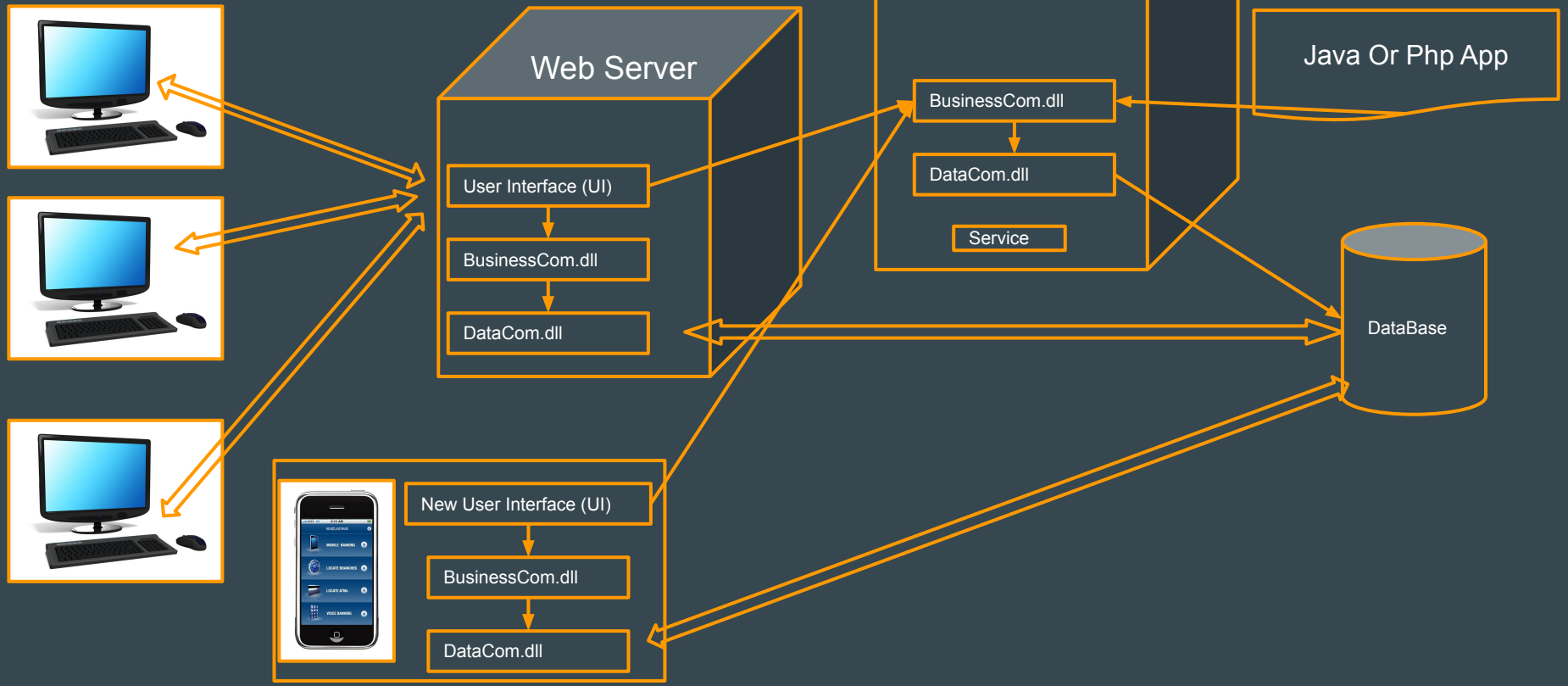
www.ManzoorTheTrainer.com

Learn
ASP.Net Core 2.2 Web APIs
From Scratch
...

Mohd Manzoor Ahmed
(MCT | Founder Of www.ManzoorTheTrainer.com)

Why SOA?


Service Host
Web Server
Java Or Php App
BusinessCom.dll
User Interface (UI)
DataCom.dll
BusinessCom.dll
Service
DataCom.dll
DataBase
New User Interface (UI)
BusinessCom.dll
DataCom.dll

What is SOA?

- It is a reusable component on the network.
- It is the collection of services on a network that communicate with one another.
- For example, verifying a credit card transaction or processing a purchase order.
- Loosely coupled (meaning that an application doesn't have to know the technical details of another application in order to talk to it).
- Services are well-defined, platform-independent interfaces, and are reusable.

How To Achieve SOA?

Few famous ways of achieving SOA

- .NET Remoting
- Web services
- WCF
- **RESTful Services Web APIs**

OLD

NEXT

That's why REST?

- Service for any device with front-end
- Easy
- Simple
- Light weight
- All features of HTTP
- ReST-ful (Representational State Transfer) Services fulfill all the above needs.

