create database Inventory

use Inventory

Create Table Products(

ProductId Int Identity(1,1) Primary Key,

Name Varchar(100) Not Null,

Category Varchar(100),

Color Varchar(20),

UnitPrice Decimal Not Null,

AvailableQuantity Int Not Null)

GO

insert into Products values('Product1','Cat1','Colour1',100,10);

insert into Products values('Product2','Cat1','Colour2',100,11);

insert into Products values('Product3','Cat1','Colour3',100,12);

insert into Products values('Product4','Cat2','Colour1',100,10);

insert into Products values('Product5','Cat2','Colour2',100,11);

insert into Products values('Product6','Cat2','Colour3',100,12);

Select \* from Products

Create Table UserInfo(

UserId Int Identity(1,1) Not null Primary Key,

FirstName Varchar(30) Not null,

LastName Varchar(30) Not null,

UserName Varchar(30) Not null,

Email Varchar(50) Not null,

[Password] Varchar(20) Not null,

CreatedDate DateTime Default(GetDate()) Not Null)

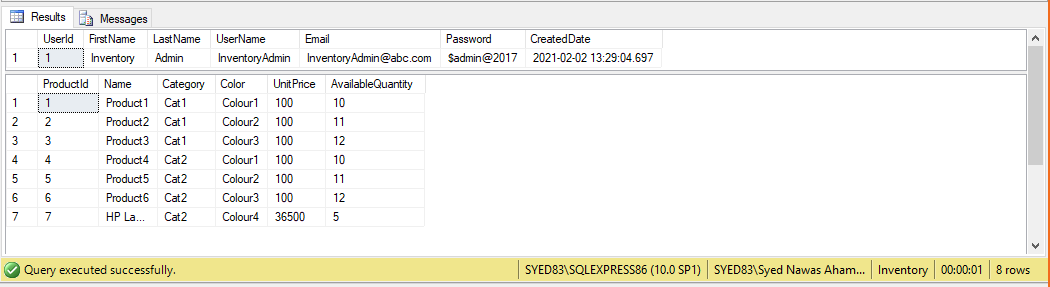
GO

Insert Into UserInfo(FirstName, LastName, UserName, Email, Password)

Values ('Inventory', 'Admin', 'InventoryAdmin', 'InventoryAdmin@abc.com', '$admin@2017')

Select \* from UserInfo

select \* from Products



Inventory Database

This package helps generate controllers and views.

* **Install-Package Microsoft.VisualStudio.Web.CodeGeneration.Design -Version 3.1.4**

This package helps create database context and model classes from the database.

* **Install-Package Microsoft.EntityFrameworkCore.Tools -Version 3.1.8**

Database provider allows Entity Framework Core to work with SQL Server.

* **Install-Package Microsoft.EntityFrameworkCore.SqlServer -Version 3.1.8**

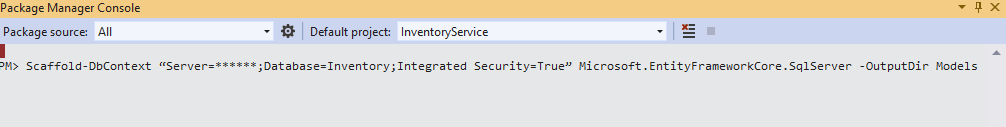
It provides support for creating and validating a JWT token.

* **Install-Package System.IdentityModel.Tokens.Jwt** **-Version 5.6.0**

This is the middleware that enables an ASP.NET Core application to receive a bearer token in the request pipeline.

* **Install-Package Microsoft.AspNetCore.Authentication.JwtBearer -Version 3.1.8**

**Scaffold-DbContext “Server=\*\*\*\*\*\*;Database=Inventory;Integrated Security=True” Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models**

  
using System;

using System.Collections.Generic;

namespace InventoryService.Models

{

public partial class UserInfo

{

public int UserId { get; set; }

public string FirstName { get; set; }

public string LastName { get; set; }

public string UserName { get; set; }

public string Email { get; set; }

public string Password { get; set; }

public DateTime CreatedDate { get; set; }

}

}

using System;

using System.Collections.Generic;

namespace InventoryService.Models

{

public partial class Products

{

public int ProductId { get; set; }

public string Name { get; set; }

public string Category { get; set; }

public string Color { get; set; }

public decimal UnitPrice { get; set; }

public int AvailableQuantity { get; set; }

}

}

using System;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Metadata;

namespace InventoryService.Models

{

public partial class InventoryContext : DbContext

{

public InventoryContext(DbContextOptions<InventoryContext> options)

