MVP Community Meetup Chengdu

基于 IdentityServer4 单点登录系统设计

李志强

MVP
Community
Meetup
Chengdu

自我介绍

- 李志强 （网络ID：晓晨Master）
- 微软最有价值专家（MVP）
- .NET Core Community(NCC) 成员
- 开源爱好者、爱玩 github
- 喜欢并专注于前沿技术
- 喜爱写作、经常发布技术博客（博客园）

MVP
Community
Meetup
Chengdu

目录
CONTENTS

01 IdentityServer4 介绍

02 IdentityServer4 应用

03 IdentityServer4 单点登录设计

04 Q&A

MVP
Community
Meetup
Chengdu

IdentityServer4 是什么?

IdentityServer4 是什么?

IdentityServer4 是基于 ASP.NET Core 的实现 OpenId Connect 和 OAuth 2.0 协议的框架。它具有以下特点:

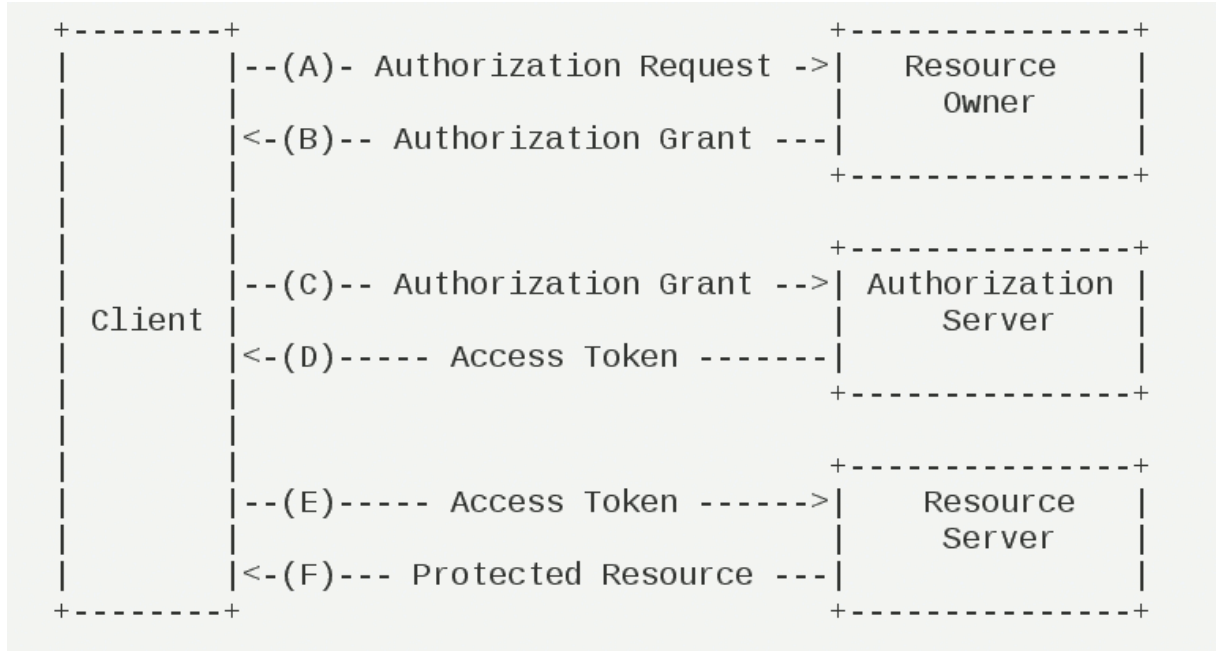
- 身份认证即服务
- 单点登录/注销
- API访问控制
- 联盟网关（Federation Gateway）
- 自由扩展
- 成熟的开源社区生态
- 免费和商业支持

应用场景

- 微服务架构中的身份认证服务以及各个微服务的访问控制
- 多类型应用程序统一单点登录/注销
- 开放平台
- 无状态用户认证
- …

OAuth 2.0 协议

OAuth 2.0 是目前最流行的授权机制，用来授权第三方应用（客户端），获取用户数据



OpenId Connect 协议

OpenID Connect和OAuth 2.0非常相似。事实上OpenID Connect是OAuth 2.0之上的扩展。解决两个基本的安全问题，即身份验证和API访问，被合并为一个协议，通常只需一次往返认证服务。

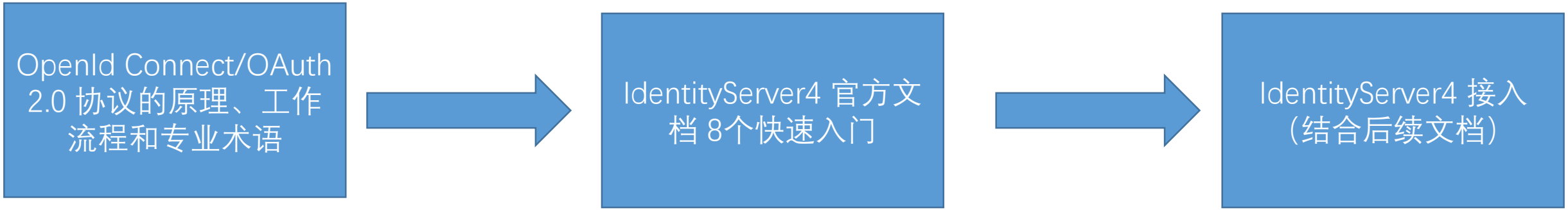
授权模式

- 客户端模式
- 密码模式
- 简单模式
- 授权码模式
- 混合模式（OpenId Connect）

常用术语

- Scope
- API Resource
- Client
- Identity Resource

学习路线


OpenId Connect/OAuth 2.0 协议的原理、工作流程和专业术语
IdentityServer4 官方文档 8个快速入门
IdentityServer4 接入（结合后续文档）


