Securing microservices

(Demo prep)

- Remove all tokens from Postman
- Clear Authorization values in Postman
- Delete player1 from DB

User Authentication via OpenID Connect

Let's enable our Postman app to request tokens from our Identity microservice.

In Identity Microservice

1. Delete **ScaffoldingReadme.txt**

2. Start web server

3. Register the player1@play.com user in the browser

   https://localhost:5003/Identity/Account/register

4. Log out the user

5. Update **appsettings.Development.json**:

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  },
"IdentityServerSettings": {
    "Clients": [
      {
        "ClientId": "postman",
        "AllowedGrantTypes": [
          "authorization_code"
        ],
        "RequireClientSecret": false,
        "RedirectUris": [
          "urn:ietf:wg:oauth:2.0:oob"
        ],
        "AllowedScopes": [
          "openid"
        ]
      }
    ]
  }
}
```

6. Update **IdentityServerSettings.cs**:

```
namespace Play.Identity.Service.Settings
{
    public class IdentityServerSettings
    {
        ...
        public IReadOnlyCollection<Client> Clients { get; init; }
        ...
}
```

7. Update **Startup.cs**:

```
public void ConfigureServices(IServiceCollection services)
{
    ...
    var mongoDbSettings = Configuration.GetSection(nameof(MongoDbSettings)).Get<MongoDbSettings>();
    var identityServerSettings =
Configuration.GetSection(nameof(IdentityServerSettings)).Get<IdentityServerSettings>();

    services.AddDefaultIdentity<ApplicationUser>()
    ...

    services.AddIdentityServer(options =>
        {
            options.Events.RaiseSuccessEvents = true;
            options.Events.RaiseFailureEvents = true;
            options.Events.RaiseErrorEvents = true;
        })
        .AddAspNetIdentity<ApplicationUser>()
        ...
}
```

8. Request an ID Token with:

Field	**Value**	**Description**

Grant type	Authorization Code (with PKCE)	"Proof Key for Code Exchange" extension of the OAuth 2.0 spec for Authorization Code Grant flows
Callback URL	urn:ietf:wg:oauth:2.0:oob	The redirect URI is the URL within your application that will receive OAuth 2.0 credentials. The redirect URI for installed applications is **urn:ietf:wg:oauth:2.0:oob**
Auth URL	https://localhost:5003/connect/authorize or https://localhost:5003/connect/authorize?prompt=login	The endpoint used to get the authorization code
Access token URL	https://localhost:5003/connect/token	The endpoint used to exchange the authorization code for an access token
Client ID	postman	The identifier of the client
Scope	openid	The scope of the access request

9. Show the logs in Console

10. Sign in player1@play.com

11. Show the logs in Console again

12. Explore the generated **ID Token** in https://jwt.ms or https://jwt.io

13. Update **IdentityServerSettings.cs**:

```
namespace Play.Identity.Service.Settings
{
    public class IdentityServerSettings
    {
        ...
        public IReadOnlyCollection<IdentityResource> IdentityResources =>
            new IdentityResource[]
            {
                new IdentityResources.OpenId(),
                new IdentityResources.Profile()
            };
    }
}
```

14. Update **appsettings.Development.json**:

```
"IdentityServerSettings": {
  "Clients": [
    {
      "ClientId": "postman",
      "AllowedGrantTypes": [
        "authorization_code"
      ],
      "RequireClientSecret": false,
      "RedirectUris": [
        "urn:ietf:wg:oauth:2.0:oob"
      ],
      "AllowedScopes": [
        "openid",
        "profile"
      ],
      "AlwaysIncludeUserClaimsInIdToken": true
    }
  ]
}
```

15. Start server and request the additional **profile** scope in Postman.

16. Decode **ID Token** and show the new claims (**preferred_username**, **name**).

In the next lesson we will learn more about the structure and value of these tokens.