**TƯ LIỆU CHO PROJECT**

**Tên Project:**

**XÂY DỰNG WEBSITE TIN TỨC NỘI BỘ HÒN RẠNG**

Nội dung

[I. Phần mở đầu 1](#_Toc468277523)

[1. Giới thiệu về project 1](#_Toc468277524)

[2. Prototype của project 2](#_Toc468277525)

[II. Thiết kế database 4](#_Toc468277526)

[1. Table Account 4](#_Toc468277527)

[2. Table AccountProperty 4](#_Toc468277528)

[3. Table AccountRoles 4](#_Toc468277529)

[4. Table Ads 5](#_Toc468277530)

[5. Table Group 5](#_Toc468277531)

[6. Table Roles 6](#_Toc468277532)

[7. Table Routing 6](#_Toc468277533)

[8. Table SysPara 6](#_Toc468277534)

[III. Thiết kế hệ thống 7](#_Toc468277535)

[1. Mô hình kiến trúc hệ thống 7](#_Toc468277536)

[2. Mô tả kiến trúc hệ thống 7](#_Toc468277537)

[2.1 Tầng Store.Data 7](#_Toc468277538)

[2.2 Tầng Store.Service 13](#_Toc468277539)

[2.3 Tầng Store.web 15](#_Toc468277540)

[IV. Bắt đầu thực hiện project 18](#_Toc468277541)

[1. Thiết lập chuỗi kết nối 18](#_Toc468277542)

[2. Thực hiện chức năng Create Account 18](#_Toc468277543)

[2.1 Tạo Class Account trong EntityModels 18](#_Toc468277544)

[2.2 Tạo Class AccountResponsitory 19](#_Toc468277545)

[2.3 Tạo AccountService 20](#_Toc468277546)

[2.4 Tạo AccountDetailViewModel 22](#_Toc468277547)

[2.5 Tạo Controller Account 23](#_Toc468277548)

[2.6 Tạo View 23](#_Toc468277549)

[2.7 Tạo ViewBag.Roles 24](#_Toc468277550)

[2.8 Tạo Action Create 25](#_Toc468277551)

[3. Thực hiện chức năng Create Group (Thêm mục nội dung) 27](#_Toc468277552)

[3.1 Tạo Class Group trong EntityModels 27](#_Toc468277553)

[3.2 Tạo Class GroupResponsitory 28](#_Toc468277554)

[3.3 Tạo GroupService 29](#_Toc468277555)

[3.4 Tạo GroupViewModels 32](#_Toc468277556)

[3.5 Tạo Controller Group 33](#_Toc468277557)

[3.6 Tạo View 33](#_Toc468277558)

[3.7 Tạo View GroupViewModel 35](#_Toc468277559)

[3.8 Tạo Action Create 37](#_Toc468277560)

# 

# Phần mở đầu

## Giới thiệu về project

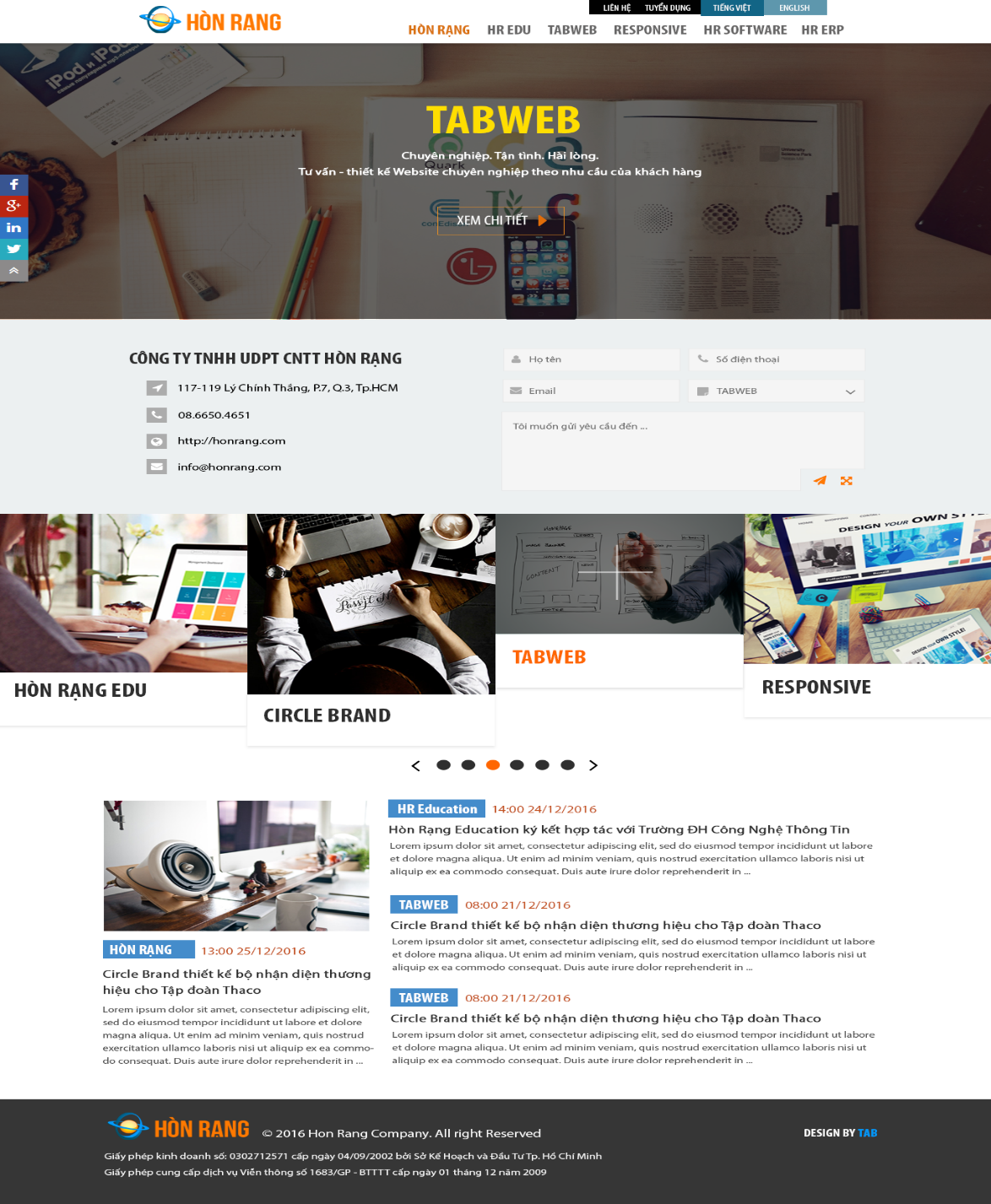
* Website Hòn Rạng được thiết kế nhằm cung cấp những thông tin về công ty Hòn Rạng, cũng như các sản phẩm mà công ty đã, đang và sẽ làm.
* Các chức năng có trong website Hòn Rạng:
  + **Trang chủ:**
    - Bố cục: logo, biểu tượng công ty
    - Thể hiện những hình ảnh, sản phẩm dịch vụ mà công ty cung cấp, hoặc những tin tức nổi bật của công ty.
    - Hệ thống menu được bố trí tiện lợi, hài hòa, dễ theo dõi
  + **Module tin tức**
    - Cấp chuyên mục tin và cấp tin bài.
    - Quản trị có thể chủ động thêm mới không hạn chế số chuyên mục và tin bài.
  + **Module liên hệ:**
    - Giúp người quản lý nhận thông tin, phản hồi, yêu cầu từ phía khách hàng.
    - Hỗ trợ người quản trị quản lý danh sách các yêu cầu, liên hệ của khách hàng thông qua trang quản trị của website hoặc gửi về email.
  + **Module quảng cáo:**
    - Tổng hợp các hình ảnh về sản phẩm, công ty
    - Danh sách các quảng cáo:  
       +  Hình ảnh

