ASP.Net Identity

Marwa Ahmad
Software Developer

Content

01 | Identity Overview

- What is Identity ?
- History overview
- Architecture of ASP.NET Identity
- ASP.NET Identity Customization

What is Identity?

- Identity is Users, Authentication, Authorization.
  It is a claims based system; stores login, roles, claims

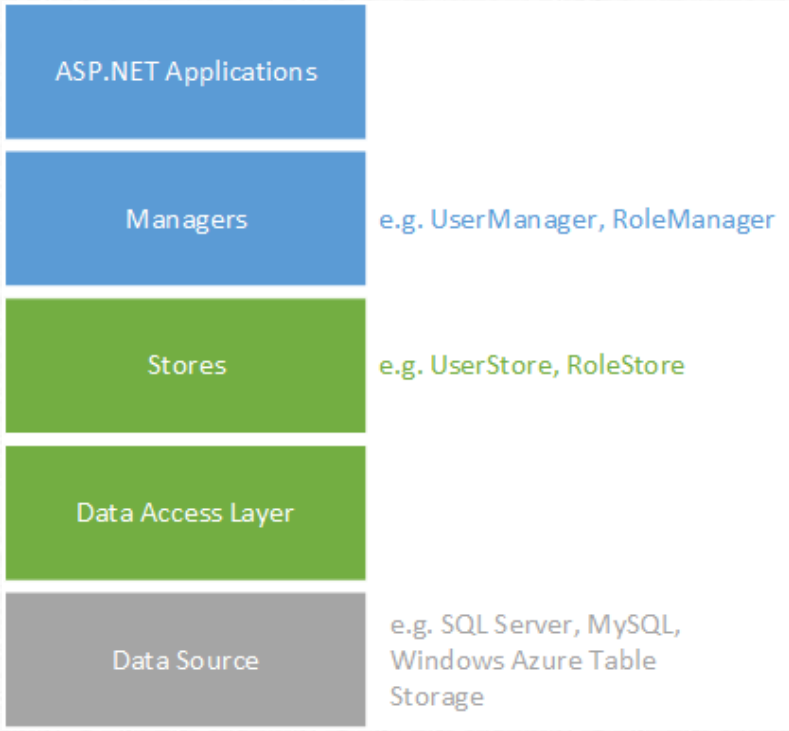
- Supports claims, roles, custom data stores, individual database backed auth, Oauth/OpenId, Organizational –AD,
  Azure AD, Single Sign On (SSO), Social Login providers

History Overview

Nov 2005 ASP.NET 2.0 – Introducing Membership! • **SQL Server, SQL Express**	**Oct 2013 ASP.NET Identity v1** • **Completely new model**
May 2012 Universal Providers (First NuGet) • SQL CE, Azure, one provider to access all SQL	Mar 2014 ASP.NET Identity v2 • VS 2013 Update 2. Two factor authN, account lockout, confirmation, reset, etc
Aug 2012 Simple Membership • Sourced in Web Pages, came to MVC / Web Forms	Oct 2014 (alpha) ASP.NET 5 – Identity v3 • VS 2013 Update 3. Changes to work with ASP.NET 5

