ASP.NET WEB API ANGULAR JS IN VS 2015 CTP VNEXT

Author: David Lizhong Huang

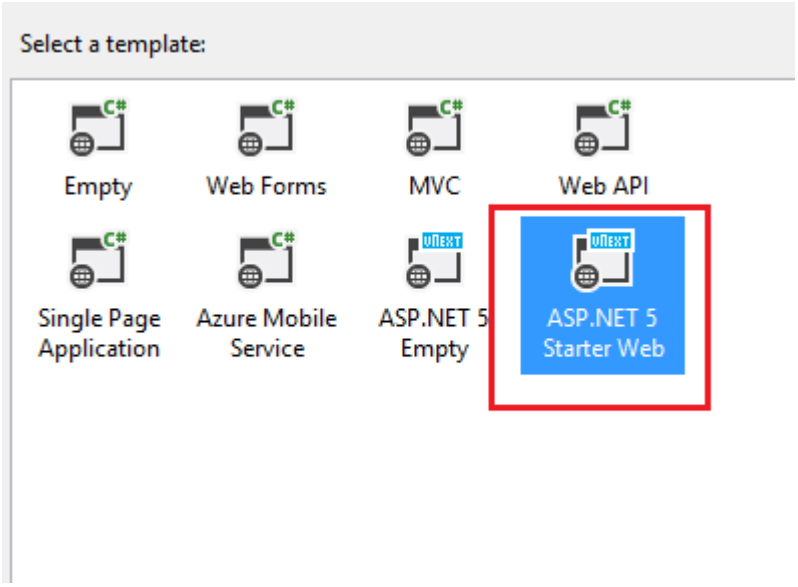
Date: 10, Feb. 2015

1, INTRODUCTION

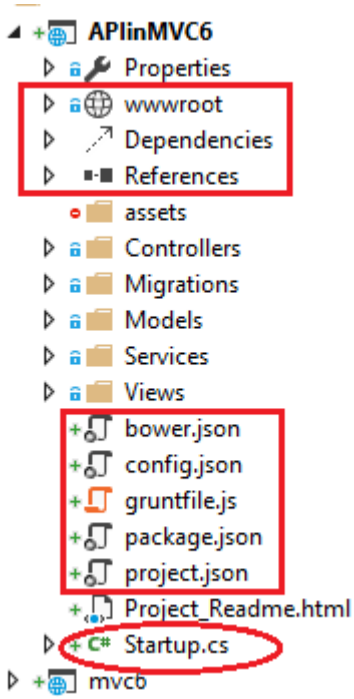
Dependency Injection in VNext VS 2015 will allow the data service in VS 2015 easy to be consumed by Web API web service. Web API web service then can be easily called via Http service. Those data services connect to SQL database via DbContext object and then are injected into MVC application. This demo project will show you how to implement those designs in VS 2015.

2, GET STARTED

1, Create a new asp.net5 starter web project in VS 2015 as below



This template create brand new asp.net web application templates in application as below example



Asp.net 5 does not contain web.config we can configure the web. It uses startup class to boot application similar to the bootstrap class in WPF application. The example startup.cs file is as below

```
    public class Startup
    {
        public Startup(IHostingEnvironment env)
        {
            Configuration = new Configuration()
                .AddJsonFile("config.json")
                .AddEnvironmentVariables();
        }

        public IConfiguration Configuration { get; set; }

        public void ConfigureServices(IServiceCollection services)
        {
            // Add EF services to the services container.
            services.AddEntityFramework(Configuration)
                .AddSqlServer()
                .AddDbContext<ApplicationDbContext>();

            // Add Identity services to the services container.
            services.AddIdentity<ApplicationUser, IdentityRole>(Configuration)
                .AddEntityFrameworkStores<ApplicationDbContext>();

            // Add MVC services to the services container.
            services.AddMvc();

            services.AddSingleton<IProductService, ProductService>();

        }

        public void Configure(IApplicationBuilder app, IHostingEnvironment env,
ILoggerFactory loggerfactory)
        {

            loggerfactory.AddConsole();
```

```
          if (string.Equals(env.EnvironmentName, "Development",
StringComparison.OrdinalIgnoreCase))
            {
                app.UseBrowserLink();
                app.UseErrorPage(ErrorPageOptions.ShowAll);
                app.UseDatabaseErrorPage(DatabaseErrorPageOptions.ShowAll);
            }
            else
            {
                app.UseErrorHandler("/Home/Error");
            }

            app.UseStaticFiles();

            app.UseIdentity();

            app.UseMvc(routes =>
            {
                routes.MapRoute(
                    name: "default",
                    template: "{controller}/{action}/{id?}",
                    defaults: new { controller = "Home", action = "Index" });

            });
        }

    }
```