MVP

学习路线-后续文档

MVP Community

TOPICS

- Startup
- Defining Resources
- Defining Clients
- Sign-in
- Sign-in with External Identity Providers
- Windows Authentication
- Sign-out
- Sign-out of External Identity Providers
- Federated Sign-out
- Federation Gateway
- Consent
- Protecting APIs
- Deployment
- Logging
- Events
- Cryptography, Keys and HTTPS
- Grant Types
- Secrets
- Extension Grants
- Resource Owner Password Validation
- Refresh Tokens
- Reference Tokens
- Mutual TLS
- Authorize Request Objects
- Custom Token Request Validation and Issuance
- CORS
- Discovery
- Adding more API Endpoints
- Adding new Protocols
- Tools

REFERENCE

- Identity Resource
- API Resource
- Client
- GrantValidationResult
- Profile Service
- IdentityServer Interaction Service
- Device Flow Interaction Service
- IdentityServer Options
- Entity Framework Support
- ASP.NET Identity Support

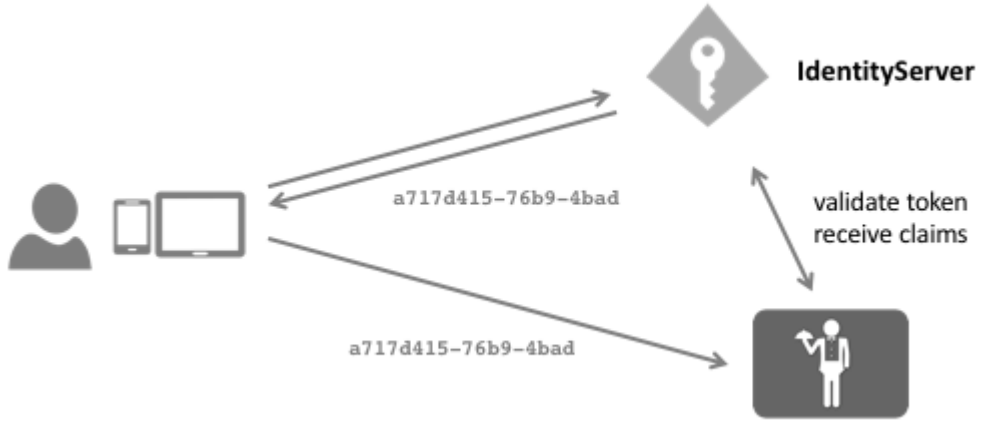
Google & Github Issues

Community
Meetup
Chengdu

IdentityServer4 应用

Reference Token

IdentityServer 会将令牌的内容存储在数据存储中，并且只会将此令牌的唯一标识符发回给客户端。然后，接收此引用的API必须与IdentityServer交互通信以验证令牌。




```
client.AccessTokenType = AccessTokenType.Reference;
```

Json Web Token(JWT)

JSON Web Token (JWT)是一个开放标准(RFC 7519)，它定义了一种紧凑的、自包含的方式，用JSON对象在各方之间安全地传输信息。该信息可以被验证和信任，因为它是数字签名的。

Json Web Token 示例

MVP Community

Encoded PASTE A TOKEN HERE


```
eyJhbGciOiJSUzI1NiIsImtpZCI6ImI5OGQ5Njgw
OWQ1MGU4MjM1NmM3M2FjNWQ5MjI2YmNiIiwidHlw
IjoiSldUIn0.eyJuYmYiOjE1NDg4MTIwNDAsImV4
cCI6MTU0ODgxNTY0MCwiaXNzIjoiaHR0cDovL2xv
Y2FsaG9zdDo1MDAwIiwiYXVkIjpbImh0dHA6Ly9s
b2NhbGhvc3Q6NTAwMC9yZXNvdXJjZXMiLCJhcGkx
Il0sImNsaWVudF9pZCI6InJvLmNsaWVudCIsInN1
YiI6IjEiLCJhdXRoX3RpbWUiOjE1NDg4MTIwNDAs
ImlkcCI6ImxvY2FsIiwic2NvcGUiOlsiYXBpMSJd
LCJhbXIiOlsicHdkIl19.ixDLOb0loWo19qi4yz4
WT9kp8sZkJAXoMnja8M-
90OXexBEZptMZBJkeE3Iv6h9wmv7JBaQYtbEO3FK
BrAabOMCculfwKRY8-
WixAHjjcqlF53p7YYMNm7B08RALBO1Mvd3Jz6YnZ
RpqWBgtkpsPoOxk_QBzkanmVGajVbNBHQ8szOAVw
SC9fnM4VZgH5hwoN_LSOz7wHlmH8b7IS053T_vMJ
LkCH6Ngt6Pg5TXyjNzAhMA2Hxi56T3uQR8vEz50A
EdIvIYhrbBYjO5sTAGM_6RkIhirPI-
M1jzdHSK4JmdlAjcru4cChpjXkTUb8Ubwr__2PGY
HFI2Gzfgpmbxsog
```


Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE


```
{
  "alg": "RS256",
  "kid": "b98d96809d50e82356c73ac5d9226bcb",
  "typ": "JWT"
}
```

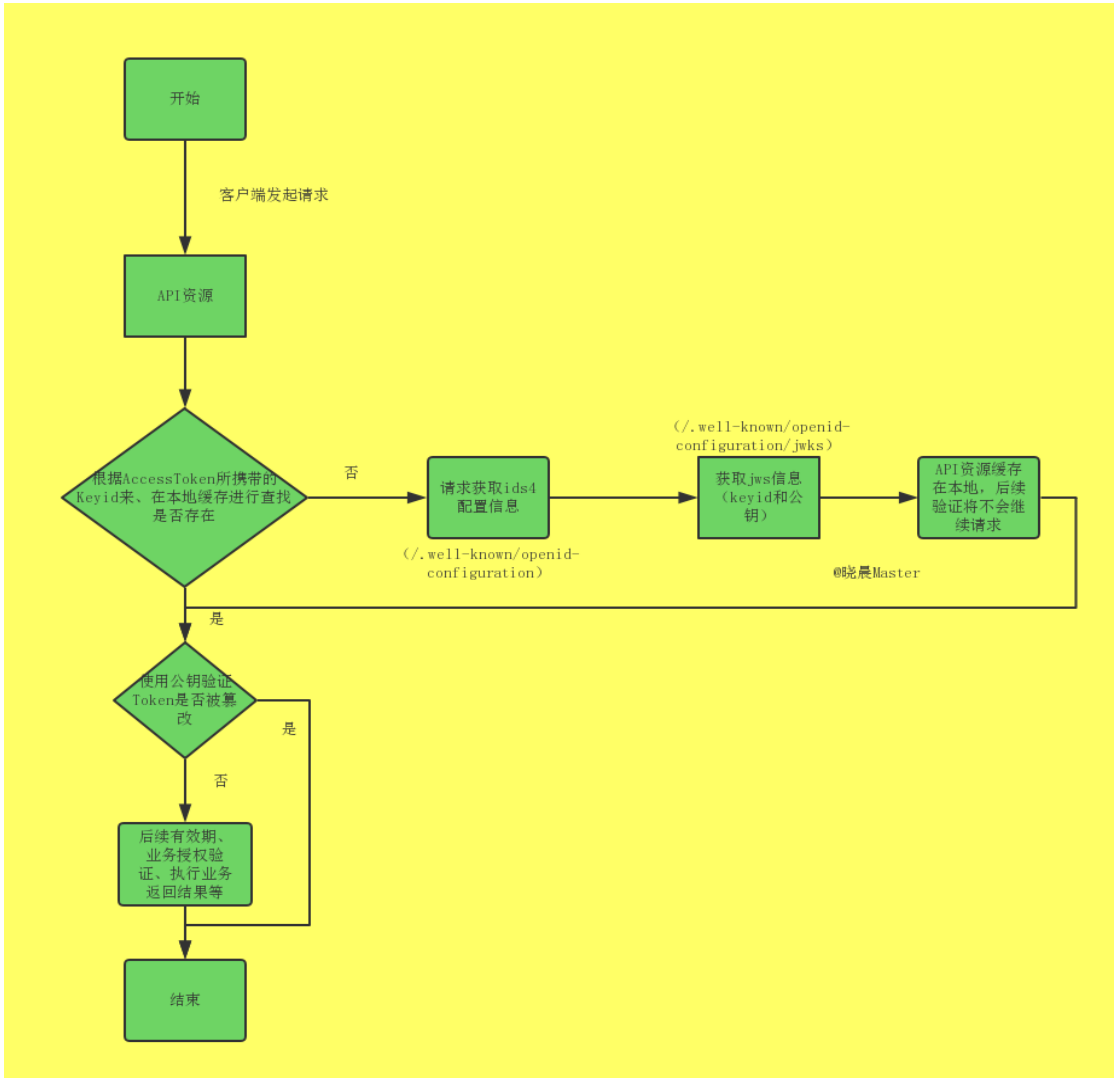

PAYLOAD: DATA


```
{
  "nbf": 1548812040,
  "exp": 1548815640,
  "iss": "http://localhost:5000",
  "aud": [
    "http://localhost:5000/resources",
    "api1"
  ],
  "client_id": "ro.client",
  "sub": "1",
  "auth_time": 1548812040,
  "idp": "local",
  "scope": [
    "api1"
  ],
  "amr": [
    "pwd"
  ]
}
```


