7.1.0 – ASP.NET Security Introduction

Introduction

In previous lessons we created a web form that allowed us to **Add**, **Update**, or **Delete** a product from the database. We would not want to make these abilities available to all the web visitors. Additionally, there is the **Log in** option on the web site:

If we press the **Log in** link, we get the following:

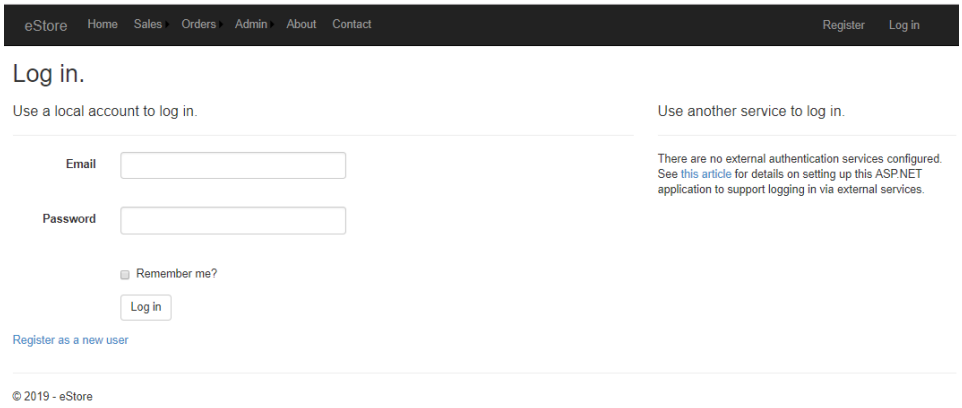

Figure 1: Login Page

Also the **Register** link gives us:

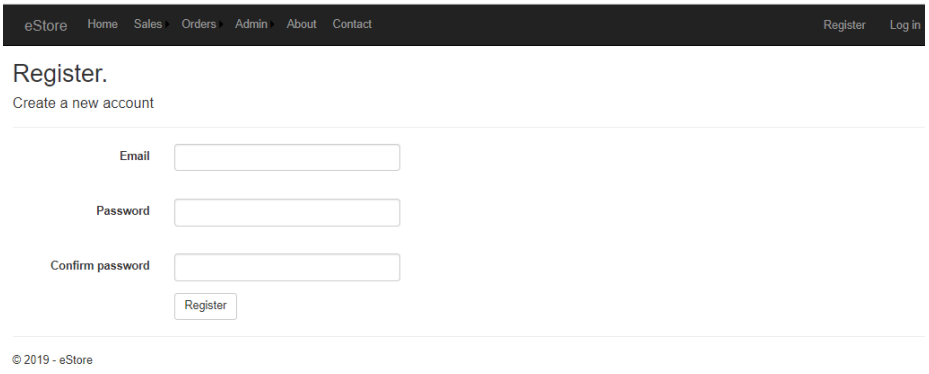

Figure 2: Register Page

What we need to do is to configure the web site to allow us to use these pages.

What is ASP.NET Security

ASP.NET Security is a framework that is built-in to all ASP web applications. This framework was created for us when we created our web application. The following screenshot shows what is created for us:

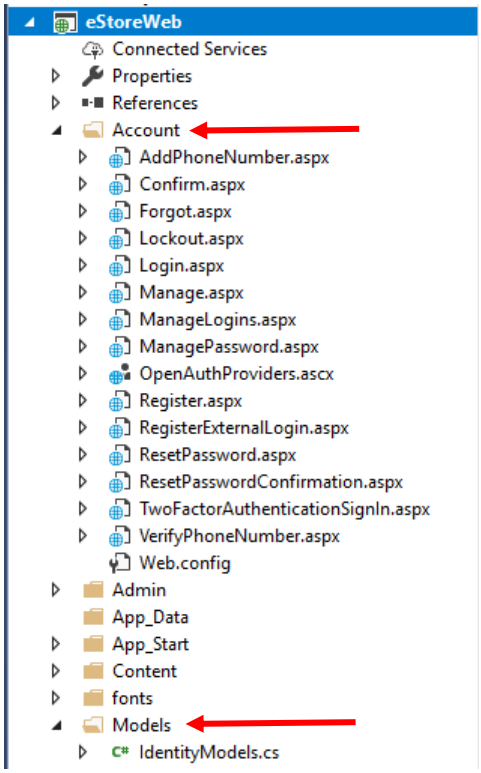

Figure 3: Security Framework Files

Notice all the files in the **`Account`** folder of the web application, and that there is a file in the **`Models`** folder. These were created for us automatically.

