**SAM (System Access Management) Tool**

SAM Tool consist of user management,role management and access management for each application.

STEP1. Run the SAMScripts.sql in the new database ,which will create required tables and procedures for SAM tool functioning.it also creates some default table entries

STEP2.

In order to include SAM Tool in new project, we have to refer three assemblies from SAMTool namely-:

SAMTool.BusinessServices.dll

SAMTool.DataAccessObject.dll

SAMTool.RepositoryServices.dll

Add above 3 DLLS to new projects UI layer

STEP3

Create ViewModels in new project for SAMTool

public class LoginViewModel

{

[Required(ErrorMessage = "Please enter login name")]

//[Display(Name = "Login Name")]

[StringLength(250)]

public string LoginName { get; set; }

[Required(ErrorMessage = "Password is required.")]

//[Display(Name = "Password")]

[StringLength(250)]

public string Password { get; set; }

public bool IsFailure { get; set; }

public string Message { get; set; }

}

public class UserViewModel

{

public Guid? ID { get; set; }

[Required(ErrorMessage = "Please enter user name")]

[Display(Name = "User Name")]

public string UserName { get; set; }

[Required(ErrorMessage = "Please enter login name")]

[Display(Name = "Login Name")]

public string LoginName { get; set; }

[Display(Name = "Active")]

public bool Active { get; set; }

[Required(ErrorMessage = "Please enter email")]

[Display(Name = "Email")]

public string Email { get; set; }

[Display(Name = "Password")]

[RegularExpression(@"^(?=.\*[A-Za-z])(?=.\*\d)(?=.\*[$@$!%\*#?&])[A-Za-z\d$@$!%\*#?&]{6,}$", ErrorMessage = "should have minimum 6 Char, one alphabet,one numeric and one special character")]

[StringLength(250, MinimumLength = 6, ErrorMessage = "{0} should be minimum 6 Char")]

public string Password { get; set; }

[Display(Name = "Confirm Password")]

[RegularExpression(@"^(?=.\*[A-Za-z])(?=.\*\d)(?=.\*[$@$!%\*#?&])[A-Za-z\d$@$!%\*#?&]{6,}$", ErrorMessage = "should have minimum 6 Char, one alphabet,one numeric and one special character")]

[Compare("Password", ErrorMessage = "Password and Confirmation Password must match.")]

[StringLength(250, MinimumLength = 6, ErrorMessage = "{0} should be minimum 6 Char")]

public string ConfirmPassword { get; set; }

public string RoleCSV { get; set; }

public string RoleIDCSV { get; set; }

[Display(Name = "Select Roles")]

public List<RolesViewModel> RoleList { get; set; }

public CommonViewModel commonDetails { get; set; }

}

public class SysMenuViewModel

{

public Guid ID { get; set; }

public string LinkName { get; set; }

public string LinkDescription { get; set; }

public string Controller { get; set; }

public string Action { get; set; }

public string Type { get; set; }

public int Order { get; set; }

}

public class SAMPanelViewModel

{

public List<SysMenuViewModel> \_LHSSysMenuViewModel { get; set; }

public List<SysMenuViewModel> \_RHSSysMenuViewModel { get; set; }

}

public class RolesViewModel

{

public Guid? ID { get; set; }

[Required(ErrorMessage = "Please Select Application Name")]

public Guid? AppID { get; set; }

public string ApplicationName { get; set; }

[Required(ErrorMessage = "Please enter Role Name")]

[Display(Name = "Role Name")]

public string RoleName { get; set; }

[Required(ErrorMessage = "Please enter Role Description")]

[Display(Name = "Role Description")]

public string RoleDescription { get; set; }

public List<SelectListItem> ApplicationList { get; set; }

public CommonViewModel commonDetails { get; set; }

public string PathToAttachment { get; set; }

public HttpPostedFile FileUpload { get; set; }

public string Name1 { get; set; }

}

public class ApplicationViewModel

{

public Guid ID { get; set; }

[Required(ErrorMessage = "Please Enter Application name")]

[Display(Name = "Application Name")]

public string Name { get; set; }

public CommonViewModel commonDetails { get; set; }

}

public class AppObjectViewModel

{

public Guid ID { get; set; }

public Guid AppID { get; set; }

public string AppName { get; set; }

[Required(ErrorMessage = "Please Enter Object name")]