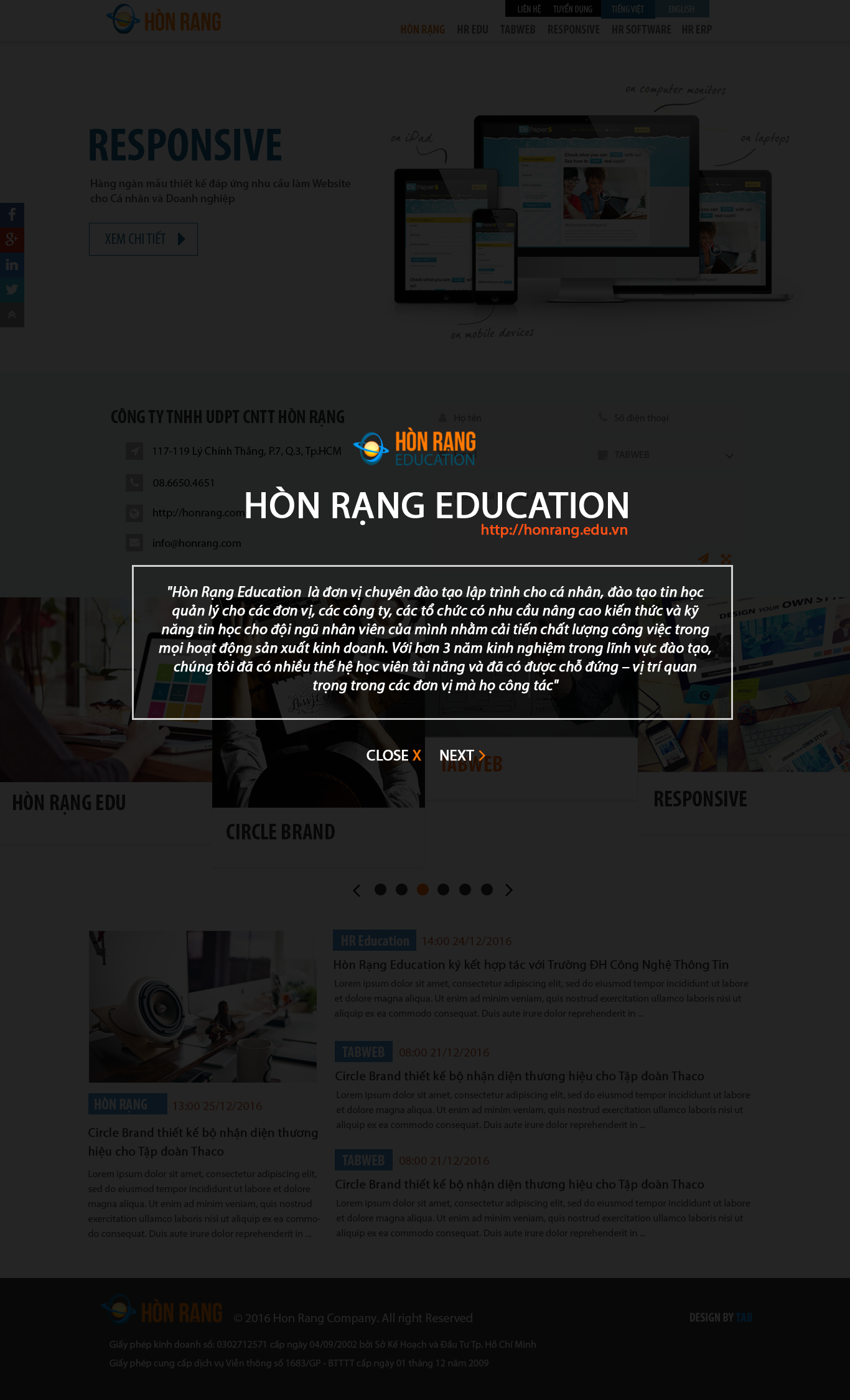
 +  Ghi chú

 +  Link tới website

* + …

## Prototype của project





# Thiết kế database

## Table Account

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Filed Name** | **Type** | **Size** | **Null** | **Description** |
| 1 | ID | int | 50 | No | PRIMARY KEY (Auto increment) |
| 2 | Email | nvarchar | 256 | Yes |  |
| 3 | EmailConfirmed | Bit |  | No |  |
| 4 | PasswordHash | Nvarchar | Max | Yes |  |
| 5 | SecurityStamp | Nvarchar | Max | Yes |  |
| 6 | PhoneNumber | Nvarchar | Max | Yes |  |
| 7 | PhoneNumberConfirmed | Bit |  | No |  |
| 8 | TwoFactorEnabled | bit |  | No |  |
| 9 | LockoutEndDateUtc | Datetime |  | Yes |  |
| 10 | LockoutEnabled | Bit |  | No |  |
| 11 | AccessFailedCount | Int |  | No |  |
| 12 | Username | Nvarchar | 256 | No |  |

## Table AccountProperty

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Filed Name** | **Type** | **Size** | **Null** | **Description** |
| 1 | AccountID | Int |  | No | PRIMARY KEY(link với table Account) |
| 2 | Property | Nchar | 64 | No |  |
| 3 | Value | Nvarchar | Max | No |  |
| 4 | ModifiedDate | Datetime |  | No |  |
| 5 | CreateDate | Datetime |  | No |  |

## Table AccountRoles

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Filed Name** | **Type** | **Size** | **Null** | **Description** |
| 1 | AccountId | Int |  | No | PRIMARY KEY(link với table Account) |
| 2 | RoleId | Int |  | No | PRIMARY KEY(link với table Role) |

## Table Ads

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Filed Name** | **Type** | **Size** | **Null** | **Description** |
| 1 | AdID | Int |  | No | PRIMARY KEY |
| 2 | AdName | Nvarchar | 100 | No |  |
| 3 | Url | Varchar | 100 | No |  |
| 4 | ImageUrl | Varchar | 100 | No |  |
| 5 | Counter | Int |  | No |  |
| 6 | AdKey | Varchar | 10 | No |  |
| 7 | Status | Int |  | No |  |
| 8 | Note | Nvarchar | Max | No |  |
| 9 | Author | Int |  | No |  |
| 10 | ModifiedUser | Int |  | No |  |
| 11 | ModifiedDate | Datetime |  | No |  |
| 12 | CreateDate | Datetime |  | No |  |
| 13 | ThumImage | Varchar | 100 | Yes |  |
| 14 | ThumTitle | Nvarchar | 250 | Yes |  |
| 15 | Language | Nvarchar | 12 | Yes |  |

## Table Group

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Filed Name** | **Type** | **Size** | **Null** | **Description** |
| 1 | GroupID | Int |  | No | PRIMARY KEY |
| 2 | Protected | Int |  | No |  |
| 3 | Language | Varchar | 5 | No |  |
| 4 | Title | Nvarchar | 64 | No |  |
| 5 | Abbreviation | Varchar | 32 | Yes |  |
| 6 | ImageUrl | Varchar | 250 | Yes |  |
| 7 | Description | Nvarchar | Max | Yes |  |
| 8 | Status | Int |  | No |  |
| 9 | GroupType | Int |  | No |  |
| 10 | GroupParentID | Int |  | No |  |
| 11 | ModifiedDate | Datetime |  | No |  |
| 12 | CreationDate | Datetime |  | No |  |
| 13 | Layout | Int |  | Yes |  |
| 14 | Level | Int |  | No |  |
| 15 | Priority | Int |  | No |  |
| 16 | ShowOnTopMenu | Int |  | Yes |  |
| 17 | ShowOnBottomMenu | Int |  | Yes |  |
| 18 | ShowOnArea | Int |  | Yes |  |
| 19 | Banner | Varchar | 250 | Yes |  |

## Table Roles

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Filed Name** | **Type** | **Size** | **Null** | **Description** |
| 1 | Id | Int |  | No | PRIMARY KEY |
| 2 | Name | Nvarchar | 64 | No |  |
| 3 | Description | Nvarchar | 64 | Yes |  |

## Table Routing

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Filed Name** | **Type** | **Size** | **Null** | **Description** |
| 1 | RoutingId | Int |  | No | PRIMARY KEY |
| 2 | FriendlyUrl | Varchar | 128 | No |  |
| 3 | Controller | Varchar | 128 | No |  |
| 4 | Action | Varchar | 128 | No |  |
| 5 | EntityId | Int |  | No |  |

## Table SysPara

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Filed Name** | **Type** | **Size** | **Null** | **Description** |
| 1 | ID | Int |  | No | PRIMARY KEY |
| 2 | ParaName | Nvarchar | 80 | No |  |
| 3 | ParaValue | Nvarchar | 2000 | No |  |
| 4 | Description | Nvarchar | Max | Yes |  |

…

# III. Thiết kế hệ thống

## 1. Mô hình kiến trúc hệ thống

- Kiến trúc hệ thống được chia làm 3 tầng:

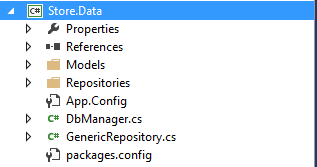
**+ Tầng Store.Data**: Tầng giao tiếp với hệ quản trị CSDL.

**+ Tầng Store.Service** (sẽ kết nối với tầng Store.Data): Xử lý các nghiệp vụ của chương trình như tính toán, xử lý hợp lệ và toàn vẹn về mặt dữ liệu.

**+ Tầng Store.Web** (sẽ kết nối với tầng Store.Data & Store.Web): Thành phần giao diện của chương trình tương tác với người sử dụng.

## 2. Mô tả kiến trúc hệ thống

### 2.1 Tầng Store.Data



#### 2.1.1 Folder Models

**- Class EntityModels:**

**+** Một bộ ánh xạ đối tượng – quan hệ cho phép người lập trình .NET  làm việc với dữ liệu quan hệ qua các đối tượng (object), do đó developer không phải viết các đoạn mã truy cập cơ sở dữ liệu thường cần phải viết.

+ Tên class và tên thuộc tính trong class phải trùng với tên table và thuộc tính của table trong database.

