshtml** như sau

```
@model IPagedList<Post>

@{
    ViewData["PageTitle"] = "Trang chủ";
}

<div class="p-4">
    @foreach (var postItem in Model)
    {
        <partial name="_PostItem" model="postItem"/>
    }
</div>
```

Nhấn tổ hợp phím **Ctrl + Shift + B** hoặc chọn menu **Build > Build Solution** để biên dịch, kiểm tra và sửa lỗi (nếu có).

Chạy chương trình và kiểm tra kết quả.



10. Định nghĩa Route Template để tạo Friendly URL

Để ý rằng, khi rê chuột vào nút **Xem chi tiết** hoặc các từ khóa trong tag list, bạn sẽ thấy địa chỉ URL trông có vẻ rối rắm. Chẳng hạn:

- https://localhost:7248/Blog/Post?year=2022&month=10&day=15&slug=beware-of-records-with-expressions-and-calculated-properties
- https://localhost:7248/Blog/Tag?slug=blazor
- https://localhost:7248/Blog/Category?slug=dot-net-core

Ta sẽ định nghĩa các route template để chuyển các URL về dạng thân thiện hơn với người dùng như sau:

- https://localhost:7248/blog/post/2022/10/15/beware-of-records-with-expressions-and-calculated-properties
- https://localhost:7248/blog/tag/blazor
- https://localhost:7248/blog/category/dot-net-core

Trong tập tin **Program.cs**, bổ sung thêm mã nguồn để định nghĩa các route template như sau:

```
49      // Định nghĩa route template, route constraint cho các
50      // endpoints kết hợp với các action trong các controller.
51      app.MapControllerRoute(
52          name: "posts-by-category",
53          pattern: "blog/category/{slug}",
54          defaults: new { controller = "Blog", action = "Category" });
55
56      app.MapControllerRoute(
57          name: "posts-by-tag",
58          pattern: "blog/tag/{slug}",
59          defaults: new { controller = "Blog", action = "Tag" });
60
61      app.MapControllerRoute(
62          name: "single-post",
63          pattern: "blog/post/{year:int}/{month:int}/{day:int}/{slug}",
64          defaults: new { controller = "Blog", action = "Post" });
65
66      app.MapControllerRoute(
67          name: "default",
68          pattern: "{controller=Blog}/{action=Index}/{id?}");
```

Chạy lại chương trình, rê chuột vào nút **Xem chi tiết** hoặc các từ khóa (tag) để xem sự thay đổi.

11. Tạo thanh điều hướng trang (Pager)

Trong thư mục **TatBlog.WebApp > Views > Blog**, tạo partial view, đặt tên là **_Pager.cshtml**. Nhập đoạn mã sau:

```
1   @model IPagedList<Post>
2
3   @{
4       var postQuery = (ViewBag.PostQuery as PostQuery) ?? new PostQuery();
5       var actionName = "Index";
6       var slug = "";
7
8       if (!string.IsNullOrWhiteSpace(postQuery.AuthorSlug))
9       {
10          actionName = "Author";
11      }
```

