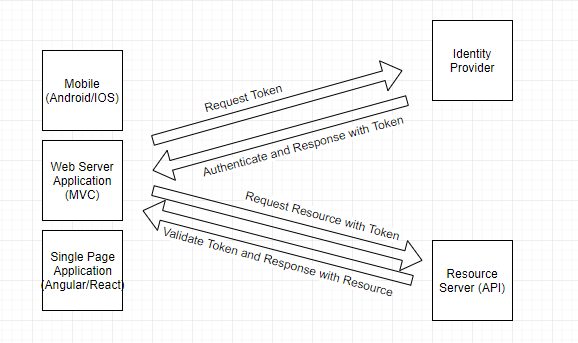
**Securing an ASP.Net MVC Application Which Uses a SECURE API and Use JWT Token to Enable Single Sign On**

To achieve SSO, first, we’ll need a centralize API to serve as an Identity Provider to authenticate and/or authorize user by granting user a token to access the resource from Resource Server. Later on, we can share the token among different client applications while navigating across application(s). The overall big picture is as per diagram below.



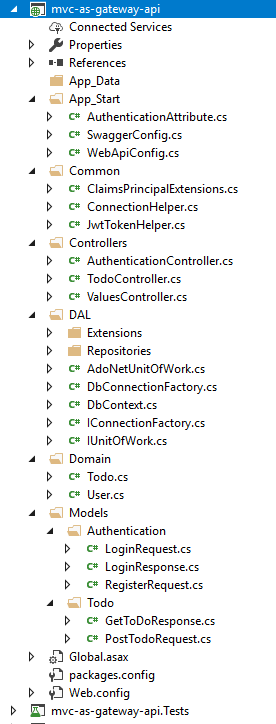
To achieve the POC with minimum effort, I’ve created two Web Application:

1. mvc-as-gateway-api : combination of JWT identity provider and resource server, separated by route.
2. mvc-as-gateway-web : the server side of the web app is basically serve as a gateway to call api or microservices.

In real life scenario, the identity provider should be separated and we can use some ready made solution like IdentityServer, OAuth, GoogleAuth etc.

**mvc-as-gateway-api**

The Api project is basically being broken down into the structure below. What I’ve added on top of the default template are basically swagger (<https://swagger.io/>), Jwt Token Generation & Validation, DAL & Domains.



App\_Start – Middleware layer to configure the app for routing, dependency injection, intercepts events etc.

Common – Some commonly used helper Class.

Controller – the main entry point of the API and is used to control the flow of application execution.

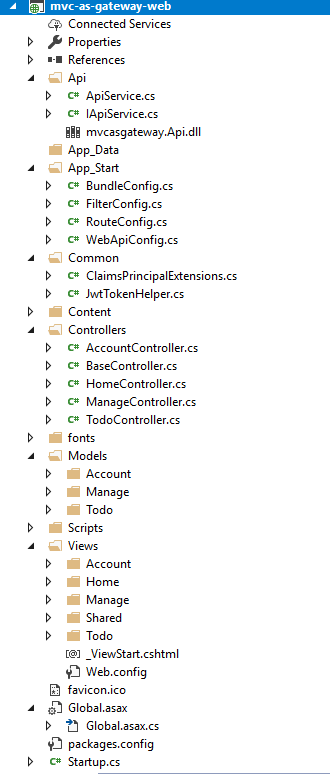
DAL – Data Access Layer in Repositories Pattern.

Domain – The Domain Model, typically translated directly from the Table/Document from the database and to be used during ORM, similar to the DataContext if you’re using Entity Framework.

Models – To group classes that is used as a DTO (Data Transfer Objects), for API request and response.

**mvc-as-gateway-web**

The Web project is basically being broken down into the structure below. What I’ve added on top of the default template are basically the Api Service Layer, Jwt Token Helper, authenticate using API and a simple Todo module to make sure the token obtained from Identity Provider can works in API.



Api – The place to put all 3rd part services SDK, interface and implementation of the SDK.  
App\_Start – Middleware layer to configure the app for routing, dependency injection, intercepts events etc.

Common – Some commonly used helper Class.

Content – Template’s bootstrap css.

Controller – The main entry point of the API and is used to control the flow of application execution.

fonts – Template’s font file.

