Internet Programming
(CSE-686), Spring 2019

Vaibhav Kumar
Syracuse University
vkumar05@syr.edu

Authentication, Roles and Authorization in ASP.Net Core

Introduction

- A web application can have many end-users with different roles.
- An end user is granted with access privileges based upon its role.
- All these can be achieved by authenticating the user based upon login credentials.
- In the following slides, we are going to explore how a user can be authenticated and how the authorization works depending upon the role in an ASP.Net Core Web App.

AUTHENTICATION

IT IS A PROCESS OF VALIDATING A USER'S CREDENTIALS AGAINST THE STORED VALUES IN THE DATABASE OR OTHER SOURCES.

ROLES

THERE CAN BE DIFFERENT END USERS OF A WEB APPLICATION WITH DIFFERENT ROLES. EXAMPLE: ADMINISTRATOR, REGISTERED USER OR GUEST ETC.

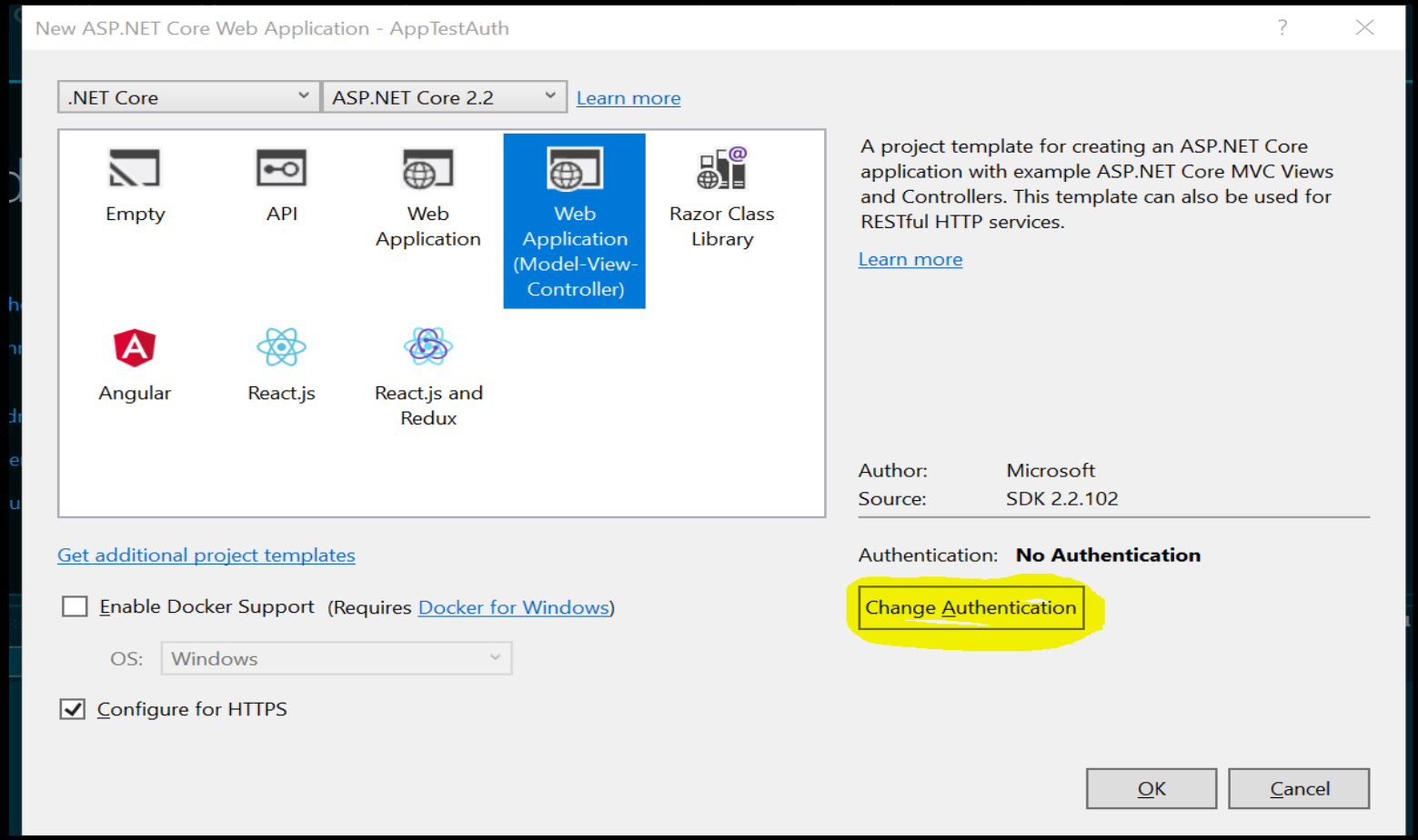
AUTHORIZATION:

IT IS A PROCESS TO IDENTIFY THE PRIVILEGES OF THE USER BASED UPON ITS ROLE.

