Xavier Hahn · Follow
13 de novembro de 2017 · 5 minutos de leitura · Ouvir

Adicionando autorização ao aplicativo Asp.Net Core usando o Keycloak

Usando as funcionalidades de funções do Keycloak para configurar a autorização em aplicativos web e APIs Asp.Net Core.

Esta é uma continuação da minha história anterior, onde expliquei como configurar um aplicativo Asp.Net Core e um aplicativo Angular para autenticar usuários por meio de um servidor Keycloak usando OpenID Connect. Neste artigo, daremos um passo adiante e adicionaremos autorização usando as "funções" internas do Keycloak.

Autorização (geralt do pixabay)

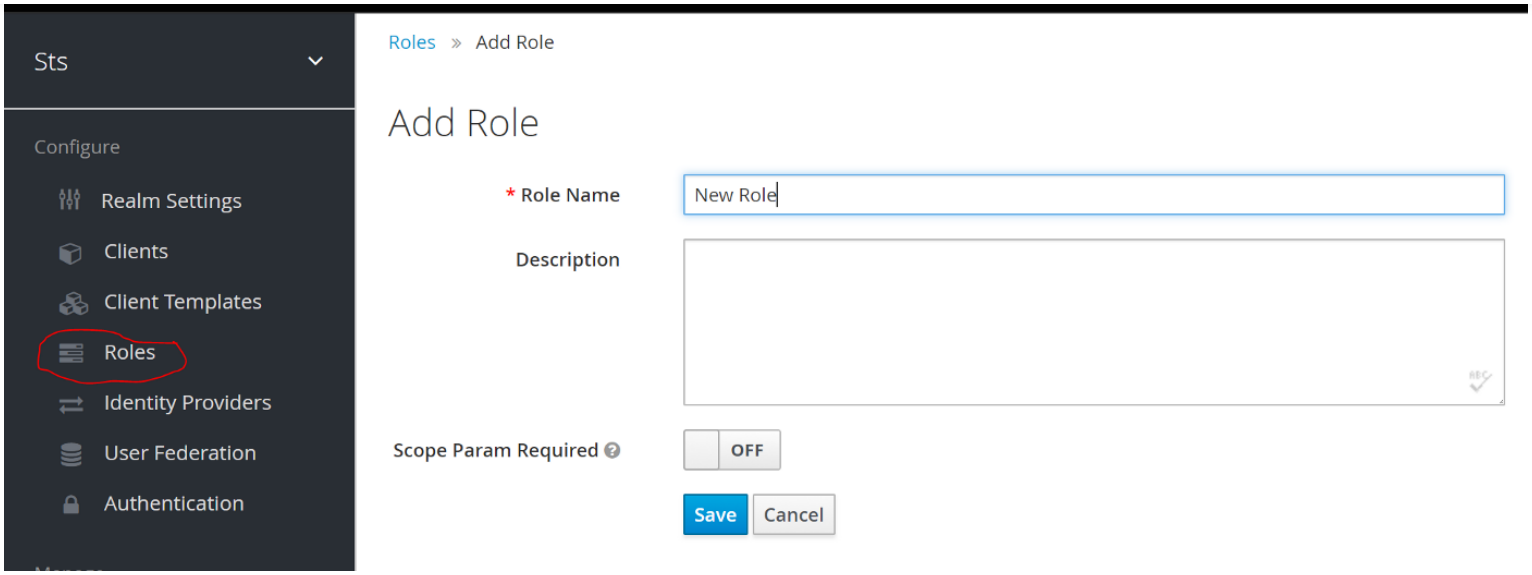
Configurando o Keycloak

Não entrarei em todos os detalhes sobre como configurar um servidor Keycloak neste artigo, pois isso foi abordado extensivamente no artigo anterior. Portanto, você deve ter um servidor Keycloak em execução, com alguns usuários configurados e um aplicativo cliente configurado para sua API Asp.Net Core.

