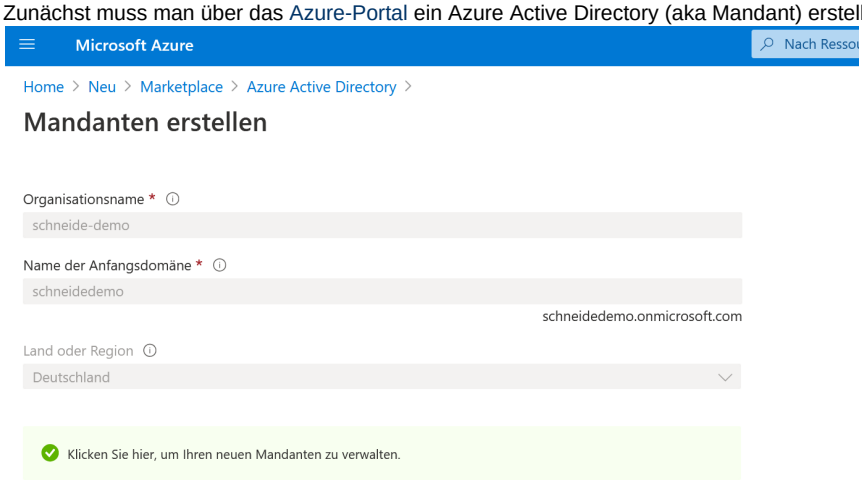
Entwicklerdokumentation

Damit man die Authentifizierung einer Webanwendung über Microsoft's Azure Active Directory abwickeln lassen kann, muss sowohl auf der Azure-Seite als auch anwendungsseitig einiges konfiguriert werden. Dabei soll dieses Dokument helfen.

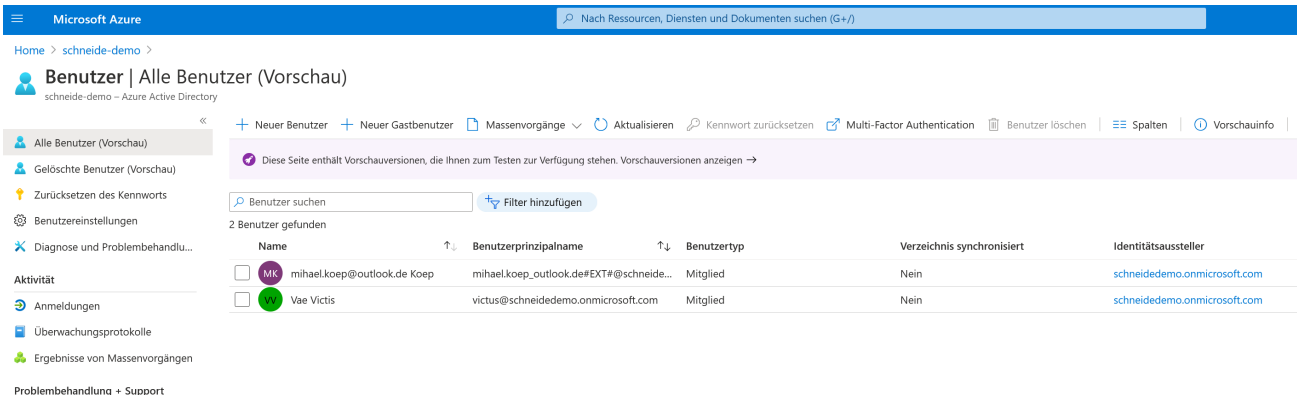
Azure Active Directory

Man benötigt (natürlich) einen Microsoft Account. Ein Microsoft Live/Outlook-Account funktioniert auch, so dass man auch als Entwickler gut entwickeln und testen kann, ohne Produktiv-Instanzen zu beeinflussen oder administrieren zu müssen.