ASP.NET Identity Architecture

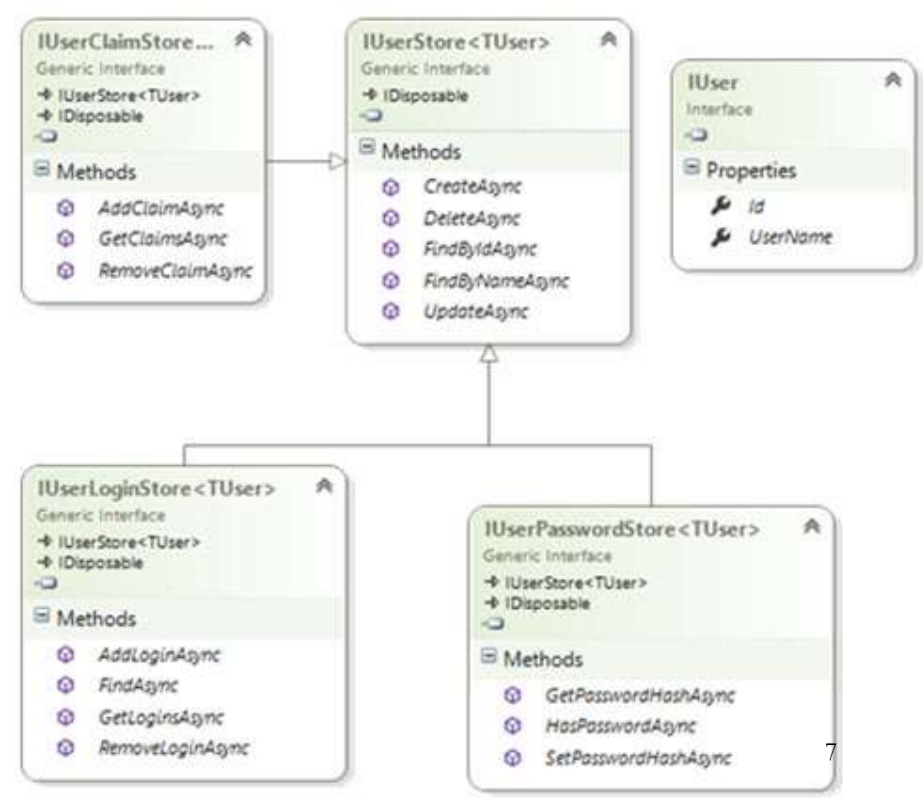
- Consists of Managers & Stores
- Managers
  - High-level classes; not concerned with how user info is stored, registering new users, validating credentials and loading user information
  - Ex: SigninManager, RoleManager, UserManager



ASP.NET Identity Architecture (cont.)

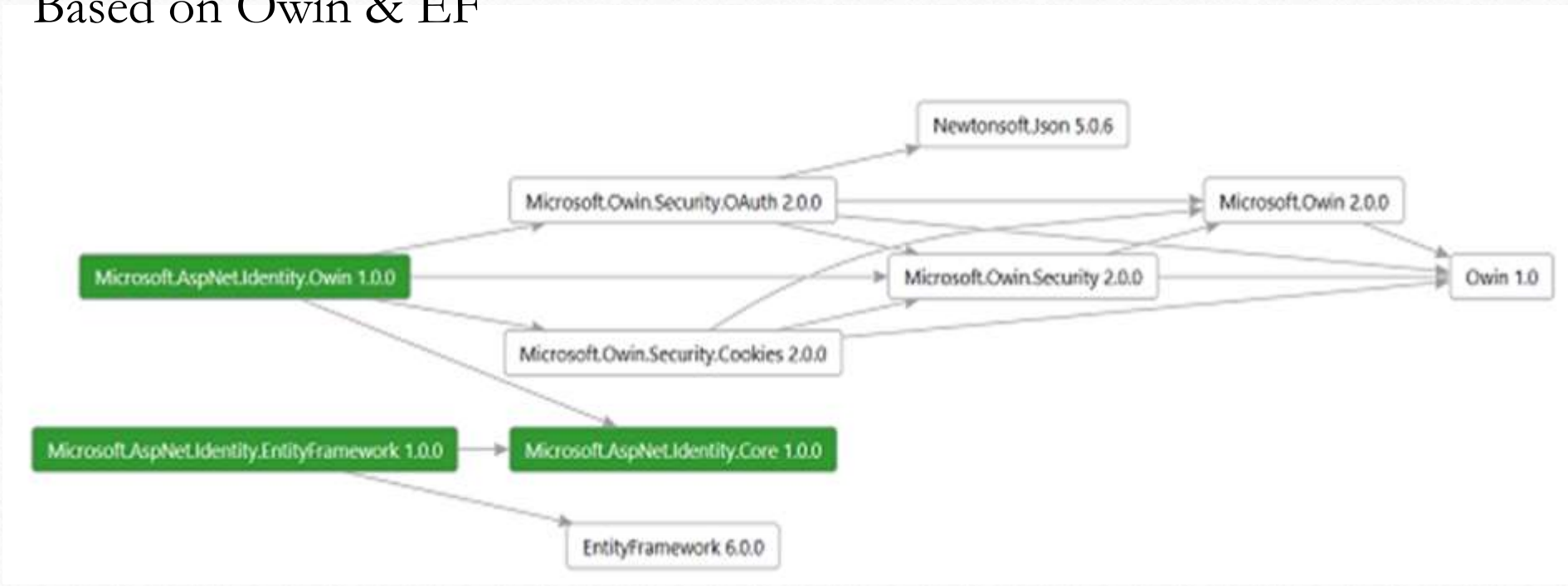
- Stores
  - Deals with DAL; CRUD functionality
  - Closely coupled with the persistence mechanism
  - By default EF Code First used to create tables