Os papéis do Keycloak

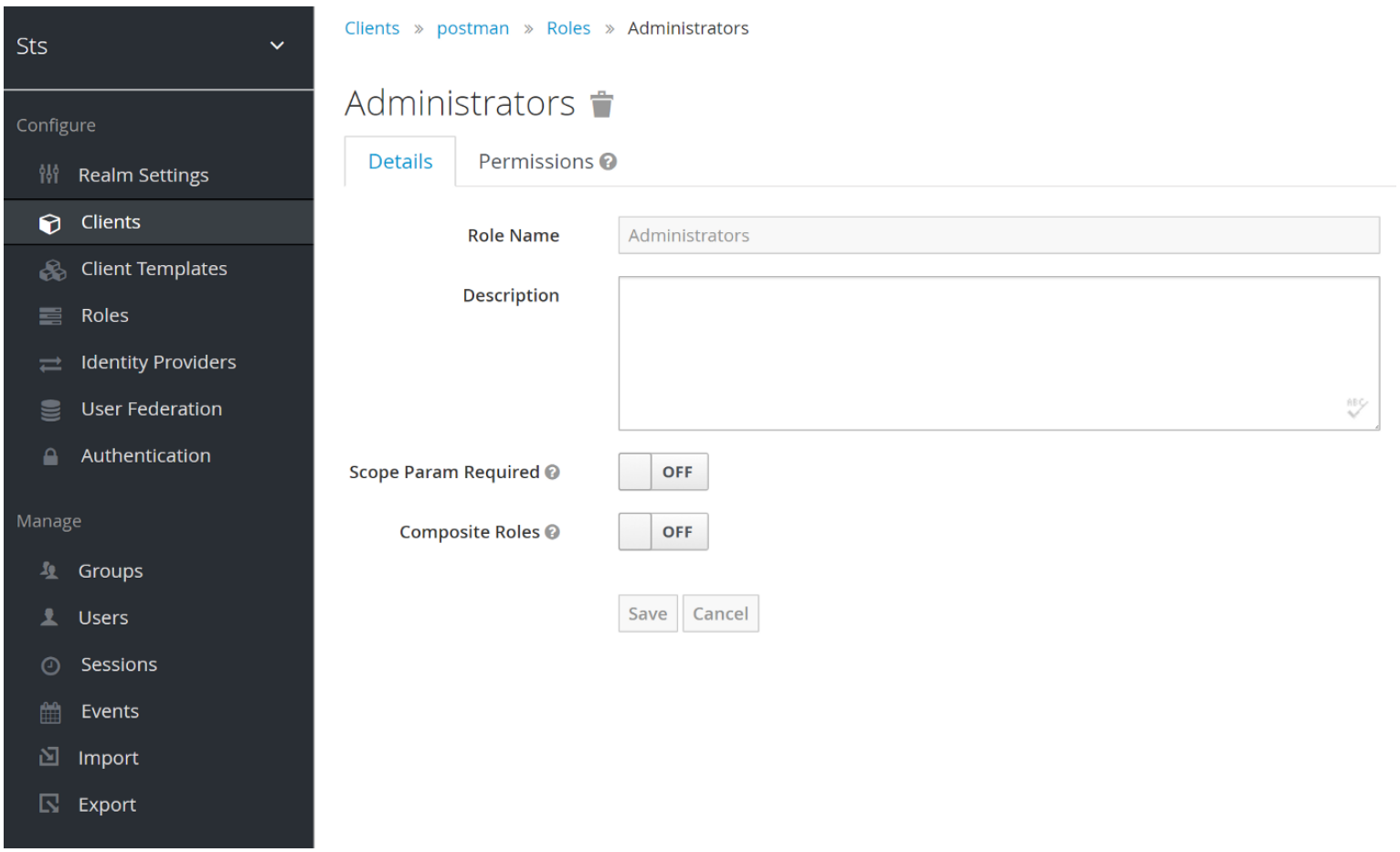
Existem dois tipos de funções no Keycloak, funções de território *e funções* de *aplicativo*. As funções de realm são funções que são configuradas no nível "Realm", isso é útil se você precisar "compartilhar" funções entre os limites dos aplicativos. Por exemplo, se você tiver funções globais para a empresa, é aí que você deseja adicioná-las.

Para adicionar funções de realm, basta conectar-se ao console administrativo do Keycloak, selecionar a função para a qual deseja configurar funções e clicar na guia *Funções* . Ele listará as funções atuais disponíveis no reino, clique no botão *Adicionar função* . Adicione o nome da função, selecione salvar e você terá sua nova função.



