ASP.NET 4.5

Lesson 10: ASP.NET Identity

Capgemini

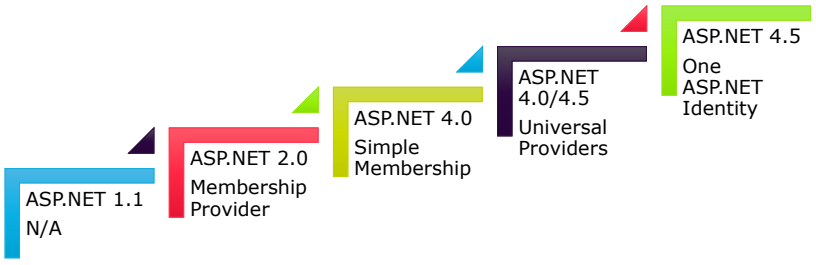
Lesson Objectives

- In this lesson, you will learn :
  - What is ASP.NET Identity?
  - Features of ASP.NET Identity
  - Claim Based Authentication
  - External Authentication

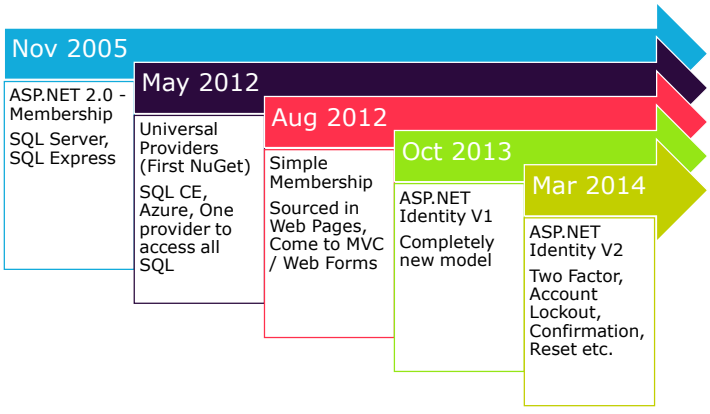
Authentication and Authorization

- Authentication is the process of verifying the identity of a user by accepting credentials.
- If the credentials are valid, the authorization process starts.
- Authentication process always proceeds to Authorization process
- Authorization is the process of allowing an authenticated user to access the resources by checking whether the user has access rights to the system.
- Authorization helps to control access rights by granting or denying specific permissions to an authenticated user.

From Membership to ASP.NET Identity



History of ASP.NET Account Services



ASP.NET Membership

- ASP.NET Membership 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.
- Since the log-in/log-out functionality is based on Forms Authentication, the membership system can't use 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.

ASP.NET Simple Membership

- It was developed as a membership system for ASP.NET Web Pages.
- It was released with WebMatrix and Visual Studio 2010 SP1.
- The goal of Simple Membership was to make it easy to add membership functionality to a Web Pages application.
- Simple Membership did make it easier to customize user profile information, but it still shares the other problems with ASP.NET Membership, and it has some limitations:
  - It was hard to persist membership system data in a non-relational store.
  - You can't use it with OWIN.
  - It doesn't work well with existing ASP.NET Membership providers, and it's not extensible.

ASP.NET Universal Providers

- It was developed to make it possible to persist membership information in Microsoft Azure SQL Database, and they also work with SQL Server Compact.
- The Universal Providers were built on Entity Framework Code First, which means that the Universal Providers can be used to persist data in any store supported by EF.
- The Universal Providers are built on the ASP.NET Membership infrastructure, so they still carry the same limitations as the SqlMembership Provider.
  - That is, they were designed for relational databases and it's hard to customize profile and user information.
  - These providers also still use Forms Authentication for log-in and log-out functionality.

What is ASP.NET Identity?

- The ASP.NET Identity system is designed to replace the previous ASP.NET Membership and Simple Membership systems.
- It includes profile support, OAuth integration, works with OWIN, and is included with the ASP.NET templates shipped with Visual Studio 2013.

ASP.NET Identity?

- 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.
- Ease of plugging in profile data about the user
  - You have control over the schema of user and profile information. For example, you can easily enable the system to store birth dates entered by users when they register an account in your application.