: base(options)

{

}

public virtual DbSet<Products> Products { get; set; }

public virtual DbSet<UserInfo> UserInfo { get; set; }

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<Products>(entity =>

{

entity.HasKey(e => e.ProductId)

.HasName("PK\_\_Products\_\_B40CC6CD7F60ED59");

entity.Property(e => e.Category)

.HasMaxLength(100)

.IsUnicode(false);

entity.Property(e => e.Color)

.HasMaxLength(20)

.IsUnicode(false);

entity.Property(e => e.Name)

.IsRequired()

.HasMaxLength(100)

.IsUnicode(false);

entity.Property(e => e.UnitPrice).HasColumnType("decimal(18, 0)");

});

modelBuilder.Entity<UserInfo>(entity =>

{

entity.HasKey(e => e.UserId)

.HasName("PK\_\_UserInfo\_\_1788CC4C03317E3D");

entity.Property(e => e.CreatedDate)

.HasColumnType("datetime")

.HasDefaultValueSql("(getdate())");

entity.Property(e => e.Email)

.IsRequired()

.HasMaxLength(50)

.IsUnicode(false);

entity.Property(e => e.FirstName)

.IsRequired()

.HasMaxLength(30)

.IsUnicode(false);

entity.Property(e => e.LastName)

.IsRequired()

.HasMaxLength(30)

.IsUnicode(false);

entity.Property(e => e.Password)

.IsRequired()

.HasMaxLength(20)

.IsUnicode(false);

entity.Property(e => e.UserName)

.IsRequired()

.HasMaxLength(30)

.IsUnicode(false);

});

OnModelCreatingPartial(modelBuilder);

}

partial void OnModelCreatingPartial(ModelBuilder modelBuilder);

}

}

appsettings.json

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft": "Warning",

"Microsoft.Hosting.Lifetime": "Information"

}

},

"AllowedHosts": "\*",

"ConnectionStrings": {

"InventoryDBConStr": "Server=SYED83\\SQLEXPRESS86;Database=Inventory;Integrated Security=True"

},

"Jwt": {

"Key": "SomeUserSecretKey",

"Issuer": "InventoryServiceServer",

"Audience": "AuthenticatedClients",

"Subject": "InventoryServiceAccessToken"

}

}

Startup.cs

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.AspNetCore.HttpsPolicy;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using Microsoft.Extensions.Logging;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

//Added...

using InventoryService.Models;

using Microsoft.EntityFrameworkCore;

//Added

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using System.Text;

namespace InventoryService

{

public class Startup

{

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public IConfiguration Configuration { get; }

// This method gets called by the runtime. Use this method to add services to the container.

public void ConfigureServices(IServiceCollection services)

{

//Constr Info

var connection = Configuration.GetConnectionString("InventoryDBConStr");

services.AddDbContext<InventoryContext>(options => options.UseSqlServer(connection));

services.AddControllers();

services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)

.AddJwtBearer(Options =>

{

Options.RequireHttpsMetadata = false;

Options.SaveToken = true;

Options.TokenValidationParameters = new TokenValidationParameters()

{

ValidateIssuer = true,

ValidateAudience = true,

ValidAudience = Configuration["Jwt:Audience"],

ValidIssuer = Configuration["Jwt:Issuer"],

IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(Configuration["Jwt:Key"]))

};

});

}

// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

app.UseHttpsRedirection();

app.UseRouting();

app.UseAuthentication();

app.UseAuthorization();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllers();

});

}

}

}

using Microsoft.EntityFrameworkCore;

using InventoryService.Models;

using System.Configuration;

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using System.Text;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

string conStr = builder.Configuration.GetConnectionString("MyConStr");

builder.Services.AddDbContext<InventoryContext>(options => options.UseSqlServer(conStr));

builder.Services.AddControllers();

builder.Services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)

.AddJwtBearer(Options =>

{

Options.RequireHttpsMetadata = false;

Options.SaveToken = true;

Options.TokenValidationParameters = new TokenValidationParameters()

