dotnetexample

Release 0.1

August 20, 2015

Contents

CHAPTER 1

Sphinx AutoAPI Index

This page is the top-level of your generated API documentation. Below is a list of all items that are documented here.

1.1 Microsoft.AspNet.Builder Namespace

1.1.1 BuilderExtensions Class

Summary

Identity extensions for IApplicationBuilder.

Inheritance Hierarchy

- `System.Object`
- *`Microsoft.AspNet.Builder.BuilderExtensions`*

Syntax

```
public class BuilderExtensions
```

GitHub

View on GitHub

class `Microsoft.AspNet.Builder.`**`BuilderExtensions`**

Methods

Enables ASP.NET identity for the current application.

UseIdentity (*IApplicationBuilder*) **Arguments**

- **app** (*IApplicationBuilder*) – The IApplicationBuilder instance this method extends.

Return type IApplicationBuilder

Returns The <see cref="!:IApplicationBuilder" /> instance this method extends.

```
public static IApplicationBuilder UseIdentity(IApplicationBuilder app)
```

namespace Microsoft.AspNet.**Builder**

Classes

class ***Microsoft.AspNet.Builder.BuilderExtensions*** Identity extensions for IApplicationBuilder.

1.2 Microsoft.AspNet.Identity Namespace

1.2.1 ClaimsIdentityOptions Class

Summary

Options used to configure the claim types used for well known claims.

Inheritance Hierarchy

- System.Object
- *Microsoft.AspNet.Identity.ClaimsIdentityOptions*

Syntax

```
public class ClaimsIdentityOptions
```

GitHub

View on GitHub

class Microsoft.AspNet.Identity.**ClaimsIdentityOptions**

Properties

Gets or sets the ClaimType used for a Role claim.

RoleClaimType()

Return type System.String

```
public string RoleClaimType { get; set; }
```

SecurityStampClaimType()

Gets or sets the ClaimType used for the security stamp claim..

Return type System.String

```
public string SecurityStampClaimType { get; set; }
```

UserIdClaimType()

Gets or sets the ClaimType used for the user identifier claim.

Return type System.String

```
public string UserIdClaimType { get; set; }
```

UserNameClaimType()

Gets or sets the ClaimType used for the user name claim.

Return type System.String

```
public string UserNameClaimType { get; set; }
```

1.2.2 DataProtectionTokenProviderOptions Class

Summary

Contains options for the DataProtectorTokenProvider<TUser>.

Inheritance Hierarchy

- System.Object
- *Microsoft.AspNet.Identity.DataProtectionTokenProviderOptions*

Syntax

```
public class DataProtectionTokenProviderOptions
```

GitHub

View on GitHub

class Microsoft.AspNet.Identity.**DataProtectionTokenProviderOptions**

Properties

Gets or sets the name of the DataProtectorTokenProvider<TUser>.

Name()

Return type

System.String

```
public string Name { get; set; }
```

TokenLifespan()

Gets or sets the amount of time a generated token remains valid.

Return type System.TimeSpan

```
public TimeSpan TokenLifespan { get; set; }
```

1.2.3 DataProtectorTokenProvider Class

Summary

Provides protection and validation of identity tokens.

Inheritance Hierarchy

- System.Object
- *Microsoft.AspNet.Identity.DataProtectorTokenProvider*

Syntax

```
public class DataProtectorTokenProvider : IUserTokenProvider
```

GitHub

View on GitHub

class `Microsoft.AspNet.Identity.`**`DataProtectorTokenProvider`**

Constructors

Initializes a new instance of the DataProtectorTokenProvider<TUser> class.

`DataProtectorTokenProvider`(*IDataProtectionProvider*, *IOptions<Microsoft.AspNet.Identity.DataProtectionTokenProviderOptions>*) **Arguments**

- **`dataProtectionProvider`** (*IDataProtectionProvider*) – The system data protection provider.
- **`options`** (*IOptions{Microsoft.AspNet.Identity.DataProtectionTokenProviderOptions}*) – The configured Microsoft.AspNet.Identity.DataProtectionTokenProviderOptions.

```
public DataProtectorTokenProvider(IDataProtectionProvider dataProtectionProvider, IOptions<D
```

Methods

Returns a boolean indicating whether a token generated by this instance can be used as a Two Factor Authentication token as an asynchronous operation.

`CanGenerateTwoFactorTokenAsync<TUser>`(*Microsoft.AspNet.Identity.UserManager<TUser>*, *TUser*) **Arguments**

- **`manager`** (*Microsoft.AspNet.Identity.UserManager{TUser}*) – The Microsoft.AspNet.Identity.UserManager‘1 to retrieve user properties from.
- **`user`** (*{TUser}*) – The TUser the token was generated for.

Return type System.Threading.Tasks.Task{System.Boolean}

Returns A <see cref=”T:System.Threading.Tasks.Task‘1” /> that represents the result of the asynchronous query, containing true if a token generated by this instance can be used as a Two Factor Authentication token, otherwise false.

```
public virtual Task<bool> CanGenerateTwoFactorTokenAsync<TUser>(UserManager<TUser> manager,
```

`GenerateAsync<TUser>`(*System.String*, *Microsoft.AspNet.Identity.UserManager<TUser>*, *TUser*)

Generates a protected token for the specified `user` as an asynchronous operation.

Arguments

- **`purpose`** (*System.String*) – The purpose the token will be used for.
- **`manager`** (*Microsoft.AspNet.Identity.UserManager{TUser}*) – The Microsoft.AspNet.Identity.UserManager‘1 to retrieve user properties from.
- **`user`** (*{TUser}*) – The TUser the token will be generated from.

Return type System.Threading.Tasks.Task{System.String}

Returns A <see cref=”T:System.Threading.Tasks.Task‘1” /> representing the generated token.

```
public virtual Task<string> GenerateAsync<TUser>(string purpose, UserManager<TUser> manager,
```

NotifyAsync<TUser> (*System.String*, *Microsoft.AspNet.Identity.UserManager<TUser>*, *TUser*)
Creates a notification for the specified `user` based on the supplied `token` as an asynchronous operation.

Arguments

- **token** (*System.String*) – The token to generate notifications for.
- **manager** (*Microsoft.AspNet.Identity.UserManager{{TUser}}*) – The Microsoft.AspNet.Identity.UserManager`1 to retrieve user properties from.
- **user** (*{TUser}*) – The TUser the token was generated for.

Return type System.Threading.Tasks.Task

Returns A <see cref="T:System.Threading.Tasks.Task`1" /> that represents the asynchronous notification.

```
public virtual Task NotifyAsync<TUser>(string token, UserManager<TUser> manager, TUser user)
```

ValidateAsync<TUser> (*System.String*, *System.String*, *Microsoft.AspNet.Identity.UserManager<TUser>*, *TUser*)
Validates the protected `token` for the specified `user` and `purpose` as an asynchronous operation.

Arguments

- **purpose** (*System.String*) – The purpose the token was be used for.
- **token** (*System.String*) – The token to validate.
- **manager** (*Microsoft.AspNet.Identity.UserManager{{TUser}}*) – The Microsoft.AspNet.Identity.UserManager`1 to retrieve user properties from.
- **user** (*{TUser}*) – The TUser the token was generated for.

Return type System.Threading.Tasks.Task{System.Boolean}

Returns A <see cref="T:System.Threading.Tasks.Task`1" /> that represents the result of the asynchronous validation, containing true if the token is valid, otherwise false.

```
public virtual Task<bool> ValidateAsync<TUser>(string purpose, string token, UserManager<TUs
```

Properties

Gets the name of this instance.

Name()

Return type
System.String

```
public string Name { get; }
```

Options()
Gets the Microsoft.AspNet.Identity.DataProtectionTokenProviderOptions for this instance.

Return type Microsoft.AspNet.Identity.DataProtectionTokenProviderOptions

```
protected DataProtectionTokenProviderOptions Options { get; }
```

Protector()
Gets the IDataProtector for this instance.

Return type IDataProtector

```
protected IDataProtector Protector { get; }
```

1.2.4 EmailTokenProvider Class

Summary

TokenProvider that generates tokens from the user's security stamp and notifies a user via email.

Inheritance Hierarchy

- `System.Object`
- *`Microsoft.AspNet.Identity.TotpSecurityStampBasedTokenProvider`*
- *`Microsoft.AspNet.Identity.EmailTokenProvider`*

Syntax

```
public class EmailTokenProvider : TotpSecurityStampBasedTokenProvider, IUserTokenProvider
```

GitHub

View on GitHub

class `Microsoft.AspNet.Identity.`**`EmailTokenProvider`**

Constructors

Initializes a new instance of the EmailTokenProvider<TUser> class.

`EmailTokenProvider`(*IOptions<Microsoft.AspNet.Identity.EmailTokenProviderOptions>*, *System.String*)

Arguments

- **`options`** (*IOptions{Microsoft.AspNet.Identity.EmailTokenProviderOptions}*) – The configured Microsoft.AspNet.Identity.DataProtectionTokenProviderOptions.
- **`name`** (*System.String*) – The unique name for this instance of EmailTokenProvider<TUser>.

```
public EmailTokenProvider(IOptions<EmailTokenProviderOptions> options, string name = "")
```

Methods

Checks if a two factor authentication token can be generated for the specified `user`.

CanGenerateTwoFactorTokenAsync<TUser> (*Microsoft.AspNet.Identity.UserManager<TUser>*, *TUser*)

Arguments

- **manager** (*Microsoft.AspNet.Identity.UserManager{TUser}*) – The Microsoft.AspNet.Identity.UserManager`1 to retrieve the `user` from.
- **user** (*{TUser}*) – The TUser to check for the possibility of generating a two factor authentication token.

Return type System.Threading.Tasks.Task{System.Boolean}

Returns True if the user has an email address set, otherwise false.

```
public override Task<bool> CanGenerateTwoFactorTokenAsync<TUser>(UserManager<TUser> manager,
```

GetUserModifierAsync<TUser> (*System.String*, *Microsoft.AspNet.Identity.UserManager<TUser>*, *TUser*)

Returns the a value for the user used as entropy in the generated token.

Arguments

- **purpose** (*System.String*) – The purpose of the two factor authentication token.
- **manager** (*Microsoft.AspNet.Identity.UserManager{TUser}*) – The Microsoft.AspNet.Identity.UserManager`1 to retrieve the `user` from.
- **user** (*{TUser}*) – The TUser to check for the possibility of generating a two factor authentication token.

Return type System.Threading.Tasks.Task{System.String}

Returns A string suitable for use as entropy in token generation.

```
public override Task<string> GetUserModifierAsync<TUser>(string purpose, UserManager<TUser>
```

Properties

Gets the unique name for this instance of EmailTokenProvider<TUser>.

Name ()

Return type

System.String

```
public override string Name { get; }
```

Options ()

Gets the options for this instance of EmailTokenProvider<TUser>.

Return type Microsoft.AspNet.Identity.EmailTokenProviderOptions

```
public EmailTokenProviderOptions Options { get; }
```

1.2.5 EmailTokenProviderOptions Class

Summary

Options for the EmailTokenProvider<TUser> class.

Inheritance Hierarchy

- `System.Object`
- *`Microsoft.AspNet.Identity.EmailTokenProviderOptions`*

Syntax

```
public class EmailTokenProviderOptions
```

GitHub

View on GitHub

class `Microsoft.AspNet.Identity.`**`EmailTokenProviderOptions`**

Properties

Gets or sets the unique name used for an instance of EmailTokenProvider<TUser>.

`Name``()` **Return type**

System.String

```
public string Name { get; set; }
```

1.2.6 ExternalLoginInfo Class

Summary

Represents login information, source and externally source principal for a user record

Inheritance Hierarchy

- System.Object
- *Microsoft.AspNet.Identity.UserLoginInfo*
- *Microsoft.AspNet.Identity.ExternalLoginInfo*

Syntax

```
public class ExternalLoginInfo : UserLoginInfo
```

GitHub

View on GitHub

class Microsoft.AspNet.Identity.**ExternalLoginInfo**

Constructors

Creates a new instance of Microsoft.AspNet.Identity.ExternalLoginInfo

ExternalLoginInfo(*System.Security.Claims.ClaimsPrincipal*, *System.String*, *System.String*, *System.String*)

Arguments

- **externalPrincipal** (*System.Security.Claims.ClaimsPrincipal*) – The System.Security.Claims.ClaimsPrincipal to associate with this login.
- **loginProvider** (*System.String*) – The provider associated with this login information.
- **providerKey** (*System.String*) – The unique identifier for this user provided by the login provider.
- **displayName** (*System.String*) – The display name for this user provided by the login provider.

```
public ExternalLoginInfo(ClaimsPrincipal externalPrincipal, string loginProvider, string pro
```

Properties

Gets or sets the System.Security.Claims.ClaimsPrincipal associated with this login.

ExternalPrincipal()

Return type

System.Security.Claims.ClaimsPrincipal

```
public ClaimsPrincipal ExternalPrincipal { get; set; }
```

1.2.7 ILookupNormalizer Interface

Summary

Provides an abstraction for normalizing keys for lookup purposes.

Syntax

```
public interface ILookupNormalizer
```

GitHub

View on GitHub

interface `Microsoft.AspNet.Identity.`**`ILookupNormalizer`**

Methods

Returns a normalized representation of the specified `key`.

`Normalize`(*System.String*) **Arguments**

- **`key`** (*System.String*) – The key to normalize.

Return type System.String

Returns A normalized representation of the specified <paramref name="key" />.

```
string Normalize(string key)
```

1.2.8 IPasswordHasher Interface

Summary

Provides an abstraction for hashing passwords.

Syntax

```
public interface IPasswordHasher
```

GitHub

View on GitHub

interface `Microsoft.AspNet.Identity.`**`IPasswordHasher`**

Methods

Returns a hashed representation of the supplied `password` for the specified `user`.

`HashPassword<TUser>` (*TUser*, *System.String*)

Arguments

- **`user`** (*{TUser}*) – The user whose password is to be hashed.
- **`password`** (*System.String*) – The password to hash.

Return type System.String

Returns A hashed representation of the supplied <paramref name="password" /> for the specified <paramref name="user" />.

```
string HashPassword<TUser>(TUser user, string password)where TUser : class
```

`VerifyHashedPassword<TUser>` (*TUser*, *System.String*, *System.String*)

Returns a Microsoft.AspNet.Identity.PasswordVerificationResult indicating the result of a password hash comparison.

Arguments

- **`user`** (*{TUser}*) – The user whose password should be verified.
- **`hashedPassword`** (*System.String*) – The hash value for a user's stored password.
- **`providedPassword`** (*System.String*) – The password supplied for comparison.

Return type Microsoft.AspNet.Identity.PasswordVerificationResult

Returns A <see cref="T:Microsoft.AspNet.Identity.PasswordVerificationResult" /> indicating the result of a password hash comparison.

```
PasswordVerificationResult VerifyHashedPassword<TUser>(TUser user, string hashedPassword, st
```

1.2.9 IPasswordValidator Interface

Summary

Provides an abstraction for validating passwords.

Syntax

```
public interface IPasswordValidator
```

GitHub

View on GitHub

interface `Microsoft.AspNet.Identity.`**`IPasswordValidator`**

Methods

Validates a password as an asynchronous operation.

`ValidateAsync<TUser>` (*Microsoft.AspNet.Identity.UserManager<TUser>*, *TUser*, *System.String*)

Arguments

- **`manager`** (*Microsoft.AspNet.Identity.UserManager{{TUser}}*) – The Microsoft.AspNet.Identity.UserManager`1 to retrieve the `user` properties from.
- **`user`** (*{TUser}*) – The user whose password should be validated.
- **`password`** (*System.String*) – The password supplied for validation

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

Returns The task object representing the asynchronous operation.

```
        Task<IdentityResult> ValidateAsync<TUser>(UserManager<TUser> manager, TUser user, string pas
```

1.2.10 IQueryableRoleStore<TRole> Interface

Summary

Provides an abstraction for querying roles in a Role store.

Syntax

```
public interface IQueryableRoleStore<TRole> : IRoleStore<TRole>, IDisposable where TRole : class
```

GitHub

View on GitHub

interface `Microsoft.AspNet.Identity.`**`IQueryableRoleStore<TRole>`**

Properties

Returns an System.Linq.IQueryable‘1 collection of roles.

`Roles``()` **Return type**
System.Linq.IQueryable{{TRole}}

```
IQueryable<TRole> Roles { get; }
```

1.2.11 IQueryableUserStore<TUser> Interface

Summary

Provides an abstraction for querying roles in a User store.

Syntax

```
public interface IQueryableUserStore<TUser> : IUserStore<TUser>, IDisposable where TUser : class
```

GitHub

View on GitHub

interface `Microsoft.AspNet.Identity.`**`IQueryableUserStore<TUser>`**

Properties

Returns an System.Linq.IQueryable‘1 collection of users.

`Users``()` **Return type**
System.Linq.IQueryable{{TUser}}

```
IQueryable<TUser> Users { get; }
```

1.2.12 IRoleClaimStore<TRole> Interface

Summary

Provides an abstraction for a store of role specific claims.

Syntax

```
public interface IRoleClaimStore<TRole> : IRoleStore<TRole>, IDisposable where TRole : class
```

GitHub

View on GitHub

interface `Microsoft.AspNet.Identity.`**`IRoleClaimStore<TRole>`**

Methods

Add a new claim to a role as an asynchronous operation.

`AddClaimAsync`(*TRole*, *System.Security.Claims.Claim*, *System.Threading.CancellationToken*)

Arguments

- **`role`** (*{TRole}*) – The role to add a claim to.
- **`claim`** (*System.Security.Claims.Claim*) – The System.Security.Claims.Claim to add.
- **`cancellationToken`** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task

Returns The task object representing the asynchronous operation.

```
Task AddClaimAsync(TRole role, Claim claim, CancellationToken cancellationToken = null)
```

`GetClaimsAsync`(*TRole*, *System.Threading.CancellationToken*)

Gets a list of System.Security.Claims.Claim‘s to be belonging to the specified ‘‘role‘ as an asynchronous operation.

Arguments

- **`role`** (*{TRole}*) – The role whose claims to retrieve.
- **`cancellationToken`** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task{System.Collections.Generic.IList{System.Security.Claims.Claim}}

Returns A <see cref="T:System.Threading.Tasks.Task`1" /> that represents the result of the asynchronous query, a list of <see cref="T:System.Security.Claims.Claim" />s.

```
Task<IList<Claim>> GetClaimsAsync(TRole role, CancellationToken cancellationToken = null)
```

RemoveClaimAsync(*TRole*, *System.Security.Claims.Claim*, *System.Threading.CancellationToken*)
Remove a claim from a role as an asynchronous operation.

Arguments

- **role** (*{TRole}*) – The role to remove the claim from.
- **claim** (*System.Security.Claims.Claim*) – The System.Security.Claims.Claim to remove.
- **cancellationToken** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task

Returns The task object representing the asynchronous operation.

```
Task RemoveClaimAsync(TRole role, Claim claim, CancellationToken cancellationToken = null)
```

1.2.13 IRoleStore<TRole> Interface

Summary

Provides an abstraction for a storage and management of roles.

Syntax

```
public interface IRoleStore<TRole> : IDisposable where TRole : class
```

GitHub

View on GitHub

interface `Microsoft.AspNet.Identity.`**IRoleStore<TRole>**

Methods

Creates a new role in a store as an asynchronous operation.

CreateAsync(*TRole*, *System.Threading.CancellationToken*) **Arguments**

- **role** (*{TRole}*) – The role to create in the store.

- **cancellationToken** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

Returns A <see cref="T:System.Threading.Tasks.Task`1" /> that represents the <see cref="T:Microsoft.AspNet.Identity.IdentityResult" /> of the asynchronous query.

```
Task<IdentityResult> CreateAsync(TRole role, CancellationToken cancellationToken)
```

DeleteAsync(*TRole*, *System.Threading.CancellationToken*)

Deletes a role from the store as an asynchronous operation.

Arguments

- **role** (*{TRole}*) – The role to delete from the store.
- **cancellationToken** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

Returns A <see cref="T:System.Threading.Tasks.Task`1" /> that represents the <see cref="T:Microsoft.AspNet.Identity.IdentityResult" /> of the asynchronous query.

```
Task<IdentityResult> DeleteAsync(TRole role, CancellationToken cancellationToken)
```

FindByIdAsync(*System.String*, *System.Threading.CancellationToken*)

Finds the role who has the specified ID as an asynchronous operation.

Arguments

- **roleId** (*System.String*) – The role ID to look for.
- **cancellationToken** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task{{TRole}}

Returns A <see cref="T:System.Threading.Tasks.Task`1" /> that result of the look up.

```
Task<TRole> FindByIdAsync(string roleId, CancellationToken cancellationToken)
```

FindByNameAsync(*System.String*, *System.Threading.CancellationToken*)

Finds the role who has the specified normalized name as an asynchronous operation.

Arguments

- **normalizedRoleName** (*System.String*) – The normalized role name to look for.
- **cancellationToken** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task{{TRole}}

Returns A <see cref="T:System.Threading.Tasks.Task`1" /> that result of the look up.

```
Task<TRole> FindByNameAsync(string normalizedRoleName, CancellationToken cancellationToken)
```

GetNormalizedRoleNameAsync(*TRole*, *System.Threading.CancellationToken*)

Get a role's normalized name as an asynchronous operation.

Arguments

- **role** (*{TRole}*) – The role whose normalized name should be retrieved.
- **cancellationToken** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task{System.String}

Returns A <see cref="T:System.Threading.Tasks.Task`1" /> that contains the name of the role.

```
Task<string> GetNormalizedRoleNameAsync(TRole role, CancellationToken cancellationToken)
```

GetRoleIdAsync(*TRole*, *System.Threading.CancellationToken*)

Gets the ID for a role from the store as an asynchronous operation.

Arguments

- **role** (*{TRole}*) – The role whose ID should be returned.
- **cancellationToken** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task{System.String}

Returns A <see cref="T:System.Threading.Tasks.Task`1" /> that contains the ID of the role.

```
Task<string> GetRoleIdAsync(TRole role, CancellationToken cancellationToken)
```

GetRoleNameAsync(*TRole*, *System.Threading.CancellationToken*)

Gets the name of a role from the store as an asynchronous operation.

Arguments

- **role** (*{TRole}*) – The role whose name should be returned.
- **cancellationToken** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task{System.String}

Returns A <see cref="T:System.Threading.Tasks.Task`1" /> that contains the name of the role.

```
Task<string> GetRoleNameAsync(TRole role, CancellationToken cancellationToken)
```

SetNormalizedRoleNameAsync(*TRole*, *System.String*, *System.Threading.CancellationToken*)

Set a role's normalized name as an asynchronous operation.

Arguments

- **role** (*{TRole}*) – The role whose normalized name should be set.
- **normalizedName** (*System.String*) – The normalized name to set
- **cancellationToken** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation.

```
Task SetNormalizedRoleNameAsync(TRole role, string normalizedName, CancellationToken cancell
```

SetRoleNameAsync(*TRole*, *System.String*, *System.Threading.CancellationToken*)
Sets the name of a role in the store as an asynchronous operation.

Arguments

- **role** (*{TRole}*) – The role whose name should be set.
- **roleName** (*System.String*) – The name of the role.
- **cancellationToken** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation.

```
Task SetRoleNameAsync(TRole role, string roleName, CancellationToken cancellationToken)
```

UpdateAsync(*TRole*, *System.Threading.CancellationToken*)
Updates a role in a store as an asynchronous operation.

Arguments

- **role** (*{TRole}*) – The role to update in the store.
- **cancellationToken** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

Returns A <see cref="T:System.Threading.Tasks.Task`1" /> that represents the <see cref="T:Microsoft.AspNet.Identity.IdentityResult" /> of the asynchronous query.

```
Task<IdentityResult> UpdateAsync(TRole role, CancellationToken cancellationToken)
```

1.2.14 IRoleValidator Interface

Summary

Provides an abstraction for a validating a role.

Syntax

```
public interface IRoleValidator
```

GitHub

View on GitHub

interface `Microsoft.AspNet.Identity.`**`IRoleValidator`**

Methods

Validates a role as an asynchronous operation.

`ValidateAsync<TRole>` (*Microsoft.AspNet.Identity.RoleManager<TRole>*, *TRole*) **Arguments**

- **`manager`** (*Microsoft.AspNet.Identity.RoleManager{{TRole}}*) – The Microsoft.AspNet.Identity.RoleManager‘1 managing the role store.
- **`role`** (*{TRole}*) – The role to validate.

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

Returns A <see cref=”T:System.Threading.Tasks.Task‘1” /> that represents the <see cref=”T:Microsoft.AspNet.Identity.IdentityResult” /> of the asynchronous validation.

```
Task<IdentityResult> ValidateAsync<TRole>(RoleManager<TRole> manager, TRole role)where TRole
```

1.2.15 ISecurityStampValidator Interface

Summary

Provides an abstraction for a validating a security stamp of an incoming identity, and regenerating or rejecting the identity based on the validation result.

Syntax

```
public interface ISecurityStampValidator
```

GitHub

View on GitHub

interface `Microsoft.AspNet.Identity.`**`ISecurityStampValidator`**

Methods

Validates a security stamp of an identity as an asynchronous operation, and rebuilds the identity if the validation succeeds, otherwise rejects the identity.

ValidateAsync(*CookieValidatePrincipalContext*) **Arguments**

- **context** (*CookieValidatePrincipalContext*) – The context containing the ClaimsPrincipaland AuthenticationProperties to validate.

Return type System.Threading.Tasks.Task

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous validation operation.

```
Task ValidateAsync(CookieValidatePrincipalContext context)
```

1.2.16 IUserClaimStore<TUser> Interface

Summary

Provides an abstraction for a store of claims for a user.

Syntax

```
public interface IUserClaimStore<TUser> : IUserStore<TUser>, IDisposable where TUser : class
```

GitHub

View on GitHub

interface Microsoft.AspNet.Identity.**IUserClaimStore<TUser>**

Methods

Add claims to a user as an asynchronous operation.

AddClaimsAsync(*TUser*, *System.Collections.Generic.IEnumerable<System.Security.Claims.Claim>*, *System.Threading.CancellationToken*) **Arguments**

- **user** (*{TUser}*) – The user to add the claim to.
- **claims** (*System.Collections.Generic.IEnumerable{System.Security.Claims.Claim}*) – The collection of :dn:ref:‘System.Security.Claims.Claim‘s to add.

- **cancellationToken** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task

Returns The task object representing the asynchronous operation.

```
Task AddClaimsAsync(TUser user, IEnumerable<Claim> claims, CancellationToken cancellationTok
```

GetClaimsAsync(*TUser*, *System.Threading.CancellationToken*)

Gets a list of System.Security.Claims.Claim's to be belonging to the specified ''user' as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The role whose claims to retrieve.
- **cancellationToken** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task{System.Collections.Generic.IList{System.Security.Claims.Claim}}

Returns A <see cref="T:System.Threading.Tasks.Task'1" /> that represents the result of the asynchronous query, a list of <see cref="T:System.Security.Claims.Claim" />s.

```
Task<IList<Claim>> GetClaimsAsync(TUser user, CancellationToken cancellationToken)
```

GetUsersForClaimAsync(*System.Security.Claims.Claim*, *System.Threading.CancellationToken*)

Returns a list of users who contain the specified System.Security.Claims.Claim.

