Google OAuth2.0 Authentication — using Node JS and PassportJS

This module allows you to use Google to authenticate your Node.js applications. Google authentication may be simply and unobtrusively implemented into any application or framework that supports Connect-style middleware, such as **Express**, by linking into Passport.

## What is Passport

Passport is a Node.js authentication middleware. Passport is a highly versatile and modular web application framework that can be seamlessly integrated into any Express-based web application. Authentication with a username and password, as well as Facebook, Twitter, and other social media platforms, is supported by a diverse collection of mechanisms.

## What are strategies in Passport:

Authenticating requests is the responsibility of strategies, which they accomplish by implementing an authentication mechanism. Authentication techniques specify how to encode a credential in a request, such as a password or a claim from an identity provider (IdP). There is 500+ strategies in Passport. You can also read about other strategies at: <https://www.passportjs.org/packages/>

## What is middleware’s:

Middleware sits in the gap between an original request and the final desired path. Middleware features are frequently called in the order in which they are introduced to the stack. Middleware is commonly used to conduct URL-encoded activities, such as JSON request body parsing and cookie parsing for easier cookie management.

### Installation:

npm install passport-google-oauth20

## Initial Setup:

1. Create an empty directory with any name you want.
2. Open the folder with any code editor. I would like to prefer VS Code.
3. Now press ctrl + shift + ~ to open terminal.
4. Now type ***npm init -y*** and hit enter. It will create package.json file and install necessary node modules
5. Now type ***npm install express nodemon passport cors dotenv cookie-session passport-google-oauth20***

### Registering application

You must first register an application with Google before using passport-google-oauth20. A new project can be created via the [Google Developer Console](https://developers.google.com/) if you haven't previously. A client ID and client secret will be assigned to your application, which must be submitted to the strategy. You'll also need to set up a redirect URI that corresponds to the route in your app. [Read more about how to register app to google console.](https://developers.google.com/)

### Configuring Strategy

Users are authenticated using a Google account and OAuth 2.0 tokens in the Google authentication technique. When constructing a strategy, the client ID and secret obtained when building an application are provided as choices. A verify callback, which receives the access token and optional refresh token, as well as profile, which includes the authorized user's Google profile, are also required by the strategy. To complete authentication, the verify callback must contact cb supplying a user.

In your src directory, create file server.js and put below code in it.

const express = require("express");

const dotEnv = require("dotenv");

const cors = require("cors");

const app = express();

const passport = require("passport");

const cookieSession = require("cookie-session");

dotEnv.config(); //for reading data from env file

app.use(express.json()); //for fetching data from request body

app.use(cors()); //for cross origin requests

const port = process.env.PORT || 5000;

app.get("/", (req, res) => {

  console.log(res.send("api is running..."));

});

app.listen(port, () => {

  console.log(`port is running on ${port}`);

});

Now create .env file in root directory and paste following content in it

PORT = 5000

GOOGLE\_CLIENT\_ID = your client  id here

GOOGLE\_CLIENT\_SECRET = your client secret id here

### Directory Structure:

Graphical user interface, text, application

Description automatically generated

Now create another file passport.js and put following code.

### Passport.js:

/\*\*

 \* Configuring Strategy

 \*/

const passport = require("passport");

const GoogleStrategy = require("passport-google-oauth20").Strategy;

passport.serializeUser(function (user, done) {

  done(null, user);

});

passport.deserializeUser(function (user, done) {

  done(null, user);

});

passport.use(

  new GoogleStrategy(

    {

      clientID: process.env.GOOGLE\_CLIENT\_ID,

      clientSecret: process.env.GOOGLE\_CLIENT\_SECRET,

      callbackURL: "http://localhost:5000/auth/callback",

      passReqToCallback: true,

    },

    function (request, accessToken, refreshToken, profile, done) {

      //console.log(profile);

      return done(null, profile);

    }

  )

);

### Authenticating Requests:

//registering routes

//on Auth Failed, URL

app.get("/failed", (req, res) => {

  res.send("Not Authorized");

});

//on Success, URL

app.get("/success", (req, res) => {

  res.send(`Welcome ${req.user.\_json.email}`);

});

//requesting google route

app.get(

  "/auth/google",

  passport.authenticate("google", { scope: ["email", "profile"] })

);

//will act as a callback URL which will be called if Google Authentication is Successful.

app.get(

  "/auth/callback",

  passport.authenticate("google", {

    failureRedirect: "/failed",

  }),

  function (req, res) {

    res.redirect("/success");

  }

)

Now create file server.js, if you haven’t and put below code in it.

### Finalizing:

const express = require("express");

const dotEnv = require("dotenv");

const cors = require("cors");

const app = express();

const passport = require("passport");

const cookieSession = require("cookie-session");

dotEnv.config(); //for reading data from env file

app.use(express.json()); //for fetching data from request body

app.use(cors()); //for cross origin requests

const port = process.env.PORT || 5000;

//configuring passport strategy

require("./passport");

//registering middlewares

app.use(

  cookieSession({

    name: "google-auth-session",

    keys: ["key1", "key2"],

  })

);

app.use(passport.initialize());

app.use(passport.session());

//registering routes

//on Auth Failed, URL

app.get("/failed", (req, res) => {

  res.send("Not Authorized");

});

//on Success, URL

app.get("/success", (req, res) => {

  res.send(`Welcome ${req.user.\_json.email}`);

});

//requesting google route

app.get(

  "/auth/google",

  passport.authenticate("google", { scope: ["email", "profile"] })

);

//will act as a callback URL which will be called if Google Authentication is Successful.

app.get(

  "/auth/callback",

  passport.authenticate("google", {

    failureRedirect: "/failed",

  }),

  function (req, res) {

    res.redirect("/success");

  }

);

//setting up server

app.listen(port, () => {

  console.log(`server is up on ${port}`);

});

And inside package.json file add

Graphical user interface

Description automatically generated

Text

Description automatically generated



Graphical user interface, text, application, email

Description automatically generated

Congratulations 😊, we are now authenticated via google. I hope you really enjoyed this tutorial. Stick with us for such awesome tutorials.