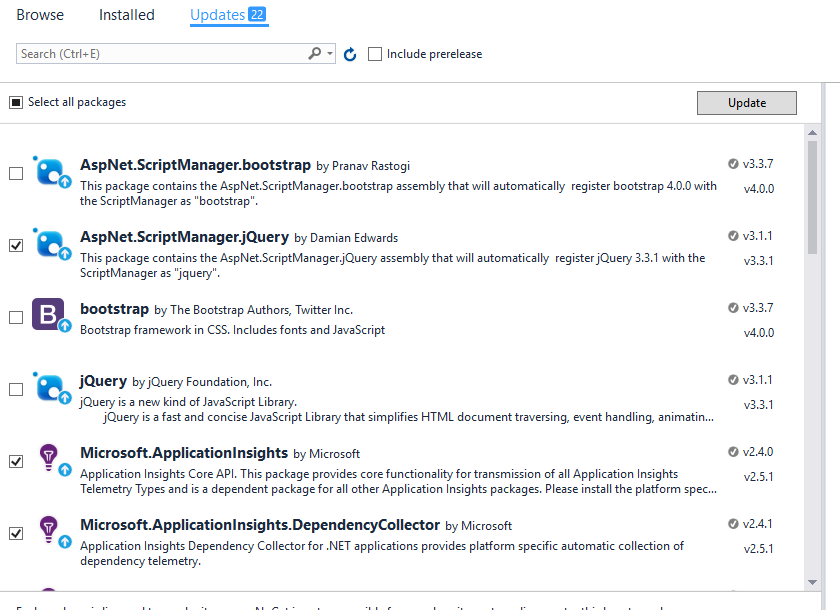
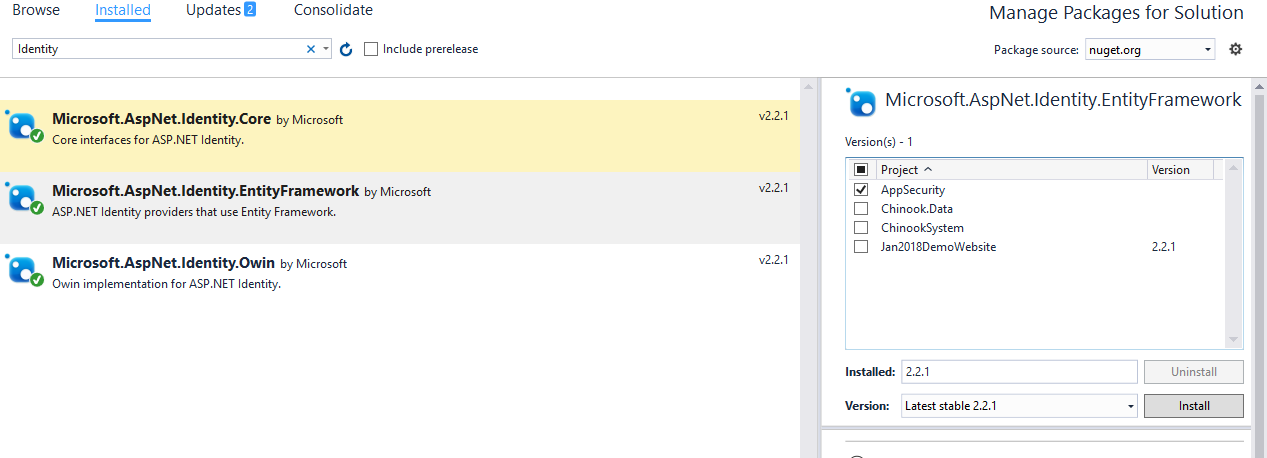
# Security Migration

1. Ensure Nuget manager packages all updated, do not mass update ScriptManager and Bootstrap. If you get the Restart yellow banner, YOU MUST CLICK Restart!!!!