Arguments

- **claim** (*System.Security.Claims.Claim*) – The claim to look for.
- **cancellationToken** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task{System.Collections.Generic.IList{{TUser}}}

Returns A <see cref="T:System.Threading.Tasks.Task'1" /> that represents the result of the asynchronous query, a list of <typeparamref name="TUser" /> who contain the specified claim.

```
Task<IList<TUser>> GetUsersForClaimAsync(Claim claim, CancellationToken cancellationToken)
```

RemoveClaimsAsync(*TUser*, *System.Collections.Generic.IEnumerable<System.Security.Claims.Claim>*, *System.Threading.CancellationToken*)

Removes the specified `claims` from the given `user`.

Arguments

- **user** (*{TUser}*) – The user to remove the specified `claims` from.
- **claims** (*System.Collections.Generic.IEnumerable{System.Security.Claims.Claim}*) – A collection of :dn:ref:'System.Security.Claims.Claim's to remove.
- **cancellationToken** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task

Returns The task object representing the asynchronous operation.

```
Task RemoveClaimsAsync(TUser user, IEnumerable<Claim> claims, CancellationToken cancellation
```

ReplaceClaimAsync(*TUser*, *System.Security.Claims.Claim*, *System.Security.Claims.Claim*, *System.Threading.CancellationToken*)

Replaces the given `claim` on the specified `user` with the `newClaim`

Arguments

- **user** (*{TUser}*) – The user to replace the claim on.
- **claim** (*System.Security.Claims.Claim*) – The claim to replace.
- **newClaim** (*System.Security.Claims.Claim*) – The new claim to replace the existing `claim` with.
- **cancellationToken** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task

Returns The task object representing the asynchronous operation.

```
Task ReplaceClaimAsync(TUser user, Claim claim, Claim newClaim, CancellationToken cancellati
```

1.2.17 IUserClaimsPrincipalFactory<TUser> Interface

Summary

Provides an abstraction for a factory to create a System.Security.Claims.ClaimsPrincipal from a user.

Syntax

```
public interface IUserClaimsPrincipalFactory<TUser> where TUser : class
```

GitHub

View on GitHub

interface `Microsoft.AspNet.Identity.`**IUserClaimsPrincipalFactory<TUser>**

Methods

Creates a System.Security.Claims.ClaimsPrincipal from an user asynchronously.

CreateAsync(*TUser*) **Arguments**

- **user** (*{TUser}*) – The user to create a System.Security.Claims.ClaimsPrincipal from.

Return type System.Threading.Tasks.Task{System.Security.Claims.ClaimsPrincipal}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous creation operation, containing the created <see cref="T:System.Security.Claims.ClaimsPrincipal" />.

```
Task<ClaimsPrincipal> CreateAsync(TUser user)
```

1.2.18 IUserEmailStore<TUser> Interface

Summary

Provides an abstraction for the storage and management of user email addresses.

Syntax

```
public interface IUserEmailStore<TUser> : IUserStore<TUser>, IDisposable where TUser : class
```

GitHub

View on GitHub

interface `Microsoft.AspNet.Identity.`**`IUserEmailStore<TUser>`**

Methods

Gets the user, if any, associated with the specified, normalized email address, as an asynchronous operation.

`FindByEmailAsync`(*System.String*, *System.Threading.CancellationToken*)

Arguments

- **`normalizedEmail`** (*System.String*) – The normalized email address to return the user for.
- **`cancellationToken`** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task{{TUser}}

Returns The task object containing the results of the asynchronous lookup operation, the user if any associated with the specified normalized email address.

```
Task<TUser> FindByEmailAsync(string normalizedEmail, CancellationToken cancellationToken)
```

GetEmailAsync(*TUser*, *System.Threading.CancellationToken*)

Gets the email address for the specified `user`, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user whose email should be returned.
- **cancellationToken** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task{System.String}

Returns The task object containing the results of the asynchronous operation, the email address for the specified <paramref name="user" />.

```
Task<string> GetEmailAsync(TUser user, CancellationToken cancellationToken)
```

GetEmailConfirmedAsync(*TUser*, *System.Threading.CancellationToken*)

Gets a flag indicating whether the email address for the specified `user` has been verified, true if the email address is verified otherwise false, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user whose email confirmation status should be returned.
- **cancellationToken** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task{System.Boolean}

Returns The task object containing the results of the asynchronous operation, a flag indicating whether the email address for the specified <paramref name="user" /> has been confirmed or not.

```
Task<bool> GetEmailConfirmedAsync(TUser user, CancellationToken cancellationToken)
```

GetNormalizedEmailAsync(*TUser*, *System.Threading.CancellationToken*)

Returns the normalized email for the specified `user`, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user whose email address to retrieve.
- **cancellationToken** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task{System.String}

Returns The task object containing the results of the asynchronous lookup operation, the normalized email address if any associated with the specified user.

```
Task<string> GetNormalizedEmailAsync(TUser user, CancellationToken cancellationToken)
```

SetEmailAsync(*TUser*, *System.String*, *System.Threading.CancellationToken*)

Sets the `email` address for a `user`, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user whose email should be set.

- **email** (*System.String*) – The email to set.
- **cancellationToken** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task

Returns The task object representing the asynchronous operation.

```
Task SetEmailAsync(TUser user, string email, CancellationToken cancellationToken)
```

SetEmailConfirmedAsync(*TUser*, *System.Boolean*, *System.Threading.CancellationToken*)

Sets the flag indicating whether the specified `user`'s email address has been confirmed or not, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user whose email confirmation status should be set.
- **confirmed** (*System.Boolean*) – A flag indicating if the email address has been confirmed, true if the address is confirmed otherwise false.
- **cancellationToken** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task

Returns The task object representing the asynchronous operation.

```
Task SetEmailConfirmedAsync(TUser user, bool confirmed, CancellationToken cancellationToken)
```

SetNormalizedEmailAsync(*TUser*, *System.String*, *System.Threading.CancellationToken*)

Sets the normalized email for the specified `user`, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user whose email address to set.
- **normalizedEmail** (*System.String*) – The normalized email to set for the specified `user`.
- **cancellationToken** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task

Returns The task object representing the asynchronous operation.

```
Task SetNormalizedEmailAsync(TUser user, string normalizedEmail, CancellationToken cancellat
```

1.2.19 IUserLockoutStore<TUser> Interface

Summary

Provides an abstraction for a storing information which can be used to implement account lockout, including access failures and lockout status

Syntax

```
public interface IUserLockoutStore<TUser> : IUserStore<TUser>, IDisposable where TUser : class
```

GitHub

View on GitHub

interface `Microsoft.AspNet.Identity.`**`IUserLockoutStore<TUser>`**

Methods

Retrieves the current failed access count for the specified `user`, as an asynchronous operation..

`GetAccessFailedCountAsync`(*TUser*, *System.Threading.CancellationToken*) **Arguments**

- **`user`** (*{TUser}*) – The user whose failed access count should be retrieved.
- **`cancellationToken`** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task{System.Int32}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the failed access count.

```
Task<int> GetAccessFailedCountAsync(TUser user, CancellationToken cancellationToken)
```

`GetLockoutEnabledAsync`(*TUser*, *System.Threading.CancellationToken*)

Retrieves a flag indicating whether user lockout can enabled for the specified user, as an asynchronous operation.

Arguments

- **`user`** (*{TUser}*) – The user whose ability to be locked out should be returned.
- **`cancellationToken`** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task{System.Boolean}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, true if a user can be locked out, otherwise false.

```
Task<bool> GetLockoutEnabledAsync(TUser user, CancellationToken cancellationToken)
```

`GetLockoutEndDateAsync`(*TUser*, *System.Threading.CancellationToken*)

Gets the last System.DateTimeOffset a user's last lockout expired, if any, as an asynchronous operation. Any time in the past should be indicates a user is not locked out.

Arguments

- **user** (*{TUser}*) – The user whose lockout date should be retrieved.
- **cancellationToken** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task{System.Nullable{System.DateTimeOffset}}

Returns A <see cref="T:System.Threading.Tasks.Task`1" /> that represents the result of the asynchronous query, a <see cref="T:System.DateTimeOffset" /> containing the last time a user's lockout expired, if any.

```
Task<DateTimeOffset? > GetLockoutEndDateAsync(TUser user, CancellationToken cancellationToke
```

IncrementAccessFailedCountAsync(*TUser*, *System.Threading.CancellationToken*)

Records that a failed access has occurred, incrementing the failed access count, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user whose cancellation count should be incremented.
- **cancellationToken** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task{System.Int32}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the incremented failed access count.

```
Task<int> IncrementAccessFailedCountAsync(TUser user, CancellationToken cancellationToken)
```

ResetAccessFailedCountAsync(*TUser*, *System.Threading.CancellationToken*)

Resets a user's failed access count, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user whose failed access count should be reset.
- **cancellationToken** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation.

```
Task ResetAccessFailedCountAsync(TUser user, CancellationToken cancellationToken)
```

SetLockoutEnabledAsync(*TUser*, *System.Boolean*, *System.Threading.CancellationToken*)

Set the flag indicating if the specified `user` can be locked out, as an asynchronous operation..

Arguments

- **user** (*{TUser}*) – The user whose ability to be locked out should be set.
- **enabled** (*System.Boolean*) – A flag indicating if lock out can be enabled for the specified `user`.
- **cancellationToken** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation.

```
Task SetLockoutEnabledAsync(TUser user, bool enabled, CancellationToken cancellationToken)
```

SetLockoutEndDateAsync(*TUser*, *System.Nullable<System.DateTimeOffset>*, *System.Threading.CancellationToken*)

Locks out a user until the specified end date has passed, as an asynchronous operation. Setting a end date in the past immediately unlocks a user.

Arguments

- **user** (*{TUser}*) – The user whose lockout date should be set.
- **lockoutEnd** (*System.Nullable{System.DateTimeOffset}*) – The System.DateTimeOffset after which the `user`'s lockout should end.
- **cancellationToken** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation.

```
Task SetLockoutEndDateAsync(TUser user, DateTimeOffset? lockoutEnd, CancellationToken cancel
```

1.2.20 IUserLoginStore<TUser> Interface

Summary

Provides an abstraction for storing information that maps external login information provided by Microsoft Account, Facebook etc. to a user account.

Syntax

```
public interface IUserLoginStore<TUser> : IUserStore<TUser>, IDisposable where TUser : class
```

GitHub

View on GitHub

interface `Microsoft.AspNet.Identity.`**IUserLoginStore<TUser>**

Methods

Adds an external Microsoft.AspNet.Identity.UserLoginInfo to the specified `user`, as an asynchronous operation.

AddLoginAsync(*TUser*, *Microsoft.AspNet.Identity.UserLoginInfo*, *System.Threading.CancellationToken*)

Arguments

- **user** (*{TUser}*) – The user to add the login to.
- **login** (*Microsoft.AspNet.Identity.UserLoginInfo*) – The external Microsoft.AspNet.Identity.UserLoginInfo to add to the specified `user`.
- **cancellationToken** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation.

```
Task AddLoginAsync(TUser user, UserLoginInfo login, CancellationToken cancellationToken)
```

FindByLoginAsync(*System.String*, *System.String*, *System.Threading.CancellationToken*)

Retrieves the user associated with the specified login provider and login provider key, as an asynchronous operation..

Arguments

- **loginProvider** (*System.String*) – The login provider who provided the `providerKey`.
- **providerKey** (*System.String*) – The key provided by the `loginProvider` to identify a user.
- **cancellationToken** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task{{TUser}}

Returns The <see cref="T:System.Threading.Tasks.Task" /> for the asynchronous operation, containing the user, if any which matched the specified login provider and key.

```
Task<TUser> FindByLoginAsync(string loginProvider, string providerKey, CancellationToken can
```

GetLoginsAsync(*TUser*, *System.Threading.CancellationToken*)

Retrieves the associated logins for the specified <param ref="user" />, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user whose associated logins to retrieve.
- **cancellationToken** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task{System.Collections.Generic.IList{Microsoft.AspNet.Identity.UserLoginIn

Returns The <see cref="T:System.Threading.Tasks.Task" /> for the asynchronous operation, containing a list of <see cref="T:Microsoft.AspNet.Identity.UserLoginInfo" /> for the specified <paramref name="user" />, if any.

```
Task<IList<UserLoginInfo>> GetLoginsAsync(TUser user, CancellationToken cancellationToken)
```

RemoveLoginAsync(*TUser*, *System.String*, *System.String*, *System.Threading.CancellationToken*)
Attempts to remove the provided login information from the specified `user`, as an asynchronous operation. and returns a flag indicating whether the removal succeed or not.

Arguments

- **user** (*{TUser}*) – The user to remove the login information from.
- **loginProvider** (*System.String*) – The login provide whose information should be removed.
- **providerKey** (*System.String*) – The key given by the external login provider for the specified user.
- **cancellationToken** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task

Returns The <see cref="T:System.Threading.Tasks.Task" /> that contains a flag the result of the asynchronous removing operation. The flag will be true if the login information was existed and removed, otherwise false.

```
Task RemoveLoginAsync(TUser user, string loginProvider, string providerKey, CancellationToke
```

1.2.21 IUserPasswordStore<TUser> Interface

Summary

Provides an abstraction for a store containing users' password hashes..

Syntax

```
public interface IUserPasswordStore<TUser> : IUserStore<TUser>, IDisposable where TUser : class
```

GitHub

View on GitHub

interface `Microsoft.AspNet.Identity.`**IUserPasswordStore<TUser>**

Methods

Gets the password hash for the specified `user`, as an asynchronous operation.

GetPasswordHashAsync (*TUser*, *System.Threading.CancellationToken*)

Arguments

- **user** (*{TUser}*) – The user whose password hash to retrieve.
- **cancellationToken** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task{System.String}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, returning the password hash for the specified <paramref name="user" />.

```
Task<string> GetPasswordHashAsync(TUser user, CancellationToken cancellationToken)
```

HasPasswordAsync (*TUser*, *System.Threading.CancellationToken*)

Gets a flag indicating whether the specified `user` has a password, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user to return a flag for, indicating whether they have a password or not.
- **cancellationToken** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task{System.Boolean}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, returning true if the specified <paramref name="user" /> has a password otherwise false.

```
Task<bool> HasPasswordAsync(TUser user, CancellationToken cancellationToken)
```

SetPasswordHashAsync (*TUser*, *System.String*, *System.Threading.CancellationToken*)

Sets the password hash for the specified `user`, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user whose password hash to set.
- **passwordHash** (*System.String*) – The password hash to set.
- **cancellationToken** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation.

```
Task SetPasswordHashAsync(TUser user, string passwordHash, CancellationToken cancellationTok
```

1.2.22 IUserPhoneNumberStore<TUser> Interface

Summary

Provides an abstraction for a store containing users' telephone numbers.

Syntax

```
public interface IUserPhoneNumberStore<TUser> : IUserStore<TUser>, IDisposable where TUser : class
```

GitHub

View on GitHub

interface `Microsoft.AspNet.Identity.`**`IUserPhoneNumberStore<TUser>`**

Methods

Gets the telephone number, if any, for the specified `user`, as an asynchronous operation.

`GetPhoneNumberAsync`(*TUser*, *System.Threading.CancellationToken*)

Arguments

- **`user`** (*{TUser}*) – The user whose telephone number should be retrieved.
- **`cancellationToken`** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task{System.String}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the user's telephone number, if any.

```
Task<string> GetPhoneNumberAsync(TUser user, CancellationToken cancellationToken)
```

`GetPhoneNumberConfirmedAsync`(*TUser*, *System.Threading.CancellationToken*)

Gets a flag indicating whether the specified `user`'s telephone number has been confirmed, as an asynchronous operation.

Arguments

- **`user`** (*{TUser}*) – The user to return a flag for, indicating whether their telephone number is confirmed.
- **`cancellationToken`** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task{System.Boolean}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, returning true if the specified <paramref name="user" /> has a confirmed telephone number otherwise false.

```
Task<bool> GetPhoneNumberConfirmedAsync(TUser user, CancellationToken cancellationToken)
```

SetPhoneNumberAsync(*TUser*, *System.String*, *System.Threading.CancellationToken*)

Sets the telephone number for the specified `user`, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user whose telephone number should be set.
- **phoneNumber** (*System.String*) – The telephone number to set.
- **cancellationToken** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation.

```
Task SetPhoneNumberAsync(TUser user, string phoneNumber, CancellationToken cancellationToken
```

SetPhoneNumberConfirmedAsync(*TUser*, *System.Boolean*, *System.Threading.CancellationToken*)

Sets a flag indicating if the specified `user`'s phone number has been confirmed, as an asynchronous operation..

Arguments

- **user** (*{TUser}*) – The user whose telephone number confirmation status should be set.
- **confirmed** (*System.Boolean*) – A flag indicating whether the user's telephone number has been confirmed.
- **cancellationToken** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation.

```
Task SetPhoneNumberConfirmedAsync(TUser user, bool confirmed, CancellationToken cancellation
```

1.2.23 IUserRoleStore<TUser> Interface

Summary

Provides an abstraction for a store which maps users to roles.

Syntax

```
public interface IUserRoleStore<TUser> : IUserStore<TUser>, IDisposable where TUser : class
```

GitHub

View on GitHub

interface `Microsoft.AspNet.Identity.`**`IUserRoleStore<TUser>`**

Methods

Add a the specified `user` to the named role, as an asynchronous operation.

`AddToRoleAsync`(*TUser*, *System.String*, *System.Threading.CancellationToken*) **Arguments**

- **`user`** (*{TUser}*) – The user to add to the named role.
- **`roleName`** (*System.String*) – The name of the role to add the user to.
- **`cancellationToken`** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation.

```
Task AddToRoleAsync(TUser user, string roleName, CancellationToken cancellationToken)
```

`GetRolesAsync`(*TUser*, *System.Threading.CancellationToken*)

Gets a list of role names the specified `user` belongs to, as an asynchronous operation.

Arguments

- **`user`** (*{TUser}*) – The user whose role names to retrieve.
- **`cancellationToken`** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task{System.Collections.Generic.IList{System.String}}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing a list of role names.

```
Task<IList<string>> GetRolesAsync(TUser user, CancellationToken cancellationToken)
```

`GetUsersInRoleAsync`(*System.String*, *System.Threading.CancellationToken*)

Returns a list of Users who are members of the named role.

Arguments

- **`roleName`** (*System.String*) – The name of the role whose membership should be returned.

- **cancellationToken** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task{System.Collections.Generic.IList{{TUser}}}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing a list of users who are in the named role.

```
Task<IList<TUser>> GetUsersInRoleAsync(string roleName, CancellationToken cancellationToken)
```

IsInRoleAsync(*TUser*, *System.String*, *System.Threading.CancellationToken*)

Returns a flag indicating whether the specified `user` is a member of the give named role, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user whose role membership should be checked.
- **roleName** (*System.String*) – The name of the role to be checked.
- **cancellationToken** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task{System.Boolean}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing a flag indicating whether the specified <see cref="!:user" /> is a member of the named role.

```
Task<bool> IsInRoleAsync(TUser user, string roleName, CancellationToken cancellationToken)
```

RemoveFromRoleAsync(*TUser*, *System.String*, *System.Threading.CancellationToken*)

Add a the specified `user` from the named role, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user to remove the named role from.
- **roleName** (*System.String*) – The name of the role to remove.
- **cancellationToken** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation.

```
Task RemoveFromRoleAsync(TUser user, string roleName, CancellationToken cancellationToken)
```

1.2.24 IUserSecurityStampStore<TUser> Interface

Summary

Provides an abstraction for a store which stores a user's security stamp.

Syntax

```
public interface IUserSecurityStampStore<TUser> : IUserStore<TUser>, IDisposable where TUser : class
```

GitHub

View on GitHub

interface `Microsoft.AspNet.Identity.`**`IUserSecurityStampStore<TUser>`**

Methods

Get the security stamp for the specified `user`, as an asynchronous operation.

`GetSecurityStampAsync`(*TUser*, *System.Threading.CancellationToken*)

Arguments

- **`user`** (*{TUser}*) – The user whose security stamp should be set.
- **`cancellationToken`** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task{System.String}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the security stamp for the specified <paramref name="user" />.

```
Task<string> GetSecurityStampAsync(TUser user, CancellationToken cancellationToken)
```

`SetSecurityStampAsync`(*TUser*, *System.String*, *System.Threading.CancellationToken*)

Sets the provided security `stamp` for the specified `user`, as an asynchronous operation.

Arguments

- **`user`** (*{TUser}*) – The user whose security stamp should be set.
- **`stamp`** (*System.String*) – The security stamp to set.
- **`cancellationToken`** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation.

```
Task SetSecurityStampAsync(TUser user, string stamp, CancellationToken cancellationToken)
```

1.2.25 IUserStore<TUser> Interface

Summary

Provides an abstraction for a store which manages user accounts.

Syntax

```
public interface IUserStore<TUser> : IDisposable where TUser : class
```

GitHub

View on GitHub

interface `Microsoft.AspNet.Identity.`**`IUserStore<TUser>`**

Methods

Creates the specified `user` in the user store, as an asynchronous operation.

`CreateAsync`(*TUser*, *System.Threading.CancellationToken*)

Arguments

- **`user`** (*{TUser}*) – The user to create.
- **`cancellationToken`** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the <see cref="T:Microsoft.AspNet.Identity.IdentityResult" /> of the creation operation.

```
Task<IdentityResult> CreateAsync(TUser user, CancellationToken cancellationToken)
```

`DeleteAsync`(*TUser*, *System.Threading.CancellationToken*)

Deletes the specified `user` from the user store, as an asynchronous operation.

Arguments

- **`user`** (*{TUser}*) – The user to delete.
- **`cancellationToken`** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the <see cref="T:Microsoft.AspNet.Identity.IdentityResult" /> of the update operation.

```
Task<IdentityResult> DeleteAsync(TUser user, CancellationToken cancellationToken)
```

FindByIdAsync(*System.String*, *System.Threading.CancellationToken*)
Finds and returns a user, if any, who has the specified `userId`.

Arguments

- **userId** (*System.String*) – The user ID to search for.
- **cancellationToken** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task{{TUser}}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the user matching the specified <paramref name="userID" /> if it exists.

```
Task<TUser> FindByIdAsync(string userId, CancellationToken cancellationToken)
```

FindByNameAsync(*System.String*, *System.Threading.CancellationToken*)
Finds and returns a user, if any, who has the specified normalized user name.

Arguments

- **normalizedUserName** (*System.String*) – The normalized user name to search for.
- **cancellationToken** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task{{TUser}}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the user matching the specified <paramref name="userID" /> if it exists.

```
Task<TUser> FindByNameAsync(string normalizedUserName, CancellationToken cancellationToken)
```

GetNormalizedUserNameAsync(*TUser*, *System.Threading.CancellationToken*)
Gets the normalized user name for the specified `user`, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user whose normalized name should be retrieved.
- **cancellationToken** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task{System.String}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the normalized user name for the specified <paramref name="user" />.

```
Task<string> GetNormalizedUserNameAsync(TUser user, CancellationToken cancellationToken)
```

GetUserIdAsync(*TUser*, *System.Threading.CancellationToken*)
Gets the user identifier for the specified `user`, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user whose identifier should be retrieved.
- **cancellationToken** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task{System.String}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the identifier for the specified <paramref name="user" />.

```
Task<string> GetUserIdAsync(TUser user, CancellationToken cancellationToken)
```

GetUserNameAsync(*TUser*, *System.Threading.CancellationToken*)

Gets the user name for the specified `user`, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user whose name should be retrieved.
- **cancellationToken** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task{System.String}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the name for the specified <paramref name="user" />.

```
Task<string> GetUserNameAsync(TUser user, CancellationToken cancellationToken)
```

SetNormalizedUserNameAsync(*TUser*, *System.String*, *System.Threading.CancellationToken*)

Sets the given normalized name for the specified `user`, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user whose name should be set.
- **cancellationToken** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation.

```
Task SetNormalizedUserNameAsync(TUser user, string normalizedName, CancellationToken cancell
```

SetUserNameAsync(*TUser*, *System.String*, *System.Threading.CancellationToken*)

Sets the given `userName` for the specified `user`, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user whose name should be set.
- **userName** (*System.String*) – The user name to set.
- **cancellationToken** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation.

```
Task SetUserNameAsync(TUser user, string userName, CancellationToken cancellationToken)
```

UpdateAsync(*TUser*, *System.Threading.CancellationToken*)
Updates the specified `user` in the user store, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user to update.
- **cancellationToken** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the <see cref="T:Microsoft.AspNet.Identity.IdentityResult" /> of the update operation.

```
Task<IdentityResult> UpdateAsync(TUser user, CancellationToken cancellationToken)
```

1.2.26 IUserTokenProvider Interface

Summary

Provides an abstraction for token generators.

Syntax

```
public interface IUserTokenProvider
```

GitHub

View on GitHub

interface `Microsoft.AspNet.Identity.`**IUserTokenProvider**

Methods

Returns a flag indicating whether the token provider can generate a token suitable for two factor authentication token for the specified `ref`.

CanGenerateTwoFactorTokenAsync<TUser> (*Microsoft.AspNet.Identity.UserManager<TUser>*, *TUser*)

Arguments

- **manager** (*Microsoft.AspNet.Identity.UserManager{{TUser}}*) – The Microsoft.AspNet.Identity.UserManager`1 that can be used to retrieve user properties.
- **user** (*{TUser}*) – The user a token could be generated for.

Return type System.Threading.Tasks.Task{System.Boolean}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the a flag indicating if a two factor token could be generated by this provider for the specified <paramref name="user" /> and <paramref name="purpose" />. The task will return true if a two factor authentication token could be generated, otherwise false.

```
Task<bool> CanGenerateTwoFactorTokenAsync<TUser>(UserManager<TUser> manager, TUser user)wher
```

GenerateAsync<TUser> (*System.String*, *Microsoft.AspNet.Identity.UserManager<TUser>*, *TUser*)

Generates a token for the specified `ref` and `purpose`, as an asynchronous operation.

Arguments

- **purpose** (*System.String*) – The purpose the token will be used for.
- **manager** (*Microsoft.AspNet.Identity.UserManager{{TUser}}*) – The Microsoft.AspNet.Identity.UserManager`1 that can be used to retrieve user properties.
- **user** (*{TUser}*) – The user a token should be generated for.

Return type System.Threading.Tasks.Task{System.String}

Returns

The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the token for the specified

<paramref name="user" /> and <paramref name="purpose" />.

```
Task<string> GenerateAsync<TUser>(string purpose, UserManager<TUser> manager, TUser user)whe
```

ValidateAsync<TUser> (*System.String*, *System.String*, *Microsoft.AspNet.Identity.UserManager<TUser>*, *TUser*)

Returns a flag indicating whether the specified `token` is valid for the given

`user` and `purpose`, as an asynchronous operation.

Arguments

- **purpose** (*System.String*) – The purpose the token will be used for.
- **token** (*System.String*) – The token to validate.
- **manager** (*Microsoft.AspNet.Identity.UserManager{{TUser}}*) – The Microsoft.AspNet.Identity.UserManager`1 that can be used to retrieve user properties.
- **user** (*{TUser}*) – The user a token should be validated for.

