Advanced ASP.NET Identity

Brock Allen
brockallen@gmail.com
http://brockallen.com
@BrockLAllen
Advanced The complicated bits of ASP.NET Identity

Brock Allen
brockallen@gmail.com
http://brockallen.com
@BrockLAllen
Overview

• Password reset
• Two-factor authentication
• External logins
• “Unit” testing
Password reset

• Forgot password: generate token and email to user
  • GeneratePasswordResetTokenAsync

• Reset password: prompt for new password
  • ResetPasswordAsync

• Requires token providers in DI
Email confirmation

- Might be useful to know if email is confirmed
  - Can check with IsEmailConfirmedAsync

- Send token to email at registration (or at successful login)
  - GenerateEmailConfirmationTokenAsync used to generate token

- User confirms token
  - ConfirmEmailAsync used to validate token
Requiring confirmed email

• Email confirmation doesn’t guarantee it’s the right email for the user
  • e.g. typos in email
  • Don’t want email owner to reset password and take over account

• Can solve by requiring confirmation for password reset
  • Can orphan account if never confirmed and password forgotten

• Can solve by requiring confirmation to login
  • RequireConfirmedEmail on options to enforce if using SigninManager
  • Can orphan account if never confirmed and password forgotten

• Can solve by requiring password on confirmation page
  • Can orphan account if password forgotten
Redesigning registration for confirmation

• Consider different registration flow
  • Registration page only asks for email
  • Confirm page then collects other registration info (password, et al.)

• Registration is simple, and email confirmation is built-in
Changing email

• Must confirm new email before switching
  • `GenerateChangeEmailTokenAsync` generates token for new email
  • Should track new email before overwriting old email

• Confirm token
  • `ChangeEmailAsync` will validate token and set new email
  • Might also need to keep username in sync with `SetUserNameAsync`

• Require authenticated user to ensure correct user
Two-factor authentication

• Current design targets SMS or email
  • SMS is considered weak (and so is email) but still better than nothing
  • Improvements will come in the future, but you can do your own in the interim

• Given what we have...
  • User must set phone with system
  • Must confirm phone prior to using for two-factor
  • User must configure account to use two-factor
  • Enforce two-factor on login
Assigning a phone number

• Generate code and send to user
  • GenerateChangePhoneNumberTokenAsync

• Use confirms code
  • ChangePhoneNumberAsync

• Require authenticated user to ensure correct user
Enable account for two-factor

• Configure user for two-factor
  • SetTwoFactorEnabledAsync

• Enforce two-factor at login
  • GetTwoFactorEnabledAsync

• SignInManager will check as well if using for sign-in
  • Provides flag on SignInResult
Use two-factor at sign-in

• Generate code to send to user
  • GenerateTwoFactorTokenAsync to create code
  • Use TokenOptions.DefaultPhoneProvider for SMS provider

• Temporary two-factor sign-in cookie will be created for user
  • Ensures the code is matched to the correct user
  • Created by SignInManager in PasswordSignInAsync

• TwoFactorSignInAsync on SignInManager to complete sign-in
  • GetTwoFactorAuthenticationUserAsync to query user
Remembering two-factor

• Users won’t want to use two-factor on every login
  • Can use a persistent cookie to track two-factor was used on device
  • TwoFactorSignInAsync provides flag to issue cookie

• Use IsTwoFactorClientRememberedAsync on login to bypass
  • Done by SigninManager in PasswordSignInAsync
Removing phone

• If user removes phone, must disable two-factor
  • SetPhoneNumberAsync passing null
  • SetTwoFactorEnabledAsync passing false
  • ForgetTwoFactorClientAsync to clear remember two-factor on device
Lost phone

• Recovery codes allow for bypassing 2FA in case of lost device
• Will be added in next release
  • User requests codes
    • GenerateNewTwoFactorRecoveryCodesAsync
  • User can use code to as alternative
    • RedeemTwoFactorRecoveryCodeAsync
Linked accounts

• User want to use an external authentication provider
  • Instead of a local password

• Must send user to external login page
  • ConfigureExternalAuthenticationProperties and Challenge

• External middleware does protocol processing and tracks user with external cookie (not primary authentication cookie)
  • GetExternalLoginInfoAsync
External callback processing

• Callback code need to locate matching user from external login
  • ExternalLoginSignInAsync

• Must decide how to proceed for no matching user (register or deny)
  • External provider claims useful to pre-populate registration page
  • AddLoginAsync used to associate external login with new account

• ExternalLoginSignInAsync is close to worthless
  • Checks if email confirmed (email only necessary for password resets)
  • Checks for two-factor (can be disabled)
  • Can use FindByLoginAsync/SignInAsync instead
How to associate external to existing users

• At registration time, if email already exists consider prompting user for password
  • Then use AddLoginAsync

• For logged in users, can proactively allow adding/removing
  • To add: trigger Challenge with different callback processing
  • To remove: simply use RemoveLoginAsync (beware last one)

• Perhaps allow password to be added
  • AddPasswordAsync
  • But then email confirmation is also needed
Design and unit testing

• Likely will want to encapsulate UserManager calls into custom class
  • Given the various workflows and numerous APIs
• Unit testing is then desirable, but difficult given UserManager APIs
  • Essentially need to setup a DI container to be able to unit test
Summary

• Account management is complicated
• Don’t make any assumptions
• Understand your identity framework
• Review your requirements
• Consider centralizing identity