Implementing Secure User Management

Scott Brady
IDENTITY & ACCESS CONTROL LEAD

@scottbrady91 www.scottbrady91.com

Overview

Simplifying basic user authentication logic

Improving the user experience with password management functionality

Protecting user data with email confirmation

ASP.NET Core Identity token handlers in-depth

Demo

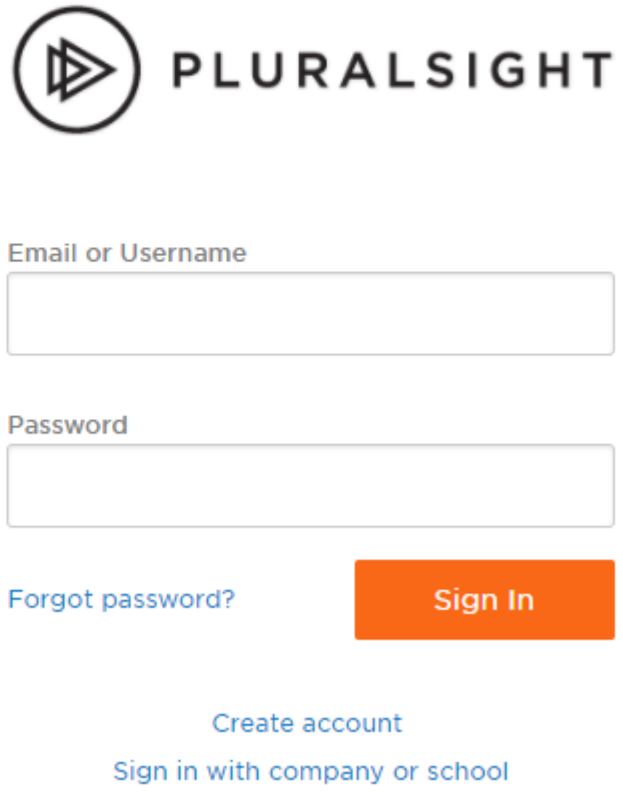
Simplifying claims generation using UserClaimsPrincipalFactory

Secure account management using SignInManager

SignInManager vs. UserManager

Password Reset

Password Reset



PLURALSIGHT

Email or Username

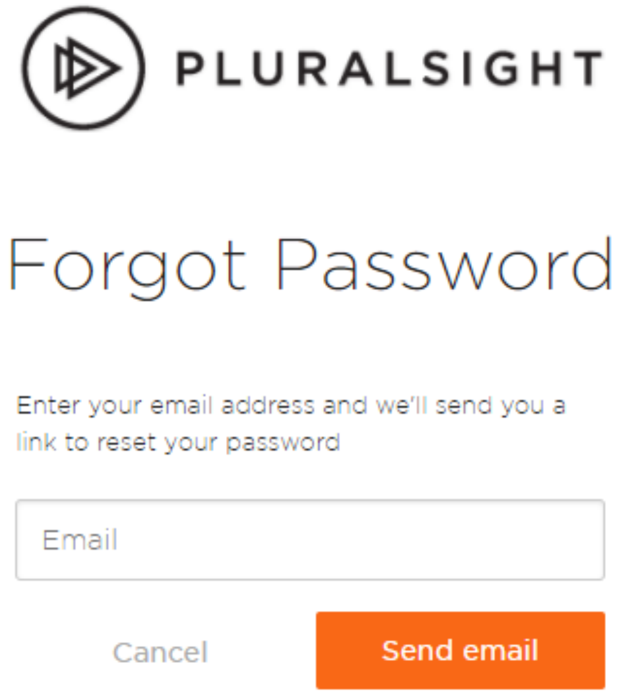
Password

Forgot password?

Sign In

Create account

Sign in with company or school



PLURALSIGHT

Forgot Password

Enter your email address and we'll send you a link to reset your password

Email

Cancel

Send email

Password Reset Link

Only for that user

Time sensitive

Random

One time use

UserManager Reset Password Token

```
Task<string> GeneratePasswordResetTokenAsync(user)
```

UserManager Reset Password Token

```
Task<string> GenerateUserTokenAsync(
    user,
    Options.Tokens.PasswordResetTokenProvider, // “Default”
    ResetPasswordTokenPurpose) // “RestPassword”
```

UserManager Reset Password Token

```
tokenProviders[tokenProvider].GenerateAsync(purpose, this,
user);
```