Return type System.Threading.Tasks.Task{System.Boolean}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the a flag indicating the result of validating the <paramref name="token"> for the specified </paramref> <paramref name="user" /> and <paramref name="purpose" />. The task will return true if the token is valid, otherwise false.

```
Task<bool> ValidateAsync<TUser>(string purpose, string token, UserManager<TUser> manager, TU
```

Properties

Gets the name of the token provider.

Name()

Return type
System.String

```
string Name { get; }
```

1.2.27 IUserTwoFactorStore<TUser> Interface

Summary

Provides an abstraction to store a flag indicating whether a user has two factor authentication enabled.

Syntax

```
public interface IUserTwoFactorStore<TUser> : IUserStore<TUser>, IDisposable where TUser : class
```

GitHub

View on GitHub

interface Microsoft.AspNet.Identity.**IUserTwoFactorStore<TUser>**

Methods

Returns a flag indicating whether the specified ‘‘user ‘‘has two factor authentication enabled or not, as an asynchronous operation.

GetTwoFactorEnabledAsync(*TUser*, *System.Threading.CancellationToken*)

Arguments

- **user** (*{TUser}*) – The user whose two factor authentication enabled status should be set.
- **cancellationToken** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task{System.Boolean}

Returns

The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing a flag indicating whether the specified

<paramref name="user " />has two factor authentication enabled or not.

```
Task<bool> GetTwoFactorEnabledAsync(TUser user, CancellationToken cancellationToken)
```

SetTwoFactorEnabledAsync(*TUser*, *System.Boolean*, *System.Threading.CancellationToken*)

Sets a flag indicating whether the specified ‘‘user ‘‘has two factor authentication enabled or not, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user whose two factor authentication enabled status should be set.
- **enabled** (*System.Boolean*) – A flag indicating whether the specified `user` has two factor authentication enabled.
- **cancellationToken** (*System.Threading.CancellationToken*) – The System.Threading.CancellationToken used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation.

```
Task SetTwoFactorEnabledAsync(TUser user, bool enabled, CancellationToken cancellationToken)
```

1.2.28 IUserValidator Interface

Summary

Provides an abstraction for user validation.

Syntax

```
public interface IUserValidator
```

GitHub

View on GitHub

interface `Microsoft.AspNet.Identity.`**IUserValidator**

Methods

Validates the specified `user` as an asynchronous operation.

`ValidateAsync<TUser>` (*Microsoft.AspNet.Identity.UserManager<TUser>*, *TUser*)

Arguments

- **`manager`** (*Microsoft.AspNet.Identity.UserManager{TUser}*) – The Microsoft.AspNet.Identity.UserManager`1 that can be used to retrieve user properties.
- **`user`** (*{TUser}*) – The user to validate.

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the <see cref="T:Microsoft.AspNet.Identity.IdentityResult" /> of the validation operation.

```
Task<IdentityResult> ValidateAsync<TUser>(UserManager<TUser> manager, TUser user)where TUser
```

1.2.29 IdentityBuilder Class

Summary

Helper functions for configuring identity services.

Inheritance Hierarchy

- `System.Object`
- *`Microsoft.AspNet.Identity.IdentityBuilder`*

Syntax

```
public class IdentityBuilder
```

GitHub

View on GitHub

class `Microsoft.AspNet.Identity.`**`IdentityBuilder`**

Constructors

Creates a new instance of Microsoft.AspNet.Identity.IdentityBuilder.

IdentityBuilder (*System.Type*, *System.Type*, *IServiceCollection*)

Arguments

- **user** (*System.Type*) – The System.Type to use for the users.
- **role** (*System.Type*) – The System.Type to use for the roles.
- **services** (*IServiceCollection*) – The IServiceCollection to attach to.

```
public IdentityBuilder(Type user, Type role, IServiceCollection services)
```

Methods

Adds the default token providers.

AddDefaultTokenProviders ()

Return type
Microsoft.AspNet.Identity.IdentityBuilder

Returns The <see cref="T:Microsoft.AspNet.Identity.IdentityBuilder" />.

```
public virtual IdentityBuilder AddDefaultTokenProviders()
```

AddErrorDescriber<TDescriber> ()

Adds an Microsoft.AspNet.Identity.IdentityErrorDescriber.

Return type Microsoft.AspNet.Identity.IdentityBuilder

Returns The <see cref="T:Microsoft.AspNet.Identity.IdentityBuilder" />.

```
public virtual IdentityBuilder AddErrorDescriber<TDescriber>()where TDescriber : IdentityErr
```

AddPasswordValidator<TValidator> ()

Adds an IPasswordValidator<TUser> for the *Microsoft.AspNet.Identity.IdentityBuilder.UserTyp*

Return type Microsoft.AspNet.Identity.IdentityBuilder

Returns The <see cref="T:Microsoft.AspNet.Identity.IdentityBuilder" />.

```
public virtual IdentityBuilder AddPasswordValidator<TValidator>()where TValidator : class, I
```

AddRoleManager<TRoleManager> ()

Adds a Microsoft.AspNet.Identity.RoleManager`1 for the *Microsoft.AspNet.Identity.IdentityBuilder.Ro*

Return type Microsoft.AspNet.Identity.IdentityBuilder

Returns The <see cref="T:Microsoft.AspNet.Identity.IdentityBuilder" />.

```
public virtual IdentityBuilder AddRoleManager<TRoleManager>()where TRoleManager : class
```

AddRoleStore<T> ()

Adds a Microsoft.AspNet.Identity.IRoleStore`1 for the *Microsoft.AspNet.Identity.IdentityBuilder.Role*

Return type Microsoft.AspNet.Identity.IdentityBuilder

Returns The <see cref="T:Microsoft.AspNet.Identity.IdentityBuilder" />.

```
public virtual IdentityBuilder AddRoleStore<T>()where T : class
```

AddRoleValidator<TValidator> ()

Adds an IRoleValidator<TRole> for the *Microsoft.AspNet.Identity.IdentityBuilder.RoleType*.

Return type Microsoft.AspNet.Identity.IdentityBuilder

Returns The <see cref="T:Microsoft.AspNet.Identity.IdentityBuilder" />.

```
public virtual IdentityBuilder AddRoleValidator<TValidator>()where TValidator : class, IRole
```

AddTokenProvider<TProvider>()
Adds a token provider.

Return type Microsoft.AspNet.Identity.IdentityBuilder

Returns The <see cref="T:Microsoft.AspNet.Identity.IdentityBuilder" />.

```
public virtual IdentityBuilder AddTokenProvider<TProvider>()where TProvider : class, IUserTo
```

AddUserManager<TUserManager>()
Adds a Microsoft.AspNet.Identity.UserManager`1 for the *Microsoft.AspNet.Identity.IdentityBuilder.Us*

Return type Microsoft.AspNet.Identity.IdentityBuilder

Returns The <see cref="T:Microsoft.AspNet.Identity.IdentityBuilder" />.

```
public virtual IdentityBuilder AddUserManager<TUserManager>()where TUserManager : class
```

AddUserStore<T>()
Adds an Microsoft.AspNet.Identity.IUserStore`1 for the *Microsoft.AspNet.Identity.IdentityBuilder.Use*

Return type Microsoft.AspNet.Identity.IdentityBuilder

Returns The <see cref="T:Microsoft.AspNet.Identity.IdentityBuilder" />.

```
public virtual IdentityBuilder AddUserStore<T>()where T : class
```

AddUserValidator<TValidator>()
Adds an Microsoft.AspNet.Identity.IUserValidator for the *Microsoft.AspNet.Identity.IdentityBuilder.Us*

Return type Microsoft.AspNet.Identity.IdentityBuilder

Returns The <see cref="T:Microsoft.AspNet.Identity.IdentityBuilder" />.

```
public virtual IdentityBuilder AddUserValidator<TValidator>()where TValidator : class, IUser
```

Properties

Gets the System.Type used for roles.

RoleType() **Return type**
System.Type

```
public Type RoleType { get; }
```

Services()
Gets the IServiceCollection services are attached to.

Return type IServiceCollection

```
public IServiceCollection Services { get; }
```

UserType()
Gets the System.Type used for users.

Return type System.Type

```
        public Type UserType { get; }
```

1.2.30 IdentityEntityFrameworkServices Class

Summary

Default services

Inheritance Hierarchy

- `System.Object`
- *`Microsoft.AspNet.Identity.IdentityEntityFrameworkServices`*

Syntax

```
public class IdentityEntityFrameworkServices
```

GitHub

View on GitHub

class `Microsoft.AspNet.Identity.`**`IdentityEntityFrameworkServices`**

Methods

`GetDefaultServices`(*System.Type*, *System.Type*, *System.Type*, *System.Type*) **Return type**
IServiceCollection

```
        public static IServiceCollection GetDefaultServices(Type userType, Type roleType, Type conte
```

1.2.31 IdentityError Class

Summary

Encapsulates an error from the identity subsystem.

Inheritance Hierarchy

- `System.Object`
- *`Microsoft.AspNet.Identity.IdentityError`*

Syntax

```
public class IdentityError
```

GitHub

View on GitHub

class `Microsoft.AspNet.Identity.`**`IdentityError`**

Properties

Gets or sets the code for this error.

`Code``()`

Return type System.String

```
public string Code { get; set; }
```

`Description``()`

Gets or sets the description for this error.

Return type System.String

```
public string Description { get; set; }
```

1.2.32 IdentityErrorDescriber Class

Summary

Service to enable localization for application facing identity errors.

Inheritance Hierarchy

- `System.Object`
- *`Microsoft.AspNet.Identity.IdentityErrorDescriber`*

Syntax

```
public class IdentityErrorDescriber
```

GitHub

View on GitHub

class `Microsoft.AspNet.Identity.`**`IdentityErrorDescriber`**

Methods

Returns an Microsoft.AspNet.Identity.IdentityError indicating a concurrency failure.

`ConcurrencyFailure``()` **Return type**
Microsoft.AspNet.Identity.IdentityError

Returns An <see cref="T:Microsoft.AspNet.Identity.IdentityError" /> indicating a concurrency failure.

```
public virtual IdentityError ConcurrencyFailure()
```

`DefaultError``()`
Returns the default Microsoft.AspNet.Identity.IdentityError.

Return type Microsoft.AspNet.Identity.IdentityError

Returns The default <see cref="T:Microsoft.AspNet.Identity.IdentityError" />,

```
public virtual IdentityError DefaultError()
```

`DuplicateEmail`(*System.String*)
Returns an Microsoft.AspNet.Identity.IdentityError indicating the specified `email` is already associated with an account.

Arguments

- **`email`** (*System.String*) – The email that is already associated with an account.

Return type Microsoft.AspNet.Identity.IdentityError

Returns An <see cref="T:Microsoft.AspNet.Identity.IdentityError" /> indicating the specified <paramref name="email" /> is already associated with an account.

```
public virtual IdentityError DuplicateEmail(string email)
```

`DuplicateRoleName`(*System.String*)
Returns an Microsoft.AspNet.Identity.IdentityError indicating the specified `role` name already exists.

Arguments

- **`role`** (*System.String*) – The duplicate role.

Return type Microsoft.AspNet.Identity.IdentityError

Returns An <see cref="T:Microsoft.AspNet.Identity.IdentityError" /> indicating the specific role <paramref name="role" /> name already exists.

```
public virtual IdentityError DuplicateRoleName(string role)
```

DuplicateUserName (*System.String*)
Returns an Microsoft.AspNet.Identity.IdentityError indicating the specified `userName` already exists.

Arguments

- **userName** (*System.String*) – The user name that already exists.

Return type Microsoft.AspNet.Identity.IdentityError

Returns An <see cref="T:Microsoft.AspNet.Identity.IdentityError" /> indicating the specified <paramref name="userName" /> already exists.

```
public virtual IdentityError DuplicateUserName(string userName)
```

InvalidEmail (*System.String*)
Returns an Microsoft.AspNet.Identity.IdentityError indicating the specified `email` is invalid.

Arguments

- **email** (*System.String*) – The email that is invalid.

Return type Microsoft.AspNet.Identity.IdentityError

Returns An <see cref="T:Microsoft.AspNet.Identity.IdentityError" /> indicating the specified <paramref name="email" /> is invalid.

```
public virtual IdentityError InvalidEmail(string email)
```

InvalidRoleName (*System.String*)
Returns an Microsoft.AspNet.Identity.IdentityError indicating the specified `role` name is invalid.

Arguments

- **role** (*System.String*) – The invalid role.

Return type Microsoft.AspNet.Identity.IdentityError

Returns An <see cref="T:Microsoft.AspNet.Identity.IdentityError" /> indicating the specific role <paramref name="role" /> name is invalid.

```
public virtual IdentityError InvalidRoleName(string role)
```

InvalidToken ()
Returns an Microsoft.AspNet.Identity.IdentityError indicating an invalid token.

Return type Microsoft.AspNet.Identity.IdentityError

Returns An <see cref="T:Microsoft.AspNet.Identity.IdentityError" /> indicating an invalid token.

```
public virtual IdentityError InvalidToken()
```

InvalidUserName (*System.String*)
Returns an Microsoft.AspNet.Identity.IdentityError indicating the specified user `userName` is invalid.

Arguments

- **userName** (*System.String*) – The user name that is invalid.

Return type Microsoft.AspNet.Identity.IdentityError

Returns An <see cref="T:Microsoft.AspNet.Identity.IdentityError" /> indicating the specified user <paramref name="userName" /> is invalid.

```
public virtual IdentityError InvalidUserName(string userName)
```

LoginAlreadyAssociated()

Returns an Microsoft.AspNet.Identity.IdentityError indicating an external login is already associated with an account.

Return type Microsoft.AspNet.Identity.IdentityError

Returns An <see cref="T:Microsoft.AspNet.Identity.IdentityError" /> indicating an external login is already associated with an account.

```
public virtual IdentityError LoginAlreadyAssociated()
```

PasswordMismatch()

Returns an Microsoft.AspNet.Identity.IdentityError indicating a password mismatch.

Return type Microsoft.AspNet.Identity.IdentityError

Returns An <see cref="T:Microsoft.AspNet.Identity.IdentityError" /> indicating a password mismatch.

```
public virtual IdentityError PasswordMismatch()
```

PasswordRequiresDigit()

Returns an Microsoft.AspNet.Identity.IdentityError indicating a password entered does not contain a numeric character, which is required by the password policy.

Return type Microsoft.AspNet.Identity.IdentityError

Returns An <see cref="T:Microsoft.AspNet.Identity.IdentityError" /> indicating a password entered does not contain a numeric character.

```
public virtual IdentityError PasswordRequiresDigit()
```

PasswordRequiresLower()

Returns an Microsoft.AspNet.Identity.IdentityError indicating a password entered does not contain a lower case letter, which is required by the password policy.

Return type Microsoft.AspNet.Identity.IdentityError

Returns An <see cref="T:Microsoft.AspNet.Identity.IdentityError" /> indicating a password entered does not contain a lower case letter.

```
public virtual IdentityError PasswordRequiresLower()
```

PasswordRequiresNonLetterAndDigit()

Returns an Microsoft.AspNet.Identity.IdentityError indicating a password entered does not contain a non-alphanumeric character, which is required by the password policy.

Return type Microsoft.AspNet.Identity.IdentityError

Returns An <see cref="T:Microsoft.AspNet.Identity.IdentityError" /> indicating a password entered does not contain a non-alphanumeric character.

```
public virtual IdentityError PasswordRequiresNonLetterAndDigit()
```

PasswordRequiresUpper()

Returns an Microsoft.AspNet.Identity.IdentityError indicating a password entered does not contain an upper case letter, which is required by the password policy.

Return type Microsoft.AspNet.Identity.IdentityError

Returns An <see cref="T:Microsoft.AspNet.Identity.IdentityError" /> indicating a password entered does not contain an upper case letter.

```
public virtual IdentityError PasswordRequiresUpper()
```

PasswordTooShort(*System.Int32*)

Returns an Microsoft.AspNet.Identity.IdentityError indicating a password of the specified `length` does not meet the minimum length requirements.

Arguments

- **length** (*System.Int32*) – The length that is not long enough.

Return type Microsoft.AspNet.Identity.IdentityError

Returns An <see cref="T:Microsoft.AspNet.Identity.IdentityError" /> indicating a password of the specified <paramref name="length" /> does not meet the minimum length requirements.

```
public virtual IdentityError PasswordTooShort(int length)
```

UserAlreadyHasPassword()

Returns an Microsoft.AspNet.Identity.IdentityError indicating a user already has a password.

Return type Microsoft.AspNet.Identity.IdentityError

Returns An <see cref="T:Microsoft.AspNet.Identity.IdentityError" /> indicating a user already has a password.

```
public virtual IdentityError UserAlreadyHasPassword()
```

UserAlreadyInRole(*System.String*)

Returns an Microsoft.AspNet.Identity.IdentityError indicating a user is already in the specified `role`.

Arguments

- **role** (*System.String*) – The duplicate role.

Return type Microsoft.AspNet.Identity.IdentityError

Returns An <see cref="T:Microsoft.AspNet.Identity.IdentityError" /> indicating a user is already in the specified <paramref name="role" />.

```
public virtual IdentityError UserAlreadyInRole(string role)
```

UserLockoutNotEnabled()

Returns an Microsoft.AspNet.Identity.IdentityError indicating user lockout is not enabled.

Return type Microsoft.AspNet.Identity.IdentityError

Returns An <see cref="T:Microsoft.AspNet.Identity.IdentityError" /> indicating user lockout is not enabled..

```
public virtual IdentityError UserLockoutNotEnabled()
```

UserNotInRole(*System.String*)

Returns an Microsoft.AspNet.Identity.IdentityError indicating a user is not in the specified `role`.

Arguments

- **role** (*System.String*) – The duplicate role.

Return type Microsoft.AspNet.Identity.IdentityError

Returns An <see cref="T:Microsoft.AspNet.Identity.IdentityError" /> indicating a user is not in the specified <paramref name="role" />.

```
public virtual IdentityError UserNotInRole(string role)
```

1.2.33 IdentityOptions Class

Summary

Represents all the options you can used to configure the identity system.

Inheritance Hierarchy

- `System.Object`
- *`Microsoft.AspNet.Identity.IdentityOptions`*

Syntax

```
public class IdentityOptions
```

GitHub

View on GitHub

class `Microsoft.AspNet.Identity.`**`IdentityOptions`**

Properties

Gets or sets the scheme used to identify application authentication cookies.

`ApplicationCookieAuthenticationScheme``()` **Return type**
System.String

```
public static string ApplicationCookieAuthenticationScheme { get; set; }
```

`ApplicationCookieAuthenticationType``()`
Gets or sets the authentication type used when constructing an *`Microsoft.AspNet.Identity.IdentityOptions.ClaimsIdentity`* from an application cookie.

Return type System.String

```
public static string ApplicationCookieAuthenticationType { get; set; }
```

ChangeEmailTokenProvider()
Gets or sets the *Microsoft.AspNet.Identity.IdentityOptions.ChangeEmailTokenProvider* used to generate tokens used in email change confirmation emails.

Return type System.String

```
public string ChangeEmailTokenProvider { get; set; }
```

ClaimsIdentity()
Gets or sets the Microsoft.AspNet.Identity.ClaimsIdentityOptions for the identity system.

Return type Microsoft.AspNet.Identity.ClaimsIdentityOptions

```
public ClaimsIdentityOptions ClaimsIdentity { get; set; }
```

EmailConfirmationTokenProvider()
Gets or sets the *Microsoft.AspNet.Identity.IdentityOptions.EmailConfirmationTokenProvider* used to generate tokens used in account confirmation emails.

Return type System.String

```
public string EmailConfirmationTokenProvider { get; set; }
```

ExternalCookieAuthenticationScheme()
Gets or sets the scheme used to identify external authentication cookies.

Return type System.String

```
public static string ExternalCookieAuthenticationScheme { get; set; }
```

ExternalCookieAuthenticationType()
Gets or sets the authentication type used when constructing an *Microsoft.AspNet.Identity.IdentityOptions.ClaimsIdentity* from an external identity cookie.

Return type System.String

```
public static string ExternalCookieAuthenticationType { get; set; }
```

Lockout()
Gets or sets the Microsoft.AspNet.Identity.LockoutOptions for the identity system.

Return type Microsoft.AspNet.Identity.LockoutOptions

```
public LockoutOptions Lockout { get; set; }
```

Password()
Gets or sets the Microsoft.AspNet.Identity.PasswordOptions for the identity system.

Return type Microsoft.AspNet.Identity.PasswordOptions

```
public PasswordOptions Password { get; set; }
```

PasswordResetTokenProvider()
Gets or sets the *Microsoft.AspNet.Identity.IdentityOptions.PasswordResetTokenProvider* used to generate tokens used in password reset emails.

Return type System.String

```
public string PasswordResetTokenProvider { get; set; }
```

SecurityStampValidationInterval()
Gets or sets the System.TimeSpan after which security stamps are re-validated.

Return type System.TimeSpan

```
public TimeSpan SecurityStampValidationInterval { get; set; }
```

SignIn()
Gets or sets the Microsoft.AspNet.Identity.SignInOptions for the identity system.

Return type Microsoft.AspNet.Identity.SignInOptions

```
public SignInOptions SignIn { get; set; }
```

TwoFactorRememberMeCookieAuthenticationScheme()
Gets or sets the scheme used to identify Two Factor authentication cookies for saving the Remember Me state.

Return type System.String

```
public static string TwoFactorRememberMeCookieAuthenticationScheme { get; set; }
```

TwoFactorRememberMeCookieAuthenticationType()
Gets or sets the authentication type used when constructing an *Microsoft.AspNet.Identity.IdentityOptions.ClaimsIdentity* from an two factor remember me authentication cookie.

Return type System.String

```
public static string TwoFactorRememberMeCookieAuthenticationType { get; set; }
```

TwoFactorUserIdCookieAuthenticationScheme()
Gets or sets the scheme used to identify Two Factor authentication cookies for round tripping user identities.

Return type System.String

```
public static string TwoFactorUserIdCookieAuthenticationScheme { get; set; }
```

TwoFactorUserIdCookieAuthenticationType()
Gets or sets the authentication type used when constructing an *Microsoft.AspNet.Identity.IdentityOptions.ClaimsIdentity* from an two factor authentication cookie.

Return type System.String

```
public static string TwoFactorUserIdCookieAuthenticationType { get; set; }
```

User()
Gets or sets the Microsoft.AspNet.Identity.UserOptions for the identity system.

Return type Microsoft.AspNet.Identity.UserOptions

```
public UserOptions User { get; set; }
```

1.2.34 IdentityResult Class

Summary

Represents the result of an identity operation.

Inheritance Hierarchy

- `System.Object`
- *`Microsoft.AspNet.Identity.IdentityResult`*

Syntax

```
public class IdentityResult
```

GitHub

View on GitHub

class `Microsoft.AspNet.Identity.`**`IdentityResult`**

Properties

An System.Collections.Generic.IEnumerable‘1 of :dn:ref:‘Microsoft.AspNet.Identity.IdentityError‘s containing an errors that occurred during the identity operation.

`Errors``()`

Return type System.Collections.Generic.IEnumerable{Microsoft.AspNet.Identity.IdentityError}

```
public IEnumerable<IdentityError> Errors { get; }
```

`Succeeded``()`

Flag indicating whether if the operation succeeded or not.

Return type System.Boolean

```
public bool Succeeded { get; protected set; }
```

`Success``()`

Returns an Microsoft.AspNet.Identity.IdentityResult indicating a successful identity operation.

Return type Microsoft.AspNet.Identity.IdentityResult

Returns An <see cref=”T:Microsoft.AspNet.Identity.IdentityResult” /> indicating a successful operation.

```
public static IdentityResult Success { get; }
```

Methods

Creates an Microsoft.AspNet.Identity.IdentityResult indicating a failed identity operation, with a list of `errors` if applicable.

`Failed`(*Microsoft.AspNet.Identity.IdentityError[]*)

Arguments

- **`errors`** (*Microsoft.AspNet.Identity.IdentityError[]*) – An optional array of :dn:ref:‘Microsoft.AspNet.Identity.IdentityError‘s which caused the operation to fail.

Return type Microsoft.AspNet.Identity.IdentityResult

Returns An <see cref="T:Microsoft.AspNet.Identity.IdentityResult" /> indicating a failed identity operation, with a list of <paramref name="errors" /> if applicable.

```
public static IdentityResult Failed(params IdentityError[] errors)
```

`ToString`()

Converts the value of the current Microsoft.AspNet.Identity.IdentityResult object to its equivalent string representation.

Return type System.String

Returns A string representation of the current <see cref="T:Microsoft.AspNet.Identity.IdentityResult" /> object.

```
public override string ToString()
```

1.2.35 LockoutOptions Class

Summary

Options for configuring user lockout.

Inheritance Hierarchy

- `System.Object`
- *`Microsoft.AspNet.Identity.LockoutOptions`*

Syntax

```
public class LockoutOptions
```

GitHub

View on GitHub

class Microsoft.AspNet.Identity.**LockoutOptions**

Properties

AllowedForNewUsers()

Return type System.Boolean

```
public bool AllowedForNewUsers { get; set; }
```

DefaultLockoutTimeSpan()

Gets or sets the System.TimeSpan a user is locked out for when a lockout occurs.

Return type System.TimeSpan

```
public TimeSpan DefaultLockoutTimeSpan { get; set; }
```

MaxFailedAccessAttempts()

Gets or sets the number of failed access attempts allowed before a user is locked out, assuming lock out is enabled.

Return type System.Int32

```
public int MaxFailedAccessAttempts { get; set; }
```

1.2.36 PasswordHasher Class

Summary

Implements the standard Identity password hashing.

Inheritance Hierarchy

- System.Object
- *Microsoft.AspNet.Identity.PasswordHasher*

Syntax

```
public class PasswordHasher : IPasswordHasher
```

GitHub

View on GitHub

class Microsoft.AspNet.Identity.**PasswordHasher**

Constructors

Creates a new instance of PasswordHasher<TUser>.

PasswordHasher(*IOptions<Microsoft.AspNet.Identity.PasswordHasherOptions>*)

```
public PasswordHasher(IOptions<PasswordHasherOptions> optionsAccessor = null)
```

Methods

Returns a hashed representation of the supplied `password` for the specified `user`.

HashPassword<TUser>(*TUser*, *System.String*)

Arguments

- **user** (*{TUser}*) – The user whose password is to be hashed.
- **password** (*System.String*) – The password to hash.

Return type System.String

Returns A hashed representation of the supplied <paramref name="password" /> for the specified <paramref name="user" />.

```
public virtual string HashPassword<TUser>(TUser user, string password)where TUser : class
```

VerifyHashedPassword<TUser>(*TUser*, *System.String*, *System.String*)

Returns a Microsoft.AspNet.Identity.PasswordVerificationResult indicating the result of a password hash comparison.

Arguments

- **user** (*{TUser}*) – The user whose password should be verified.
- **hashedPassword** (*System.String*) – The hash value for a user's stored password.
- **providedPassword** (*System.String*) – The password supplied for comparison.

Return type Microsoft.AspNet.Identity.PasswordVerificationResult

Returns A <see cref="T:Microsoft.AspNet.Identity.PasswordVerificationResult" /> indicating the result of a password hash comparison.

```
public virtual PasswordVerificationResult VerifyHashedPassword<TUser>(TUser user, string has
```

1.2.37 PasswordHasherCompatibilityMode Enum

Summary

Specifies the format used for hashing passwords.