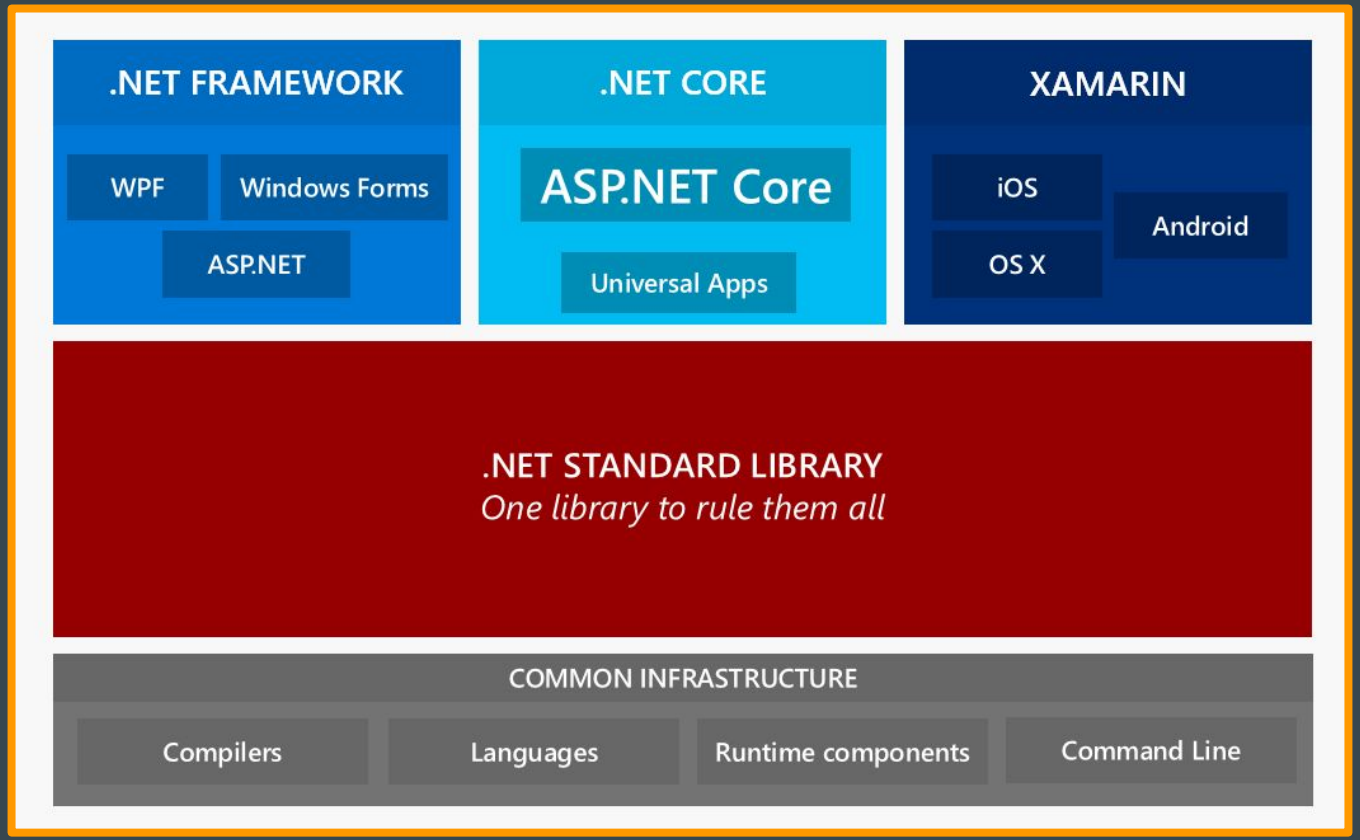
What is REST?

"Architectural Styles and the Design of Network-based Software Architectures" was initially proposed by Roy Thomas Fielding in his 2000 Ph.D. dissertation.

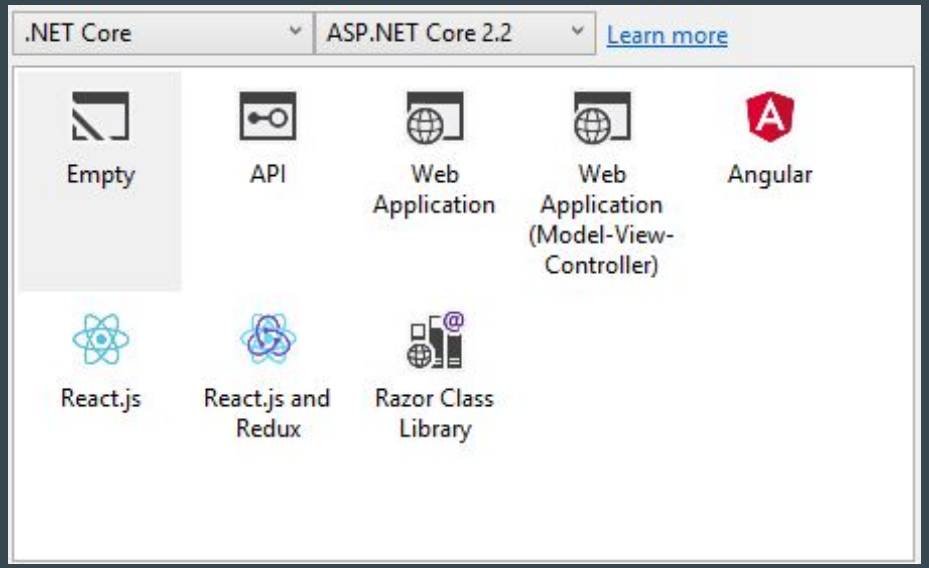
What Is RESTful Web APIs?

- Web service APIs that adhere to the REST architectural constraints.
- HTTP-based RESTful APIs Clients should know:
  - Base URI, such as http://manzoorthetrainer.com/courses/
  - An Internet media type for the data. This is often JSON but can be any other valid Internet media type (e.g., XML, images, etc.)
  - Standard HTTP methods (e.g., GET, PUT, POST, or DELETE)
- Few famous Web APIs are Google Maps, Twitter, YouTube, Flickr, Facebook, Amazon Product, Advertising, etc.,

.NET Framework Vs .NET Core



ASP.NET Core Web APIs



Getting Started With ASP.Net Core 2.2 Web APIs

- Creating Simple Web API
- Hosting Web API on Live On www.myasp.net (60 days free hosting server)
- Consuming Web API using jQuery based ajax call in Web App

4 Steps : Creating A Simple ASP.Net Core 2.2 Web API

www.ManzoorTheTrainer.com

1. Startup Class
   - Add services.AddMvc(); in ConfigureServices method
   - Add app.UseMvc(); in Configure method
2. Add FirstController.cs in Controllers folder
3. Add Get() method in the controller class
4. Add these two attributes on the controller
   - [Route("api/[controller]")]
   - [ApiController]

```
public class Startup
{
    // This method gets called by the runtime. Use this method to add servic
    // For more information on how to configure your application, visit http
    public void ConfigureServices(IServiceCollection services)
    {
        services.AddMvc();
    }

    // This method gets called by the runtime. Use this method to configure
    public void Configure(IApplicationBuilder app, IHostingEnvironment env)
    {
        app.UseMvc();
    }
}
```

```
[Route("api/[controller]")]
[ApiController]
public class FirstController:ControllerBase
{
    public string Get()
    {
        return "Welcome To ManzoorTheTrainer!";
    }
}
```

