

## LAB 3. TAT-BLOG: CHỨC NĂNG CHO NGƯỜI QUẢN TRỊ

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 người quản trị của một trang blog đơn giản. Giao diện dành cho người quản lý blog được phác thảo như hình sau.

Tips & Tricks Chu đề Tác giả Thẻ Bài viết Bình luận

### Danh sách bài viết

Nhập từ khóa...	-- Chọn tác giả --	-- Chọn chủ đề --
Nhập năm...	-- Chọn tháng --	<button>Tìm/Lọc</button> <button>Thêm mới</button>

Tiêu đề	Tác giả	Chủ đề	Trạng thái
<a href="#">Test 2</a> Test	Jason Mouth	test	True
<a href="#">Beware of records, with expressions and calculated properties Updated</a>  I've been using C# records a lot since the feature was introduced. However, when using them, we really need to understand how they work, otherwise we might face unexpected surprises.	Kathy Smith	Domain Driven Design	True

Tips & Tricks Chu đề Tác giả Thẻ Bài viết Bình luận

### Thêm/Cập nhật bài viết

Tiêu đề	Beware of records, with expressions and calculated properties Updated
Giới thiệu	I've been using C# records a lot since the feature was introduced. However, when using them, we really need to understand how they work, otherwise we might face
Nội dung	Given their terseness, even ignoring the other records characteristics, as soon as I'm creating a type that I'm expecting to be immutable, I immediately start with a record. However this might not always be adequate, and in this post we'll look at a very specific example where I introduced a bug in the code due to not taking into consideration all of the records characteristics. This solves the problem, and might be a valid solution in general. It has one potential issue though: every time we use SomeCalculatedValue, like the method that the property get accessor is, the value will be calculated. Depending on the way the property is used, it might not be a problem, or it might be, as we're always repeating the same logic and creating new objects to return. In my case, I was using the property multiple times, and the

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

- Cách tạo MVC Area và các thành phần M/V/C để xây dựng các chức năng cho một phân hệ riêng của ứng dụng.
- Cách xây dựng các chức năng thêm/cập nhật/xóa dữ liệu.
- Cách kiểm tra tính hợp lệ của dữ liệu sử dụng các Attribute có sẵn và sử dụng gói thư viện FluentValidation.
- Cách sử dụng gói thư viện Mapster để đơn giản hóa việc sao chép dữ liệu giữa các đối tượng.
- Cách cấu hình và ghi lại nhật ký hệ thống sử dụng gói thư viện NLog.

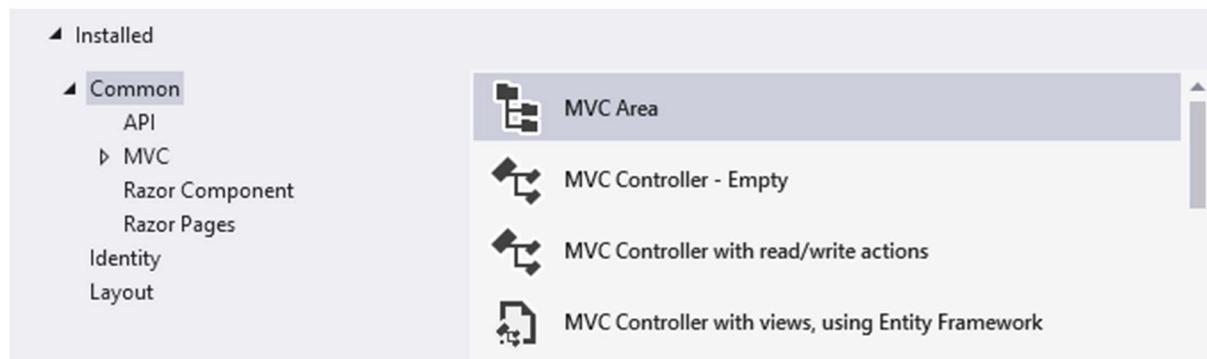
**Yêu cầu: Sinh viên tự làm phần “B. Hướng dẫn thực hành” ở nhà và nộp lên hệ thống LMS. Tại phòng Lab, sinh viên làm phần “C. Bài tập thực hành” dựa trên dự án đã hoàn thành ở phần B.**

## B. Hướng dẫn thực hành

### 1. Tạo Area cho phân hệ dành cho người quản trị

Nhấp phải chuột vào dự án TatBlog.WebApp, chọn Add > New Scaffolded Item. Trong cửa sổ mới, chọn **MVC Area**.

#### Add New Scaffolded Item

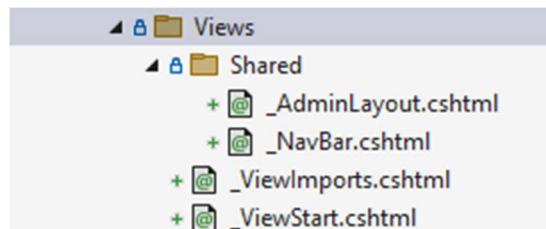


Đặt tên cho phân vùng mới là **Admin**. Nhấn nút **Add** và chờ chương trình tạo cấu trúc thư mục.

#### Add MVC Area

Area name	Admin
Add	Cancel

Trong thư mục **Areas > Admin**, xóa thư mục **Data**. Trong thư mục **Areas > Admin > Views**, tạo thư mục **Shared**. Sau đó, tạo các tập tin như hình sau (sinh viên có thể copy các tập tin từ thư mục **Views** của dự án trước và cập nhật lại cho phù hợp).



Cập nhật mã nguồn trong tập tin **\_AdminLayout.cshtml** để tạo bố cục cho trang quản trị.

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="utf-8" />
5      <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6      <title>@ViewData["PageTitle"] - Tips & Tricks Blog</title>
7
8      <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
9      <link rel="stylesheet" href="~/lib/font-awesome/css/font-awesome.min.css" />
10     <link rel="stylesheet" href="~/css/main.css" asp-append-version="true" />
11
12     @await RenderSectionAsync("styles", required: false)
13 </head>
14 <body>
15     
16     <partial name="_NavBar" />
17
18     <div class="container-fluid">
19         
20         @RenderBody()
21     </div>
22
23     <footer class="border-top footer text-muted">
24         <div class="container-fluid text-center">
25             © 2023 - Tips & Tricks Blog
26         </div>
27     </footer>
28
29     <script src="~/lib/jquery/dist/jquery.min.js"></script>
30     <script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
31     <script src="~/js/main.js" asp-append-version="true"></script>
32
33     @await RenderSectionAsync("scripts", required: false)
34 </body>
35 </html>
```

Thiết lập bố cục mặc định cho phần quản trị là **\_AdminLayout** bằng cách cập nhật mã trong tập tin **\_ViewStart.cshtml**.

```
_ViewStart.cshtml
1 @{
2     Layout = "_AdminLayout";
3 }
```

Tạo các menu trên thanh tiêu đề bằng cách cập nhật mã nguồn trong tập tin `_NavBar.cshtml`.

```
1 <header>
2   <nav class="navbar navbar-expand-sm
3       navbar-toggleable-sm navbar-light
4       bg-white border-bottom box-shadow">
5     <div class="container-fluid">
6       <a class="navbar-brand"
7           asp-area="Admin"
8           asp-controller="Dashboard"
9           asp-action="Index">
10          Tips & Tricks
11        </a>
12
13       <button class="navbar-toggler"
14           type="button"
15           data-bs-toggle="collapse"
16           data-bs-target=".navbar-collapse"
17           aria-controls="navbarSupportedContent"
18           aria-expanded="false"
19           aria-label="Toggle navigation">
20         <span class="navbar-toggler-icon"></span>
21       </button>
22
23       <div class="navbar-collapse collapse
24           d-sm-inline-flex
25           justify-content-between">
26         <ul class="navbar-nav flex-grow-1">
27           <li class="nav-item">
28             <a class="nav-link text-dark"
29                 asp-area="Admin"
30                 asp-controller="Categories"
31                 asp-action="Index"
32                 title="Xem danh sách chủ đề">
33               Chủ đề
34             </a>
35           </li>
36           <li class="nav-item">
37             <a class="nav-link text-dark"
38                 asp-area="Admin"
39                 asp-controller="Authors"
40                 asp-action="Index"
41                 title="Xem danh sách tác giả">
42               Tác giả
43             </a>
44           </li>
```

```

45   <li class="nav-item">
46     <a class="nav-link text-dark"
47       asp-area="Admin"
48       asp-controller="Tags"
49       asp-action="Index"
50       title="Xem danh sách thẻ/từ khóa">
51       Thẻ
52     </a>
53   </li>
54   <li class="nav-item">
55     <a class="nav-link text-dark"
56       asp-area="Admin"
57       asp-controller="Posts"
58       asp-action="Index"
59       title="Xem danh sách bài viết">
60       Bài viết
61     </a>
62   </li>
63   <li class="nav-item">
64     <a class="nav-link text-dark"
65       asp-area="Admin"
66       asp-controller="Comments"
67       asp-action="Index"
68       title="Xem danh sách bình luận">
69       Bình luận
70     </a>
71   </li>
72 </ul>
73 </div>
74 </div>
75 </nav>
76 </header>

```

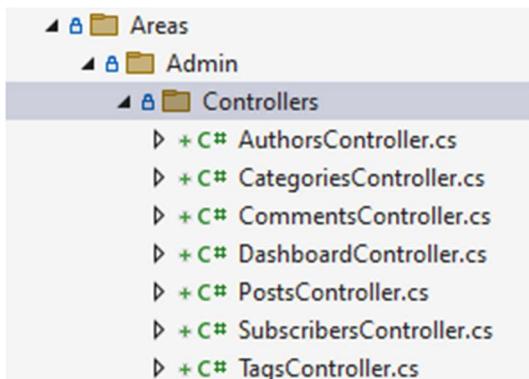
Trong thư mục **Areas > Admin > Controllers**, tạo lớp **DashboardController** và định nghĩa phương thức (action) **Index** như hình sau.

