Trying to change table names in ASP.NET Identity 2.0

asp.net-identity-2 c# entity-framework

English (en)

Question

I want to change the names of the tables used by ASP.NET Identity 2.0. I've seen various similar questions but nothing that solves my problem. For instance this type of solution: http://www.goatly.net/customizing-table-names-for-identitydbcontextwithcustomuser-data-contexts seems to depend on Code First(?). At any rate implementing OnModelCreating does nothing for me. I've basically renamed the AspNetUsers and similar tables to my own names and want to be able to use these. I have an existing EF context (MyContext) that I want to use, so I modified it to derive from IdentityDbContext (editing the t4 template), and then in Startup.Auth I have:

```
        UserManagerFactory = () =>
        {
            var context = new MyContext();

            Database.SetInitializer<MyContext>(new
IdentityDbInitializer());

            var userStore = new UserStore<IdentityUser>
(context){
                DisposeContext = true
            };

            return new UserManager<IdentityUser>
(userStore);
        };
```

But the problem occurs when I try to log in that I get "cannot contact server". I imagine that is because my UserStore has no way of knowing which are the User tables on my context. I'd hoped this is what the OnModelCreating code would do, but I'm hazy on this. I imagine UserStore is still looking for an "AspNetUsers" table, though I don't know where these names come from. I'm also uneasy about modifying the t4 template for my context - is this the way I'm supposed to derive from IdentityDbContext?

Any help greatly appreciated.

see sharper

Accepted Answer

Few steps to follow:

- Install NuGet Package: `Microsoft.AspNet.Identity.EntityFramework`
- Add a `connection string` to your web.config/app.config

Now you have to define your custom `Database Context`:

```
public class MyContext : IdentityDbContext
{
    public MyContext()
        : base(<connection string name>)
    {

    }

    protected override void
OnModelCreating(DbModelBuilder modelBuilder)
    {
        base.OnModelCreating(modelBuilder);

        modelBuilder.Entity<IdentityUser>()
            .ToTable("Users");

        modelBuilder.Entity<IdentityRole>()
            .ToTable("Roles");

        modelBuilder.Entity<IdentityUserRole>()
            .ToTable("UserRoles");

        modelBuilder.Entity<IdentityUserClaim>()
            .ToTable("UserClaims");

        modelBuilder.Entity<IdentityUserLogin>()
            .ToTable("UserLogins");
    }
}
```

As you can see I've used `DbModelBuilder` to map all the entities to a new tables.

- Open NuGet Package Manager Console
- Execute command `Enable-Migrations`

It will create a folder `Migrations` with the configuration file `Configuration.cs`.

It should look something like this:

```
internal sealed class Configuration :
DbMigrationsConfiguration<ConsoleApplication1.Models.MyCon

{
    public Configuration()
    {
        AutomaticMigrationsEnabled = false;
    }

    protected override void
Seed(ConsoleApplication1.Models.MyContext context)
    {

    }
}
```

In the constructor change the property `AutomaticMigrationsEnabled` to `true`.

- Open NuGet Package Manager Console
- Execute command `Update-Database`

It should run the script to create the new tables.

You can customize your entities (and Ids) creating custom class for each interface defined.

```
public class MyUser : IdentityUser<string, MyUserLogin,
MyUserRole, MyUserClaim>
{
}
```

Since you're using Owin you can define your UserStore:

```
public class MyUserStore: UserStore<MyUser, MyRole,
string, MyUserLogin, MyUserRole, MyUserClaim>
{
    public MyUserStore(MyContext context)
        : base(context)
    {
    }
}
```

and your UserManager implementation:

```
public class ApplicationUserManager :
UserManager<ASPNETIdentity2.Models.MyUser, string>
{
    public
ApplicationUserManager(IUserStore<ASPNETIdentity2.Models.My
 string> store)
        : base(store)
    {

    }

    public static ApplicationUserManager
Create(IdentityFactoryOptions<ApplicationUserManager>
options, IOwinContext context)
    {
        var manager = new ApplicationUserManager(new
MyUserStore(context.Get<MyContext>()));

        manager.UserValidator = new
UserValidator<MyUser, string>(manager)
        {
            AllowOnlyAlphanumericUserNames = false,
            RequireUniqueEmail = true
        };

        manager.PasswordValidator = new
PasswordValidator()
        {
            RequiredLength = 5,
            RequireNonLetterOrDigit = false,     // true
            // RequireDigit = true,
            RequireLowercase = false,
            RequireUppercase = false,
        };

        return (manager);
    }
}
```

And your `Owin.Startup` should look something like this:

```
public class Startup
{
    public void Configuration(IAppBuilder app)
    {
        app.CreatePerOwinContext(MyContext.Create);
        app.CreatePerOwinContext<ApplicationUserManager>
(ApplicationUserManager.Create);
    }
}
```

If you want to have a look at a custom implementation you can check my GitHub repository with a simple working solution on ASP.NET MVC.

UPDATE:

There's another project in that solution with a minimal setup; basically you only need to define your context (IdentityDbContext) if you only want to rename your tables.

The project can be found here.

LeftyX

Popular Answer

Identity 2 with Entity Framework assumes Code First. This project seems to do what I need: https://github.com/cbfrank/AspNet.Identity.EntityFramework

see sharper

View more on Stack Overflow