Even though the framework was created for us, we need to customize the framework to meet our specific needs.

Security for eStore

There are files in the **Code Files (7.1.0)** folder on Moodle that are required for this lesson. Download these files to your computer and follow these steps below to setup and modify the framework to work with the eStore_2018 database:

Steps

1. Create a new folder, called **`Security`**, in the **eStoreWeb** project.
2. Create a web form, with master page,in this new folder with the name **`EmployeeAdmin.aspx`**.
3. Replace the

```
<asp:Content ID="Content1" ContentPlaceHolderID="MainContent" runat="server">
</asp:Content>
```
with the contents of the **EmployeeAdmin_aspx.txt** file.

4. Open the **EmployeeAdmin.aspx.cs** file and replace the contents of the **namespace** with the contents of the **EmployeeAdmin_aspx_cs.txt** file. Add the following using statements:

```
using eStoreSystem.BLL;
using eStoreData.POCOs;
```

5. Modify the **Admin** menu in the **Site.Master** file to be:

```
<asp:MenuItem Text="Admin">
    <asp:MenuItem Text="Product Maintenance" NavigateUrl="~/Admin/ProductMaintenance.aspx" />
    <asp:MenuItem Text="Employee Admin" NavigateUrl="~/Security/EmployeeAdmin.aspx" />
</asp:MenuItem>
```

New menu item

6. Add the **SecurityController.cs** and **SecurityDbContextInitializer.cs** code files to the **Security** folder. This will be done by adding an existing item. In the **SecurityController.cs**, comment out line 108 and 109 to be:

```
//CustomerController controller = new CustomerController();
//controller.UpdateCustomer(custId, existing.Email, existing.PhoneNumber);
```

7. In the **Models** folder, open the **IdentityModels.cs** code file. Add the following using statements:

```
using eStoreWeb.Security;
using System.Data.Entity;
```

8. Just after the class definition, modify the code to be:

```
public class ApplicationUser : IdentityUser
{
    #region Custom Properties
    //these properties will become attributes on the AspNetUsers Sql Table
    //once a numeric datatype is made nullable the default for that data is null.
    public int? EmployeeId { get; set; }
    public int? CustomerId { get; set; }
    #endregion
```

This code sets up custom class properties so that we can use the existing **EmployeeID** or **CustomerID**. The id needed depends on whether we have a customer or an employee.

9. In the same file, locate the constructor as shown below:

```
public ApplicationDbContext()
    : base("DefaultConnection", throwIfV1Schema: false)
{
}
```

Modify this code to be:

```
public ApplicationDbContext()
    : base("DefaultConnection", throwIfV1Schema: false)
{
    //this method will be used to load the sql database with a default
    // webmaster and default customer IF the security tables DO NOT exist
    // in your database, if they do this line will be ignored.
    Database.SetInitializer<ApplicationDbContext>(new SecurityDbContextInitializer());
}
```

This code will add the security tables to the database, using the `DefaultConnection`, only if they are not already added.

10. Open the **`Web.config`** file and below the code line `</configSections>` add the following code to setup some web site defaults. These are needed to allow us to have startup roles and users (HINT: Use the **`AppSettings.txt`** file contents):

```
<appSettings >
    <add key="startupRoles" value="Administrators;Employees;Customers;RegisteredUsers" />

    <add key="adminUserName" value="Webmaster" />
    <add key="adminEmail" value="Webmaster@eStore.tba" />
    <add key="adminPassword" value="Pa$$w0rd1" />
    <add key="adminPhone" value="8885551212"/>

    <add key="customerUserName" value="HansenB" />
    <add key="customerEmail" value="HansenB@hotmail.com" />
    <add key="customerPassword" value="Pa$$w0rd1" />
    <add key="customerPhone" value="7805551212"/>

    <add key="adminRole" value="Administrators" />
    <add key="customerRole" value="Customers" />
    <add key="employeeRole" value="Employees" />

    <add key="newUserPassword" value="Pa$$word1" />
</appSettings>
```