**- Class IdentityModel:**

**+** Dùng để xác thực user, role, account

+ Identity nó hỗ trợ cơ chế phân quyền, quản lý tài khoản, đăng nhập qua gmail, facebook, twitter, microsoft, nó còn giúp xác thực tài khoản qua email, qua mã OTP, hỗ trợ tự động gửi mail.... Nó rất hay trong việc bảo mật.

#### Folder Repositories

**- Gồm các class Respository**

**- Các khái niệm về Respository**

- Repository Pattern là một lớp trừu trượng (abstraction ) giữa tầng Store.Data và tầng Store.Service của một ứng dụng. Đây là một mô hình truy cập dữ liệu chặt chẽ hơn, bảo mật hơn cũng như gọn gàng hơn. Chúng ta sẽ tạo ra một logic truy cập dữ liệu trong một lớp riêng biệt, hoặc một tập hợp các lớp gọi là Repository, với nhiệm vụ là kết nối với tầng Store.Service của ứng dụng. Mục đích tạo ra lớp này để cách ly với việc tiếp cận Store.Data sao cho những thay đổi không ảnh hưởng trực tiếp đến tầng Store.Service.

- Nó cho phép tất cả các code của bạn có thể truy cập và sử dụng các đối tượng mà không cần quan tâm đến đối tượng đó tồn tại như thế nào. Tất cả các thông tin được lưu lại, bao gồm cả các mapping từ table tới các object đều được lưu trữ an toàn trong Repository.

- Thông thường, chúng ta sẽ tìm thấy các truy vấn SQL hay Linq nằm rải rác trong các code và khi bạn muốn thực hiện một thao tác với database thì bạn sẽ phải tìm trong code cũng như tìm các thuộc tính của bảng để xử lý. Điều này gây nên sự lãng phí về thời gian cũng như ảnh hưởng đến code chung.

- Với Repository Pattern, bạn chỉ cần thay đổi một đối tượng và một Repository, việc này tiết kiệm khá nhiều công sức cũng như ảnh hưởng rất nhỏ đến code chung.

- Một số lý do chúng ta nên sử dụng Repository Pattern:

+ Một nơi duy nhất để thay đổi quyền truy cập dữ liệu cũng như xử lý dữ liệu

+ Một nơi duy nhất chịu trách nhiệm cho việc mapping các table vào object.

    + Tăng tính bảo mật và rõ ràng cho code.

+ Khi thay đổi logic của tầng Store.Data hoặc Store.Service, không cần thay đổi Repository

#### 2.1.3 Class DbManager

- Thiết lập chuỗi kết nối đến hệ quản trị cơ sở dữ liệu

#### 2.1.4 Class GenericRepository

- Project thường đòi hỏi truy xuất một cách tốt nhất, nên tạo ra tất cả các repository logic ở cùng một nơi. Chúng ta cần tạo 1 và chỉ 1 repository cho việc thao tác với toàn bộ các class Entity. Vậy để giải quyết điều này chúng ta phải sử dụng Generic Repository Pattern.

- Lợi ích của Generic Repository Pattern:

+ Giảm thiểu sự trùng lặp code

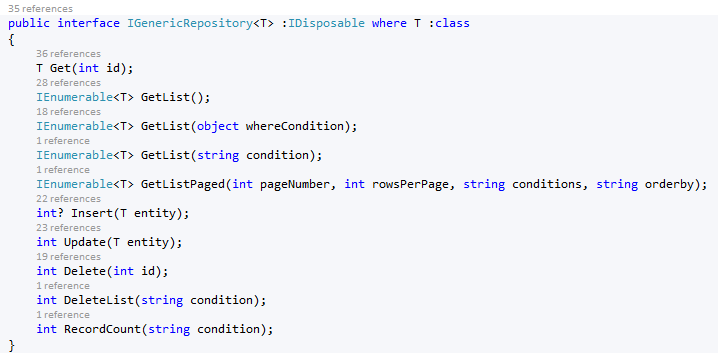
+ Đảm bảo các coder dùng chung 1 pattern

+ Ít lỗi hơn

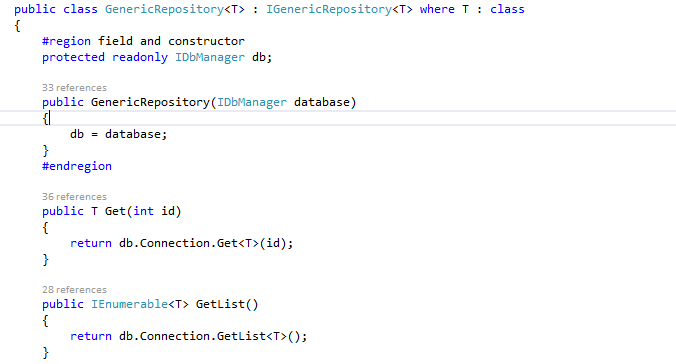
#### 2.1.5 Ví dụ về Repository

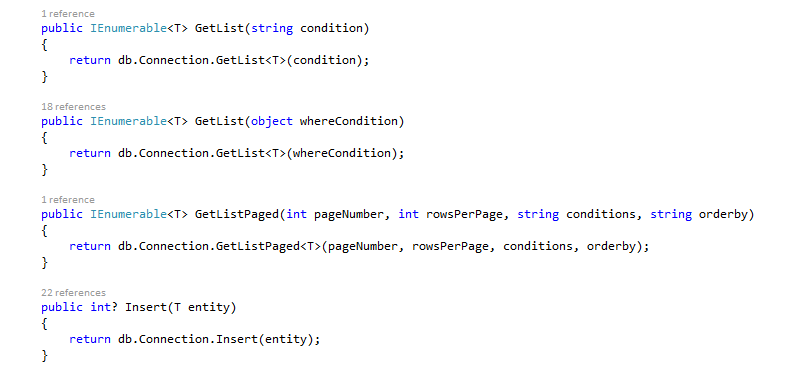
**- Ngữ cảnh**

+ Giả sử bạn cần viết 1 ứng dụng quản lý học sinh. Bạn cần phải thực thi các hành động CRUD (Create, Read, Update, Delete) lên mỗi thực thể (entity), ví dụ như học sinh. Tiếp đó, bạn cũng thực thi những hành động CRUD này lên giáo viên, rồi lớp học, môn học... Có thể thấy rằng các hành động thêm, xóa, sửa được lặp đi lặp lại, trong đó chỉ có thực thể là thay đổi. Mẫu Repository giúp bạn giải quyết được vấn đề này bằng cách hết sức đơn giản.

**- Respository Interface** 

=> Interface định nghĩa 10 phương thức dùng để thao tác với mỗi entity. Bạn có thể định nghĩa nhiều hơn. Và chú ý rằng, ở đây mình sử dụng kiểu Generic (template T) để định nghĩa cho interface. Điều này dễ hiểu bởi vì bạn có thể sử dụng kiểu này cho nhiều loại đối tượng sau này.

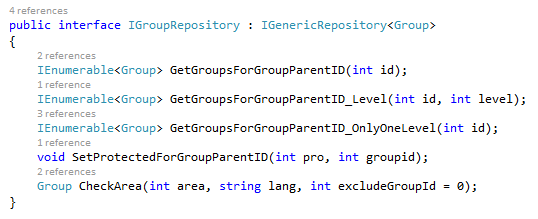
**- Thực thi interface**



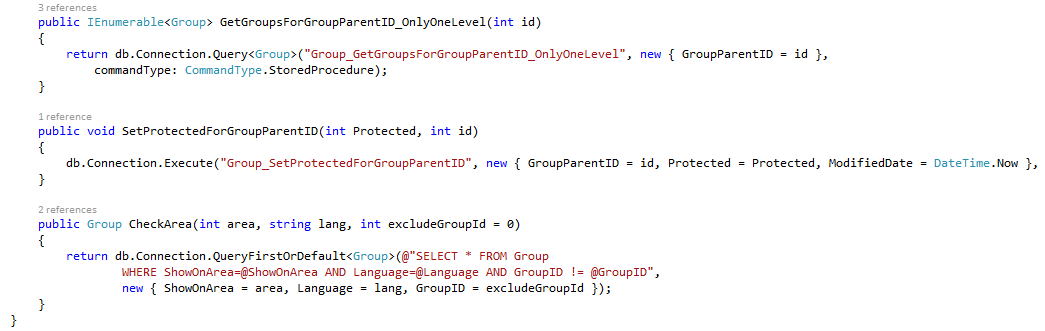
**- Sử dụng**

+ Ta có Entity Group, ta sẽ tạo class GroupRepository

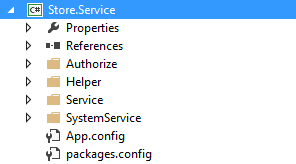
+ Đầu tiên Ta sẽ định nghĩa một Interface IGroupRepository kế thừa từ class IGenericRepository<Group>



+ Tiếp theo, tạo một class GroupRepository kế thừa GenericRepository<Group>, IGroupRepository. Ở đây bạn sẽ thấy T chính là Group. GroupRepository có thể sử dụng mọi phương thức của GenericRepository



### 2.2 Tầng Store.Service



#### 2.2.1 Folder Authorize

- Gồm các class ApplicationRoleStore, ApplicationUserStore, IdentityConfig dùng để xác thực user, role, account

- **Authorization:** dùng đểxác nhận những gì user có thể làm. Ví dụ: user được phép truy cập vào website, đăng thông tin lên web nhưng user không được phép truy cập vào trang admin.