Syntax

```
public enum PasswordHasherCompatibilityMode
```

GitHub

View on GitHub

enum `Microsoft.AspNet.Identity.`**`PasswordHasherCompatibilityMode`**

Fields

Indicates hashing passwords in a way that is compatible with ASP.NET Identity versions 1 and 2.

```
IdentityV2 = 0
```

IdentityV2()

IdentityV3()

Indicates hashing passwords in a way that is compatible with ASP.NET Identity version 3.

```
IdentityV3 = 1
```

1.2.38 PasswordHasherOptions Class

Summary

Specifies options for password hashing.

Inheritance Hierarchy

- System.Object
- *Microsoft.AspNet.Identity.PasswordHasherOptions*

Syntax

```
public class PasswordHasherOptions
```

GitHub

View on GitHub

class Microsoft.AspNet.Identity.**PasswordHasherOptions**

Properties

Gets or sets the compatibility mode used when hashing passwords.

CompatibilityMode()

Return type
Microsoft.AspNet.Identity.PasswordHasherCompatibilityMode

```
public PasswordHasherCompatibilityMode CompatibilityMode { get; set; }
```

IterationCount()

Gets or sets the number of iterations used when hashing passwords using PBKDF2.

Return type System.Int32

```
public int IterationCount { get; set; }
```

1.2.39 PasswordOptions Class

Summary

Specifies options for password requirements.

Inheritance Hierarchy

- System.Object
- *Microsoft.AspNet.Identity.PasswordOptions*

Syntax

```
public class PasswordOptions
```

GitHub

View on GitHub

class Microsoft.AspNet.Identity.**PasswordOptions**

Properties

Gets or sets a flag indicating if passwords must contain a digit.

RequireDigit()

Return type System.Boolean

```
public bool RequireDigit { get; set; }
```

RequireLowercase()

Gets or sets a flag indicating if passwords must contain a lower case ASCII character.

Return type System.Boolean

```
public bool RequireLowercase { get; set; }
```

RequireNonLetterOrDigit()

Gets or sets a flag indicating if passwords must contain a digit or other non-alphabetical character.

Return type System.Boolean

```
public bool RequireNonLetterOrDigit { get; set; }
```

RequireUppercase()

Gets or sets a flag indicating if passwords must contain a upper case ASCII character.

Return type System.Boolean

```
public bool RequireUppercase { get; set; }
```

RequiredLength()

Gets or sets the minimum length a password must be.

Return type System.Int32

```
public int RequiredLength { get; set; }
```

1.2.40 PasswordValidator Class

Summary

Provides the default password policy for Identity.

Inheritance Hierarchy

- `System.Object`
- *`Microsoft.AspNet.Identity.PasswordValidator`*

Syntax

```
public class PasswordValidator : IPasswordValidator
```

GitHub

View on GitHub

class `Microsoft.AspNet.Identity.`**`PasswordValidator`**

Constructors

Constructions a new instance of Microsoft.AspNet.Identity.PasswordValidator.

`PasswordValidator`(*Microsoft.AspNet.Identity.IdentityErrorDescriber*) **Arguments**

- **`errors`** (*Microsoft.AspNet.Identity.IdentityErrorDescriber*) – The Microsoft.AspNet.Identity.IdentityErrorDescriber to retrieve error text from.

```
public PasswordValidator(IdentityErrorDescriber errors = null)
```

Properties

Gets the Microsoft.AspNet.Identity.IdentityErrorDescriber used to supply error text.

`Describer`() **Return type**
Microsoft.AspNet.Identity.IdentityErrorDescriber

```
public IdentityErrorDescriber Describer { get; }
```

Methods

Returns a flag indicting whether the supplied character is a digit.

`IsDigit`(*System.Char*) **Arguments**

- **`c`** (*System.Char*) – The character to check if it is a digit.

Return type System.Boolean

Returns True if the character is a digit, otherwise false.

```
public virtual bool IsDigit(char c)
```

IsLetterOrDigit (*System.Char*)
Returns a flag indicting whether the supplied character is an ASCII letter or digit.

Arguments

- **c** (*System.Char*) – The character to check if it is an ASCII letter or digit.

Return type System.Boolean

Returns True if the character is an ASCII letter or digit, otherwise false.

```
public virtual bool IsLetterOrDigit(char c)
```

IsLower (*System.Char*)
Returns a flag indicting whether the supplied character is a lower case ASCII letter.

Arguments

- **c** (*System.Char*) – The character to check if it is a lower case ASCII letter.

Return type System.Boolean

Returns True if the character is a lower case ASCII letter, otherwise false.

```
public virtual bool IsLower(char c)
```

IsUpper (*System.Char*)
Returns a flag indicting whether the supplied character is an upper case ASCII letter.

Arguments

- **c** (*System.Char*) – The character to check if it is an upper case ASCII letter.

Return type System.Boolean

Returns True if the character is an upper case ASCII letter, otherwise false.

```
public virtual bool IsUpper(char c)
```

ValidateAsync<TUser> (*Microsoft.AspNet.Identity.UserManager<TUser>*, *TUser*, *System.String*)
Validates a password as an asynchronous operation.

Arguments

- **manager** (*Microsoft.AspNet.Identity.UserManager{{TUser}}*) – The Microsoft.AspNet.Identity.UserManager`1 to retrieve the `user` properties from.
- **user** (*{TUser}*) – The user whose password should be validated.
- **password** (*System.String*) – The password supplied for validation

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

Returns The task object representing the asynchronous operation.

```
public virtual Task<IdentityResult> ValidateAsync<TUser>(UserManager<TUser> manager, TUser u
```

1.2.41 PasswordVerificationResult Enum

Summary

Specifies the results for password verification.

Syntax

```
public enum PasswordVerificationResult
```

GitHub

View on GitHub

enum `Microsoft.AspNet.Identity.`**`PasswordVerificationResult`**

Fields

Indicates password verification failed.

```
Failed = 0
```

`Success``()`
Indicates password verification was successful.

```
Success = 1
```

`SuccessRehashNeeded``()`
Indicates password verification was successful however the password was encoded using a deprecated algorithm and should be rehashed and updated.

```
SuccessRehashNeeded = 2
```

1.2.42 PhoneNumberTokenProvider Class

Summary

Represents a token provider that generates tokens from a user's security stamp and sends them to the user via their phone number.

Inheritance Hierarchy

- `System.Object`
- *`Microsoft.AspNet.Identity.TotpSecurityStampBasedTokenProvider`*
- *`Microsoft.AspNet.Identity.PhoneNumberTokenProvider`*

Syntax

```
public class PhoneNumberTokenProvider : TotpSecurityStampBasedTokenProvider, IUserTokenProvider
```

GitHub

View on GitHub

class `Microsoft.AspNet.Identity.`**`PhoneNumberTokenProvider`**

Constructors

Creates a new instance of PhoneNumberTokenProvider<TUser> with the specified `options`.

`PhoneNumberTokenProvider`(*IOptions<Microsoft.AspNet.Identity.PhoneNumberTokenProviderOptions>*) **Arguments**

- **`options`** (*IOptions{Microsoft.AspNet.Identity.PhoneNumberTokenProviderOptions}*) – The options to use for the created instance of a PhoneNumberTokenProvider<TUser>.

```
public PhoneNumberTokenProvider(IOptions<PhoneNumberTokenProviderOptions> options)
```

Methods

Returns a flag indicating whether the token provider can generate a token suitable for two factor authentication token for the specified `ref`.

`CanGenerateTwoFactorTokenAsync<TUser>`(*Microsoft.AspNet.Identity.UserManager<TUser>*, *TUser*) **Arguments**

- **`manager`** (*Microsoft.AspNet.Identity.UserManager{TUser}*) – The Microsoft.AspNet.Identity.UserManager`1 that can be used to retrieve user properties.
- **`user`** (*TUser*) – The user a token could be generated for.

Return type System.Threading.Tasks.Task{System.Boolean}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the a flag indicating if a two factor token could be generated by this provider for the specified <paramref name="user" /> and <paramref name="purpose" />. The task will return true if a two factor authentication token could be generated as the user has a telephone number, otherwise false.

```
public override Task<bool> CanGenerateTwoFactorTokenAsync<TUser>(UserManager<TUser> manager,
```

GetUserModifierAsync<TUser> (*System.String*, *Microsoft.AspNet.Identity.UserManager<TUser>*, *TUser*)

Returns a constant, provider and user unique modifier used for entropy in generated tokens from user information, as an asynchronous operation.

Arguments

- **purpose** (*System.String*) – The purpose the token will be generated for.
- **manager** (*Microsoft.AspNet.Identity.UserManager{TUser}*) – The Microsoft.AspNet.Identity.UserManager`1 that can be used to retrieve user properties.
- **user** (*{TUser}*) – The user a token should be generated for.

Return type System.Threading.Tasks.Task{System.String}

Returns

The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing a constant modifier for the specified

<paramref name="user" /> and <paramref name="purpose" />.

```
public override Task<string> GetUserModifierAsync<TUser>(string purpose, UserManager<TUser>
```

Properties

Gets the name for this instance of PhoneNumberTokenProvider<TUser>.

Name() **Return type**

System.String

```
public override string Name { get; }
```

Options()

Gets the options for this instance of PhoneNumberTokenProvider<TUser>.

Return type Microsoft.AspNet.Identity.PhoneNumberTokenProviderOptions

```
public PhoneNumberTokenProviderOptions Options { get; }
```

1.2.43 PhoneNumberTokenProviderOptions Class

Summary

Encapsulations options for a PhoneNumberTokenProvider<TUser>.

Inheritance Hierarchy

- `System.Object`
- *`Microsoft.AspNet.Identity.PhoneNumberTokenProviderOptions`*

Syntax

```
public class PhoneNumberTokenProviderOptions
```

GitHub

View on GitHub

class `Microsoft.AspNet.Identity.`**`PhoneNumberTokenProviderOptions`**

Properties

Gets or sets the name for the PhoneNumberTokenProvider<TUser>.

`Name``()` **Return type**

System.String

```
public string Name { get; set; }
```

1.2.44 RoleManager<TRole> Class

Summary

Provides the APIs for managing roles in a persistence store.

Inheritance Hierarchy

- `System.Object`
- *`Microsoft.AspNet.Identity.RoleManager<TRole>`*

Syntax

```
public class RoleManager<TRole> : IDisposable where TRole : class
```

GitHub

View on GitHub

class Microsoft.AspNet.Identity.**RoleManager<TRole>**

Constructors

Constructs a new instance of Microsoft.AspNet.Identity.RoleManager‘1.

RoleManager(*Microsoft.AspNet.Identity.IRoleStore<TRole>*, *System.Collections.Generic.IEnumerable<Microsoft.AspNet.Identity.IRoleValidator>*, *Microsoft.AspNet.Identity.ILookupNormalizer*, *Microsoft.AspNet.Identity.IdentityErrorDescriber*, *ILogger<Microsoft.AspNet.Identity.RoleManager<TRole>>*, *IHttpContextAccessor*)

Arguments

- **store** (*Microsoft.AspNet.Identity.IRoleStore{TRole}*) – The persistence store the manager will operate over.
- **roleValidators** (*System.Collections.Generic.IEnumerable{Microsoft.AspNet.Identity.IRoleValidator}*) – A collection of validators for roles.
- **keyNormalizer** (*Microsoft.AspNet.Identity.ILookupNormalizer*) – The normalizer to use when normalizing role names to keys.
- **errors** (*Microsoft.AspNet.Identity.IdentityErrorDescriber*) – The Microsoft.AspNet.Identity.IdentityErrorDescriber used to provider error messages.
- **logger** (*ILogger{Microsoft.AspNet.Identity.RoleManager‘1}*) – The logger used to log messages, warnings and errors.
- **contextAccessor** (*IHttpContextAccessor*) – The accessor used to access the HttpContext.

```
public RoleManager(IRoleStore<TRole> store, IEnumerable<IRoleValidator> roleValidators, ILoc
```

Methods

Adds a claim to a role, as an asynchronous operation.

AddClaimAsync(*TRole*, *System.Security.Claims.Claim*)

Arguments

- **role** (*{TRole}*) – The role to add the claim to.
- **claim** (*System.Security.Claims.Claim*) – The claim to add.

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

Returns The <see cref=”T:System.Threading.Tasks.Task” /> that represents the asynchronous operation, containing the <see cref=”T:Microsoft.AspNet.Identity.IdentityResult” /> of the operation.

```
public virtual Task<IdentityResult> AddClaimAsync(TRole role, Claim claim)
```

CreateAsync(*TRole*)

Creates the specified `role` in the persistence store, as an asynchronous operation.

Arguments

- **role** (*{TRole}*) – The role to create.

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation.

```
public virtual Task<IdentityResult> CreateAsync(TRole role)
```

DeleteAsync(*TRole*)

Deletes the specified `role`, as an asynchronous operation.

Arguments

- **role** (*{TRole}*) – The role to delete.

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the <see cref="T:Microsoft.AspNet.Identity.IdentityResult" /> for the delete.

```
public virtual Task<IdentityResult> DeleteAsync(TRole role)
```

Dispose()

Releases all resources used by the role manager.

```
public void Dispose()
```

Dispose(*System.Boolean*)

Releases the unmanaged resources used by the role manager and optionally releases the managed resources.

Arguments

- **disposing** (*System.Boolean*) – true to release both managed and unmanaged resources; false to release only unmanaged resources.

```
protected virtual void Dispose(bool disposing)
```

FindByIdAsync(*System.String*)

Finds the role associated with the specified `roleId` if any, as an asynchronous operation.

Arguments

- **roleId** (*System.String*) – The role ID whose role should be returned.

Return type System.Threading.Tasks.Task{{TRole}}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the role associated with the specified <paramref name="roleId" />

```
public virtual Task<TRole> FindByIdAsync(string roleId)
```

FindByNameAsync(*System.String*)

Finds the role associated with the specified `roleName` if any, as an asynchronous operation.

Arguments

- **roleName** (*System.String*) – The name of the role to be returned.

Return type System.Threading.Tasks.Task{{TRole}}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the role associated with the specified <paramref name="roleName" />

```
public virtual Task<TRole> FindByNameAsync(string roleName)
```

GetClaimsAsync(*TRole*)

Gets a list of claims associated with the specified `role`, as an asynchronous operation.

Arguments

- **role** (*{TRole}*) – The role whose claims should be returned.

Return type System.Threading.Tasks.Task{System.Collections.Generic.IList{System.Security.Claims.Claim}}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the list of <see cref="T:System.Security.Claims.Claim" />s associated with the specified <paramref name="role" />.

```
public virtual Task<IList<Claim>> GetClaimsAsync(TRole role)
```

GetRoleIdAsync(*TRole*)

Gets the ID of the specified `role`, as an asynchronous operation.

Arguments

- **role** (*{TRole}*) – The role whose ID should be retrieved.

Return type System.Threading.Tasks.Task{System.String}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the ID of the specified <paramref name="role" />.

```
public virtual Task<string> GetRoleIdAsync(TRole role)
```

GetRoleNameAsync(*TRole*)

Gets the name of the specified `role`, as an asynchronous operation.

Arguments

- **role** (*{TRole}*) – The role whose name should be retrieved.

Return type System.Threading.Tasks.Task{System.String}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the name of the specified <paramref name="role" />.

```
public virtual Task<string> GetRoleNameAsync(TRole role)
```

NormalizeKey(*System.String*)

Gets a normalized representation of the specified `key`.

Arguments

- **key** (*System.String*) – The value to normalize.

Return type System.String

Returns A normalized representation of the specified <paramref name="key" />.

```
public virtual string NormalizeKey(string key)
```

RemoveClaimAsync(*TRole*, *System.Security.Claims.Claim*)

Removes a claim from a role, as an asynchronous operation.

Arguments

- **role** (*{TRole}*) – The role to remove the claim from.
- **claim** (*System.Security.Claims.Claim*) – The claim to add.

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the <see cref="T:Microsoft.AspNet.Identity.IdentityResult" /> of the operation.

```
public virtual Task<IdentityResult> RemoveClaimAsync(TRole role, Claim claim)
```

RoleExistsAsync(*System.String*)

Gets a flag indicating whether the specified `roleName` exists, as an asynchronous operation.

Arguments

- **roleName** (*System.String*) – The role name whose existence should be checked.

Return type System.Threading.Tasks.Task{System.Boolean}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing true if the role name exists, otherwise false.

```
public virtual Task<bool> RoleExistsAsync(string roleName)
```

SetRoleNameAsync(*TRole*, *System.String*)

Sets the name of the specified `role`, as an asynchronous operation.

Arguments

- **role** (*{TRole}*) – The role whose name should be set.
- **name** (*System.String*) – The name to set.

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the <see cref="T:Microsoft.AspNet.Identity.IdentityResult" /> of the operation.

```
public virtual Task<IdentityResult> SetRoleNameAsync(TRole role, string name)
```

UpdateAsync(*TRole*)

Updates the specified `role`, as an asynchronous operation.

Arguments

- **role** (*{TRole}*) – The role to updated.

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the <see cref="T:Microsoft.AspNet.Identity.IdentityResult" /> for the update.

```
public virtual Task<IdentityResult> UpdateAsync(TRole role)
```

UpdateNormalizedRoleNameAsync(*TRole*)

Updates the normalized name for the specified `role`, as an asynchronous operation.

Arguments

- **role** (*{TRole}*) – The role whose normalized name needs to be updated.

Return type System.Threading.Tasks.Task

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation.

```
public virtual Task UpdateNormalizedRoleNameAsync(TRole role)
```

Properties

Gets the ILogger used to log messages from the manager.

Logger()

Return type

ILogger

```
protected virtual ILogger Logger { get; set; }
```

Roles()

Gets an IQueryable collection of Roles if the persistence store is an IQueryableRoleStore, otherwise throws a System.NotSupportedException.

Return type System.Linq.IQueryable{{TRole}}

```
public virtual IQueryable<TRole> Roles { get; }
```

Store()

Gets the persistence store this instance operates over.

Return type Microsoft.AspNet.Identity.IRoleStore{{TRole}}

```
protected IRoleStore<TRole> Store { get; }
```

SupportsQueryableRoles()

Gets a flag indicating whether the underlying persistence store supports returning an System.Linq.IQueryable collection of roles.

Return type System.Boolean

```
public virtual bool SupportsQueryableRoles { get; }
```

SupportsRoleClaims()

Gets a flag indicating whether the underlying persistence store supports :dn:ref:‘System.Security.Claims.Claim‘s for roles.

Return type System.Boolean

```
public virtual bool SupportsRoleClaims { get; }
```

1.2.45 RoleValidator Class

Summary

Provides the default validation of roles.

Inheritance Hierarchy

- `System.Object`
- *`Microsoft.AspNet.Identity.RoleValidator`*

Syntax

```
public class RoleValidator : IRoleValidator
```

GitHub

View on GitHub

class `Microsoft.AspNet.Identity.`**`RoleValidator`**

Constructors

Creates a new instance of RoleValidator<TRole>/

`RoleValidator` (*Microsoft.AspNet.Identity.IdentityErrorDescriber*) **Arguments**

- **`errors`** (*Microsoft.AspNet.Identity.IdentityErrorDescriber*) – The Microsoft.AspNet.Identity.IdentityErrorDescriber used to provider error messages.

```
public RoleValidator(IdentityErrorDescriber errors = null)
```

Methods

Validates a role as an asynchronous operation.

`ValidateAsync<TRole>` (*Microsoft.AspNet.Identity.RoleManager<TRole>*, *TRole*) **Arguments**

- **`manager`** (*Microsoft.AspNet.Identity.RoleManager{{TRole}}*) – The Microsoft.AspNet.Identity.RoleManager‘1 managing the role store.
- **`role`** (*{TRole}*) – The role to validate.

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

Returns A <see cref=”T:System.Threading.Tasks.Task‘1” /> that represents the <see cref=”T:Microsoft.AspNet.Identity.IdentityResult” /> of the asynchronous validation.

```
public virtual Task<IdentityResult> ValidateAsync<TRole>(RoleManager<TRole> manager, TRole r
```

1.2.46 SecurityStampValidator Class

Summary

Static helper class used to configure a CookieAuthenticationNotifications to validate a cookie against a user's security stamp.

Inheritance Hierarchy

- `System.Object`
- *`Microsoft.AspNet.Identity.SecurityStampValidator`*

Syntax

```
public class SecurityStampValidator
```

GitHub

View on GitHub

class `Microsoft.AspNet.Identity.`**`SecurityStampValidator`**

Methods

Validates a principal against a user's stored security stamp. the identity.

`ValidatePrincipalAsync`(*CookieValidatePrincipalContext*)

Arguments

- **`context`** (*CookieValidatePrincipalContext*) – The context containing the :dn:ref:‘System.Security.Claims.ClaimsPrincipal‘and AuthenticationProperties to validate.

Return type System.Threading.Tasks.Task

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous validation operation.

```
public static Task ValidatePrincipalAsync(CookieValidatePrincipalContext context)
```

1.2.47 SecurityStampValidator<TUser> Class

Summary

Provides default implementation of validation functions for security stamps.

Inheritance Hierarchy

- `System.Object`
- *`Microsoft.AspNet.Identity.SecurityStampValidator<TUser>`*

Syntax

```
public class SecurityStampValidator<TUser> : ISecurityStampValidator where TUser : class
```

GitHub

View on GitHub

class `Microsoft.AspNet.Identity.`**`SecurityStampValidator<TUser>`**

Methods

Validates a security stamp of an identity as an asynchronous operation, and rebuilds the identity if the validation succeeds, otherwise rejects the identity.

`ValidateAsync`(*CookieValidatePrincipalContext*)

Arguments

- **`context`** (*CookieValidatePrincipalContext*) – The context containing the :dn:ref:‘System.Security.Claims.ClaimsPrincipal‘and AuthenticationProperties to validate.

Return type System.Threading.Tasks.Task

Returns The <see cref=”T:System.Threading.Tasks.Task” /> that represents the asynchronous validation operation.

```
public virtual Task ValidateAsync(CookieValidatePrincipalContext context)
```

1.2.48 SignInManager<TUser> Class

Summary

Provides the APIs for user sign in.

Inheritance Hierarchy

- `System.Object`
- *`Microsoft.AspNet.Identity.SignInManager<TUser>`*

Syntax

```
public class SignInManager<TUser> where TUser : class
```

GitHub

View on GitHub

class `Microsoft.AspNet.Identity.`**`SignInManager<TUser>`**

Constructors

Creates a new instance of Microsoft.AspNet.Identity.SignInManager`1.

`SignInManager`(*Microsoft.AspNet.Identity.UserManager<TUser>*, *IHttpContextAccessor*, *Microsoft.AspNet.Identity.IUserClaimsPrincipalFactory<TUser>*, *IOptions<Microsoft.AspNet.Identity.IdentityOptions>*, *ILogger<Microsoft.AspNet.Identity.SignInManager<TUser>>*) **Arguments**

- **`userManager`** (*Microsoft.AspNet.Identity.UserManager{{TUser}}*) – An instance of `Microsoft.AspNet.Identity.SignInManager`1.UserManager` used to retrieve users from and persist users.
- **`contextAccessor`** (*IHttpContextAccessor*) – The accessor used to access the HttpContext.
- **`claimsFactory`** (*Microsoft.AspNet.Identity.IUserClaimsPrincipalFactory{{TUser}}*) – The factory to use to create claims principals for a user.
- **`optionsAccessor`** (*IOptions{Microsoft.AspNet.Identity.IdentityOptions}*) – The accessor used to access the Microsoft.AspNet.Identity.IdentityOptions.

- **logger** (*ILogger{Microsoft.AspNet.Identity.SignInManager`1}*) – The logger used to log messages, warnings and errors.

```
public SignInManager(UserManager<TUser> userManager, IHttpContextAccessor contextAccessor, I
```

Methods

Returns a flag indicating whether the specified user can sign in.

CanSignInAsync(*TUser*) **Arguments**

- **user** (*{TUser}*) – The user whose sign-in status should be returned.

Return type System.Threading.Tasks.Task{System.Boolean}

Returns The task object representing the asynchronous operation, containing a flag that is true if the specified user can sign-in, otherwise false.

```
public virtual Task<bool> CanSignInAsync(TUser user)
```

ConfigureExternalAuthenticationProperties(*System.String*, *System.String*, *System.String*)

Configures the redirect URL and user identifier for the specified external login `provider`.

Arguments

- **provider** (*System.String*) – The provider to configure.
- **redirectUrl** (*System.String*) – The external login URL users should be redirected to during the login glow.
- **userId** (*System.String*) – The current user's identifier, which will be used to provide CSRF protection.

Return type AuthenticationProperties

Returns A configured <see cref="!:AuthenticationProperties" />.

```
public virtual AuthenticationProperties ConfigureExternalAuthenticationProperties(string pro
```

CreateUserPrincipalAsync(*TUser*)

Creates a System.Security.Claims.ClaimsPrincipal for the specified `user`, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user to create a System.Security.Claims.ClaimsPrincipal for.

Return type System.Threading.Tasks.Task{System.Security.Claims.ClaimsPrincipal}

Returns The task object representing the asynchronous operation, containing the ClaimsPrincipal for the specified user.

```
public virtual Task<ClaimsPrincipal> CreateUserPrincipalAsync(TUser user)
```

ExternalLoginSignInAsync(*System.String*, *System.String*, *System.Boolean*)

Signs in a user via a previously registered third party login, as an asynchronous operation.

Arguments

- **loginProvider** (*System.String*) – The login provider to use.
- **providerKey** (*System.String*) – The unique provider identifier for the user.

- **isPersistent** (*System.Boolean*) – Flag indicating whether the sign-in cookie should persist after the browser is closed.

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.SignInResult}

Returns The task object representing the asynchronous operation containing the <see name="SignInResult" /> for the sign-in attempt.

```
public virtual Task<SignInResult> ExternalLoginSignInAsync(string loginProvider, string prov
```

ForgetTwoFactorClientAsync()

Clears the "Remember this browser flag" from the current browser, as an asynchronous operation.

Return type System.Threading.Tasks.Task

Returns The task object representing the asynchronous operation.

```
public virtual Task ForgetTwoFactorClientAsync()
```

GetExternalAuthenticationSchemes()

Gets a collection of AuthenticationDescriptions for the known external login providers.

Return type System.Collections.Generic.IEnumerable{AuthenticationDescription}

Returns A collection of <see cref="!:AuthenticationDescription" />s for the known external login providers.

```
public virtual IEnumerable<AuthenticationDescription> GetExternalAuthenticationSchemes()
```

GetExternalLoginInfoAsync(*System.String*)

Gets the external login information for the current login, as an asynchronous operation.

Arguments

- **expectedXsrf** (*System.String*) – Flag indication whether a Cross Site Request Forgery token was expected in the current request.

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.ExternalLoginInfo}

Returns The task object representing the asynchronous operation containing the <see name="ExternalLoginInfo" /> for the sign-in attempt.

```
public virtual Task<ExternalLoginInfo> GetExternalLoginInfoAsync(string expectedXsrf = null)
```

GetTwoFactorAuthenticationUserAsync()

Gets the `TUser` for the current two factor authentication login, as an asynchronous operation.