VERIFY SIGNATURE

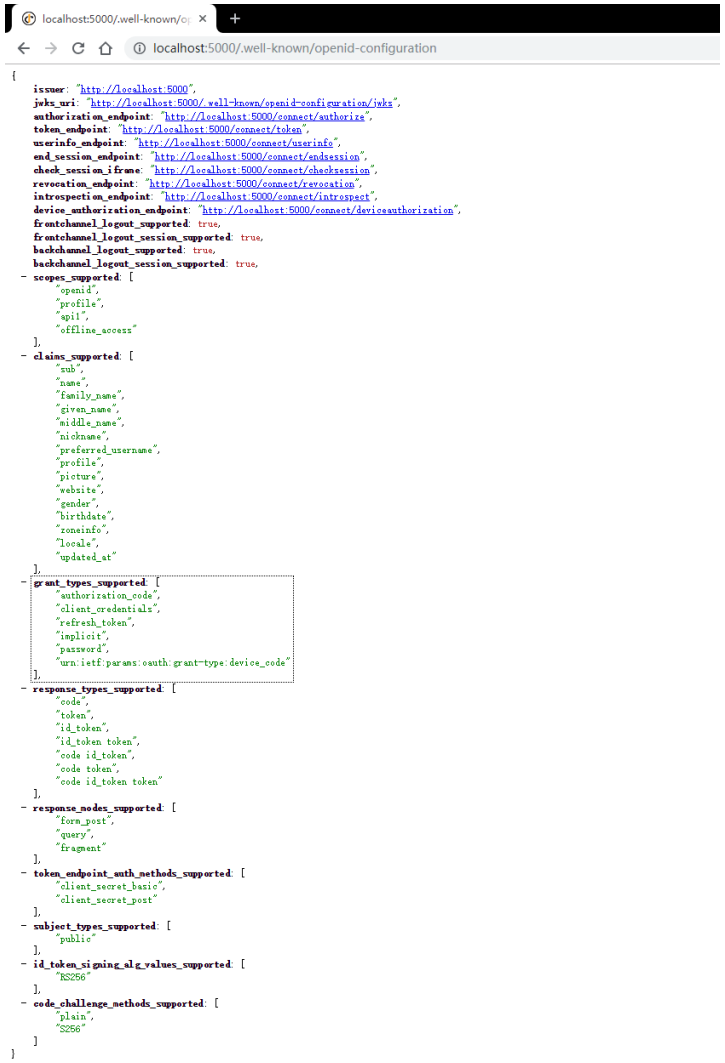
```
RSASHA256(
  base64UrlEncode(header) + "." +
```

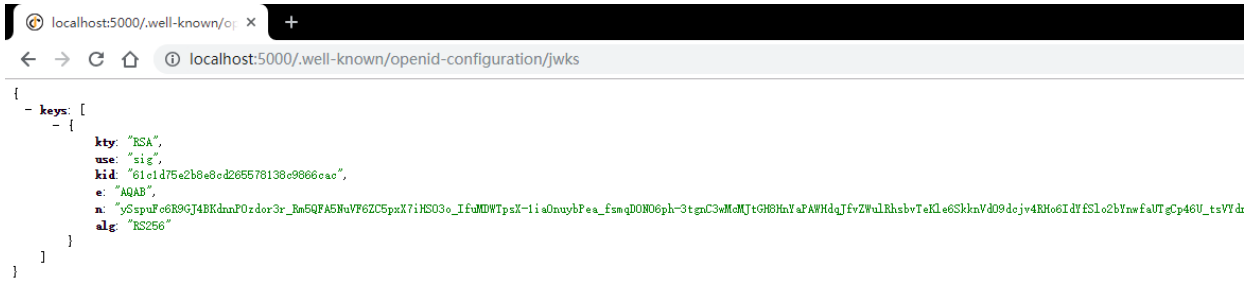
交互流程


开始
客户端发起请求
API资源
根据AccessToken所携带的Keyid来、在本地缓存进行查找是否存在
否
请求获取ids4配置信息
(/.well-known/openid-configuration)
(/.well-known/openid-configuration/jwks)
获取jws信息（keyid和公钥）
API资源缓存在本地，后续验证将不会继续请求
@晓晨Master
是
使用公钥验证Token是否被篡改
是
否
后续有效期、业务授权验证、执行业务返回结果等
结束


MVP

交互流程-详情


localhost:5000/.well-known/openid-configuration
{
issuer: "http://localhost:5000",
jwks_uri: "http://localhost:5000/.well-known/openid-configuration/jwks",
authorization_endpoint: "http://localhost:5000/connect/authorize",
token_endpoint: "http://localhost:5000/connect/token",
userinfo_endpoint: "http://localhost:5000/connect/userinfo",
end_session_endpoint: "http://localhost:5000/connect/endsession",
check_session_iframe: "http://localhost:5000/connect/checksession",
revocation_endpoint: "http://localhost:5000/connect/revocation",
introspection_endpoint: "http://localhost:5000/connect/introspect",
device_authorization_endpoint: "http://localhost:5000/connect/deviceauthorization",
frontchannel_logout_supported: true,
frontchannel_logout_session_supported: true,
backchannel_logout_supported: true,
backchannel_logout_session_supported: true,
scopes_supported: [
"openid",
"profile",
"api1",
"offline_access"
],
claims_supported: [
"sub",
"name",
"family_name",
"given_name",
"middle_name",
"nickname",
"preferred_username",
"profile",
"picture",
"website",
"gender",
"birthdate",
"zoneinfo",
"locale",
"updated_at"
],
grant_types_supported: [
"authorization_code",
"client_credentials",
"refresh_token",
"implicit",
"password",
"urn:ietf:params:oauth:grant-type:device_code"
],
response_types_supported: [
"code",
"token",
"id_token",
"id_token token",
"code id_token",
"code token",
"code id_token token"
],
response_modes_supported: [
"form_post",
"query",
"fragment"
],
token_endpoint_auth_methods_supported: [
"client_secret_basic",
"client_secret_post"
],
subject_types_supported: [
"public"
],
id_token_signing_alg_values_supported: [
"RS256"
],
code_challenge_methods_supported: [
"plain",
"S256"
]
}