#### 2.2.2 Folder Helper

- Chứa những class mở rộng: ví dụ như là dùng regular expression để làm việc với text-editor (Chắc là sẽ trùng với ý tưởng của tác giả)

**\* Regular expression** là 1 khuôn mẫu các ký tự dùng để so khớp và xử lý chuỗi ký tự đầu vào. Bằng cách định nghĩa các pattern (mẫu), regular expression (regex) cho phép extract, replace, remove... chuỗi 1 cách hiệu quả nhất.

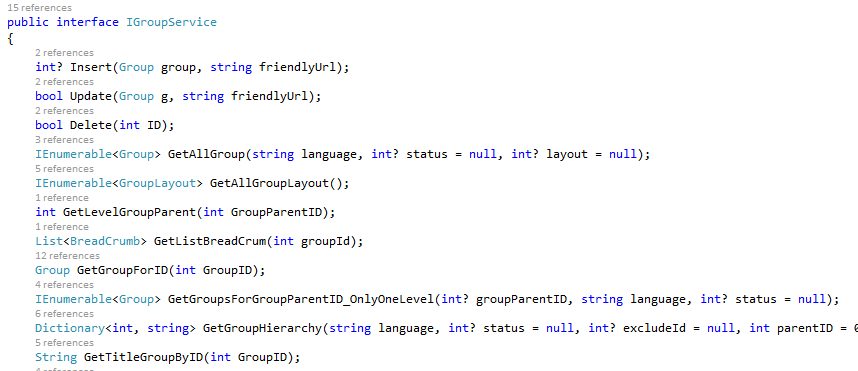
#### 2.2.3 Folder Service

**- Gồm các class Service:** dùng để thực thi các nghiệp vụ logic

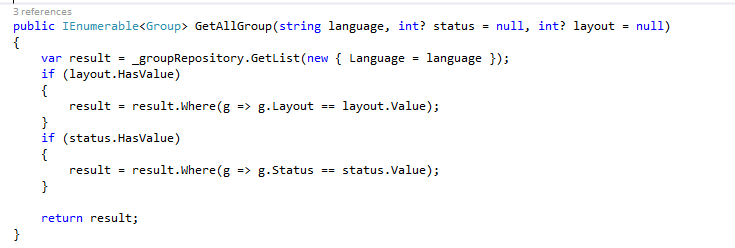
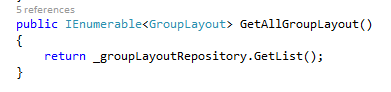
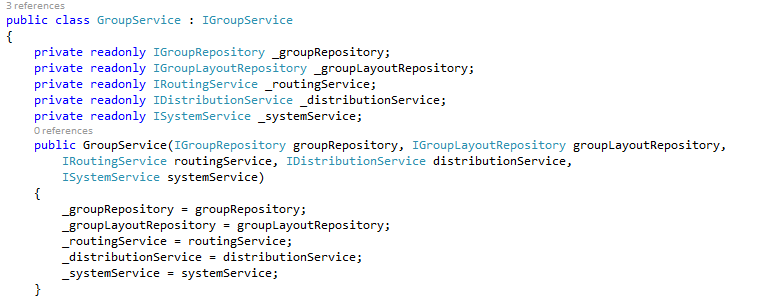
- Ví dụ:

+ Đầu tiên tạo 1 class service - GroupService

+ Ta sẽ định nghĩa một Interface IGroupService



+ Tiếp theo, tạo một class GroupService kế thừa IGroupService



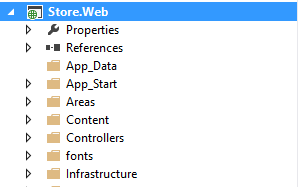
#### 2.2.4 Folder SystemService

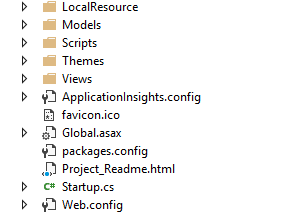
- Gồm các class RoutingService, ThemeService.

- Cấu trúc y như folder Service.

- Gồm các class service của hệ thống (Chắc ý tưởng của tác giả là muốn để riêng 2 file này để dễ quản lý – mỗi lần kiếm không phải mò)

### 2.3 Tầng Store.web





#### 2.3.1 Folder App\_Data

- Chứa các file cơ sở dữ liệu (nếu có)

#### 2.3.2 Folder App\_Start

- Chứa các lớp được chạy 1 lần khi ứng dụng bắt đầu

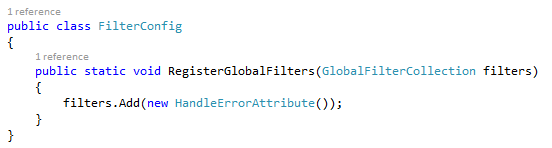
- Gồm có:

+ Class Bootstrapper: là nơi để đăng ký các class repository, service, identity, auto mapper

+ Class BundleConfig: Đăng ký các gói bundle mà mỗi bundle có thể có 1 hoặc nhiều files ( js hoặc css ).

+ Class FilterConfig

- Filtering là thành phần cho phép ta can thiệp vào quá trình xử lý các thông tin của request hay actions.



- Ở đây ta có 1 Filter có sẵn của hệ thống là “HandleErrorAttribute”. “HandleErrorAttribute” cho phép ta xử lý lỗi mà được ném ra từ Actions.

- Khi sử dụng Filter này, mặc định, khi 1 exception được ném ra hệ thống sẽ tự động gọi 1 view tên “Error” (“Error.cshtml”) nằm trong folder “Views/Shared” . Đây là lý do khi dùng “HandleErrorAttribute” ta cần phải tạo 1 view “Error.cshtml” trong folder “Views/Shared”.

+ Class RouteConfig: là nơi đăng ký Route

+ Class Startup.Auth

+ Class WebApiConfig

#### 2.3.3 Folder Area

- Chứa các Controller, Model, View,.. dành cho phần admin

#### 2.3.4 Folder Content

- Chứa các tài nguyên như image, css,..

#### 2.3.4 Folder Controllers

- Chứa tất cả các controller của ứng dụng

#### 2.3.5 Folder Infrastructure

- Folder Mapping gồm có:

+ DomainToFrontEndViewModelMappingProfile

+ DomainToViewModelMappingProfile

+ FrondEndViewModelToDomainMappingProfile

=> Là nơi đăng ký các class Model, ViewModel, Entity

=> Sử dụng cơ chế AutoMapper: AutoMapper là 1 thư viện mapping object-object. Nó cho phép copy giá trị từ 2 object có các property (thuộc tính) giống tên nhau - không phân biệt hoa thường. Hơn thế nữa, nó còn có thể map các property giống tên nhưng khác kiểu dữ liệu và cho phép bạn custom việc chuyển đổi giữa các property theo ý mình. Ví dụ, bạn có thể mapping một Enity Group tới một đối tượng thuộc lớp GroupViewModel hoàn toàn tự động bằng cách sử dụng AutoMapper.

#### 2.3.6 Folder LocalResource

- Dùng cho chức năng multiple language

#### 2.3.7 Folder Models

- Các lớp mô hình dữ liệu được tổ chức để nắm giữ và thao tác dữ liệu

#### 2.3.8 Folder Scripts

- Chứa javascript cần thiết của ứng dụng

#### 2.3.9 Folder Themes

- Custom Theme

#### 2.3.10 Folder Views

- Chứa các mẫu giao diện người dùng của ứng dụng

# IV. Bắt đầu thực hiện project

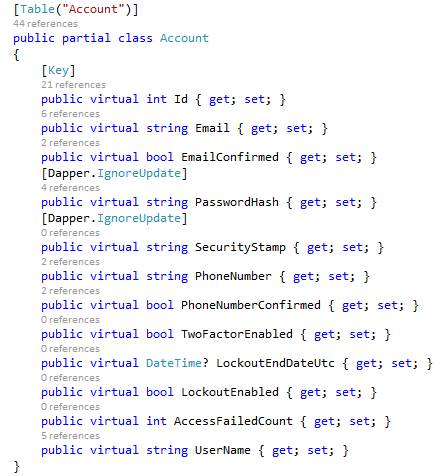
## 1. Thiết lập chuỗi kết nối

* Web.config (Trong tầng Store.Web)

C:\Users\son09\AppData\Local\Microsoft\Windows\INetCacheContent.Word\111.png

## 2. Thực hiện chức năng Create Account

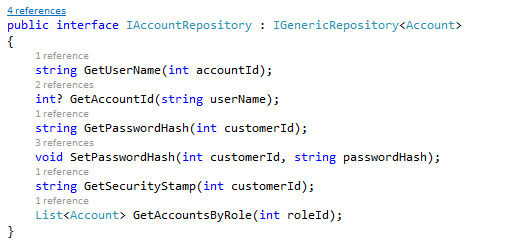
### 2.1 Tạo Class Account trong EntityModels