Models – To group view model class to holder properties that is required for the Views.

Scripts – Templates’s script file.

Views – To group view (cshtml) file which displays data and handle user interaction.

**Setup & Running**

**Prerequisite**

1. .net 4.6
2. Visual studio 2015 & above
3. Microsoft SQL Server

**Database Setup**

Create a database with any name, then run the following script on the database.

CREATE TABLE [dbo].[Users](

[UserID] [int] IDENTITY(1,1) NOT NULL,

[FirstName] [nvarchar](25) NULL,

[LastName] [nvarchar](25) NULL,

[UserName] [nvarchar](50) NULL,

[Password] [nvarchar](25) NULL,

[IsActive] [bit] NULL,

[IsDeleted] [bit] NULL,

[CreatedBy] [int] NULL,

[CreatedAt] [datetime] NULL,

[UpdatedBy] [int] NULL,

[UpdatedAt] [datetime] NULL,

[Email] [nvarchar](50) NULL,

PRIMARY KEY CLUSTERED

(

[UserID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

CREATE TABLE [dbo].[Todos](

[Id] [int] IDENTITY(1,1) NOT NULL,

[Name] [nvarchar](50) NOT NULL,

[UserId] [int] NOT NULL,

[CreatedBy] [int] NOT NULL,

[CreatedAt] [datetime] NOT NULL,

[UpdatedBy] [int] NULL,

[UpdatedAt] [datetime] NULL

) ON [PRIMARY]

INSERT INTO [dbo].[Users]([FirstName],[LastName],[UserName],[Password],[IsActive],[IsDeleted],[CreatedBy],[CreatedAt],[UpdatedBy],[UpdatedAt],[Email])

VALUES('Admin','Baron', 'admin@admin.com', 'abcd1234', 1, 0, null, getutcdate(), null, null, 'admin@admin.com')

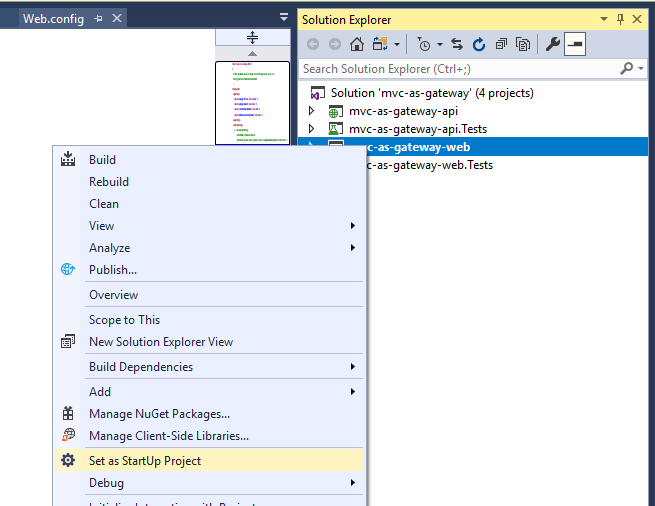
Open the mvc-as-gateway.sln, Under mvc-as-gateway, Open the Web.config, locate the following line and change the connectionString.

<add name="MyConString"

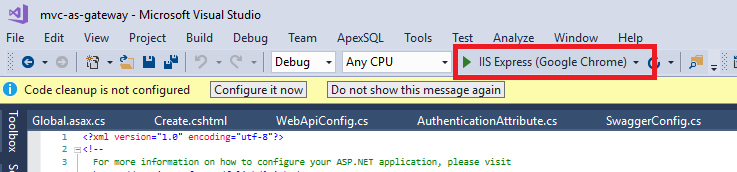
providerName="System.Data.SqlClient"

connectionString="Data Source=MYHL-LNVWEP012\SQLEXPRESS;Initial Catalog=mvcasgateway;User ID=sa;Password=sqlsa516;Integrated Security=False;"/>

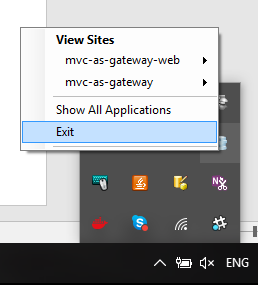
Right click the mvc-as-gateway-web and choose Set as StartUp Project.