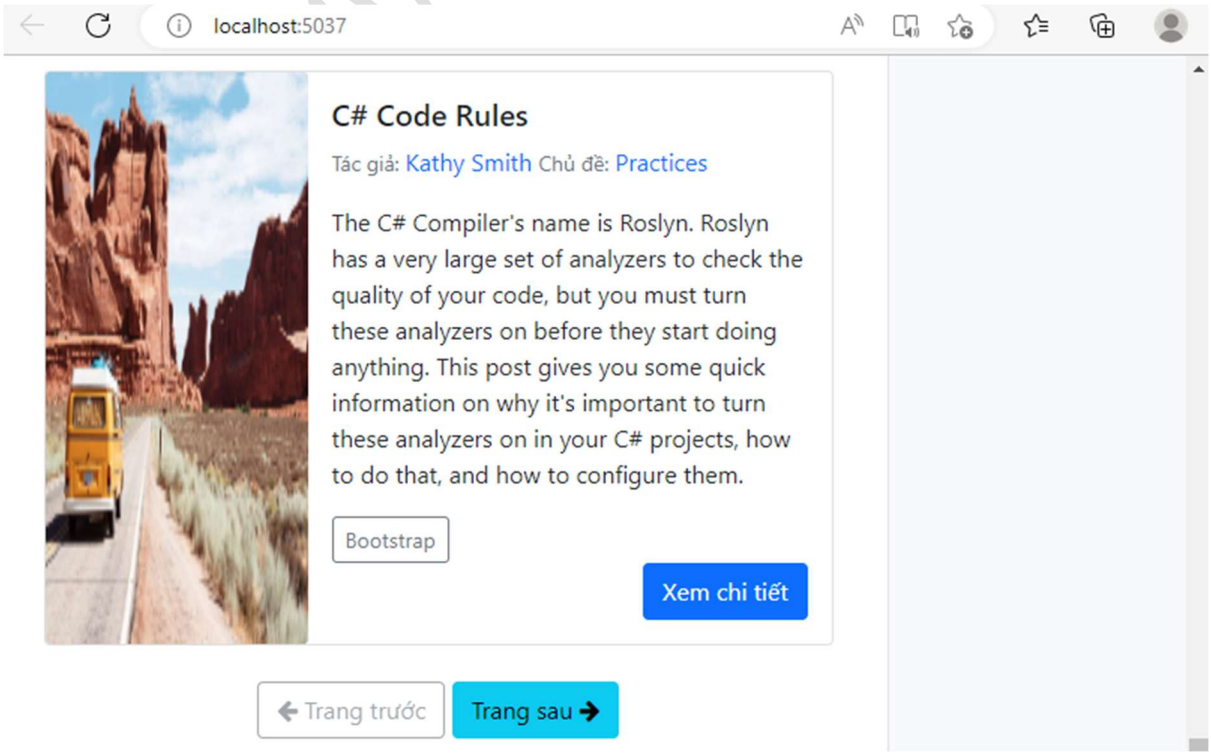
```
12      else if (!string.IsNullOrWhiteSpace(postQuery.CategorySlug))
13      {
14          actionName = "Category";
15      }
16      else if (!string.IsNullOrWhiteSpace(postQuery.TagSlug))
17      {
18          actionName = "Tag";
19      }
20  }
21
22  @if (Model.PageCount > 1)
23  {
24      <div class="text-center my-4">
25          @if (Model.HasPreviousPage)
26          {
27              <a asp-controller="Blog"
28                 asp-action="@actionName"
29                 asp-route-slug="@slug"
30                 asp-route-k="@postQuery.Keyword"
31                 asp-route-p="@(Model.PageNumber - 1)"
32                 asp-route-ps="@Model.PageSize"
33                 class="btn btn-info">
34                  <i class="fa fa-arrow-left"></i>
35                  Trang trước
36              </a>
37          }
38          else
39          {
40              <button type="button"
41                      class="btn btn-outline-secondary"
42                      disabled>
43                  <i class="fa fa-arrow-left"></i>
44                  Trang trước
45              </button>
46          }
47
48          @if (Model.HasNextPage)
49          {
50              <a asp-controller="Blog"
51                 asp-action="@actionName"
52                 asp-route-slug="@slug"
53                 asp-route-k="@postQuery.Keyword"
54                 asp-route-p="@(Model.PageNumber + 1)"
55                 asp-route-ps="@Model.PageSize"
56                 class="btn btn-info">
57                  Trang sau
58                  <i class="fa fa-arrow-right"></i>
59              </a>
60          }
```

```
61          else
62          {
63              <button type="button"
64                      class="btn btn-outline-secondary"
65                      disabled>
66                  Trang sau
67                  <i class="fa fa-arrow-right"></i>
68              </button>
69          }
70      </div>
71  }
```

Cập nhật mã nguồn trong tập tin **Views > Blog > Index.cshtml** để sử dụng partial view vừa tạo.

```
1   @model IPagedList<Post>
2
3   @{
4       ViewData["PageTitle"] = "Trang chủ";
5   }
6
7   <div class="p-4">
8       @foreach (var postItem in Model)
9       {
10          <partial name="_PostItem" model="postItem"/>
11      }
12
13      <partial name="_Pager" model="@Model"/>
14  </div>
```

Chạy chương trình và kiểm tra kết quả



12. Xây dựng chức năng tìm kiếm

Trong thư mục **TatBlog.WebApp > Views > Shared**, tạo thêm partial view **_SearchForm.cshtml**. Viết mã để tạo ra một biểu mẫu đơn giản cho phép người dùng nhập vào từ khóa để tìm bài viết.

```
<div class="mb-4">
    <form asp-controller="Blog"
          asp-action="Index"
          method="get">
        <div class="input-group mb-3">
            <input type="text"
                   class="form-control"
                   name="k"
                   placeholder="Enter keyword"
                   aria-label="Enter keyword"
                   aria-describedby="btnSearchPost">

            <button class="btn btn-outline-secondary"
                    type="button"
                    id="btnSearchPost">
                <i class="fa fa-search"></i>
            </button>
        </div>
    </form>
</div>
```