**- Chú ý:** Tên class và tên thuộc tính trong class phải trùng với tên table và thuộc tính của table trong database

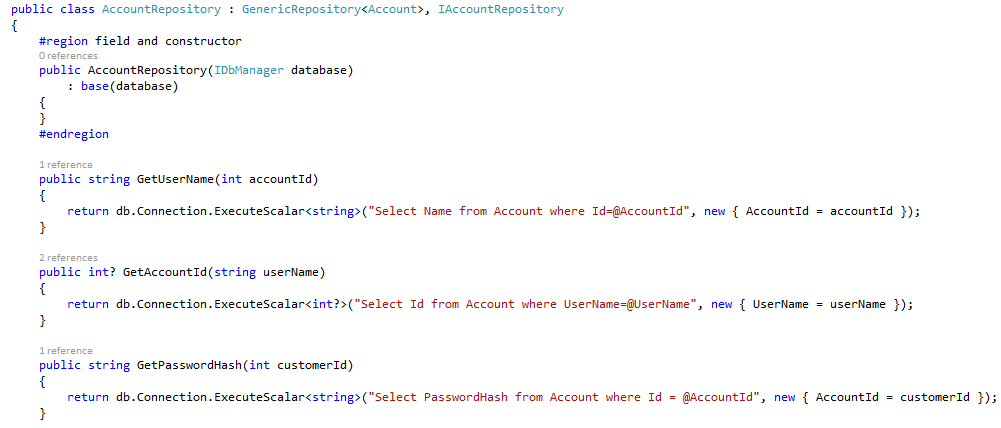
### 2.2 Tạo Class AccountResponsitory

- Đầu tiên Ta sẽ định nghĩa một Interface IAccountRepository kế thừa từ class IGenericRepository<Account>



- Tiếp theo, tạo một class AccountRepository kế thừa GenericRepository<Account>, IaccountRepository. AccountRepository có thể sử dụng mọi phương thức của GenericRepository

- Tiết đó viết mấy câu truy vấn dữ liệu cần thiết



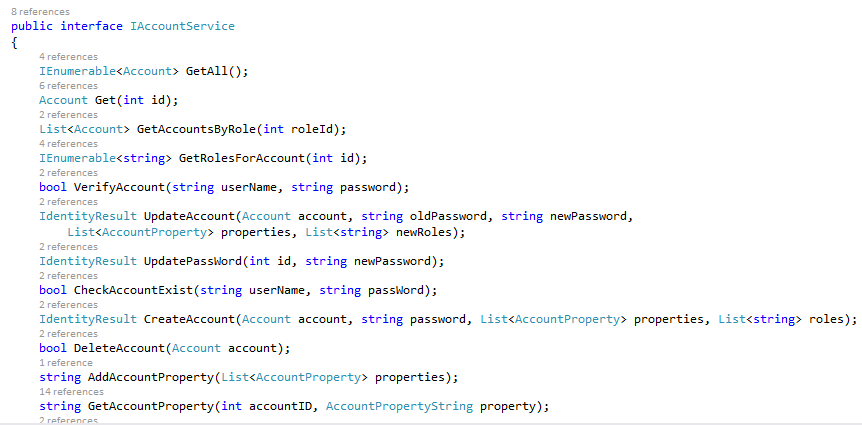


- Vào class Bootstrapper (trong folder App\_Start) để đăng ký responsitory AccountResponsitory

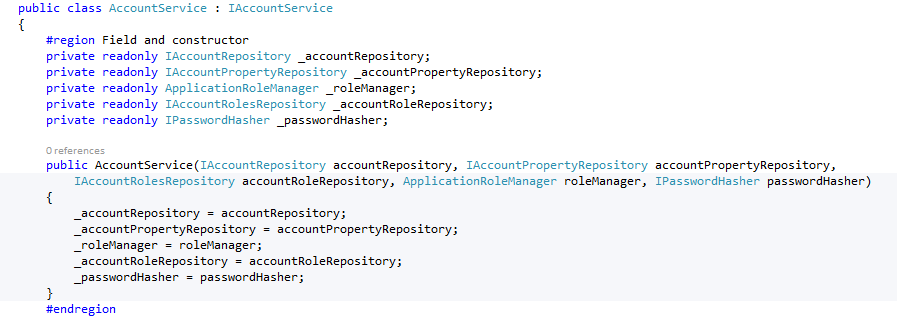
C:\Users\son09\AppData\Local\Microsoft\Windows\INetCacheContent.Word\5.png

### 2.3 Tạo AccountService

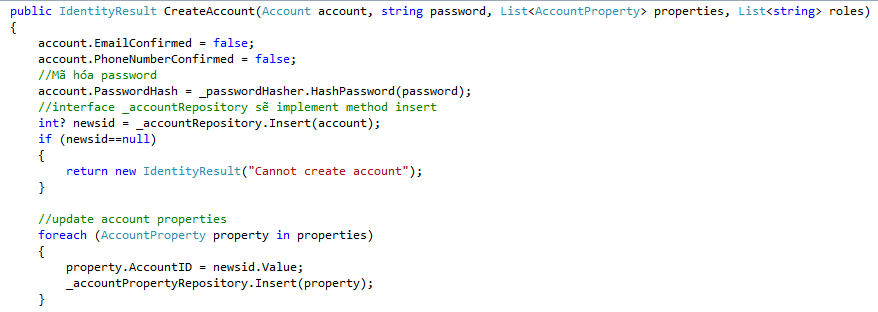
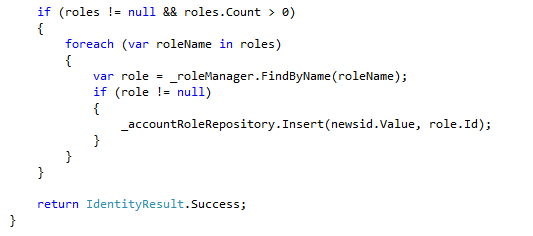
- Tạo 1 class AccountService trong tầng Store.Service

- Ta sẽ định nghĩa một Interface IAccountService

- Tiếp theo, tạo một class AccountService kế thừa IAccountService



- Tạo method CreateAccount



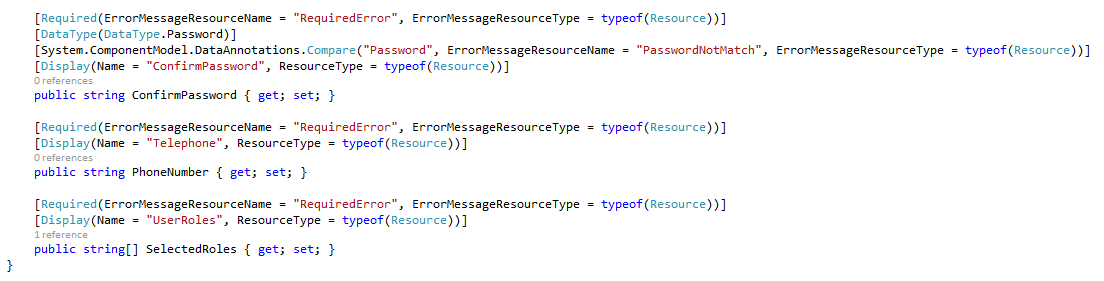
- Vào class Bootstrapper (trong folder App\_Start) để đăng ký service AccountService

C:\Users\son09\AppData\Local\Microsoft\Windows\INetCacheContent.Word\126.png

### 2.4 Tạo AccountDetailViewModel

- Tại tầng Store.Web, vào trong Folder Area rồi tạo 1 class AccountDetailViewModel trong Folder Models





- [Display(Name = "Fullname", ResourceType = typeof(Resource))]:

+ Thuộc tính Display: Sẽ hiển thị tên trường FullName ra ngoài view

+ Thuộc tính ResourceType: Ứng với Name = "Fullname" sẽ tương ứng với 1 FullName trong Resource mà mình đã thêm vào trong Resource. Dựa vào đó thì sẽ hiển thị cái Fullname tương ứng ra ngoài view

-[Required(ErrorMessageResourceName = "RequiredError", ErrorMessageResourceType = typeof(Resource))]

+ Thuộc tính Required: yêu cầu phải nhập vào trường đó.