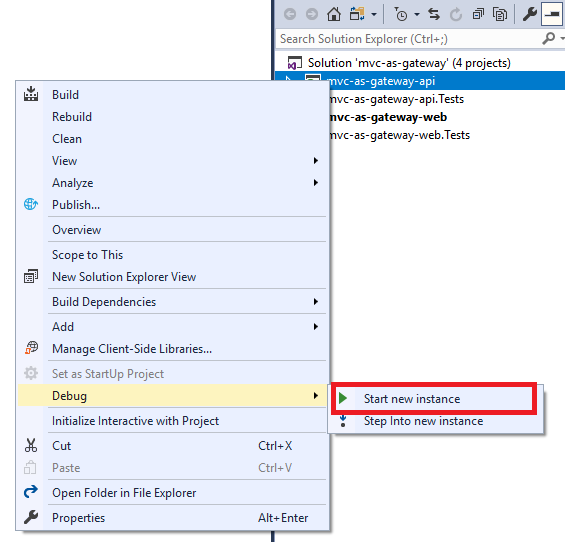
Locate the button below, click on it to start the web application



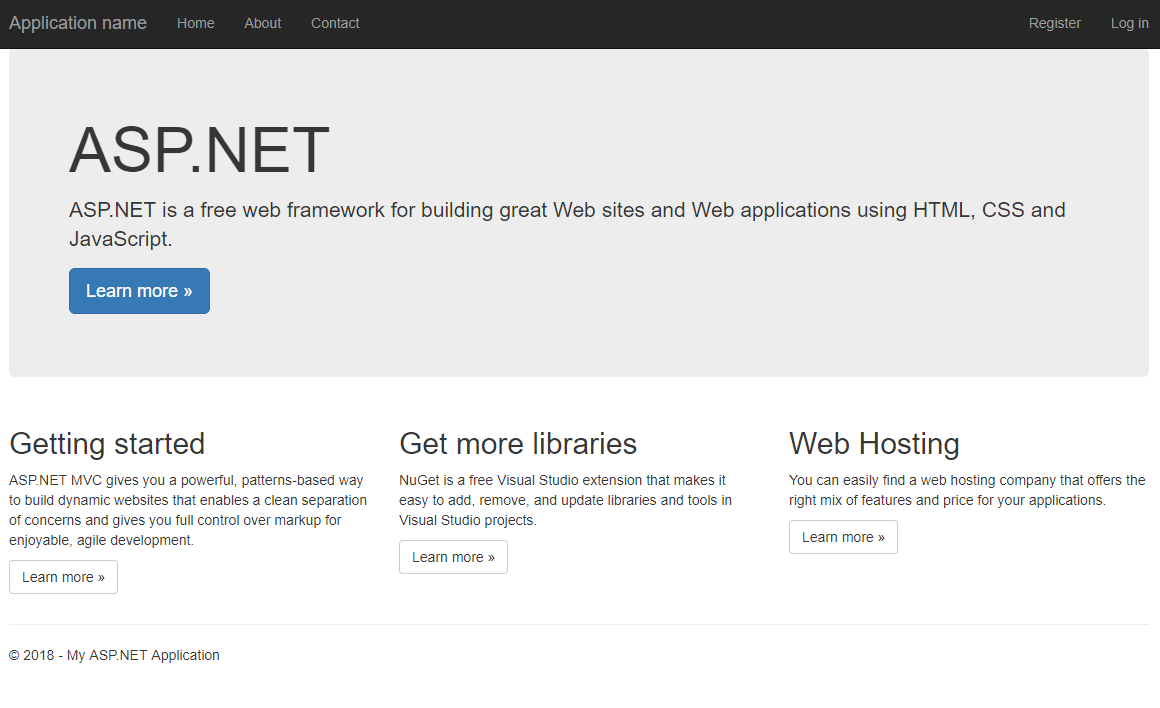
Check is IIS Express has started both application, if it does you may skip the next step



