Single Sign On

Introduction:

Single Sign-On (SSO) is an authentication technique where the user uses one set of login credentials to access multiple web applications.

### About Platform:

. Asp.net core MVC  
. Microsoft Identity Login  
. Dotnet core   
. SQL LocalDB

### Core Concept:

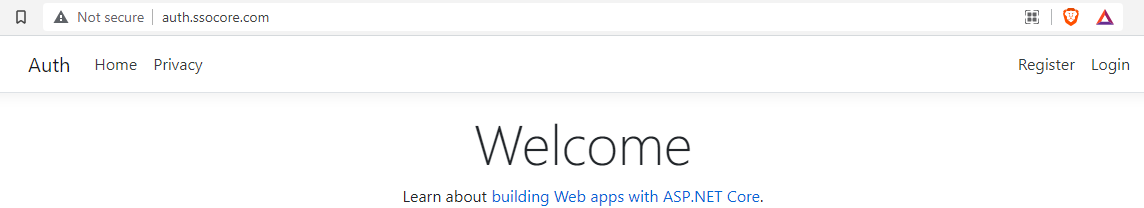
In the dotnet core, by sharing authentication cookie we can achieve SSO for the subdomains.  
  
A subdomain is a domain which part of another domain.  Additional name prefixed to a domain that URL can be a subdomain. For example "http://auth.ssocore.com/", “http://desktop.ssocore.com/”, “http://mobile.ssocore.com/” where “ssocore.com” is a domain.

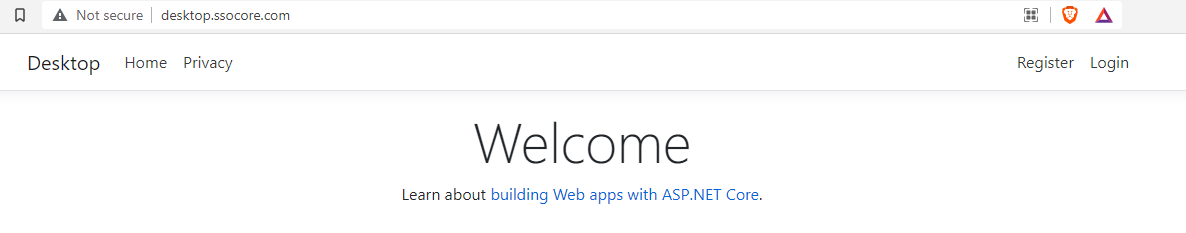
### Create and host MVC Application:

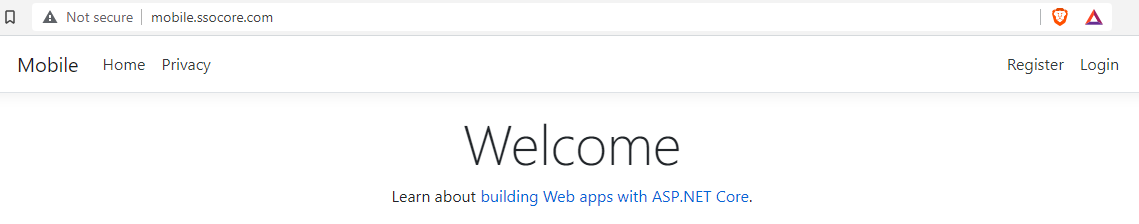
Create an Asp.net MVC Core application by selecting a login individual user template from a visual studio. For this application use, LocalDB as a database then host this application to local IIS, bind "http://auth.ssocore.com/", “http://desktop.ssocore.com/” and “http://mobile.ssocore.com/” as a domain to test sharing authentication cookie.

### Test Application Hosting In IIS:

After hosting the application successfully, navigate to the application using "http://auth.ssocore.com/", “http://desktop.ssocore.com/” and “http://mobile.ssocore.com/”.







And then register a user to login in any one of them.

### Register Data Protection API Service:

Now open a startup.cs file and in "ConfigureServie()" method we need to register Data Protection API service as below in "http://auth.ssocore.com/" Project:

services.AddDataProtection()

.PersistKeysToFileSystem(GetKyRingDirectoryInfo())

.SetApplicationName("SharedCookieApp");

services.ConfigureApplicationCookie(options =>

{

options.Cookie.Name = ".AspNet.SharedCookie";

options.Cookie.Domain = ".ssocore.com";

});

### Dynamically Create Data Protection API Key:

Now we need to provide DirectoryInfo to save the Data Protection API key. To do that add the following piece of code as follows only in any one of the project "http://auth.ssocore.com/".

private DirectoryInfo GetKyRingDirectoryInfo()

{

string applicationBasePath = System.AppContext.BaseDirectory;

DirectoryInfo directoryInof = new DirectoryInfo(applicationBasePath);

string keyRingPath = Configuration.GetSection("AppKeys").GetValue<string>("keyRingPath");

do

{

directoryInof = directoryInof.Parent;

DirectoryInfo keyRingDirectoryInfo = new DirectoryInfo($"{directoryInof.FullName}{keyRingPath}");

if (keyRingDirectoryInfo.Exists)

{

return keyRingDirectoryInfo;

}

}

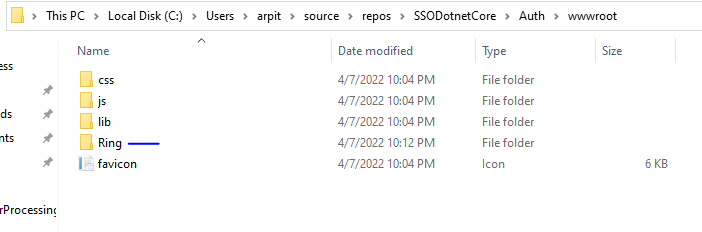
while (directoryInof.Parent != null);

throw new Exception($"key ring path not found");