    SQL Server
  - Implementations available for Azure Table Storage, RavenDB and MongoDB



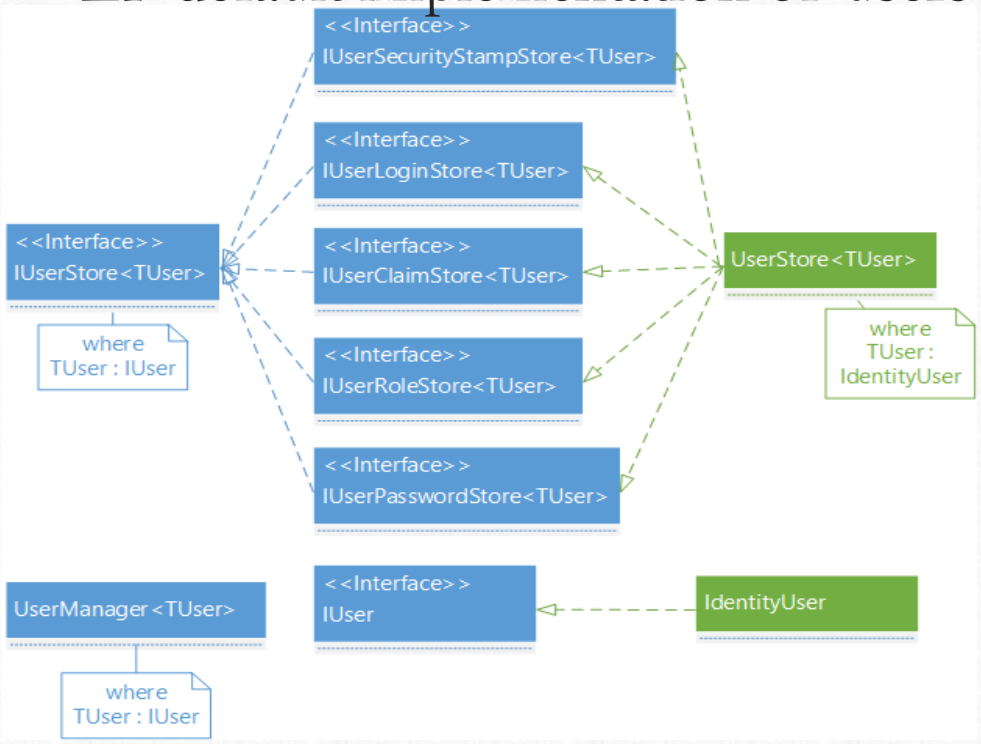
ASP.NET Identity Architecture (cont.)