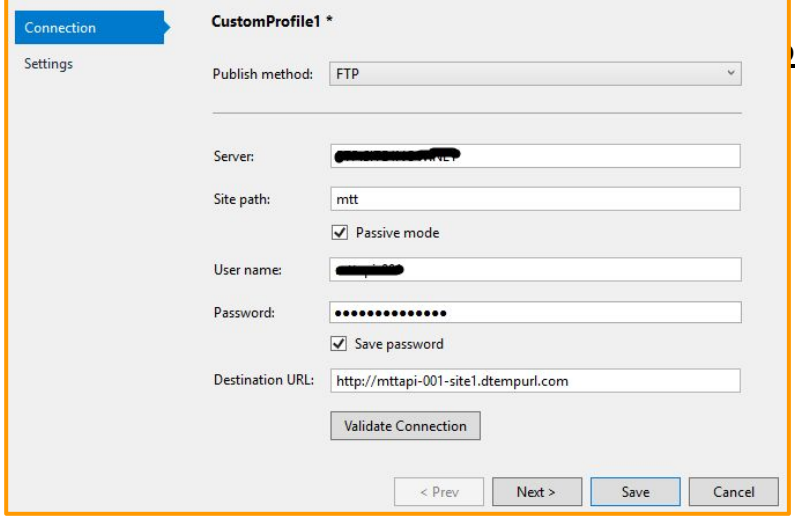
5 Steps : Initial FTP Setup

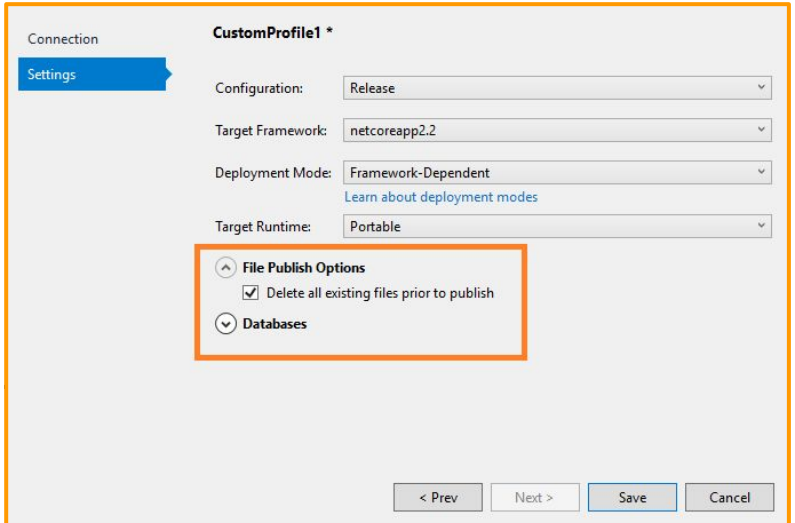
www.ManzoorTheTrainer.com

1. Signup on www.myasp.net for 60 days free trail & no cc info required and login through customer login link.
2. Activate your account using your mobile number activation code
3. Goto the control panel and click on try now.
4. Enter new hosting account info to get ftp details and submit.
5. Get FTP and Destination url details.

5 Steps: Publishing A Simple ASP.Net Core WebAPI

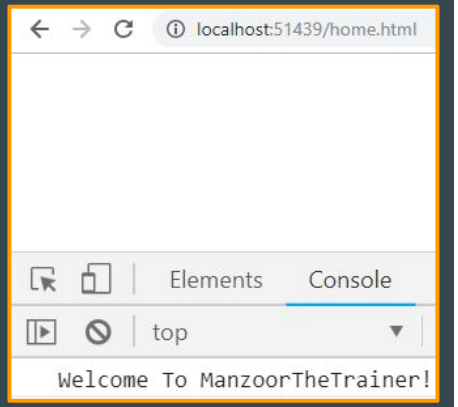
1. Right click the project and click on publish
2. Select IIS, FTP, etc., option
3. Select Publish method as FTP and fill the FTP details with temp url.
4. In setting check delete all existing files options and save.
5. Finally Publish to bring your api live





3 Steps: Consuming A Web API Using jQuery In web APP

1. Enable CORS in Web API and re-publish it.
2. Create a new empty web project and add home.html, jquery and jquery.unobtrusive-ajax files
3. Finally make a call to api.




```
public void ConfigureServices(IServiceCollection services)
{
    services.AddMvc();
    services.AddCors();

}

// This method gets called by the runtime. Use this method to configure
public void Configure(IApplicationBuilder app, IHostingEnvironment env)
{

    app.UseCors(builder =>
        builder.WithOrigins("http://localhost:51439"));

    app.UseMvc();
}
```

```
<head>
    <meta charset="utf-8" />
    <title></title>
    <script src="lib/js/jquery.min.js"></script>
    <script src="lib/js/jquery.unobtrusive-ajax.min.js"></script>
    <script>
        $(document).ready(function () {
                $.ajax({
                    url: 'http://mttapi-001-site1.dtempurl.com/api/first',
                    type: 'GET',
                    success: function (data) {
                        console.log(data);
                    }
                });
        });
    </script>
</head>
```

HTTP Methods

- POST → Create
- GET → Read
- PUT → Update
- DELETE → Delete
  - Eg: (GET or DELETE) URI
    http://manzoorthetrainer.com/courses/1

CRUD Operations Using EF Core 2.2

- Implementation of CRUD operation using EF.
  - CRUD Library Project from EF Core (dll)
  - CRUD ASP.Net Core Web API
- Invoking from fiddler
  - Web Debugger Telerik (http://www.telerik.com/download/fiddler)

Create Class Library EF Core

- Start a new .Net Core Class Library
- Install - Nuget Packages
- Microsoft.EntityFrameworkCore.SqlServer
- Create an Entity Department
- Create a OrganizationContext Class
- OrganizationContext Class Should have DbSet<Department> properties for each entity or object.

```
class Department
{
    [Key]
    public int Did { get; set; }
    public string Name { get; set; }
    public string Description { get; set; }
}
```

```
class OrganizationContext:DbContext
{
    public DbSet<Department> Departments { get; set; }
}
```