Cập nhật mã nguồn của file **Shared > _SideBar.cshtml** để hiển thị form tìm kiếm.

```
<div class="my-4">
    <!-- Form tìm kiếm -->
    <partial name="_SearchForm" />

    <!-- Các chuyên mục/chủ đề -->
    <h1>
        Các chủ đề
    </h1>

    <!-- Top bài viết nổi bật -->
    <h1>
        Bài viết nổi bật
    </h1>

    <!-- Đăng ký nhận tin mới -->
    <h1>
        Đăng ký nhận tin mới
    </h1>

    <!-- Tag cloud -->
    <h1>
        Tag cloud
    </h1>
</div>
```

Trong phương thức **Index** của lớp **BlogController**, ta bổ sung thêm mã để lấy được từ khóa do người dùng nhập và truy vấn bài viết dựa trên từ khóa đó.

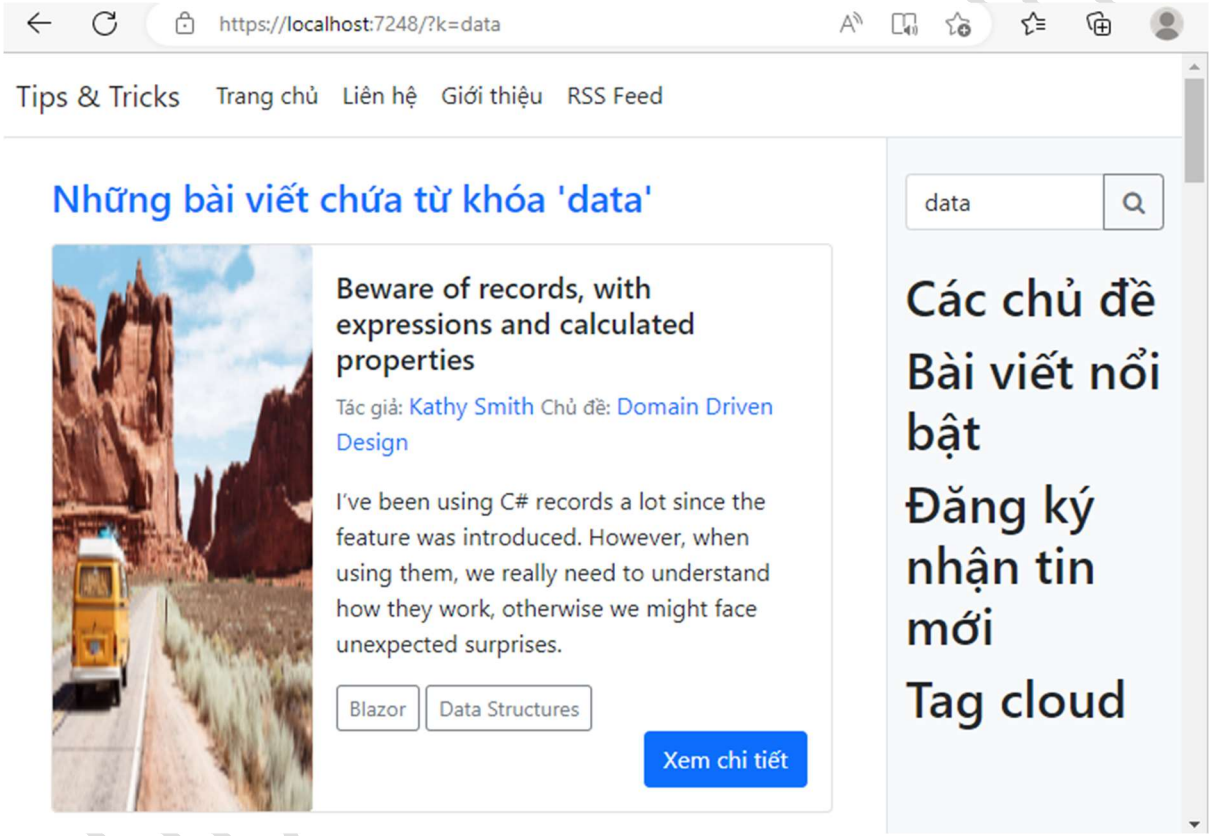
```
16        // Action này xử lý HTTP request đến trang chủ của
17        // ứng dụng web hoặc tìm kiếm bài viết theo từ khóa
          0 references | 0 changes | 0 authors, 0 changes
18        public async Task<IActionResult> Index(
19            [FromQuery(Name = "k")] string keyword = null,
20            [FromQuery(Name = "p")] int pageNumber = 1,
21            [FromQuery(Name = "ps")] int pageSize = 10)
22        {
23            // Tạo đối tượng chứa các điều kiện truy vấn
24            var postQuery = new PostQuery()
25            {
26                // Chỉ lấy những bài viết có trạng thái Published
27                PublishedOnly = true,
28
29                // Tìm bài viết theo từ khóa
30                Keyword = keyword
31            };
32
33            // Truy vấn các bài viết theo điều kiện đã tạo
34            var postsList = await _blogRepository
35                .GetPagedPostsAsync(postQuery, pageNumber, pageSize);
36
37            // Lưu lại điều kiện truy vấn để hiển thị trong View
38            ViewBag.PostQuery = postQuery;
39
40            // Truyền danh sách bài viết vào View để render ra HTML
41            return View(postsList);
42        }
```