[Display(Name = "Object Name")]

public string ObjectName { get; set; }

public CommonViewModel commonDetails { get; set; }

public List<SelectListItem> ApplicationList { get; set; }

}

public class AppSubobjectViewmodel

{

public Guid ID { get; set; }

public Guid AppID { get; set; }

[Required(ErrorMessage = "Please Enter Sub-object name")]

public Guid ObjectID { get; set; }

[Display(Name = "Sub Object Name")]

[Required(ErrorMessage = "Please Enter Sub-object name")]

public string SubObjName { get; set; }

public string AppName { get; set; }

public string ObjectName { get; set; }

public CommonViewModel commonDetails { get; set; }

public List<SelectListItem> ApplicationList { get; set; }

public List<SelectListItem> ObjectList { get; set; }

}

public class CommonViewModel

{

public string CreatedBy { get; set; }

public DateTime CreatedDate { get; set; }

public string CreatedDatestr { get; set; }

public string UpdatedBy { get; set; }

public DateTime UpdatedDate { get; set; }

public string UpdatedDateString { get; set; }

}

public class ManageAccessViewModel

{

public Guid ID { get; set; }

public Guid ObjectID { get; set; }

public Guid RoleID { get; set; }

public bool Read { get; set; }

public bool Write { get; set; }

public bool Delete { get; set; }

public bool Special { get; set; }

public CommonViewModel commonObj { get; set; }

public AppObjectViewModel AppObjectObj { get; set; }

public List<ManageAccessViewModel> ManageAccessList { get; set; }

public List<SelectListItem> ApplicationList { get; set; }

public List<SelectListItem> RoleList { get; set; }

}

public class ManageSubObjectAccessViewModel

{

public Guid ID { get; set; }

public Guid SubObjectID { get; set; }

public Guid RoleID { get; set; }

public bool Read { get; set; }

public bool Write { get; set; }

public CommonViewModel commonObj { get; set; }

public AppObjectViewModel AppObjectObj { get; set; }

public AppSubobjectViewmodel AppSubObjectObj { get; set; }

public ManageAccessViewModel ManageAccessObj { get; set; }

public List<ManageSubObjectAccessViewModel> ManageSubObjectAccessList { get; set; }

public List<SelectListItem> ObjectList { get; set; }

public List<SelectListItem> RoleList { get; set; }

}

public class PrivilegesViewModel

{

public Guid ID { get; set; }

[Required(ErrorMessage = "Please select Role Name")]

[Display(Name = "Role Name")]

public Guid RoleID { get; set; }

[Required(ErrorMessage = "Please select Application Name")]

[Display(Name = "Application Name")]

public Guid AppID { get; set; }

[Required(ErrorMessage = "Please select Module Name")]

[Display(Name = "Module Name")]

public string ModuleName { get; set; }

[Display(Name = "Description")]

public string AccessDescription { get; set; }

public CommonViewModel commonDetails { get; set; }

public string ApplicationName { get; set; }

public string RoleName { get; set; }

public List<SelectListItem> RoleList { get; set; }

public List<SelectListItem> ApplicationList { get; set; }

}

After creating above view model we need to map these viewmodels to SAMTool DTO models

So we use automapper.in the mappingConfig class register maps method in app\_start folder

//\*\*\*\*\*SAMTOOL MODELS

config.CreateMap<LoginViewModel, SAMTool.DataAccessObject.DTO.User>().ReverseMap();

config.CreateMap<UserViewModel, SAMTool.DataAccessObject.DTO.User>().ReverseMap();

config.CreateMap<SysMenuViewModel, SAMTool.DataAccessObject.DTO.SysMenu>().ReverseMap();

config.CreateMap<RolesViewModel, SAMTool.DataAccessObject.DTO.Roles>().ReverseMap();

config.CreateMap<ApplicationViewModel, SAMTool.DataAccessObject.DTO.Application>().ReverseMap();

config.CreateMap<AppObjectViewModel, SAMTool.DataAccessObject.DTO.AppObject>().ReverseMap();

config.CreateMap<AppSubobjectViewmodel, SAMTool.DataAccessObject.DTO.AppSubobject>().ReverseMap();

config.CreateMap<CommonViewModel, SAMTool.DataAccessObject.DTO.Common>().ReverseMap();

config.CreateMap<ManageAccessViewModel, SAMTool.DataAccessObject.DTO.ManageAccess>().ReverseMap();

config.CreateMap<ManageSubObjectAccessViewModel, SAMTool.DataAccessObject.DTO.ManageSubObjectAccess > ().ReverseMap();

config.CreateMap<PrivilegesViewModel, SAMTool.DataAccessObject.DTO.Privileges>().ReverseMap();

//\*\*\*\*SAMTOOL MODELS

STEP4.

