ASP.NET MVC

MVC 5

MVC Core

Eng. Basma Hussien

Hosting MVC on IIS

Deploying ASP.net MVC Application

Deploying ASP.net MVC Application:

- Publish to IIS web server
- Publish to Azure App Service
- Publish to a Filesystem Folder

Deployment prerequisites:

- SQL server installed
- IIS installed

Validate MVC App Using External Logins

Live Accounts Authentication

- You should Choose "***Individual User***" Authentication MVC template when creating your web application

- You have to create a project on "Google.developer.console" or what so ever provider you will use for authentication process delegation

- You need a "***Client ID & Client Secret***" to be able to use provider API for authenticating your web application

Membership & Identity Authentication

ASP.NET Membership

- It was designed to solve site membership requirements that were common in 2005, which involved Forms Authentication, and a SQL Server database for user names, passwords, and profile data.

- Today there is a much broader array of data storage options for web applications, and most developers want to enable their sites to use social identity providers for authentication and authorization functionality.

ASP.NET Membership Limitations

- The database schema was designed for SQL Server and you can't change it.
- Membership database is limited to describe users identity.
- Since the log-in/log-out functionality is based on Forms Authentication, the membership system can't use OWIN.

OWIN

- OWIN includes middleware components for authentication, including support for log-ins using external identity providers (like Microsoft Accounts, Facebook, Google, Twitter), and log-ins using organizational accounts from on-premises Active Directory or Azure Active Directory.

- OWIN also includes support for OAuth 2.0, JWT and CORS.

ASP.NET Identity

ASP.NET Identity was developed with the following goals:

- One ASP.NET Identity system
- Persistence control
- Claims Based
- Role provider
- Social Login Providers
- OWIN Integration
- Unit testability

One ASP.NET Identity system

- ASP.NET Identity can be used with all of the ASP.NET frameworks, such as ASP.NET MVC, Web Forms, Web Pages, Web API, and SignalR.

- ASP.NET Identity can be used when you are building web, phone, store, or hybrid applications.

Persistence control

- By default, the ASP.NET Identity system stores all the user information in a database. ASP.NET Identity uses ***Entity Framework Code First*** to implement all of its persistence mechanism.
- Since you control the database schema, common tasks such as changing table names or changing the data type of primary keys is simple to do.
- It's easy to plug in different storage mechanisms such as SharePoint, Azure Storage Table Service, NoSQL databases, etc., without having to throw System.NotImplementedExceptions exceptions.

Claims Based

- ASP.NET Identity supports claims-based authentication, where the user's identity is represented as a set of claims.
- Ease of plugging in profile data about the user, You have control over the schema of user and profile information.
- Claims allow developers to be a lot more expressive in describing a user's identity than roles allow.
- For more info about ClaimTypes: https://docs.microsoft.com/en-us/dotnet/api/system.security.claims.claimtypes?view=net-5.0

Social Login Providers

- You can easily add social log-ins such as Microsoft Account, Facebook, Twitter, Google, and others to your application, and store the user-specific data in your application.

OWIN Integration

- ASP.NET authentication is now based on OWIN middleware that can be used on any OWIN-based host.
- ASP.NET Identity does not have any dependency on System.Web. It is a fully compliant OWIN framework and can be used in any OWIN hosted application.
- ASP.NET Identity uses OWIN Authentication for log-in/log-out of users in the web site. This means that instead of using FormsAuthentication to generate the cookie, the application uses OWIN CookieAuthentication to do that.

Role provider

- There is a role provider which lets you restrict access to parts of your application by roles.
- You can easily create roles such as "Admin" and add users to roles.

Unit testability

- ASP.NET Identity makes the web application more unit testable.
- You can write unit tests for the parts of your application that use ASP.NET Identity.

Demo