{

ValidateIssuer = true,

ValidateAudience = true,

ValidAudience = builder.Configuration["Jwt:Audience"],

ValidIssuer = builder.Configuration["Jwt:Issuer"],

IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(builder.Configuration["Jwt:Key"]))

};

});

// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

var app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseAuthorization();

app.MapControllers();

app.Run();

TokenController.cs (WebAPI Controller)

//Unused defaults

using Microsoft.AspNetCore.Http;

using System.Collections.Generic;

using System.Linq;

//Defaults

using System;

using Microsoft.AspNetCore.Mvc;

using System.Threading.Tasks;

//Added

using System.Text;

using InventoryService.Models;

using Microsoft.EntityFrameworkCore;

using Microsoft.Extensions.Configuration;

using Microsoft.IdentityModel.Tokens;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

namespace InventoryService.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class TokenController : ControllerBase

{

public IConfiguration \_configuration;

private readonly InventoryContext \_context;

public TokenController(IConfiguration config, InventoryContext context)

{

\_configuration = config;

\_context = context;

}

[HttpPost]

public async Task<IActionResult> Post(UserInfo \_userData)

{

if (\_userData != null && \_userData.Email != null && \_userData.Password != null)

{

var user = await GetUser(\_userData.Email, \_userData.Password);

if (user != null)

{

//create claims details based on the user information

var claims = new[] {

new Claim(JwtRegisteredClaimNames.Sub, \_configuration["Jwt:Subject"]),

new Claim(JwtRegisteredClaimNames.Jti, Guid.NewGuid().ToString()),

new Claim(JwtRegisteredClaimNames.Iat, DateTime.UtcNow.ToString()),

new Claim("Id", user.UserId.ToString()),

new Claim("FirstName", user.FirstName),

new Claim("LastName", user.LastName),

new Claim("UserName", user.UserName),

new Claim("Email", user.Email)

};

var key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(\_configuration["Jwt:Key"]));

var signIn = new SigningCredentials(key, SecurityAlgorithms.HmacSha256);

var token = new JwtSecurityToken(\_configuration["Jwt:Issuer"], \_configuration["Jwt:Audience"], claims, expires: DateTime.UtcNow.AddDays(1), signingCredentials: signIn);

return Ok(new JwtSecurityTokenHandler().WriteToken(token));

}

else

{

return BadRequest("Invalid credentials");

}

}

else

{

return BadRequest();

}

}

private async Task<UserInfo> GetUser(string email, string password)

{

return await \_context.UserInfo.FirstOrDefaultAsync(u => u.Email == email && u.Password == password);

}

}

}

launchsettings.json

{

"$schema": "http://json.schemastore.org/launchsettings.json",

"iisSettings": {

"windowsAuthentication": false,

"anonymousAuthentication": true,

"iisExpress": {

"applicationUrl": "http://localhost:2230",

"sslPort": 44378

}

},

"profiles": {

"IIS Express": {

"commandName": "IISExpress",

"launchBrowser": true,

"launchUrl": "api/products",

"environmentVariables": {

"ASPNETCORE\_ENVIRONMENT": "Development"

}

},

"InventoryService": {

"commandName": "Project",

"launchBrowser": true,

"launchUrl": "api/products",

"applicationUrl": "https://localhost:5001;http://localhost:5000",

"environmentVariables": {

"ASPNETCORE\_ENVIRONMENT": "Development"

}

}

}

}

ProductsController.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

//Added

using Microsoft.EntityFrameworkCore;

using InventoryService.Models;

using Microsoft.AspNetCore.Authorization;

namespace InventoryService.Controllers

{

[Authorize]

[Route("api/[controller]")]

[ApiController]

public class ProductsController : ControllerBase

{

private readonly InventoryContext \_context;

public ProductsController(InventoryContext context)

{

\_context = context;

}

// GET: api/Products

[HttpGet]

public async Task<ActionResult<IEnumerable<Products>>> GetProducts(bool? inStock, int? skip, int? take)

{

var products = \_context.Products.AsQueryable();

if (inStock != null) // Adds the condition to check availability

{

products = \_context.Products.Where(i => i.AvailableQuantity > 0);

}

if (skip != null)

{

products = products.Skip((int)skip);

}

if (take != null)

{

products = products.Take((int)take);

}

return await products.ToListAsync();

}

// GET: api/Products/5

[HttpGet("{id}")]

public async Task<ActionResult<Products>> GetProducts(int id)

{

var products = await \_context.Products.FindAsync(id);

if (products == null)

{

return NotFound();

}

return products;

}

// PUT: api/Products/5

// To protect from overposting attacks, enable the specific properties you want to bind to, for

// more details, see https://go.microsoft.com/fwlink/?linkid=2123754.

