WIP

LICENSE MIT

Built with: dotnet8

Prerequisites: 📋

This project was made with the purpose to attend only applications that follows the current .Net Supported versions.

Why Feijuca? 🫘

Feijuca is a nickname for a famous Brazilian dish called Feijoada. I wanted to use a name representing my country on this project, and Feijuca was chosen.

About the project: 🧾

This repository aims to provide a configuration option for .NET projects that are using or planning to use Keycloak for authentication and authorization. The project consists of two distinct parts:

1. **Feijuca.Keycloak.Auth.MultiTenancy**
2. **Feijuca.Keycloak.TokenManager**

Attention: 👊

The projects work in isolation way, there is no dependency between them. **You do not need use one to use other**, note that each project has different purpose.

Below, you can understand better the purpose about which one project. 👇

Feijuca.Keycloak.Auth.MultiTenancy 💻

It is a NuGet package that enables the implementation of multi-tenancy concepts using Keycloak. With this package, each realm acts as a different tenant, allowing for unique configurations for each one. This ensures that each tenant within your application can have its own settings and configurations within Keycloak.

Features ⛲

With this package you can:

- Use all Keycloak features following a multi-tenancy concept based on your realms, so you can handle different configurations based on each tenant (realm).
- Get information from a token, such as: finding claims, finding out which tenant this token belongs to, which user this token belongs to, and so on. (See more)

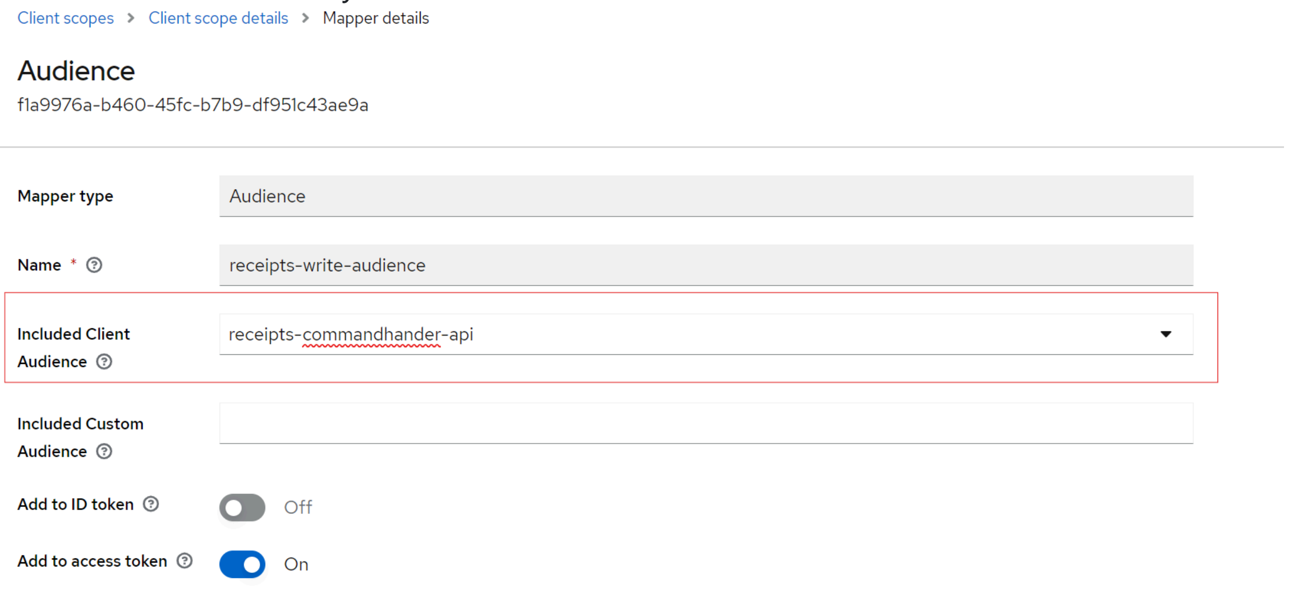
- **(If you want to implement a feature to retrieve something else related to the token, open a PR)**

Getting Started on Feijuca.Keycloak.Auth.MultiTenancy

- Prerequisites It is assumed that you already have your Keycloak instance configured, including the creation of clients with their respective settings (scopes, etc.).

- Keycloak configuration steps:

  - 1. **Configuring audience**: Create a new audience related to the scopes used your client and include the audience on your client:



Audience
f1a9976a-b460-45fc-b7b9-df951c43ae9a

Mapper type	Audience
Name *	receipts-write-audience
Included Client Audience	receipts-commandhander-api
Included Custom Audience	
Add to ID token	Off
Add to access token	On

This step is important and mandatory because on each request received the tool will validate the token audience.