Cập nhật mã trong tập tin **Views > Blog > Index.cshtml** để hiển thị thêm tiêu đề trang.

```
1   @model IPagedList<Post>
2
3   @{
4       ViewData["PageTitle"] = "Trang chủ";
5
6       var postQuery = (ViewBag.PostQuery as PostQuery) ?? new PostQuery();
7   }
8
9   <div class="p-4">
10      @if (!string.IsNullOrWhiteSpace(postQuery.Keyword))
11      {
12          <h3 class="text-primary mb-3">
13              Những bài viết chứa từ khóa '@postQuery.Keyword'
14          </h3>
15      }
16
```

```
16
17      @foreach (var postItem in Model)
18      {
19          <partial name="_PostItem" model="postItem"/>
20      }
21
22      <partial name="_Pager" model="@Model"/>
23  </div>
```

Chạy chương trình, thử nhập một từ khóa vào form tìm kiếm và kiểm tra kết quả.



13. Hiển thị danh sách chủ đề

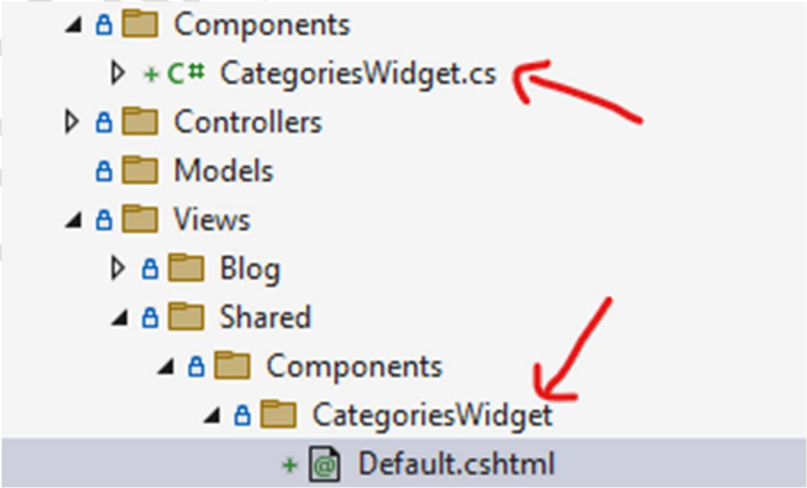
Trong phần này, ta sẽ tạo một **ViewComponent** để hiển thị danh sách các chủ đề nằm trong phần sidebar ở phía bên phải của trang blog.

Trong dự án **TatBlog.WebApp**, tạo thư mục **Components**. Tại thư mục này, tạo một lớp mới, đặt tên là **CategoriesWidget**. Cài đặt lớp **CategoriesWidget** như trong hình sau đây.

```
using Microsoft.AspNetCore.Mvc;
using TatBlog.Services.Blogs;

namespace TatBlog.WebApp.Components;

1 reference | 0 changes | 0 authors, 0 changes
public class CategoriesWidget : ViewComponent
{
    private readonly IBlogRepository _blogRepository;

    0 references | 0 changes | 0 authors, 0 changes
    public CategoriesWidget(IBlogRepository blogRepository)
    {
        _blogRepository = blogRepository;
    }

    0 references | 0 changes | 0 authors, 0 changes
    public async Task<IViewComponentResult> InvokeAsync()
    {
        // Lấy danh sách chủ đề
        var categories = await _blogRepository.GetCategoriesAsync();

        return View(categories);
    }
}
```

Trong thư mục **TatBlog.WebApp > Views > Shared**, tạo thêm một thư mục **Components** để chứa các view trình bày dữ liệu trả về từ **ViewComponent**.

