ASP.NET Identity reset password

86

How can I get the password of a user in the new ASP.NET Identity system? Or how can I reset without knowing the current one (user forgot password)?

asp.net asp.net-identity

★ 35

asked Oct 22 '13 at 17:06

daniel
10.2k 32 79 134

8 Answers

94 ✓

In current release

Assuming you have handled the verification of the request to reset the forgotten password, use following code as a sample code steps.

```
ApplicationDbContext =new ApplicationDbContext()
String userId = "<YourLogicAssignsRequestedUserId>";
String newPassword = "<PasswordAsTypedByUser>";
ApplicationUser cUser = UserManager.FindById(userId);
String hashedNewPassword = UserManager.PasswordHasher.HashPassword(newPassword);
UserStore<ApplicationUser> store = new UserStore<ApplicationUser>();
store.SetPasswordHashAsync(cUser, hashedNewPassword);
```

In AspNet Nightly Build

The framework is updated to work with Token for handling requests like ForgetPassword. Once in release, simple code guidance is expected.

Update:

```
ApplicationDbContext context = new ApplicationDbContext();
UserStore<ApplicationUser> store = new UserStore<ApplicationUser>(context);
UserManager<ApplicationUser> UserManager = new UserManager<ApplicationUser>(store);
String userId = User.Identity.GetUserId();//"<YourLogicAssignsRequestedUserId>";
String newPassword = "test@123"; //"<PasswordAsTypedByUser>";
String hashedNewPassword = UserManager.PasswordHasher.HashPassword(newPassword);
ApplicationUser cUser = await store.FindByIdAsync(userId);
await store.SetPasswordHashAsync(cUser, hashedNewPassword);
await store.UpdateAsync(cUser);
```

edited Oct 27 '13 at 7:17

answered Oct 22 '13 at 19:29
jd4u
5,234 1 22 28

do you know when version 1.1 will be released? – graycrow Oct 23 '13 at 9:11

Its still in alpha, and 1.0 is just released. So assume many months. myget.org/gallery/aspnetwebstacknightly – jd4u Oct 23 '13 at 11:37

11 Oddly the store.SetPasswordHashAsync(cUser, hashedNewPassword) method call did not work for me, instead I had to manually set cUser.PasswordHash = hashedNewPassword and then call UserManager.UpdateAsync(user); – Andy Mehalick Oct 26 '13 at 18:25

1 Code not working is only possible if the User retrieval context and store context are different. The code was only a sample steps, not accurate. Will soon update the answer to avoid this issue for others. – jd4u Oct 27 '13 at 7:14

1 Framework 1 does not provide. But Framework 2-alpha does have few features that can provide simple process for handling password reset requests. aspnetidentity.codeplex.com – jd4u Mar 18 '14 at 19:01

0

I think Microsoft guide for ASP.NET Identity is a good start.

https://docs.microsoft.com/en-us/aspnet/identity/overview/features-api/account-confirmation-and-password-recovery-with-aspnet-identity

Note:

If you do not use AccountController and wan't to reset your password, use `Request.GetOwinContext().GetUserManager<ApplicationUserManager>();` . If you dont have the same OwinContext you need to create a new

Could be created like this:

Without Owin:

```
[HttpGet]
[AllowAnonymous]
[Route("testReset")]
public IHttpActionResult TestReset()
{
    var db = new ApplicationDbContext();
    var manager = new ApplicationUserManager(new UserStore<ApplicationUser>(db));
    var provider = new DpapiDataProtectionProvider("SampleAppName");
    manager.UserTokenProvider = new DataProtectorTokenProvider<ApplicationUser>(
        provider.Create("SampleTokenName"));

    var email = "test@test.com";

    var user = new ApplicationUser() { UserName = email, Email = email };

    var identityUser = manager.FindByEmail(email);

    if (identityUser == null)
    {
        manager.Create(user);
        identityUser = manager.FindByEmail(email);
    }

    var token = manager.GeneratePasswordResetToken(identityUser.Id);
    return Ok(HttpUtility.UrlEncode(token));
}

[HttpGet]
[AllowAnonymous]
[Route("testReset")]
public IHttpActionResult TestReset(string token)
{
    var db = new ApplicationDbContext();
    var manager = new ApplicationUserManager(new UserStore<ApplicationUser>(db));
    var provider = new DpapiDataProtectionProvider("SampleAppName");
    manager.UserTokenProvider = new DataProtectorTokenProvider<ApplicationUser>(
        provider.Create("SampleTokenName"));
    var email = "test@test.com";
    var identityUser = manager.FindByEmail(email);
    var valid = Task.Run(() => manager.UserTokenProvider.ValidateAsync("ResetPassword",
token, manager, identityUser)).Result;
```

With Owin:

```
[HttpGet]
[AllowAnonymous]
[Route("testResetWithOwin")]
public IHttpActionResult TestResetWithOwin()
{
    var manager = Request.GetOwinContext().GetUserManager<ApplicationUserManager>();

    var email = "test@test.com";

    var user = new ApplicationUser() { UserName = email, Email = email };

    var identityUser = manager.FindByEmail(email);

    if (identityUser == null)
    {
        manager.Create(user);
        identityUser = manager.FindByEmail(email);
    }

    var token = manager.GeneratePasswordResetToken(identityUser.Id);
    return Ok(HttpUtility.UrlEncode(token));
}

[HttpGet]
[AllowAnonymous]
[Route("testResetWithOwin")]
public IHttpActionResult TestResetWithOwin(string token)
{
    var manager = Request.GetOwinContext().GetUserManager<ApplicationUserManager>();

    var email = "test@test.com";
    var identityUser = manager.FindByEmail(email);
    var valid = Task.Run(() => manager.UserTokenProvider.ValidateAsync("ResetPassword",
token, manager, identityUser)).Result;
    var result = manager.ResetPassword(identityUser.Id, token, "TestingTest1!");
    return Ok(result);
}
```

The `DpapiDataProtectionProvider` and `DataProtectorTokenProvider` needs to be created with the same name for a password reset to work. Using Owin for creating the password reset token and then creating a new `DpapiDataProtectionProvider` with another name won't work.