11. Open the **`WebConnectionStrings.config`** file. As this file contains the database connection strings, it needs to be modified so that the security tables will be added to the database. The file needs to be modified to look like:

```
<connectionStrings>
    <!-- <add name="DefaultConnection" connectionString="Data
Source=(LocalDb)\MSSQLLocalDB;AttachDbFilename=|DataDirectory|\aspnet-eStoreWeb-
20190219093257.mdf;Initial Catalog=aspnet-eStoreWeb-20190219093257;Integrated Security=True"
    providerName="System.Data.SqlClient" /> -->
  <add name="eStoreContext" connectionString="data source=.\SQLEXPRESS;initial
catalog=eStore_2018;integrated security=True;MultipleActiveResultSets=True;App=EntityFramework"
    providerName="System.Data.SqlClient" />
  <add name="DefaultConnection" connectionString="data source=.\SQLEXPRESS;initial
catalog=eStore_2018;integrated security=True;MultipleActiveResultSets=True;App=EntityFramework"
    providerName="System.Data.SqlClient" />
</connectionStrings>
```

The changes here are: (1) we commented out the original "DefaultConnection", then added a new "DefaultConnection" that points to the eStore_2018 database.

12. In the **Account** folder, open the **Login.aspx** web form. Modify the lines 19-21 from the original:

```
<div class="form-group">
    <asp:Label runat="server" AssociatedControlID="Email" CssClass="col-md-2 control-
label">Email</asp:Label>
    <div class="col-md-10">
        <asp:TextBox runat="server" ID="Email" CssClass="form-control" TextMode="Email" />
        <asp:RequiredFieldValidator runat="server" ControlToValidate="Email"
            CssClass="text-danger" ErrorMessage="The email field is required." />
    </div>
</div>
```

To:

```
<div class="form-group">
    <asp:Label runat="server" AssociatedControlID="Email" CssClass="col-md-2 control-label">User
Name</asp:Label>
    <div class="col-md-10">
        <%-- remove the TextMode="Email" from the Textbox change the Label text to User Name--%>
        <asp:TextBox runat="server" ID="Email" CssClass="form-control"  />
        <asp:RequiredFieldValidator runat="server" ControlToValidate="Email"
             CssClass="text-danger" ErrorMessage="The email field is required." />
    </div>
</div>
```

We do this to bypass the email validation; use only the username and password for login credentials.

13. In the **Account** folder, open the **Register.aspx** web form. **Above** the lines 13-20 from the original (shown below). These changes need to be made so that this web form uses the username and not the email:

```
<div class="form-group">
    <asp:Label runat="server" AssociatedControlID="Email" CssClass="col-md-2 control-
label">Email</asp:Label>
    <div class="col-md-10">
        <asp:TextBox runat="server" ID="Email" CssClass="form-control" TextMode="Email" />
        <asp:RequiredFieldValidator runat="server" ControlToValidate="Email"
             CssClass="text-danger" ErrorMessage="The email field is required." />
    </div>
</div>
```

Add:

```
<%-- copy the Email form group and alter to accept your User Name --%>
<div class="form-group">
    <asp:Label runat="server" AssociatedControlID="UserName" CssClass="col-md-2 control-
label">User Name</asp:Label>
    <div class="col-md-10">
        <asp:TextBox runat="server" ID="UserName" CssClass="form-control" />
        <asp:RequiredFieldValidator runat="server" ControlToValidate="UserName"
            CssClass="text-danger" ErrorMessage="The user name field is required." />
```

```
    </div>
</div>
```

14. Open the code behind file, **`Register.aspx.cs`** and verify line 18 to be:

```
var user = new ApplicationUser() { UserName = Email.Text, Email = Email.Text };
```

15. Next, we need to add the **`EmployeeController.cs`** class file to the **BLL** folder. We can do this by adding an existing file (the file is in the **Code Files (7.1.0)** folder).
16. Finally, we need to add the **`UserContact.cs`** code file to the **POCOs** folder.
17. **BUILD** your solution, correct any errors before running the web application.