ASP.NET Identity? (Contd…)

- 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.
- 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.
- 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.

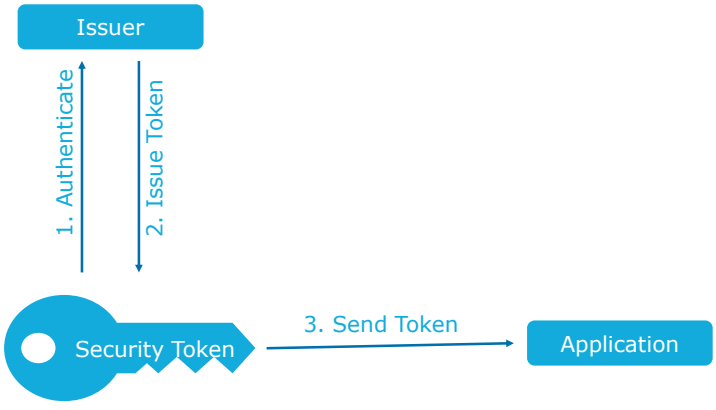
Demo

- Simple ASP.NET Identity

Claim Based Authentication

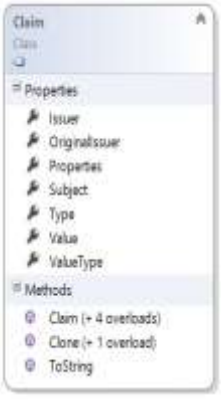
- ASP.NET Identity supports claims-based authentication, where the user's identity is represented as a set of claims.
- Claims allow developers to be a lot more expressive in describing a user's identity than roles allow.
- Whereas role membership is just a Boolean (member or non-member), a claim can include rich information about the user's identity and membership.

Claim Based Authentication



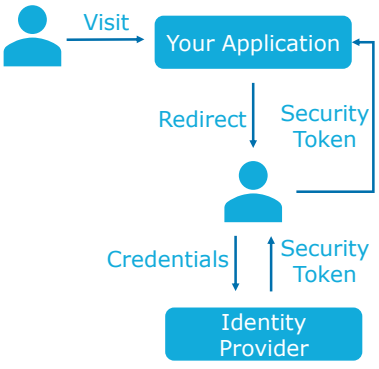
What are Claims?

- Identity makes extensive use of Claims
- User delivers claims to your app
  - Where do they come from?
  - Serialized in secured token
- Can contain much information about user
- Roles are single valued
  - Ex "Admins"
- Claims are key/value per user
  - Ex "Facebook Access Token", "CAAVl6UvghVkBAIGZB…"



Claim Based Authentication

- An application designed to take advantage of claim-based identity redirects the user to the identity provider of choice.
- The user interacts with the site of the provider and enters any information that the provider reckons to be useful to authenticate the request.
- If the operation is successful, the identity provider issues a security token and redirects back to the application.
- The security token that is then handed over to the application contains claims about the user.
- These claims are trusted by the application.