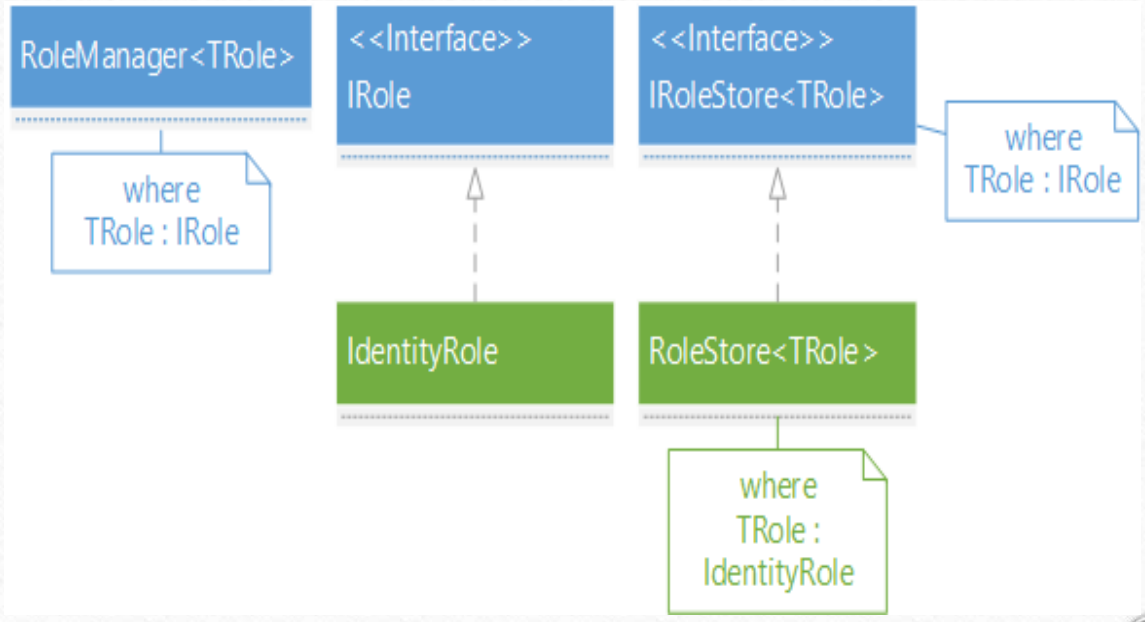
- Based on Owin & EF



ASP.NET Identity Architecture (cont.)

- EF default implementation of users & roles





ASP.NET Identity Customization

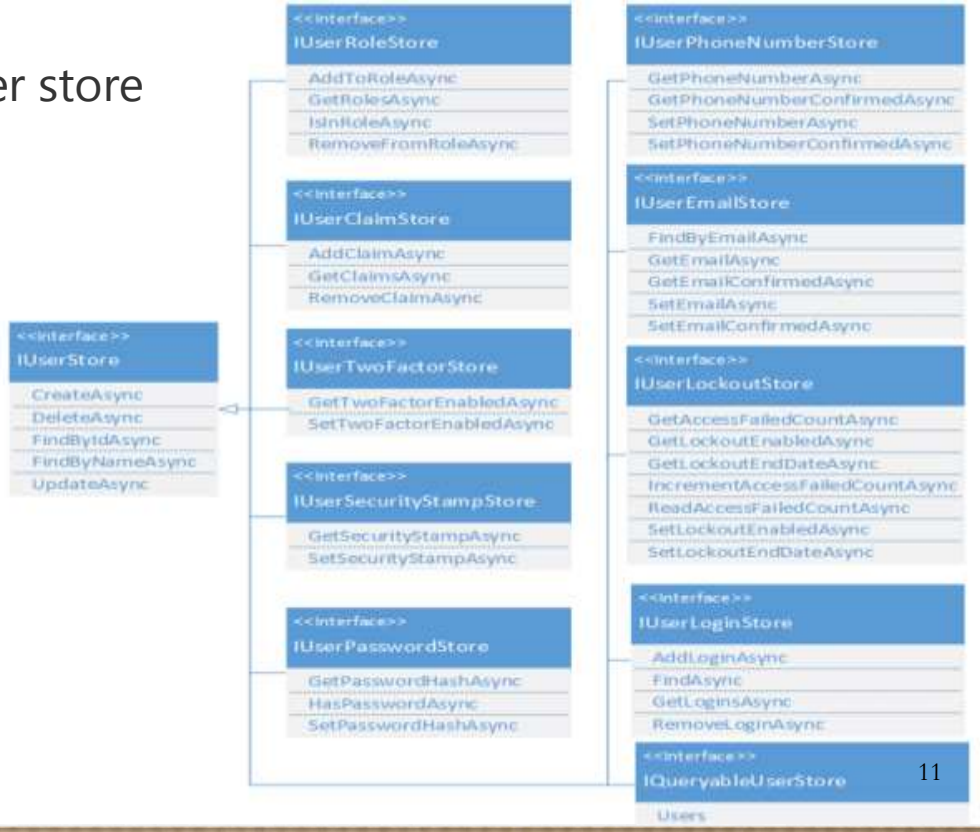
- Customize the user store the same applies to role store

```
public class IdentityUser : IUser<int>
{
    public IdentityUser() { ... }
    public IdentityUser(string userName) { ... }
    public int Id { get; set; }
    public string UserName { get; set; }
    // can also define optional properties such as:
    //    PasswordHash
    //    SecurityStamp
    //    Claims
    //    Logins
    //    Roles
}
```

```
public class UserStore : IUserStore<IdentityUser, int>
{
    public UserStore() { ... }
    public UserStore(ExampleStorage database) { ... }
    public Task CreateAsync(IdentityUser user) { ... }
    public Task DeleteAsync(IdentityUser user) { ... }
    public Task<IdentityUser> FindByIdAsync(int userId) { ... }
    public Task<IdentityUser> FindByNameAsync(string userName) { ... }
    public Task UpdateAsync(IdentityUser user) { ... }
    public void Dispose() { ... }
}
```

ASP.NET Identity Customization (cont.)