Return type System.Threading.Tasks.Task{{TUser}}

Returns The task object representing the asynchronous operation containing the <typeparamref name="TUser" /> for the sign-in attempt.

```
public virtual Task<TUser> GetTwoFactorAuthenticationUserAsync()
```

IsTwoFactorClientRememberedAsync(*TUser*)

Returns a flag indicating if the current client browser has been remembered by two factor authentication for the user attempting to login, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user attempting to login.

Return type System.Threading.Tasks.Task{System.Boolean}

Returns The task object representing the asynchronous operation containing true if the browser has been remembered for the current user.

```
public virtual Task<bool> IsTwoFactorClientRememberedAsync(TUser user)
```

PasswordSignInAsync (*System.String*, *System.String*, *System.Boolean*, *System.Boolean*)
Attempts to sign in the specified `userName` and `password` combination as an asynchronous operation.

Arguments

- **userName** (*System.String*) – The user name to sign in.
- **password** (*System.String*) – The password to attempt to sign in with.
- **isPersistent** (*System.Boolean*) – Flag indicating whether the sign-in cookie should persist after the browser is closed.

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.SignInResult}

Returns The task object representing the asynchronous operation containing the <see name="SignInResult" /> for the sign-in attempt.

```
public virtual Task<SignInResult> PasswordSignInAsync(string userName, string password, bool
```

PasswordSignInAsync (*TUser*, *System.String*, *System.Boolean*, *System.Boolean*)
Attempts to sign in the specified `user` and `password` combination as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user to sign in.
- **password** (*System.String*) – The password to attempt to sign in with.
- **isPersistent** (*System.Boolean*) – Flag indicating whether the sign-in cookie should persist after the browser is closed.
- **lockoutOnFailure** (*System.Boolean*) – Flag indicating if the user account should be locked if the sign in fails.

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.SignInResult}

Returns The task object representing the asynchronous operation containing the <see name="SignInResult" /> for the sign-in attempt.

```
public virtual Task<SignInResult> PasswordSignInAsync(TUser user, string password, bool isPe
```

RefreshSignInAsync (*TUser*)
Regenerates the user's application cookie, whilst preserving the existing AuthenticationProperties like rememberMe, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user whose sign-in cookie should be refreshed.

Return type System.Threading.Tasks.Task

Returns The task object representing the asynchronous operation.

```
public virtual Task RefreshSignInAsync(TUser user)
```

RememberTwoFactorClientAsync (*TUser*)
Sets a flag on the browser to indicate the user has selected "Remember this browser" for two factor authentication purposes, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user who choose "remember this browser".

Return type System.Threading.Tasks.Task

Returns The task object representing the asynchronous operation.

```
public virtual Task RememberTwoFactorClientAsync(TUser user)
```

SignInAsync(*TUser*, *AuthenticationProperties*, *System.String*)
Signs in the specified `user`.

Arguments

- **user** (*{TUser}*) – The user to sign-in.
- **authenticationProperties** (*AuthenticationProperties*) – Properties applied to the login and authentication cookie.
- **authenticationMethod** (*System.String*) – Name of the method used to authenticate the user.

Return type System.Threading.Tasks.Task

Returns The task object representing the asynchronous operation.

```
public virtual Task SignInAsync(TUser user, AuthenticationProperties authenticationPropertie
```

SignInAsync(*TUser*, *System.Boolean*, *System.String*)
Signs in the specified `user`.

Arguments

- **user** (*{TUser}*) – The user to sign-in.
- **isPersistent** (*System.Boolean*) – Flag indicating whether the sign-in cookie should persist after the browser is closed.
- **authenticationMethod** (*System.String*) – Name of the method used to authenticate the user.

Return type System.Threading.Tasks.Task

Returns The task object representing the asynchronous operation.

```
public virtual Task SignInAsync(TUser user, bool isPersistent, string authenticationMethod =
```

SignOutAsync()
Signs the current user out of the application.

Return type System.Threading.Tasks.Task

```
public virtual Task SignOutAsync()
```

TwoFactorSignInAsync(*System.String*, *System.String*, *System.Boolean*, *System.Boolean*)
Validates the two faction sign in code and creates and signs in the user, as an asynchronous operation.

Arguments

- **provider** (*System.String*) – The two factor authentication provider to validate the code against.
- **code** (*System.String*) – The two factor authentication code to validate.
- **isPersistent** (*System.Boolean*) – Flag indicating whether the sign-in cookie should persist after the browser is closed.

- **rememberClient** (*System.Boolean*) – Flag indicating whether the current browser should be remember, suppressing all further two factor authentication prompts.

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.SignInResult}

Returns The task object representing the asynchronous operation containing the <see name="SignInResult" /> for the sign-in attempt.

```
public virtual Task<SignInResult> TwoFactorSignInAsync(string provider, string code, bool is
```

ValidateSecurityStampAsync(*System.Security.Claims.ClaimsPrincipal*, *System.String*)

Validates the security stamp for the specified `principal` against the persisted stamp for the `userId`, as an asynchronous operation.

Arguments

- **principal** (*System.Security.Claims.ClaimsPrincipal*) – The principal whose stamp should be validated.
- **userId** (*System.String*) – The ID for the user.

Return type System.Threading.Tasks.Task{{TUser}}

Returns The task object representing the asynchronous operation. The task will contain the <typeparamref name="TUser" /> if the stamp matches the persisted value, otherwise it will return false.

```
public virtual Task<TUser> ValidateSecurityStampAsync(ClaimsPrincipal principal, string user
```

Properties

Gets the ILogger used to log messages from the manager.

Logger() **Return type**

ILogger

```
protected virtual ILogger Logger { get; set; }
```

1.2.49 SignInOptions Class

Summary

Options for configuring sign in..

Inheritance Hierarchy

- `System.Object`

- *Microsoft.AspNet.Identity.SignInOptions*

Syntax

```
public class SignInOptions
```

GitHub

View on GitHub

class Microsoft.AspNet.Identity.**SignInOptions**

Properties

Gets or sets a flag indicating whether a confirmed email address is required to sign in.

RequireConfirmedEmail()

Return type System.Boolean

```
public bool RequireConfirmedEmail { get; set; }
```

RequireConfirmedPhoneNumber()

Gets or sets a flag indicating whether a confirmed telephone number is required to sign in.

Return type System.Boolean

```
public bool RequireConfirmedPhoneNumber { get; set; }
```

1.2.50 SignInResult Class

Summary

Represents the result of a sign-in operation.

Inheritance Hierarchy

- System.Object
- *Microsoft.AspNet.Identity.SignInResult*

Syntax

```
public class SignInResult
```

GitHub

View on GitHub

class Microsoft.AspNet.Identity.**SignInResult**

Properties

Returns a Microsoft.AspNet.Identity.SignInResult that represents a failed sign-in.

Failed()

Return type Microsoft.AspNet.Identity.SignInResult

Returns A <see cref="T:Microsoft.AspNet.Identity.SignInResult" /> that represents a failed sign-in.

```
public static SignInResult Failed { get; }
```

IsLockedOut()

Returns a flag indication whether the user attempting to sign-in is locked out.

Return type System.Boolean

```
public bool IsLockedOut { get; protected set; }
```

IsNotAllowed()

Returns a flag indication whether the user attempting to sign-in is not allowed to sign-in.

Return type System.Boolean

```
public bool IsNotAllowed { get; protected set; }
```

LockedOut()

Returns a Microsoft.AspNet.Identity.SignInResult that represents a sign-in attempt that failed because the user was logged out.

Return type Microsoft.AspNet.Identity.SignInResult

Returns A <see cref="T:Microsoft.AspNet.Identity.SignInResult" /> that represents sign-in attempt that failed due to the user being locked out.

```
public static SignInResult LockedOut { get; }
```

NotAllowed()

Returns a Microsoft.AspNet.Identity.SignInResult that represents a sign-in attempt that failed because the user is not allowed to sign-in.

Return type Microsoft.AspNet.Identity.SignInResult

Returns A <see cref="T:Microsoft.AspNet.Identity.SignInResult" /> that represents sign-in attempt that failed due to the user is not allowed to sign-in.

```
public static SignInResult NotAllowed { get; }
```

RequiresTwoFactor()

Returns a flag indication whether the user attempting to sign-in requires two factor authentication.

Return type System.Boolean

```
public bool RequiresTwoFactor { get; protected set; }
```

Succeeded()
Returns a flag indication whether the sign-in was successful.

Return type System.Boolean

```
public bool Succeeded { get; protected set; }
```

Success()
Returns a Microsoft.AspNet.Identity.SignInResult that represents a successful sign-in.

Return type Microsoft.AspNet.Identity.SignInResult

Returns A <see cref="T:Microsoft.AspNet.Identity.SignInResult" /> that represents a successful sign-in.

```
public static SignInResult Success { get; }
```

TwoFactorRequired()
Returns a Microsoft.AspNet.Identity.SignInResult that represents a sign-in attempt that needs two-factor authentication.

Return type Microsoft.AspNet.Identity.SignInResult

Returns A <see cref="T:Microsoft.AspNet.Identity.SignInResult" /> that represents sign-in attempt that needs two-factor authentication.

```
public static SignInResult TwoFactorRequired { get; }
```

Methods

Converts the value of the current Microsoft.AspNet.Identity.SignInResult object to its equivalent string representation.

ToString() **Return type**
System.String

Returns A string representation of value of the current <see cref="T:Microsoft.AspNet.Identity.SignInResult" /> object.

```
public override string ToString()
```

1.2.51 TotpSecurityStampBasedTokenProvider Class

Summary

Represents a token provider that generates time based codes using the user's security stamp.

Inheritance Hierarchy

- `System.Object`
- *`Microsoft.AspNet.Identity.TotpSecurityStampBasedTokenProvider`*

Syntax

```
public abstract class TotpSecurityStampBasedTokenProvider : IUserTokenProvider
```

GitHub

View on GitHub

class `Microsoft.AspNet.Identity.`**`TotpSecurityStampBasedTokenProvider`**

Methods

Returns a flag indicating whether the token provider can generate a token suitable for two factor authentication token for the specified `ref`.

`CanGenerateTwoFactorTokenAsync<TUser>` (*Microsoft.AspNet.Identity.UserManager<TUser>*, *TUser*)

Arguments

- **`manager`** (*Microsoft.AspNet.Identity.UserManager{TUser}*) – The Microsoft.AspNet.Identity.UserManager`1 that can be used to retrieve user properties.
- **`user`** (*TUser*) – The user a token could be generated for.

Return type System.Threading.Tasks.Task{System.Boolean}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the a flag indicating if a two factor token could be generated by this provider for the specified <paramref name="user" /> and <paramref name="purpose" />. The task will return true if a two factor authentication token could be generated, otherwise false.

```
public abstract Task<bool> CanGenerateTwoFactorTokenAsync<TUser>(UserManager<TUser> manager,
```

`GenerateAsync<TUser>` (*System.String*, *Microsoft.AspNet.Identity.UserManager<TUser>*, *TUser*)

Generates a token for the specified `ref` and `purpose`, as an asynchronous operation.

Arguments

- **`purpose`** (*System.String*) – The purpose the token will be used for.
- **`manager`** (*Microsoft.AspNet.Identity.UserManager{TUser}*) – The Microsoft.AspNet.Identity.UserManager`1 that can be used to retrieve user properties.
- **`user`** (*TUser*) – The user a token should be generated for.

Return type System.Threading.Tasks.Task{System.String}

Returns

The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the token for the specified

<paramref name="user" /> and <paramref name="purpose" />.

```
public virtual Task<string> GenerateAsync<TUser>(string purpose, UserManager<TUser> manager,
```

GetUserModifierAsync<TUser> (*System.String*, *Microsoft.AspNet.Identity.UserManager<TUser>*, *TUser*)

Returns a constant, provider and user unique modifier used for entropy in generated tokens from user information, as an asynchronous operation.

Arguments

- **purpose** (*System.String*) – The purpose the token will be generated for.
- **manager** (*Microsoft.AspNet.Identity.UserManager{{TUser}}*) – The Microsoft.AspNet.Identity.UserManager`1 that can be used to retrieve user properties.
- **user** (*{TUser}*) – The user a token should be generated for.

Return type System.Threading.Tasks.Task{System.String}

Returns

The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing a constant modifier for the specified

<paramref name="user" /> and <paramref name="purpose" />.

```
public virtual Task<string> GetUserModifierAsync<TUser>(string purpose, UserManager<TUser> m
```

ValidateAsync<TUser> (*System.String*, *System.String*, *Microsoft.AspNet.Identity.UserManager<TUser>*, *TUser*)

Returns a flag indicating whether the specified `token` is valid for the given

`user` and `purpose`, as an asynchronous operation.

Arguments

- **purpose** (*System.String*) – The purpose the token will be used for.
- **token** (*System.String*) – The token to validate.
- **manager** (*Microsoft.AspNet.Identity.UserManager{{TUser}}*) – The Microsoft.AspNet.Identity.UserManager`1 that can be used to retrieve user properties.
- **user** (*{TUser}*) – The user a token should be validated for.

Return type System.Threading.Tasks.Task{System.Boolean}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the a flag indicating the result of validating the <paramref name="token"> for the specified </paramref> <paramref name="user" /> and <paramref name="purpose" />. The task will return true if the token is valid, otherwise false.

```
public virtual Task<bool> ValidateAsync<TUser>(string purpose, string token, UserManager<TUs
```

Properties

Gets the name of the token provider.

Name()

Return type

System.String

```
public abstract string Name { get; }
```

1.2.52 UpperInvariantLookupNormalizer Class

Summary

Implements Microsoft.AspNet.Identity.ILookupNormalizer by converting keys to their upper cased invariant culture representation.

Inheritance Hierarchy

- `System.Object`
- *`Microsoft.AspNet.Identity.UpperInvariantLookupNormalizer`*

Syntax

```
public class UpperInvariantLookupNormalizer : ILookupNormalizer
```

GitHub

View on GitHub

class `Microsoft.AspNet.Identity.`**`UpperInvariantLookupNormalizer`**

Methods

Returns a normalized representation of the specified `key` by converting keys to their upper cased invariant culture representation.

Normalize(*System.String*)

Arguments

- **key** (*System.String*) – The key to normalize.

Return type System.String

Returns A normalized representation of the specified <paramref name="key" />.

```
public virtual string Normalize(string key)
```

1.2.53 UserClaimsPrincipalFactory<TUser, TRole> Class

Summary

Provides methods to create a claims principal for a given user.

Inheritance Hierarchy

- `System.Object`
- *`Microsoft.AspNet.Identity.UserClaimsPrincipalFactory<TUser, TRole>`*

Syntax

```
public class UserClaimsPrincipalFactory<TUser, TRole> : IUserClaimsPrincipalFactory<TUser> where TUse
```

GitHub

View on GitHub

class `Microsoft.AspNet.Identity.`**`UserClaimsPrincipalFactory<TUser, TRole>`**

Constructors

Initializes a new instance of the ClaimsIdentityFactory class.

`UserClaimsPrincipalFactory`(*Microsoft.AspNet.Identity.UserManager<TUser>*, *Microsoft.AspNet.Identity.RoleManager<TRole>*, *IOptions<Microsoft.AspNet.Identity.IdentityOptions>*)

Arguments

- **`userManager`** (*Microsoft.AspNet.Identity.UserManager{TUser}*) – The Microsoft.AspNet.Identity.UserManager‘1 to retrieve user information from.
- **`roleManager`** (*Microsoft.AspNet.Identity.RoleManager{TRole}*) – The Microsoft.AspNet.Identity.RoleManager‘1 to retrieve a user’s roles from.
- **`optionsAccessor`** (*IOptions{Microsoft.AspNet.Identity.IdentityOptions}*) – The configured Microsoft.AspNet.Identity.IdentityOptions.

```
        public UserClaimsPrincipalFactory(UserManager<TUser> userManager, RoleManager<TRole> roleMan
```

Methods

Creates a System.Security.Claims.ClaimsPrincipal from an user asynchronously.

CreateAsync(*TUser*) **Arguments**

- **user** (*{TUser}*) – The user to create a System.Security.Claims.ClaimsPrincipal from.

Return type System.Threading.Tasks.Task{System.Security.Claims.ClaimsPrincipal}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous creation operation, containing the created <see cref="T:System.Security.Claims.ClaimsPrincipal" />.

```
public virtual Task<ClaimsPrincipal> CreateAsync(TUser user)
```

Properties

Gets the Microsoft.AspNet.Identity.IdentityOptions for this factory.

Options() **Return type**

Microsoft.AspNet.Identity.IdentityOptions

```
public IdentityOptions Options { get; }
```

RoleManager()

Gets the Microsoft.AspNet.Identity.RoleManager‘1 for this factory.

Return type Microsoft.AspNet.Identity.RoleManager{{TRole}}

```
public RoleManager<TRole> RoleManager { get; }
```

UserManager()

Gets the Microsoft.AspNet.Identity.UserManager‘1 for this factory.

Return type Microsoft.AspNet.Identity.UserManager{{TUser}}

```
public UserManager<TUser> UserManager { get; }
```

1.2.54 UserLoginInfo Class

Summary

Represents login information and source for a user record.

Inheritance Hierarchy

- System.Object
- *Microsoft.AspNet.Identity.UserLoginInfo*

Syntax

```
public class UserLoginInfo
```

GitHub

View on GitHub

class Microsoft.AspNet.Identity.**UserLoginInfo**

Constructors

Creates a new instance of Microsoft.AspNet.Identity.UserLoginInfo

UserLoginInfo (*System.String*, *System.String*, *System.String*) **Arguments**

- **loginProvider** (*System.String*) – The provider associated with this login information.
- **providerKey** (*System.String*) – The unique identifier for this user provided by the login provider.
- **displayName** (*System.String*) – The display name for this user provided by the login provider.

```
public UserLoginInfo(string loginProvider, string providerKey, string displayName)
```

Properties

Gets or sets the provider for this instance of Microsoft.AspNet.Identity.UserLoginInfo.

LoginProvider() **Return type**
System.String

```
public string LoginProvider { get; set; }
```

ProviderDisplayName()
Gets or sets the display name for the provider.

Return type System.String

```
public string ProviderDisplayName { get; set; }
```

ProviderKey()
Gets or sets the unique identifier for the user identity user provided by the login provider.

Return type System.String

```
public string ProviderKey { get; set; }
```

1.2.55 UserManager<TUser> Class

Summary

Provides the APIs for managing user in a persistence store.

Inheritance Hierarchy

- `System.Object`
- *`Microsoft.AspNet.Identity.UserManager<TUser>`*

Syntax

```
public class UserManager<TUser> : IDisposable where TUser : class
```

GitHub

View on GitHub

class `Microsoft.AspNet.Identity.`**`UserManager<TUser>`**

Constructors

Constructs a new instance of Microsoft.AspNet.Identity.UserManager‘1.

`UserManager` (*Microsoft.AspNet.Identity.IUserStore<TUser>*, *IOptions<Microsoft.AspNet.Identity.IdentityOptions>*, *Microsoft.AspNet.Identity.IPasswordHasher*, *System.Collections.Generic.IEnumerable<Microsoft.AspNet.Ident*, *System.Collections.Generic.IEnumerable<Microsoft.AspNet.Identity.IPasswordValidator>*, *Microsoft.AspNet.Identity.ILookupNormalizer*, *Microsoft.AspNet.Identity.IdentityErrorDescriber*, *System.Collections.Generic.IEnumerable<Microsoft.AspNet.Identity.IUserTokenProvider>*, *ILogger<Microsoft.AspNet.Identity.UserManager<TUser>>*, *IHttpContextAccessor*)

Arguments

- **`store`** (*Microsoft.AspNet.Identity.IUserStore{{TUser}}*) – The persistence store the manager will operate over.
- **`optionsAccessor`** (*IOptions{Microsoft.AspNet.Identity.IdentityOptions}*) – The accessor used to access the Microsoft.AspNet.Identity.IdentityOptions.
- **`passwordHasher`** (*Microsoft.AspNet.Identity.IPasswordHasher*) – The password hashing implementation to use when saving passwords.

- **userValidators** (*System.Collections.Generic.IEnumerable{Microsoft.AspNet.Identity.IUserValidator}*) – A collection of IUserValidator<TUser> to validate users against.
- **passwordValidators** (*System.Collections.Generic.IEnumerable{Microsoft.AspNet.Identity.IPasswordValid* – A collection of IPasswordValidator<TUser> to validate passwords against.
- **keyNormalizer** (*Microsoft.AspNet.Identity.ILookupNormalizer*) – The Microsoft.AspNet.Identity.ILookupNormalizer to use when generating index keys for users.
- **errors** (*Microsoft.AspNet.Identity.IdentityErrorDescriber*) – The Microsoft.AspNet.Identity.IdentityErrorDescriber used to provider error messages.
- **logger** (*ILogger{Microsoft.AspNet.Identity.UserManager`1}*) – The logger used to log messages, warnings and errors.
- **contextAccessor** (*IHttpContextAccessor*) – The accessor used to access the HttpContext.

```
public UserManager(IUserStore<TUser> store, IOptions<IdentityOptions> optionsAccessor, IPass
```

Methods

Increments the access failed count for the user as an asynchronous operation. If the failed access account is greater than or equal to the configured maximum number of attempts, the user will be locked out for the configured lockout time span.

AccessFailedAsync(*TUser*)

Arguments

- **user** (*{TUser}*) – The user whose failed access count to increment.

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the <see cref="T:Microsoft.AspNet.Identity.IdentityResult" /> of the operation.

```
public virtual Task<IdentityResult> AccessFailedAsync(TUser user)
```

AddClaimAsync(*TUser*, *System.Security.Claims.Claim*)

Adds the specified `claim` to the `user`, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user to add the claim to.
- **claim** (*System.Security.Claims.Claim*) – The claim to add.

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the <see cref="T:Microsoft.AspNet.Identity.IdentityResult" /> of the operation.

```
public virtual Task<IdentityResult> AddClaimAsync(TUser user, Claim claim)
```

AddClaimsAsync(*TUser*, *System.Collections.Generic.IEnumerable<System.Security.Claims.Claim>*)

Adds the specified `claims` to the `user`, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user to add the claim to.

- **claims** (*System.Collections.Generic.IEnumerable{System.Security.Claims.Claim}*) – The claims to add.

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the <see cref="T:Microsoft.AspNet.Identity.IdentityResult" /> of the operation.

```
public virtual Task<IdentityResult> AddClaimsAsync(TUser user, IEnumerable<Claim> claims)
```

AddLoginAsync(*TUser*, *Microsoft.AspNet.Identity.UserLoginInfo*)

Adds an external Microsoft.AspNet.Identity.UserLoginInfo to the specified `user`, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user to add the login to.
- **login** (*Microsoft.AspNet.Identity.UserLoginInfo*) – The external Microsoft.AspNet.Identity.UserLoginInfo to add to the specified `user`.

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the <see cref="T:Microsoft.AspNet.Identity.IdentityResult" /> of the operation.

```
public virtual Task<IdentityResult> AddLoginAsync(TUser user, UserLoginInfo login)
```

AddPasswordAsync(*TUser*, *System.String*)

Adds the `password` to the specified `user` only if the user does not already have a password, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user whose password should be set.
- **password** (*System.String*) – The password to set.

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the <see cref="T:Microsoft.AspNet.Identity.IdentityResult" /> of the operation.

```
public virtual Task<IdentityResult> AddPasswordAsync(TUser user, string password)
```

AddToRoleAsync(*TUser*, *System.String*)

Add the specified `user` to the named role, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user to add to the named role.

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the <see cref="T:Microsoft.AspNet.Identity.IdentityResult" /> of the operation.

```
public virtual Task<IdentityResult> AddToRoleAsync(TUser user, string role)
```

AddToRolesAsync(*TUser*, *System.Collections.Generic.IEnumerable<System.String>*)
Add the specified `user` to the named roles, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user to add to the named roles.

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the <see cref="T:Microsoft.AspNet.Identity.IdentityResult" /> of the operation.

```
public virtual Task<IdentityResult> AddToRolesAsync(TUser user, IEnumerable<string> roles)
```

ChangeEmailAsync(*TUser*, *System.String*, *System.String*)
Updates a users emails if the specified email change `token` is valid for the user.

Arguments

- **user** (*{TUser}*) – The user whose email should be updated.
- **newEmail** (*System.String*) – The new email address.
- **token** (*System.String*) – The change email token to be verified.

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the <see cref="T:Microsoft.AspNet.Identity.IdentityResult" /> of the operation.

```
public virtual Task<IdentityResult> ChangeEmailAsync(TUser user, string newEmail, string tok
```

ChangePasswordAsync(*TUser*, *System.String*, *System.String*)
Changes a user's password after confirming the specified `currentPassword` is correct, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user whose password should be set.
- **currentPassword** (*System.String*) – The current password to validate before changing.
- **newPassword** (*System.String*) – The new password to set for the specified `user`.

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the <see cref="T:Microsoft.AspNet.Identity.IdentityResult" /> of the operation.

```
public virtual Task<IdentityResult> ChangePasswordAsync(TUser user, string currentPassword,
```

ChangePhoneNumberAsync(*TUser*, *System.String*, *System.String*)
Sets the phone number for the specified `user` if the specified change `token` is valid.

Arguments

- **user** (*{TUser}*) – The user whose phone number to set.
- **phoneNumber** (*System.String*) – The phone number to set.
- **token** (*System.String*) – The phone number confirmation token to validate.

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the <see cref="T:Microsoft.AspNet.Identity.IdentityResult" /> of the operation.

```
public virtual Task<IdentityResult> ChangePhoneNumberAsync(TUser user, string phoneNumber, s
```

CheckPasswordAsync(*TUser*, *System.String*)

Returns a flag indicating whether the given `password` is valid for the specified `user`, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user whose password should be validated.
- **password** (*System.String*) – The password to validate

Return type System.Threading.Tasks.Task{System.Boolean}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing true if the specified <paramref name="password" /> matches the one store for the <paramref name="user" />, otherwise false.

```
public virtual Task<bool> CheckPasswordAsync(TUser user, string password)
```

ConfirmEmailAsync(*TUser*, *System.String*)

Validates that an email confirmation token matches the specified `user`, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user to validate the token against.
- **token** (*System.String*) – The email confirmation token to validate.

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the <see cref="T:Microsoft.AspNet.Identity.IdentityResult" /> of the operation.

```
public virtual Task<IdentityResult> ConfirmEmailAsync(TUser user, string token)
```

CreateAsync(*TUser*)

Creates the specified `user` in the backing store with no password, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user to create.

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the <see cref="T:Microsoft.AspNet.Identity.IdentityResult" /> of the operation.

```
public virtual Task<IdentityResult> CreateAsync(TUser user)
```

CreateAsync(*TUser*, *System.String*)

Creates the specified `user` in the backing store with given password, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user to create.

- **password** (*System.String*) – The password for the user to hash and store.

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the <see cref="T:Microsoft.AspNet.Identity.IdentityResult" /> of the operation.

```
public virtual Task<IdentityResult> CreateAsync(TUser user, string password)
```

DeleteAsync(*TUser*)

Deletes the specified `user` from the backing store, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user to delete.