```

1  using Microsoft.AspNetCore.Mvc;
2
3  namespace TatBlog.WebApp.Areas.Admin.Controllers;
4
5  public class DashboardController : Controller
6  {
7    public IActionResult Index()
8    {
9      return View();
10 }
11 }

```

Tương tự như trên, hãy tạo các lớp **CategoriesController**, **AuthorsController**, **PostsController**, **TagsController**, **CommentsController** và **SubscribersController**. Cấu trúc thư mục **Controllers** sau khi tạo các lớp:



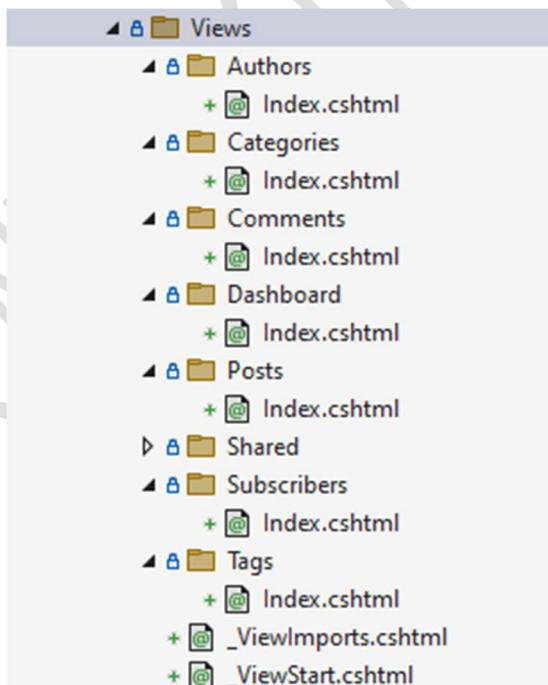
Trong thư mục **Areas > Admin > Views**, tạo thư mục **Dashboard**. Sau đó tạo view **Index.cshtml**. Nhập đoạn mã sau vào view vừa tạo.

```

1  @{
2      ViewData["PageTitle"] = "Bảng điều khiển";
3  }
4
5  <h1>
6      Đây là khu vực dành cho người quản trị
7  </h1>

```

Tương tự, hãy tạo các thư mục chứa view templates tương ứng với các controller còn lại. Cấu trúc thư mục **Views** sau khi tạo các tập tin:



Định nghĩa route template để ánh xạ các yêu cầu truy cập trang quản trị đến các action trong các controller của phân vùng **Admin** bằng cách cập nhật phương thức **UseBlogRoutes** trong class **Extensions > RouteExtensions**.

```

endpoints.MapControllerRoute(
    name: "single-post",
    pattern: "blog/post/{year:int}/{month:int}/{day:int}/{slug}",
    defaults: new { controller = "Blog", action = "Post" });

endpoints.MapControllerRoute(
    name: "admin-area",
    pattern: "admin/{controller=Dashboard}/{action=Index}/{id?}",
    defaults: new { area = "Admin" });

endpoints.MapControllerRoute(
    name: "default",
    pattern: "[controller=Blog]/[action=Index]/[id?]");

```

Nhấn tổ hợp phím **Ctrl + Shift + B** hoặc menu **Build > Build Solution** để biên dịch, kiểm tra và sửa lỗi (nếu có). Chạy chương trình, nhấn chuột vào các menu và kiểm tra kết quả.

## Đây là khu vực dành cho người quản trị

© 2023 - Tips & Tricks Blog

## Danh sách bài viết

© 2023 - Tips & Tricks Blog

## Danh sách chủ đề

© 2023 - Tips & Tricks Blog

## Danh sách tác giả

© 2023 - Tips & Tricks Blog

## 2. Xây dựng các chức năng xem danh sách & tìm bài viết

Trong thư mục **Areas > Admin > Models**, tạo lớp **PostFilterModel**. Cài đặt các thuộc tính và phương thức cho lớp này như sau.

```

7 3 references | 0 changes | 0 authors, 0 changes
8  public class PostFilterModel
9  {
10     [DisplayName("Từ khóa")]
11     1 reference | 0 changes | 0 authors, 0 changes
12     public string Keyword { get; set; }
13
14     [DisplayName("Tác giả")]
15     1 reference | 0 changes | 0 authors, 0 changes
16     public int? AuthorId { get; set; }
17
18     [DisplayName("Chủ đề")]
19     1 reference | 0 changes | 0 authors, 0 changes
20     public int? CategoryId { get; set; }
21
22     [DisplayName("Năm")]
23     1 reference | 0 changes | 0 authors, 0 changes
24     public int? Year { get; set; }
25
26     [DisplayName("Tháng")]
27     1 reference | 0 changes | 0 authors, 0 changes
28     public int? Month { get; set; }
29
30
31     1 reference | 0 changes | 0 authors, 0 changes
32     public IEnumerable<SelectListItem> AuthorList { get; set; }
33
34     1 reference | 0 changes | 0 authors, 0 changes
35     public IEnumerable<SelectListItem> CategoryList { get; set; }
36
37     1 reference | 0 changes | 0 authors, 0 changes
38     public IEnumerable<SelectListItem> MonthList { get; set; }
39
40
41     0 references | 0 changes | 0 authors, 0 changes
42     public PostFilterModel()
{
```

- 33     MonthList = Enumerable.Range(1, 12)
  - 34         .Select(m => new SelectListItem()
    - 35             Value = m.ToString(),
 Text = CultureInfo.CurrentCulture
 .DateTimeFormat.GetMonthName(m)
    - 36         ).ToList();

Trong lớp **PostsController**, tạo biến thành viên và định nghĩa phương thức khởi tạo để khởi gán giá trị cho các biến.

```
11     private readonly IBlogRepository _blogRepository;  
12  
13     public PostsController(IBlogRepository blogRepository)  
14     {  
15         _blogRepository = blogRepository;  
16     }
```

Cũng trong lớp **PostsController**, định nghĩa phương thức dưới đây để gán giá trị cho các thuộc tính của một đối tượng **PostFilterModel**.

```
1 reference | 0 changes | 0 authors, 0 changes  
37     private async Task PopulatePostFilterModelAsync(PostFilterModel model)  
38     {  
39         var authors = await _blogRepository.GetAuthorsAsync();  
40         var categories = await _blogRepository.GetCategoriesAsync();  
41  
42         model.AuthorList = authors.Select(a => new SelectListItem()  
43         {  
44             Text = a.FullName,  
45             Value = a.Id.ToString()  
46         });  
47  
48         model.CategoryList = categories.Select(c => new SelectListItem()  
49         {  
50             Text = c.Name,  
51             Value = c.Id.ToString()  
52         });  
53     }
```

Sau đó cài đặt phương thức **Index** của lớp **PostsController**.

```
0 references | 0 changes | 0 authors, 0 changes  
18     public async Task<IActionResult> Index(PostFilterModel model)  
19     {  
20         var postQuery = new PostQuery()  
21         {  
22             Keyword = model.Keyword,  
23             CategoryId = model.CategoryId,  
24             AuthorId = model.AuthorId,  
25             Year = model.Year,  
26             Month = model.Month  
27         };  
28  
29         ViewBag.PostsList = await _blogRepository  
30             .GetPagedPostsAsync(postQuery, 1, 10);  
31  
32         await PopulatePostFilterModelAsync(model);  
33  
34         return View(model);  
35     }
```

Để hiển thị danh sách bài viết trả về từ phương thức **Index**, cập nhật mã nguồn trong tập tin **Views > Posts > Index.cshtml** như sau.

