

The Tutorial **MvcAuth**: <https://docs.microsoft.com/en-us/aspnet/mvc/overview/security/create-an-aspnet-mvc-5-app-with-facebook-and-google-oauth2-and-openid-sign-on>

Works through many more steps than the other examples in this document and is recommended....

1. Setting up SSL in the Project
2. Creating a Google app for OAuth 2 and connecting the app to the project
3. Creating the app in Facebook and connecting the app to the project
4. Examine the Membership Data
5. Adding Profile Data to the User Class
6. Examine the Membership Data
7. Logging off your App and Logging in With Another Account

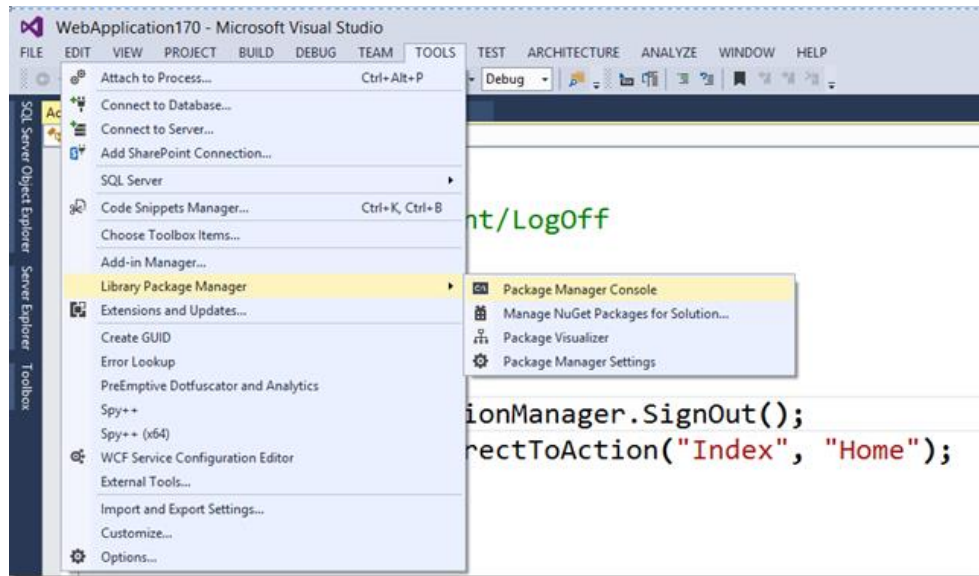
Customizing User Profile fields (Nutshell)

1. Enable migrations
2. Ctrl-F: Find "ApplicationUser" class and add property that you want (e.g FullName).

Example: <https://blogs.msdn.microsoft.com/webdev/2013/10/16/customizing-profile-information-in-asp-net-identity-in-vs-2013-templates/>

Add profile information about the user.

- In Visual Studio 201X, **create a new ASP.NET MVC application**.
 - You can create a Web Forms application and follow the same steps to add profile information.
- **Run the application and register a user**
 - You would notice that at the registration screen, the application only asks for a UserName and password.
 - **Add one more field called BirthDate** that the users should specify when registering a user.
- **Enable Entity Framework Migrations**
 - Migrations are recommended to use when you are modifying the schema of the database or you are changing the Code First Model. ASP.NET Identity uses Entity Framework Code First as an underlying framework. For more information on EF migrations please visit <http://msdn.microsoft.com/en-us/data/jj591621>
 - Open Package Manager Console by doing Tools – Library Package Manager – Package Manager Console



- Type in "Enable-Migrations" and press enter
- **Add properties for the Profile information that you want to store.**
 - In our example we will add BirthDate as a property
 - **Goto Models\IdentityModels.cs and add the following property to ApplicationUser**

```
public class ApplicationUser : IdentityUser { public DateTime BirthDate { get; set; } }
```

- Add using System; at the top of the class.
- **Add-Migrations to modify the database**
 - Update the database to reflect this change by doing the following:
 - Goto Package Manager console and do the following
 - Add-Migration "Birthdate"
 - This will add a migration file to your project
 - Update-Database
 - This command will run all the migration files and update the database schema to add a BirthDate column
- **Modify the application to add a BirthDate field**
 - The steps here are specific to how you would add a view to MVC, but you can do similar view specific rendering in Web Forms as well
 - Add BirthDate to RegisterViewModel in Models\AccountViewModels.cs

```
1: public class RegisterViewModel {  
  
2: [Required] [Display(Name = "User name")]  
  
3: public string UserName { get; set; }  
  
4:  
  
5: [Required]  
  
6: [StringLength(100, ErrorMessage = "The {0} must be  
    at least {2} characters long.", MinimumLength = 6)]  
  
7: [DataType(DataType.Password)] [Display(Name = "Password")]  
  
8: public string Password { get; set; } [DataType(DataType.Password)]  
    [Display(Name = "Confirm password")]  
  
9:  
  
10: [Compare("Password", ErrorMessage =  
    "The password and confirmation password do not match.")]  
  
11: public string ConfirmPassword { get; set; }  
  
    public DateTime BirthDate { get; set; } }
```

