Architecture/Programming Guide

# Introduction

Purpose of this document is to explain the reason for inclusion of different files in the solution:

# Application File Explanation

Files:

|  |  |
| --- | --- |
| wwwroot  css/site.css  js/site.js  lib/bootstrap  lib/jquery  lib/jquery-validation  lib/jquery-validation-unobtrusive | Default with .NET Core 2 ASP.NET Identity 2 Solution |
| Controllers  AccountController  HomeController  ManageController | Default with .NET Core 2 ASP.NET Identity 2 Solution |
| Data  Migrations  ApplicationDbContext | Default with .NET Core 2 ASP.NET Identity 2 Solution |
| Extensions  EmailSenderExtensions  UrlHelperExtensions | Default with .NET Core 2 ASP.NET Identity 2 Solution |
| Models/AccountViewModels  ExternalLoginViewModel  ForgotPasswordViewModel  LoginViewModel  LoginWith2faViewModel  LoginWithRecoveryCodeViewModel  RegisterViewModel  ResetPasswordViewModel | Default with .NET Core 2 ASP.NET Identity 2 Solution |
| Models/ManageViewModels  ChangePasswordViewModel  EnableAuthenticatorViewModel  ExternalLoginsViewModel  IndexViewModel  RemoveLoginViewModel  SetPasswordViewModel  ShowRecoveryCodesViewModel  TwoFactorAuthenticationViewModel | Default with .NET Core 2 ASP.NET Identity 2 Solution |
| Models/ApplicationUser | Default with .NET Core 2 ASP.NET Identity 2 Solution |
| Models/ErrorViewModel | Default with .NET Core 2 ASP.NET Identity 2 Solution |
| Services  EmailSender  IEmailSender | Default with .NET Core 2 ASP.NET Identity 2 Solution |
| Views/Account  AccessDenied  ConfirmEmail  ExternalLogin  ForgotPassword  ForgotPasswordConformation  Lockout  Login  LoginWith2fa  LoginWithRecoveryCode  Register  ResetPassword  ResetPasswordConfirmation  SignedOut | Default with .NET Core 2 ASP.NET Identity 2 Solution |
| Views/Home  About  Contact  Index | Default with .NET Core 2 ASP.NET Identity 2 Solution |
| Views/Manage  \_Layout  \_ManageNav  \_StatusMessage  \_ViewImports  ChangePassword  Disable2fa  EnableAuthenticator  ExternalLogins  GenerateRecoveryCodes  Index  ManageNavPages  ResetAuthenticator  SetPassword  ShowRecoveryCodes  TwoFactorAuthentication | Default with .NET Core 2 ASP.NET Identity 2 Solution |
| Views/Shared  \_Layout  \_LoginPartial  \_ValidationScriptsPartial  Error | Default with .NET Core 2 ASP.NET Identity 2 Solution |
| Views/\_ViewImports | Default with .NET Core 2 ASP.NET Identity 2 Solution |
| Views/\_ViewStart | Default with .NET Core 2 ASP.NET Identity 2 Solution |
| Appsettings.json | Default with .NET Core 2 ASP.NET Identity 2 Solution |
| Bundleconfig.json | Default with .NET Core 2 ASP.NET Identity 2 Solution |
| Program.cs | Default with .NET Core 2 ASP.NET Identity 2 Solution |
| Startup.cs | Default with .NET Core 2 ASP.NET Identity 2 Solution |

# Dependencies

|  |  |
| --- | --- |
| IdentityModel3.0.0 | IdentityServer 4.0 |
| IdentityServer4.2.1.1 | IdentityServer 4.0 |
| IdentityServer4.AspNetIdentity.2.1.0 | IdentityServer 4.0 |
| BuildBundleMinifier | Used for bundling/minification which runs during project build |
|  |  |

# Data Access Layer

## Generating Data Access Layer:

When there are changes to the database that need to be modeled in the application, after changing the database structure, run this script to generate the code models from database:

Scaffold-DbContext "Server=.\SQLEXPRESS;Database=TRIBAL\_SVC\_PORTAL;Trusted\_Connection=True;" Microsoft.EntityFrameworkCore.SqlServer -OutputDir Data/Models -t T\_PRT\_CLIENT\_ROLES, T\_PRT\_CLIENTS, T\_PRT\_TENANT\_CLIENT\_ALIAS, T\_PRT\_TENANT\_USER\_CLIENT, T\_PRT\_TENANT\_USERS, T\_PRT\_TENANTS, T\_OE\_SYS\_LOG -f -Context "ApplicationDbContextTemp"

After running, copy the red code into the generated ApplicationDBContext file:

using Microsoft.AspNetCore.Identity; //add each time

using Microsoft.AspNetCore.Identity.EntityFrameworkCore; //add each time

using Microsoft.Extensions.Configuration; //add each time

using System.IO; //add each time

namespace TribalSvcPortal.Data.Models

{

public partial class ApplicationDbContext : IdentityDbContext<ApplicationUser> //modify each time

{

public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)//add each time

: base(options)

{

}

//generated stuff here

public virtual DbSet<TOeSysLog> TOeSysLog { get; set; }

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

if (!optionsBuilder.IsConfigured)

{

var config = new ConfigurationBuilder()

.SetBasePath(Directory.GetCurrentDirectory())

.AddJsonFile("appsettings.json")

.Build();

optionsBuilder.UseSqlServer(config.GetConnectionString("DefaultConnection"));

}

}

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

base.OnModelCreating(modelBuilder); //add each time

modelBuilder.Entity<IdentityRole>().ToTable("T\_PRT\_ROLES"); //add each time

modelBuilder.Entity<IdentityUserToken<string>>().ToTable("T\_PRT\_USER\_TOKENS"); //add each time

modelBuilder.Entity<ApplicationUser>().ToTable("T\_PRT\_USERS"); //add each time

modelBuilder.Entity<IdentityRoleClaim<string>>().ToTable("T\_PRT\_ROLE\_CLAIMS"); //add each time

modelBuilder.Entity<IdentityUserClaim<string>>().ToTable("T\_PRT\_USER\_CLAIMS"); //add each time

modelBuilder.Entity<IdentityUserLogin<string>>().ToTable("T\_PRT\_USER\_LOGINS"); //add each time

modelBuilder.Entity<IdentityUserRole<string>>().ToTable("T\_PRT\_USER\_ROLES"); //add each time

//generated stuff here

}

}

}

# Presentation Layer

## Injecting Javascript into Header

When constructing a view, use the following code to inject javascript that appears in the header of a document. Otherwise javascript will appear at end of DOM

@section HeaderJS

{

<link href="@Url.Content("~/Content/jstree/themes/proton/style.css")" rel="stylesheet" />

<script src="@Url.Content("~/Content/jstree/jstree.min.js")"></script>

<script>

$(function () {

//stuff

});

</script>

}

## Injecting Custom CSS Styles into Header

When constructing a view, use the following code to inject custom css styling that appears in the header of a document. Otherwise javascript will appear at end of DOM