```
1  @model TatBlog.WebApp.Areas.Admin.Models.PostFilterModel
2  @{
3      ViewData["PageTitle"] = "Danh sách bài viết";
4
5      var postsList = ViewBag.PostsList as IPagedList<Post>;
6  }
7
8  <h1>
9      Danh sách bài viết
10 </h1>
11
12
13 @if (postsList != null && postsList.Count > 0)
14 {
15     <table class="table table-responsive table-striped">
16         <thead>
17             <tr>
18                 <th>Tiêu đề</th>
19                 <th>Tác giả</th>
20                 <th>Chủ đề</th>
21                 <th>Xuất bản</th>
22             </tr>
23         </thead>
24         <tbody>
25             @foreach (var post in postsList)
26             {
27                 <tr>
28                     <td>
29                         <a asp-area="Admin"
30                             asp-controller="Posts"
31                             asp-action="Edit"
32                             asp-route-id="@post.Id"
33                             class="text-bold">
34                             @post.Title
35                         </a>
36
37                         <p class="text-muted">
38                             @post.ShortDescription
39                         </p>
40                     </td>
41                     <td>
42                         @post.Author.FullName
43                     </td>
44                     <td>
45                         @post.Category.Name
46                     </td>
47                     <td>
48                         @(post.Published ? "Có" : "Không")
49                     </td>
50                 </tr>
51             }
52 }
```

```

52         </tbody>
53     </table>
54 }
55 else {
56 {
57     <h1 class="text-danger">
58         Không tìm thấy bài viết nào
59     </h1>
60 }

```

Biên dịch chương trình, kiểm tra và sửa lỗi (nếu có). Chạy chương trình, nhấn vào menu “**Bài viết**” để xem kết quả.

Tips & Tricks Chu đề Tác giả Thẻ Bài viết Bình luận

## Danh sách bài viết

Tiêu đề	Tác giả	Chủ đề	Xuất bản
<a href="#">6 Productivity Shortcuts on Windows 10 &amp; 11</a> I don't know about you, but I'm obsessed with shortcuts. I'm much more productive when using just the keyboard, and having to use the mouse annoys me deeply. Over the years, I've learned many useful shortcuts that increase productivity. Many of them are for IDEs or other apps, but some of the best shortcuts are part of the operating system itself. Today we'll cover 6 amazing shortcuts in Windows 10 and 11 that transformed the way I work and can make you much more productive. Some of them are more known than others, but all are brilliant.	Jessica Wonder	OOP	Có
<a href="#">Papilio: An Intro</a> Flutter gives you a powerful toolset for building rich cross-platform apps. You can build single-source apps on macOS, Windows or Linux and run those apps	Jason Mouth	Azure	Có

Tiếp theo, ta sẽ xây dựng chức năng tìm/ lọc các bài viết theo từ khóa, tác giả, chủ đề, ...

Trong thư mục **Views > Posts**, tạo một partial view **\_PostFilterPane.cshtml**. Nhập đoạn mã sau để tạo form tìm kiếm.

```

1 @model TatBlog.WebApp.Areas.Admin.Models.PostFilterModel
2
3 <form asp-area="Admin"
4     asp-controller="Posts"
5     asp-action="Index"
6     method="get"
7     class="row gy-2 gx-3 align-items-center">
8

```

```
9   <div class="col-auto">
10    <label class="visually-hidden" asp-for="Keyword"></label>
11    <input type="text"
12       class="form-control"
13       asp-for="Keyword"
14       placeholder="Nhập từ khóa... ">
15  </div>
16
17  <div class="col-auto">
18    <label class="visually-hidden" asp-for="AuthorId"></label>
19    <select class="form-select"
20       asp-for="AuthorId"
21       asp-items="Model.AuthorList">
22      <option value="" selected>-- Chọn tác giả --</option>
23    </select>
24  </div>
25
26  <div class="col-auto">
27    <label class="visually-hidden" asp-for="CategoryId"></label>
28    <select class="form-select"
29       asp-for="CategoryId"
30       asp-items="Model.CategoryList">
31      <option value="" selected>-- Chọn chủ đề --</option>
32    </select>
33  </div>
34
35  <div class="col-auto">
36    <label class="visually-hidden" asp-for="Year"></label>
37    <input type="text"
38       class="form-control"
39       asp-for="Year"
40       placeholder="Nhập năm... ">
41  </div>
42
43  <div class="col-auto">
44    <label class="visually-hidden" asp-for="Month"></label>
45    <select class="form-select"
46       asp-for="Month"
47       asp-items="Model.MonthList">
48      <option value="" selected>-- Chọn tháng --</option>
49    </select>
50  </div>
51
52  <div class="col-auto">
53    <button type="submit" class="btn btn-primary">Tim/Lọc</button>
54
55  <a asp-area="Admin"
56    asp-controller="Posts"
57    asp-action="Edit"
58    class="btn btn-success">
59    Thêm mới
60  </a>
61 </div>
62 </form>
```

Bổ sung thêm mã lệnh vào tập tin **Posts > Index.cshtml** để hiển thị Form tìm kiếm.

```

8 8 <h1>
9   Danh sách bài viết
10  </h1>
11
12  <!-- Hiển thị Form để tìm bài viết -->
13  <partial name="_PostFilterPane" model="Model"/>
14
15 @if (postsList != null && postsList.Count > 0)

```

Chạy chương trình và kiểm tra kết quả.

Tips & Tricks Chu đề Tác giả Thẻ Bài viết Bình luận

## Danh sách bài viết

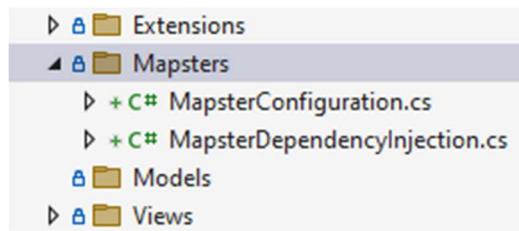
<input type="text" value="windows"/>	<input type="button" value="-- Chọn tác giả --"/>	<input type="button" value="-- Chọn chủ đề --"/>	
<input type="text" value="Nhập năm..."/>	<input type="button" value="-- Chọn tháng --"/>	<input type="button" value="Tim/Lọc"/> <input type="button" value="Thêm mới"/>	
<hr/>			
Tiêu đề	Tác giả	Chủ đề	Xuất bản
<a href="#">6 Productivity Shortcuts on Windows 10 &amp; 11</a> I don't know about you, but I'm obsessed with shortcuts. I'm much more productive when using just the keyboard, and having to use the mouse annoys me deeply. Over the years, I've learned many useful shortcuts that increase productivity. Many of them are for IDEs or other apps, but some of the best shortcuts are part of the operating system itself. Today we'll cover 6 amazing shortcuts in Windows 10 and 11 that transformed the way I work and can make you much more productive. Some of them are more known than others, but all are brilliant.	Jessica Wonder	OOP	Có

### 3. Sử dụng gói thư viện Mapster

Trong phần này, ta sẽ cài đặt gói thư viện **Mapster** và thiết lập các tùy chỉnh cần thiết để sử dụng **Mapster** trong việc copy dữ liệu giữa các đối tượng.

Trước tiên, cài đặt gói thư viện **Mapster.DependencyInjection** vào dự án **TatBlog.WebApp**. Khi cài đặt gói này, hệ thống sẽ tự cài đặt gói phụ thuộc là **Mapster**, chúng ta không cần phải cài đặt riêng 2 gói thư viện. (Sinh viên xem lại Lab 01 để biết cách cài đặt NuGet packages).

Tiếp theo, tạo thư mục **Mapsters** trong thư mục gốc của dự án **TatBlog.WebApp**. Tạo 2 lớp **MapsterConfiguration** và **MapsterDependencyInjection** trong thư mục Mapsters.



Định nghĩa lớp **MapsterConfiguration** để thiết lập việc sao chép dữ liệu giữa các đối tượng như sau:

```

8 1 reference | 0 changes | 0 authors, 0 changes
8  public class MapsterConfiguration : IRegister
9  {
10  0 references | 0 changes | 0 authors, 0 changes
10  public void Register(TypeAdapterConfig config)
11  {
12      config.NewConfig<Post, PostItem>()
13          .Map(dest => dest.CategoryName, src => src.Category.Name)
14          .Map(dest => dest.Tags, src => src.Tags.Select(x => x.Name));
15
16      config.NewConfig<PostFilterModel, PostQuery>()
17          .Map(dest => dest.PublishedOnly, src => false);
18  }
19

```

Trong lớp tĩnh **MapsterDependencyInjection**, định nghĩa phương thức **ConfigureMapster** để thêm các dịch vụ cần thiết của Mapster vào DI Container.

```

6 0 references | 0 changes | 0 authors, 0 changes
6  public static class MapsterDependencyInjection
7  {
8  1 reference | 0 changes | 0 authors, 0 changes
8  public static WebApplicationBuilder ConfigureMapster(
9      this WebApplicationBuilder builder)
10 {
11     var config = TypeAdapterConfig.GlobalSettings;
12     config.Scan(typeof(MapsterConfiguration).Assembly);
13
14     builder.Services.AddSingleton(config);
15     builder.Services.AddScoped<IMapper, ServiceMapper>();
16
17     return builder;
18 }
19

```

Trong **Program.cs**, gọi phương thức **ConfigureMapster** từ đối tượng **builder** (WebApplicationBuilder).

```
4  var builder = WebApplication.CreateBuilder(args);
5  {
6      builder
7          .ConfigureMvc()
8          .ConfigureServices()
9          .ConfigureMapster();
10 }
```

Cập nhật lại mã nguồn trong lớp **PostsController** và action **Index** để sử dụng **Mapster** cho việc copy dữ liệu.