If we have a WPF Prism 4.0 design pattern experience, we can easily pick up such a concept now has been used in asp.net. WPF bootstrap.cs class used DI to inject dependency in bootstrap class. VNext 5 do the same job as WPF do such as the DI container is used to inject services via configuration in this object. So when application starts, this startup object hook up all dependencies this application needed such as error and log components hookup.

2, Create Data Service

Asp.net 5 starter web uses Code first mechanism to generate database from code via the helps from K ef command. K command is a new tool VNext uses to do code first data migration and create database in SQL server. K ef command looks up the project.json and DbContext class in the root path of web application, run

K ef migration add initialize

K ef migration apply

The DbSet<object> database then is created baed on the connection string in config.json file that is added by startup object as below

```
// Add EF services to the services container.
Services
.AddEntityFramework(Configuration)
.AddSqlServer()
.AddDbContext<ApplicationDbContext>();.
```

When services try to initiate ApplicationDbContext object , it will decide if needs to create a new database as the code in this object's constructor as below

```
  public ApplicationDbContext()
        {
            if (!_created)
```

```
            {
                Database.AsMigrationsEnabled().ApplyMigrations();
                _created = true;
            }
        }
```

Simple, is n't it?

After `ApplicationDbContext` has been created, we can then create data service class to consume this context object which can bring data in SQL database to data service that then can handle those data in its own object such as an example below

```
public class ProductService :IProductService
    {
        private readonly ApplicationDbContext _db;
        public ProductService(ApplicationDbContext db)
        {
            _db = db;
        }
        public IEnumerable<Product> AllProducts()
        {
             return _db.Products;
        }

    }
```

This data service exposes its interface as below

```
    public interface IProductService
    {
       IEnumerable<Product> AllProducts();

    }
```

Now in order to use this product service in web api controller, we need to inject this service into Web Api controller such as the code below

```
  [Route("api/[controller]")]
    public class ProductController : Controller
    {

        APIinMVC6.Services.IProductService _ps;
        public ProductController(APIinMVC6.Services.IProductService ps)
        {
          _ps=ps;
        }

        // GET: api/values
        [HttpGet]
        public IEnumerable<Product> Get()
        {
            return _ps.AllProducts ();

        }

       ……
```

As we know in asp.net MVC 5 application, we need to create controllerfactory.dll or use third party DI container to register this data service before it can be injected

into the controller's constructor. In VNext VS 2015, we know DI container is built in. So we can simply register this data service in startup.cs DI container as below

```
services.AddSingleton<IProductService, ProductService>();
```

Then we can inject this data service in api controller default constructor.

3, Run Asp.Net Web Api

Now we run the application and put url below into browser, Route in startup.cs file can automatically bring all product details as JSON data to client UIs.