Adicionando uma nova função

Funções de clientes são funções que estão disponíveis apenas para um aplicativo cliente específico. Esse é um cenário muito provável, pois os usuários que são administradores de um aplicativo podem ser usuários simples de outro. Para adicionar funções de cliente, você precisa selecionar o menu *Clientes* e, em seguida, selecionar o cliente ao qual deseja adicionar funções. Selecione o cliente que criamos para nosso aplicativo no artigo anterior. Em seguida, selecione a guia *Funções , você deverá ver uma lista vazia.* Clique em *Adicionar função* e adicione uma função chamada *Administradores.* Ao salvar, você deve acessar os detalhes da função recém-criada.



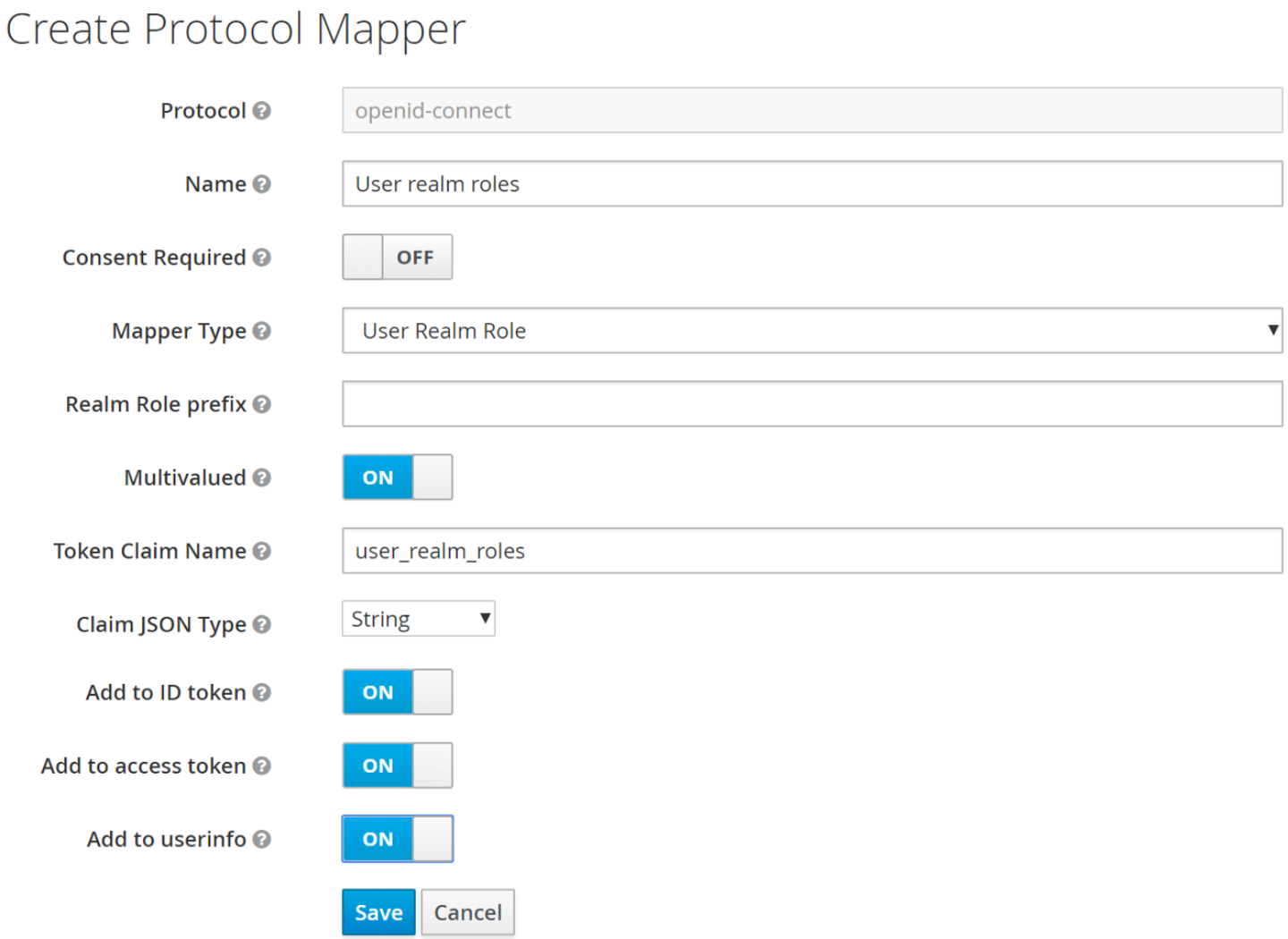
Adicionando uma função a um usuário

Agora, vamos ao menu *Usuários e mapear uma função para um usuário.* Na lista de usuários, procure um usuário e clique em seu *id* para acessar sua página de detalhes. A partir daí, selecione a guia *Mapeamento de função* e você deverá ver a função de realm que criou na parte superior. Selecione o cliente na lista suspensa e você deverá ver a função *Administradores* que você criou anteriormente.