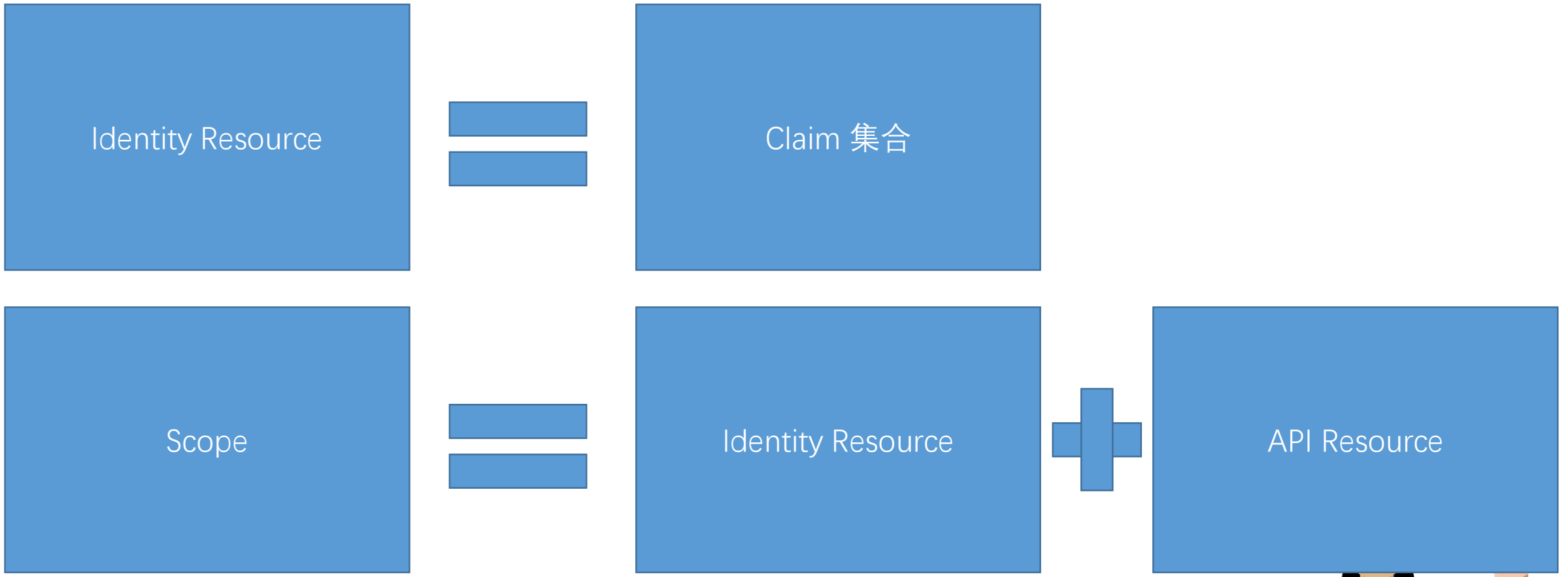

localhost:5000/.well-known/openid-configuration/jwks
{
keys: [
{
kty: "RSA",
use: "sig",
kid: "61c1d75e2b8e8cd265578138c9866cac",
e: "AQAB",
n: "ySspuFc6R9GJ4BKdnnFOzdor3r_Rm5QFA5NuVF6ZC5pxX7iHSO3o_IfuMDWTpsX-1iaOnuybPea_fsmqDONO6ph-3tgnC3wMcMJtGH8MnYaPAWHdqJfvZWulKhsbvTeKle6SkknVdO9dojv4KHo6IdYfSlo2bYnwfaUTgCp46U_tsVYdr
alg: "RS256"
}
]
}


MVP

Claim

用户身份信息单元，表示一个用户身份的某个信息，
采用键值对表现形式，一组Claim组成一个用户身份。

Claim & Resource & Scope



接入实现

1.客户端信息：IClientStore、ICorsPolicyService
2.用户验证：IResourceOwnerPasswordValidator
3.API资源和身份资源：IResourceStore
4.Claim 组装：IProfileService

接入实现-ResourceOwnerPassword


```
public class TestUserResourceOwnerPasswordValidator : IResourceOwnerPasswordValidator
{
    private readonly TestUserStore _users;
    private readonly ISystemClock _clock;

    /// <summary>
    /// Initializes a new instance of the <see cref="TestUserResourceOwnerPasswordValidator"/> class.
    /// </summary>
    /// <param name="users">The users.</param>
    /// <param name="clock">The clock.</param>
    public TestUserResourceOwnerPasswordValidator(TestUserStore users, ISystemClock clock)
    {
        _users = users;
        _clock = clock;
    }

    /// <summary>
    /// Validates the resource owner password credential
    /// </summary>
    /// <param name="context">The context.</param>
    /// <returns></returns>
    public Task ValidateAsync(ResourceOwnerPasswordValidationContext context)
    {
        if (_users.ValidateCredentials(context.UserName, context.Password))
        {
            var user = _users.FindByUsername(context.UserName);
            context.Result = new GrantValidationResult(
                user.SubjectId ?? throw new ArgumentException("Subject ID not set", nameof(user.SubjectId)),
                OidcConstants.AuthenticationMethods.Password, _clock.UtcNow.UtcDateTime,
                user.Claims);
        }

        return Task.CompletedTask;
    }
}
}
```

