ASP.NET Identity

Working with claims & roles

ASP.NET Identity

In this exercise we are going to look at how you can:

- Create & Delete **Identities**
- Create & Delete **Claims**
- Create, Delete & Edit **Roles**
- Assign **Roles** to **Identities**

Note that this exercise requires that you have an existing identity database structure.

ASP.NET Identity

First of all we are going to create a **UserManager** and a **RoleManager**. These managers will give you everything you need to create a **CRUD** for users and roles, and much more.

```
private static readonly RoleManager<IdentityRole> roleManager =
new RoleManager<IdentityRole>(new RoleStore<IdentityRole>());

private static readonly UserManager<IdentityUser> userManager =
new UserManager<IdentityUser>(new UserStore<IdentityUser>(new
  IdentityDbContext<IdentityUser>()));
```

ASP.NET Identity

We **create** an **Identity** and use the **UserManager** to add it to the database

```
public static void AddIdentity(string email, string phone, string password)
{
            var user = new IdentityUser()
            { UserName = email, Email = email, PhoneNumber = phone };
            userManager.Create(user, password);
}
```

The UserManager contains several useful methods which can be used to manage your users
http://msdn.microsoft.com/en-us/library/dn613059%28v=vs.108%29.aspx

ASP.NET Identity

We find an existing **Identity** by its name and **delete** it, using the **Usermanager**

```
public static void DeleteIdentity(string username)
{
    var user = userManager.FindByName(username);
    userManager.Delete(user);
}
```

ASP.NET Identity

We **create** a **Claim** and **add** it to an existing **Identity**. Please note that all claims in the database are bound to a specific user.

```
 public static void AddClaimToUser(string username, string type, string
value)
        {
            var user = userManager.FindByName(username);
            userManager.AddClaim(user.Id, new Claim(type, value));
        }
```

ASP.NET Identity

We **delete** a **Claim** from an existing **Identity**.

```
 public static void DeleteClaim(string username, string type, string
value)
        {
            var user = userManager.FindByName(username);
            var userClaim = userManager.GetClaims(user.Id).Where(c =>
                c.Value.Equals(value) && c.Type.Equals(type)).First();

            userManager.RemoveClaim(user.Id, userClaim);
        }
```

ASP.NET Identity

We **create** a **Role** by using the **Rolemanager** to add it to the database

```
public static void AddRole(string roleName)
{
    roleManager.Create(new IdentityRole(roleName));
}
```

The RoleManager contains several useful methods which can be used to manage your roles
http://msdn.microsoft.com/en-us/library/dn613286(v=vs.108).aspx

ASP.NET Identity

We **delete** an already existing **Role**, using the **Rolemanager**

```
public static void DeleteRole(string roleName)
        {
            var role = roleManager.FindByName(roleName);
            roleManager.Delete(role);
        }
```

ASP.NET Identity

We **update** an already existing **Role**, using the **Rolemanager**

```
public static void UpdateRole(string oldRoleName, string newRoleName)
        {
            var role = roleManager.FindByName(oldRoleName);
            role.Name = newRoleName;
            roleManager.Update(role);
        }
```

ASP.NET Identity

We **add** an existing **Identity** to an existing **Role** using the **Usermanager**

```
public static void AddUserToRole(string username, string roleName)
        {
            var user = userManager.FindByName(username);
            userManager.AddToRole(user.Id, roleName);
        }
```