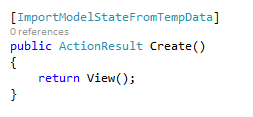
+ Thuộc tính ErrorMessageResourceType: Nếu không nhập hoặc nhập sai kiểu dữ liệu thì sẽ show ra ErrorMessage đã được định nghĩa trong Resource

- [StringLength(100, ErrorMessageResourceName = "StringLengthErrorMessage", ErrorMessageResourceType = typeof(Resource), MinimumLength = 5)]

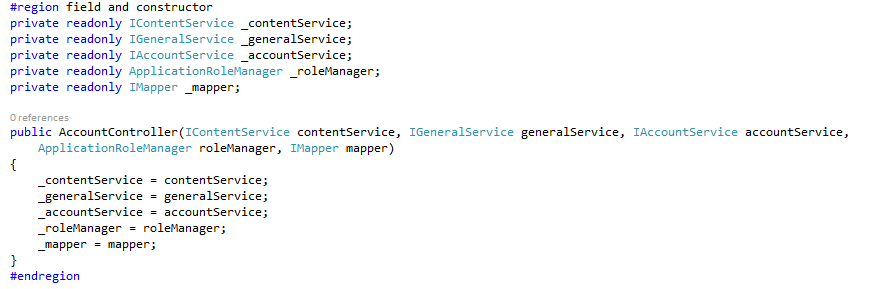
+ Thuộc tính StringLength: Cho biết độ dài ký tự tối đa được nhập vào là 100 ký tự.

+ Thuộc tính MinimumLength: cho biết phải nhập ít nhất 5 ký tự

### 2.5 Tạo Controller Account

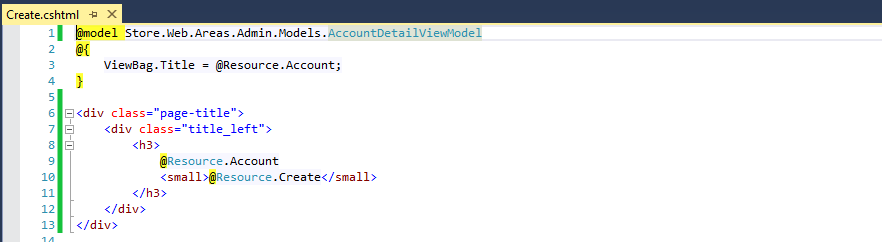
- Tại tầng Store.Web, vào trong Folder Area rồi tạo 1 Controller Account trong Folder Controller

- Sau đó tạo ra 1 Action Create

- Khởi tạo các interface service

### 2.6 Tạo View

- Tạo 1 view tương ứng với action Create này



Các tên hiển thị (title,..) sẽ được lấy trong Resource. Nếu như chưa định nghĩa trong Resource thì vào trong Resource thêm vào

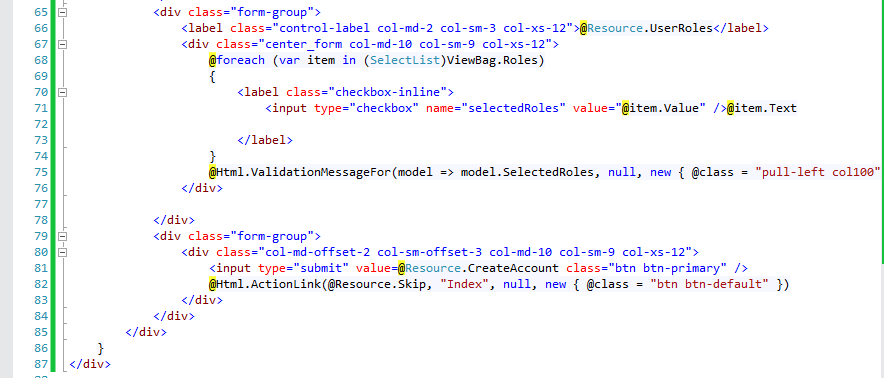


Dùng để chia sẽ dữ liệu từ Controller sang view

 dùng để kiểm tra dữ liệu của Property của Model

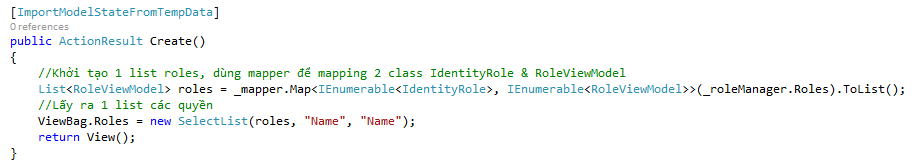
Tạo 1 hidden field (anti-forgery token) sử dụng trong form submit. Cái này liên quan đến security của website. Chặn request giả mạo

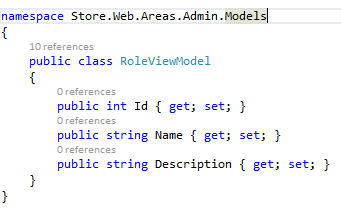
hiển thị error chung khi có lỗi



 Dùng để tạo link đến các Action khác cùng View hoặc khác View.

### 2.7 Tạo ViewBag.Roles

- Tạo 1 ViewBag.Roles trong action Create để gửi dữ liệu sang view Create

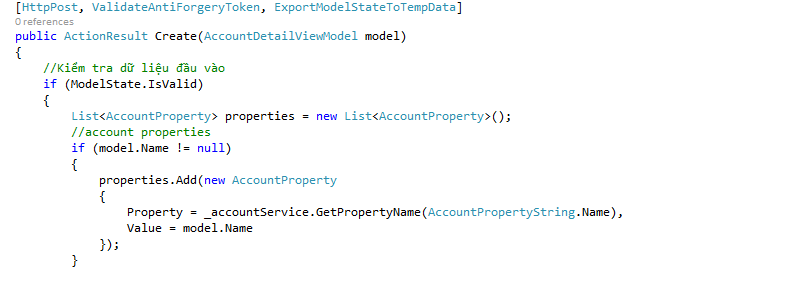
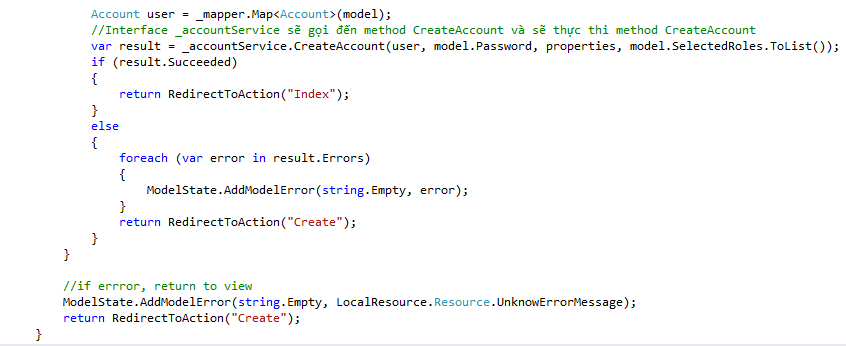
- Tại tầng Store.Web, vào trong Folder Area rồi tạo 1 class RoleViewModel trong Folder Models

C:\Users\son09\AppData\Local\Microsoft\Windows\INetCacheContent.Word\1000.png- Ta vào trong folder Mapping (trong Folder Infrastructure) vào class DomainToViewModelMappingProfile để đăng ký mapper:

* Sau khi làm xong ta sẽ được view

### 2.8 Tạo Action Create

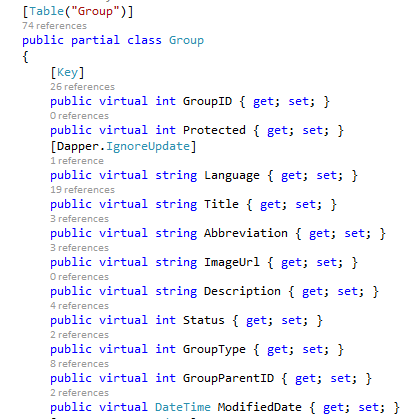
- Tạo action create có phương thức HttpPost để lấy dữ liệu từ phía client

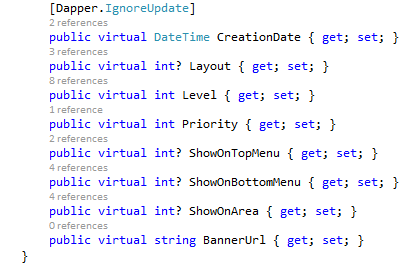


* Xong chức năng create Account

## 3. Thực hiện chức năng Create Group (Thêm mục nội dung)

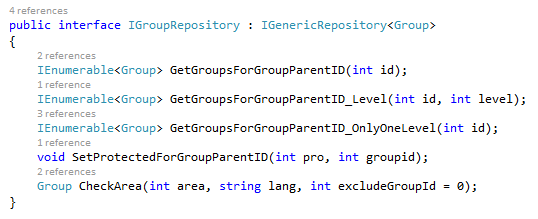
### 3.1 Tạo Class Group trong EntityModels