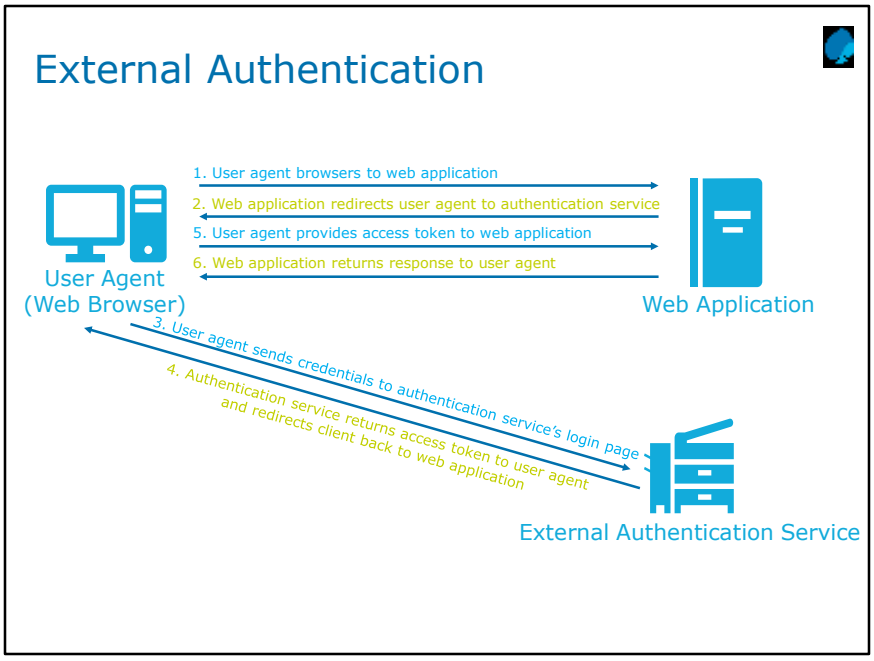


Demo

- Claim Based Authentication

ASP.NET Identity…

- 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.
- Visual Studio 2013 and ASP.NET 4.5.1 make integration with external authentication services easier for developers by providing built-in integration for the following authentication services:
  - Facebook
  - Google
  - Microsoft Accounts (Windows Live ID accounts)
  - Twitter



A simple request flow for a user agent (web browser) that is requesting information from a web application that is configured to use an external authentication service:

In the above diagram, the user agent (or web browser in this example) makes a request to a web application, which redirects the web browser to an external authentication service. The user agent sends its credentials to the external authentication service, and if the user agent has successfully authenticated, the external authentication service will redirect the user agent to the original web application with some form of token which the user agent will send to the web application. The web application will use the token to verify that the user agent has been successfully authenticated by the external authentication service, and the web application may use the token to gather more information about the user agent. Once the application is done processing the user agent's information, the web application will return the appropriate response to the user agent based on its authorization settings.

Demo

- External Authentication

NuGet Package

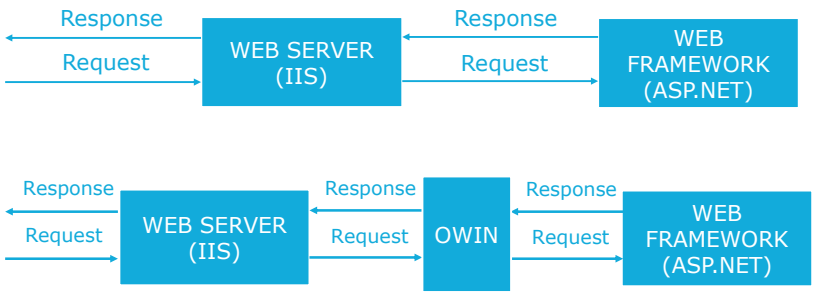
- ASP.NET Identity is redistributed as a NuGet package which is installed in the ASP.NET MVC, Web Forms and Web API templates that ship with Visual Studio 2013.
- You can download this NuGet package from the NuGet gallery.
- Releasing ASP.NET Identity as a NuGet package makes it easier for the ASP.NET team to iterate on new features and bug fixes, and deliver these to developers in an agile manner.

OWIN (Open Web Interface for .NET) 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.

OWIN

- Identity uses security middleware
- OWIN defines easy interface for items to communicate

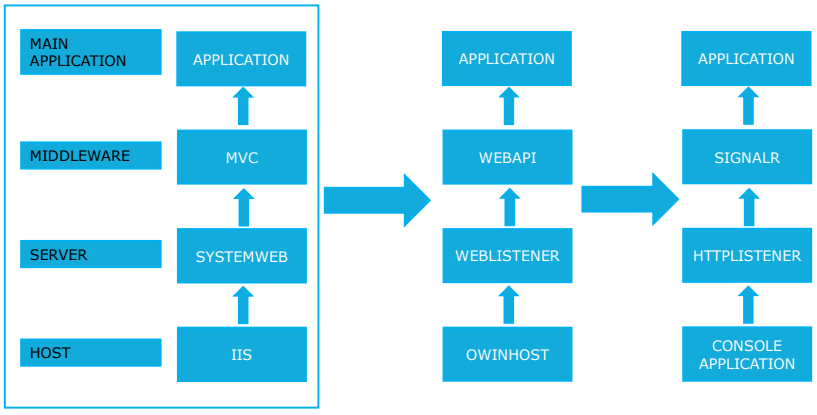


OWIN defines an abstraction between .NET web servers and web applications. By decoupling the web server from the application, OWIN makes it easier to create middleware for .NET web development. Also, OWIN makes it easier to port web applications to other hosts—for example, self-hosting in a Windows service or other process.
OWIN is a community-owned specification, not an implementation. The Katana project is a set of open-source OWIN components developed by Microsoft