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the <see cref="T:Microsoft.AspNet.Identity.IdentityResult" /> of the operation.

```
public virtual Task<IdentityResult> DeleteAsync(TUser user)
```

Dispose()

Releases all resources used by the user manager.

```
public void Dispose()
```

Dispose(*System.Boolean*)

Releases the unmanaged resources used by the role manager and optionally releases the managed resources.

Arguments

- **disposing** (*System.Boolean*) – true to release both managed and unmanaged resources; false to release only unmanaged resources.

```
protected virtual void Dispose(bool disposing)
```

FindByEmailAsync(*System.String*)

Gets the user, if any, associated with the specified, normalized email address.

Return type System.Threading.Tasks.Task{{TUser}}

Returns The task object containing the results of the asynchronous lookup operation, the user if any associated with the specified normalized email address.

```
public virtual Task<TUser> FindByEmailAsync(string email)
```

FindByIdAsync(*System.String*)

Finds and returns a user, if any, who has the specified `userId`.

Arguments

- **userId** (*System.String*) – The user ID to search for.

Return type System.Threading.Tasks.Task{{TUser}}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the user matching the specified <paramref name="userID" /> if it exists.

```
public virtual Task<TUser> FindByIdAsync(string userId)
```

FindByLoginAsync (*System.String*, *System.String*)
Retrieves the user associated with the specified external login provider and login provider key, as an asynchronous operation..

Arguments

- **loginProvider** (*System.String*) – The login provider who provided the `providerKey`.
- **providerKey** (*System.String*) – The key provided by the `loginProvider` to identify a user.

Return type System.Threading.Tasks.Task{{TUser}}

Returns The <see cref="T:System.Threading.Tasks.Task" /> for the asynchronous operation, containing the user, if any which matched the specified login provider and key.

```
public virtual Task<TUser> FindByLoginAsync(string loginProvider, string providerKey)
```

FindByNameAsync (*System.String*)
Finds and returns a user, if any, who has the specified normalized user name.

Return type System.Threading.Tasks.Task{{TUser}}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the user matching the specified <paramref name="userID" /> if it exists.

```
public virtual Task<TUser> FindByNameAsync(string userName)
```

GenerateChangeEmailTokenAsync (*TUser*, *System.String*)
Generates an email change token for the specified user, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user to generate an email change token for.

Return type System.Threading.Tasks.Task{System.String}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, an email change token.

```
public virtual Task<string> GenerateChangeEmailTokenAsync(TUser user, string newEmail)
```

GenerateChangePhoneNumberTokenAsync (*TUser*, *System.String*)
Generates a telephone number change token for the specified user, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user to generate a telephone number token for.
- **phoneNumber** (*System.String*) – The new phone number the validation token should be sent to.

Return type System.Threading.Tasks.Task{System.String}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the telephone change number token.

```
public virtual Task<string> GenerateChangePhoneNumberTokenAsync(TUser user, string phoneNumb
```

GenerateConcurrencyStampAsync(*TUser*)
Generates a value suitable for use in concurrency tracking, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user to generate the stamp for.

Return type System.Threading.Tasks.Task{System.String}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the security stamp for the specified <paramref name="user" />.

```
public virtual Task<string> GenerateConcurrencyStampAsync(TUser user)
```

GenerateEmailConfirmationTokenAsync(*TUser*)
Generates an email confirmation token for the specified user, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user to generate an email confirmation token for.

Return type System.Threading.Tasks.Task{System.String}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, an email confirmation token.

```
public virtual Task<string> GenerateEmailConfirmationTokenAsync(TUser user)
```

GeneratePasswordResetTokenAsync(*TUser*)
Generates a password reset token for the specified `user`, using the configured password reset token provider, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user to generate a password reset token for.

Return type System.Threading.Tasks.Task{System.String}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing a password reset token for the specified <paramref name="user" />.

```
public virtual Task<string> GeneratePasswordResetTokenAsync(TUser user)
```

GenerateTwoFactorTokenAsync(*TUser*, *System.String*)
Gets a two factor authentication token for the specified `user`, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user the token is for.
- **tokenProvider** (*System.String*) – The provider which will generate the token.

Return type System.Threading.Tasks.Task{System.String}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents result of the asynchronous operation, a two factor authentication token for the user.

```
public virtual Task<string> GenerateTwoFactorTokenAsync(TUser user, string tokenProvider)
```

GenerateUserTokenAsync(*TUser*, *System.String*, *System.String*)
Generates a token for the given `user` and `purpose`, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user the token will be for.

- **tokenProvider** (*System.String*) – The provider which will generate the token.
- **purpose** (*System.String*) – The purpose the token will be for.

Return type System.Threading.Tasks.Task{System.String}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents result of the asynchronous operation, a token for the given user and purpose.

```
public virtual Task<string> GenerateUserTokenAsync(TUser user, string tokenProvider, string
```

GetAccessFailedCountAsync(*TUser*)

Retrieves the current number of failed accesses for the given `user`, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user whose access failed count should be retrieved for.

Return type System.Threading.Tasks.Task{System.Int32}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that contains the result the asynchronous operation, the current failed access count for the user..

```
public virtual Task<int> GetAccessFailedCountAsync(TUser user)
```

GetClaimsAsync(*TUser*)

Gets a list of System.Security.Claims.Claim's to be belonging to the specified ``user'' as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The role whose claims to retrieve.

Return type System.Threading.Tasks.Task{System.Collections.Generic.IList{System.Security.Claims.Claim}}

Returns A <see cref="T:System.Threading.Tasks.Task`1" /> that represents the result of the asynchronous query, a list of <see cref="T:System.Security.Claims.Claim" />s.

```
public virtual Task<IList<Claim>> GetClaimsAsync(TUser user)
```

GetEmailAsync(*TUser*)

Gets the email address for the specified `user`, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user whose email should be returned.

Return type System.Threading.Tasks.Task{System.String}

Returns The task object containing the results of the asynchronous operation, the email address for the specified <paramref name="user" />.

```
public virtual Task<string> GetEmailAsync(TUser user)
```

GetLockoutEnabledAsync(*TUser*)

Retrieves a flag indicating whether user lockout can enabled for the specified user, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user whose ability to be locked out should be returned.

Return type System.Threading.Tasks.Task{System.Boolean}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, true if a user can be locked out, otherwise false.

```
public virtual Task<bool> GetLockoutEnabledAsync(TUser user)
```

GetLockoutEndDateAsync(*TUser*)

Gets the last System.DateTimeOffset a user's last lockout expired, if any, as an asynchronous operation. Any time in the past should be indicates a user is not locked out.

Arguments

- **user** (*{TUser}*) – The user whose lockout date should be retrieved.

Return type System.Threading.Tasks.Task{System.Nullable{System.DateTimeOffset}}

Returns A <see cref="T:System.Threading.Tasks.Task`1" /> that represents the lookup, a <see cref="T:System.DateTimeOffset" /> containing the last time a user's lockout expired, if any.

```
public virtual Task<DateTimeOffset? > GetLockoutEndDateAsync(TUser user)
```

GetLoginsAsync(*TUser*)

Retrieves the associated logins for the specified <param ref="user" />, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user whose associated logins to retrieve.

Return type System.Threading.Tasks.Task{System.Collections.Generic.IList{Microsoft.AspNet.Identity.UserLoginIn

Returns The <see cref="T:System.Threading.Tasks.Task" /> for the asynchronous operation, containing a list of <see cref="T:Microsoft.AspNet.Identity.UserLoginInfo" /> for the specified <paramref name="user" />, if any.

```
public virtual Task<IList<UserLoginInfo>> GetLoginsAsync(TUser user)
```

GetPhoneNumberAsync(*TUser*)

Gets the telephone number, if any, for the specified `user`, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user whose telephone number should be retrieved.

Return type System.Threading.Tasks.Task{System.String}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the user's telephone number, if any.

```
public virtual Task<string> GetPhoneNumberAsync(TUser user)
```

GetRolesAsync(*TUser*)

Gets a list of role names the specified `user` belongs to, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user whose role names to retrieve.

Return type System.Threading.Tasks.Task{System.Collections.Generic.IList{System.String}}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing a list of role names.

```
public virtual Task<IList<string>> GetRolesAsync(TUser user)
```

GetSecurityStampAsync(*TUser*)

Get the security stamp for the specified `user`, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user whose security stamp should be set.

Return type System.Threading.Tasks.Task{System.String}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the security stamp for the specified <paramref name="user" />.

```
public virtual Task<string> GetSecurityStampAsync(TUser user)
```

GetTwoFactorEnabledAsync(*TUser*)

Returns a flag indicating whether the specified `user` has two factor authentication enabled or not, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user whose two factor authentication enabled status should be retrieved.

Return type System.Threading.Tasks.Task{System.Boolean}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, true if the specified <paramref name="user " /> has two factor authentication enabled, otherwise false.

```
public virtual Task<bool> GetTwoFactorEnabledAsync(TUser user)
```

GetUserIdAsync(*TUser*)

Gets the user identifier for the specified `user`, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user whose identifier should be retrieved.

Return type System.Threading.Tasks.Task{System.String}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the identifier for the specified <paramref name="user" />.

```
public virtual Task<string> GetUserIdAsync(TUser user)
```

GetUserNameAsync(*TUser*)

Gets the user name for the specified `user`, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user whose name should be retrieved.

Return type System.Threading.Tasks.Task{System.String}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the name for the specified <paramref name="user" />.

```
public virtual Task<string> GetUserNameAsync(TUser user)
```

GetUsersForClaimAsync(*System.Security.Claims.Claim*)

Return type System.Threading.Tasks.Task{System.Collections.Generic.IList{{TUser}}}

```
public virtual Task<IList<TUser>> GetUsersForClaimAsync(Claim claim)
```

GetUsersInRoleAsync(*System.String*)

Returns a list of users from the user store who have the specified System.Security.Claims.Claim.

Return type System.Threading.Tasks.Task{System.Collections.Generic.IList{{TUser}}}

Returns A <see cref="T:System.Threading.Tasks.Task`1" /> that represents the result of the asynchronous query, a list of <typeparamref name="TUser" />s who have the specified claim.

```
public virtual Task<IList<TUser>> GetUsersInRoleAsync(string roleName)
```

GetValidTwoFactorProvidersAsync(*TUser*)

Gets a list of valid two factor token providers for the specified `user`, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user the whose two factor authentication providers will be returned.

Return type System.Threading.Tasks.Task{System.Collections.Generic.IList{System.String}}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents result of the asynchronous operation, a list of two factor authentication providers for the specified user.

```
public virtual Task<IList<string>> GetValidTwoFactorProvidersAsync(TUser user)
```

HasPasswordAsync(*TUser*)

Gets a flag indicating whether the specified `user` has a password, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user to return a flag for, indicating whether they have a password or not.

Return type System.Threading.Tasks.Task{System.Boolean}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, returning true if the specified <paramref name="user" /> has a password otherwise false.

```
public virtual Task<bool> HasPasswordAsync(TUser user)
```

IsEmailConfirmedAsync(*TUser*)

Gets a flag indicating whether the email address for the specified `user` has been verified, true if the email address is verified otherwise false, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user whose email confirmation status should be returned.

Return type System.Threading.Tasks.Task{System.Boolean}

Returns The task object containing the results of the asynchronous operation, a flag indicating whether the email address for the specified <paramref name="user" /> has been confirmed or not.

```
public virtual Task<bool> IsEmailConfirmedAsync(TUser user)
```

IsInRoleAsync(*TUser*, *System.String*)

Returns a flag indicating whether the specified `user` is a member of the give named role, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user whose role membership should be checked.
- **role** (*System.String*) – The name of the role to be checked.

Return type System.Threading.Tasks.Task{System.Boolean}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing a flag indicating whether the specified <see cref="!:user" /> is a member of the named role.

```
public virtual Task<bool> IsInRoleAsync(TUser user, string role)
```

IsLockedOutAsync(*TUser*)

Returns a flag indicating whether the specified `user` his locked out, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user whose locked out status should be retrieved.

Return type System.Threading.Tasks.Task{System.Boolean}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, true if the specified <paramref name="user " /> is locked out, otherwise false.

```
public virtual Task<bool> IsLockedOutAsync(TUser user)
```

IsPhoneNumberConfirmedAsync(*TUser*)

Gets a flag indicating whether the specified `user`'s telephone number has been confirmed, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user to return a flag for, indicating whether their telephone number is confirmed.

Return type System.Threading.Tasks.Task{System.Boolean}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, returning true if the specified <paramref name="user" /> has a confirmed telephone number otherwise false.

```
public virtual Task<bool> IsPhoneNumberConfirmedAsync(TUser user)
```

NormalizeKey(*System.String*)

Normalize a key (user name, email) for consistent comparisons.

Arguments

- **key** (*System.String*) – The key to normalize.

Return type System.String

Returns A normalized value representing the specified <paramref name="key" />.

```
public virtual string NormalizeKey(string key)
```

RegisterTokenProvider(*Microsoft.AspNet.Identity.IUserTokenProvider*)

Registers a token provider.

Arguments

- **provider** (*Microsoft.AspNet.Identity.IUserTokenProvider*) – The provider to register.

```
public virtual void RegisterTokenProvider(IUserTokenProvider provider)
```

RemoveClaimAsync(*TUser*, *System.Security.Claims.Claim*)

Removes the specified `claim` from the given `user`.

Arguments

- **user** (*{TUser}*) – The user to remove the specified `claims` from.

- **claim** (*System.Security.Claims.Claim*) – The System.Security.Claims.Claim to remove.

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the <see cref="T:Microsoft.AspNet.Identity.IdentityResult" /> of the operation.

```
public virtual Task<IdentityResult> RemoveClaimAsync(TUser user, Claim claim)
```

RemoveClaimsAsync(*TUser*, *System.Collections.Generic.IEnumerable<System.Security.Claims.Claim>*)

Removes the specified `claims` from the given `user`.

Arguments

- **user** (*{TUser}*) – The user to remove the specified `claims` from.
- **claims** (*System.Collections.Generic.IEnumerable{System.Security.Claims.Claim}*) – A collection of :dn:ref:'System.Security.Claims.Claim's to remove.

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the <see cref="T:Microsoft.AspNet.Identity.IdentityResult" /> of the operation.

```
public virtual Task<IdentityResult> RemoveClaimsAsync(TUser user, IEnumerable<Claim> claims)
```

RemoveFromRoleAsync(*TUser*, *System.String*)

Removes the specified `user` from the named role, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user to remove from the named role.

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the <see cref="T:Microsoft.AspNet.Identity.IdentityResult" /> of the operation.

```
public virtual Task<IdentityResult> RemoveFromRoleAsync(TUser user, string role)
```

RemoveFromRolesAsync(*TUser*, *System.Collections.Generic.IEnumerable<System.String>*)

Removes the specified `user` from the named roles, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user to remove from the named roles.

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the <see cref="T:Microsoft.AspNet.Identity.IdentityResult" /> of the operation.

```
public virtual Task<IdentityResult> RemoveFromRolesAsync(TUser user, IEnumerable<string> rol
```

RemoveLoginAsync(*TUser*, *System.String*, *System.String*)

Attempts to remove the provided external login information from the specified `user`, as an asynchronous operation. and returns a flag indicating whether the removal succeed or not.

Arguments

- **user** (*{TUser}*) – The user to remove the login information from.

- **loginProvider** (*System.String*) – The login provide whose information should be removed.
- **providerKey** (*System.String*) – The key given by the external login provider for the specified user.

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the <see cref="T:Microsoft.AspNet.Identity.IdentityResult" /> of the operation.

```
public virtual Task<IdentityResult> RemoveLoginAsync(TUser user, string loginProvider, strin
```

RemovePasswordAsync(*TUser*, *System.Threading.CancellationToken*)

Removes a user's password, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user whose password should be removed.
- **cancellationToken** (*System.Threading.CancellationToken*) – The `Microsoft.AspNet.Identity.UserManager`1.CancellationToken` used to propagate notifications that the operation should be canceled.

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the <see cref="T:Microsoft.AspNet.Identity.IdentityResult" /> of the operation.

```
public virtual Task<IdentityResult> RemovePasswordAsync(TUser user, CancellationToken cancel
```

ReplaceClaimAsync(*TUser*, *System.Security.Claims.Claim*, *System.Security.Claims.Claim*)

Replaces the given `claim` on the specified `user` with the `newClaim`

Arguments

- **user** (*{TUser}*) – The user to replace the claim on.
- **claim** (*System.Security.Claims.Claim*) – The claim to replace.
- **newClaim** (*System.Security.Claims.Claim*) – The new claim to replace the existing `claim` with.

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the <see cref="T:Microsoft.AspNet.Identity.IdentityResult" /> of the operation.

```
public virtual Task<IdentityResult> ReplaceClaimAsync(TUser user, Claim claim, Claim newClai
```

ResetAccessFailedCountAsync(*TUser*)

Resets the access failed count for the specified `user`, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user whose failed access count should be reset.

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the <see cref="T:Microsoft.AspNet.Identity.IdentityResult" /> of the operation.

```
public virtual Task<IdentityResult> ResetAccessFailedCountAsync(TUser user)
```

ResetPasswordAsync(*TUser*, *System.String*, *System.String*)

Resets the `user`'s password to the specified `newPassword` after validating the given password reset `token`, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user whose password should be reset.
- **token** (*System.String*) – The password reset token to verify.
- **newPassword** (*System.String*) – The new password to set if reset token verification fails.

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the <see cref="T:Microsoft.AspNet.Identity.IdentityResult" /> of the operation.

```
public virtual Task<IdentityResult> ResetPasswordAsync(TUser user, string token, string newP
```

SetEmailAsync(*TUser*, *System.String*)

Sets the `email` address for a `user`, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user whose email should be set.
- **email** (*System.String*) – The email to set.

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the <see cref="T:Microsoft.AspNet.Identity.IdentityResult" /> of the operation.

```
public virtual Task<IdentityResult> SetEmailAsync(TUser user, string email)
```

SetLockoutEnabledAsync(*TUser*, *System.Boolean*)

Sets a flag indicating whether the specified `user` is locked out, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user whose locked out status should be set.
- **enabled** (*System.Boolean*) – Flag indicating whether the user is locked out or not.

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, the <see cref="T:Microsoft.AspNet.Identity.IdentityResult" /> of the operation

```
public virtual Task<IdentityResult> SetLockoutEnabledAsync(TUser user, bool enabled)
```

SetLockoutEndDateAsync(*TUser*, *System.Nullable<System.DateTimeOffset>*)

Locks out a user until the specified end date has passed, as an asynchronous operation. Setting a end date in the past immediately unlocks a user.

Arguments

- **user** (*{TUser}*) – The user whose lockout date should be set.
- **lockoutEnd** (*System.Nullable{System.DateTimeOffset}*) – The System.DateTimeOffset after which the `user`'s lockout should end.

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the <see cref="T:Microsoft.AspNet.Identity.IdentityResult" /> of the operation.

```
public virtual Task<IdentityResult> SetLockoutEndDateAsync(TUser user, DateTimeOffset? lockc
```

SetPhoneNumberAsync(*TUser*, *System.String*)
Sets the phone number for the specified `user`.

Arguments

- **user** (*{TUser}*) – The user whose phone number to set.
- **phoneNumber** (*System.String*) – The phone number to set.

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the <see cref="T:Microsoft.AspNet.Identity.IdentityResult" /> of the operation.

```
public virtual Task<IdentityResult> SetPhoneNumberAsync(TUser user, string phoneNumber)
```

SetTwoFactorEnabledAsync(*TUser*, *System.Boolean*)
Sets a flag indicating whether the specified `user` has two factor authentication enabled or not, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user whose two factor authentication enabled status should be set.

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, the <see cref="T:Microsoft.AspNet.Identity.IdentityResult" /> of the operation

```
public virtual Task<IdentityResult> SetTwoFactorEnabledAsync(TUser user, bool enabled)
```

SetUserNameAsync(*TUser*, *System.String*)
Sets the given `userName` for the specified `user`, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user whose name should be set.
- **userName** (*System.String*) – The user name to set.

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation.

```
public virtual Task<IdentityResult> SetUserNameAsync(TUser user, string userName)
```

UpdateAsync(*TUser*)
Updates the specified `user` in the backing store, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user to update.

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the <see cref="T:Microsoft.AspNet.Identity.IdentityResult" /> of the operation.

```
public virtual Task<IdentityResult> UpdateAsync(TUser user)
```

UpdateNormalizedEmailAsync(*TUser*)

Updates the normalized email for the specified `user`.

Arguments

- **user** (*{TUser}*) – The user whose email address should be normalized and updated.

Return type System.Threading.Tasks.Task

Returns The task object representing the asynchronous operation.

```
public virtual Task UpdateNormalizedEmailAsync(TUser user)
```

UpdateNormalizedUserNameAsync(*TUser*)

Updates the normalized user name for the specified `user`, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user whose user name should be normalized and updated.

Return type System.Threading.Tasks.Task

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation.

```
public virtual Task UpdateNormalizedUserNameAsync(TUser user)
```

UpdateSecurityStampAsync(*TUser*)

Regenerates the security stamp for the specified `user`, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user whose security stamp should be regenerated.

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the <see cref="T:Microsoft.AspNet.Identity.IdentityResult" /> of the operation.

```
public virtual Task<IdentityResult> UpdateSecurityStampAsync(TUser user)
```

VerifyChangePhoneNumberTokenAsync(*TUser*, *System.String*, *System.String*)

Returns a flag indicating whether the specified `user`'s phone number change verification token is valid for the given `phoneNumber`, as an asynchronous operation.

Arguments

- **user** (*{TUser}*) – The user to validate the token against.
- **token** (*System.String*) – The telephone number change token to validate.
- **phoneNumber** (*System.String*) – The telephone number the token was generated for.

Return type System.Threading.Tasks.Task{System.Boolean}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, returning true if the <paramref name="token" /> is valid, otherwise false.

```
public virtual Task<bool> VerifyChangePhoneNumberTokenAsync(TUser user, string token, string
```

VerifyPasswordAsync(*Microsoft.AspNet.Identity.IUserPasswordStore<TUser>*, *TUser*, *System.String*)

Returns a Microsoft.AspNet.Identity.PasswordVerificationResult indicating the result of a password hash comparison.

Arguments

- **store** (*Microsoft.AspNet.Identity.IUserPasswordStore{TUser}*) – The store containing a user's password.
- **user** (*TUser*) – The user whose password should be verified.
- **password** (*System.String*) – The password to verify.

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.PasswordVerificationResult}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the <see cref="T:Microsoft.AspNet.Identity.PasswordVerificationResult" /> of the operation.

```
protected virtual Task<PasswordVerificationResult> VerifyPasswordAsync(IUserPasswordStore<TU
```

VerifyTwoFactorTokenAsync(*TUser*, *System.String*, *System.String*)

Verifies the specified two factor authentication `token` against the `user`, as an asynchronous operation.

Arguments

- **user** (*TUser*) – The user the token is supposed to be for.
- **tokenProvider** (*System.String*) – The provider which will verify the token.
- **token** (*System.String*) – The token to verify.

Return type System.Threading.Tasks.Task{System.Boolean}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents result of the asynchronous operation, true if the token is valid, otherwise false.

```
public virtual Task<bool> VerifyTwoFactorTokenAsync(TUser user, string tokenProvider, string
```

VerifyUserTokenAsync(*TUser*, *System.String*, *System.String*, *System.String*)

Returns a flag indicating whether the specified `token` is valid for the given `user` and `purpose`, as an asynchronous operation.

Arguments

- **user** (*TUser*) – The user to validate the token against.
- **tokenProvider** (*System.String*) – The token provider used to generate the token.
- **purpose** (*System.String*) – The purpose the token should be generated for.
- **token** (*System.String*) – The token to validate

Return type System.Threading.Tasks.Task{System.Boolean}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, returning true if the <paramref name="token" /> is valid, otherwise false.

```
public virtual Task<bool> VerifyUserTokenAsync(TUser user, string tokenProvider, string purp
```

Properties

Gets the ILogger used to log messages from the manager.

Logger()

Return type

ILogger

```
protected virtual ILogger Logger { get; set; }
```

Store()

Gets or sets the persistence store the manager operates over.

Return type Microsoft.AspNet.Identity.IUserStore{{TUser}}

```
protected IUserStore<TUser> Store { get; set; }
```

SupportsQueryableUsers()

Gets a flag indicating whether the backing user store supports returning

System.Linq.IQueryable collections of information.

Return type System.Boolean

```
public virtual bool SupportsQueryableUsers { get; }
```

SupportsUserClaim()

Gets a flag indicating whether the backing user store supports user claims.

Return type System.Boolean

```
public virtual bool SupportsUserClaim { get; }
```

SupportsUserEmail()

Gets a flag indicating whether the backing user store supports user emails.

Return type System.Boolean

```
public virtual bool SupportsUserEmail { get; }
```

SupportsUserLockout()

Gets a flag indicating whether the backing user store supports user lock-outs.

Return type System.Boolean

```
public virtual bool SupportsUserLockout { get; }
```

SupportsUserLogin()

Gets a flag indicating whether the backing user store supports external logins.

Return type System.Boolean

```
public virtual bool SupportsUserLogin { get; }
```

SupportsUserPassword()

Gets a flag indicating whether the backing user store supports user passwords.

Return type System.Boolean

```
public virtual bool SupportsUserPassword { get; }
```

SupportsUserPhoneNumber()

Gets a flag indicating whether the backing user store supports user telephone numbers.

Return type System.Boolean

```
public virtual bool SupportsUserPhoneNumber { get; }
```

SupportsUserRole()
Gets a flag indicating whether the backing user store supports user roles.

Return type System.Boolean

```
public virtual bool SupportsUserRole { get; }
```

SupportsUserSecurityStamp()
Gets a flag indicating whether the backing user store supports security stamps.

Return type System.Boolean

```
public virtual bool SupportsUserSecurityStamp { get; }
```

SupportsUserTwoFactor()
Gets a flag indicating whether the backing user store supports two factor authentication.

Return type System.Boolean

```
public virtual bool SupportsUserTwoFactor { get; }
```

Users()
Returns an IQueryable of users if the store is an IQueryableUserStore

Return type System.Linq.IQueryable{{TUser}}

```
public virtual IQueryable<TUser> Users { get; }
```

1.2.56 UserOptions Class

Summary

Options for user validation.

Inheritance Hierarchy

- `System.Object`
- *`Microsoft.AspNet.Identity.UserOptions`*

Syntax

```
public class UserOptions
```

GitHub

View on GitHub

class Microsoft.AspNet.Identity.**UserOptions**

Properties

Gets or sets the list of allowed characters in the username used to validate user names.

AllowedUserNameCharacters()

Return type
System.String

```
public string AllowedUserNameCharacters { get; set; }
```

RequireUniqueEmail()
Gets or sets a flag indicating whether the application requires unique emails for its users.

Return type System.Boolean

```
public bool RequireUniqueEmail { get; set; }
```

1.2.57 UserValidator Class

Summary

Provides validation services for user classes.

