PASSPORT-OUTLOOK

SEMINARIO Y TALLER DE SOFTWARE

Grupo #9

Genesis Barahona 1706 1999 00299
Erik Reyes 1702 1999 00156
Joshua Flores 1702 1999 00156
Juan Hernández 1702 1999 00156

2021

Passport-outlook

Estrategia de pasaporte para autenticarse con cuentas de Outlook (también conocido como Windows Live) usando la API de OAuth 2.0.

Este módulo le permite autenticarse usando Outlook REST API v2 en sus aplicaciones Node.js. Al conectarse a Passport, la autenticación de Outlook REST API v2 se puede integrar fácil y discretamente en cualquier aplicación o marco que admita middleware de estilo Connect , incluido Express .

A diferencia de los módulos alternativos, este paquete se autentica con los últimos puntos finales de Outlook.com (Office 365) v2, ya que se puede probar en su Outlook Dev Center OAuth Sandbox.

Instalar en pc

$ npm install --save passport-outlook

Uso

v3

No hay cambios de comportamiento, pero a partir de la v3, la versión mínima requerida de NodeJS es la v10. Esto no debería afectar a la mayoría de los usuarios, pero es un cambio radical.

Actualización para v2

Si estaba utilizando el paquete antes v2.0.0, tenga en cuenta que el profile JSON devuelto se ha actualizado para que coincida con el esquema de contacto normalizado descrito por Passport y utilizado por otras estrategias.

Por lo tanto, deberá actualizar su aplicación para que coincida con estas propiedades JSON modificadas.

Crear una aplicación

Antes de usar passport-outlook, debe registrar una aplicación con Microsoft. Si aún no lo ha hecho, se puede crear una nueva aplicación en el Portal de registro de aplicaciones . Su aplicación recibirá una identificación de cliente y un secreto de cliente, que deben proporcionarse a la estrategia. También deberá configurar una URL de redireccionamiento que coincida con la ruta en su aplicación.

Configurar estrategia

La estrategia de autenticación de Outlook REST API v2 autentica a los usuarios mediante una cuenta de Outlook.com y tokens de OAuth 2.0. La estrategia requiere una verify devolución de llamada, que acepta estas credenciales y llamadas que doneproporcionan un usuario, además de optionsespecificar un ID de cliente, un secreto de cliente y una URL de devolución de llamada.

```
passport.use(new OutlookStrategy({
   clientID: OUTLOOK_CLIENT_ID,
   clientSecret: OUTLOOK_CLIENT_SECRET,
   callbackURL: 'http://www.example.com/auth/outlook/callback'
 },
 function(accessToken, refreshToken, profile, done) {
   var user = {
     outlookId: profile.id,
     name: profile.DisplayName,
     email: profile.EmailAddress,
     accessToken:  accessToken
   };
   if (refreshToken)
     user.refreshToken = refreshToken;
   if (profile.MailboxGuid)
     user.mailboxGuid = profile.MailboxGuid;
   if (profile.Alias)
     user.alias = profile.Alias;
   User.findOrCreate(user, function (err, user) {
     return done(err, user);
   });
 }
));
```

Las opciones adicionales son compatibles como parte de la descrita flujo subvención implícita : prompt, login_hinty domain_hint.

Nota: Si desea utilizar la solicitud expresa, debe usar la opción passReqToCallback: true, luego Passport enviará la solicitud como primer parámetro.

```
passport.use(new OutlookStrategy({
   clientID: OUTLOOK_CLIENT_ID,
   clientSecret: OUTLOOK_CLIENT_SECRET,
   callbackURL: 'http://www.example.com/auth/outlook/callback',
   passReqToCallback: true
  },
  function(req, accessToken, refreshToken, profile, done) {
   var user = {
    outlookId: profile.id,
    name: profile.DisplayName,
    email: profile.EmailAddress,
    accessToken:  accessToken
   };
   if (refreshToken)
    user.refreshToken = refreshToken;
   if (profile.MailboxGuid)
    user.mailboxGuid = profile.MailboxGuid;
   if (profile.Alias)
    user.alias = profile.Alias;
   User.findOrCreate(user, function (err, user) {
    return done(err, user);
   });
```

```
  }
));
```

Autenticar solicitudes

Utilice passport.authenticate(), especificando la 'windowslive'estrategia (o con un nombre personalizado), para autenticar solicitudes.

Por ejemplo, como middleware de ruta en una aplicación Express :

```
app.get('/auth/outlook',
  passport.authenticate('windowslive', {
    scope: [
      'openid',
      'profile',
      'offline_access',
      'https://outlook.office.com/Mail.Read'
    ]
  })
);


app.get('/auth/outlook/callback',
  passport.authenticate('windowslive', { failureRedirect: '/login' }),
  function(req, res) {
    // Successful authentication, redirect home.
    res.redirect('/');
  });
```

Nota: Los alcances específicos de la API REST que está utilizando deben estar completamente calificados para coincidir con el dominio de Outlook, por ejemplo: en https://outlook.office.com/Mail.Readlugar de Mail.Read Esto es muy importante, de lo contrario, recibirá 401respuestas.

'offline_access'es un alcance necesario para obtener un refresh_token. Hay más información disponible en el Centro de desarrollo de MSDN .

Personalización de puntos finales

Si necesita personalizar las URL utilizadas por la estrategia (como conectarse a la API de Microsoft Graph en lugar de Office365 REST), esto es posible modificando el optionspaso a la estrategia:

```
passport.use(new OutlookStrategy({
   clientID: OUTLOOK_CLIENT_ID,
   clientSecret: OUTLOOK_CLIENT_SECRET,
   callbackURL: 'http://www.example.com/auth/graph/callback',
   userProfileURL: 'https://graph.microsoft.com/v1.0/me?$select=userPrincipalName',
   name: 'msgraph'
 },
 function(accessToken, refreshToken, profile, done) {
    // Callback logic as per examples
 }
));
```

En el ejemplo anterior, la estrategia namese cambia del valor predeterminado de 'windowslive' a 'msgraph'y userProfileURLse cambia al punto final correcto de la API de Microsoft Graph. Si realiza este cambio, recuerde utilizar los ámbitos adecuados para la API.

Ejemplos de

Para obtener un ejemplo completo y funcional, consulte el ejemplo de inicio de sesión .

Pruebas

Cualquier sistema puede ejecutar la suite de pruebas en desarrollo desde la terminal.

```
$ npm install
$ npm test
```

Contribuyendo

Pruebas

La suite de pruebas se encuentra en el test/directorio. Se espera que todas las características nuevas tengan los casos de prueba correspondientes. Asegúrese de que se apruebe el conjunto de pruebas completo ejecutando:

$ make test

Cobertura

Se espera que todas las funciones nuevas tengan cobertura de prueba. Los parches que aumentan la cobertura de la prueba se aceptan con gusto. Los informes de cobertura se pueden ver ejecutando:

$ make test-cov

$ make view-cov