Web.Config:

```
<add key="AllowedHosts" value="example.com,example2.com" />
```

AccountController.cs:

```
[Route("RequestResetPasswordToken/{email}/")]
[HttpGet]
[AllowAnonymous]
public async Task<IHttpActionResult> GetResetPasswordToken([FromUri]string email)
{
    if (!ModelState.IsValid)
        return BadRequest(ModelState);

    var user = await UserManager.FindByEmailAsync(email);
    if (user == null)
    {
        Logger.Warn("Password reset token requested for non existing email");
        // Don't reveal that the user does not exist
        return NoContent();
    }

    //Prevent Host Header Attack -> Password Reset Poisoning.
    //If the IIS has a binding to accept connections on 80/443 the host parameter can be
changed.
    //See https://security.stackexchange.com/a/170759/67046
    if
(!ConfigurationManager.AppSettings["AllowedHosts"].Split(',').Contains(Request.RequestUri.
 {
            Logger.Warn($"Non allowed host detected for password reset
{Request.RequestUri.Scheme}://{Request.Headers.Host}");
            return BadRequest();
    }

    Logger.Info("Creating password reset token for user id {0}", user.Id);

    var host = $"{Request.RequestUri.Scheme}://{Request.Headers.Host}";
    var token = await UserManager.GeneratePasswordResetTokenAsync(user.Id);
    var callbackUrl = $"
{host}/resetPassword/{HttpContext.Current.Server.UrlEncode(user.Email)}/{HttpContext.Curre


    var subject = "Client - Password reset.";
```

```
link to reset your password </a></p>" +
               "</body></html>";

    var message = new IdentityMessage
    {
        Body = body,
        Destination = user.Email,
        Subject = subject
    };

    await UserManager.EmailService.SendAsync(message);

    return NoContent();
}

[HttpPost]
[Route("ResetPassword/")]
[AllowAnonymous]
public async Task<IHttpActionResult> ResetPasswordAsync(ResetPasswordRequestModel model)
{
    if (!ModelState.IsValid)
        return NoContent();

    var user = await UserManager.FindByEmailAsync(model.Email);
    if (user == null)
    {
        Logger.Warn("Reset password request for non existing email");
        return NoContent();
    }

    if (!await UserManager.UserTokenProvider.ValidateAsync("ResetPassword", model.Token,
UserManager, user))
    {
        Logger.Warn("Reset password requested with wrong token");
        return NoContent();
    }

    var result = await UserManager.ResetPasswordAsync(user.Id, model.Token,
model.NewPassword);

    if (result.Succeeded)
    {
        Logger.Info("Creating password reset token for user id {0}", user.Id);

        const string subject = "Client - Password reset success.";
        var body = "<html><body>" +
not request this change.</p>" +
                   "</body></html>";

        var message = new IdentityMessage
        {
            Body = body,
            Destination = user.Email,
            Subject = subject
        };

        await UserManager.EmailService.SendAsync(message);
    }

    return NoContent();
}

public class ResetPasswordRequestModel
{
    [Required]
    [Display(Name = "Token")]
    public string Token { get; set; }

    [Required]
    [Display(Name = "Email")]
    public string Email { get; set; }

    [Required]
    [StringLength(100, ErrorMessage = "The {0} must be at least {2} characters long.",
MinimumLength = 10)]
    [DataType(DataType.Password)]
    [Display(Name = "New password")]
    public string NewPassword { get; set; }

    [DataType(DataType.Password)]
    [Display(Name = "Confirm new password")]
    [Compare("NewPassword", ErrorMessage = "The new password and confirmation password
do not match.")]
    public string ConfirmPassword { get; set; }
}
```

edited Sep 20 '18 at 15:33

answered Sep 20 '18 at 7:18
Ogglas

1

In case of password reset, it is recommended to reset it through sending password reset token to registered user email and ask user to provide new password. If have created a easily usable .NET library over Identity framework with default configuration settins. You can find details at blog link and source code at github.

edited Aug 9 '16 at 9:58

answered Aug 5 '16 at 19:11

Rahul Garg
2,258 20 26

2

Create method in `UserManager<TUser, TKey>`

```
public Task<IdentityResult> ChangePassword(int userId, string newPassword)
{
     var user = Users.FirstOrDefault(u => u.Id == userId);
     if (user == null)
          return new Task<IdentityResult>(() => IdentityResult.Failed());

     var store = Store as IUserPasswordStore<User, int>;
     return base.UpdatePassword(store, user, newPassword);
}
```

answered Dec 2 '15 at 15:18

tmg
13.4k 3 48 58

68

Deprecated

This was the original answer. It does work, but has a problem. What if `AddPassword` fails? The user is left without a password.

The original answer: we can use three lines of code:

```
UserManager<IdentityUser> userManager =
    new UserManager<IdentityUser>(new UserStore<IdentityUser>());
```

See also: http://msdn.microsoft.com/en-us/library/dn457095(v=vs.111).aspx

Now Recommended

It's probably better to use the answer that EdwardBrey proposed and then DanielWright later elaborated with a code sample.

edited May 23 '17 at 12:18
Community ♦
1 1

answered Mar 18 '14 at 20:53
Shaun Luttin
68.1k 40 250 309

1 Thank god for this, I thought I'd have to create a new user store until I saw this! – Luke Dec 18 '14 at 14:05

Is there any way to do this directly in SQL? I'd love to hand my DBA a sproc to call when needed instead of an executable. – Mark Richman Feb 24 '15 at 19:19

@MarkRichman That's a new question. One thing you could do, though, is to inspect the generated T-SQL that runs on SQL Server. – Shaun Luttin Feb 25 '15 at 16:00

3 Watch out with this on, whenever AddPassword fails (ie insufficient password complexity), the user will be left without a password. – Chris Apr 15 '15 at 20:18

1 Well the cleanest approach without bypassing any business rules (because when you access the password hasher directly there is no password complexity validation) is what Daniel Wright has proposed. – Chris Apr 17 '15 at 17:58

119

> Or how can I reset without knowing the current one (user forgot password)?