Tiếp đến, tạo thư mục con của thư mục **Components**, đặt tên là **CategoriesWidget**. Sau đó, tạo một **View**, đặt tên là **Default.cshtml**. Hình dưới đây cho thấy cấu trúc thư mục sau khi tạo các thư mục và tập tin cần thiết.



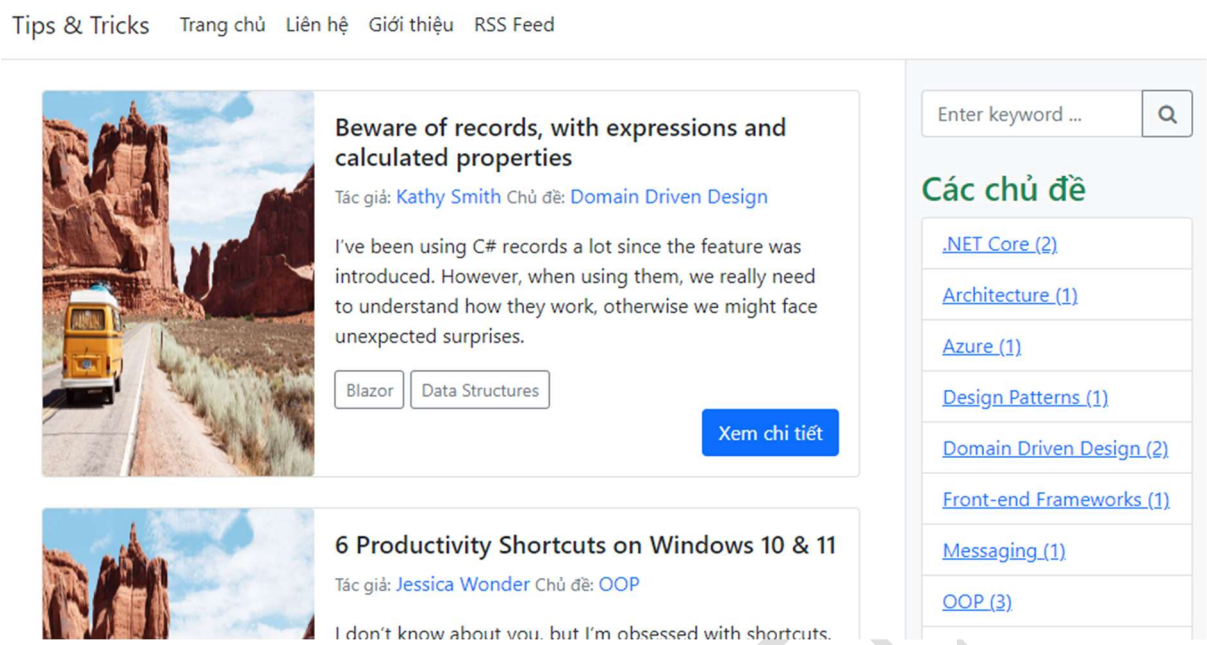
Cài đặt mã nguồn vào tập tin **Default.cshtml** để hiển thị danh sách các chủ đề.

```
@model IList<CategoryItem>

<div class="mb-4">
    <h3 class="text-success mb-2">
        Các chủ đề
    </h3>

    <ul class="list-group">
        @foreach (var item in Model)
        {
            <li class="list-group-item">
                <a asp-controller="Blog"
                   asp-action="Category"
                   asp-route-slug="@item.UrlSlug"
                   title="@item.Description">
                    @item.Name
                    <span>(@item.PostCount)</span>
                </a>
            </li>
        }
    </ul>
</div>
```

Trong tập tin **_SideBar.cshtml**, cập nhật lại mã nguồn để gọi tới ViewComponent vừa tạo.

```
<div class="my-4">
    <!-- Form tìm kiếm -->
    <partial name="_SearchForm" />

    <!-- Các chuyên mục/chủ đề -->
    @await Component.InvokeAsync("CategoriesWidget")

    <!-- Top bài viết nổi bật -->
    <h1>
        Bài viết nổi bật
    </h1>

    <!-- Đăng ký nhận tin mới -->
    <h1>
        Đăng ký nhận tin mới
    </h1>

    <!-- Tag cloud -->
    <h1>
        Tag cloud
    </h1>
</div>
```