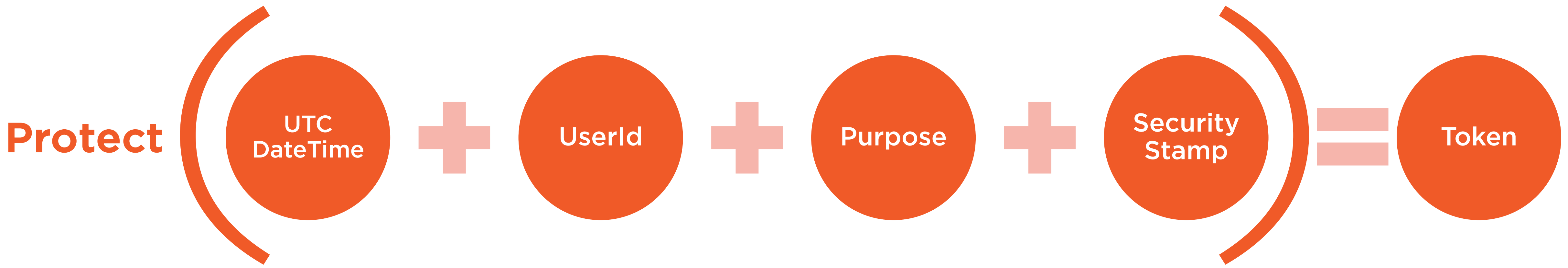
```
Task<string> GenerateAsync(
    string purpose, UserManager<TUser> manager, TUser user);

Task<bool> ValidateAsync(
    string purpose, string token, UserManager<TUser> manager, TUser user);

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

IUserTwoFactorTokenProvider<TUser>

CanGenerateTwoFactorTokenAsync will return false in this case

Purpose differentiates token types

DataProtectorTokenProvider



UserManager Reset Password Validation

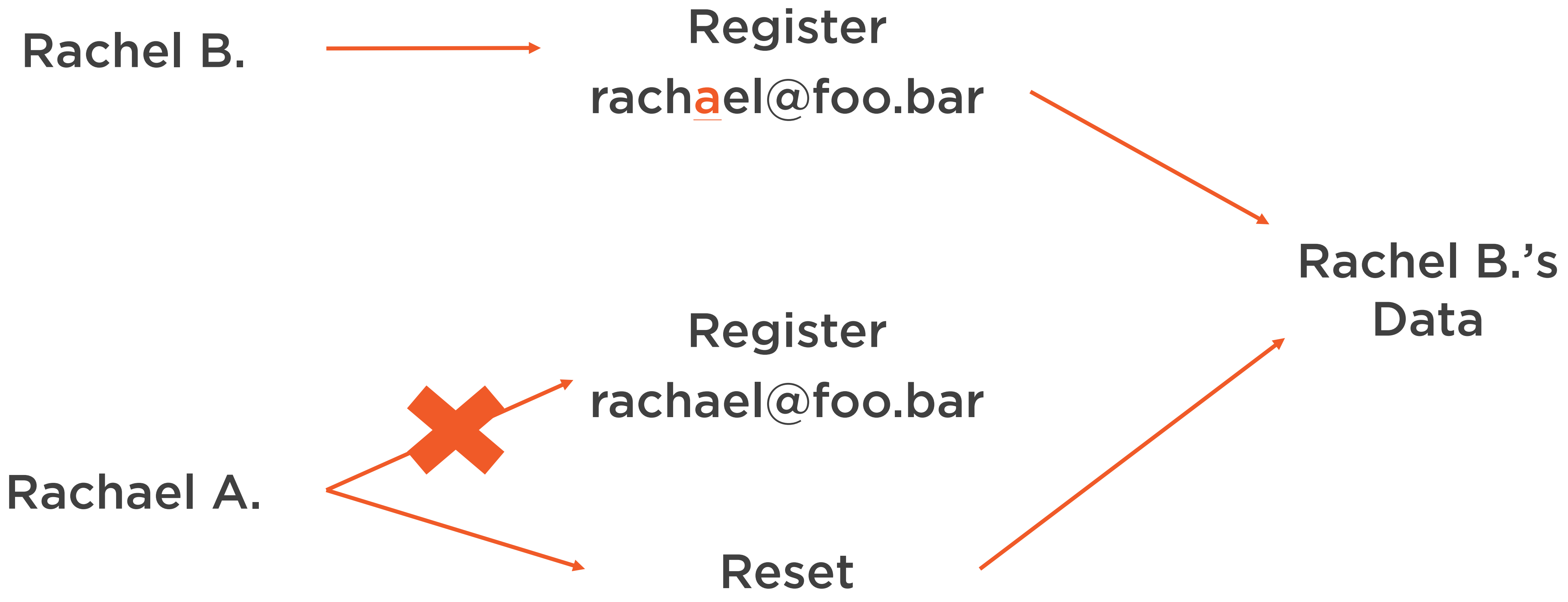
```
Task<IdentityResult> ResetPasswordAsync(
    TUser user, string token, string newPassword)


Task<IdentityResult> VerifyUserTokenAsync(
    user,
    Options.Tokens.PasswordResetTokenProvider,
    ResetPasswordTokenPurpose,
    token)
```

(creationTime + Options.TokenLifespan) < DateTime.UtcNow

Demo

Secure password reset using a default token provider

Email Confirmation


Rachel B.
Register
rachael@foo.bar
Rachel B.'s
Data
Register
rachael@foo.bar
Rachael A.
Reset

```
Task<string> GenerateEmailConfirmationTokenAsync(TUser user)

Task<string> GenerateUserTokenAsync(
    user,
    Options.Tokens. EmailConfirmationTokenProvider, // “Default”
    ConfirmEmailTokenPurpose) // “EmailConfirmation”
```

Email Confirmation

By default uses the same token provider configuration as password reset

Token Providers

```
foreach (var providerName in Options.Tokens.ProviderMap.Keys) {

    var description = Options.Tokens.ProviderMap[providerName];

    var provider = (description.ProviderInstance
        ?? services.GetRequiredService(description.ProviderType))
        as IUserTwoFactorTokenProvider<TUser>;

    if (provider != null) {

        RegisterTokenProvider(providerName, provider);

    }

}
```

Demo

Confirming email addresses

Registering a custom token provider for email confirmation

```
userManager.ChangePasswordAsync(
    user,
    “old/current password",
    “new password")
```

Password Update

Allow authenticated user to update their password

Requires existing password

Help! My tokens keep invalidating!

docs.microsoft.com/aspnet/core/security/data-protection

Summary

Simplifying basic user authentication logic

Improving the user experience with password management functionality

Protecting user data with email confirmation

ASP.NET Core Identity token handlers in-depth