接入实现-ClientStore


```
/// <summary>
/// Implementation of IClientStore thats uses EF.
/// </summary>
/// <seealso cref="IdentityServer4.Stores.IClientStore" />
public class ClientStore : IClientStore
{
    private readonly IConfigurationDbContext _context;
    private readonly ILogger<ClientStore> _logger;

    /// <summary>
    /// Initializes a new instance of the <see cref="ClientStore"/> class.
    /// </summary>
    /// <param name="context">The context.</param>
    /// <param name="logger">The logger.</param>
    /// <exception cref="ArgumentNullException">context</exception>
    public ClientStore(IConfigurationDbContext context, ILogger<ClientStore> logger)
    {
        _context = context ?? throw new ArgumentNullException(nameof(context));
        _logger = logger;
    }

    /// <summary>
    /// Finds a client by id
    /// </summary>
    /// <param name="clientId">The client id</param>
    /// <returns>
    /// The client
    /// </returns>
    public Task<Client> FindClientByIdAsync(string clientId)
    {
        var client = _context.Clients
            .Include(x => x.AllowedGrantTypes)
            .Include(x => x.RedirectUris)
            .Include(x => x.PostLogoutRedirectUris)
            .Include(x => x.AllowedScopes)
            .Include(x => x.ClientSecrets)
            .Include(x => x.Claims)
            .Include(x => x.IdentityProviderRestrictions)
            .Include(x => x.AllowedCorsOrigins)
            .Include(x => x.Properties)
            .AsNoTracking()
            .FirstOrDefault(x => x.ClientId == clientId);
        var model = client?.ToModel();

        _logger.LogDebug("{clientId} found in database: {clientIdFound}", clientId, model != null);

        return Task.FromResult(model);
    }
}
```

接入实现-ResourceStore


```
/// <summary>
/// Finds the API resource by name.
/// </summary>
/// <param name="name">The name.</param>
/// <returns></returns>
public Task<ApiResource> FindApiResourceAsync(string name)
{
    var query =
        from apiResource in _context.ApiResources
        where apiResource.Name == name
        select apiResource;

    var apis = query
        .Include(x => x.Secrets)
        .Include(x => x.Scopes)
            .ThenInclude(s => s.UserClaims)
        .Include(x => x.UserClaims)
        .Include(x => x.Properties)
        .AsNoTracking();

    var api = apis.FirstOrDefault();

    if (api != null)
    {
        _logger.LogDebug("Found {api} API resource in database", name);
    }
    else
    {
        _logger.LogDebug("Did not find {api} API resource in database", name);
    }

    return Task.FromResult(api.ToModel());
}

/// <summary>
```

接入实现-ProfileService


```
public async Task GetProfileDataAsync(ProfileDataRequestContext context)
{
    context.LogProfileRequest(_logger);

    if (context.RequestedClaimTypes.Any())
    {
        var claims = await _manager.GetUserClaimsAsync(int.Parse(context.Subject.GetSubjectId()));
        context.AddRequestedClaims(context.FilterClaims(claims));
    }

    context.LogIssuedClaims(_logger);
}

public async Task IsActiveAsync(IsActiveContext context)
{
    _logger.LogDebug("IsActive called from: {caller}", context.Caller);

    var userId = int.Parse(context.Subject.GetSubjectId());
    var user = await _manager.FindUserAsync(userId);

    context.IsActive = !user.Lock;

    _logger.LogDebug($"IsActive called result: {context.IsActive}, UserId: {userId}.");
}
```