```
1 reference | 0 changes | 0 authors, 0 changes
10 public class PostsController : Controller
11 {
12     private readonly IBlogRepository _blogRepository;
13     private readonly IMapper _mapper;
14
15     public PostsController(
16         IBlogRepository blogRepository,
17         IMapper mapper)
18     {
19         _blogRepository = blogRepository;
20         _mapper = mapper;
21     }
22
23     public async Task<IActionResult> Index(PostFilterModel model)
24     {
25         // Sử dụng Mapster để tạo đối tượng PostQuery
26         // từ đối tượng PostFilterModel model
27         var postQuery = _mapper.Map<PostQuery>(model);
28
29         ViewBag.PostsList = await _blogRepository
30             .GetPagedPostsAsync(postQuery, 1, 10);
31
32         await PopulatePostFilterModelAsync(model);
33
34         return View(model);
35     }
}
```

Biên dịch, kiểm tra và sửa lỗi (nếu có). Chạy chương trình và so sánh kết quả ở phần 2.

Sinh viên tìm hiểu thêm về cách sử dụng thư viện Mapster tại đây:

- <https://github.com/MapsterMapper/Mapster/wiki/>
- <https://code-maze.com/mapster-aspnetcore-introduction/>

## 4. Xây dựng chức năng thêm/cập nhật bài viết

Trong phần này, ta sẽ xây dựng chức năng thêm mới hoặc cập nhật một bài viết cho trước. Trước hết, ta cần một view model để hiển thị Form cho phép người dùng nhập dữ liệu và sau đó nhận dữ liệu nhập bởi người dùng.

Trong thư mục **Areas > Admin > Models**, tạo một lớp mới, đặt tên là **PostEditModel**.

```
9 references | 0 changes | 0 authors, 0 changes
8  public class PostEditModel
9  {
10     public int Id { get; set; }
11
12     [DisplayName("Tiêu đề")]
13     [Required(ErrorMessage = "Tiêu đề không được để trống")]
14     [MaxLength(500, ErrorMessage = "Tiêu đề tối đa 500 ký tự")]
15     public string Title { get; set; }
16
17     [DisplayName("Giới thiệu")]
18     [Required(ErrorMessage = "Giới thiệu không được để trống")]
19     [MaxLength(2000, ErrorMessage = "Giới thiệu tối đa 2000 ký tự")]
20     public string ShortDescription { get; set; }
21
22     [DisplayName("Nội dung")]
23     [Required(ErrorMessage = "Nội dung không được để trống")]
24     [MaxLength(5000, ErrorMessage = "Nội dung tối đa 5000 ký tự")]
25     public string Description { get; set; }
26
27     [DisplayName("Metadata")]
28     [Required(ErrorMessage = "Metadata không được để trống")]
29     [MaxLength(1000, ErrorMessage = "Metadata tối đa 1000 ký tự")]
30     public string Meta { get; set; }
31
32     [DisplayName("Slug")]
33     [Remote("VerifyPostSlug", "Posts", "Admin",
34         HttpMethod = "POST", AdditionalFields = "Id")]
35     [Required(ErrorMessage = "URL slug không được để trống")]
36     [MaxLength(200, ErrorMessage = "Slug tối đa 200 ký tự")]
37     public string UrlSlug { get; set; }
38
39     [DisplayName("Chọn hình ảnh")]
40     public IFormFile ImageFile { get; set; }
41
42     [DisplayName("Hình hiện tại")]
43     public string ImageUrl { get; set; }
```

```

44 [DisplayName("Xuất bản ngay")]
45 0 references | 0 changes | 0 authors, 0 changes
46 public bool Published { get; set; }
47
48 [DisplayName("Chủ đề")]
49 [Required(ErrorMessage = "Bạn chưa chọn chủ đề")]
50 0 references | 0 changes | 0 authors, 0 changes
51 public int CategoryId { get; set; }
52
53 [DisplayName("Tác giả")]
54 [Required(ErrorMessage = "Bạn chưa chọn tác giả")]
55 0 references | 0 changes | 0 authors, 0 changes
56 public int AuthorId { get; set; }
57
58 [DisplayName("Từ khóa (mỗi từ 1 dòng)")]
59 [Required(ErrorMessage = "Bạn chưa nhập tên thẻ")]
60 1 reference | 0 changes | 0 authors, 0 changes
61 public string SelectedTags { get; set; }
62
63
64 // Tách chuỗi chứa các thẻ thành một mảng các chuỗi
65 0 references | 0 changes | 0 authors, 0 changes
66 public List<string> GetSelectedTags()
67 {
68     return (SelectedTags ?? "")
69         .Split(new[] { ',', ';', '\r', '\n' },
70               StringSplitOptions.RemoveEmptyEntries)
71         .ToList();
72 }
73

```

Thiết lập cách sao chép dữ liệu giữa 2 đối tượng **Post** và **PostEditModel** trong phương thức **Register** của lớp **MapsterConfiguration**.

```

19 config.NewConfig<PostEditModel, Post>()
20     .Ignore(dest => dest.Id)
21     .Ignore(dest => dest.ImageUrl);
22
23 config.NewConfig<Post, PostEditModel>()
24     .Map(dest => dest.SelectedTags, src =>
25         string.Join("\r\n", src.Tags.Select(x => x.Name)))
26     .Ignore(dest => dest.CategoryList)
27     .Ignore(dest => dest.AuthorList)
28     .Ignore(dest => dest.ImageFile);

```

Trong lớp **PostsController** định nghĩa thêm phương thức **Edit** để xử lý yêu cầu thêm mới hoặc cập nhật một bài viết có mã số (ID) cho trước.

```
37 [HttpGet]
38 0 references | 0 changes | 0 authors, 0 changes
39 public async Task<IActionResult> Edit(int id = 0)
40 {
41     // ID = 0 <=> Thêm bài viết mới
42     // ID > 0 : Đọc dữ liệu của bài viết từ CSDL
43     var post = id > 0
44         ? await _blogRepository.GetPostByIdAsync(id, true)
45         : null;
46
47     // Tạo view model từ dữ liệu của bài viết
48     var model = post == null
49         ? new PostEditModel()
50         : _mapper.Map<PostEditModel>(post);
51
52     // Gán các giá trị khác cho view model
53     await PopulatePostEditModelAsync(model);
54
55     return View(model);
56 }
```

Tiếp đến, ta sẽ tạo view để hiển thị Form cho phép người dùng nhập dữ liệu. Trong thư mục **Admin > Views > Posts**, tạo view **Edit.cshtml**. Nhập đoạn mã sau để tạo Form.

```
1 @model TatBlog.WebApp.Areas.Admin.Models.PostEditModel
2
3 @{
4     ViewData["PageTitle"] = "Thêm/Cập nhật bài viết";
5 }
6
7 <h1>Thêm/Cập nhật bài viết</h1>
8
9 <form asp-area="Admin"
10    asp-controller="Posts"
11    asp-action="Edit"
12    enctype="multipart/form-data"
13    method="post">
14     <div asp-validation-summary="ModelOnly"></div>
15     <input type="hidden" asp-for="Id" />
16
17     <div class="row mb-3">
18         <label asp-for="Title"
19             class="col-sm-2 col-form-label"></label>
20         <div class="col-sm-10">
21             <input type="text" class="form-control" asp-for="Title">
22             <span asp-validation-for="Title"
23                 class="text-danger"></span>
24         </div>
25     </div>
26 
```

```
27     <div class="row mb-3">
28         <label asp-for="UrlSlug"
29             class="col-sm-2 col-form-label"></label>
30         <div class="col-sm-10">
31             <input type="text" class="form-control" asp-for="UrlSlug">
32             <span asp-validation-for="UrlSlug"
33                 class="text-danger"></span>
34         </div>
35     </div>
36
37     <div class="row mb-3">
38         <label asp-for="ShortDescription"
39             class="col-sm-2 col-form-label"></label>
40         <div class="col-sm-10">
41             <textarea class="form-control"
42                 asp-for="ShortDescription"></textarea>
43             <span asp-validation-for="ShortDescription"
44                 class="text-danger"></span>
45         </div>
46     </div>
47
48     <div class="row mb-3">
49         <label asp-for="Description"
50             class="col-sm-2 col-form-label"></label>
51         <div class="col-sm-10">
52             <textarea class="form-control"
53                 asp-for="Description" rows="10"></textarea>
54             <span asp-validation-for="Description"
55                 class="text-danger"></span>
56         </div>
57     </div>
58
59     <div class="row mb-3">
60         <label asp-for="Meta"
61             class="col-sm-2 col-form-label"></label>
62         <div class="col-sm-10">
63             <input type="text" class="form-control" asp-for="Meta">
64             <span asp-validation-for="Meta"
65                 class="text-danger"></span>
66         </div>
67     </div>
68
69     <div class="row mb-3">
70         <label asp-for="AuthorId"
71             class="col-sm-2 col-form-label"></label>
72         <div class="col-sm-10">
73             <select class="form-control"
74                 asp-for="AuthorId" asp-items="Model.AuthorList">
75                 <option value="">-- Chọn tác giả --</option>
76             </select>
77             <span asp-validation-for="AuthorId"
78                 class="text-danger"></span>
79         </div>
80     </div>
```