Dependency injection for SAMTool Bussines and SAMTool Repository.

Inorder to use SAMTool service we need to inject into our active assemblies. For this register it in UnityConfig.cs RegisterTypes method,paste following code in

//SAMTOOL DEPENDENCIES

container.RegisterTypes(AllClasses.FromLoadedAssemblies().Where(t => t.Namespace == "SAMTool.BusinessServices.Services"), WithMappings.FromMatchingInterface, WithName.Default, WithLifetime.Transient);

container.RegisterTypes(AllClasses.FromLoadedAssemblies().Where(t => t.Namespace == "SAMTool.RepositoryServices.Services"), WithMappings.FromMatchingInterface, WithName.Default, WithLifetime.Transient);

Now onwards SAMTool services can be accessed

STEP5.

Create controllers and views using above Viewmodel

Account Controller,User,SAMPanel,Role,Application,AppObject,ManageAccess,previlegs.

Each controller must use SAMTool methods

For example

public class ApplicationController : Controller

{

private IApplicationBusiness \_applicationBusiness;

public ApplicationController(IApplicationBusiness applicationBusiness)

{

\_applicationBusiness = applicationBusiness;

}

}

Here application controller’s constructor make use of SAMTool’s IApplicationBusiness

STEP6.

Create login method, which uses forms authentication cookie

As follows

[HttpPost]

[ValidateAntiForgeryToken]

[AllowAnonymous]

public ActionResult Login(LoginViewModel loginvm)

{

UserViewModel uservm = null;

try

{

if (!ModelState.IsValid)

{

loginvm.IsFailure = true;

loginvm.Message = \_const.LoginFailed;

return View("Index", loginvm);

}

uservm = Mapper.Map<User, UserViewModel>(\_userBusiness.CheckUserCredentials(Mapper.Map<LoginViewModel, User>(loginvm)));

if (uservm != null)

{

if (uservm.RoleList == null || uservm.RoleList.Count == 0 && string.IsNullOrEmpty(uservm.RoleIDCSV))

{

loginvm.IsFailure = true;

loginvm.Message = \_const.LoginFailedNoRoles;

return View("Index", loginvm);

}

FormsAuthenticationTicket ticket = new FormsAuthenticationTicket(1, uservm.UserName, DateTime.Now, DateTime.Now.AddHours(24), true, uservm.RoleCSV);

string encryptedTicket = FormsAuthentication.Encrypt(ticket);

Response.Cookies.Add(new HttpCookie(FormsAuthentication.FormsCookieName, encryptedTicket));

//session setting

UA ua = new UA();

ua.UserName = uservm.LoginName;

ua.AppID = AppID;

Session.Add("TvmValid", ua);

if (uservm.RoleCSV.Contains("SAdmin") || uservm.RoleCSV.Contains("CEO"))

{

return RedirectToAdminDashboard();

}

else {

return RedirectToLocal();

}

}

else

{

loginvm.IsFailure = true;

loginvm.Message = \_const.LoginFailed;

return View("Index", loginvm);

}

}

catch (Exception ex)

{

throw ex;

}

}

To Implement Authentication and Authorization, create a class

public class AuthSecurityFilter : ActionFilterAttribute, IAuthenticationFilter, IAuthorizationFilter.

For this purpose we use MVC filters such as ActionFilterAttribute, IAuthenticationFilter, IAuthorizationFilter.

We must implement IAuthenticationFilter’s two methods. They are

OnAuthentication method

OnAuthenticationChallenge method

OnAuthentication method triggers before every action call, here we check for session validation and cookie validation.

OnAuthenticationChallenge method invokes after executing action method. Here we define the result based on authentication challenge

IAuthorizationFilter interface has one method to implement that is OnAuthorization

OnAuthorization method authorizes current request, In SAM Tool we get access code for a particular object and check whether it has required privileges and stores it in session as Permission type.