签名证书

Community

1.AddDeveloperSigningCredential


```
{
    "KeyId": "61c1d75e2b8e8cd265578138c9866cac",
    "Parameters":{
        "D": "xGLhOLc/CxuxWNFzlFuEGYwSZdunGhWvq2UDQVQOXpKAeazgTt8Mw3NEAUZchWQnOxMAI2Miz9RzP6hJ36q/+cGzoYMs4tZQIsh7zn08hvjOpKnQxaIPraIf78fJFbiiD2qfJjbOMQyGsl1/KgKUu7GYTMKzTc5qu44q8mUEMub8dolTux5+UMZiQwYin6j8gS39Y1ZrWMdMGmSnQnzp4l4Bshv8E45jXyCtfK+ux31FTWELyt/g/gewMGDqx1b+//YT2hZn3fnhiW3n6ryyfvErsarMhB8LltwN/E/lVFOw5pQDLBVinfsDOeKmnqgzd67Uedj04fGlvG99TMoPJQ==",
        "DP": "pxOS5JxTkyqhZQN9yeWBwJOlvCew2GUf7MxyDMXzGxuaDWPpLeHoJO+boXWUnmihl3bamb3Mf3AJABsswhNkuRAPF3nPTihPmdHAMDN7KIneZdty5bKxXOdE1FlrTv5WaclAK+GQxm97Zl3N66yRJYaf1fLoA9QoGO2l5PltM4c=",
        "DQ": "1z6WzaxZM+p8fa%ndF1HRtuK6FsAxfZbxo7JzlTSNP55Va2GE4I9MyzutDb4khik/U6+XBkANazs8Jz8R/ZUzOuaOnseY6rGwmk2MBhLaLvm9rB3VcO5tSgS1N7ZObKi7rLCN94lpImibhgiAvXhaBMD86HOO1NEgoRInUfxI+c=",
        "Exponent": "AQAB",
        "InverseQ": "ULZonMx297fnee7U7VVYbTniftKVIk5ngPWAI74llCysQp8sR/SuEqitXQZ6lIJzqOwHTnGWh9fW6+k3BsPvL/2VZlB4oM45aljf/V3x/YY+MMktESb194GMroVYgrEuJ/Yz79wklfnVFVeyRrtvvMD3qMRqUAMrT9Rm9xlPdY8=",
        "Modulus": "ySspuFc6R9GJ4BKdnnPOzdor3r/Rn5QFA5NuVF6ZC5pxX7iHSO3o/IfuMDWTpsX+1iaOnuybPea/fsmqDONO6ph+3tgnC3wMcMJtGMBHnYaPAWHdqJfvZWulRhsbvTeKle6SkknVd09dcjv4RNo6IdYfSlo2bYnwfaUTgCp46U/tsVYdnexFMelRYeWG2TF4Z9UUCDzK/4uOGntA3C5oe9gjh4ecEcDJISA9c9p3W6WifM/UJoVoG9PuCtbRw7/6HSNMjcqPaC8ZwTjh5/kVWKztIq2ulYKojsEi+wYA727cjEjVGEGjd+BZCjHhTwcvvYjFjDJteoWxEbM9ocEaVQ==",
        "P": "6CWzDLDUaKeVUI3/zSEiFH/MYWYzRNOF7tGHRBQrA3CAqM5/p3ulAbgcL3yOe7i/WnD1PY5DGgyxIigG+clLfnA7L49195BEh1OLpfG/YUh3y3i6fvVZ/dfg2AyiJ5NZKJn6hjtpybGPf1o+t7alY1Pu5UM7BHfAolX598z+Ovs=",
        "Q": "3dadSokHCMHWB2ycIXdBpF/3pPS5f8Xn56l2MxUiRXb+VHMjb4mcTSj4evTQuHxQjqB6LAQfq4rLNQYzCyh5mnpKMFN7zVwFSWUtFz5CMn2AMIiMiyr/h8+wkM8FLOkE2TWsHq2wgMKPYCr23qdGk7nzQ8p84QsO8Gcry6A5/u8="
    }
}
```


tempkey.rsa

2.AddSigningCredential(x509cert)

其他常见问题

1. 生产环境部署并非一定需要HTTPS。详情
2. 如果是用于分布式系统、使用了反向代理和部署多个副本则必须设置 Issuer。详情
3. AccessToken 过期时间默认具有5分钟偏移。详情