- Interfaces to implement when customizing user store



Content

- 01 | Identity Overview
- **02 | Locally Authenticated Users**
- 03 | oAuth and Social Providers
- 04 | Two Factor Authentication
- 05 | Asp.Net Identity with Webapi
- 06 | Identity Tips & Recommendations

02 | Locally Authenticated Users

- What are locally authenticated users?
  - Uses DB to authenticate; no third party i.e. authentication is on the same server (AspNetUsers table)
- Customizing the SQL database & entities

  ApplicationUser : IdentityUser
- Customizing the type of user store
  - Create your own UserStore and IdentityUser. RoleStore as well if you want that.
    Storage provider custom implementations exist(MySql, Azure Table Storage, RavenDB, etc

Content

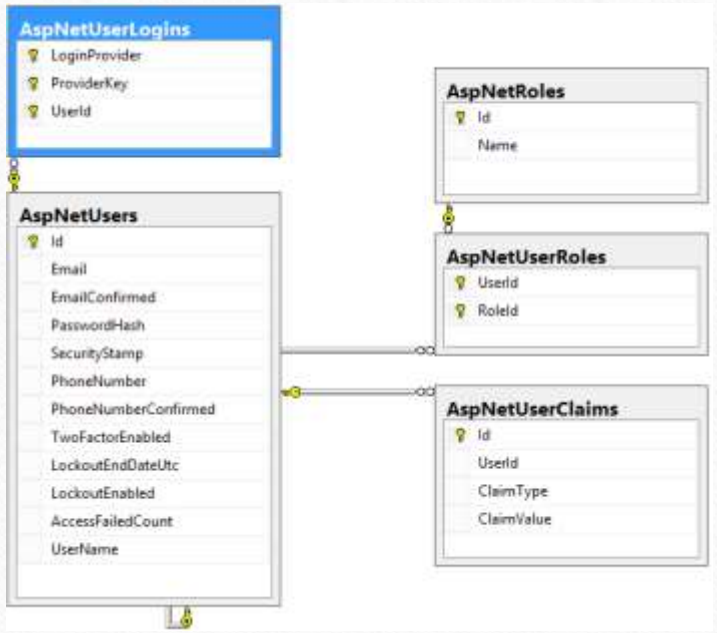
- 01 | Identity Overview
- 02 | Locally Authenticated Users
- **03 | oAuth and Social Providers**
- 04 | Two Factor Authentication
- 05 | Asp.Net Identity with Webapi
- 06 | Identity Tips & Recommendations