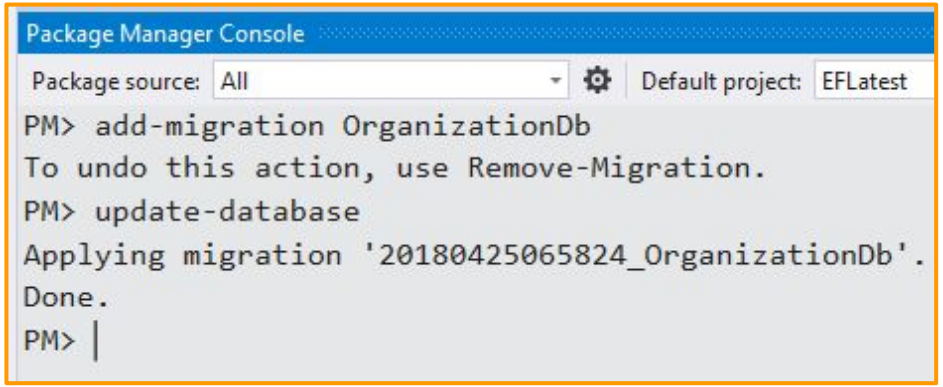
www.ManzoorTheTrainer.com

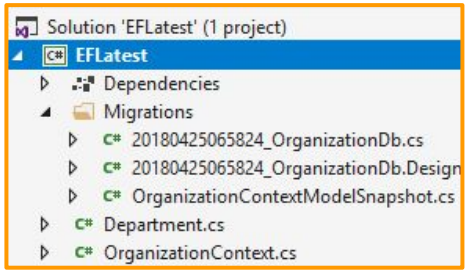
Creating A Database - 5 Steps

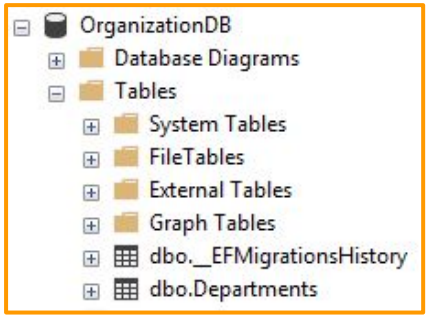
1. Install - Nuget Packages Microsoft.EntityFrameworkCore.Tools
2. Override OnConfiguring() method in OrganizationContext with connection string.
3. Open Package Manager Console and run the below commands
4. add-migration OrganizationDb
5. update-database

```
class OrganizationContext:DbContext
{
    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    {
        optionsBuilder.UseSqlServer(@"Server=XXXX;Database=OrganizationDB;Trusted_Connection=True;");
    }

    public DbSet<Department> Departments { get; set; }
}
```







Creating An API - Read

1. Add reference of EFCoreCRUD dll to api project.
2. In Startup Class
   a. Add services.AddMvc(); services.AddDbContext<OrganizationContext>(); in ConfigureServices method.
   b. Add app.UseMvc(); in Configure method.
3. Add DepartmentController.cs in Controllers folder
4. Add Get() method in the controller class
5. Add [Route("api/[controller]")]
6. Add [ApiController] on Controller

```
[Route("api/[controller]")]
[ApiController]
public class DepartmentController : Controller
{
    OrganizationContext organizationContext;

    public DepartmentController(OrganizationContext _organizationContext)
    {
        organizationContext = _organizationContext;
    }
    // GET: api/<controller>
    [HttpGet]
    public IEnumerable<Department> Get()
    {
        var depts = organizationContext.Departments.ToList();
        return depts;
    }
    // GET api/<controller>/5
    [HttpGet("{id}")]
    public Department Get(int id)
    {
        Department D= organizationContext.Departments
                .Where(x => x.Did == id).FirstOrDefault();
        return D;
    }
```