Community
Meetup
Chengdu

IdentityServer4 单点登录设计

选择授权模式

单点登录常用的授权模式主要有三种：

1. 授权码模式
2. 混合模式
3. 简单模式

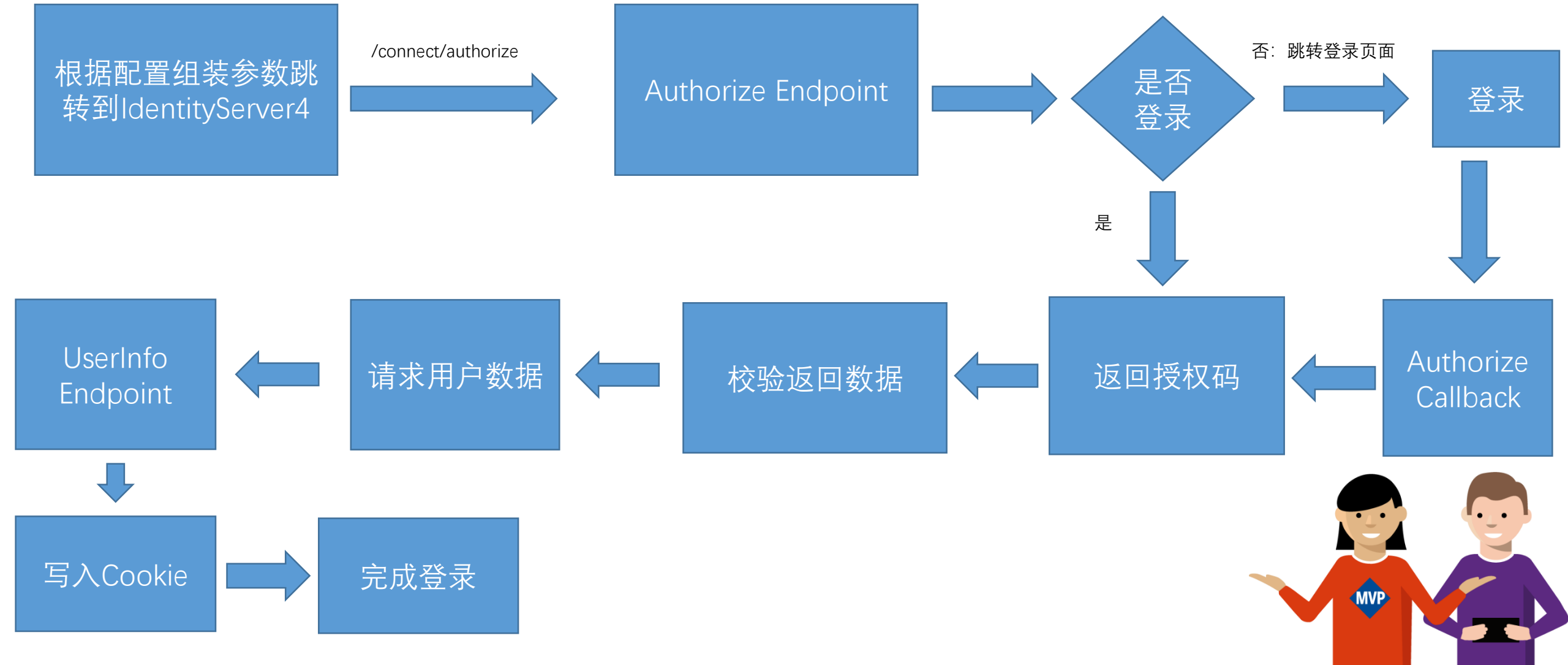
ASP.NET Core Web 应用接入

Community

```
new Client
{
    ClientId = "mvc",
    ClientName = "MVC Client",
    AllowedGrantTypes = GrantTypes.Code,

    // where to redirect to after login
    RedirectUris = { "http://localhost:5002/signin-oidc" },

    // where to redirect to after logout
    PostLogoutRedirectUris = { "http://localhost:5002/signout-callback-oidc" },

    AllowedScopes = new List<string>
    {
        IdentityServerConstants.StandardScopes.OpenId,
        IdentityServerConstants.StandardScopes.Profile
    }
}
```

```
services.AddAuthentication(configureOptions: options =>
    {
        options.DefaultScheme = "Cookies";
        options.DefaultChallengeScheme = "oidc";
    })
    .AddCookie("Cookies")
    .AddOpenIdConnect(authenticationScheme: "oidc", configureOptions: options =>
    {
        options.Authority = "http://localhost:5000";
        options.RequireHttpsMetadata = false;
        options.ClientId = "mvc";
        options.SaveTokens = true;
    });
```

IdentityServer4 授权码模式处理流程



老系统接入

ASP.NET MVC5 Owin 有现成的中间件可以直接使用。其他版本的 ASP.NET MVC以及 WebForm 需要自己实现或采用社区开源组件。自己实现推荐 IdentityModel.OidcClient

老系统接入-OidcClient实现


```
#region 登录

/// <summary>
/// 组装登录参数，返回跳转至SSO Url
/// </summary>
/// <returns></returns>
public static async Task<string> PrepareLoginAsync(string returnUrl=null)
{
    var authorizeState = await _client.PrepareLoginAsync();
    //登录状态10分钟过期
    _cache.Add(authorizeState.State, new LoginState(){AuthState = authorizeState,ReturnUrl = returnUrl}, DateTimeOffset.Now.AddMinutes(10));
    return authorizeState.StartUrl;
}


/// <summary>
/// 组装登录参数，返回跳转至SSO Url（能用异步请用异步方法）
/// </summary>
/// <param name="returnUrl"></param>
/// <returns></returns>
public static string PrepareLogin(string returnUrl = null)
{
    return AsyncHelper.RunSync(async () => await PrepareLoginAsync(returnUrl));
}



/// <summary>
/// 处理登录回调（必须是POST请求）
/// </summary>
/// <param name="request"></param>
/// <exception cref="SSOException"></exception>
/// <returns></returns>
public static async Task<SSOUser> ProcessLoginAsync(HttpRequestBase request)
{
    if (request.HttpMethod.ToLower() != "post")
    {
        throw new SSOException("HTTP request method is incorrect, must be POST.");
    }

    if (request.Form["state"] == null)
    {
        throw new SSOException("Query string is incorrect, lack of 'state' parameter.");
    }
    else
    {
        var state = request.Form["state"];
        var str = request.Form.ToString();
        var loginState = _cache.Get(state) as LoginState;
        if (loginState == null)
```

统一注销

IdentityServer4提供了两个客户端设置：

- FrontChannelLogoutUri
- BackChannelLogoutUri

分别对应前端注销以及后端注销，前端注销可通过隐藏iframe来批量注销，后端可通过请求相对性的api来进行通知注销

第三方系统以及其他语言接入

若第三方系统比如 Jira、gitlab 等系统直接支持（或插件）OAuth 2.0 或者 OpenId Connect，则可以直接配置IdentityServer地址接入。

其他语言系统比如Java、Go、PHP等，可以实现 OAuth 2.0 或者 OpenId Connect 后可接入

两步认证（双因认证）

1.直接使用 ASP.NET Core Identity
2.使用 TwoFactorAuth.Net

它们都使用了基于时间的一次性密码算法（TOTP），
可以接入支持TOTP的APP：Google Authenticator、
Microsoft Authenticator、2FAS Auth等等

权限设计

IdentityServer4 本身不提供权限，其默认的Scope校验属于Token的校验流程，不算做是权限。但结合IdentityServer4来做权限验证是可行的，可以使用Identity&Claim来做复杂的权限

最后

Q&A

联系我

博客园：https://www.cnblog.com/stulzq

THANK YOU

*