WorkshopPLUS – Modern Authentication and Authorization

Developing applications

Student Lab Manual

Instructor Edition (Book Title Hidden Style)

Version 1.1

**Conditions and Terms of Use**

**Microsoft Confidential - For Internal Use Only**

This training package is proprietary and confidential, and is intended only for uses described in the training materials. Content and software is provided to you under a Non-Disclosure Agreement and cannot be distributed. Copying or disclosing all or any portion of the content and/or software included in such packages is strictly prohibited.

The contents of this package are for informational and training purposes only and are provided "as is" without warranty of any kind, whether express or implied, including but not limited to the implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

Training package content, including URLs and other Internet Web site references, is subject to change without notice. Because Microsoft must respond to changing market conditions, the content should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication. Unless otherwise noted, the companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred.

© 2016 Microsoft Corporation. All rights reserved.

**Copyright and Trademarks**

© 2016 Microsoft Corporation. All rights reserved.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

For more information, see Use of Microsoft Copyrighted Content at  
[*http*://www.microsoft.com/about/legal/permissions/](http://www.microsoft.com/about/legal/permissions/)

Microsoft®, Internet Explorer®, and Windows® are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Other Microsoft products mentioned herein may be either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. All other trademarks are property of their respective owners.

Contents

[Introduction: 6](#_Toc508614491)

[Exercise 1: Create a Web API application 7](#_Toc508614492)

[Exercise 2 – Create a Web UI application 8](#_Toc508614493)

[Exercise 3 – Add authorization to the Web API project 10](#_Toc508614494)

[Exercise 4 – Add access token request to the WebUI project 10](#_Toc508614495)

[Exercise 5 – Add application roles 12](#_Toc508614496)

# Introduction:

#### Objectives

The purpose of this lab is to help the student acquire hands on experience of developing an application using the common identity infrastructure. By the end of the lab the student will have developed an application consisting of the following components and functionality:

1. A Web UI application serving html pages, able to authenticate the user.
2. The Web UI application will use role-based access control.
3. A Web API application providing a REST interface used by the Web UI application. Access to the APIs will be restricted to authorized users.

#### Prerequisites (if applicable)

VS.NET 2015 or 2017

Web browser

#### Estimated time to complete this lab

2.0h

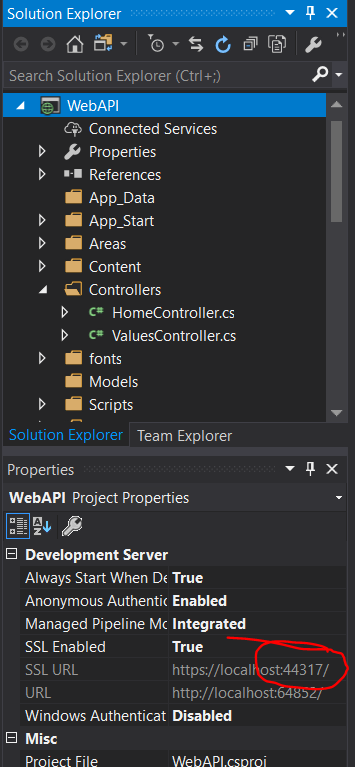
# Exercise 1: Create a Web API application

|  |  |  |
| --- | --- | --- |
|  | **Procedure** | **Purpose** |
| 1.0 | In VS.NET create a new, empty solution. Call it AADLab. (In New Project dialog scroll the left hand panel to Other Project Types->Visual Studio Solutions->Blank Solution or search for Blank Solutions in using the search text entry field) | Create solution for our projects |
| 1.1 | Add a new C# ASP.NET API project (.NET 4.x):   1. Web project 2. ASP.NET Web Application 3. Name project WebAPI 4. Press OK | Initiate creation of a simple Web API application. |
| 1.2 | Select ASP.NET Web API template | Type of web application |
| 1.3 | Click on the Change Authentication button  Select Work or School Accounts  Use ‘Cloud – Single organization’, enter your AAD domain name and click OK. | Configure token validation using AAD |
| 1.4 | Open the ValuesController.cs file and comment out the [Authorize] attribute (line 10):  // [Authorize] | Allow un-authorized calls to the API for the time being. |

# Exercise 2 – Create a Web UI application

|  |  |  |
| --- | --- | --- |
|  | Procedure | Purpose |
| 2.1 | Add a new C# ASP.NET UI project (.NET 4.x):   1. Web project 2. ASP.NET Web Application 3. Name project WebUI   Press OK   1. Select the MVC template | Initiate creation of a simple Web UI application. |
| 2.2 | Click on the Change Authentication button  Select Work or School Accounts  Use ‘Cloud – Single organization’, enter your AAD domain name and click OK. | Configure use of the OpenIDConnect protocol using AAD |
| 2.3 | Open the HomeController in the Web UI application and replace its About method with contents of InitialAbout.txt.  Replace the <webapi port no> with the port number of your WebAPI project (see instructions below). | Call Web API from the UI app |
| 2.4 | Right-click on the solution in the Solution Explorer and select Properties. Select Multiple Startup Project in the dialog and choose Start option for both projects. | Set both projects to run in debug simultaneously |
| 2.5 | Press F5 and click on the About option when the browser shows your page. | Test initial solution, calling the API without a token. |

To determine that press F4 to ensure the Properties View is open, select the API project in the solution explorer. You should see the port number in the SSL Url property:



# Exercise 3 – Add authorization to the Web API project

|  |  |  |
| --- | --- | --- |
|  | Procedure | Purpose |
| 3.1 | Uncomment the [Authorize] attribute in the WebAPI project’s ValuesController.  Run your solution and select the About option. You should get an Unauthorized response. | Change API to require an OAuth2 authorization token. |

# Exercise 4 – Add access token request to the WebUI project

|  |  |  |
| --- | --- | --- |
| 4.1 | Still in the portal, find the WebUI project:   1. open its *Settings* 2. open *Required Permissions.* 3. Select *Add* to add a new permission 4. Click *Select API*. 5. Type WebAPI in the Search filed 6. Select the WebAPI project from list of available APIs 7. Click *Select* 8. In the Select permissions tab, select *Access WebAPI* (delegated permission) | Configure Azure AD to allow the WebUI to obtain tokens on behlaf of the user to call the WebAPI. |
| 4.2 | Select the WebUI project in the VS.NET solution explorer. Open its *Startup.auth.cs* file – you will see how it changes as the result of what you do below.   1. From the VS.NET *Project* menu select the *Add connect services* option. 2. Select *Authentication with Azure Active Directory* 3. Click *Next* several times till you get to the *Directory Access* option. Check the *Read directory* data button. | Add OAuth2 access token request to the UI project (until now the project was only configured to obtain an id\_token). |
| 4.3 | In the Azure Portal, *App Registrations* blade, find the WebAPI application and open its *Settings* and then *Properties*. Copy the *App ID URI* to clipboard.  Paste it instead of graphResourceId value in the Startup.auth.cs class of the WebUI project. Rename the graphResourceId to resourceId:  string resourceId = "https://<dir>.onmicrosoft.com/WebUI"; | This is the ID we will use to identify the API app in calls to AAD (Application ID GUID is an alternative). |
| 4.4 | Change all the private fields in the Startup.auth.cs class (of the WebUI project) to public. (Note: in VS.NET Ctrl+Alt will allow you to select text in multiple lines). | We will need to use these fields when getting the access token in our controller class. |
| 4.3 | Put a breakpoint on the line immediately after the AcquireTokenByAuthorizationCode method and run the application. Sign in and when you hit the break inspect the result field. Copy the AccessToken from it and view it in [http://](NULL)jwt.ms | Test whether you are getting the right token. It should have your WebAPI in the aud claim. |
| 4.4 | Replace the HomeController About method with contents of the Snippets.FinalAbout.txt file. Modify its <<dir>> and <webapi port no> placeholders with your directory name and your WebAPI project port number. | Add token acquisition to access WebAPI from WebUI as client. |
| 4.5 | In the WebAPI project replace the contents of the ValuesController Get method with those in Snippets/FinalGetAPI.txt | Return something that depends on the contents of the token. |
| 4.6 | Run both projects and test whether you are getting the correct response from the WebAPI and displaying it when selecting the About method. | Success!!! |

# Exercise 5 – Add application roles

|  |  |  |
| --- | --- | --- |
| 4.1 | In [https:/portal.azure.com](https://manage.windowsazure.com) locate your AAD tenant and go to the *App Registration* tab. Find your Web UI application. Select its Manifest button to Edit it. | Modify roles property of an application. |
| 4.2 | Replace “appRoles”: [] with contents of the appRoles.txt file and Save the new manifest. | Add two new roles to this application: **Admin** and **Clerk**. This will allow you later to assign users and groups to the roles and use the roles in your application code to grant/deny access. |
| 4.3 | Still in the portal select the Enterprise Applications blade for your Azure AD. Find and open settings for your WebUI application. | Configure access to the WebUI application. |
| 4.4 | Select *Users and groups* tab for the application. Click on Add user and configure a user in the Admin role (note the new role will appear because of what you did in 4.2 above). | Assign a user to an application role. |
| 4.5 | Add the following code to the OpenIDConnectAuthentication-Options in the Startup.Auth.cs code:  TokenValidationParameters = new System.IdentityModel.Tokens.TokenValidationParameters()  {  RoleClaimType = "roles"  }, | Tell OWIN to extract user role claims from a JWT property called ‘roles’. In SAML tokens, the role claim typically has a different name. |
| 4.6 | Modify the Authorize attribute to include user role:  [Authorize(Roles ="admin")] | Allow only users in Admin role to access the Home controller. |
| 4.7 | Run the application and sign in once as the above user and once as a different user. (Make sure you close the browser between the sign-ins to logoff properly). | You should get an endless loop when logging in as a non-admin user – application will refuse and non-admin user authentication but AAD will just issue a new token for same signed-in user. |
| 4.8 | Add a new class AuthorizeAttribute to your WebUI project (Snippets/AuthorizeAttribute.cs). | Default AuthorizeAttribute class returns Http 401 on both authentication and authorization failure, which results in an endless set of redirections between the application and AAD when user is authenticated but not authorized to access something in your application. This implementation fixes that problem. |
| 4.9 | Replace the existing AuthorizeAttribute reference in the WebUI project with [WebUI.Utils.Authorize(Roles = “admin”)]. | Allow only ‘admin’ to use this controller. |
| 4.10 | Press F5 to run the application. Sign in as user assigned to manager role and then again with a user not assigned to that role. Click on About in both cases. You should get an unauthorized exception in the latter case. | Test Role Based Authorization Control (RBAC). |