Esta é uma configuração que precisamos fazer nas opções do cliente do Keycloak. Volte para a página de opções do cliente do aplicativo .Net Core e selecione a guia *Mapeadores* . Você verá vários mapeadores integrados.

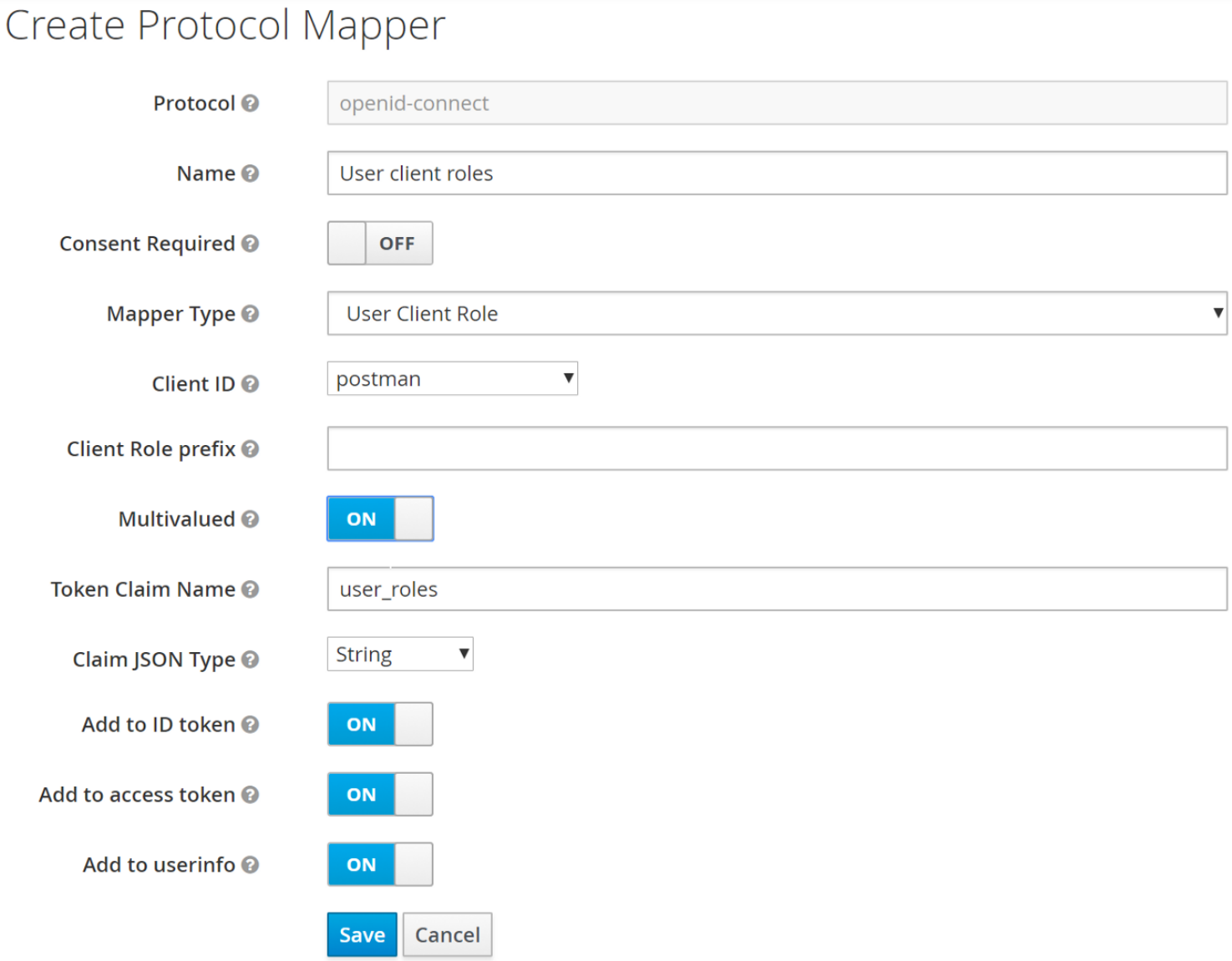
Clique no botão *Criar* e preencha o formulário da seguinte forma:

- **Nome:** pode ser o que você quiser, estou usando “Funções do domínio do usuário”
- **Tipo de mapeador:** funções do domínio do usuário
- **Multivalorado:** verdadeiro
- **Token Claim Name:** novamente, o que você quiser, eu uso user_realm_roles
- **Tipo JSON de reivindicação:** string
- **Adicionar ao token de ID, adicionar ao token de acesso, adicionar às informações do usuário:** true



Crie outro mapeador, desta vez para a função de cliente.