```
81
82     <div class="row mb-3">
83         <label asp-for="CategoryId"
84             class="col-sm-2 col-form-label"></label>
85         <div class="col-sm-10">
86             <select class="form-control"
87                 asp-for="CategoryId" asp-items="Model.CategoryList">
88                 <option value="">-- Chọn tác giả --</option>
89             </select>
90             <span asp-validation-for="CategoryId"
91                 class="text-danger"></span>
92         </div>
93     </div>
94
95     <div class="row mb-3">
96         <label asp-for="SelectedTags"
97             class="col-sm-2 col-form-label"></label>
98         <div class="col-sm-10">
99             <textarea class="form-control"
100                asp-for="SelectedTags" rows="5"></textarea>
101             <span asp-validation-for="SelectedTags"
102                 class="text-danger"></span>
103         </div>
104     </div>
105
106     @if (!string.IsNullOrWhiteSpace(Model.ImageUrl))
107     {
108         <div class="row mb-3">
109             <label asp-for="ImageUrl"
110                 class="col-sm-2 col-form-label"></label>
111             <div class="col-sm-10">
112                 
113             </div>
114         </div>
115     }
116
117     <div class="row mb-3">
118         <label asp-for="ImageFile"
119             class="col-sm-2 col-form-label"></label>
120         <div class="col-sm-10">
121             <input type="file"
122                 class="form-control"
123                 asp-for="ImageFile">
124         </div>
125     </div>
126
127     <div class="row mb-3">
128         <div class="col-sm-10 offset-sm-2">
129             <div class="form-check">
130                 <input class="form-check-input"
131                     type="checkbox"
132                     asp-for="Published">
133                 <label class="form-check-label"
134                     asp-for="Published"></label>

```

```

135           </div>
136       </div>
137   </div>
138
139   <div class="text-center">
140     <button type="submit" class="btn btn-primary">
141       Lưu Các Thay Đổi
142     </button>
143
144     <a asp-area="Admin"
145       asp-controller="Posts"
146       asp-action="Index"
147       class="btn btn-danger">
148       Hủy & Quay lại
149     </a>
150   </div>
151 </form>
152
153 @section scripts
154 {
155   <partial name="_ValidationScriptsPartial" />
156 }
157

```

Biên dịch chương trình, kiểm tra và sửa lỗi (nếu có). Chạy chương trình, truy cập vào trang Bài viết, nhấn chuột vào tiêu đề một bài viết để xem Form nhập dữ liệu.

Tips & Tricks   Chu đề   Tác giả   Thẻ   Bài viết   Bình luận

## Thêm/Cập nhật bài viết

Tiêu đề	Beware of records, with expressions and calculated properties Updated
Slug	beware-of-records-with-expressions-and-calculated-properties
Giới thiệu	I've been using C# records a lot since the feature was introduced. However, when using them, we really need to understand how they work, otherwise we might face
Nội dung	Given their terseness, even ignoring the other records characteristics, as soon as I'm creating a type that I'm expecting to be immutable, I immediately start with a record. However this might not always be adequate, and in this post we'll look at a very specific example where I introduced a bug in the code due to not taking into consideration all of the records characteristics. This solves the problem, and might be a valid solution in general. It has one potential issue though: every time we use SomeCalculatedValue, like the method that the property get accessor is, the value will be calculated. Depending on the way the property is used, it might not be a problem, or it might be, as we're always repeating the same logic and creating new

Để xử lý việc lưu các thay đổi mà người dùng đã nhập vào, ta cần tạo một phương thức khác. Trong lớp **PostsController**, định nghĩa một action **Edit** khác như sau:

```

58  [HttpPost]
59  public async Task<IActionResult> Edit(PostEditModel model)
60  {
61      if (!ModelState.IsValid)
62      {
63          await PopulatePostEditModelAsync(model);
64          return View(model);
65      }
66
67      var post = model.Id > 0
68          ? await _blogRepository.GetPostByIdAsync(model.Id)
69          : null;
70
71      if (post == null)
72      {
73          post = _mapper.Map<Post>(model);
74
75          post.Id = 0;
76          post.PostedDate = DateTime.Now;
77      }
78      else
79      {
80          _mapper.Map(model, post);
81
82          post.Category = null;
83          post.ModifiedDate = DateTime.Now;
84      }
85
86      await _blogRepository.CreateOrUpdatePostAsync(
87          post, model.GetSelectedTags());
88
89      return RedirectToAction(nameof(Index));
90  }

```

Ngoài ra, ta định nghĩa thêm action **VerifyPostSlug** để kiểm tra xem **UrlSlug** đã được sử dụng cho một bài viết khác hay chưa.

```

92  [HttpPost]
93  public async Task<IActionResult> VerifyPostSlug(
94      int id, string urlSlug)
95  {
96      var slugExisted = await _blogRepository
97          .IsPostSlugExistedAsync(id, urlSlug);
98
99      return slugExisted
100         ? Json($"Slug '{urlSlug}' đã được sử dụng")
101         : Json(true);
102  }

```

Biên dịch chương trình, kiểm tra và sửa lỗi (nếu có). Chạy chương trình, kiểm tra hoạt động của chức năng vừa xây dựng bằng cách truy cập trang “**Bài viết**”, nhấn nút “**Thêm mới**”. Nhập dữ liệu để tạo một bài viết mới, sau đó nhấn nút “**Lưu các thay đổi**” để lưu bài viết.

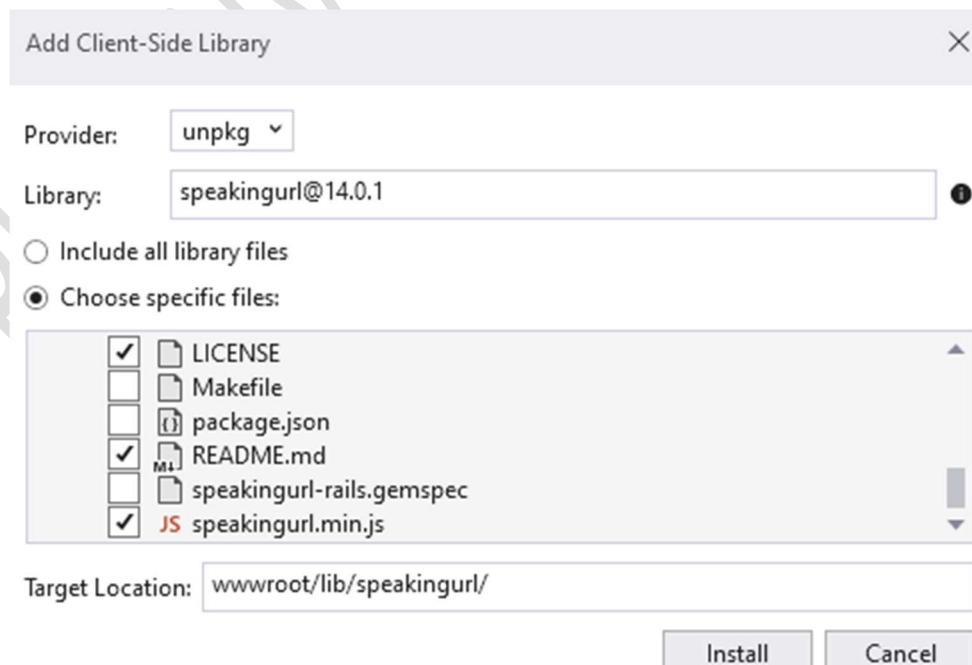
Tips & Tricks Chu đề Tác giả Thẻ Bài viết Bình luận

## Thêm/Cập nhật bài viết

Tiêu đề	What's On Tap for .NET 8
Slug	net-8-preview
Giới thiệu	The first preview of .NET 8 is coming in a couple of weeks or so said Microsoft's David Ortinau during a livestreamed tech event held in Stockholm.
Nội dung	We will start previewing .NET 8 in a few hours, not too long from now," Ortinau said. "Actually, it'll be two weeks, I think, from now-ish. That's not public information, except for the part that I just announced it on our livestream. But I didn't give you a date. So really, it's still pretty vague. We're good.

Để tiện cho người dùng, thông thường slug được tạo ra từ tiêu đề của bài viết (người dùng có thể sửa lại sau). Ta sẽ bổ sung mã JavaScript vào tập tin **Edit.cshtml** để khi người dùng nhập xong tiêu đề bài viết thì slug sẽ được tạo tự động.

Ta cần cài gói thư viện **speakingurl** bằng cách sử dụng chức năng **Add > Client-Side Library**.



Sau đó, bổ sung đoạn mã JavaScript sau vào cuối tập tin **Edit.cshtml**.

```

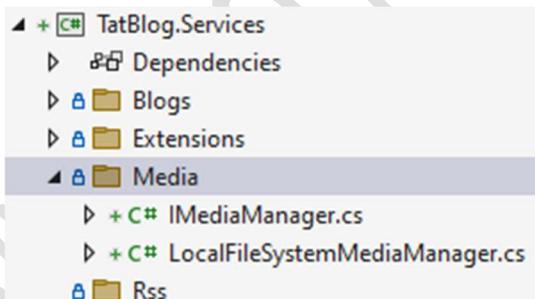
153 @section scripts
154 {
155     <partial name="_ValidationScriptsPartial" />
156
157     <script src="~/lib/speakingurl/speakingurl.min.js"></script>
158
159     <script>
160         $(function () {
161             $('#Title').change(function () {
162                 var slug = getSlug(this.value);
163                 $('#UrlSlug').val(slug).trigger('blur');
164             });
165         });
166     </script>
167 }
168

```

Chạy chương trình và kiểm tra kết quả.

## 5. Bổ sung tính năng upload hình ảnh

Trong dự án **TatBlog.Services**, tạo thêm thư mục **Media**. Sau đó, tạo interface **IMediaManager** và lớp **LocalFileSystemMediaManager**.



Cài đặt mã lệnh cho interface **IMediaManager**:

```

3 4 references | 0 changes | 0 authors, 0 changes
3 public interface IMediaManager
4 {
5     2 references | 0 changes | 0 authors, 0 changes
5         Task<string> SaveFileAsync(
6             Stream buffer,
7             string originalFileName,
8             string contentType,
9             CancellationToken cancellationToken = default);
10
11    2 references | 0 changes | 0 authors, 0 changes
11        Task<bool> DeleteFileAsync(
12            string filePath,
13            CancellationToken cancellationToken = default);
14

```

Lớp **LocalFileSystemMediaManager** thực thi interface **IMediaManager** và cài đặt cụ thể cách lưu trữ tập tin được gửi lên server vào thư mục **wwwroot > uploads**.