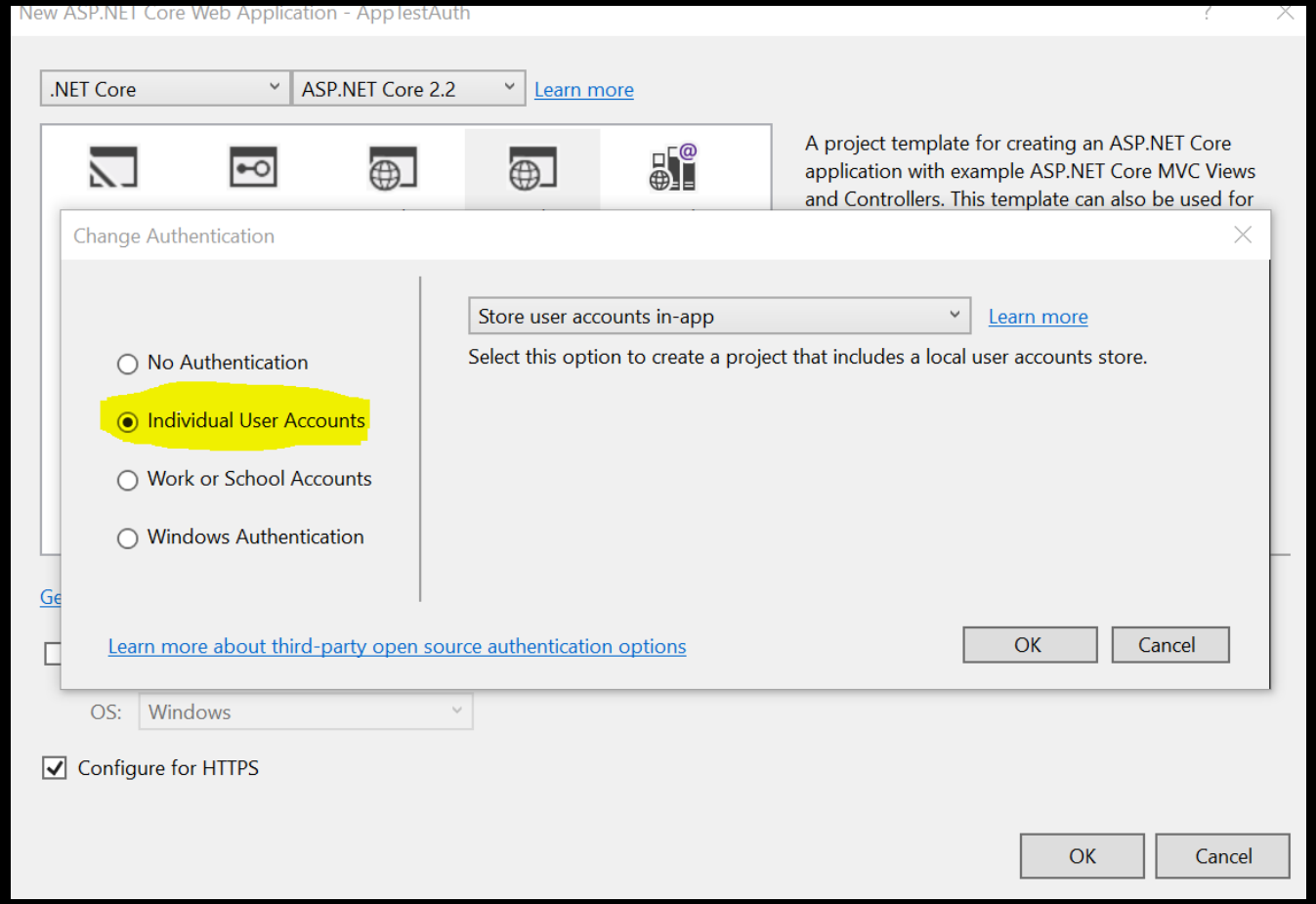
Authentication

- Authentication in ASP.Net Core applications is provided by ASP.NET Core Identity. (There are some other third party services that does the same)
- ASP.NET Core Identity is a membership or identity management system that comes with ASP.NET Core web development stack.
- Some of the facilities provided by ASP.NET Identity are user registration, login etc.