}

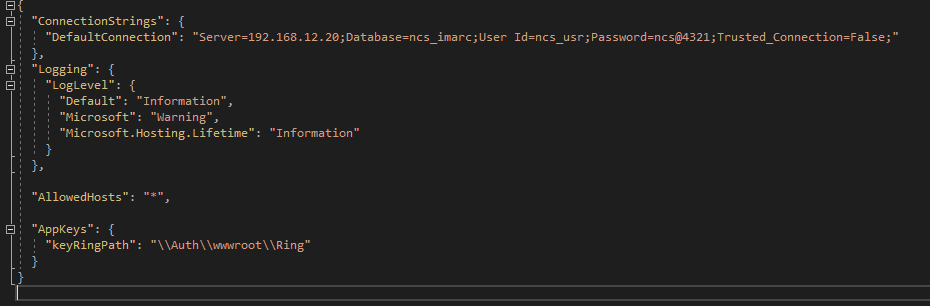
here I'm checking the folder path until able to found the path. You can directly assign the physical path, instead of checking for folder path by leveling up a parent folder on every iteration until its found. And Create a “Ring” Folder inside project of "http://auth.ssocore.com/" at path.

C:\Users\arpit\source\repos\SSODotnetCore\Auth\wwwroot\Ring



### Data Protection API Key Path:

Now add a new folder in wwwroot folder and name it as "Ring". Where Data Protection Key gets created and saved. This key will be used to validate cookies as well as used in the encryption of sensitive data in the cookie.  
  
Add the keyring folder path in the appSetting.json file as below



In rest of the 2 Project(“http://desktop.ssocore.com/” and “http://mobile.ssocore.com/”) Project you just need copy the path of Ring Folder of the first("http://auth.ssocore.com/") Project as shown below.

services.AddDataProtection()

.PersistKeysToFileSystem(new System.IO.DirectoryInfo("C:\\Users\\arpit\\source\\repos\\SSODotnetCore\\Auth\\wwwroot\\Ring"))

.SetApplicationName("SharedCookieApp");

services.ConfigureApplicationCookie(options =>

{

options.Cookie.Name = ".AspNet.SharedCookie";

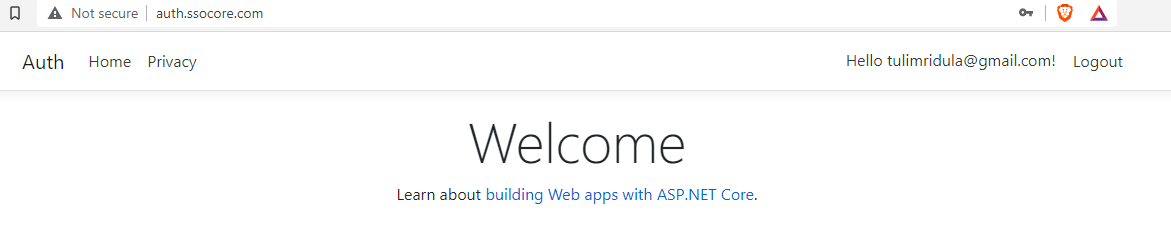
options.Cookie.Domain = ".ssocore.com";

options.Cookie.Path = "/";

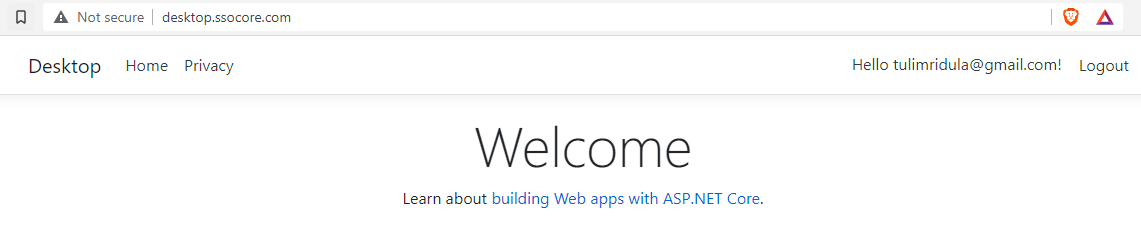
});

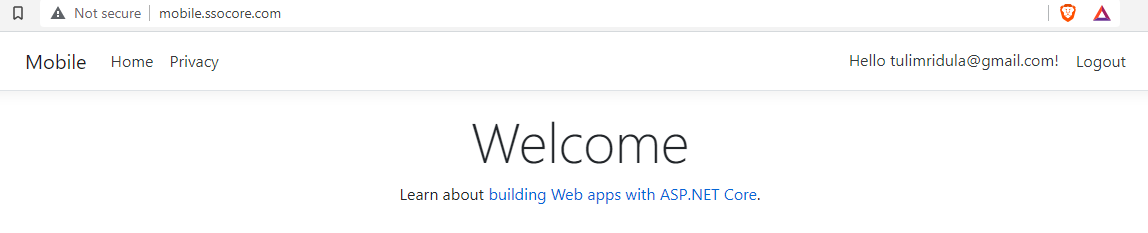
### Test Login Cookie Sharing:

Now navigate to "http://auth.ssocore.com/" and get logged in as below



Now if you navigate to rest of Url’s you find it as logged in already with same user.





Go to a "Ring" folder physical location used in our application. we can find the Data Protection key will get created which used to encrypting and validating our login cookie. Key sample as below



### Summary:

SSO (Single Sign-On) was implemented successfully with subdomains of an application. Cookie encryption and decryption were done with the help of the Data Protection API provided by ASP.NET core. A similar cookie sharing concept can be used for an application runs under another application in IIS[.](https://learnmoreseekmore.blogspot.com/2019/09/part-2-share-authentication-cookie.html)