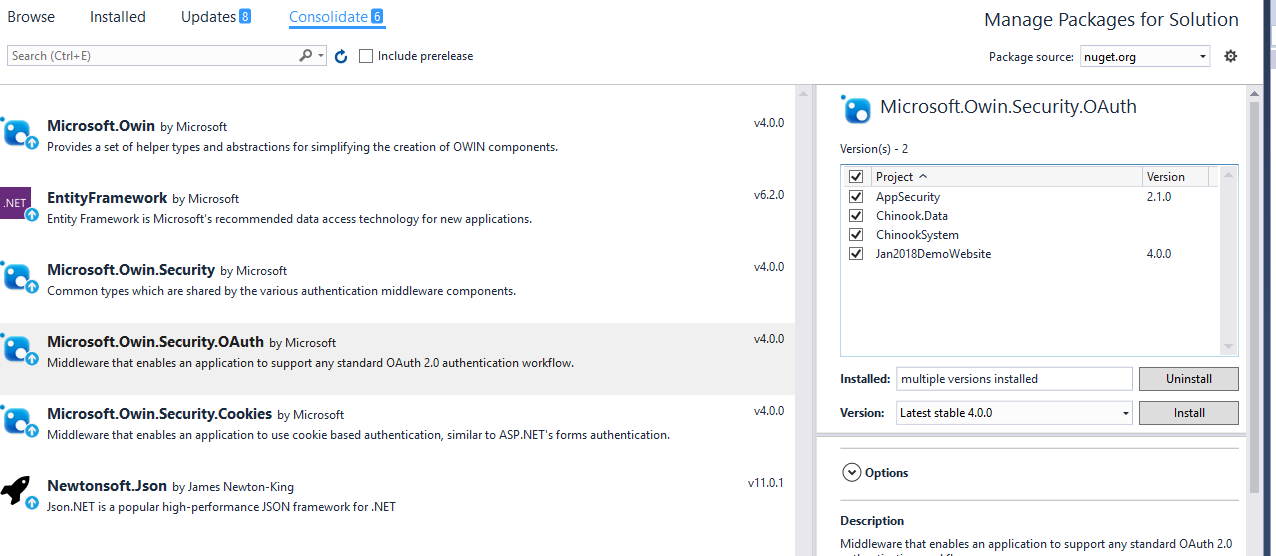




1. Update AspNet.ScriptManager to 3.3.7 ( will bring bootstrap in line)
2. Create a new class library for security AppSecurity
3. Save all work. Close and reopen solution. This will ensure all project config files are properly up to date.
4. Add folders BLL/ DAL/Entities (class organization) . Delete Class1.cs file.
5. Open Nuget Manager for **Solution** and go to Installed
6. Search for Identity. Select MicroSoft.AspNet.Identity.EntityFramework, check AppSecurity, Install.



1. Select .Identity.Owin, check AppSecurity, Install.
2. Consolidate: choose .Owin.Security.OAuth , any remaining packages



1. **Restart yellow line if it appears.**Close and reopen your application. Execute ctrl + F5.
2. Create three new classes under the AppSecurity.BLL folder called ApplicationUserManager, EmailService and SmsService. This will be filled with code during Step 12.
3. Under the App\_Start folder of the wep application, move the classes ApplicationUserManager, EmailService and SmsService from IdentityConfig.cs file to separate class files in BLL. Ensure you have actually removed **only** these 3 classes from IdentityConfig.
4. Open Models/IdentityModels.cs
5. Create an ApplicationUser class in Entities. Move the ApplicationUser class from Models/IdentityModels.cs to the AppSecurity class.
6. Create an ApplicationDbContext class in DAL. Move the ApplicationDbContext class from Models/IdentityModels.cs to the AppSecurity class.
7. **DO NOT touch anything in the Helpers region**. You can delete the excess empty name space from Models/IdentityModels.cs.
8. Resolve all references in AppSecurity project file: ApplicationUser, ApplicationDbContext, EmailService, SmsService then ApplicationUserManager.
9. Add a reference in your WebApplication to AppSecurity.
10. Rebuild your solution.
11. There will be several references to resolve in the web application. Resolve all reference errors in WebApplication. Also, if you deleted the empty namespace in Step 16, you will need to remove these references from the web application wherever they exist. You may have to rebuild your solution several times until all references have been located and resolve. All this work is simply because we migrated the security out of the web application project into it’s own security project.
12. Rebuild AppSecurity until you should get a clean build.
13. Open your WebConnectionStrings.config. Alter your Data Source and Initial Catalog to point to your database.

# Create Default User

1. Add the following constants to ApplicationUserManager

#region Constants

internal const string STR\_DEFAULT\_PASSWORD = "Pa$$word1";

/// <summary>Requires FirstName and LastName</summary>

internal const string STR\_USERNAME\_FORMAT = "{0}.{1}";

/// <summary>Requires UserName</summary>

internal const string STR\_EMAIL\_FORMAT = "{0}@Chinook.ca";

internal const string STR\_WEBMASTER\_USERNAME = "Webmaster";

#endregion

1. Add a reference to AppSecurity for your application and data libraries to your database: ChinookSystem.
2. Add the following code to your ApplicationUser class. This will allow you to place your own data on the AspNetUser table in sql.