### 3.2 Tạo Class GroupResponsitory

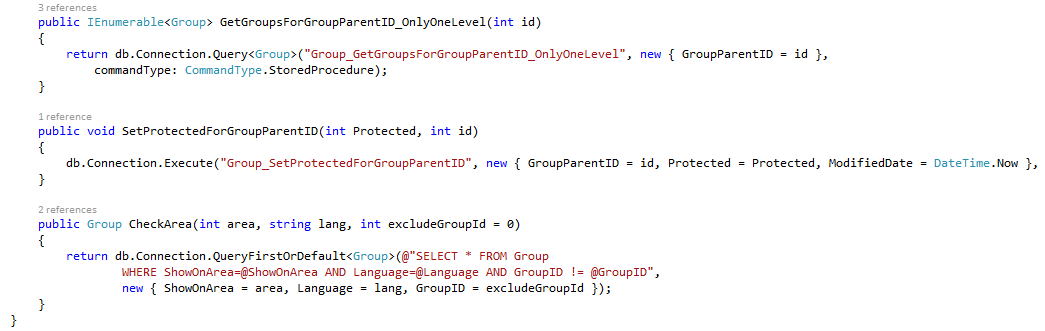
- Đầu tiên Ta sẽ định nghĩa một Interface IGroupRepository kế thừa từ class IGenericRepository<Group>



- Tiếp theo, tạo một class GroupRepository kế thừa GenericRepository<Group>, IGroupRepository. GroupRepository có thể sử dụng mọi phương thức của GenericRepository

- Tiếp đó viết mấy câu truy vấn dữ liệu cần thiết





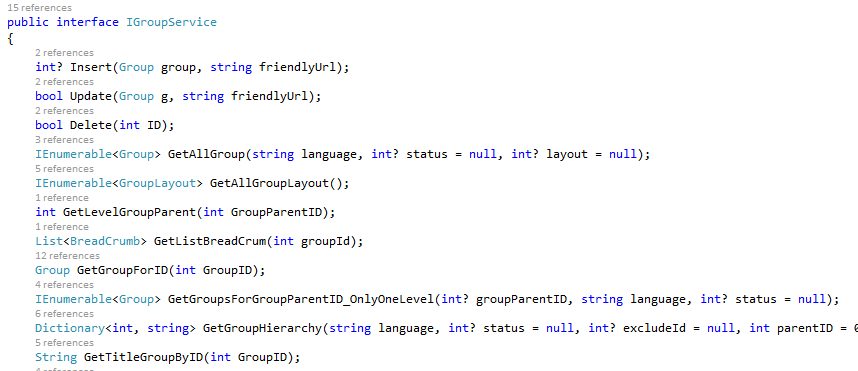
- Vào class Bootstrapper (trong folder App\_Start) để đăng ký responsitory GroupResponsitory

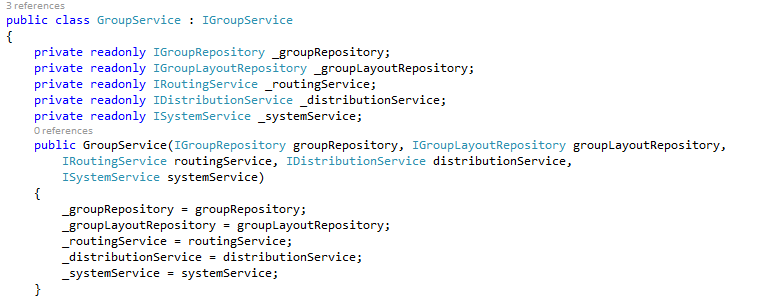
C:\Users\son09\AppData\Local\Microsoft\Windows\INetCacheContent.Word\10.png

### 3.3 Tạo GroupService

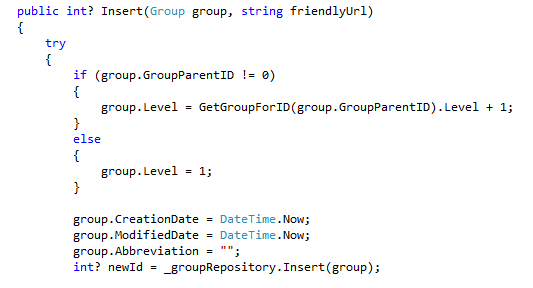
- Tạo 1 class GroupService trong tầng Store.Service

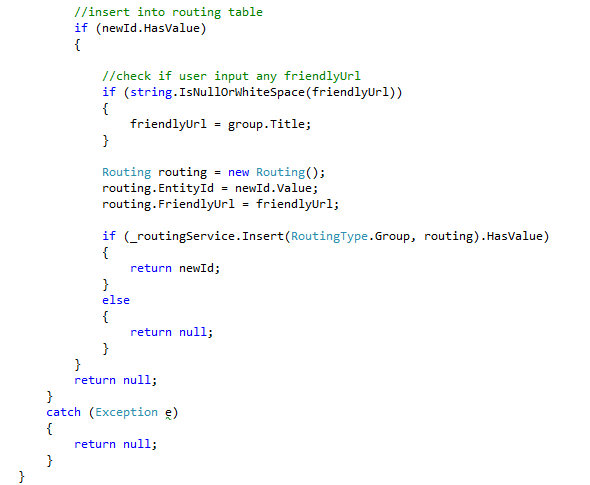
- Ta sẽ định nghĩa một Interface IGroupService



- Tiếp theo, tạo một class GroupService kế thừa IGroupService

- Tạo method Insert

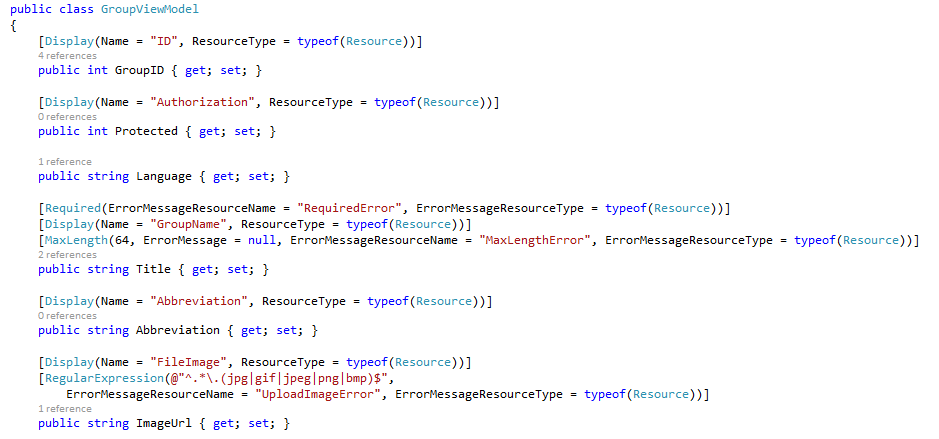




- Vào class Bootstrapper (trong folder App\_Start) để đăng ký service GroupService

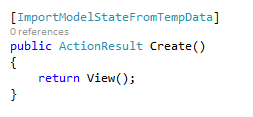
C:\Users\son09\AppData\Local\Microsoft\Windows\INetCacheContent.Word\10.png

### 3.4 Tạo GroupViewModels

- Tại tầng Store.Web, vào trong Folder Area rồi tạo 1 class AccountDetailViewModel trong Folder Models

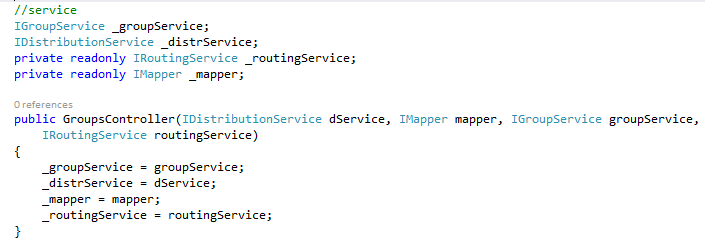
Kiểm tra định dạng file image

### 3.5 Tạo Controller Group

- Tại tầng Store.Web, vào trong Folder Area rồi tạo 1 Controller Group trong Folder Controller

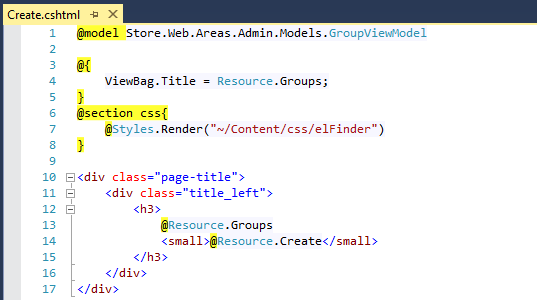
- Sau đó tạo ra 1 Action Create

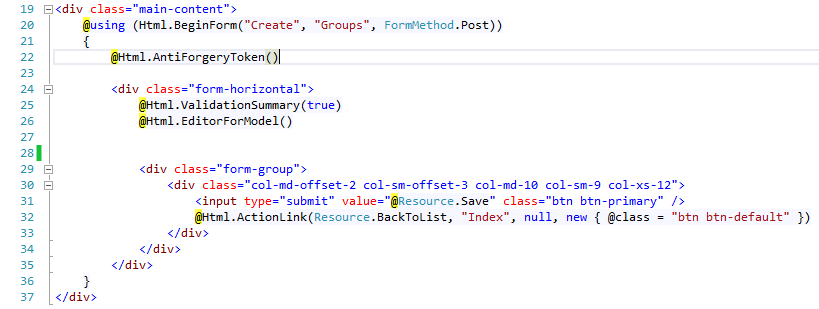
- Khởi tạo các interface service



### 3.6 Tạo View

- Tạo 1 view tương ứng với action Create này





**@section script** có tác dụng khi mình cần gọi js ở trang này nhưng không muốn trang sau cũng phải xử lý những js không cần thiết. Trong view \_Layout.cshtml bạn phải đặt một render section: **@RenderSection("Script", required: false)**