```
5  public class LocalFileSystemMediaManager : IMediaManager
6  {
7      private const string PicturesFolder = "uploads/pictures/{0}{1}";
8      private readonly ILogger<LocalFileSystemMediaManager> _logger;
9
10     0 references | 0 changes | 0 authors, 0 changes
11     public LocalFileSystemMediaManager(
12         ILogger<LocalFileSystemMediaManager> logger)
13     {
14         _logger = logger;
15     }
16
17     2 references | 0 changes | 0 authors, 0 changes
18     public async Task<string> SaveFileAsync(
19         Stream buffer,
20         string originalFileName,
21         string contentType,
22         CancellationToken cancellationToken = default)
23     {
24         try
25         {
26             if (!buffer.CanRead || !buffer.CanSeek || buffer.Length == 0)
27                 return null;
28
29             var fileExt = Path.GetExtension(originalFileName).ToLower();
30             var returnedFilePath = CreateFilePath(
31                 fileExt, contentType.ToLower());
32             var fullPath = Path.GetFullPath(
33                 Path.Combine(Environment.CurrentDirectory,
34                     "wwwroot", returnedFilePath));
35
36             // Make sure we are at beginning of source stream
37             buffer.Position = 0;
38
39             await using var fileStream = new FileStream(
40                 fullPath, FileMode.Create);
41             await buffer.CopyToAsync(fileStream, cancellationToken);
42
43             return returnedFilePath;
44         }
45         catch (Exception ex)
46         {
47             _logger.LogError(ex, $"Could not save file '{originalFileName}'.");
48         }
49     }
50
51     2 references | 0 changes | 0 authors, 0 changes
52     public Task<bool> DeleteFileAsync(
53         string filePath,
54         CancellationToken cancellationToken = default)
55     {
56         try
57         {
```

```

56     if (string.IsNullOrWhiteSpace(filePath))
57         Task.FromResult(true);
58
59     var fullPath = Path.GetFullPath(Path.Combine(
60         Environment.CurrentDirectory, "wwwroot", filePath));
61     File.Delete(fullPath);
62
63     return Task.FromResult(true);
64 }
65 catch (Exception ex)
66 {
67     _logger.LogError(ex, $"Could not delete file '{filePath}'.");
68     return Task.FromResult(false);
69 }
70
71
72
73     1 reference | 0 changes | 0 authors, 0 changes
74     private string CreateFilePath(
75         string fileExt, string contentType = null)
76     {
77         return string.Format(PicturesFolder,
78             Guid.NewGuid().ToString("N"), fileExt);
79     }

```

Trong lớp **TatBlog.WebApp > Extensions > WebApplicationExtensions**, cập nhật lại phương thức **ConfigureServices** để thêm các lớp vừa tạo vào DI Container.

```

// Đăng ký các dịch vụ với DI Container
1 reference | 0 changes | 0 authors, 0 changes
public static WebApplicationBuilder ConfigureServices(
    this WebApplicationBuilder builder)
{
    builder.Services.AddDbContext<BlogDbContext>(options =>
        options.UseSqlServer(
            builder.Configuration
                .GetConnectionString("DefaultConnection")));

    builder.Services.AddScoped<IMediaManager, LocalFileSystemMediaManager>();
    builder.Services.AddScoped<IBlogRepository, BlogRepository>();
    builder.Services.AddScoped<IDataSeeder, DataSeeder>();

    return builder;
}

```

Để có thể sử dụng phương thức **SaveFileAsync** trong action **Edit**, ta cần cập nhật lại lớp **PostsController** như sau:

```

1 reference | 0 changes | 0 authors, 0 changes
12 public class PostsController : Controller
13 {
14     private readonly IBlogRepository _blogRepository;
15     private readonly IMediaManager _mediaManager;
16     private readonly IMapper _mapper;
17
18     0 references | 0 changes | 0 authors, 0 changes
19     public PostsController(
20         IBlogRepository blogRepository,
21         IMediaManager mediaManager,
22         IMapper mapper)
23     {
24         _blogRepository = blogRepository;
25         _mediaManager = mediaManager;
26         _mapper = mapper;
27     }

```

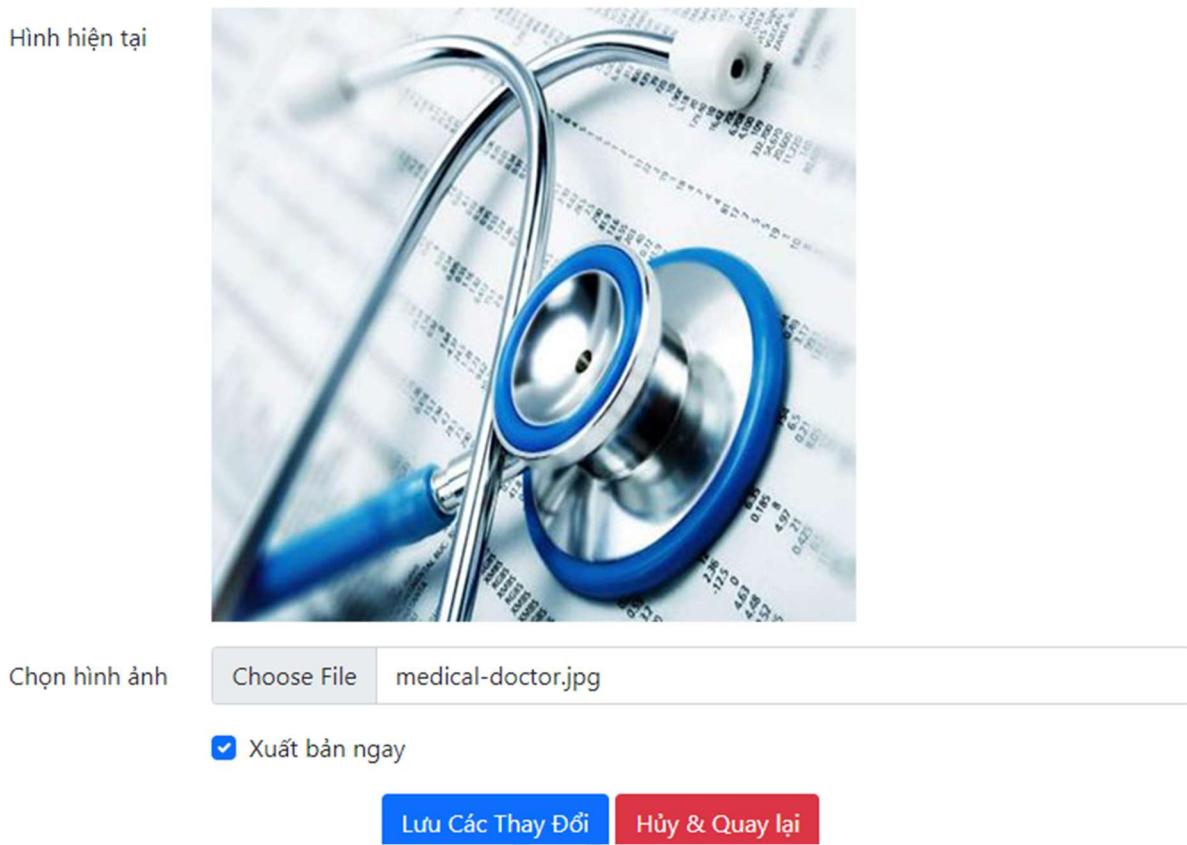
Bổ sung mã nguồn vào phương thức Edit để thực hiện việc lưu tập tin hình ảnh