//Add nullable fields for linking user data (ie employee, customer) to security  
 data

public int? EmployeeID { get; set; }

public int? CustomerID { get; set; }

1. Create the following class in your AppSecurity.Entities folder.

public static class SecurityRoles

{

public const string WebsiteAdmins = "WebsiteAdmins";

public const string RegisteredUsers = "RegisteredUsers";

public const string Staff = "Staff";

public static List<string> DefaultSecurityRoles

{

get

{

List<string> value = new List<string>();

value.Add(WebsiteAdmins);

value.Add(RegisteredUsers);

value.Add(Staff);

return value;

}

}

}

1. Add the following 2 POCOs to your AppSecurity library. Create a separate POCOs folder.

public class EmployeeListPOCO

{

public int EmployeeId { get; set; }

public string FirstName { get; set; }

public string LastName { get; set; }

}

public class RegisteredEmployeePOCO

{

public string UserName { get; set; }

public int EmployeeId { get; set; }

}

1. Add the following code to ApplicationUserManager to automatically generate user accounts and roles for existing employees on the database. This will also create a webmaster account. **You will need to change the class access level in your application library context class to public.** Resolve any references.

public void AddDefaultUsers()

{

using (var context = new ChinookContext())

{

// Add a web master user

//Users accesses all the records on the AspNetUsers table

//UserName is the user logon user id (dwelch)

if (!Users.Any(u => u.UserName.Equals(STR\_WEBMASTER\_USERNAME)))

{

var webMasterAccount = new ApplicationUser()

{

//create a new instance that will be used as the data to

// add a new record to the AspNetUsers table

//dynamically fill two attributes of the instance

UserName = STR\_WEBMASTER\_USERNAME,

Email = string.Format(STR\_EMAIL\_FORMAT, STR\_WEBMASTER\_USERNAME)

};

//place the webmaster account on the AspNetUsers table

this.Create(webMasterAccount, STR\_DEFAULT\_PASSWORD);

//place an account role record on the AspNetUserRoles table

//.Id comes from the webmasterAccount and is the pkey of the Users table

//role will comes from the Entities.Security.SecurityRoles

this.AddToRole(webMasterAccount.Id, SecurityRoles.WebsiteAdmins);

}

//get all current employees

//linq query will not execute as yet

//return datatype will be IQueryable<EmployeeListPOCO>

var currentEmployees = from x in context.Employees

select new EmployeeListPOCO

{

EmployeeId = x.EmployeeId,

FirstName = x.FirstName,

LastName = x.LastName

};

//get all employees who have an user account

//Users needs to be in memory therfore use .ToList()

//POCO EmployeeID is an int

//the Users Employee id is an int?

//since we will only be retrieving

// Users that are employees (ID is not null)

// we need to convert the nullable int into

// a require int

//the results of this query will be in memory

var UserEmployees = from x in Users.ToList()

where x.EmployeeID.HasValue

select new RegisteredEmployeePOCO

{

UserName = x.UserName,

EmployeeId = int.Parse(x.EmployeeID.ToString())

};

//loop to see if auto generation of new employee

//Users record is needed

//the foreach cause the delayed execution of the

//linq above

foreach (var employee in currentEmployees)

{

//does the employee NOT have a logon (no User record)

if (!UserEmployees.Any(us => us.EmployeeId == employee.EmployeeId))

{

//create a suggested employee UserName

//firstname initial + LastName: dwelch

var newUserName = employee.FirstName.Substring(0, 1) + employee.LastName;

//create a new User ApplicationUser instance

var userAccount = new ApplicationUser()

{

UserName = newUserName,

Email = string.Format(STR\_EMAIL\_FORMAT, newUserName),

EmailConfirmed = true

};

userAccount.EmployeeID = employee.EmployeeId;

//create the Users record

IdentityResult result = this.Create(userAccount, STR\_DEFAULT\_PASSWORD);

//result hold the return value of the creation attempt

//if true, account was created,

//if false, an account already exists with that username

if (!result.Succeeded)

{

//name already in use

//get a UserName that is not in use

newUserName = VerifyNewUserName(newUserName);

userAccount.UserName = newUserName;

this.Create(userAccount, STR\_DEFAULT\_PASSWORD);

}

//create the staff role in UserRoles

this.AddToRole(userAccount.Id, SecurityRoles.Staff);