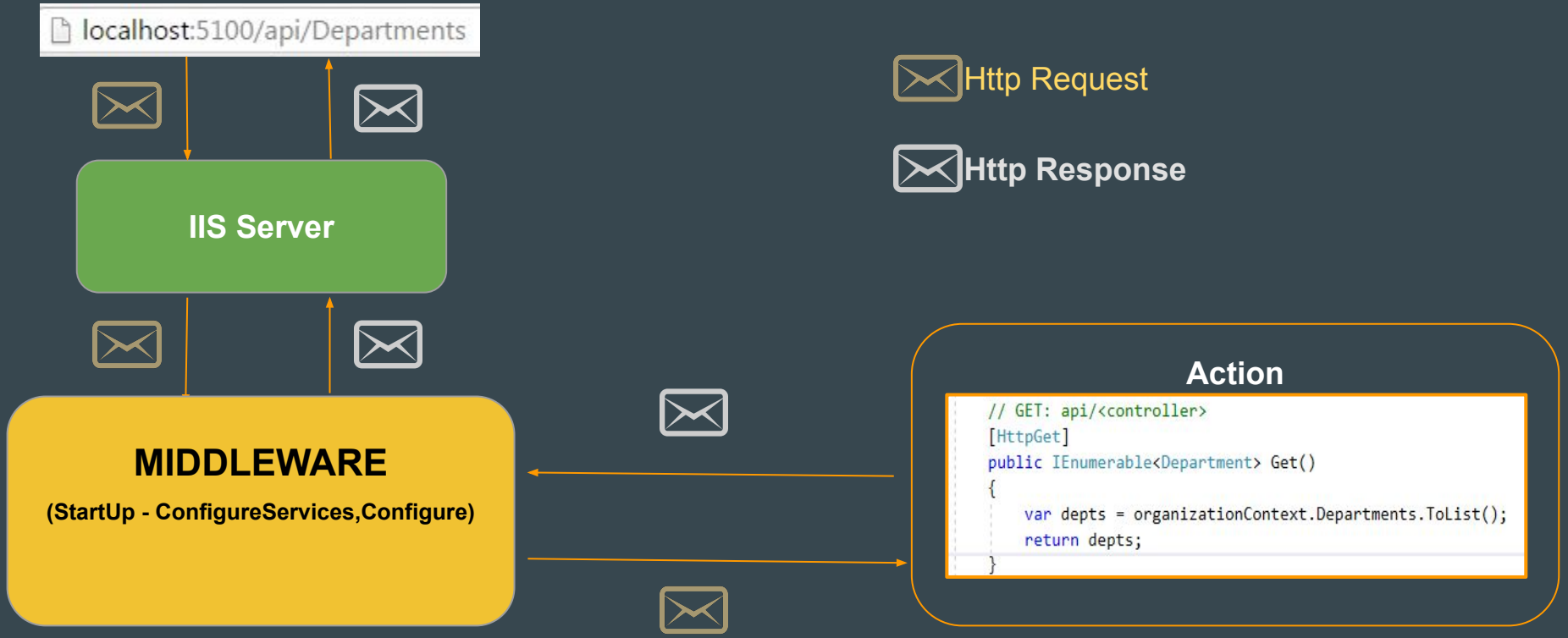
Insert, Update & Delete

1. Add Post(), Put() & Delete method in the controller class.
2. Let's Use Fiddler to test our Web API

Note: While Performing Insert and Update, don't forget to set content-type in request header.
Content-Type: application/json

```
[HttpPost]
public Department Post(Department D)
{
        organizationContext.Add<Department>(D);
        organizationContext.SaveChanges();
        return D;
}
//PUT api/<controller>/5
[HttpPut]
public string Put( Department D)
{
        organizationContext.Update<Department>(D);
        organizationContext.SaveChanges();
        return "Record Updated Successfully";
}
// DELETE api/<controller>/5
[HttpDelete("{id}")]
public string Delete(int id)
{
    Department D = organizationContext.Departments
        .Where(x => x.Did == id).FirstOrDefault();
        organizationContext.Remove<Department>(D);
        organizationContext.SaveChanges();
        return "Record Deleted Successfully";
}
```

ASP.Net Core Web API Life Cycle



Http Request

- **Content**(body)
- **Headers**
- **Method** (Get,Post,Put or Delete)
- Properties
- **RequestUri**
- Version

HTTP Response (Data + Response Code)

- Response Code
  - 200 - HttpStatusCode.OK
  - 201 - HttpStatusCode.Created
  - 204 - HttpStatusCode.NoContent
  - 400 - HttpStatusCode.BadRequest
  - 401 - HttpStatusCode.Unauthorized
  - 403 - HttpStatusCode.Forbidden
  - 404 - HttpStatusCode.NotFound
  - 500 - HttpStatusCode.InternalServerError
  - For more http://www.w3schools.com/tags/ref_httpmessages.asp

Creating Response - Get

- Say I am looking for a department as D.
  - OK();
  - NotFound();
- Return type can be
  - ActionResult<Department>
  - IActionResult

```
[HttpGet()]
public ActionResult<Department> Get(int id)
{
    Department D= organizationContext.Departments
                .Where(x => x.Did == id).FirstOrDefault();
    if (D != null)
    {
        return Ok(D);// Data + 200 (Response Code)
    }
    else
    {
        return NotFound(); //404 (Response Code)
    }
}
```

```
[HttpGet()]
public IActionResult Get(int id)
{
    Department D= organizationContext.Departments
                .Where(x => x.Did == id).FirstOrDefault();
    if (D != null)
    {
        return Ok(D);// Data + 200 (Response Code)
    }
    else
    {
        return NotFound(); //404 (Response Code)
    }
}
```

Creating Response - Post

- Say I am inserting a Department
  - CreatedAtAction()
  - BadRequest()

Note: CreatedAtAction will return object created along with the uri to access that object. (Location)

```
[HttpPost]
public IActionResult Post(Department D)
{
    if (ModelState.IsValid)
    {
        organizationContext.Add<Department>(D);
        organizationContext.SaveChanges();
        return CreatedAtAction("Get",new { id = D.Did }, D);
    }
    else
    {
        return BadRequest(ModelState);
    }
}
```

Transport

Location: http://localhost:55629/api/Department/18

Transfer-Encoding: chunked

Creating Response - Put

- Say I am updating a department
  - BadRequest()
  - NotFound()
  - NoContent()
  - OK()

```
[HttpPut]
0 references | 0 requests | 0 exceptions
public IActionResult Put(Department D)
{
    if (ModelState.IsValid)
    {
        var Dept = dbContext.Departments
                    .Where(x => x.Did == D.Did)
                    .AsNoTracking().FirstOrDefault();
        if (Dept != null)
        {
            dbContext.Update(D);
            dbContext.SaveChanges();
            return NoContent();//or Ok(D);
        }
        else
        {
            return NotFound();
        }
    }
    else
    {
        return BadRequest(ModelState);
    }
}
```

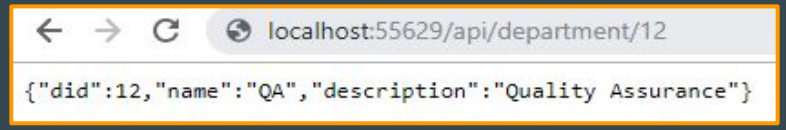
Creating Response - Delete

- Say I am deleting a Department
  - NotFound()
  - NoContent()
  - OK()

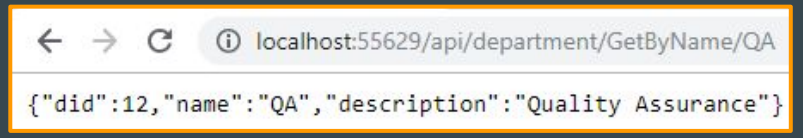
```
[HttpDelete("{id}")]
public IActionResult Delete(int id)
{
    Department D = organizationContext.Departments
                .Where(x => x.Did == id).FirstOrDefault();
    if (D != null)
    {
        organizationContext.Remove<Department>(D);
        organizationContext.SaveChanges();
        return NoContent();//  or return Ok(D);
    }
    else
    {
        return NotFound();
    }
}
```