http://localhost:5000/api/product

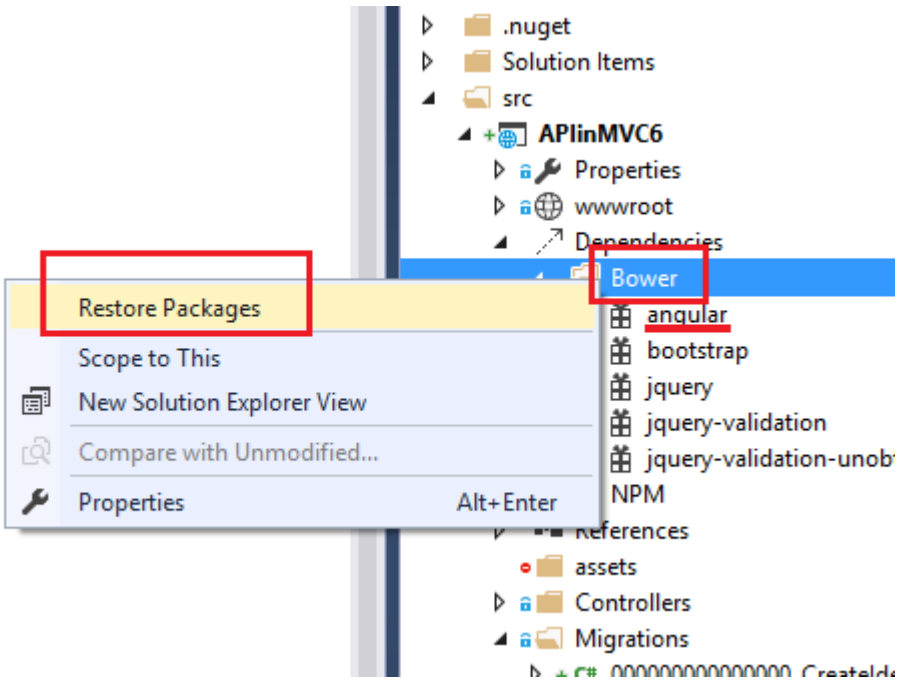
4, Consume this data service in angular js

Now we build up a simply data service. We need to use angular JS to call this asp.net Web API via url http://localhost:5000/api/product

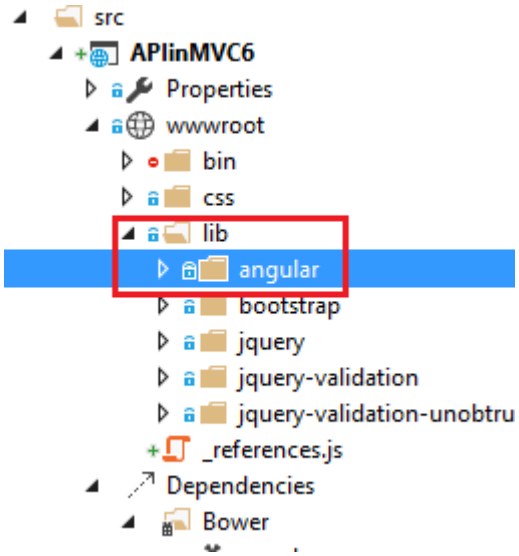
At first, we need to include angularjs in product.json configuration file as below

```
{
    "name": "APIinMVC6",
    "private": true,
    "dependencies": {
        "bootstrap": "~3.0.0",
        "jquery": "~1.10.2",
        "jquery-validation": "~1.11.1",
        "jquery-validation-unobtrusive": "
        "angular": "^1.3.12"
    },
    "exportsOverride": {
        "bootstrap": {
            "js": "dist/js/*.*",
            "css": "dist/css/*.*",
            "fonts": "dist/fonts/*.*"
        },
        "jquery": {
            "js": "jquery.{js,min.js,min.ma
        },
        "jquery-validation": {
            "": "jquery.validate.js"
        },
        "jquery-validation-unobtrusive": {
            "": "jquery.validate.unobtrusiv
        }
    }
}
```