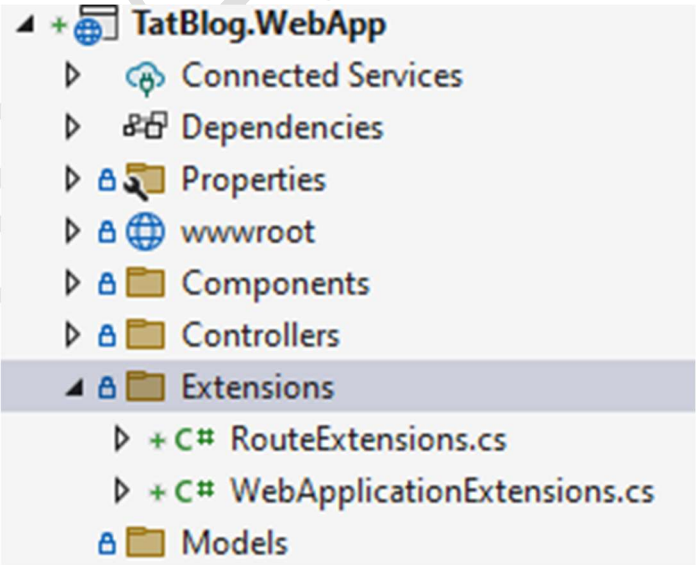
Chạy chương trình và kiểm tra kết quả



14. Tinh gọn mã nguồn trong tập tin Program.cs

Trong phần này, ta sẽ làm gọn lại các đoạn mã trong tập tin **Program** bằng cách sử dụng các phương thức mở rộng (**extension methods**).

Trước tiên, trong dự án **TatBlog.WebApp**, tạo thư mục mới **Extensions**. Sau đó, tạo tiếp 2 lớp tĩnh (**static class**), đặt tên lần lượt là **WebApplicationExtensions** và **RouteExtensions**.



Trong lớp **WebApplicationExtensions**, lần lượt định nghĩa các phương thức sau (sinh viên có thể copy nội dung từ tập tin **Program.cs** sang).

```
10      // Thêm các dịch vụ được yêu cầu bởi MVC Framework
        1 reference | 0 changes | 0 authors, 0 changes
11      public static WebApplicationBuilder ConfigureMvc(
12          this WebApplicationBuilder builder)
13      {
14          builder.Services.AddControllersWithViews();
15          builder.Services.AddResponseCompression();
16
17          return builder;
18      }
19
20      // Đăng ký các dịch vụ với DI Container
        1 reference | 0 changes | 0 authors, 0 changes
21      public static WebApplicationBuilder ConfigureServices(
22          this WebApplicationBuilder builder)
23      {
24          builder.Services.AddDbContext<BlogDbContext>(options =>
25              options.UseSqlServer(
26                  builder.Configuration
27                      .GetConnectionString("DefaultConnection")));
28
29          builder.Services.AddScoped<IBlogRepository, BlogRepository>();
30          builder.Services.AddScoped<IDataSeeder, DataSeeder>();
31
32          return builder;
33      }
34
35      // Cấu hình HTTP Request pipeline
        1 reference | 0 changes | 0 authors, 0 changes
36      public static WebApplication UseRequestPipeline(
37          this WebApplication app)
38      {
39          // Thêm middleware để hiển thị thông báo lỗi
40          if (app.Environment.IsDevelopment())
41          {
42              app.UseDeveloperExceptionPage();
43          }
44          else
45          {
46              app.UseExceptionHandler("/Blog/Error");
47
48              // Thêm middleware cho việc áp dụng HSTS (thêm header
49              // Strict-Transport-Security vào HTTP Response).
50              app.UseHsts();
51          }
52
53          // Thêm middleware để tự động nén HTTP response
54          app.UseResponseCompression();
55
56          // Thêm middleware để chuyển hướng HTTP sang HTTPS
57          app.UseHttpsRedirection();
58
59          // Thêm middleware phục vụ các yêu cầu liên quan
```

```
60        // tới các tập tin nội dung tĩnh như hình ảnh, css, ...
61        app.UseStaticFiles();
62
63        // Thêm middleware lựa chọn endpoint phù hợp nhất
64        // để xử lý một HTTP request.
65        app.UseRouting();
66
67        return app;
68    }
```

```
70    // Thêm dữ liệu mẫu vào CSDL
      1 reference | 0 changes | 0 authors, 0 changes
71    public static IApplicationBuilder UseDataSeeder(
72        this IApplicationBuilder app)
73    {
74        using var scope = app.ApplicationServices.CreateScope();
75
76        try
77        {
78            scope.ServiceProvider
79                .GetRequiredService<IDataSeeder>()
80                .Initialize();
81        }
82        catch (Exception ex)
83        {
84            scope.ServiceProvider
85                .GetRequiredService<ILogger<Program>>()
86                .LogError(ex, "Could not insert data into database");
87        }
88
89        return app;
90    }
```