Parameters Mechanism And Get Action OverLoading

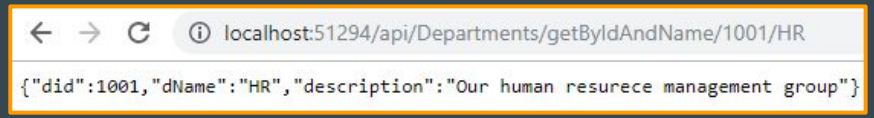
```
[HttpGet("{id}")]
public IActionResult Get(int id)...
```



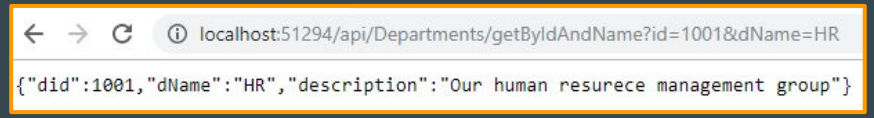

```
[Route("[action]/{Name}")]
[HttpGet]
public IActionResult GetByName(string Name)...
```




```
[HttpGet("getByIdAndName/{id}/{dName}")]
0 references | 0 requests | 0 exceptions
public IActionResult GetByIdAndName(int id, string dName)...
```




```
[HttpGet("getByIdAndName")]
0 references | 0 requests | 0 exceptions
public IActionResult GetByIdAndName(int id, string dName)...
```



Swagger Or OpenAPI - 3 Steps

Swagger generates documentation for Web API so that developer can understand & consume Web API.

1. Add Package NSwag.AspNetCore
2. Add services.AddSwaggerDocument(); in as service in startup.cs
3. Use app.UseOpenApi(); app.UseSwaggerUi3(); In configure of startup.cs

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddMvc();
    services.AddDbContext<OrganizationDbContext>();
    services.AddSwaggerDocument();
}

// This method gets called by the runtime. Use this method
0 references | 0 exceptions
public void Configure(IApplicationBuilder app, IHostingEnv
{
    app.UseDeveloperExceptionPage();
    app.UseOpenApi();
    app.UseSwaggerUi3();
    app.UseMvc();
}
```

Asynchronous Actions & Scaffolding

- async,Task and await
- Adding Employee entity and Scaffolding EmployeesController

```
[Route("api/[controller]")]
[ApiController]
1 reference
public class EmployeesController : ControllerBase
{
    private readonly OrganizationDbContext _context;
    0 references | 0 exceptions
    public EmployeesController(OrganizationDbContext context)
    {
        _context = context;
    }

    [HttpGet]
    0 references | 0 requests | 0 exceptions
    public async Task<ActionResult<IEnumerable<Employee>>> GetEmployees()
    {
        var Emps = await _context.Employees.ToListAsync();
        return Ok(Emps);
    }

    [HttpGet("{id}")]
    0 references | 0 requests | 0 exceptions
    public async Task<ActionResult<Employee>> GetEmployee(int id)
    {
        var employee = await _context.Employees.FindAsync(id);
```

Solution To Circular Reference Problem

Approaches - 1:

Write specific Select Lamda expression

```
[HttpGet]
0 references | 0 requests | 0 exceptions
public async Task<ActionResult<IEnumerable<Employee>>> GetEmployees()
{
    var Emps = await _context.Employees.Include(x => x.Department)
                .Select(x => new Employee
                {
                    Eid = x.Eid,
                    Name = x.Name,
                    Gender = x.Gender,
                    Did = x.Did,
                    Department = x.Department
                })
                .ToListAsync();
    return Ok(Emps);
}
```

Use Lamda Expression

```
[HttpGet]
0 references | 0 requests | 0 exceptions
public async Task<IActionResult> Get()
{
    var Depts = await dbContext.Departments.Include(x => x.Employees)
        .Select(x => new Department
        {
            Did = x.Did,
            DName = x.DName,
            Description = x.Description,
            Employees = x.Employees.Select(y => new Employee
            {
                Eid = y.Eid,
                Name = y.Name,
                Gender = y.Gender
            })
        })
        .ToListAsync();
    return Ok(Depts);
}
```

Lamda Expression

Solution To Circular Reference Problem

Approaches - 2:

1. Add Newtonsoft.Json from Manage nuget packages.
2. Serialize the Depts object to JSON string using JsonConvert by ignoring reference looping in serialization setting

```
[HttpGet]
0 references | 0 requests | 0 exceptions
public async Task<IActionResult> Get()
{
    var Depts = await dbContext.Departments
                        .Include(x => x.Employees).ToListAsync();

    if (Depts.Count != 0)
    {
        var jsonResult = JsonConvert.SerializeObject(Depts,
            Formatting.None,
            new JsonSerializerSettings()
            {
                ReferenceLoopHandling = ReferenceLoopHandling.Ignore
            });
        return Ok(jsonResult);
    }
    else
        return NotFound();
}
```

Response - Get As XML

- Default json: String, Specific, ActionResult<Type>, IActionResult
- For XML
  - XMLInstall the package Microsoft.AspNetCore.Mvc.Formatters.Xml
  - Add xml format in service
    services.AddMvc()
    .AddXmlSerializerFormatters()
    .AddXmlDataContractSerializerFormatters();
  - Add Filter on the controller
    [Produces("application/xml")]

```
services.AddMvc()
        .AddXmlSerializerFormatters()
        .AddXmlDataContractSerializerFormatters();
```

```
[Produces("application/xml")]
[Route("api/[controller]")]
[ApiController]
public class DepartmentController : Controller
{
    OrganizationContext organizationContext;
```

Exception Handling
Action Level

```
[HttpGet("{id}")]
public IActionResult Get(int id)
{
    try
    {
        Department D = organizationContext
                        .Departments
                        .Where(x => x.Did == id)
                        .FirstOrDefault();
        if (D != null)...
        else...
    }
    catch (Exception E)
    {
        //Write to exception Logger
        return StatusCode(500, E.Message);
    }
}
```

Exception Handling Globally - Custom Filter