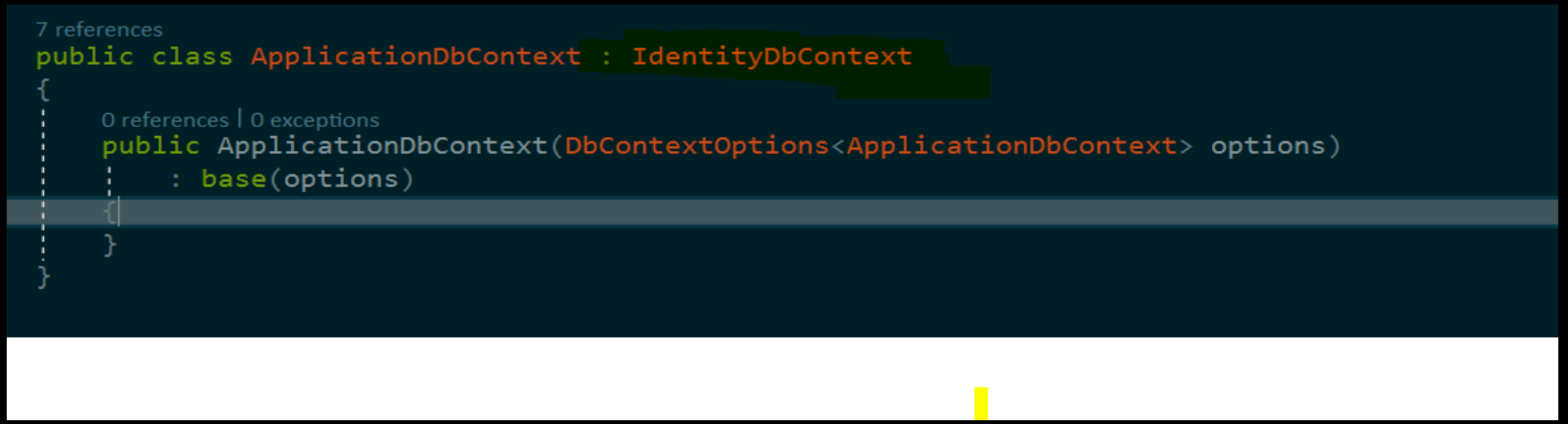
Adding Authentication



Adding Authentication (Contd.)



Adding Authentication (Contd.)



ApplicationDbContext inherits from IdentityDbContext in order to provide Authentication facilities.

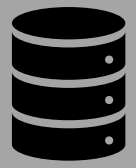
Adding Authentication (Contd.)

```
services.AddIdentity<IdentityUser, IdentityRole>()
  .AddEntityFrameworkStores<ApplicationDbContext>()
  .AddDefaultTokenProviders();
```

Identity services should be registered in the startup.cs along with services.UseAuthentication(). services.UseAuthentication() is necessary as it activates the Identity Services.

Roles

Roles should ideally be created during the development time.



Everytime a new user is registered, a role should be added along with its other profile data.

These roles helps in identifying the privileges while implementing authorization.

Authorization

services.AddAuthorization() in startup.cs is used to implement Authorizartion services.

Authorization or access privileges are checked in two ways

- Role based
- Policy based

Role Based Authorization

“Authorize” attribute is used to define authority of a user with a given role to access controller method.

Example:

```
[Authorize(Roles = "User")]
```

The above usage of attribute gives access to the following controller method to only the user with role as “User”

Policy Based Authorization

- Policy based authorization requires the user to adhere the defined policy in order to get access to a controller method.

```
services.AddAuthorization(options =>
{
    options.AddPolicy("OnlyAdminAccess", policy => policy.RequireRole("Admin"));
});
```

```
[Authorize(Policy = "OnlyAdminAccess")]
0 references | 0 requests | 0 exceptions
public IActionResult PolicyExample()
{
    ViewData["role"] = "Admin";
    return View("Test");
}
```

AllowAnonymous Attribute

- AllowAnonymous attribute is used in case where the access is not restricated to any particular role or policy.

Demo

References

- https://docs.microsoft.com/en-us/aspnet/core/security/?view=aspnetcore-2.2
- https://www.c-sharpcorner.com/register?check=r&ReturnURL=https://www.c-sharpcorner.com/article/role-base-authorization-in-asp-net-core-2-1/

Thank You