}

}

}

}

public string VerifyNewUserName(string suggestedUserName)

{

//get a list of all current usernames (customers and employees)

// that start with the suggestusername

//list of strings

//will be in memory

var allUserNames = from x in Users.ToList()

where x.UserName.StartsWith(suggestedUserName)

orderby x.UserName

select x.UserName;

//set up the verified unique UserName

var verifiedUserName = suggestedUserName;

//the following for() loop will continue to loop until

// an unsed UserName has been generated

//the condition searches all current UserNames for the

//currently generated verified used name (inside loop code)

//if found the loop will generate a new verified name

// based on the original suggest username and the counter

//This loop continues until an unused username is found

//OrdinalIgnoreCase : case does not matter

for (int i = 1; allUserNames.Any(x => x.Equals(verifiedUserName,

StringComparison.OrdinalIgnoreCase)); i++)

{

verifiedUserName = suggestedUserName + i.ToString();

}

//return teh finalized new verified user name

return verifiedUserName;

}

1. Create an ApplicationRoleManager class in the AppSecurity BLL. Add the following code to this class. Resolve any references.

#region Additional Namespaces

using Microsoft.AspNet.Identity.EntityFramework;

using Microsoft.AspNet.Identity;

using AppSecurity.DAL;

using AppSecurity.Entities;

#endregion

public class ApplicationRoleManager:RoleManager<IdentityRole>

{

private ApplicationUserManager UserManager { get; set; }

public ApplicationRoleManager() :

base(new RoleStore<IdentityRole>(new ApplicationDbContext()))

{

//needed to add this to get UserManager in ListAllRoles() to have a value.

UserManager = new ApplicationUserManager(new UserStore<ApplicationUser>(new ApplicationDbContext()));

}

public ApplicationRoleManager(ApplicationUserManager userManager) :

base(new RoleStore<IdentityRole>(new ApplicationDbContext()))

{

UserManager = userManager;

}

public void AddDefaultRoles()

{

foreach (string roleName in SecurityRoles.DefaultSecurityRoles)

{

// Check if it exists

if (!Roles.Any(r => r.Name == roleName))

{

this.Create(new IdentityRole(roleName));

}

}

}

}

1. Open your web application Global.asax file. Add the following lines to the Application\_Start method after the existing code. This will cause the application to add the default roles and employee users.

#region Additional Namespaces

using AppSecurity.BLL;

using AppSecurity.DAL;

using AppSecurity.Entities;

using Microsoft.AspNet.Identity.EntityFramework;

#endregion

var RoleManager = new ApplicationRoleManager();

RoleManager.AddDefaultRoles();

var UserManager = new ApplicationUserManager(new   
 UserStore<ApplicationUser>(new ApplicationDbContext()));

UserManager.AddDefaultUsers();

# Testing Login

1. Currently the Login.aspx window is setup for email address logins. We need to alter that to user UserName logins. In your web application, open the Login.aspx form. Select the Email Label and Textbox, copy and paste immediately after it.

<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>

Change to:

<asp:Label runat="server" AssociatedControlID="Email" CssClass="col-md-2 control-label">User Name</asp:Label>

<div class="col-md-10">

<asp:TextBox runat="server" ID="Email" CssClass="form-control" />

<asp:RequiredFieldValidator runat="server" ControlToValidate="Email"

CssClass="text-danger" ErrorMessage="The user name field is required." />

</div>

1. Add the following code to your About page Page\_Load event

if (!IsPostBack)

{

if (!Request.IsAuthenticated)

{

Response.Redirect("~/Account/Login.aspx");

}

else

{

if (!User.IsInRole(SecurityRoles.WebsiteAdmins))

{

Response.Redirect("~/Account/Login.aspx");

}

}

}

1. Add the following code to your Contact page Page\_Load event

if (!IsPostBack)

{

if (!Request.IsAuthenticated)

{

Response.Redirect("~/Account/Login.aspx");

}

else

{

if (!User.IsInRole(SecurityRoles.Staff))

{

Response.Redirect("~/Account/Login.aspx");

}

}

}

1. Run your application.
   1. Access your Home page or one of the sample pages. Now try either the About or Contact page. These pages require a log in.
   2. Access your About page. Notice you are restricted. Log on using AAdams with a password of Pa$$word1. You are still restricted. However, you can get to the Home page. About is restricted to a webmaster. Try WebMaster/Pa$$word1.
   3. Access your Contact page. As the webmaster you are restricted. The Contact page is for Staff. Try AAdams/Pa$$word1.