Create it under project directly, AuthSecurityFilter Implementation as follows

public class AuthSecurityFilter : ActionFilterAttribute, IAuthenticationFilter, IAuthorizationFilter

{

public string LoggedUserName { get; set; }

public string ProjectObject { get; set; }

public string Mode { get; set; }

[Dependency]

public IUserBusiness \_userBusiness { get; set; }

public void OnAuthentication(AuthenticationContext filterContext)

{

if ((filterContext.HttpContext.Session == null) || (filterContext.HttpContext.Session["TvmValid"] == null))

{

filterContext.Result = new HttpUnauthorizedResult();

return;

}

var authCookie = filterContext.HttpContext.Request.Cookies[FormsAuthentication.FormsCookieName];

if (authCookie == null)

{

// Unauthorized

filterContext.Result = new HttpUnauthorizedResult();

return;

}

// Get the forms authentication ticket.

FormsAuthenticationTicket authTicket = FormsAuthentication.Decrypt(authCookie.Value);

if (authTicket == null)

{

filterContext.Result = new HttpUnauthorizedResult(); // mark unauthorized\*/

}

else

{

filterContext.HttpContext.User = new System.Security.Principal.GenericPrincipal(

new System.Security.Principal.GenericIdentity(authTicket.Name, "Forms"), authTicket.UserData.Split(',').Select(t => t.Trim()).ToArray());

UA \_ua = (UA)filterContext.HttpContext.Session["TvmValid"];

AppUA appUA = new AppUA();

appUA.RolesCSV = authTicket.UserData;

appUA.UserName = \_ua.UserName;

LoggedUserName = appUA.UserName;

SPAccounts.DataAccessObject.DTO.Common common = new SPAccounts.DataAccessObject.DTO.Common();

appUA.DateTime = common.GetCurrentDateTime();

appUA.AppID = \_ua.AppID;

filterContext.HttpContext.Session.Add("AppUA", appUA);

}

}

public void OnAuthenticationChallenge(AuthenticationChallengeContext filterContext)

{

if (filterContext.Result == null || filterContext.Result is HttpUnauthorizedResult)

{

if (filterContext.HttpContext.Request.IsAjaxRequest())

{

JsonResult json = new JsonResult

{

Data = new { Result = "UNAUTH", Message = "Authorization failed!." }

};

//filterContext.Result= new JsonResult() { Data = new { Result = "UNAUTH", Message = "" }, JsonRequestBehavior = JsonRequestBehavior.AllowGet };

filterContext.Result = new JsonResult() { Data = new JavaScriptSerializer().Serialize(json.Data), JsonRequestBehavior = JsonRequestBehavior.AllowGet };

//filterContext.Result = new JsonResult() { Data = JsonConvert.SerializeObject(new { Result = "UNAUTH", Message = "" }), JsonRequestBehavior = JsonRequestBehavior.AllowGet };

}

else

{

filterContext.Result = new RedirectToRouteResult(new System.Web.Routing.RouteValueDictionary() { { "controller", "Account" }, { "action", "Index" } });

}

}

}

public void OnAuthorization(AuthorizationContext filterContext)

{

Permission \_permission =null;

\_permission = \_userBusiness.GetSecurityCode(LoggedUserName, ProjectObject);

if (\_permission.AccessCode.Contains(Mode))

{

//Allows Permission

filterContext.HttpContext.Session.Add("UserRights", \_permission);

}

else

{

if (filterContext.HttpContext.Request.IsAjaxRequest())

{

filterContext.Result = new HttpUnauthorizedResult();

}

else

{ //unauthorized page show here

filterContext.Result = new RedirectToRouteResult(new System.Web.Routing.RouteValueDictionary() { { "controller", "Account" }, { "action", "NotAuthorized" } });

}

}

}

}

STEP 7.

Decorate each action method with filter attribute as follows

[AuthSecurityFilter(ProjectObject = "Application", Mode = "R")]

public ActionResult Index()

{

return View();

}

ProjectObject is the name of the object created in SAMTool object.

Mode Can be(R=Read,W=Write,D=Delete). Each authorization validate whether security code consist of any of the code.

Authorized action method can access session values by

AppUA \_appUA = Session["AppUA"] as AppUA;

Var loggedusername=\_appUA.UserName;