**Section04 Authentication and Authorization**

**Notes:-**

**1-authentication: to check if the token is valid or not**

**2-authorization: to check if the user allowed by check the claims provided on each request that user send to.**

**3-we can use either role or claims in apply authorization**

**Role: is an legacy mode authorization while claims is modern mode authorization**

**4-each CliamType is authorication scheam like ClaimTypes.DateOfBirth**

**5-in order to make custom class that handle as middleware in authorization we have to make two class one inherit from authoricationRequirement and use it as argument inside second class that is authorizationHandler**

**Steps:-**

**1-on the startup.cs we set the add authorization as below**

**public void ConfigureServices(IServiceCollection services){**

**//we have to inject the authenticaiton in order to access the authorization middleware**

**//we see that we specify the login path foreach authorize action in any controller it will redirect to the Authenticate**

**services.AddAuthentication("CookieAuth")**

**.AddCookie("CookieAuth", config =>{**

**config.Cookie.Name = "Grandmas.Cookie";**

**config.LoginPath = "/Home/Authenticate";**

**config.AccessDeniedPath = "/Home/AccessDenied";});**

**//add authorization and set the default requirement to default policy that used on each [Authorize] only**

**services.AddAuthorization(config =>{**

**var defaultAuthBuilder = new AuthorizationPolicyBuilder();**

**//it will create two requirements for the policy**

**var defaultAuthPolicy = defaultAuthBuilder**

**.RequireAuthenticatedUser()**

**.RequireClaim(ClaimTypes.DateOfBirth)**

**.Build();**

**config.DefaultPolicy = defaultAuthPolicy;});**

**//with using Authorization with custom policy**

**//normal way (manual inject claim types into the custom policy)**

**services.AddAuthorization(config =>{**

**config.AddPolicy("Claim.Dob", policyBuilder =>{**

**policyBuilder.RequireClaim(ClaimTypes.DateOfBirth);});});**

**//with custom handler to do this automatically**

**services.AddAuthorization(config =>{**

**config.AddPolicy("Claim.Dob", policyBuilder =>{**

**policyBuilder.AddRequirements(new CustomRequireClaims(ClaimTypes.DateOfBirth));});});**

**//we can make it like extension method on AuthorizationPolicyBuildeExtensions**

**services.AddAuthorization(config =>{**

**config.AddPolicy("Claim.Dob", policyBuilder =>{**

**policyBuilder.RequireCustomClaim(ClaimTypes.DateOfBirth);});**

**//we can add policy on role also as below**

**config.AddPolicy("Admin", policyBuilder =>{**

**policyBuilder.RequireClaim(ClaimTypes.Role, "Admin");});});**

**//inject the authorization handler**

**services.AddScoped<IAuthorizationHandler, CustomeRequireClaimHandler>();**

**//to apply controllers with views**

**services.AddControllersWithViews();}**

**2-we create custom authorization class that used as authorization middleware handler that check on each request incoming**

**using Microsoft.AspNetCore.Authorization;**

**using System.Linq;**

**using System.Threading.Tasks;**

**namespace Basics.Authorization{**

**//its requiremetns for the request to be authorized**

**public class CustomRequireClaims : IAuthorizationRequirement{**

**public string ClaimType { get; }**

**public CustomRequireClaims(string claimType){ClaimType = claimType;}}**

**//this handle take the requirements and used to authorized**

**public class CustomeRequireClaimHandler : AuthorizationHandler<CustomRequireClaims>{**

**protected override Task HandleRequirementAsync(AuthorizationHandlerContext context,**

**CustomRequireClaims requirement){**

**var hasClaim = context.User.Claims.Any(x => x.Type == requirement.ClaimType);**

**if (hasClaim){context.Succeed(requirement);}**

**return Task.CompletedTask;}}**

**public static class AuthorizationPolicyBuildeExtensions{**

**public static AuthorizationPolicyBuilder RequireCustomClaim(this AuthorizationPolicyBuilder builder,string claimType){**

**builder.AddRequirements(new CustomRequireClaims(claimType));**

**return builder;}}}**

**3-on the HomeController.cs we set the following code as below**

**[Authorize(Policy ="Claim.DoB")]**

**public IActionResult Secret()**

**{return View();}**

**[Authorize(Roles ="Admin")]**

**public IActionResult SecretRole()**

**{return View("Secret");}**