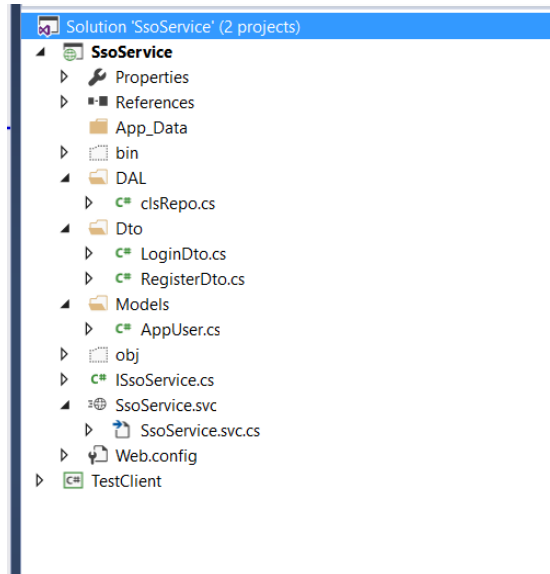


Single Sign On Service

- A web service name **SsoService** was developed using Microsoft WCF to act as the Identity Provider and single sign on authentication service for clients that required authentication to access it's the protected resources.
- The structure of the Project:



- clsRepo class consists of the code dealing with database, no ORM was used here.
- The service consists of 2 service methods named **Login** and **Register**, clients can consume these two service methods to register a new user or login and authenticated.

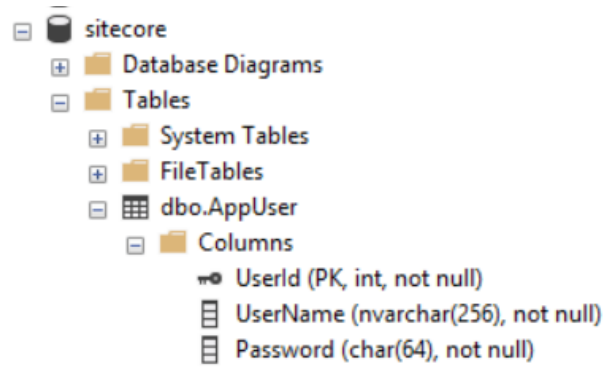
```
[ServiceContract]
public interface ISsoService
{
    [OperationContract]
    LoginDto Login(string userName, string password);

    [OperationContract]
    RegisterDto Register(string userName, string password);
}
```

-
-
- The service connected to MSSQL server database to store the user name and password credential, the connection string was set in web config file.

```
<appSettings>
  <add key="aspnet:UseTaskFriendlySynchronizationContext" value="true" />
  <add key="strDBconn" value="Data Source=DESKTOP-FR7I756;Initial Catalog=sitecore;Integrated Security=True" />
</appSettings>
```

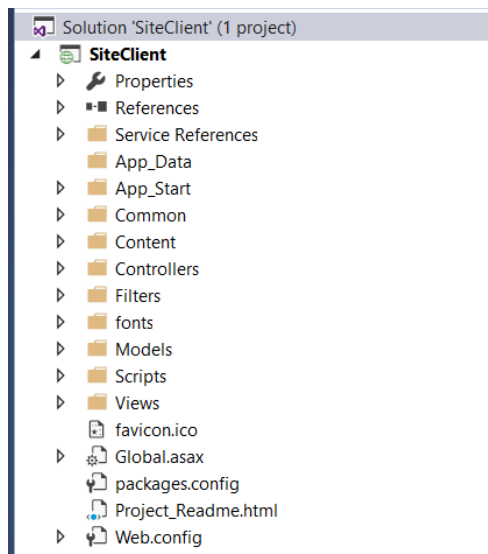
- For demonstration purpose, the database only consists of one table name “AppUser” with column UserId, UserName and Password where userId is the unique primary key of the table.



- The password should be encrypted before store into the database in real world case, I had omitted this step here for the sake of brevity.

Client

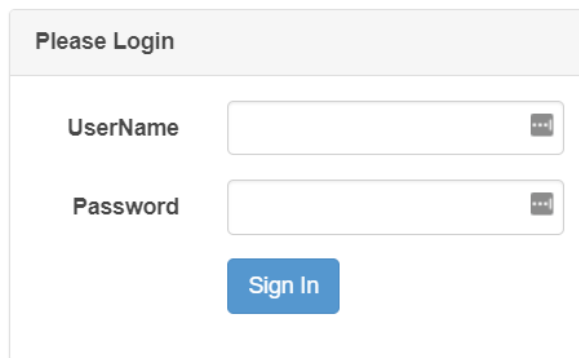
- A web application (SiteClient) was developed to consume the SsoService. It is a simple .net mvc web application that use the MVC web application template that come with visual studio.
- Structure of the solution as show at the snapshot below:



- User need to be authenticated before it can view the resources of the application.
- To protect the action of a controller from anonymous access, the controller needed to be derived from a BaseController.

```
public class BaseController:Controller
{
    protected override void OnActionExecuting(ActionExecutingContext filterContext)
    {
        var notRequiredSignIn = IsNotNeedSignIn(filterContext);
        if (!notRequiredSignIn)
        {
            if (SessionMng.CurrentAccount == null)
            {
                var returnUrl = "";
                if (!Request.IsAjaxRequest())
                {
                    returnUrl = HttpUtility.UrlEncode(filterContext.HttpContext.Request.Url.OriginalString);
                }
                var redirectUrl = string.IsNullOrEmpty(returnUrl)
                    ? filterContext.HttpContext.Request.ApplicationPath
                    : "/Account/Login" + "?returnurl=" + returnUrl;
                filterContext.Result = new RedirectResult(redirectUrl);
                return;
            }
        }
        base.OnActionExecuting(filterContext);
    }
}
```

- I had wrote an overriding method of a MVC controller base class, **OnActionExecuting**, where this method will be invoked before the controller action methods being invoked.
- This overriding method will check its http context session for a key I kept in the session when a user was successfully login and authenticated. If there is no such key, it will redirect the call to login page, otherwise it will continue to serve the protected resources.



- The login action will consume the ssoService to perform login authentication, if the user name and password were correct, the key will be saved in the http context session and redirect user to the home page.

Welcome to my system.

Sign out

-