- Project configurations steps:

  - 1. appsettings.json Filled out appsettings file on your application, relate all of yours realms (tenants) `sh { "AuthSettings": { "Realms": [{ "Name": "yourTenantName1", "Audience": "your-audience-defined-on-step1", "Issuer": "https://url-keycloakt/realms/yourTenantName1" }, { "Name": "yourTenantName2", "Audience": "your-audience-defined, "Issuer": "https://url-keycloakt/realms/yourTenantName2" }, { "Name": "yourTenantName3", "Audience": "your-audience-defined", "Issuer": "https://url-keycloakt/realms/yourTenantName3" }], "ClientId": "your-client-id", "ClientSecret": "your-client-secret", "AuthServerUrl": "https://url-keycloak" } }`
  - 2. Get appsettings values:

    Map appsettings configurations values (Note that AuthSettings is a model defined on **Feijuca.Keycloak.Auth.MultiTenancy**, I recommend you use the GetSection method to map the appsettings configs to the AuthSettings model:

```
    var settings = configuration.GetSection("AuthSettings").Get<AuthSettings>();
```

- 3. Add dependency:

  Add the service to the service collection from your application, I recommend you create a new extension method as below:

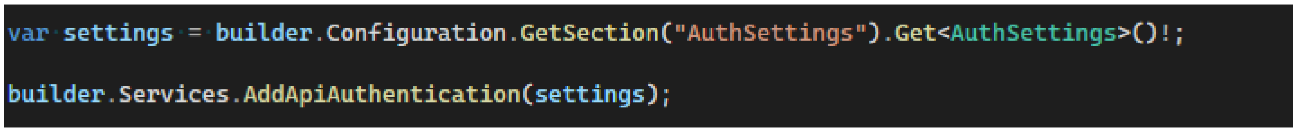
```
public static class AuthExtension
 {
     public static IServiceCollection AddApiAuthentication(this IServiceCollection
services, AuthSettings authSettings)
     {
         services.AddHttpContextAccessor();
         services.AddSingleton<JwtSecurityTokenHandler>();
         services.AddKeyCloakAuth(authSettings!);

         return services;
     }
 }
```

And after it, call it on your Program.cs:

```
builder.Services.AddApiAuthentication(applicationSettings.AuthSettings);
```

- 4. Conclusion:

  Your configs should be like:



```
var settings = builder.Configuration.GetSection("AuthSettings").Get<AuthSettings>()!;

builder.Services.AddApiAuthentication(settings);
```

  And with this configuration you should be able to use Keycloak following a multi tenancy contenxt using .NET.

  Following this link you can understand what is the logic used to validate the token received.

Feijuca.Keycloak.TokenManager 👨🏽‍💻

Managing certain actions in the Keycloak API can be complicated. For example, creating a new user involves several steps: obtaining a token, creating the user, setting attributes, and setting a password. Feijuca.Keycloak.TokenManager aims to simplify these processes and abstract the complexity related to Keycloak API calls.

Feijuca.Keycloak.TokenManager is an API that abstracts, facilitates and simplifies calls to perform actions in Keycloak. Over time, the goal is to encapsulate multiple Keycloak endpoints, making it easier to perform actions that would be more complex using just the Keycloak API.

Features ⛲

- Every action in one place. Forget about call multiples endpoints to do actions about users on keycloak. Do actions related to the user (Creation, remotion, e-mail confirming, password redefinition, and so on) based on predefined endpoints.
- Custom endpoints based on your necessities (If you think it could be helpful to the project, open a PR to discuss additional features).

Getting Started - Using Token Manager Api

- Keycloak configuration steps:
    - 1. **Giving permissions to the realm:** To be possible manage users using the Keycloak Api, it is necessary to provide some permissions on your keycloak client. You can handle it on an existing realm, or you can create a new realm. You can follow this link to understand how provide these permissions.
    - 2. Once you created/configureted a realm to have permissions related to users handling, enough you change the appsettings setting the values related to the created/configured realm.

```
{
 "Settings": {
   "AuthSettings": {
     "Realms": [
       {
          "Name": "yourTenantName1",
          "Audience": "your-audience-defined-on-step1",
          "Issuer": "https://url-keycloakt/realms/yourTenantName1"
       }
     ],
      "ClientId": "your-client-id",
      "ClientSecret": "your-client-secret",
      "Resource": "",
      "AuthServerUrl": ""
   }
 }
}
```

Contributing

This is a project in costant evolution, therefore, if you have some suggestion, enter in contact with me or open a pull request and we can discuss.

License

Distributed under the MIT License. See `LICENSE.txt` for more information.

Contact

LINKEDIN

Namespace TokenManager.Api.Controllers

Classes

ClientController

GroupController

GroupRolesController

GroupUsersController

RoleController

UserController