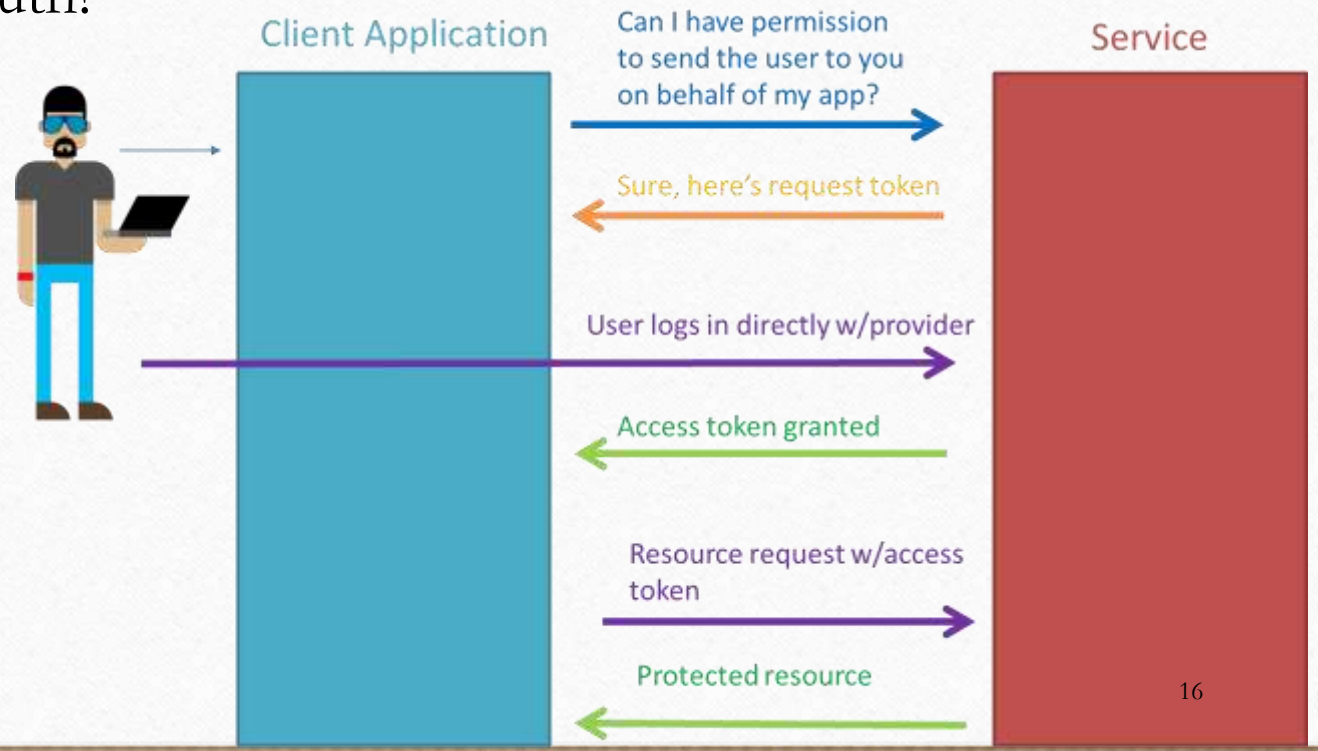
03 | oAuth and Social Providers

- What is oAuth?
  - oAuth is a protocol
  - The protocol allows for third party applications to access resources without users giving credentials to third party
  - Supports desktop, web, mobile, etc
- How does Identity use oAuth?
- Integrating with social/other providers

03 | oAuth and Social Providers (cont.)

- How does Identity use oAuth?

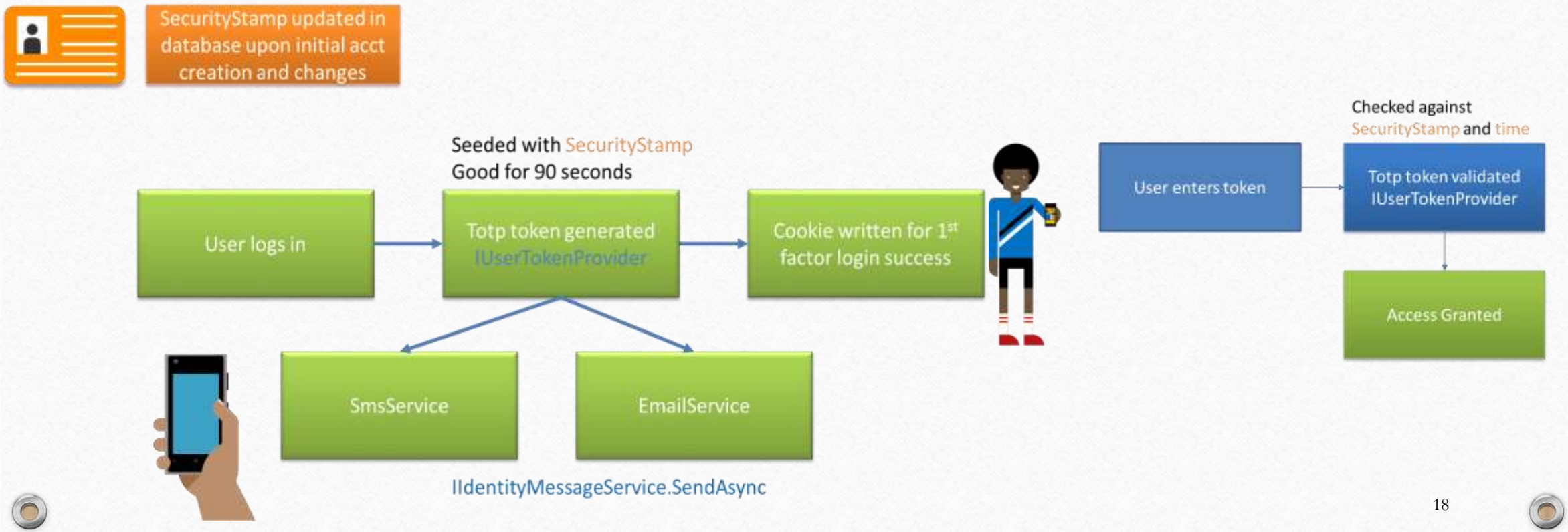




Content

- 01 | Identity Overview
- 02 | Locally Authenticated Users
- 03 | oAuth and Social Providers
- **04 | Two Factor Authentication**
- 05 | Asp.Net Identity with Webapi
- 06 | Identity Tips & Recommendations