Trong lớp **RouteExtensions**, cài đặt phương thức định nghĩa các routes cần thiết cho ứng dụng blog.

```
5     // Định nghĩa route template, route constraint cho các
6     // endpoints kết hợp với các action trong các controller.
      1 reference | 0 changes | 0 authors, 0 changes
7     public static IEndpointRouteBuilder UseBlogRoutes(
8         this IEndpointRouteBuilder endpoints)
9     {
10        endpoints.MapControllerRoute(
11            name: "posts-by-category",
12            pattern: "blog/category/{slug}",
13            defaults: new { controller = "Blog", action = "Category" });
14
15        endpoints.MapControllerRoute(
16            name: "posts-by-tag",
17            pattern: "blog/tag/{slug}",
18            defaults: new { controller = "Blog", action = "Tag" });
```

```
20        endpoints.MapControllerRoute(
21            name: "single-post",
22            pattern: "blog/post/{year:int}/{month:int}/{day:int}/{slug}",
23            defaults: new { controller = "Blog", action = "Post" });
24
25        endpoints.MapControllerRoute(
26            name: "default",
27            pattern: "{controller=Blog}/{action=Index}/{id?}");
28
29        return endpoints;
30    }
```

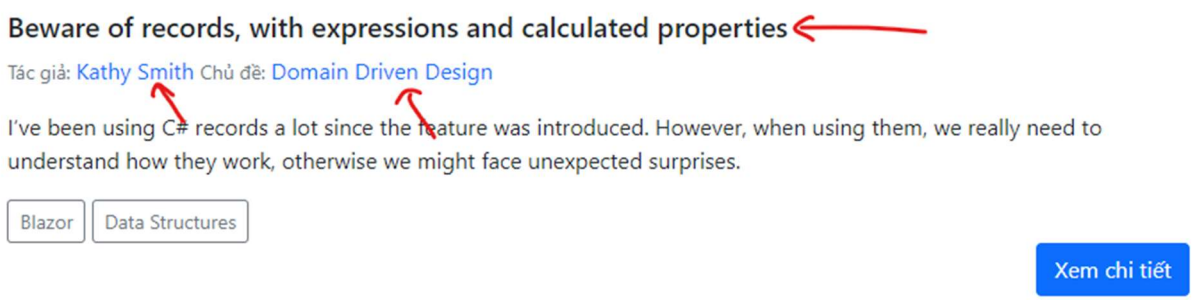
Cập nhật lại mã nguồn trong tập tin **Program.cs** để gọi tới các phương thức vừa tạo.

```
1  using TatBlog.WebApp.Extensions;
2
3  var builder = WebApplication.CreateBuilder(args);
4  {
5      builder
6          .ConfigureMvc()
7          .ConfigureServices();
8  }
9
10 var app = builder.Build();
11 {
12     app.UseRequestPipeline();
13     app.UseBlogRoutes();
14     app.UseDataSeeder();
15 }
16
17 app.Run();
```

Chạy chương trình và kiểm tra kết quả.

C. Bài tập thực hành

1. Hãy thay đổi hoặc bổ sung mã nguồn trong tập tin **_PostItem.cshtml** để hiển thị tiêu đề bài viết, tên tác giả và tên chuyên mục dưới dạng liên kết. Các liên kết này có dạng:
   - https://localhost:7248/blog/author/kathy-smith
   - https://localhost:7248/blog/category/domain-driven-design
   - https://localhost:7248/blog/post/2023/2/1/blog-post-title