- **Nome:** pode ser o que você quiser, estou usando “Funções de cliente do usuário”
- **Tipo de mapeador:** funções de cliente do usuário
- **ID do cliente:** o ID do cliente do seu aplicativo
- **Multivalorado:** verdadeiro
- **Token Claim Name:** novamente, o que você quiser, eu uso user_roles



> *Observação: aqui estou criando uma função de cliente e uma função de realm, mas isso não é obrigatório, dependendo do seu aplicativo, você pode ter apenas funções de cliente, apenas funções de realm ou ambas.*

Testando a configuração

Podemos fazer um pequeno teste usando Postman e jwt.io para verificar se nossos mapeamentos foram enviados corretamente nas declarações.

Abra o Postman, selecione a guia *Authorization de qualquer solicitação e clique no botão Get New Access Token* . Insira um nome de token e selecione *Código de autorização* para *Tipo de concessão* . Preencha o restante do formulário da seguinte forma:

- **URL de retorno:** qualquer coisa, mas certifique-se de ter adicionado à lista de URLs permitidos para o cliente no Keycloak
- **URL de autenticação:** http://localhost:8080/auth/realms/master/protocol/openid-connect/auth
- **URL do token de acesso:** http://localhost:8080/auth/realms/master/protocol/openid-connect/token
- **ID do cliente:** o ID do cliente do seu aplicativo

Os demais campos podem ser deixados como padrão. Clique em *Solicitar Token* e autentique-se com o usuário para o qual você adicionou as funções. Isso salvará seu token de acesso. Copie-o e cole-o em jwt.io . Isso descriptografará o token, permitindo que você inspecione o conteúdo. Você deve ter *user_realm_roles* e *user_roles* . Ótimo, agora estamos prontos para adicionar a autorização à API Web do Asp.Net Core.

Adicionando autorização à API Web Asp.Net Core

Começaremos com a solução .Net Core como a deixamos no final do artigo anterior. Vamos usar a autorização baseada em função incorporada do

Abra o arquivo *Startup.cs* e adicione as seguintes linhas após a `AddAuthentication` chamada no `ConfigureServices` método.

```
services.AddAuthorization(options =>
{
  options.AddPolicy("Administrator", policy => policy.RequireClaim("user_roles", "[Administrator]"));
});
```

startup.cs hosted with ❤ by **GitHub** **view raw**

Agora, podemos simplesmente adicionar o atributo de autorização com a propriedade de função definida como Administradores às classes ou métodos do controlador que desejamos proteger.

```
[HttpGet("authorization")]
[Authorize(Roles = "Administrators")]
public string TestAuthorization()
{
    return "Authorization is working!";
}
```

testcontroller.cs hosted with ❤ by **GitHub** **view raw**

Execute o projeto e use o Postman para testar seu aplicativo com a autorização funcionando corretamente. Somente usuários com a função de administradores devem ter permissão para acessar métodos.

> *Observação: Como você pode ver, o RequireClaim parece não entender que a declaração é uma matriz e a trata como uma string, e é por isso que você precisa dos caracteres []. Isso significa que realmente não funcionará nos casos em que o usuário tiver mais de uma função. Não tenho uma resposta definitiva sobre como corrigir esse problema. A única opção que vejo por enquanto é implementar uma autorização personalizada baseada em política .*

Conclusão

Como você pode ver, adicionar autorização a uma API Web Asp.Net Core a partir das funções do Keycloak é relativamente fácil. Como o Keycloak permite funções do tipo cliente e do tipo realms, ele oferece uma ampla gama de possibilidades.

Atualizei o código-fonte do artigo anterior para adicionar o exemplo de autorização deste artigo.

Obrigado por ler este artigo, sinta-se à vontade para adicionar seus comentários, observações ou perguntas e compartilhar este artigo.