04 | Two Factor Authentication



Content

- 01 | Identity Overview
- 02 | Locally Authenticated Users
- 03 | oAuth and Social Providers
- 04 | Two Factor Authentication
- **05 | Asp.Net Identity with Webapi**
- 06 | Identity Tips & Recommendations

Asp.Net Identity with Webapi

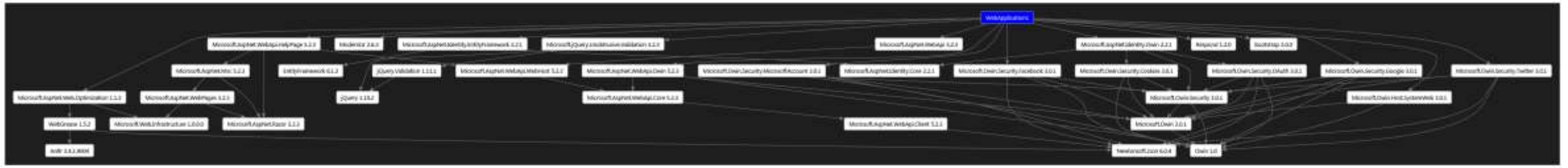
- Webapi2 Security AutheN Bearer Token tutorial; useful video; 2 mins only!
- Works with Framework 4.5, AspNet.Identity.Core 2.2.1, AspNet.Identity.EntityFramework 2.2.1, AspNet.Identity.WebApi 5.2.3
- Steps:
  - Create new Webapi project with Individual account authentication type
  - Run the project
  - Use Fiddler, call the Register endpoint; Ex: http://localhost:8070/api/Account/Register
    Request post body: then excute

    {"Email": "myemail@gmail.com",

    "Password": "Pa$$w0rd",

    "ConfirmPassword": "Pa$$w0rd"}
  - User fiddler: http://localhost:8070/token
    Request body:
    username=myemail@gmail.com&grant_type=password&Password=Pa$$w0rd
  - Now you are authorized to user any endpoint which requires [Authorize]

Asp.Net Identity with Webapi



Content

- 01 | Identity Overview
- 02 | Locally Authenticated Users
- 03 | oAuth and Social Providers
- 04 | Two Factor Authentication
- 05 | Asp.Net Identity with Webapi
- **06 | Identity Tips & Recommendations**

05 | Identity Tips & Recommendations

- Utilize SSL everywhere. **Never run without it**
  - Attacker on network can steal your cookies and hijack your session
  - Yes, even login page needs to be protected
  - Any page user can access while logged in should be protected
- Enforce a strong password policy!
  - Increase default values on manager.PasswordValidator
- Use Xsrf tokens everywhere for post methods
- Do not allow for unlimited login attempts
  - Brute forcers dream.
- Two factor authentication highly recommended
- Caution – be wary of email as a second factor authentication

Finally
What's Next?

- ASP.NET vNext (ASP.NET 5) being in development, Katana is slowly getting retired. Version 3.0 will most likely be last major release of Katana as a standalone framework
- vNext is the successor to Katana (which is why they look so similar). Katana was the beginning of the break away from System.Web and to more modular components for the web stack. You can see vNext as a continuation of that work plus (new CLR, new Project System, new http abstractions)* David Fowler vNext Architect
- Everything that exists today in Katana will make it's way into vNext
- ASP.NET vNext will be supported by .NET Framework 4.6

References

- Customizing asp.net authentication with Identity
- Securing web applications using asp.net identity
- Introduction to asp.net identity
- Creating web project; authentication modes
- Overview of custom storage provider of asp.net identity
- Asp.net identity releases
- Owin & Katana simplified
- Individual accounts in Webapi
- AspNet Identity 2.1 with AspNet WebApi 2.2; Accounts managemenet
- AspNet Identity 2.0 & WebApi- Customizing Identity Models & implementing Role-based Authorization


THANK
YOU !