Bài 13: Áp dụng Dependency Injection vào dự án sử dụng autofac.

* Bước 1 vào nuget cài 3 thư viện autofac,autofacmvc,autofacwebapi2
* Tạo file startup (Owin startup class)
* Debug PostCategoryService
* Cài đặt NuGet Microsoft.Owin.Host.SystemWeb
* Sửa lại postcategorycontroller GetAll method

Add dependence inject by autofac to web.

Bai 14: Sử dụng AutoMapper để map đối tượng

* Cài AutoMapper 4.2..1
* Tạo tầng ViewModel
* Tạo thư mục Mappings

+ Tạo class AutoMapperConfiguration.cs

+ Khai báo AutoMapperConfiguration vào Global.asax

* Tạo Class EntityExtension ( tìm hiểu extension method )

\*\* Giúp đẩy giá trị trừ ViewModel vào Model chính

* Khởi tạo Mapper vào API PostCategoryController ( get, add,update

**Bài 15: tích hợp asp.net identity để chứng thực người dung**

\*\*Identity là một cơ chế xác thực và quản lý người dung mới nhất của Microsoft dành cho ứng dụng ASP.net, tiền thân của ASP.Net Identity là Membership

Đặc điểm : là cơ chế dung chung cho tất cả các ứng dụng Web bao gồm ASP.Net MVC, Web API, WebForm và SignalR, dễ dàng them mới các trường dữ liệu khác vào user, dễ dàng unit test, quản lý quyền, hỗ trợ login với các Social dễ dàng, độc lập với web vì sử dụng cơ chế OWin, cài đặt từ Nuget

Các bước thực hiện cài đặt DI Autofac

1. Cài đặt 3 thư viện EntityFramework, Core, Owin
2. Tạo mới class User kết thừa từ IdentityUser
3. Tạo mới Role kết thừa từ IdentityRole
4. Kết thừa lớp DbContext từ dentityDbContext<User>
5. Thực hiện Migration vào database
6. Tạo class quản lý authen

Bước 1 > tạo một project mới để lấy mẫu Identity của ASP.Net > cài đặt thư viện > trong project Model tạo model ApplicationUser : IdentityUser, them mới một số properties > sửa trong project Data DBContext kết thừa từ IdentityDbContext, them phương thức tạo mới chính nó > trong project Web tạo class IdentityConfig trong thư mục App\_Start, cần 2 phương thức application SignIn và User > trong folder app\_start tạo mới một class OwinStartup, Startup.Auth > Migration add-migration Intergrate-aspnetIdentity > them vào DBContext builder.Entity > update-database lại > trong file startup.cs them configureAuth > Tron thư mục Api tạo AccountController > triển khai dependence cho các class mới trong Startup.cs > viết thử một service để test class mới AccountController > Seed sample data trong project Data > update database > có bug sửa lại phương thức trong dbContext > add-migration Changekey> điều chỉnh accountController thanh có [RoutePrefix] Login > có bug không thể test bằng postman, them mới và kiểm tra và cấu hình lại ApplicationUserStore trong IdentityConfig của thư mục App\_Start > Commit lên git add asp identity.

**Bài 16:** tổng quan về AngularJS và cách cài đặt sử dụng Bower > tạo file BowerConfig trở đường dẫn về Script> plugin > cài đặt angular 1.5.5 > tạo AdminController > tạo một view và gán angular vào

Bai 21: dựng cấu trúc cho phần quản trị với Single Page Application

Bài 22: Triển khai routing cho sử dụng Angular UI Router

Router là một cơ chế rất quan trọng trong ứng dụng SPA

Nó độc lập với cơ chế điều hướng của ASP.NET MVC

Nó tự động sử dụng view tương ứng với URL

Trong AngularJS:

* ngRoute: là một module core của AngularJS giúp điều hướng trong các kịch bản đơn giản.
* Ui-router: được phát triển bởi cộng đồng, khắc phục và them các tính năng mạnh mẽ cho ng-Route

Ui-router: cho phép view lồng nhau. Điều này là rất hưu ích với ứng dụng lớn, nơi bạn có thể có các trang mà kế thừa từ các phần khác, cho phép

Thực hành :

Bước 1 : Cài đặt Angular-ui-router : 0.2.18 > View trang quản trị > download template AdminLTE > Chép file dist của template vào Assets Admin > chép html template vào view index.html, thay đổi đường dẫn các file dist > sử dụng bower để them các thư viện mà template cần (jquery 2.2.3), bootstrap 3.3.6, slimScroll , fastclick > test nếu lỗi , xóa code trong bundleConfig > nhúng angularJS và UI-Router > đổi tên file js products.router.js thành products.module.js trong components > them homeController.js vào thư mục components > tạo thự mục modules trong shared > tạo common.js > khai báo app.js khai báo để tiến hành routing > tạo module common.js > khai báo product.module.js > khai báo homeController.js > khai báo angular cho html và them các đường dẫn của controller.js cho html.

Bước 2 : triển khai ui-router

Add thư mục home vào components > tìm phần html thân tách riêng > chỉ định khu vực để ren giao diện ra sử dụng ui-view của ui-router > sử dụng ui-sref để chuyển trang cơ chế ui-router

Bài 23 : binding dữ liệu từ WebAPI ra bảng trong HTML bằng AngularJS

Thực hành: tạo them service ProductCategoryService > tạo API ProductCategoryController > tạo them ViewModel > test API, seed them dữ liệu > Mapping cho view model mới > tạo apiService cho SPA > nhúng API vào view chính > tạo mới productCategoryListView và Controller, tạo mới module cho ProductCategory > include vào app chính > nhúng vào view chính > thay đổi HomeView > tạo productCategoryListView > khai báo ProductCategoryListController, khai báo một phương thức để lấy Data từ server, inject APIService > sử dụng ng-repeat triển khai trong productCategoryListView

Bài 24: Sử filter trong AngularJS

Thực hành > update table ProductCategory set CreatedDate > sử dụng filter để định dạng lại ngày > tạo thưc mục filter, tạo một statusFilter angularJS custom > nhúng filter vào view > sử dụng ng-class để gán điều kiện > thêm cột thao tác cho table front end > tìm icon phù hợp trên fontawesone

Bài 25: Sử dụng directive để tạo phân trang cho list

Các bước chính :

-Backend : tạo class generic chứa thông tin paging trả về, áp dụng vào phần trang cho webAPI sử dụng LINQ

-Frontend: viết mới một pager directive để phân trang, áp dụng directive vào để phân trang

Thực hành: tạo class PaginationSet vào thư mục Core > tạo phân trang vào class API ProductcategoryController vào phương thức getAll > test API bằng postman > tạo thư mục directives và pagerDirective angular > tạo một pagerdirective.html > nhung pân trang vào productCategoryListController >khai báo và nhúng vào view > commit paging for list