```
public class MyExceptionFilter : IExceptionFilter
{
    public void OnException(ExceptionContext context)
    {
        context.ExceptionHandled = true;
        HttpResponse response = context.HttpContext.Response;
        response.StatusCode = 500;
        response.ContentType = "application/json";
        context.Result = new ObjectResult(context.Exception.Message);
    }
}
```

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddMvc(config =>
    {
        config.Filters.Add(typeof(MyExceptionFilter));
    });
}
```

```
[HttpGet("{id}")]
public IActionResult Get(int id)
{
        Department D = organizationContext
                        .Departments
                        .Where(x => x.Did == id)
                        .FirstOrDefault();
        if (D != null)...
        else...
}
```

Here We Need Not To Write Try..Catch

Exception Handling Globally - In Middleware


```
public void Configure(IApplicationBuilder app, IHostingEnvironment env)
{
    app.UseExceptionHandler(
        options =>
        {
            options.Run(async context =>
            {
                context.Response.StatusCode = 500;//Internal Server Error
                context.Response.ContentType = "application/json";
                var ex = context.Features.Get<IExceptionHandlerFeature>();
                if (ex != null)
                {
                    await context.Response.WriteAsync(ex.Error.Message);
                }
            });
        }
        );
    app.UseMvc();
}
```

```
[HttpGet("{id}")]
public IActionResult Get(int id)
{
        Department D = organizationContext
                        .Departments
                        .Where(x => x.Did == id)
                        .FirstOrDefault();
        if (D != null)...
        else...
}
```

Here We Need Not To Write Try..Catch

Security In Web API

- Authentication & Authorization with ASP.Net Identity as user management
  - Cookie Based (Default)
  - Token Based Authentication (JWT)

ASP.Net Identity - Authentication

- Checking the genuinity of a user or client is called as authentication.
- It is normally achieved with the help of username and password called as forms authentication for web apps.
- In our early days .Net Framework we used Membership classes or providers to achieve authentication.
- The ASP.NET Identity system is a replacement of Membership systems which is specially designed to support social logins for building modern applications for the web, phone, or tablet.
- In Microsoft.AspNetCore.Identity we have mainly two classes using which we implement authentication and authorization i.e., UserManager and SignInManager

Initial Setup - 5 Steps

1. Add the nuget package Microsoft.AspNetCore.Identity.EntityFrameworkCore, Microsoft.Extensions.Identity.Stores
2. Inherit application's DbContext say OrganizationDbContext from IdentityDbContext.
3. Add Migration, Update-Database
4. Inject application's DbContext object and Identity object using service collection object in ConfigureServices Method of StartUp Class.
5. Add authentication configuration in HTTP pipeline using app object in Configure method of StartUp classes

```
10 references
public class OrganizationDbContext:IdentityDbContext
{
    0 references | 0 exceptions
    protected override void OnConfiguring(DbContextOptionsBui
    {
        base.OnConfiguring(optionsBuilder);
        optionsBuilder.UseSqlServer(@"Server=DESKTOP-O84LAFN;
    }
```

```
0 references | 0 exceptions
public void ConfigureServices(IServiceCollection services)
{
    services.AddMvc(x => x.Filters.Add(new AuthorizeFilter()));
    services.AddDbContext<OrganizationDbContext>();
    services.AddIdentity<IdentityUser, IdentityRole>()
        .AddEntityFrameworkStores<OrganizationDbContext>()
        .AddDefaultTokenProviders();
```

```
0 references | 0 exceptions
public void Configure(IApplicationBuilder app,
{
    app.UseAuthentication();
    app.UseMvc();
}
```

Securing Web API

1. Secure Access [Authorize]
   a. Action Level
   b. Controller Level
   c. Application Level
2. Anonymous Access [AllowAnonymous]
3. Send Response Code as 401 on OnRedirectToLogin event

```
[Authorize]
[HttpGet]
0 references | 0 requests | 0 exceptions
public async Task<IActionResult> Get()...
```

```
[Authorize]
[Route("api/[controller]")]
[ApiController]
1 reference
public class DepartmentsController : Controller
{
```

```
0 references | 0 exceptions
public void ConfigureServices(IServiceCollection services)
{
    services.AddMvc(x => x.Filters.Add(new AuthorizeFilter()));
```

```
services.ConfigureApplicationCookie(opt =>
{
    opt.Events = new CookieAuthenticationEvents
    {
        OnRedirectToLogin = redirectContext =>
        {
            redirectContext.HttpContext.Response.StatusCode = 401;
            return Task.CompletedTask;
        }
    };
});
```

User Registration - 4 Steps

1. Create RegisterViewModel
2. Create UserManager And SignInManager objects.
3. Create a Post action to handle registration request.


```
[AllowAnonymous]
[Route("api/[controller]")]
[ApiController]
1 reference
public class AccountController : ControllerBase
{
    UserManager<IdentityUser> userManager;
    SignInManager<IdentityUser> signInManager;

    0 references | 0 exceptions
    public AccountController(SignInManager<IdentityUser> _signInManager,
        UserManager<IdentityUser> _userManager)
    {
        signInManager = _signInManager;
        userManager = _userManager;
    }
```

```
[HttpPost("register")]
0 references | 0 requests | 0 exceptions
public async Task<IActionResult> Register(RegisterViewModel model)
{
    if (ModelState.IsValid)
    {
        var user = new IdentityUser()
        {
            UserName = model.UserName,
            Email = model.Email
        };
        var userResult = await userManager.CreateAsync(user, model.Password);
        if (userResult.Succeeded)
        {
            return Ok(user);
        }
    }
    return BadRequest(ModelState.Values);
}
```

SignIn And SignOut - 7 Steps

1. Create SignInViewModel
2. Create UserManager And SignInManager objects.
3. Create an action to handle SignIn and SignOut request.

Note: After SignIn we get a cookie & we need to pass that cookie in the header for secure access of any actions.