Tên phương phức

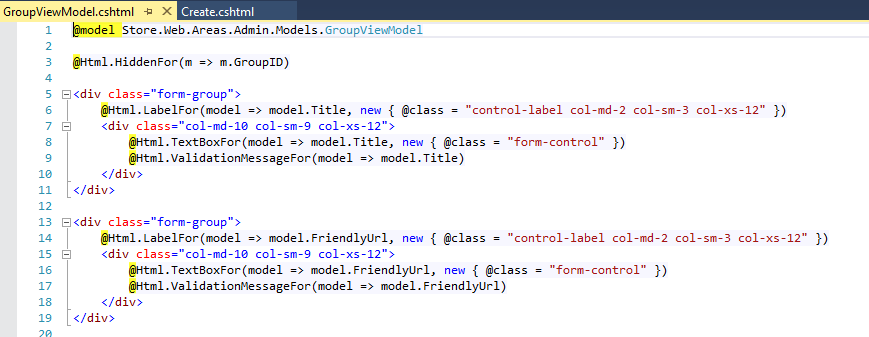
Tên controller

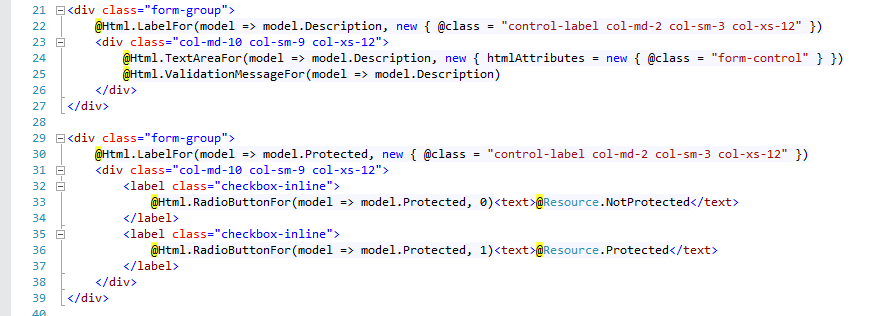
Tên action

* Chú ý: @Html.EditorForModel() được dùng để:
* Do các chức năng create, update, … có template giống nhau nên để tránh sự trùng lập, ta tạo 1 view dùng chung cho 2 chức năng này.
* Từ Folder View/Share/EditorTemplates, ta sẽ tạo 1 view GroupViewModel.cshtml
* C:\Users\son09\AppData\Local\Microsoft\Windows\INetCacheContent.Word\10.pngC:\Users\son09\AppData\Local\Microsoft\Windows\INetCacheContent.Word\11.pngChú ý nếu view Create này muốn gọi đến view GroupViewModel.cshtml thì trong mỗi view phải khai báo **@model** giống nhau

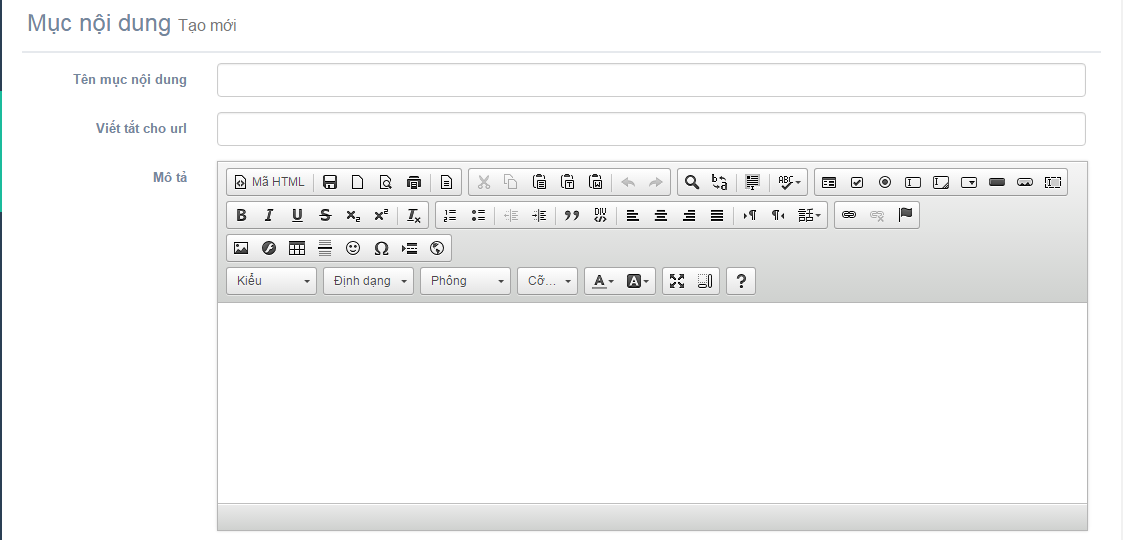
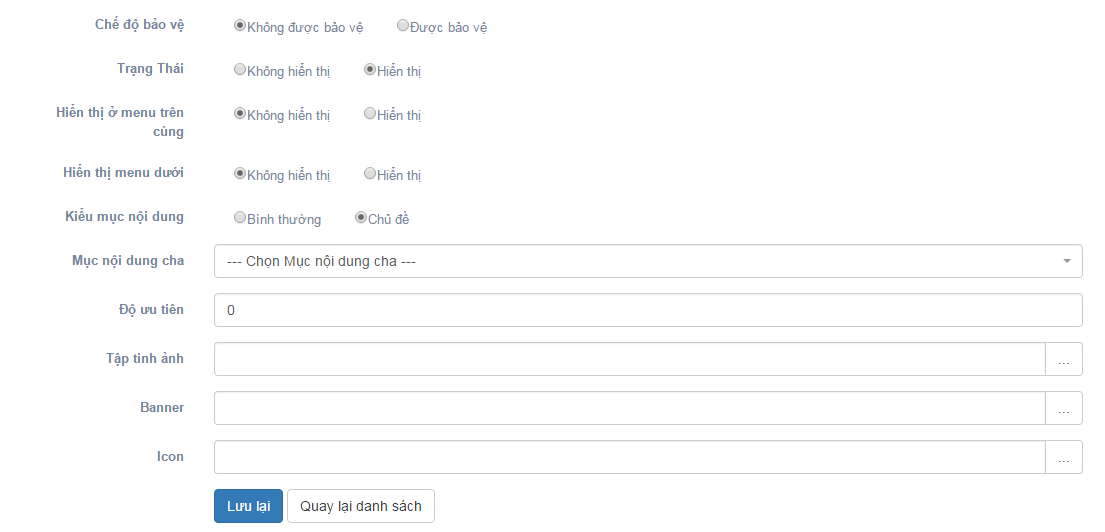
### 3.7 Tạo View GroupViewModel

- Như đã nói ở phần trên, mục đích của việc tạo view này là để dùng chung cho các action create, update,…



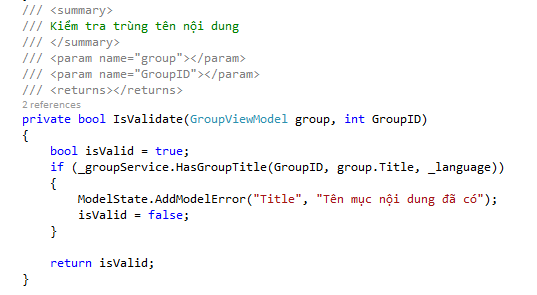


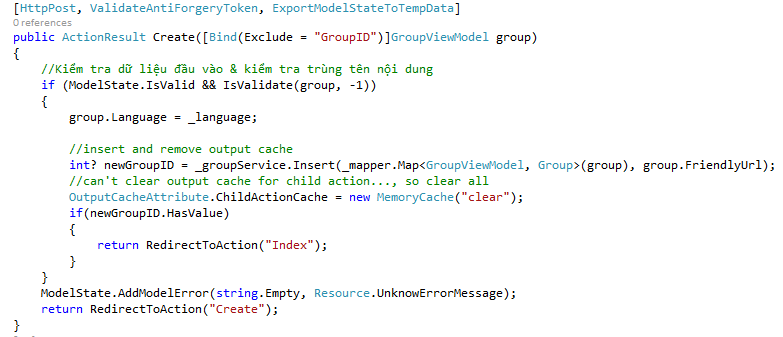
* Sau khi làm sau ta sẽ được view:



### 3.8 Tạo Action Create

- Tạo action create có phương thức HttpPost để lấy dữ liệu từ phía client





- Vào class FrondEndViewModelToDomainMappingProfile (trong folder Mapping) để đăng ký mapper. Map giữa 2 class GroupViewModel & Group

C:\Users\son09\AppData\Local\Microsoft\Windows\INetCacheContent.Word\13.png

* Xong chức năng create Group