Testing

Run your web application using **Ctrl+F5** (this will bypass debugging and all the errors will appear) as if there are any validation error messages, the **`MessageUserControl`** will be able to display them on the web form. When the web application starts, you may see something like the following when using Chrome as your web browser:

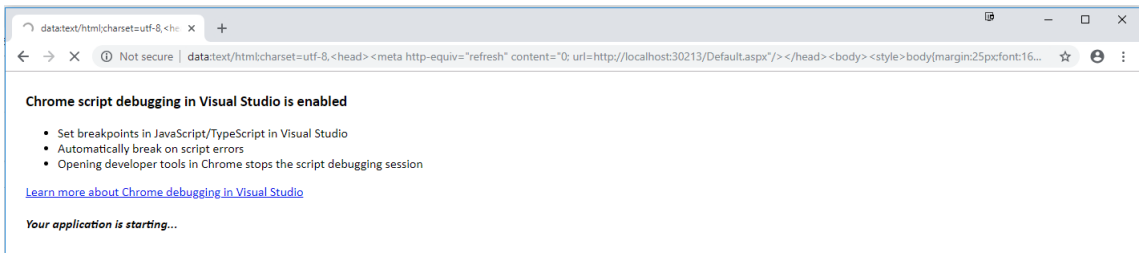

Figure 4: Web Application Starting in Google Chrome

First press the **Log in** link button and login as **`Webmaster`** with the password **`Pa$$w0rd1`**. You should see the following which shows a successful login:

Figure 5: Successful Login

Select the **`Employee Admin`** menu option:

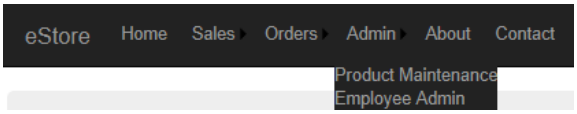

Figure 6: New Admin Menu

Now you will get:

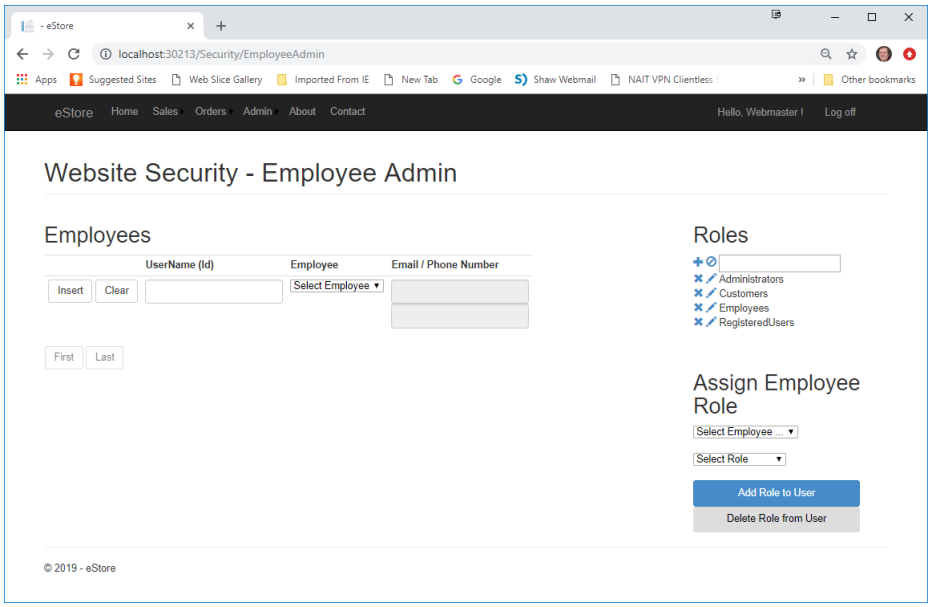

Figure 7: Security Admin Web Form

From here we can add an existing Employee (one that already exists in the database) to be able to login to the web application (no current Employee in the database can do that). Select Bob Waters:

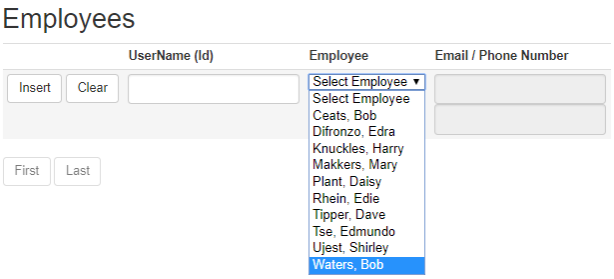

Figure 8: Select an Employee

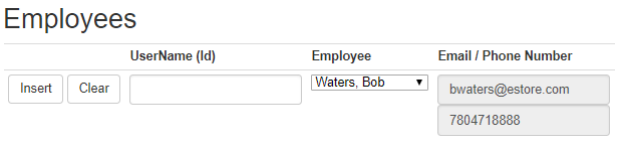

Figure 9: Employee Selected

Give the Employee a username, such as **`bwaters`**; the email and phone number are pre-populated. Press the **Insert** button. You should now see something like (the auto-generated ID will be different):

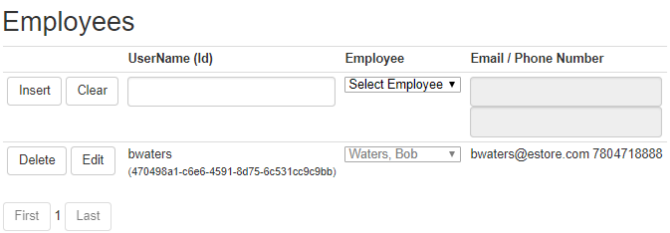

Figure 10: Employee Added to the AspNetUsers Table

From here we can edit the new user (Notice that we can only edit the **Email / Phone Number** entries as the username is a unique name in the database.):

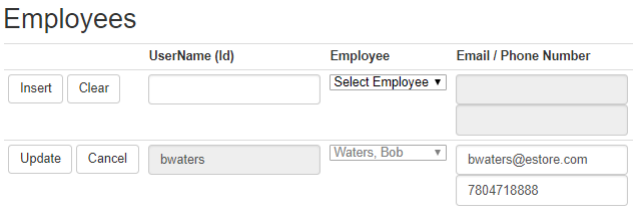

Figure 11: Edit User

When you edit the email and/or the phone number on this form, the web form will call the update in the **AspNetUsers** table and the **Employee** table in the database.

On the right side of this web form you see:

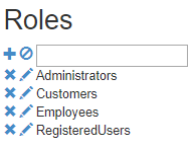

Figure 12: Roles Administration

We can Add, Edit, or Delete the default roles which were created by the line we added to the Web.config file:

```
<add key="startupRoles" value="Administrators;Employees;Customers;RegisteredUsers" />
```

We can also assign a User to a Role:

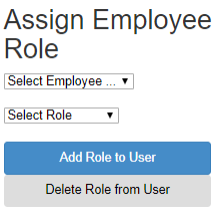

Figure 13: User Role Administration

If we try to add this new user to a role right after adding the user, the new user will not be available in the **Select Employee …** dropdown list. We can either log off or navigate to another web page on the web site, then come back to this page. When we do this, we will see:

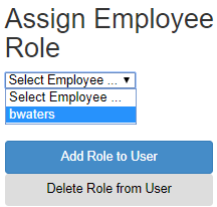

Figure 14: New User in the Database

We can now select that user and add them to a role. We want to add this user to the **Employees** and **Administrators** roles. When we press the **Add Role to User** button, and it is successful, we see:

Figure 15: Successfully Added Role to User

If we try to add the same role to this user we get:

Figure 16: User Role Processing Error

Database Changes

When we implemented ASP.NET Security, the database was updated with some new tables:

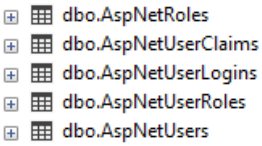

Figure 17: New Database Tables

These were added through the **`DefaultConnection`** and the **`ApplicationDbContext`** class constructor.