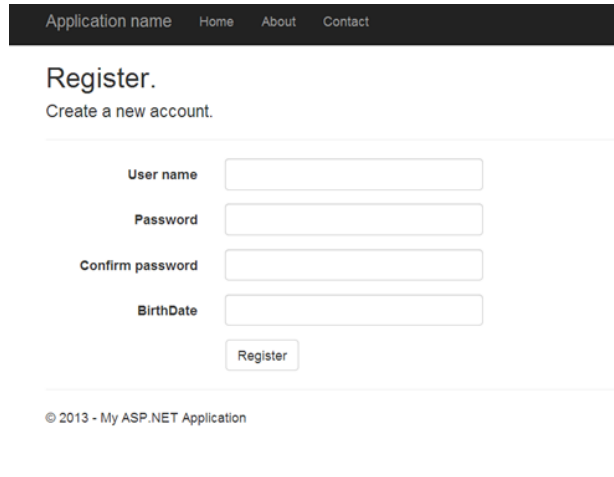
- Add using System; at the top of the class.
- Update the Register view in Views\Account to add a field to enter BirthDate.

```
1: <div class="form-group">  
  
2: @Html.LabelFor(m => m.BirthDate, new { @class = "col-md-2 control-label" })  
  
3: <div class="col-md-10"> @Html.TextBoxFor(m => m.BirthDate, new { @class = "form-control" })  
  
4: </div>  
  
5: </div>
```

- Update the Register Action in AccountController to store the BirthDate as well for the User

```
var user = new ApplicationUser() { UserName = model.UserName, BirthDate=model.BirthDate };
```

- Run the application and register a user again. You will see the BirthDate field as shown below



- **Display the profile information on the page**
 - When the User Logs in, you can display the profile information by doing the following
 - Get the current logged in UserId, so you can look the user up in ASP.NET Identity system
 - `var currentUserId = User.Identity.GetUserId();`
 - Instantiate the UserManager in ASP.Identity system so you can look up the user in the system
 - `var manager = new UserManager<MyUser>(new UserStore<MyUser>(new MyDbContext()));`
 - Get the User object
 - `var currentUser = manager.FindById(User.Identity.GetUserId());`
 - Get the profile information about the user
 - `currentUser.BirthDate`

Adding Profile information in a different table.

While this post shows you a way where you can easily add profile properties to the User Table itself. Since ASP.NET Identity uses Entity Framework, you can customize profile information as you would like to. For eg. let's say that you want to add profile information in a different table rather than the same table then you can do the following to your model

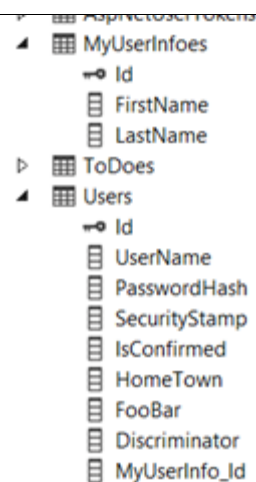
Add a new class to store Profile information**Code Snippet**

```

1. public class MyUser : IdentityUser
2. {
3.     public virtual MyUserInfo MyUserInfo { get; set; }
4. }
5.
6. public class MyUserInfo{
7.     public int Id { get; set; }
8.     public string FirstName { get; set; }
9.     public string LastName { get; set; }
10. }
11. public class MyDbContext : IdentityDbContext<MyUser>
12. {
13.     public MyDbContext()
14.         : base("DefaultConnection")
15.     {
16.     }
17.     public System.Data.Entity.DbSet<MyUserInfo> MyUserInfo { get; set; }
18. }

```

In this case when you run the application you will get the FirstName and LastName in a different table called MyUserInfo



Getting Profile information

- When the User Logs in, you can display the profile information by doing the following
- Get the current logged in UserId, so you can look the user up in ASP.NET Identity system
 - `var currentUserId = User.Identity.GetUserId();`
- Instantiate the UserManager in ASP.Identity system so you can look up the user in the system
 - `var manager = new UserManager<MyUser>(new UserStore<MyUser>(new MyDbContext()));`
- Get the User object
 - `var currentUser = manager.FindById(User.Identity.GetUserId());`
- Get the profile information about the user
 - `currentUser.MyUserInfo.FirstName`

References

1. **MvcAuth ****Create an ASP.NET MVC 5 App with Facebook, Twitter, LinkedIn and Google OAuth2 Sign-on (C#):
 - a. **Add birth date and home town to the user data during registration**
 - b. <https://docs.microsoft.com/en-us/aspnet/mvc/overview/security/create-an-aspnet-mvc-5-app-with-facebook-and-google-oauth2-and-openid-sign-on>
2. MVC - Customizing profile information in ASP.NET Identity in VS 2013 templates: <https://blogs.msdn.microsoft.com/webdev/2013/10/16/customizing-profile-information-in-asp-net-identity-in-vs-2013-templates/>
3. MVC - Customize Profile Info in ASP.NET Identity: <https://www.codeproject.com/Articles/1010804/How-to-Customize-Profile-Info-in-ASP-NET-Identity>