```

63     public async Task<IActionResult> Edit(PostEditModel model)
64     {
65         if (!ModelState.IsValid)...
66
67         var post = model.Id > 0
68             ? await _blogRepository.GetPostByIdAsync(model.Id)
69             : null;
70
71         if (post == null)...[red box]
72         else...[red box]
73
74         // Nếu người dùng có upload hình ảnh minh họa cho bài viết
75         if (model.ImageFile?.Length > 0)
76         {
77             // Thực hiện việc lưu tập tin vào thư mục uploads
78             var newImagePath = await _mediaManager.SaveFileAsync(
79                 model.ImageFile.OpenReadStream(),
80                 model.ImageFile.FileName,
81                 model.ImageFile.ContentType);
82
83             // Nếu lưu thành công, xóa tập tin hình ảnh cũ (nếu có)
84             if (!string.IsNullOrWhiteSpace(newImagePath))
85             {
86                 await _mediaManager.DeleteFileAsync(post.ImageUrl);
87                 post.ImageUrl = newImagePath;
88             }
89
90             await _blogRepository.CreateOrUpdatePostAsync(
91                 post, model.GetSelectedTags());
92
93             return RedirectToAction(nameof(Index));
94         }
95     }

```

Biên dịch chương trình, kiểm tra và sửa lỗi (nếu có). Chạy chương trình, chọn một bài viết để cập nhật, thử chọn một tập tin ảnh rồi nhấn nút “Lưu các thay đổi” để kiểm tra việc upload.



## 6. Sử dụng FluentValidation để kiểm tra tính hợp lệ của dữ liệu

Trong phần này, ta sẽ cài đặt và sử dụng gói thư viện **FluentValidation** để kiểm tra dữ liệu nhập bởi người dùng. Sinh viên tìm hiểu thêm về cách sử dụng thư viện này tại: <https://docs.fluentvalidation.net/en/latest/>.

Đầu tiên, cài gói thư viện **FluentValidation.AspNetCore** vào dự án **TatBlog.WebApp**. Khi cài đặt gói này, hai gói thư viện phụ thuộc là **FluentValidation** và **FluentValidation.DependencyInjectionExtensions** cũng được cài đặt theo.

Tiếp theo, tạo thư mục **Validations** trong dự án **TatBlog.WebApp**. Tại thư mục này, tạo hai lớp **PostValidator** và **FluentValidation.DependencyInjection**.



Trong lớp **PostValidator**, cài đặt các quy tắc cần thiết để kiểm tra dữ liệu được nhập bởi người dùng.

```

57     .Otherwise(() =>
58     {
59         RuleFor(x => x.ImageFile)
60             .MustAsync(SetImageIfNotExist)
61             .WithMessage("Bạn phải chọn hình ảnh cho bài viết");
62     });
63 }
64
65 // Kiểm tra xem người dùng đã nhập ít nhất 1 thẻ (tag)
66 // 1 reference | 0 changes | 0 authors, 0 changes
67 private bool HasAtLeastOneTag(
68     PostEditModel postModel, string selectedTags)
69 {
70     return postModel.GetSelectedTags().Any();
71 }
72
73 // Kiểm tra xem bài viết đã có hình ảnh chưa.
74 // Nếu chưa có, bắt buộc người dùng phải chọn file.
75 // 1 reference | 0 changes | 0 authors, 0 changes
76 private async Task<bool> SetImageIfNotExist(
77     PostEditModel postModel,
78     IFormFile imageFile,
79     CancellationToken cancellationToken)
80 {
81     // Lấy thông tin bài viết từ CSDL
82     var post = await _blogRepository.GetPostByIdAsync(
83         postModel.Id, false, cancellationToken);
84
85     // Nếu bài viết đã có hình ảnh => Không bắt buộc chọn file
86     if (!string.IsNullOrWhiteSpace(post?.ImageUrl))
87         return true;
88
89     // Ngược lại (bài viết chưa có hình ảnh), kiểm tra xem
90     // người dùng đã chọn file hay chưa. Nếu chưa thì báo lỗi.
91     return imageFile is { Length: > 0 };
92 }

```

Trong lớp tĩnh **FluentValidationDependencyInjection**, định nghĩa phương thức **ConfigureFluentValidation** để đăng ký các dịch vụ được sử dụng bởi **FluentValidation** với DI container.

```

9
10 public static WebApplicationBuilder ConfigureFluentValidation(
11     this WebApplicationBuilder builder)
12 {
13     // Enable client-side integration
14     builder.Services.AddFluentValidationClientsideAdapters();
15
16     // Scan and register all validators in given assembly
17     builder.Services.AddValidatorsFromAssembly(
18         Assembly.GetExecutingAssembly());
19
20     return builder;
21 }

```

Gọi phương thức vừa tạo trong lớp **Program**.

```

5  var builder = WebApplication.CreateBuilder(args);
6  {
7      builder
8          .ConfigureMvc()
9          .ConfigureServices()
10         .ConfigureMapster()
11         .ConfigureFluentValidation();
12 }

```

Trong lớp **PostEditModel**, xóa bỏ các attribute **Required** và **MaxLength** ở các thuộc tính.

Cập nhật lại phương thức (action) **Edit** trong lớp **PostsController** để sử dụng FluentValidation cho việc kiểm tra dữ liệu đầu vào.

```

64 [HttpPost]
65 0 references | 0 changes | 0 authors, 0 changes
66 public async Task<IActionResult> Edit(
67     IValidator<PostEditModel> postValidator,
68     PostEditModel model)
69 {
70     var validationResult = await postValidator.ValidateAsync(model),
71     if (!validationResult.IsValid)
72     {
73         validationResult.AddToModelState(ModelState);
74     }
75     if (!ModelState.IsValid)
76     {
77         await PopulatePostEditModelAsync(model);
78         return View(model);
79     }
80 }

```

Biên dịch chương trình, kiểm tra và sửa lỗi (nếu có). Chạy chương trình để kiểm tra kết quả.

Hãy cập nhật lớp **PostValidator** để chuyển các thông báo lỗi như trong hình sau sang tiếng Việt.

## Thêm/Cập nhật bài viết

Tiêu đề

'Title' must not be empty.

Slug

'Url Slug' must not be empty.

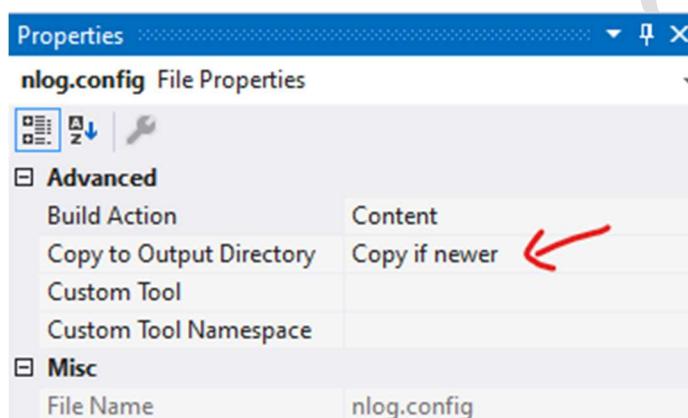
## 7. Ghi nhật ký hệ thống sử dụng NLog

Trong dự án **TatBlog.WebApp**, cài đặt gói thư viện **NLog.Web.AspNetCore**. Khi cài gói này, 2 gói phụ thuộc là **NLog** và **NLog.Extensions.Logging** cũng được tự động cài đặt theo.

Sinh viên tìm hiểu cách sử dụng **NLog** tại: <https://github.com/NLog/NLog/wiki>.

Tham khảo của bài viết tại địa chỉ <https://github.com/NLog/NLog/wiki/Getting-started-with-ASP.NET-Core-6> để tạo tập tin **nlog.config** trong dự án **TatBlog.WebApp**.

Nhấp phải chuột vào tập tin **nlog.config**, chọn **Properties**. Sau đó, chọn giá trị **Copy if newer** cho mục **Copy to Output Directory**.



Trong lớp **Extensions > WebApplicationExtensions**, định nghĩa phương thức **ConfigureNLog** để cấu hình việc sử dụng **NLog** trong việc ghi nhật ký hệ thống.

```

22 // Cấu hình việc sử dụng NLog
23 1 reference | 0 changes | 0 authors, 0 changes
24 public static WebApplicationBuilder ConfigureNLog(
25     this WebApplicationBuilder builder)
26 {
27     builder.Logging.ClearProviders();
28     builder.Host.UseNLog();
29
30     return builder;
}

```

Gọi hàm vừa tạo trong tập tin **Program.cs**.

```

5 var builder = WebApplication.CreateBuilder(args);
6 {
7     builder
8         .ConfigureMvc()
9         .ConfigureNLog()
10        .ConfigureServices()
11        .ConfigureMapster()
12        .ConfigureFluentValidation();
13 }

```

Để ghi nhật ký hệ thống, ta sử dụng đối tượng ILogger được tạo bởi DI container như sau.