1. Zunächst muss man über das Azure-Portal ein Azure Active Directory (aka Mandant) erstellen:

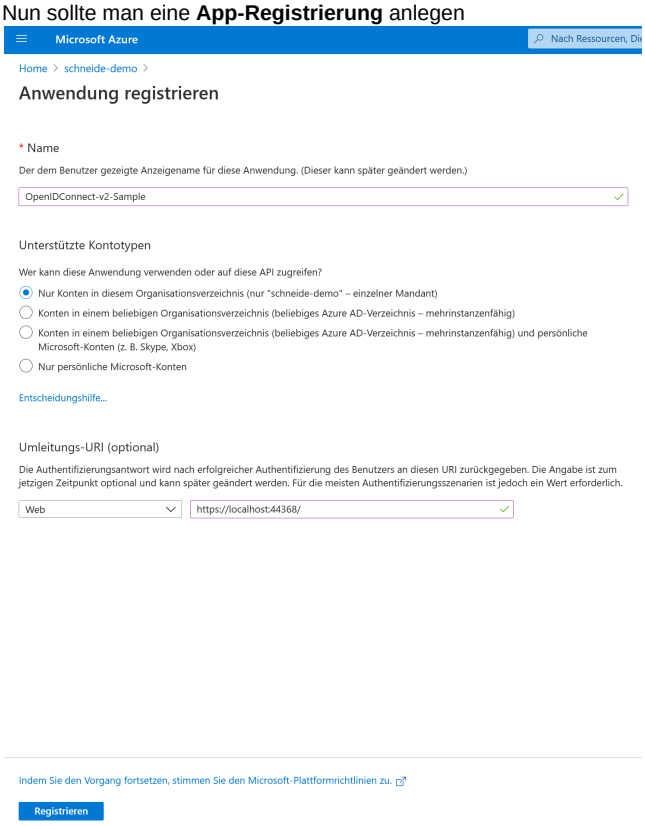


2. Dann zumindest einen Testnutzer erstellen:



Die Oberfläche ist an dieser Stelle recht intuitiv und spricht für sich selbst.

3. Nun sollte man eine **App-Registrierung** anlegen



Der Name kann frei gewählt werden, sollte aber leicht in Bezug mit der entsprechenden Anwendung gebracht werden können. In unseren Anwendungsfällen möchte man nur Konten aus dem entsprechenden Organisationsverzeichnis (aka Mandant/Tenant) zulassen. Die Umleitungs-URI ist dann die URL auf die der Nutzer nach erfolgreichem Login weitergeleitet wird und damit typischerweise die Adresse der eigenen Anwendung.

> **In der Entwicklung**
>
> In der Entwicklung ist die Umleitungs-URI normalerweise eine Adresse auf `localhost`. Die URL sollte das HTTPS-Protokoll unterstützen, verwendet man IIS Express muss der **Port mit 443 beginnen**!

Bei der App-Registrierung muss man sicherstellen, dass man den "Implicit Grant" ("Implizite Gewährung") für das ID-Token erlaubt.

Anbindung von AAD/SSO für eine normale ASP.NET Anwendung

Die Azure Active Directory Anbindung erfordert eine OWIN-Kompatible ASP.NET-Anwendung, da nur hier die Konfiguration bequem möglich ist. Diese Konfiguration erfolgt in der `Configuration()`-Methode `Startup`-Klasse mit den gewohnten Mitteln. Durch unsere AzureActiveDirectoryAuth-Bibliothek ist sehr wenig Code notwendig, um die Anbindung durchzuführen und die Authentifizierungs-Informationen abzufragen.

Authentifizierung

1. Erzeugung einer OWIN-Kompatiblen Startup-Klasse damit man die Authentifizierung konfigurieren kann. Dazu muss man OWIN bzw. Microsoft.Owin als Abhängigkeit hinzufügen.
2. Dann muss die ganze Authentifizierung konfiguriert werden. Dazu werden auch 4 Konfigurationswerte benötigt:

web.config

```
<appSettings>
  <add key="ClientId" value="355eb617-13e3-4cf8-8079-c5ebd89f7960" />
  <add key="Tenant" value="37e5d670-ede7-4eb8-a6e0-88985bf2ea1f" />
  <add key="Authority" value="https://login.microsoftonline.com/{0}
```

```
/v2.0" />
  <add key="redirectUri" value="https://localhost:44355/" />
</appSettings>
```

Die `ClientId` entnimmt man der **App-Registrierung** und den `Tenant` der Mandanten-ID des entsprechenden Azure Active-Directory. Beides kann man auch im Nachhinein im Azure-Portal einsehen.

3. Durch unsere AzureActiveDirectoryAuth-Bibliothek, die einfach in den Projektabhängigkeiten referenziert werden kann, ist der notwendige Konfigurations-Code minimal:

```
public class Startup
{
    public void Configuration(IAppBuilder app)
    {
        app.ConfigureOpenIdAuthentication(new OpenIdSettings
        {
            ClientId = System.Configuration.ConfigurationManager.
AppSettings["ClientId"],
            Tenant = System.Configuration.ConfigurationManager.
AppSettings["Tenant"],
            Authority = System.Configuration.ConfigurationManager.
AppSettings["Authority"],
            RedirectUri = System.Configuration.ConfigurationManager.
AppSettings["redirectUri"],
        });
    }
}
```

Verwendung in Controllern/ASP-Seiten

Nachdem die Anbindung an AAD erfolgt ist, kann man mit einigen wenigen Mitteln flexibel auf den Authentifizierungsstatus des Benutzers reagieren.