Inheritance Hierarchy

- System.Object
- *Microsoft.AspNet.Identity.UserValidator*

Syntax

```
public class UserValidator : IUserValidator
```

GitHub

View on GitHub

class Microsoft.AspNet.Identity.**UserValidator**

Constructors

Creates a new instance of UserValidator<TUser>/

UserValidator (*Microsoft.AspNet.Identity.IdentityErrorDescriber*) **Arguments**

- **errors** (*Microsoft.AspNet.Identity.IdentityErrorDescriber*) – The Microsoft.AspNet.Identity.IdentityErrorDescriber used to provider error messages.

```
public UserValidator(IdentityErrorDescriber errors = null)
```

Properties

Gets the Microsoft.AspNet.Identity.IdentityErrorDescriber used to provider error messages for the current UserValidator<TUser>.

Describer () **Return type**

Microsoft.AspNet.Identity.IdentityErrorDescriber

```
public IdentityErrorDescriber Describer { get; }
```

Methods

Validates the specified `user` as an asynchronous operation.

ValidateAsync<TUser> (*Microsoft.AspNet.Identity.UserManager<TUser>*, *TUser*) **Arguments**

- **manager** (*Microsoft.AspNet.Identity.UserManager{{TUser}}*) – The Microsoft.AspNet.Identity.UserManager`1 that can be used to retrieve user properties.
- **user** (*{TUser}*) – The user to validate.

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

Returns The <see cref="T:System.Threading.Tasks.Task" /> that represents the asynchronous operation, containing the <see cref="T:Microsoft.AspNet.Identity.IdentityResult" /> of the validation operation.

```
public virtual Task<IdentityResult> ValidateAsync<TUser>(UserManager<TUser> manager, TUser u
```

namespace Microsoft.AspNet.**Identity**

Enumerations

enum ***Microsoft.AspNet.Identity.PasswordHasherCompatibilityMode*** Specifies the format used for hashing passwords.

enum ***Microsoft.AspNet.Identity.PasswordVerificationResult*** Specifies the results for password verification.

Interfaces

interface ***Microsoft.AspNet.Identity.ILookupNormalizer*** Provides an abstraction for normalizing keys for lookup purposes.

interface *Microsoft.AspNet.Identity.IPasswordHasher* Provides an abstraction for hashing passwords.

interface *Microsoft.AspNet.Identity.IPasswordValidator* Provides an abstraction for validating passwords.

interface *Microsoft.AspNet.Identity.IQueryableRoleStore<TRole>* Provides an abstraction for querying roles in a Role store.

interface *Microsoft.AspNet.Identity.IQueryableUserStore<TUser>* Provides an abstraction for querying roles in a User store.

interface *Microsoft.AspNet.Identity.IRoleClaimStore<TRole>* Provides an abstraction for a store of role specific claims.

interface *Microsoft.AspNet.Identity.IRoleStore<TRole>* Provides an abstraction for a storage and management of roles.

interface *Microsoft.AspNet.Identity.IRoleValidator* Provides an abstraction for a validating a role.

interface *Microsoft.AspNet.Identity.ISecurityStampValidator* Provides an abstraction for a validating a security stamp of an incoming identity, and regenerating or rejecting the identity based on the validation result.

interface *Microsoft.AspNet.Identity.IUserClaimStore<TUser>* Provides an abstraction for a store of claims for a user.

interface *Microsoft.AspNet.Identity.IUserClaimsPrincipalFactory<TUser>* Provides an abstraction for a factory to create a System.Security.Claims.ClaimsPrincipal from a user.

interface *Microsoft.AspNet.Identity.IUserEmailStore<TUser>* Provides an abstraction for the storage and management of user email addresses.

interface *Microsoft.AspNet.Identity.IUserLockoutStore<TUser>* Provides an abstraction for a storing information which can be used to implement account lockout, including access failures and lockout status

interface *Microsoft.AspNet.Identity.IUserLoginStore<TUser>* Provides an abstraction for storing information that maps external login information provided by Microsoft Account, Facebook etc. to a user account.

interface *Microsoft.AspNet.Identity.IUserPasswordStore<TUser>* Provides an abstraction for a store containing users' password hashes..

interface *Microsoft.AspNet.Identity.IUserPhoneNumberStore<TUser>* Provides an abstraction for a store containing users' telephone numbers.

interface *Microsoft.AspNet.Identity.IUserRoleStore<TUser>* Provides an abstraction for a store which maps users to roles.

interface *Microsoft.AspNet.Identity.IUserSecurityStampStore<TUser>* Provides an abstraction for a store which stores a user's security stamp.

interface *Microsoft.AspNet.Identity.IUserStore<TUser>* Provides an abstraction for a store which manages user accounts.

interface *Microsoft.AspNet.Identity.IUserTokenProvider* Provides an abstraction for token generators.

interface *Microsoft.AspNet.Identity.IUserTwoFactorStore<TUser>* Provides an abstraction to store a flag indicating whether a user has two factor authentication enabled.

interface ***Microsoft.AspNet.Identity.IUserValidator*** Provides an abstraction for user validation.

Classes

class ***Microsoft.AspNet.Identity.ClaimsIdentityOptions*** Options used to configure the claim types used for well known claims.

class ***Microsoft.AspNet.Identity.DataProtectionTokenProviderOptions*** Contains options for the DataProtectorTokenProvider<TUser>.

class ***Microsoft.AspNet.Identity.DataProtectorTokenProvider*** Provides protection and validation of identity tokens.

class ***Microsoft.AspNet.Identity.EmailTokenProvider*** TokenProvider that generates tokens from the user's security stamp and notifies a user via email.

class ***Microsoft.AspNet.Identity.EmailTokenProviderOptions*** Options for the EmailTokenProvider<TUser> class.

class ***Microsoft.AspNet.Identity.ExternalLoginInfo*** Represents login information, source and externally source principal for a user record

class ***Microsoft.AspNet.Identity.IdentityBuilder*** Helper functions for configuring identity services.

class ***Microsoft.AspNet.Identity.IdentityEntityFrameworkServices*** Default services

class ***Microsoft.AspNet.Identity.IdentityError*** Encapsulates an error from the identity subsystem.

class ***Microsoft.AspNet.Identity.IdentityErrorDescriber*** Service to enable localization for application facing identity errors.

class ***Microsoft.AspNet.Identity.IdentityOptions*** Represents all the options you can used to configure the identity system.

class ***Microsoft.AspNet.Identity.IdentityResult*** Represents the result of an identity operation.

class ***Microsoft.AspNet.Identity.LockoutOptions*** Options for configuring user lockout.

class ***Microsoft.AspNet.Identity.PasswordHasher*** Implements the standard Identity password hashing.

class ***Microsoft.AspNet.Identity.PasswordHasherOptions*** Specifies options for password hashing.

class ***Microsoft.AspNet.Identity.PasswordOptions*** Specifies options for password requirements.

class ***Microsoft.AspNet.Identity.PasswordValidator*** Provides the default password policy for Identity.

class ***Microsoft.AspNet.Identity.PhoneNumberTokenProvider*** Represents a token provider that generates tokens from a user's security stamp and sends them to the user via their phone number.

class ***Microsoft.AspNet.Identity.PhoneNumberTokenProviderOptions*** Encapsulations options for a PhoneNumberTokenProvider<TUser>.

class ***Microsoft.AspNet.Identity.RoleManager<TRole>*** Provides the APIs for managing roles in a persistence store.

class ***Microsoft.AspNet.Identity.RoleValidator*** Provides the default validation of roles.

class ***Microsoft.AspNet.Identity.SecurityStampValidator*** Static helper class used to configure a CookieAuthenticationNotifications to validate a cookie against a user's security stamp.

class ***Microsoft.AspNet.Identity.SecurityStampValidator<TUser>*** Provides default implementation of validation functions for security stamps.

class ***Microsoft.AspNet.Identity.SignInManager<TUser>*** Provides the APIs for user sign in.

class ***Microsoft.AspNet.Identity.SignInOptions*** Options for configuring sign in..

class ***Microsoft.AspNet.Identity.SignInResult*** Represents the result of a sign-in operation.

class ***Microsoft.AspNet.Identity.TotpSecurityStampBasedTokenProvider*** Represents a token provider that generates time based codes using the user's security stamp.

class ***Microsoft.AspNet.Identity.UpperInvariantLookupNormalizer*** Implements Microsoft.AspNet.Identity.ILookupNormalizer by converting keys to their upper cased invariant culture representation.

class ***Microsoft.AspNet.Identity.UserClaimsPrincipalFactory<TUser, TRole>*** Provides methods to create a claims principal for a given user.

class ***Microsoft.AspNet.Identity.UserLoginInfo*** Represents login information and source for a user record.

class ***Microsoft.AspNet.Identity.UserManager<TUser>*** Provides the APIs for managing user in a persistence store.

class ***Microsoft.AspNet.Identity.UserOptions*** Options for user validation.

class ***Microsoft.AspNet.Identity.UserValidator*** Provides validation services for user classes.

1.3 Microsoft.AspNet.Identity.EntityFramework Namespace

1.3.1 IdentityDbContext Class

Inheritance Hierarchy

- `DbContext`
- `Microsoft.AspNet.Identity.EntityFramework.IdentityDbContext{Microsoft.AspNet.Identity.`
- *`Microsoft.AspNet.Identity.EntityFramework.IdentityDbContext`*

Syntax

```
public class IdentityDbContext : IdentityDbContext<IdentityUser, IdentityRole, string>
```

GitHub

View on GitHub

class Microsoft.AspNet.Identity.EntityFramework.**IdentityDbContext**

1.3.2 IdentityDbContext<TUser> Class

Inheritance Hierarchy

- DbContext
- Microsoft.AspNet.Identity.EntityFramework.IdentityDbContext{{TUser},Microsoft.AspNet.I
- *Microsoft.AspNet.Identity.EntityFramework.IdentityDbContext<TUser>*

Syntax

```
public class IdentityDbContext<TUser> : IdentityDbContext<TUser, IdentityRole, string> where TUser :
```

GitHub

View on GitHub

class Microsoft.AspNet.Identity.EntityFramework.**IdentityDbContext<TUser>**

1.3.3 IdentityDbContext<TUser, TRole, TKey> Class

Inheritance Hierarchy

- DbContext
- *Microsoft.AspNet.Identity.EntityFramework.IdentityDbContext<TUser, TRole, TKey>*

Syntax

```
public class IdentityDbContext<TUser, TRole, TKey> : DbContext where TUser : IdentityUser<TKey> where
```

GitHub

View on GitHub

class Microsoft.AspNet.Identity.EntityFramework.**IdentityDbContext<TUser, TRole, TKey>**

Methods

OnModelCreating(*ModelBuilder*)

```
protected override void OnModelCreating(ModelBuilder builder)
```

Properties

RoleClaims()

Return type DbSet{Microsoft.AspNet.Identity.EntityFramework.IdentityRoleClaim{{TKey}}}

```
public DbSet<IdentityRoleClaim<TKey>> RoleClaims { get; set; }
```

Roles()

Return type DbSet{{TRole}}

```
public DbSet<TRole> Roles { get; set; }
```

UserClaims()

Return type DbSet{Microsoft.AspNet.Identity.EntityFramework.IdentityUserClaim{{TKey}}}

```
public DbSet<IdentityUserClaim<TKey>> UserClaims { get; set; }
```

UserLogins()

Return type DbSet{Microsoft.AspNet.Identity.EntityFramework.IdentityUserLogin{{TKey}}}

```
public DbSet<IdentityUserLogin<TKey>> UserLogins { get; set; }
```

UserRoles()

Return type DbSet{Microsoft.AspNet.Identity.EntityFramework.IdentityUserRole{{TKey}}}

```
public DbSet<IdentityUserRole<TKey>> UserRoles { get; set; }
```

Users()

Return type DbSet{{TUser}}

```
public DbSet<TUser> Users { get; set; }
```

1.3.4 IdentityRole Class

Summary

Represents a Role entity

Inheritance Hierarchy

- `System.Object`
- `Microsoft.AspNet.Identity.EntityFramework.IdentityRole{System.String}`
- *`Microsoft.AspNet.Identity.EntityFramework.IdentityRole`*

Syntax

```
public class IdentityRole : IdentityRole<string>
```

GitHub

View on GitHub

class `Microsoft.AspNet.Identity.EntityFramework.`**`IdentityRole`**

Constructors

Constructor
`IdentityRole`()

```
public IdentityRole()
```

`IdentityRole` (*System.String*)

Constructor

```
public IdentityRole(string roleName)
```

1.3.5 IdentityRoleClaim<TKey> Class

Summary

EntityType that represents one specific role claim

Inheritance Hierarchy

- `System.Object`
- *`Microsoft.AspNet.Identity.EntityFramework.IdentityRoleClaim<TKey>`*

Syntax

```
public class IdentityRoleClaim<TKey> where TKey : IEquatable<TKey>
```

GitHub

View on GitHub

class `Microsoft.AspNet.Identity.EntityFramework.`**`IdentityRoleClaim<TKey>`**

Properties

Claim type

`ClaimType``()`

Return type
System.String

```
public virtual string ClaimType { get; set; }
```

`ClaimValue``()`
Claim value

Return type System.String

```
public virtual string ClaimValue { get; set; }
```

`Id``()`
Primary key

Return type System.Int32

```
public virtual int Id { get; set; }
```

`RoleId``()`
User Id for the role this claim belongs to

Return type {TKey}

```
        public virtual TKey RoleId { get; set; }
```

1.3.6 IdentityRole<TKey> Class

Summary

Represents a Role entity

Inheritance Hierarchy

- `System.Object`
- *`Microsoft.AspNet.Identity.EntityFramework.IdentityRole<TKey>`*

Syntax

```
public class IdentityRole<TKey> where TKey : IEquatable<TKey>
```

GitHub

View on GitHub

class `Microsoft.AspNet.Identity.EntityFramework.`**`IdentityRole<TKey>`**

Constructors

```
    IdentityRole()
        public IdentityRole()
```

`IdentityRole` (*System.String*)
Constructor

```
        public IdentityRole(string roleName)
```

Properties

Navigation property for claims in the role

Claims()

Return type System.Collections.Generic.ICollection{Microsoft.AspNet.Identity.EntityFramework.IdentityRoleClaim{{TKey}}

```
public virtual ICollection<IdentityRoleClaim<TKey>> Claims { get; }
```

ConcurrencyStamp()

A random value that should change whenever a role is persisted to the store

Return type System.String

```
public virtual string ConcurrencyStamp { get; set; }
```

Id()

Role id

Return type {TKey}

```
public virtual TKey Id { get; set; }
```

Name()

Role name

Return type System.String

```
public virtual string Name { get; set; }
```

NormalizedName()

Return type System.String

```
public virtual string NormalizedName { get; set; }
```

Users()

Navigation property for users in the role

Return type System.Collections.Generic.ICollection{Microsoft.AspNet.Identity.EntityFramework.IdentityUserRole{

```
public virtual ICollection<IdentityUserRole<TKey>> Users { get; }
```

Methods

Returns a friendly name

ToString()

Return type System.String

```
public override string ToString()
```

1.3.7 IdentityUser Class

- *Inheritance Hierarchy*
- *Syntax*
- *GitHub*
- *Constructors*

Inheritance Hierarchy

- System.Object
- Microsoft.AspNet.Identity.EntityFramework.IdentityUser{System.String}
- *Microsoft.AspNet.Identity.EntityFramework.IdentityUser*

Syntax

```
public class IdentityUser : IdentityUser<string>
```

GitHub

View on GitHub

class Microsoft.AspNet.Identity.EntityFramework.**IdentityUser**

Constructors

IdentityUser()

IdentityUser(*System.String*)

```
public IdentityUser(string userName)
```

1.3.8 IdentityUserClaim<TKey> Class

Summary

EntityType that represents one specific user claim

Inheritance Hierarchy

- System.Object
- *Microsoft.AspNet.Identity.EntityFramework.IdentityUserClaim<TKey>*

Syntax

```
public class IdentityUserClaim<TKey> where TKey : IEquatable<TKey>
```

GitHub

View on GitHub

class Microsoft.AspNet.Identity.EntityFramework.**IdentityUserClaim<TKey>**

Properties

Claim type

ClaimType()

Return type

System.String

```
public virtual string ClaimType { get; set; }
```

ClaimValue()

Claim value

Return type System.String

```
public virtual string ClaimValue { get; set; }
```

Id()

Primary key

Return type System.Int32

```
public virtual int Id { get; set; }
```

UserId()

User Id for the user who owns this claim

Return type {TKey}

```
public virtual TKey UserId { get; set; }
```

1.3.9 IdentityUserLogin<TKey> Class

Summary

Entity type for a user's login (i.e. facebook, google)

Inheritance Hierarchy

- System.Object
- *Microsoft.AspNet.Identity.EntityFramework.IdentityUserLogin<TKey>*

Syntax

```
public class IdentityUserLogin<TKey> where TKey : IEquatable<TKey>
```

GitHub

View on GitHub

class Microsoft.AspNet.Identity.EntityFramework.**IdentityUserLogin<TKey>**

Properties

The login provider for the login (i.e. facebook, google)

LoginProvider()

Return type System.String

```
public virtual string LoginProvider { get; set; }
```

ProviderDisplayName()

Display name for the login

Return type System.String

```
public virtual string ProviderDisplayName { get; set; }
```

ProviderKey()

Key representing the login for the provider

Return type System.String

```
public virtual string ProviderKey { get; set; }
```

UserId()

User Id for the user who owns this login

Return type {TKey}

```
public virtual TKey UserId { get; set; }
```

1.3.10 IdentityUserRole<TKey> Class

Summary

EntityType that represents a user belonging to a role

Inheritance Hierarchy

- System.Object
- *Microsoft.AspNet.Identity.EntityFramework.IdentityUserRole<TKey>*

Syntax

```
public class IdentityUserRole<TKey> where TKey : IEquatable<TKey>
```

GitHub

View on GitHub

class Microsoft.AspNet.Identity.EntityFramework.**IdentityUserRole<TKey>**

Properties

RoleId for the role

RoleId()

Return type {TKey}

```
public virtual TKey RoleId { get; set; }
```

UserId()

UserId for the user that is in the role

Return type {TKey}

```
public virtual TKey UserId { get; set; }
```

1.3.11 IdentityUser<TKey> Class

Inheritance Hierarchy

- System.Object
- *Microsoft.AspNet.Identity.EntityFramework.IdentityUser<TKey>*

Syntax

```
public class IdentityUser<TKey> where TKey : IEquatable<TKey>
```

GitHub

View on GitHub

class Microsoft.AspNet.Identity.EntityFramework.**IdentityUser<TKey>**

Constructors

```
public IdentityUser()
```

IdentityUser()

IdentityUser(*System.String*)

```
public IdentityUser(string userName)
```

Properties

Used to record failures for the purposes of lockout

AccessFailedCount()

Return type
System.Int32

```
public virtual int AccessFailedCount { get; set; }
```

Claims()
Navigation property for users claims

Return type System.Collections.Generic.ICollection{Microsoft.AspNet.Identity.EntityFramework.IdentityUserClaim

```
public virtual ICollection<IdentityUserClaim<TKey>> Claims { get; }
```

ConcurrencyStamp()
A random value that should change whenever a user is persisted to the store

Return type System.String

```
public virtual string ConcurrencyStamp { get; set; }
```

Email()
Email

Return type System.String

```
public virtual string Email { get; set; }
```

EmailConfirmed()
True if the email is confirmed, default is false

Return type System.Boolean

```
public virtual bool EmailConfirmed { get; set; }
```

Id()

Return type {TKey}

```
public virtual TKey Id { get; set; }
```

LockoutEnabled()
Is lockout enabled for this user

Return type System.Boolean

```
public virtual bool LockoutEnabled { get; set; }
```

LockoutEnd()
DateTime in UTC when lockout ends, any time in the past is considered not locked out.

Return type System.Nullable{System.DateTimeOffset}

```
public virtual DateTimeOffset? LockoutEnd { get; set; }
```

Logins()
Navigation property for users logins

Return type System.Collections.Generic.ICollection{Microsoft.AspNet.Identity.EntityFramework.IdentityUserLogin

```
public virtual ICollection<IdentityUserLogin<TKey>> Logins { get; }
```

NormalizedEmail()

Return type System.String

```
public virtual string NormalizedEmail { get; set; }
```

NormalizedUserName()

Return type System.String

```
public virtual string NormalizedUserName { get; set; }
```

PasswordHash()
The salted/hashed form of the user password

Return type System.String

```
public virtual string PasswordHash { get; set; }
```

PhoneNumber()
PhoneNumber for the user

Return type System.String

```
public virtual string PhoneNumber { get; set; }
```

PhoneNumberConfirmed()
True if the phone number is confirmed, default is false

Return type System.Boolean

```
public virtual bool PhoneNumberConfirmed { get; set; }
```

Roles()
Navigation property for users in the role

Return type System.Collections.Generic.ICollection{Microsoft.AspNet.Identity.EntityFramework.IdentityUserRole{

```
public virtual ICollection<IdentityUserRole<TKey>> Roles { get; }
```

SecurityStamp()
A random value that should change whenever a users credentials change (password changed, login removed)

Return type System.String

```
public virtual string SecurityStamp { get; set; }
```

TwoFactorEnabled()
Is two factor enabled for the user

Return type System.Boolean

```
public virtual bool TwoFactorEnabled { get; set; }
```

UserName()

Return type System.String

```
public virtual string UserName { get; set; }
```

Methods

Returns a friendly name

ToString() **Return type**
System.String

```
public override string ToString()
```

1.3.12 RoleStore<TRole> Class

Inheritance Hierarchy

- System.Object
- Microsoft.AspNet.Identity.EntityFramework.RoleStore{{TRole},DbContext,System.String}
- *Microsoft.AspNet.Identity.EntityFramework.RoleStore<TRole>*

Syntax

```
public class RoleStore<TRole> : RoleStore<TRole, DbContext, string>, IQueryableRoleStore<TRole>, IRol
```

GitHub

View on GitHub

class Microsoft.AspNet.Identity.EntityFramework.**RoleStore<TRole>**

Constructors

RoleStore(*DbContext*, *Microsoft.AspNet.Identity.IdentityErrorDescriber*)

```
public RoleStore(DbContext context, IdentityErrorDescriber describer = null)
```

1.3.13 RoleStore<TRole, TContext> Class

Inheritance Hierarchy

- System.Object
- Microsoft.AspNet.Identity.EntityFramework.RoleStore{{TRole},{TContext},System.String}
- *Microsoft.AspNet.Identity.EntityFramework.RoleStore<TRole, TContext>*

Syntax

```
public class RoleStore<TRole, TContext> : RoleStore<TRole, TContext, string>, IQueryableRoleStore<TR
```

GitHub

View on GitHub

class Microsoft.AspNet.Identity.EntityFramework.**RoleStore<TRole, TContext>**

Constructors

RoleStore(*TContext*, *Microsoft.AspNet.Identity.IdentityErrorDescriber*)

```
public RoleStore(TContext context, IdentityErrorDescriber describer = null)
```

1.3.14 RoleStore<TRole, TContext, TKey> Class

Inheritance Hierarchy

- System.Object
- *Microsoft.AspNet.Identity.EntityFramework.RoleStore<TRole, TContext, TKey>*

Syntax

```
public class RoleStore<TRole, TContext, TKey> : IQueryableRoleStore<TRole>, IRoleClaimStore<TRole>,
```

GitHub

View on GitHub

class Microsoft.AspNet.Identity.EntityFramework.**RoleStore<TRole, TContext, TKey>**

Constructors

RoleStore(*TContext*, *Microsoft.AspNet.Identity.IdentityErrorDescriber*)

```
public RoleStore(TContext context, IdentityErrorDescriber describer = null)
```

Methods

AddClaimAsync(*TRole*, *System.Security.Claims.Claim*, *System.Threading.CancellationToken*) **Return type** System.Threading.Tasks.Task

```
public Task AddClaimAsync(TRole role, Claim claim, CancellationToken cancellationToken = nul
```

ConvertIdFromString(*System.String*)

Return type {TKey}

```
public virtual TKey ConvertIdFromString(string id)
```

ConvertIdToString(*TKey*)

Return type System.String

```
public virtual string ConvertIdToString(TKey id)
```

CreateAsync(*TRole*, *System.Threading.CancellationToken*)

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

```
public virtual Task<IdentityResult> CreateAsync(TRole role, CancellationToken cancellationTo
```

DeleteAsync(*TRole*, *System.Threading.CancellationToken*)

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

```
public virtual Task<IdentityResult> DeleteAsync(TRole role, CancellationToken cancellationTo
```

Dispose()

Dispose the store

```
public void Dispose()
```

FindByIdAsync(*System.String*, *System.Threading.CancellationToken*)

Find a role by id

Return type System.Threading.Tasks.Task{{TRole}}

```
public virtual Task<TRole> FindByIdAsync(string id, CancellationToken cancellationToken = nu
```

FindByNameAsync(*System.String*, *System.Threading.CancellationToken*)

Find a role by normalized name

Return type System.Threading.Tasks.Task{{TRole}}

```
public virtual Task<TRole> FindByNameAsync(string normalizedName, CancellationToken cancella
```

GetClaimsAsync(*TRole*, *System.Threading.CancellationToken*)

Return type System.Threading.Tasks.Task{System.Collections.Generic.IList{System.Security.Claims.Claim}}

```
public Task<IList<Claim>> GetClaimsAsync(TRole role, CancellationToken cancellationToken = n
```

GetNormalizedRoleNameAsync(*TRole*, *System.Threading.CancellationToken*)

Return type System.Threading.Tasks.Task{System.String}

```
public virtual Task<string> GetNormalizedRoleNameAsync(TRole role, CancellationToken cancell
```

GetRoleIdAsync(*TRole*, *System.Threading.CancellationToken*)

Return type System.Threading.Tasks.Task{System.String}

```
public Task<string> GetRoleIdAsync(TRole role, CancellationToken cancellationToken = null)
```

GetRoleNameAsync(*TRole*, *System.Threading.CancellationToken*)

Return type System.Threading.Tasks.Task{System.String}

```
public Task<string> GetRoleNameAsync(TRole role, CancellationToken cancellationToken = null)
```

RemoveClaimAsync(*TRole*, *System.Security.Claims.Claim*, *System.Threading.CancellationToken*)

Return type System.Threading.Tasks.Task

```
public Task RemoveClaimAsync(TRole role, Claim claim, CancellationToken cancellationToken =
```

SetNormalizedRoleNameAsync(*TRole*, *System.String*, *System.Threading.CancellationToken*)

Return type System.Threading.Tasks.Task

```
public virtual Task SetNormalizedRoleNameAsync(TRole role, string normalizedName, Cancellati
```

SetRoleNameAsync(*TRole*, *System.String*, *System.Threading.CancellationToken*)

Return type System.Threading.Tasks.Task

```
public Task SetRoleNameAsync(TRole role, string roleName, CancellationToken cancellationToke
```

UpdateAsync(*TRole*, *System.Threading.CancellationToken*)

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

```
public virtual Task<IdentityResult> UpdateAsync(TRole role, CancellationToken cancellationTo
```

Properties

If true will call SaveChanges after CreateAsync/UpdateAsync/DeleteAsync

AutoSaveChanges()

Return type System.Boolean

```
public bool AutoSaveChanges { get; set; }
```

Context()

Return type {TContext}

```
public TContext Context { get; }
```

ErrorDescriber()

Used to generate public API error messages

Return type Microsoft.AspNet.Identity.IdentityErrorDescriber

```
public IdentityErrorDescriber ErrorDescriber { get; set; }
```