If you want to change a password using the UserManager but you do not want to supply the user's current password, you can generate a password reset token and then use it immediately instead.

```
string resetToken = await UserManager.GeneratePasswordResetTokenAsync(model.Id);
IdentityResult passwordChangeResult = await UserManager.ResetPasswordAsync(model.Id,
resetToken, model.NewPassword);
```

answered Mar 24 '15 at 15:23

6 This is by far the best and cleanest way to set a new password. The problem with the accepted answer is that it bypasses the password complexity validations by directly accessing the password hasher. – Chris Apr 17 '15 at 18:05

6 Fyi, You may get error 'No IUserTokenProvider is registered.' if you get use above logic. See this stackoverflow.com/questions/22629936/…. – Prasad Kanaparthi Aug 30 '15 at 12:30

1 This works for Microsoft.AspNet.Identity in version 2 only, I suppose. You can't find the GeneratePasswordResetTokenAsync method in version 1. – romanoza Jan 13 '16 at 11:22

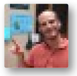
Thank you for your answer. It works like a charm for me. – Thomas.Benz Mar 16 '17 at 13:43

1 If you get **Invalid Token**, make sure the `SecurityStamp` for your user is not null. This may happen for users migrated from other databases, or users which were not created through `UserManager.CreateAsync()` method. – Alisson Jul 18 '18 at 2:49

27

On your `UserManager`, first call GeneratePasswordResetTokenAsync. Once the user has verified his identity (for example by receiving the token in an email), pass the token to ResetPasswordAsync.

answered Apr 15 '14 at 20:49



Edward Brey
27.4k 11 136 199

1 Trying to figure out why ResetPasswordAsync requires a user ID, and a reasonable way to get it from the user when they show up with a token. GeneratePasswordReset uses a token that's over 150 chars... seems like that'd be enough to cryptographically stow a user id so I don't have to implement that myself. :(– pettys Apr 17 '15 at 20:36

I assume It's asking for the user ID so it can enter the reset token into the Identity Database against that user ID. If it didn't do this how would the framework ever know if the token was valid. You should be able to pull the users ID using User.Identity.GetUserId() or similar. – Ryan Buddicom May 4 '15 at 6:07

1 Requiring the user id is a silly choice on the API's part, the token is already in the database when ResetPassword(async) is called and it should be enough just to validate it against the input. – Filip Jul 30 '15 at 14:15

@Filip, the advantage of `ResetPasswordAsync` taking a user ID is that the identity provider only needs to index user IDs, not also tokens. This lets it scale better if there are many users. – Edward Brey Jul 30 '15 at 15:38

@Edward Brey so...more roundtrips = better scalability? – Filip Jul 30 '15 at 15:51

2

```
var result = await IdentityManager.Passwords.ResetPasswordAsync(model.Token,
model.Password);
if (result.Success)
{
    message = "The password has been reset.";
    return RedirectToAction("PasswordResetCompleted", new { message = message });
}
else
{
    AddErrors(result);
}
```

This snippet of code is taken out of the AspNetIdentitySample project available on github

answered Oct 22 '13 at 17:48

sclarson
3,822 2 26 42

protected by Community ♦ Aug 17 '15 at 20:53

Thank you for your interest in this question. Because it has attracted low-quality or spam answers that had to be removed, posting an answer now requires 10 reputation on this site (the association bonus does not count).

Would you like to answer one of these unanswered questions instead?