Abfrage des Authentifizierungsstatus

Überall, wo es sinnvoll/notwendig ist, zu prüfen, ob der Request von einem authentifizierten Nutzer kommt, kann man das direkt über ein Property am `Request`-Objekt abfragen:

```
if (Request.IsAuthenticated)
{
  ...
}
```

Wenn ein Benutzer authentifiziert ist, bekommt man seine, im AAD hinterlegten Informationen über seine Identität, die auch in Controllern und ASP-Seiten automatisch verfügbar ist:

```
var userClaims = new AadInformation(User.Identity as ClaimsIdentity);

ViewBag.Name = userClaims.Name;
ViewBag.Username = userClaims.Username;
ViewBag.TenantId = userClaims.TenantId;
ViewBag.Groups = string.Join(", ", userClaims.Groups);
```

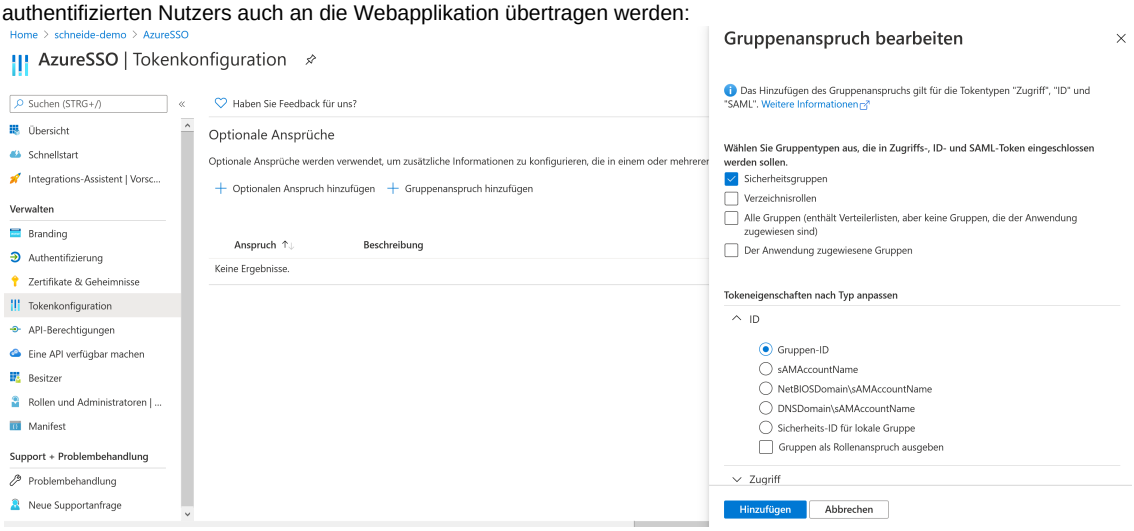
Authorisierung über AAD-Gruppen

Damit man ein einfaches Authorisierungskonzept umsetzen kann, ist ein gängiger Weg Benutzergruppen zu definieren, die unterschiedliche Berechtigungen haben. Auch hier sind Einstellungen im Azure Portal und der Webanwendung vorzunehmen:

Gruppen im Azure Portal erzeugen

Man navigiert zum entsprechenden Mandanten und wählt das "Blade" Gruppen. Dort können die Gruppen und ihre Mitglieder verwaltet werden. Gruppen sind dabei hierarchisch und können also auch Untergruppen beinhalten.

Bei der App-Registrierung in der Tokenkonfiguration muss man einen "Gruppenanspruch" (GroupClaim) hinzufügen, damit die Gruppen des authentifizierten Nutzers auch an die Webapplikation übertragen werden:



Die GUIDs der Gruppen werden dann in den User-Claims übertragen und können verwendet werden, um die Authorisierung zu implementieren:

groups	c7dbd564-32f6-42c0-94a6-64df6ee526e3
groups	5b135fdd-86ee-4549-a008-5f11918c9cbd
groups	74b69a88-e7da-4061-9533-7fb22b529d16
http://schemas.microsoft.com/identity/claims/identityprovider	https://sts.windows.net/9188040d-6c67-4c5b-b112-36a304b66dad/
name	mihael.koep@outlook.de Koep

Authorisierung in der Webanwendung über Gruppen realisieren

Die Bedeutung der Gruppen-GUIDs hinterlegt man in der Konfiguration der Anwendung:

```
<appSettings>
  <add key="GroupAdmin" value="74b69a88-e7da-4061-9533-7fb22b529d16" />
  <add key="GroupUser" value="5b135fdd-86ee-4549-a008-5f11918c9cbd" />
</appSettings>
```

Hat man nun einen eingeloggten Benutzer, kann man an den entsprechenden Stellen erfragen, ob er Mitglied der entsprechenden Gruppe ist:

```
if (!Request.IsAuthenticated) return View();

var user = new AadInformation(User.Identity as ClaimsIdentity);
// Gruppen des aktuellen Nutzer anzeigen
ViewBag.Groups = string.Join(", ", user.Groups);
if (user.IsMemberOf(GroupConfiguration.Admin))
{
    // Dinge, die nur Admins sehen dürfen definieren
}

return View();
```