Roles()

Return type System.Linq.IQueryable{{TRole}}

```
public virtual IQueryable<TRole> Roles { get; }
```

1.3.15 UserStore Class

- *Inheritance Hierarchy*
- *Syntax*
- *GitHub*
- *Constructors*

Inheritance Hierarchy

- System.Object
- Microsoft.AspNet.Identity.EntityFramework.UserStore{Microsoft.AspNet.Identity.EntityFr

- Microsoft.AspNet.Identity.EntityFramework.UserStore{Microsoft.AspNet.Identity.EntityFr
- Microsoft.AspNet.Identity.EntityFramework.UserStore{Microsoft.AspNet.Identity.EntityFr
- *Microsoft.AspNet.Identity.EntityFramework.UserStore*

Syntax

```
public class UserStore : UserStore<IdentityUser<string>>, IUserLoginStore<IdentityUser<string>>, IUse
```

GitHub

View on GitHub

class Microsoft.AspNet.Identity.EntityFramework.**UserStore**

Constructors

UserStore(*DbContext*, *Microsoft.AspNet.Identity.IdentityErrorDescriber*)

```
public UserStore(DbContext context, IdentityErrorDescriber describer = null)
```

1.3.16 UserStore<TUser> Class

Inheritance Hierarchy

- System.Object
- Microsoft.AspNet.Identity.EntityFramework.UserStore{{TUser},Microsoft.AspNet.Identity.
- Microsoft.AspNet.Identity.EntityFramework.UserStore{{TUser},Microsoft.AspNet.Identity.
- *Microsoft.AspNet.Identity.EntityFramework.UserStore<TUser>*

Syntax

```
public class UserStore<TUser> : UserStore<TUser, IdentityRole, DbContext>, IUserLoginStore<TUser>, IU
```

GitHub

View on GitHub

class Microsoft.AspNet.Identity.EntityFramework.**UserStore<TUser>**

Constructors

UserStore (*DbContext*, *Microsoft.AspNet.Identity.IdentityErrorDescriber*)

```
public UserStore(DbContext context, IdentityErrorDescriber describer = null)
```

1.3.17 UserStore<TUser, TRole, TContext> Class

Inheritance Hierarchy

- System.Object
- Microsoft.AspNet.Identity.EntityFramework.UserStore{{TUser},{TRole},{TContext},System.
- *Microsoft.AspNet.Identity.EntityFramework.UserStore<TUser, TRole, TContext>*

Syntax

```
public class UserStore<TUser, TRole, TContext> : UserStore<TUser, TRole, TContext, string>, IUserLogi
```

GitHub

View on GitHub

class Microsoft.AspNet.Identity.EntityFramework.**UserStore<TUser, TRole, TContext>**

Constructors

UserStore (*TContext*, *Microsoft.AspNet.Identity.IdentityErrorDescriber*)

```
public UserStore(TContext context, IdentityErrorDescriber describer = null)
```

1.3.18 UserStore<TUser, TRole, TContext, TKey> Class

Inheritance Hierarchy

- System.Object
- *Microsoft.AspNet.Identity.EntityFramework.UserStore<TUser, TRole, TContext, TKey>*

Syntax

```
public class UserStore<TUser, TRole, TContext, TKey> : IUserLoginStore<TUser>, IUserRoleStore<TUser>,
```

GitHub

View on GitHub

class Microsoft.AspNet.Identity.EntityFramework.**UserStore<TUser, TRole, TContext, TKey>**

Constructors

UserStore(*TContext*, *Microsoft.AspNet.Identity.IdentityErrorDescriber*)

```
public UserStore(TContext context, IdentityErrorDescriber describer = null)
```

Methods

AddClaimsAsync(*TUser*, *System.Collections.Generic.IEnumerable<System.Security.Claims.Claim>*, *System.Threading.CancellationToken*)

Return type System.Threading.Tasks.Task

```
public virtual Task AddClaimsAsync(TUser user, IEnumerable<Claim> claims, CancellationToken
```

AddLoginAsync(*TUser*, *Microsoft.AspNet.Identity.UserLoginInfo*, *System.Threading.CancellationToken*)

Return type System.Threading.Tasks.Task

```
public virtual Task AddLoginAsync(TUser user, UserLoginInfo login, CancellationToken cancell
```

AddToRoleAsync(*TUser*, *System.String*, *System.Threading.CancellationToken*)

Add a user to a role

Return type System.Threading.Tasks.Task

```
public virtual Task AddToRoleAsync(TUser user, string roleName, CancellationToken cancellati
```

ConvertIdFromString(*System.String*)

Return type {TKey}

```
public virtual TKey ConvertIdFromString(string id)
```

ConvertIdToString(*TKey*)

Return type System.String

```
public virtual string ConvertIdToString(TKey id)
```

CreateAsync(*TUser*, *System.Threading.CancellationToken*)

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

```
public virtual Task<IdentityResult> CreateAsync(TUser user, CancellationToken cancellationTo
```

DeleteAsync(*TUser*, *System.Threading.CancellationToken*)

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

```
public virtual Task<IdentityResult> DeleteAsync(TUser user, CancellationToken cancellationTo
```

Dispose()
Dispose the store

```
public void Dispose()
```

FindByEmailAsync(*System.String*, *System.Threading.CancellationToken*)
Find an user by email

Return type System.Threading.Tasks.Task{{TUser}}

```
public virtual Task<TUser> FindByEmailAsync(string normalizedEmail, CancellationToken cancel
```

FindByIdAsync(*System.String*, *System.Threading.CancellationToken*)
Find a user by id

Return type System.Threading.Tasks.Task{{TUser}}

```
public virtual Task<TUser> FindByIdAsync(string userId, CancellationToken cancellationToken
```

FindByLoginAsync(*System.String*, *System.String*, *System.Threading.CancellationToken*)

Return type System.Threading.Tasks.Task{{TUser}}

```
public virtual Task<TUser> FindByLoginAsync(string loginProvider, string providerKey, Cancel
```

FindByNameAsync(*System.String*, *System.Threading.CancellationToken*)
Find a user by normalized name

Return type System.Threading.Tasks.Task{{TUser}}

```
public virtual Task<TUser> FindByNameAsync(string normalizedUserName, CancellationToken canc
```

GetAccessFailedCountAsync(*TUser*, *System.Threading.CancellationToken*)
Returns the current number of failed access attempts. This number usually will be reset whenever the password is verified or the account is locked out.

Return type System.Threading.Tasks.Task{System.Int32}

```
public virtual Task<int> GetAccessFailedCountAsync(TUser user, CancellationToken cancellatic
```

GetClaimsAsync(*TUser*, *System.Threading.CancellationToken*)

Return type System.Threading.Tasks.Task{System.Collections.Generic.IList{System.Security.Claims.Claim}}

```
public virtual Task<IList<Claim>> GetClaimsAsync(TUser user, CancellationToken cancellationT
```

GetEmailAsync(*TUser*, *System.Threading.CancellationToken*)
Get the user's email

Return type System.Threading.Tasks.Task{System.String}

```
public virtual Task<string> GetEmailAsync(TUser user, CancellationToken cancellationToken =
```

GetEmailConfirmedAsync(*TUser*, *System.Threading.CancellationToken*)
Returns whether the user email is confirmed

Return type System.Threading.Tasks.Task{System.Boolean}

```
public virtual Task<bool> GetEmailConfirmedAsync(TUser user, CancellationToken cancellationT
```

GetLockoutEnabledAsync(*TUser*, *System.Threading.CancellationToken*)
Returns whether the user can be locked out.

Return type System.Threading.Tasks.Task{System.Boolean}

```
public virtual Task<bool> GetLockoutEnabledAsync(TUser user, CancellationToken cancellationT
```

GetLockoutEndDateAsync(*TUser*, *System.Threading.CancellationToken*)
Returns the DateTimeOffset that represents the end of a user's lockout, any time in the past should be considered not locked out.

Return type System.Threading.Tasks.Task{System.Nullable{System.DateTimeOffset}}

```
public virtual Task<DateTimeOffset? > GetLockoutEndDateAsync(TUser user, CancellationToken c
```

GetLoginsAsync(*TUser*, *System.Threading.CancellationToken*)

Return type System.Threading.Tasks.Task{System.Collections.Generic.IList{Microsoft.AspNet.Identity.UserLoginIn

```
public virtual Task<IList<UserLoginInfo>> GetLoginsAsync(TUser user, CancellationToken cance
```

GetNormalizedEmailAsync(*TUser*, *System.Threading.CancellationToken*)

Return type System.Threading.Tasks.Task{System.String}

```
public virtual Task<string> GetNormalizedEmailAsync(TUser user, CancellationToken cancellati
```

GetNormalizedUserNameAsync(*TUser*, *System.Threading.CancellationToken*)

Return type System.Threading.Tasks.Task{System.String}

```
public virtual Task<string> GetNormalizedUserNameAsync(TUser user, CancellationToken cancell
```

GetPasswordHashAsync(*TUser*, *System.Threading.CancellationToken*)
Get the password hash for a user

Return type System.Threading.Tasks.Task{System.String}

```
public virtual Task<string> GetPasswordHashAsync(TUser user, CancellationToken cancellationT
```

GetPhoneNumberAsync (*TUser*, *System.Threading.CancellationToken*)
Get a user's phone number

Return type System.Threading.Tasks.Task{System.String}

```
public virtual Task<string> GetPhoneNumberAsync(TUser user, CancellationToken cancellationTo
```

GetPhoneNumberConfirmedAsync (*TUser*, *System.Threading.CancellationToken*)
Returns whether the user phoneNumber is confirmed

Return type System.Threading.Tasks.Task{System.Boolean}

```
public virtual Task<bool> GetPhoneNumberConfirmedAsync(TUser user, CancellationToken cancell
```

GetRolesAsync (*TUser*, *System.Threading.CancellationToken*)
Get the names of the roles a user is a member of

Return type System.Threading.Tasks.Task{System.Collections.Generic.IList{System.String}}

```
public virtual Task<IList<string>> GetRolesAsync(TUser user, CancellationToken cancellationT
```

GetSecurityStampAsync (*TUser*, *System.Threading.CancellationToken*)
Get the security stamp for a user

Return type System.Threading.Tasks.Task{System.String}

```
public virtual Task<string> GetSecurityStampAsync(TUser user, CancellationToken cancellation
```

GetTwoFactorEnabledAsync (*TUser*, *System.Threading.CancellationToken*)
Gets whether two factor authentication is enabled for the user

Return type System.Threading.Tasks.Task{System.Boolean}

```
public virtual Task<bool> GetTwoFactorEnabledAsync(TUser user, CancellationToken cancellatic
```

GetUserIdAsync (*TUser*, *System.Threading.CancellationToken*)

Return type System.Threading.Tasks.Task{System.String}

```
public virtual Task<string> GetUserIdAsync(TUser user, CancellationToken cancellationToken =
```

GetUserNameAsync (*TUser*, *System.Threading.CancellationToken*)

Return type System.Threading.Tasks.Task{System.String}

```
public virtual Task<string> GetUserNameAsync(TUser user, CancellationToken cancellationToken
```

GetUsersForClaimAsync (*System.Security.Claims.Claim*, *System.Threading.CancellationToken*)
Get all users with given claim

Return type System.Threading.Tasks.Task{System.Collections.Generic.IList{{TUser}}}

```
public virtual Task<IList<TUser>> GetUsersForClaimAsync(Claim claim, CancellationToken cance
```

GetUsersInRoleAsync (*System.String*, *System.Threading.CancellationToken*)
Get all users in given role

Return type System.Threading.Tasks.Task{System.Collections.Generic.IList{{TUser}}}

```
public virtual Task<IList<TUser>> GetUsersInRoleAsync(string roleName, CancellationToken can
```

HasPasswordAsync (*TUser*, *System.Threading.CancellationToken*)
Returns true if the user has a password set

Return type System.Threading.Tasks.Task{System.Boolean}

```
public virtual Task<bool> HasPasswordAsync(TUser user, CancellationToken cancellationToken =
```

IncrementAccessFailedCountAsync(*TUser*, *System.Threading.CancellationToken*)
Used to record when an attempt to access the user has failed

Return type System.Threading.Tasks.Task{System.Int32}

```
public virtual Task<int> IncrementAccessFailedCountAsync(TUser user, CancellationToken cance
```

IsInRoleAsync(*TUser*, *System.String*, *System.Threading.CancellationToken*)
Returns true if the user is in the named role

Return type System.Threading.Tasks.Task{System.Boolean}

```
public virtual Task<bool> IsInRoleAsync(TUser user, string roleName, CancellationToken cance
```

RemoveClaimsAsync(*TUser*, *System.Collections.Generic.IEnumerable<System.Security.Claims.Claim>*, *System.Threading.CancellationToken*)

Return type System.Threading.Tasks.Task

```
public virtual Task RemoveClaimsAsync(TUser user, IEnumerable<Claim> claims, CancellationTok
```

RemoveFromRoleAsync(*TUser*, *System.String*, *System.Threading.CancellationToken*)
Remove a user from a role

Return type System.Threading.Tasks.Task

```
public virtual Task RemoveFromRoleAsync(TUser user, string roleName, CancellationToken cance
```

RemoveLoginAsync(*TUser*, *System.String*, *System.String*, *System.Threading.CancellationToken*)

Return type System.Threading.Tasks.Task

```
public virtual Task RemoveLoginAsync(TUser user, string loginProvider, string providerKey, C
```

ReplaceClaimAsync(*TUser*, *System.Security.Claims.Claim*, *System.Security.Claims.Claim*, *System.Threading.CancellationToken*)

Return type System.Threading.Tasks.Task

```
public virtual Task ReplaceClaimAsync(TUser user, Claim claim, Claim newClaim, CancellationT
```

ResetAccessFailedCountAsync(*TUser*, *System.Threading.CancellationToken*)
Used to reset the account access count, typically after the account is successfully accessed

Return type System.Threading.Tasks.Task

```
public virtual Task ResetAccessFailedCountAsync(TUser user, CancellationToken cancellationTo
```

SetEmailAsync(*TUser*, *System.String*, *System.Threading.CancellationToken*)
Set the user email

Return type System.Threading.Tasks.Task

```
public virtual Task SetEmailAsync(TUser user, string email, CancellationToken cancellationTo
```

SetEmailConfirmedAsync(*TUser*, *System.Boolean*, *System.Threading.CancellationToken*)
Set IsConfirmed on the user

Return type System.Threading.Tasks.Task

```
public virtual Task SetEmailConfirmedAsync(TUser user, bool confirmed, CancellationToken can
```

SetLockoutEnabledAsync(*TUser*, *System.Boolean*, *System.Threading.CancellationToken*)
Sets whether the user can be locked out.

Return type System.Threading.Tasks.Task

```
public virtual Task SetLockoutEnabledAsync(TUser user, bool enabled, CancellationToken cance
```

SetLockoutEndDateAsync(*TUser*, *System.Nullable<System.DateTimeOffset>*, *System.Threading.CancellationToken*)
Locks a user out until the specified end date (set to a past date, to unlock a user)

Return type System.Threading.Tasks.Task

```
public virtual Task SetLockoutEndDateAsync(TUser user, DateTimeOffset? lockoutEnd, Cancellat
```

SetNormalizedEmailAsync(*TUser*, *System.String*, *System.Threading.CancellationToken*)

Return type System.Threading.Tasks.Task

```
public virtual Task SetNormalizedEmailAsync(TUser user, string normalizedEmail, Cancellation
```

SetNormalizedUserNameAsync(*TUser*, *System.String*, *System.Threading.CancellationToken*)

Return type System.Threading.Tasks.Task

```
public virtual Task SetNormalizedUserNameAsync(TUser user, string normalizedName, Cancellati
```

SetPasswordHashAsync(*TUser*, *System.String*, *System.Threading.CancellationToken*)
Set the password hash for a user

Return type System.Threading.Tasks.Task

```
public virtual Task SetPasswordHashAsync(TUser user, string passwordHash, CancellationToken
```

SetPhoneNumberAsync(*TUser*, *System.String*, *System.Threading.CancellationToken*)
Set the user's phone number

Return type System.Threading.Tasks.Task

```
public virtual Task SetPhoneNumberAsync(TUser user, string phoneNumber, CancellationToken ca
```

SetPhoneNumberConfirmedAsync(*TUser*, *System.Boolean*, *System.Threading.CancellationToken*)
Set PhoneNumberConfirmed on the user

Return type System.Threading.Tasks.Task

```
public virtual Task SetPhoneNumberConfirmedAsync(TUser user, bool confirmed, CancellationTok
```

SetSecurityStampAsync(*TUser*, *System.String*, *System.Threading.CancellationToken*)
Set the security stamp for the user

Return type System.Threading.Tasks.Task

```
public virtual Task SetSecurityStampAsync(TUser user, string stamp, CancellationToken cancel
```

SetTwoFactorEnabledAsync(*TUser*, *System.Boolean*, *System.Threading.CancellationToken*)
Set whether two factor authentication is enabled for the user

Return type System.Threading.Tasks.Task

```
public virtual Task SetTwoFactorEnabledAsync(TUser user, bool enabled, CancellationToken can
```

SetUserNameAsync(*TUser*, *System.String*, *System.Threading.CancellationToken*)

Return type System.Threading.Tasks.Task

```
public virtual Task SetUserNameAsync(TUser user, string userName, CancellationToken cancella
```

UpdateAsync(*TUser*, *System.Threading.CancellationToken*)

Return type System.Threading.Tasks.Task{Microsoft.AspNet.Identity.IdentityResult}

```
public virtual Task<IdentityResult> UpdateAsync(TUser user, CancellationToken cancellationTo
```

Properties

If true will call SaveChanges after CreateAsync/UpdateAsync/DeleteAsync

AutoSaveChanges()

Return type System.Boolean

```
public bool AutoSaveChanges { get; set; }
```

Context()

Return type {TContext}

```
public TContext Context { get; }
```

ErrorDescriber()

Used to generate public API error messages

Return type Microsoft.AspNet.Identity.IdentityErrorDescriber

```
public IdentityErrorDescriber ErrorDescriber { get; set; }
```

Users()

Return type System.Linq.IQueryable{{TUser}}

```
public virtual IQueryable<TUser> Users { get; }
```

namespace Microsoft.AspNet.Identity.**EntityFramework**

Classes

class *Microsoft.AspNet.Identity.EntityFramework.IdentityDbContext*

class *Microsoft.AspNet.Identity.EntityFramework.IdentityDbContext<TUser>*

class *Microsoft.AspNet.Identity.EntityFramework.IdentityDbContext<TUser, TRole, TKey>*

class *Microsoft.AspNet.Identity.EntityFramework.IdentityRole* Represents a Role entity

class *Microsoft.AspNet.Identity.EntityFramework.IdentityRoleClaim<TKey>* EntityType that represents one specific role claim

class *Microsoft.AspNet.Identity.EntityFramework.IdentityRole<TKey>* Represents a Role entity

class *Microsoft.AspNet.Identity.EntityFramework.IdentityUser*

class *Microsoft.AspNet.Identity.EntityFramework.IdentityUserClaim<TKey>*
EntityType that represents one specific user claim

class *Microsoft.AspNet.Identity.EntityFramework.IdentityUserLogin<TKey>* Entity type for a user's login (i.e. facebook, google)

class *Microsoft.AspNet.Identity.EntityFramework.IdentityUserRole<TKey>*
EntityType that represents a user belonging to a role

class *Microsoft.AspNet.Identity.EntityFramework.IdentityUser<TKey>*

class *Microsoft.AspNet.Identity.EntityFramework.RoleStore<TRole>*

class *Microsoft.AspNet.Identity.EntityFramework.RoleStore<TRole, TContext>*

class *Microsoft.AspNet.Identity.EntityFramework.RoleStore<TRole, TContext, TKey>*

class *Microsoft.AspNet.Identity.EntityFramework.UserStore*

class *Microsoft.AspNet.Identity.EntityFramework.UserStore<TUser>*

class *Microsoft.AspNet.Identity.EntityFramework.UserStore<TUser, TRole, TContext>*

class *Microsoft.AspNet.Identity.EntityFramework.UserStore<TUser, TRole, TContext, TKey>*

1.4 Microsoft.Framework.DependencyInjection Namespace

1.4.1 IdentityEntityFrameworkBuilderExtensions Class

Inheritance Hierarchy

- `System.Object`
- *Microsoft.Framework.DependencyInjection.IdentityEntityFrameworkBuilderExtensions*

Syntax

```
public class IdentityEntityFrameworkBuilderExtensions
```

GitHub

View on GitHub

class `Microsoft.Framework.DependencyInjection.`**`IdentityEntityFrameworkBuilderExtensions`**

Methods

AddEntityFrameworkStores<TContext> (*Microsoft.AspNet.Identity.IdentityBuilder*) **Return type**
Microsoft.AspNet.Identity.IdentityBuilder

```
public static IdentityBuilder AddEntityFrameworkStores<TContext>(IdentityBuilder builder)whe
```

AddEntityFrameworkStores<TContext, TKey> (*Microsoft.AspNet.Identity.IdentityBuilder*)

Return type Microsoft.AspNet.Identity.IdentityBuilder

```
public static IdentityBuilder AddEntityFrameworkStores<TContext, TKey>(IdentityBuilder build
```

1.4.2 IdentityServiceCollectionExtensions Class

Summary

Contains extension methods to IServiceCollection for configuring identity services.

Inheritance Hierarchy

- `System.Object`
- *`Microsoft.Framework.DependencyInjection.IdentityServiceCollectionExtensions`*

Syntax

```
public class IdentityServiceCollectionExtensions
```

GitHub

View on GitHub

class `Microsoft.Framework.DependencyInjection.`**`IdentityServiceCollectionExtensions`**

Methods

Adds the default identity system configuration for the specified User and Role types.

AddIdentity<TUser, TRole> (*IServiceCollection*) **Arguments**

- **services** (*IServiceCollection*) – The services available in the application.

Return type Microsoft.AspNet.Identity.IdentityBuilder

Returns An <see cref="T:Microsoft.AspNet.Identity.IdentityBuilder" /> for creating and configuring the identity system.

```
public static IdentityBuilder AddIdentity<TUser, TRole>(IServiceCollection services)where TU
```

AddIdentity<TUser, TRole> (*IServiceCollection*, *System.Action<Microsoft.AspNet.Identity.IdentityOptions>*)
Adds and configures the identity system for the specified User and Role types.

Arguments

- **services** (*IServiceCollection*) – The services available in the application.
- **setupAction** (*System.Action{Microsoft.AspNet.Identity.IdentityOptions}*) – An action to configure the Microsoft.AspNet.Identity.IdentityOptions.

Return type Microsoft.AspNet.Identity.IdentityBuilder

Returns An <see cref="T:Microsoft.AspNet.Identity.IdentityBuilder" /> for creating and configuring the identity system.

```
public static IdentityBuilder AddIdentity<TUser, TRole>(IServiceCollection services, Action<
```

ConfigureIdentity (*IServiceCollection*, *IConfiguration*)
Configures a set of Microsoft.AspNet.Identity.IdentityOptions for the application

Arguments

- **services** (*IServiceCollection*) – The services available in the application.
- **config** (*IConfiguration*) – The configuration for the Microsoft.AspNet.Identity.IdentityOptions.

Return type IServiceCollection

Returns The <see cref="!:IServiceCollection" /> instance this method extends.

```
public static IServiceCollection ConfigureIdentity(IServiceCollection services, IConfigurati
```

ConfigureIdentity (*IServiceCollection*, *System.Action<Microsoft.AspNet.Identity.IdentityOptions>*)
Configures a set of Microsoft.AspNet.Identity.IdentityOptions for the application

Arguments

- **services** (*IServiceCollection*) – The services available in the application.
- **setupAction** (*System.Action{Microsoft.AspNet.Identity.IdentityOptions}*) – An action to configure the Microsoft.AspNet.Identity.IdentityOptions.

Return type IServiceCollection

Returns The <see cref="!:IServiceCollection" /> instance this method extends.

```
public static IServiceCollection ConfigureIdentity(IServiceCollection services, Action<Ident
```

ConfigureIdentityApplicationCookie (*IServiceCollection*, *System.Action<CookieAuthenticationOptions>*)
Configures a set of CookieAuthenticationOptions for the application

Arguments

- **services** (*IServiceCollection*) – The services available in the application.
- **setupAction** (*System.Action{CookieAuthenticationOptions}*) – An action to configure the CookieAuthenticationOptions.

Return type IServiceCollection

Returns The <see cref="!:IServiceCollection" /> instance this method extends.

```
public static IServiceCollection ConfigureIdentityApplicationCookie(IServiceCollection servi
```

namespace Microsoft.Framework.**DependencyInjection**

Classes

class *Microsoft.Framework.DependencyInjection.IdentityEntityFrameworkBuilderExtensions*

class ***Microsoft.Framework.DependencyInjection.IdentityServiceCollectionExtensions***
Contains extension methods to IServiceCollection for configuring identity services.

1.5 System.Security.Claims Namespace

1.5.1 PrincipalExtensions Class

Summary

Claims related extensions for System.Security.Claims.ClaimsPrincipal.

Inheritance Hierarchy

- System.Object
- *System.Security.Claims.PrincipalExtensions*

Syntax

```
public class PrincipalExtensions
```

GitHub

View on GitHub

class System.Security.Claims.**PrincipalExtensions**

Methods

Returns the value for the first claim of the specified type otherwise null the claim is not present.

FindFirstValue (*System.Security.Claims.ClaimsPrincipal*, *System.String*)

Arguments

- **principal** (*System.Security.Claims.ClaimsPrincipal*) – The System.Security.Claims.ClaimsPrincipal instance this method extends.
- **claimType** (*System.String*) – The claim type whose first value should be returned.

Return type System.String

Returns The value of the first instance of the specified claim type, or null if the claim is not present.

```
public static string FindFirstValue(ClaimsPrincipal principal, string claimType)
```

GetUserId (*System.Security.Claims.ClaimsPrincipal*)

Returns the User ID claim value if present otherwise returns null.

Arguments

- **principal** (*System.Security.Claims.ClaimsPrincipal*) – The System.Security.Claims.ClaimsPrincipal instance this method extends.

Return type System.String

Returns The User ID claim value, or null if the claim is not present.

```
public static string GetUserId(ClaimsPrincipal principal)
```

GetUserName (*System.Security.Claims.ClaimsPrincipal*)

Returns the Name claim value if present otherwise returns null.

Arguments

- **principal** (*System.Security.Claims.ClaimsPrincipal*) – The System.Security.Claims.ClaimsPrincipal instance this method extends.

Return type System.String

Returns The Name claim value, or null if the claim is not present.

```
public static string GetUserName(ClaimsPrincipal principal)
```

IsSignedIn (*System.Security.Claims.ClaimsPrincipal*)

Returns true if the principal has an identity with the application cookie identity

Arguments

- **principal** (*System.Security.Claims.ClaimsPrincipal*) – The System.Security.Claims.ClaimsPrincipal instance this method extends.

Return type System.Boolean

Returns True if the user is logged in with identity.

```
public static bool IsSignedIn(ClaimsPrincipal principal)
```

namespace System.Security.**Claims**

Classes

class ***System.Security.Claims.PrincipalExtensions*** Claims related extensions for System.Security.Claims.ClaimsPrincipal.

N