New components could be more easily developed and consumed.
Applications could be more easily ported between hosts and potentially entire platforms/operating systems.
The resulting abstraction consists of two core elements. The first is the environment dictionary. This data structure is responsible for storing all of the state necessary for processing an HTTP request and response, as well as any relevant server state. The environment dictionary is defined as follows:

IDictionary<string, object>

Application on OWIN



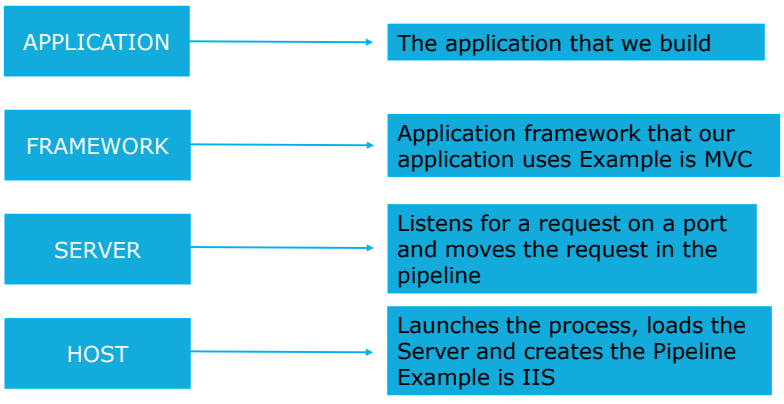
We can swap *IIS with ConsoleApplication , MVC with WebAPI , SystemWeb with WebListener without effecting our existing application. This is in contrast to the coupling between System.Web and IIS.*

KATANA

- Portable
  - Components should be able to be easily substituted for new components as they become available. This includes all types of components, from the framework to the server and host. The implication of this goal is that third party frameworks can seamlessly run on Microsoft servers while Microsoft frameworks can potentially run on third party servers and hosts.
- Modular/flexible
  - Katana project components should be small and focused, giving control over to the application developer in determining which components to use in her application.
- Lightweight/performant/scalable
  - By breaking the traditional notion of a framework into a set of small, focused components which are added explicitly by the application developer, a resulting Katana application can consume fewer computing resources, and as a result, handle more load, than with other types of servers and frameworks.

Microsoft's OWIN Implementation is Project Katana for v1, v2.Katana now fully integrated into ASP.NET 5 .No longer called Katana in Identity v3