Manzoor™
.....★The Trainer

```
[AllowAnonymous]
[Route("api/[controller]")]
[ApiController]
1 reference
public class AccountController : ControllerBase
{
    UserManager<IdentityUser> userManager;
    SignInManager<IdentityUser> signInManager;

    0 references | 0 exceptions
    public AccountController(SignInManager<IdentityUser> _signInManager,
        UserManager<IdentityUser> _userManager)
    {
        signInManager = _signInManager;
        userManager = _userManager;
    }
```

```
[HttpPost("signIn")]
0 references | 0 requests | 0 exceptions
public async Task<IActionResult> SignIn(SignInViewModel model)
{
    if (ModelState.IsValid)
    {
        var signInResult = await signInManager
            .PasswordSignInAsync(model.UserName, model.Password, false, false);
        if (signInResult.Succeeded)
        {
            return Ok();
        }
    }
    return BadRequest(ModelState);
}
[HttpPost("signOut")]
0 references | 0 requests | 0 exceptions
public async Task<IActionResult> SignOut()
{
    await signInManager.SignOutAsync();
    return NoContent();
}
```

Authorization - 4 Steps

1. Create Roles in Db Say ‘Admin’ and ‘User’.
2. Assign Role to a user from Db.
3. Add role to authorize attribute [Authorize(Roles = "Admin")]
4. At last we will see how to assign default role as ‘User’ at the time of registration.

Note : Send Response Code as 401 on OnRedirectToAccessDenied event & set cookie expiration time.

Manzoor™ .....★The Trainer

```
[HttpPost("register")]
0 references | 0 requests | 0 exceptions
public async Task<IActionResult> Register(RegisterViewModel model)
{
    if (ModelState.IsValid)
    {
        var user = new IdentityUser()...;
        var userResult = await userManager.CreateAsync(user, model.Password);
        if (userResult.Succeeded)
        {
            var roleResult = await userManager.AddToRoleAsync(user, "User");
            if (roleResult.Succeeded)
            {
                return Ok(user);
            }
        }
    }
    return BadRequest(ModelState.Values);
}
```

```
services.ConfigureApplicationCookie(opt =>
{
    opt.ExpireTimeSpan = new TimeSpan(0, 0, 30);
    opt.Events = new CookieAuthenticationEvents
    {
        OnRedirectToLogin = redirectContext =>
        {
            redirectContext.HttpContext.Response.StatusCode = 401;
            return Task.CompletedTask;
        },
        OnRedirectToAccessDenied = redirectContext =>
        {
            redirectContext.HttpContext.Response.StatusCode = 401;
            return Task.CompletedTask;
        }
    };
});
```

Few Security Concepts

- Principal
  - Identity
  - Role
- Claims
- https://github.com/microsoft/referencesource/tree/master/mscorlib/system/security
- We need
  - UserId
  - UserName
  - Role

Why JWT & Why not Cookies for Authentication?

1. It is stateless.
2. JSON parsers are common in most programming languages.
3. Client-side support on multiple platforms, especially mobile.

What is JWT (JSON Web Tokens)? (https://jwt.io)

- JSON Web Token (JWT) is a way of securely transmitting information between client and server as a JSON object.
- This information is digitally signed using a secret key (HMAC algorithm).
- Basically it is used for Authorization and Information Exchange.
- Structure Of JWT
  a. Header (ALGORITHM & TOKEN TYPE)
  b. PayLoad(Data)
  c. Sign (Header+Payload+Key)
- Eg: xxxxxxxxxx.yyyyyyyyyyyyy.zzzzzzzzzzzzzzz

How JWT Works? (Authentication & Authorization)

Web Or Mobile Client

Username & Password

JWT

Request with JWT Header

Response

Web API On Server

Validate Login & Creates JWT (SignIn Action)

Validate JWT (Middleware) & Send Response (RequestedAction)

Create JWT in SignIn Action (Steps - 5)


```
[HttpPost("signIn")]
0 references | 0 requests | 0 exceptions
public async Task<IActionResult> SignIn(SignInViewModel model)
{
    Find user and it's roles after successful login
    //Step - 1: Create IdentityClaims
    IdentityOptions identityOptions = new IdentityOptions();
    var claims = new Claim[]
    {
                new Claim("userId",user.Id),
                new Claim(identityOptions.ClaimsIdentity.UserNameClaimType,user.UserName),
                new Claim(identityOptions.ClaimsIdentity.RoleClaimType, roles[0])
    };

    //Step - 2: Create signingKey from Secretkey
    var signingKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes("this-is-my-secret-key"));

    //Step -3: Create signingCredentials from signingKey with HMAC algorithm
    var signingCredentials = new SigningCredentials(signingKey, SecurityAlgorithms.HmacSha256);

    //Step - 4: Create JWT with signingCredentials, IdentityClaims & expire duration.
    var jwt = new JwtSecurityToken(signingCredentials: signingCredentials,
                                   expires: DateTime.Now.AddMinutes(30), claims: claims);

    //Step - 5: Finally write the token as response with OK().
    return Ok(new JwtSecurityTokenHandler().WriteToken(jwt));
}
```

Validate JWT in Middleware (Steps - 5)


```
//Step-1: Create signingKey from Secretkey
var signingKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes("this-is-my-secret-key"));

//Step-2:Create Validation Parameters using signingKey
var tokenValidationParameters = new TokenValidationParameters()
{
    IssuerSigningKey = signingKey,
    ValidateIssuer = false,
    ValidateAudience = false
};

        //Step-3: Set Authentication Type as JwtBearer
services.AddAuthentication(auth =>
        {
            auth.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
        })
        //Step-4: Set Validation Parameter created above
        .AddJwtBearer(jwt =>
        {
            jwt.TokenValidationParameters = tokenValidationParameters;
        });
```

Note:

```
[Authorize(AuthenticationSchemes =JwtBearerDefaults.AuthenticationScheme,Roles ="Admin")]
[Route("api/[controller]")]
[ApiController]
1 reference
public class DepartmentsController : Controller
{
```

www.ManzoorTheTrainer.com

Thanks