[HttpPut("{id}")]

public async Task<IActionResult> PutProducts(int id, Products products)

{

if (id != products.ProductId)

{

return BadRequest();

}

\_context.Entry(products).State = EntityState.Modified;

try

{

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!ProductsExists(id))

{

return NotFound();

}

else

{

throw;

}

}

return NoContent();

}

// POST: api/Products

// To protect from overposting attacks, enable the specific properties you want to bind to, for

// more details, see https://go.microsoft.com/fwlink/?linkid=2123754.

[HttpPost]

public async Task<ActionResult<Products>> PostProducts(Products products)

{

\_context.Products.Add(products);

await \_context.SaveChangesAsync();

return CreatedAtAction("GetProducts", new { id = products.ProductId }, products);

}

// DELETE: api/Products/5

[HttpDelete("{id}")]

public async Task<ActionResult<Products>> DeleteProducts(int id)

{

var products = await \_context.Products.FindAsync(id);

if (products == null)

{

return NotFound();

}

\_context.Products.Remove(products);

await \_context.SaveChangesAsync();

return products;

}

private bool ProductsExists(int id)

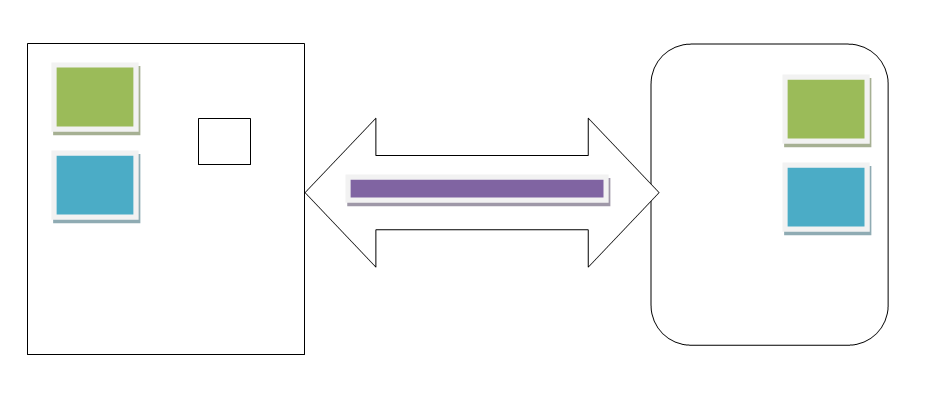
{

return \_context.Products.Any(e => e.ProductId == id);

}

}

}



Client (MVC App) Server (Web API)

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

//Added...

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace InventoryServiceClient.Models

{

public class UserInfo

{

[Key]

[DatabaseGenerated(DatabaseGeneratedOption.Identity)]

public int UserId { get; set; }

public string Email { get; set; }

public string Password { get; set; }

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace InventoryServiceClient.Models

{

public partial class Product

{

[Key]

[DatabaseGenerated(DatabaseGeneratedOption.Identity)]

public int ProductId { get; set; }

public string Name { get; set; }

public string Category { get; set; }

public string Color { get; set; }

public decimal UnitPrice { get; set; }

public int AvailableQuantity { get; set; }

}

}

using System;

using System.Linq;

using System.Collections.Generic;

///Default

using InventoryServiceClient.Models;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Logging;

using System.Diagnostics;

using System.Threading.Tasks;

///Added...

using System.Text;

using System.Net.Http;

using Newtonsoft.Json;

using Microsoft.AspNetCore.Http;

namespace InventoryServiceClient.Controllers

{

public class HomeController : Controller

{

private readonly ILogger<HomeController> \_logger;

public HomeController(ILogger<HomeController> logger)

{

\_logger = logger;

}

public IActionResult Index()

{

return View();

}

public async Task<IActionResult> LoginUser(UserInfo user)

{

HttpClientHandler clientHandler = new HttpClientHandler();

clientHandler.ServerCertificateCustomValidationCallback = (sender, cert, chain, sslPolicyErrors) => { return true; };

using (var httpClient = new HttpClient(clientHandler))