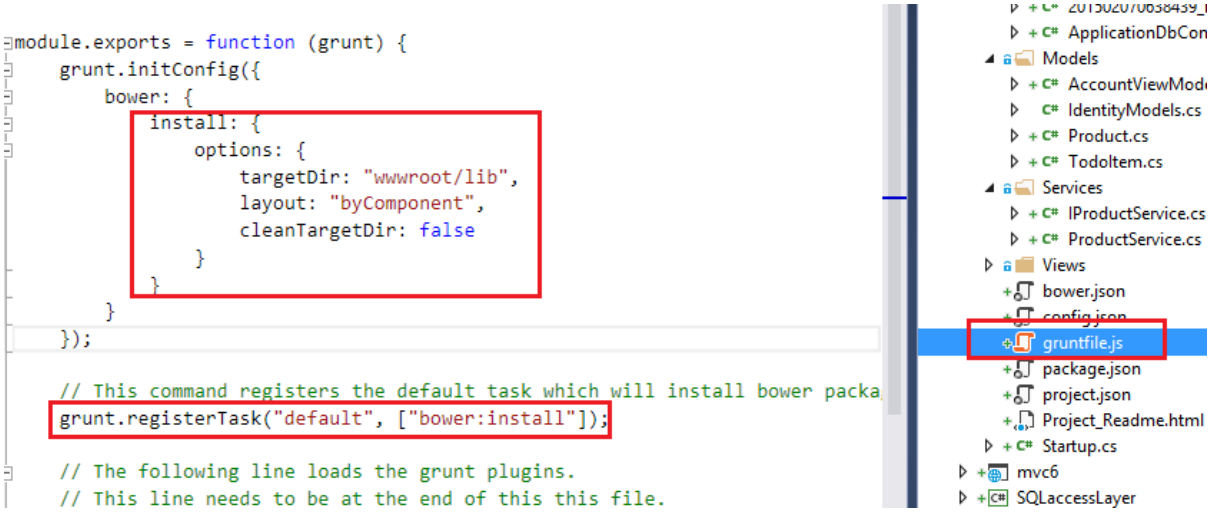
After we add “angular” in json file, go to bowser folder and select restore packages as below



Restore package command in bower folder will download angular js package from internet into application. Finally, after download, you need to upload angular js files to wwwroot/lib folder as below



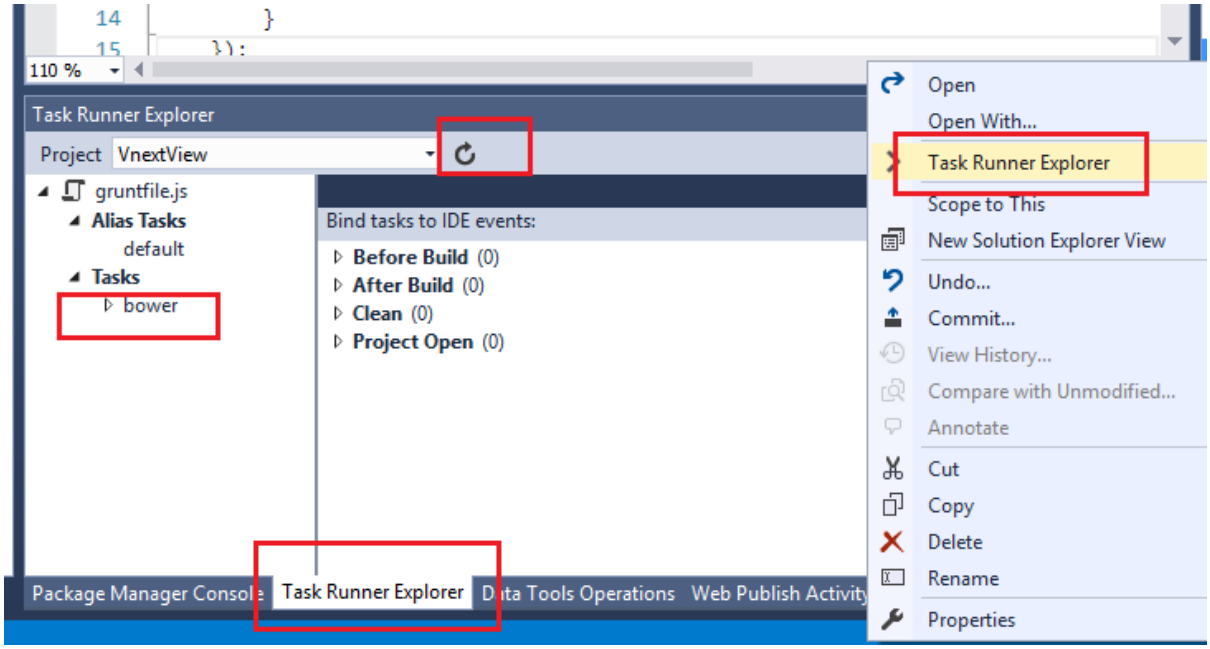
How can we do this? Use gruntfile.js in the root folder as below.



Right click gruntfile and select Task Runner Explorer, and then right click “bower” folder and select Run to run the task. Grunt will run automated task to upload or install angular js in Lib folder. Now we can angular js folder has been uploaded to lib folder.