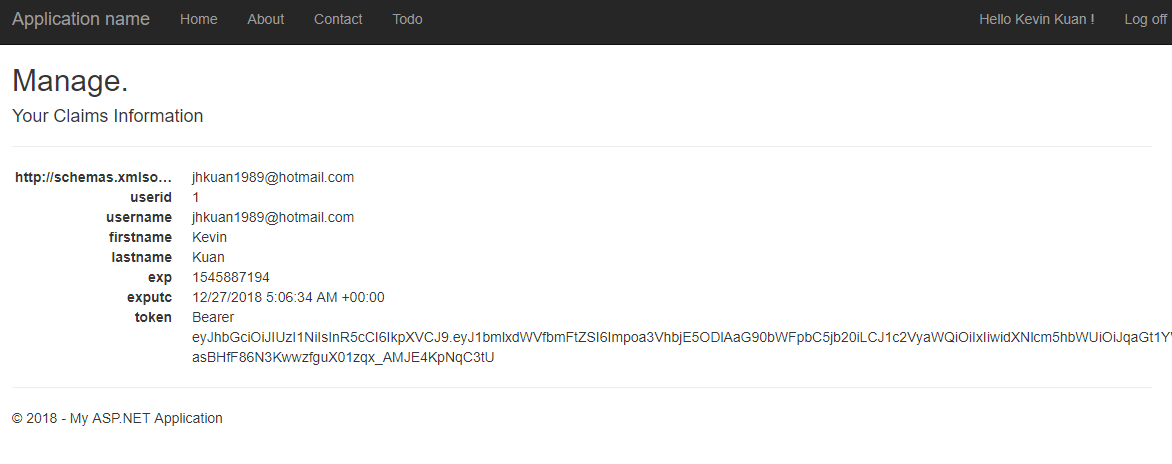
Right click mvc-as-gateway-api project -> Debug -> Start new instance to launch the Api Application



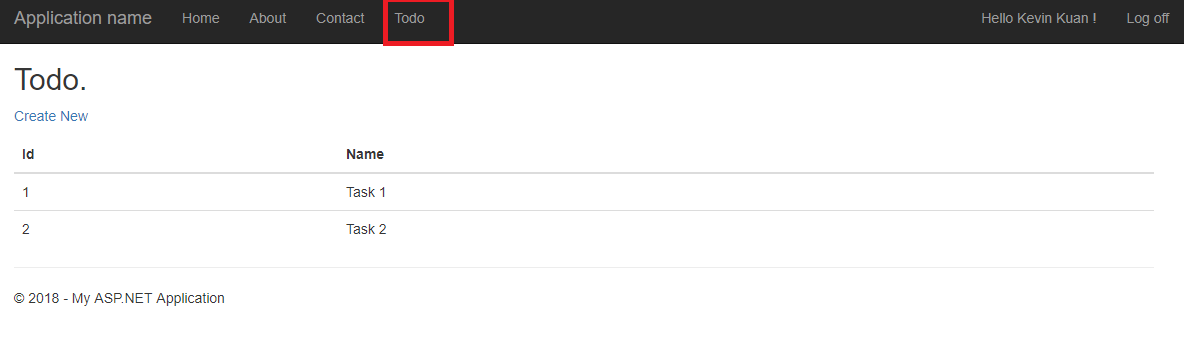
Landing page of web application is located at <https://localhost:44304/>, you may register a new user to try out the Todo List module that is only available to logged in users.



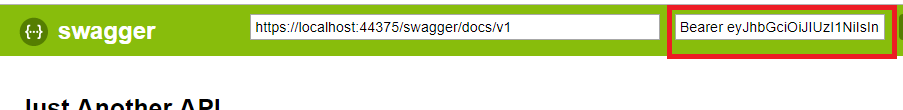
Upon Log in, you’ll be redirected to a page with your claim information. The token is the Jwt token returned from the Identity Provider for you to access the resource server.