```

3 references | 0 changes | 0 authors, 0 changes
14 public class PostsController : Controller
15 {
16     private readonly ILogger<PostsController> _logger;
17     private readonly IBlogRepository _blogRepository;
18     private readonly IMediaManager _mediaManager;
19     private readonly IMapper _mapper;
20
21     0 references | 0 changes | 0 authors, 0 changes
22     public PostsController(
23         ILogger<PostsController> logger,
24         IBlogRepository blogRepository,
25         IMediaManager mediaManager,
26         IMapper mapper)
27     {
28         _logger = logger;
29         _blogRepository = blogRepository;
30         _mediaManager = mediaManager;
31         _mapper = mapper;
32     }
33
34     public async Task<IActionResult> Index(PostFilterModel model)
35     {
36         _logger.LogInformation("Tạo điều kiện truy vấn");
37
38         // Sử dụng Mapster để tạo đối tượng PostQuery
39         // từ đối tượng PostFilterModel model
40         var postQuery = _mapper.Map<PostQuery>(model);
41
42         _logger.LogInformation("Lấy danh sách bài viết từ CSDL");
43
44         ViewBag.PostsList = await _blogRepository
45             .GetPagedPostsAsync(postQuery, 1, 10);
46
47         _logger.LogInformation("Chuẩn bị dữ liệu cho ViewModel");
48
49         await PopulatePostFilterModelAsync(model);
50
51         return View(model);
52     }

```

Biên dịch chương trình, kiểm tra và sửa lỗi (nếu có). Chạy chương trình, truy cập vào trang “**Bài viết**”.

Tìm đường dẫn đến tập tin log được thiết lập trong thuộc tính **fileName** của thẻ **target** trong tập tin **nlog.config**. Mở tập tin đó để xem kết quả ghi nhật ký hệ thống.

Ta cũng có thể lưu vết lại quá trình di chuyển giữa các trang của người dùng bằng cách tạo ra một middleware (hoặc ActionFilter) và sử dụng NLog để ghi lại thời điểm, địa chỉ IP của người dùng và địa chỉ trang mà người dùng truy cập.

Để đơn giản, phần này hướng dẫn cách tạo một middleware. Trong dự án **TatBlog.WebApp**, tạo thư mục **Middlewares**. Tại thư mục này, tạo lớp **UserActivityMiddleware**. Cài đặt lớp này như sau:

```

4 references | 0 changes | 0 authors, 0 changes
3  public class UserActivityMiddleware
4  {
5      private readonly RequestDelegate _next;
6      private readonly ILogger<UserActivityMiddleware> _logger;
7
8      0 references | 0 changes | 0 authors, 0 changes
9      public UserActivityMiddleware(
10         RequestDelegate next,
11         ILogger<UserActivityMiddleware> logger)
12     {
13         _next = next;
14         _logger = logger;
15     }
16
17     0 references | 0 changes | 0 authors, 0 changes
18     public async Task Invoke(HttpContext context)
19     {
20         _logger.LogInformation(
21             "{Time:yyyy-MM-dd HH:mm:ss} - IP: {IpAddress} - Path: {Url}",
22             DateTime.Now,
23             context.Connection.RemoteIpAddress?.ToString(),
24             context.Request.Path);
25
26         await _next(context);
}

```

Trong phương thức **UseRequestPipeline** của lớp **WebApplicationExtensions**, bổ sung mã lệnh để thêm middleware vừa tạo vào request pipeline.

```

78 // Thêm middleware lựa chọn endpoint phù hợp nhất
79 // để xử lý một HTTP request.
80 app.UseRouting();
81
82 // Thêm middleware để lưu vết người dùng
83 app.UseMiddleware<UserActivityMiddleware>();
84
85 return app;

```

Chạy chương trình, truy cập vào các trang đã xây dựng và kiểm tra tập tin log để xem kết quả.

```

2023-03-06 17:52:01.9209|0|INFO|TatBlog.WebApp.Middlewares.UserActivityMiddleware|2023-03-06 17:52:01 - IP: 127.0.0.1 - Path: /admin/Comments |url:
https://localhost/admin/Comments|action: Index|TatBlog.WebApp.Middlewares.UserActivityMiddleware.Invoke
2023-03-06 17:52:04.5958|0|INFO|TatBlog.WebApp.Middlewares.UserActivityMiddleware|2023-03-06 17:52:04 - IP: 127.0.0.1 - Path: /admin |url: https://localhost/admin|action:
Index|TatBlog.WebApp.Middlewares.UserActivityMiddleware.Invoke
2023-03-06 17:52:05.2120|0|INFO|TatBlog.WebApp.Middlewares.UserActivityMiddleware|2023-03-06 17:52:05 - IP: 127.0.0.1 - Path: /admin/Categories |url:
https://localhost/admin/Categories|action: Index|TatBlog.WebApp.Middlewares.UserActivityMiddleware.Invoke
2023-03-06 17:52:05.7145|0|INFO|TatBlog.WebApp.Middlewares.UserActivityMiddleware|2023-03-06 17:52:05 - IP: 127.0.0.1 - Path: /admin/Authors |url:
https://localhost/admin/Authors|action: Index|TatBlog.WebApp.Middlewares.UserActivityMiddleware.Invoke
2023-03-06 17:52:06.2753|0|INFO|TatBlog.WebApp.Middlewares.UserActivityMiddleware|2023-03-06 17:52:06 - IP: 127.0.0.1 - Path: /admin/Tags |url:
https://localhost/admin/Tags|action: Index|TatBlog.WebApp.Middlewares.UserActivityMiddleware.Invoke
2023-03-06 17:52:07.2034|0|INFO|TatBlog.WebApp.Middlewares.UserActivityMiddleware|2023-03-06 17:52:07 - IP: 127.0.0.1 - Path: /admin/Posts |url:
https://localhost/admin/Posts|action: Index|TatBlog.WebApp.Middlewares.UserActivityMiddleware.Invoke
2023-03-06 17:52:07.2316|0|INFO|TatBlog.WebApp.Areas.Admin.Controllers.PostsController|Tạo điều kiện truy vấn |url: https://localhost/admin/Posts|action:
Index|TatBlog.WebApp.Areas.Admin.Controllers.PostsController.Index
2023-03-06 17:52:07.2721|0|INFO|TatBlog.WebApp.Areas.Admin.Controllers.PostsController|Lấy danh sách bài viết từ CSDL |url: https://localhost/admin/Posts|action:
Index|TatBlog.WebApp.Areas.Admin.Controllers.PostsController.Index
2023-03-06 17:52:07.7349|0|INFO|TatBlog.WebApp.Middlewares.UserActivityMiddleware|2023-03-06 17:52:07 - IP: 127.0.0.1 - Path: /admin/Comments |url:
https://localhost/admin/Comments|action: Index|TatBlog.WebApp.Middlewares.UserActivityMiddleware.Invoke

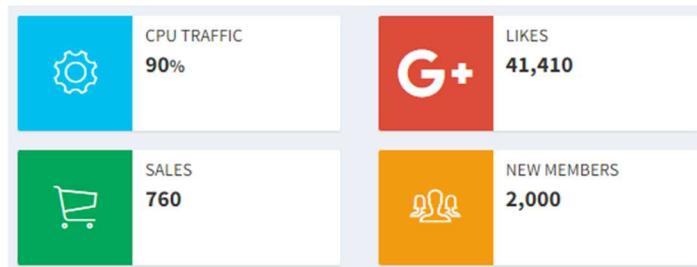
```

## C. Bài tập thực hành

### 1. Tiếp tục hoàn thiện các chức năng quản lý bài viết.

- Trong phần hướng dẫn, trang hiển thị danh sách bài viết luôn hiển thị 10 bài viết mới nhất. Hãy cập nhật lại mã nguồn và thêm điều khiển phân trang để người quản trị có thể tải và xem tất cả các bài viết. (Sinh viên tham khảo cách xây dựng điều khiển phân trang ở bài Lab trước).
- Hiện tại cột “Xuất bản” hiển thị một trong 2 giá trị: “Có”, “Không”. Hãy cập nhật lại mã nguồn để hiển thị giá trị này dưới dạng nút bấm. Khi người dùng click chuột thì đổi trạng thái Xuất bản của bài viết.
- Bổ sung thêm mã lệnh để hiển thị nút xóa trên mỗi dòng ứng với bài viết. Khi người dùng click chuột vào nút này thì hỏi “Bạn có thực sự muốn xóa bài viết này không?”. Nếu người dùng trả lời Yes, thực hiện việc xóa bài viết và tải lại trang.
- Trên khung tìm kiếm, bổ sung thêm điều kiện tìm kiếm “Chưa xuất bản”, hiển thị dưới dạng checkbox. Khi người dùng đánh dấu chọn checkbox này và nhấn tìm kiếm thì chỉ hiển thị những bài viết có cờ Published bằng false.
- Trên khung tìm kiếm, thêm nút “Bỏ lọc”. Khi người dùng nhấn vào nút này thì xóa tất cả các điều kiện tìm kiếm trong các ô nhập và tải lại trang chưa đầy đủ bài viết.

- Cài đặt các chức năng xem danh sách, thêm, xóa, cập nhật chủ đề.
- Cài đặt các chức năng xem danh sách, thêm, xóa, cập nhật tác giả.
- Cài đặt các chức năng xem danh sách, thêm, xóa, cập nhật thẻ (tag).
- Cài đặt các chức năng xem danh sách, phê duyệt, xóa các bình luận.
- Cài đặt các chức năng xem danh sách, quản lý người đăng ký theo dõi blog.
- Ở trang “Bảng điều khiển” (action Index, controller Dashboard), hãy tạo các thống kê về: Tổng số bài viết, số bài viết chưa xuất bản, số lượng chủ đề, số lượng tác giả, số lượng bình luận đang chờ phê duyệt, số lượng người theo dõi, số lượng người mới theo dõi đăng ký (lấy số liệu trong ngày).



(Ví dụ cách hiển thị các con số thống kê)

8. Tìm hiểu và áp dụng [TinyMCE](#) (hoặc thư viện tương tự) vào dự án để tạo RichTextEditor. Áp dụng cho trường nhập nội dung của bài viết.
9. Tìm hiểu cách sử dụng [jqGrid](#) ([DataTables.js](#)) hoặc các thư viện tương tự) để hiển thị danh sách bài viết (chủ đề, tác giả, ...) dưới dạng bảng, hỗ trợ phân trang, sắp xếp các mẫu tin và tải dữ liệu bằng AJAX.

--- HẾT ---

PHUCNV.DONOTCOPY