Securing microservices

Securing the Users REST API

The last REST API that we need to secure is the Users REST API. Since this REST API lives in the Identity microservice itself, we will give it a bit of special treatment and for that we will use a special IdentityServer API scope and configuration methods as opposed to the standard ASP.NET Core methods we used with the other microservices.

In Identity repo

1. Update **appsettings.json**:

```
"IdentityServerSettings": {
  "ApiScopes": [
    {
      "Name": "catalog.fullaccess"
    },
    {
      "Name": "inventory.fullaccess"
    },
    {
      "Name": "IdentityServerApi"
    }
  ],
```

2. Update **appsettings.Development.json**:

```
"IdentityServerSettings": {
  "Clients": [
    {
      …
      "RedirectUris": [
        "urn:ietf:wg:oauth:2.0:oob"
      ],
      "AllowedScopes": [
        "openid",
        "profile",
        "catalog.fullaccess",
        "inventory.fullaccess",
        "IdentityServerApi"
```

```
      ],
        "AlwaysIncludeUserClaimsInIdToken": true
      }
    ]
  }
```

3. Update **Startup.cs**:

```
public void ConfigureServices(IServiceCollection services)
{
    …
    services.AddIdentityServer()
         .AddAspNetIdentity<ApplicationUser>()
         .AddInMemoryApiScopes(identityServerSettings.ApiScopes)
         .AddInMemoryApiResources(identityServerSettings.ApiResources)
         .AddInMemoryClients(identityServerSettings.Clients)
         .AddInMemoryIdentityResources(identityServerSettings.IdentityResources)
         .AddDeveloperSigningCredential();

    services.AddLocalApiAuthentication();

    services.AddControllers();
    …
}
```

4. Update **UsersController.cs**:

```
{
[ApiController]
[Route("users")]
[Authorize(Policy = LocalApi.PolicyName)]
public class UsersController : ControllerBase
{
  …
}
```

5. Start Identity service and request token with **IdentityServerApi** scope

6. Try a GET /users with new token (200 OK)

In the next lesson we will introduce roles to our Identity microservice.