{

StringContent stringContent = new StringContent(JsonConvert.SerializeObject(user), Encoding.UTF8, "application/json");

using (var response = await httpClient.PostAsync("https://localhost:44378/api/token", stringContent))

{

string token = await response.Content.ReadAsStringAsync();

if (token == "Invalid credentials")

{

ViewBag.Message = "Incorrect Email and Password";

return Redirect("~/Home/Index");

}

HttpContext.Session.SetString("JWToken", token);

}

return Redirect("~/Products/Index");

}

}

public IActionResult Privacy()

{

return View();

}

[ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]

public IActionResult Error()

{

return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });

}

}

}

@model InventoryServiceClient.Models.UserInfo

@{

ViewData["Title"] = "Home Page";

}

<div class="text-center">

<h1 class="display-4">Welcome</h1>

</div>

<div class="row">

<div class="col-md-4">

<form asp-action="LoginUser" asp-controller="Home">

<div class="form-group">

<label asp-for="Email" class="control-label">Email</label>

<input asp-for="Email" class="form-control" />

<span asp-validation-for="Email" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="Password" class="control-label">Password</label>

<input asp-for="Password" class="form-control" />

<span asp-validation-for="Password" class="text-danger"></span>

</div>

<div class="form-group">

<input type="submit" value="login" class="btn btn-primary" />

</div>

<div asp-validation-summary="ModelOnly" class="text-danger">

</div>

<label>@Html.Raw(ViewBag.Message)</label>

</form>

</div>

</div>

using System;

using System.Linq;

using System.Collections.Generic;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

//Added...

using InventoryServiceClient.Models; //for Products

using System.Net.Http; //for HttpClient

using System.Net.Http.Headers; //for GetStringAsync

using Microsoft.AspNetCore.Http; //for Session

using Newtonsoft.Json; //for JsonConvert

namespace InventoryServiceClient.Controllers

{

public class ProductsController : Controller

{

public static string baseURL = "https://localhost:44378/api/products";

public async Task<IActionResult> Index()

{

return View(await GetProducts());

}

[HttpGet]

public async Task<List<Product>> GetProducts()

{

var accessToken = HttpContext.Session.GetString("JWToken");

HttpClientHandler clientHandler = new HttpClientHandler();

clientHandler.ServerCertificateCustomValidationCallback = (sender, cert, chain, sslPolicyErrors) => { return true; };

HttpClient client = new HttpClient(clientHandler);

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", accessToken);

string JsonStr = await client.GetStringAsync(baseURL);

var result = JsonConvert.DeserializeObject<List<Product>>(JsonStr).ToList();

return result;

}

}

}

@model IEnumerable<InventoryServiceClient.Models.Product>

@{

ViewData["Title"] = "Index";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<h1>Index</h1>

<p>

<a asp-action="Create">Create New</a>

</p>

<table class="table">

<thead>

<tr>

<th>

@Html.DisplayNameFor(model => model.ProductId)

</th>

<th>

@Html.DisplayNameFor(model => model.Name)

</th>

<th>

@Html.DisplayNameFor(model => model.Category)

</th>

<th>

@Html.DisplayNameFor(model => model.Color)

</th>

<th>

@Html.DisplayNameFor(model => model.UnitPrice)

</th>

<th>

@Html.DisplayNameFor(model => model.AvailableQuantity)

</th>

<th></th>

</tr>

</thead>

<tbody>

@foreach (var item in Model) {

<tr>

<td>

@Html.DisplayFor(modelItem => item.ProductId)

</td>

<td>

@Html.DisplayFor(modelItem => item.Name)

</td>

<td>

@Html.DisplayFor(modelItem => item.Category)

</td>

<td>

@Html.DisplayFor(modelItem => item.Color)

</td>

<td>

@Html.DisplayFor(modelItem => item.UnitPrice)

</td>

<td>

@Html.DisplayFor(modelItem => item.AvailableQuantity)

</td>

<td>

@Html.ActionLink("Edit", "Edit", new { id=item.ProductId }) |

@Html.ActionLink("Details", "Details", new { id = item.ProductId }

@Html.ActionLink("Delete", "Delete", new { id = item.ProductId })

</td>

</tr>

}

</tbody>

</table>