5, Put Anglar JS in index.cshtml

Now we configure angular in Index.cshtml page as below




```
@{
    ViewBag.Title = "Home Page";
}


<div ng-app="PtApp">
    <div ng-controller="PshowCtrl">
        <ul ng-repeat="item in items">
            <li>{{item.Id}}</li>
            <li>{{item.Name}}</li>
            <li>{{item.UnitPrice}}</li>
            <li>{{item.Desc}}</li>


        </ul>
    </div>

</div>
```

You can see we do not put any js reference here, this is because we put those in _layout.cshtml page as below

```
<!DOCTYPE html>
<html>
<body>
    <div class="container body-content">
        @RenderBody()
        <footer>
            <p>&copy; @DateTime.Now.Year - My ASP.NET Application</p>
        </footer>
    </div>
    <script src="~/lib/jquery/js/jquery.js"></script>
    <script src="~/lib/bootstrap/js/bootstrap.js"></script>
     <script src="~/lib/angular/angular.js"></script>
    <script src="~/lib/angular/app.js"></script>
    @RenderSection("scripts", required: false)
</body>
</html>
```

Wwwroot is the root directory this MVC web application owns, therefore, the source url will use “~” ro replace wwwroot folder.

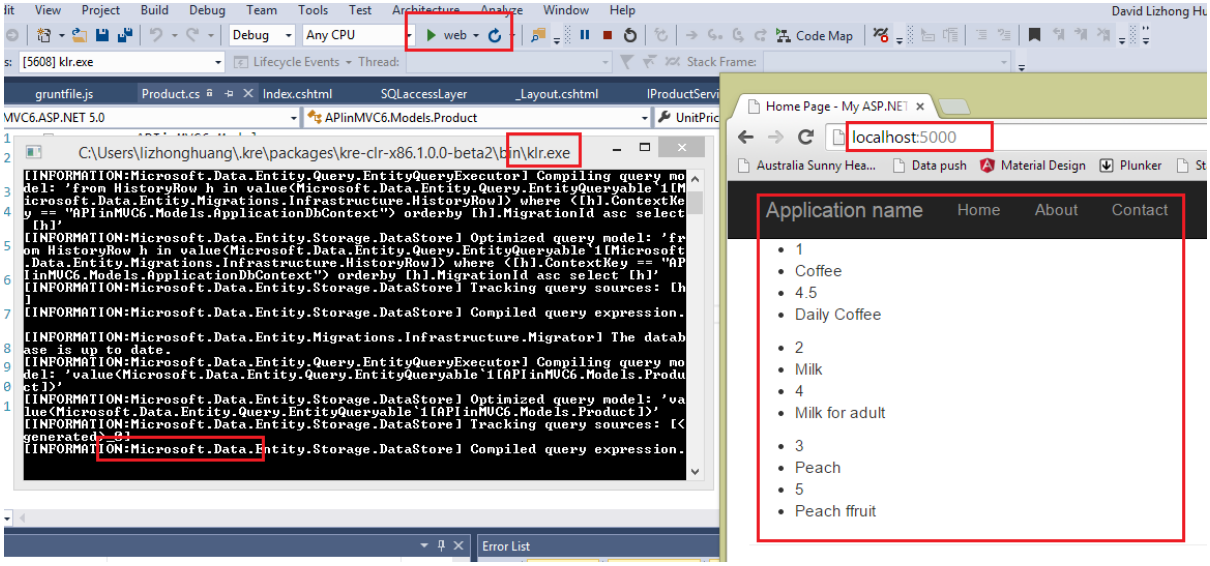
Now the last thing is to create a new app.js file in lib folder and put angular js code in as below example shown

```
'use strict';

angular.module('PtApp', [

]).controller('PshowCtrl',
    function ($http, $scope) {
        $scope.items = [];
        $http.get('api/Product')
            .then(function (result)
            {
                $scope.items = result.data;
             });
    });
```

Run application home/index.cshtml, we get the expected result as below



3, SUMMARY

So we can make sure that VNext VS2015 is a good IDE can do front and back end development in one stop. Built-in DI container will allow us to inject data services into Web API controllers that can be called by angular JS http service. Grunt and bower js framework can be used to minify and install JS and Css files in specifical folder automatically.