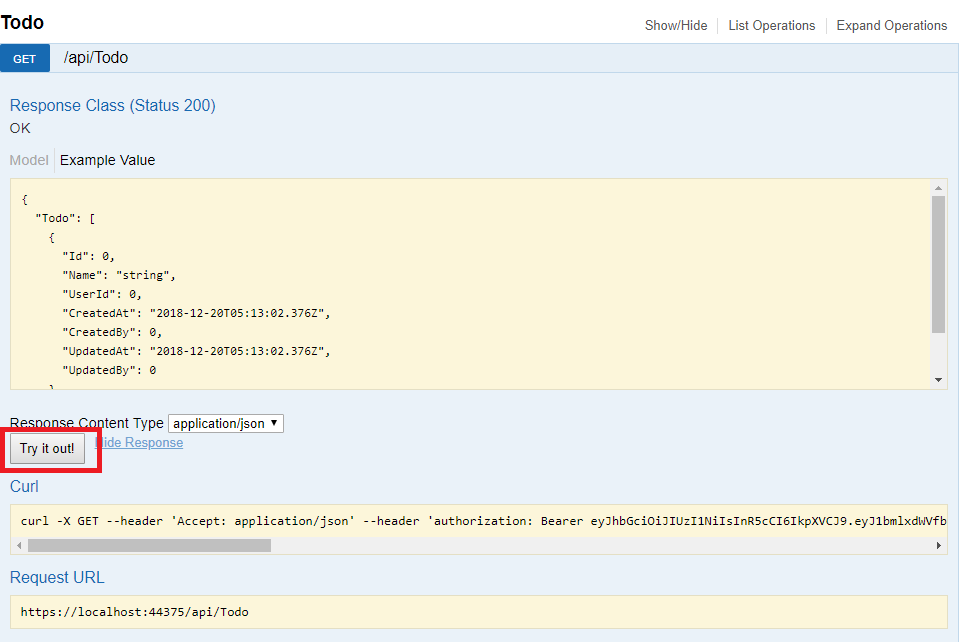
To test the usage of token in the API, you may go to the Todo list, I’ve only added the first 2 letters in the CRUD operations as this is not a POC for CRUD operation.



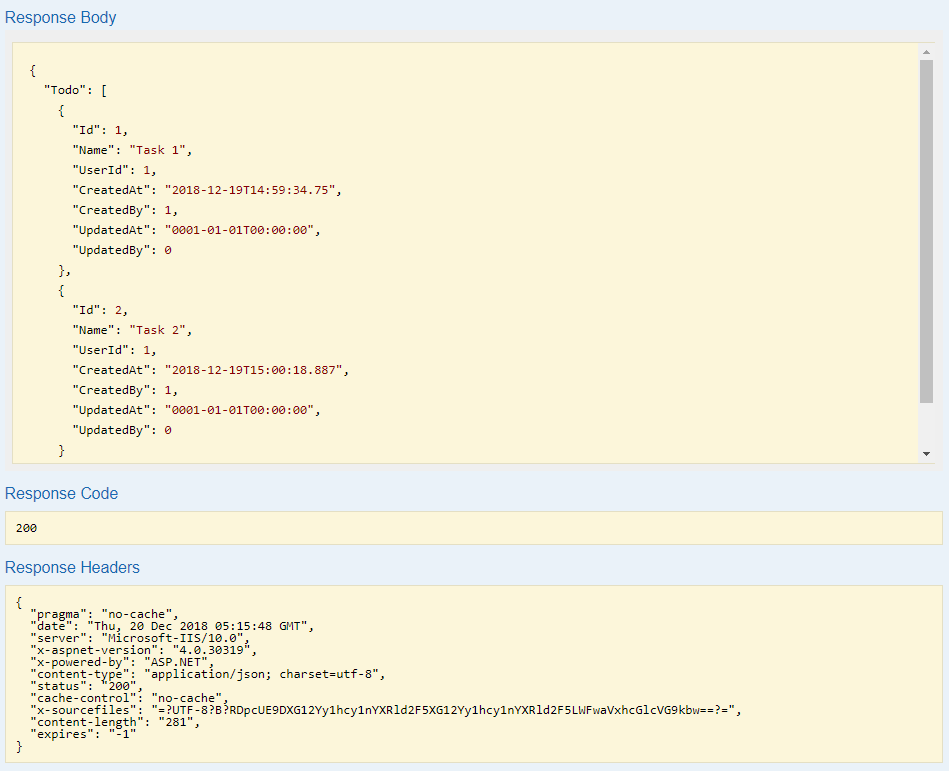
To test the usage, you may also go directly to the API url which located at <https://localhost:44375> and use the swagger UI to test. Copy the token from the Manage page and paste it the highlighted box.



After that, expand the Todo and click on Try it out!



The response will be returned as below.



Try refresh the page, and hit the Try it out button without putting in the token you’ll get unauthorized error code, meaning the resources are being protected.