The ERD for these tables is:

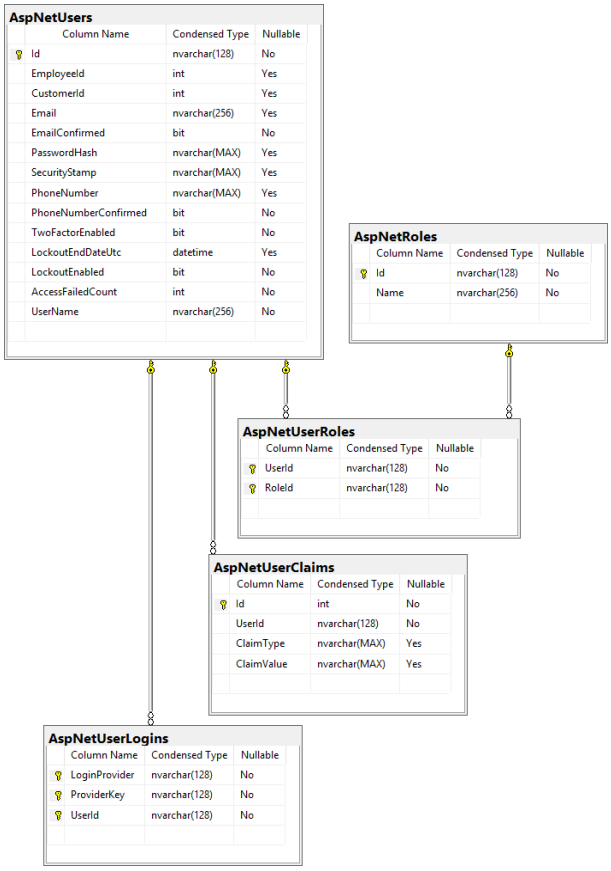

Figure 18: Security Tables ERD

During our setup and testing the **AspNetUsers** table has the following information:

	Id	EmployeeId	CustomerId	Email	EmailConfirmed	PasswordHash	SecurityStamp	PhoneNumber	PhoneNumberConfirmed	TwoFactorEnabled	LockoutEndDateUtc	LockoutEnabled	AccessFail...	UserName
1	456e2893-b4e9-40c7-93a7-df21dd166546	NULL	NULL	Webmaster@Store.tba	0	AJaQGDh12mscFlOCfKeSfYNXH7c8Zh51ndlScDSFBQIUeRxuuhl...	387ad756-3e2b-4352-a61d-05afe2103f1a	NULL	0	0	NULL	0	0	Webmaster
2	57e3f487-c654-4293-9a1f-696a1c68a7fb	NULL	NULL	HansenB@hotmail.com	0	AExlCh9W3BYKWbIH05FuuZ2mTtFnM6dWbHACcC9DHUkN...	a58dc1a6-48ac-41fd-aa78-bba1641d96fe	NULL	0	0	NULL	0	0	HansenB
3	cfe77269-d6bc-4be7-90f6-8028160eedf6	1	NULL	bwaters@estore.tbd	0	AGS3NZKu81x4C/t+KX3FQwzYLaFchUjbW98g+ApUcFMqA+bC...	8daa94c7-3b37-4da8-b1d7-e75f9025cc22	NULL	0	0	NULL	1	0	bwaters

Figure 19: AspNetUsers Data

The **AspNetRoles** table has the following data:

	Id	Name
1	937fccca-9c30-4765-8bbd-0ad00fbc9579	Administrators
2	db68e5af-b577-4e2a-a5be-e45af092457a	Customers
3	c7bb4352-de49-4dd6-a982-828b69e47cc8	Employees
4	3b3ee262-b4c0-4e9f-8f59-05262a66074b	RegisteredUsers

Figure 20: AspNetRoles Data

And, the **AspNetUserRoles** table has the following data:

	UserId	RoleId
1	456e2893-b4e9-40c7-93a7-df21dd166546	937fccca-9c30-4765-8bbd-0ad00fbc9579
2	cfe77269-d6bc-4be7-90f6-8028160eedf6	937fccca-9c30-4765-8bbd-0ad00fbc9579
3	cfe77269-d6bc-4be7-90f6-8028160eedf6	c7bb4352-de49-4dd6-a982-828b69e47cc8
4	57e3f487-c654-4293-9a1f-696a1c68a7fb	db68e5af-b577-4e2a-a5be-e45af092457a

Figure 21: AspNetUserRoles Data

Note: You should be able to see the relationship between the tables by comparing the data seen in these three data outputs.

Exercise

TBD.