2. **Cài đặt các action trong lớp BlogController và các view tương ứng.**
   - **Category(string slug)**: để hiển thị danh sách bài viết thuộc chủ đề (do người dùng chọn trong phần sidebar hoặc trong mỗi bài viết).
   - **Author(string slug)**: để hiển thị danh sách bài viết theo tác giả (do người dùng chọn trong liên kết ở dưới phần tiêu đề bài viết).
   - **Tag(string slug)**: để hiển thị danh sách bài viết chứa thẻ (tag do người dùng chọn trong phần tag cloud ở sidebar hoặc trong mỗi bài viết).
   - **Post(int year, int month, int day, string slug)**: để hiển thị chi tiết một bài viết khi người dùng nhấn vào nút **Xem chi tiết** hoặc tiêu đề bài viết ở trang chủ. Khi người dùng xem chi tiết một bài viết, tăng số lượt xem của bài viết đó lên 1 lượt. Nếu trạng thái **IsPublished** của bài viết là **False** thì hiển thị thông báo lỗi.
   - **Archives(int year, int month)**: để hiển thị danh sách bài viết được đăng trong tháng và năm đã chọn (do người dùng click chuột vào các tháng trong view component Archives ở bài tập 3).
   - **Contact()**: để hiển thị thông tin liên hệ, bản đồ và form để gửi ý kiến.
   - **About()**: để hiển thị trang giới thiệu về blog (nội dung tĩnh).

3. **Tạo các view component và thêm chúng vào sidebar:**
   - **FeaturedPosts**: Hiển thị TOP 3 bài viết được xem nhiều nhất. Người dùng có thể click chuột để xem chi tiết.
   - **RandomPosts**: Hiển thị TOP 5 bài viết ngẫu nhiên. Người dùng có thể click chuột để xem chi tiết.
   - **TagCloud**: Hiển thị danh sách các thẻ (tag). Khi người dùng click chuột vào thẻ nào thì hiển thị danh sách bài viết chứa thẻ đó.
   - **BestAuthors**: Hiển thị TOP 4 tác giả có nhiều bài viết nhất. Sử dụng view component này trên trang chủ. Khi người dùng click chuột vào tên tác giả, hiển thị danh sách bài viết của tác giả đó.
   - **Archives**: Hiển thị danh sách 12 tháng gần nhất và số lượng bài viết trong mỗi tháng dưới dạng các liên kết. Khi người dùng click chuột vào tháng nào thì hiển thị danh sách bài viết được đăng trong tháng đó. Định dạng: November 2022 (5), February 2023 (11), …

PHUONV - DO NOT COPY

4. **Xây dựng chức năng đăng ký nhận thông báo khi có bài viết mới**
   - Tạo partial view **NewsletterForm** cho phép người dùng nhập địa chỉ email để đăng ký nhận thông báo khi có bài viết mới.
   - Tạo controller mới, đặt tên là **NewsletterController**.
   - Tạo 2 action **Subscribe(string email), Unsubscribe(string email)** để thực hiện việc đăng ký và hủy đăng ký nhận thông báo bài viết mới.
   - Khi người dùng nhấn nút submit (**Subscribe**), địa chỉ email được gửi tới action **Subscribe**, sử dụng các phương thức đã cài đặt trong bài tập 3 ở Lab 01 để lưu lại thông tin. Ngoài ra, hệ thống sẽ gửi 1 email tới địa chỉ của người dùng, trong đó có nội dung cảm ơn vì đã đăng ký và một liên kết (URL) tới action **Ubsubscribe** để người dùng có thể hủy đăng ký bất cứ lúc nào.

5. **Xây dựng chức năng thảo luận, đánh giá bài viết**
   - Trong trang xem chi tiết bài viết, hiển thị form cho phép người dùng nhập họ tên và nội dung bình luận.
   - Những bình luận đã được kiểm duyệt sẽ được hiển thị ngay bên dưới form bình luận.
   - Khi người dùng nhập nội dung bình luận và nhấn nút Gửi, hệ thống lưu lại nội dung đó. Tuy nhiên, mọi bình luận đều phải được người quản trị kiểm duyệt trước khi hiển thị lên danh sách để mọi người dùng khác đều thấy.
   - Sử dụng kết quả trong bài tập 4 – Lab 01 để làm bài tập này.

6. Ở trang liên hệ, khi người dùng nhập thông tin (Họ tên, địa chỉ email, nội dung góp ý, …) rồi nhấn nút Gửi, hệ thống cần gửi 1 email tới người quản trị hệ thống để thông báo về ý kiến của người dùng. Địa chỉ email của người quản trị được và các thông tin cài đặt sử dụng cho việc gửi mail được lưu trữ trong tập tin appsettings.json.

7. Xây dựng chức năng chia sẻ bài viết lên các mạng xã hội như facebook, twitter, …

8. Hãy thiết kế hoặc tìm một template đẹp hơn, phù hợp cho blog và áp dụng template đó cho ứng dụng TatBlog của bạn.

--- HẾT ---

PHUONV - DO NOT COPY