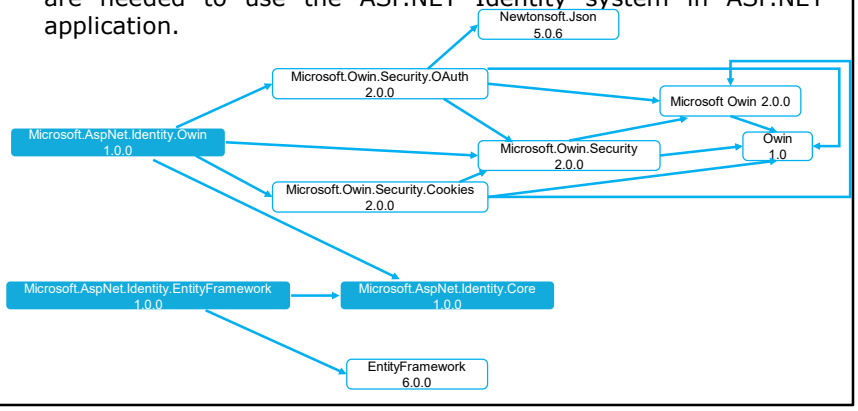
KATANA Architecture



Host: The process that runs the application and can be anything from IIS or a standalone executable, to your own custom program. The host is responsible for startup, loading of other OWIN components and shutting down gracefully.
Server: Responsible for binding to a TCP port, constructing the environment dictionary and processing requests through an OWIN pipeline.
Middleware: The name given to all of the components that handle requests in an OWIN pipeline. It can range from a simple compression component to a complete framework such as ASP.NET Web API, though from the server's perspective, it's simply a component that exposes the application delegate.
Application: This is your code. Because Katana is not a replacement for ASP.NET but rather a new way to compose and host components, existing ASP.NET Web API and SignalR applications remain unchanged, as those frameworks can participate in an OWIN pipeline. In fact, for these kinds of applications, Katana components will be visible only in a small configuration class.

ASP.NET Identity Architecture

- The diagram below shows the components of the ASP.NET Identity system. The packages in green make up the ASP.NET Identity system. All the other packages are dependencies which are needed to use the ASP.NET Identity system in ASP.NET application.



Packages for ASP.NET Project

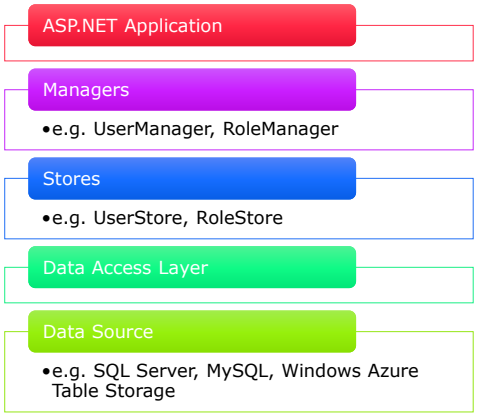
- Microsoft.AspNet.Identity.EntityFramework
  - This package has the Entity Framework implementation of ASP.NET Identity which will persist the ASP.NET Identity data and schema to SQL Server.
- Microsoft.AspNet.Identity.Core
  - This package has the core interfaces for ASP.NET Identity. This package can be used to write an implementation for ASP.NET Identity that targets different persistence stores such as Azure Table Storage, NoSQL databases etc.
- Microsoft.AspNet.Identity.OWIN
  - This package contains functionality that is used to plug in OWIN authentication with ASP.NET Identity in ASP.NET applications. This is used when you add log in functionality to your application and call into OWIN Cookie Authentication middleware to generate a cookie.

ASP.NET Identity

- Azure Active Directory
  - You can also add log-in functionality using Azure Active Directory, and store the user-specific data in your application.
- Microsoft ASP.NET tools for Azure Active Directory makes it simple to enable authentication for web applications hosted on Azure.
- You can use Azure Authentication to authenticate Office 365 users from your organization, corporate accounts synced from your on-premise Active Directory or users created in your own custom Azure Active Directory domain

Managers and Stores

- Managers
  - High-level classes
  - Operations such as create user
  - Talks to stores via Interface (i.e. pluggable)
- Stores
  - Talks to data access layer
  - Store users, roles, claims



Demo

- OWIN Demo

Problems and Solutions

Problem	Solution
Prepare an application for user authentication.	Apply the `Authorize` attribute to restrict access to action methods and define a controller to which users will be redirected to provide credentials.
Authenticate a user.	Check the name and password using the `FindAsync` method defined by the user manager class and create an implementation of the `IIdentity` interface using the `CreateIdentityMethod`. Set an authentication cookie for subsequent requests by calling the `SignIn` method defined by the authentication manager class.
Prepare an application for role-based authorization.	Create a role manager class and register it for instantiation in the OWIN startup class.
Create and delete roles.	Use the `CreateAsync` and `DeleteAsync` methods defined by the role manager class.
Manage role membership.	Use the `AddToRoleAsync` and `RemoveFromRoleAsync` methods defined by the user manager class.
Use roles for authorization.	Set the `Roles` property of the `Authorize` attribute.
Seed the database with initial content.	Use the database context initialization class.

Summary

- In this lesson, you have learnt :
  - The ASP.NET Identity system is designed to replace the previous ASP.NET Membership and Simple Membership systems.
  - ASP.NET Identity supports claims-based authentication, where the user's identity is represented as a set of claims.
  - OWIN defines an abstraction between .NET web servers and web applications.