# Authenticating with Passport.js

# Passport.js

## Login/authentication for your Express app

## Passport.js

- provides middleware that authenticates your routes
- use it to redirect users to login page
- can delegate login to 3rd party services

## Passport Strategies

- A "Strategy" is just a plugin for Passport

## Passport Strategies

- A "Strategy" is just a plugin for Passport

### Examples:

- Facebook
- Google
- Twitter
- Github
- Local Strategy (The Node.js app manages usernames/passwords)

## OAuth2

## OAuth2: an Authorization Framework

- OAuth2 is an open standard
- You log into one app (e.g., Github), which authorizes you for another app (your Node.js web app)

## Using Github for logging into your Node.js app

```
npm install passport
npm install passport-github
npm install express-session
npm install cookie-parser
```

# Using Github for logging into your Node.js app

```
npm install passport
npm install passport-github
npm install express-session
npm install cookie-parser
```

## Next steps

- Creating an app with Github
- Setting up Passport with your Github app

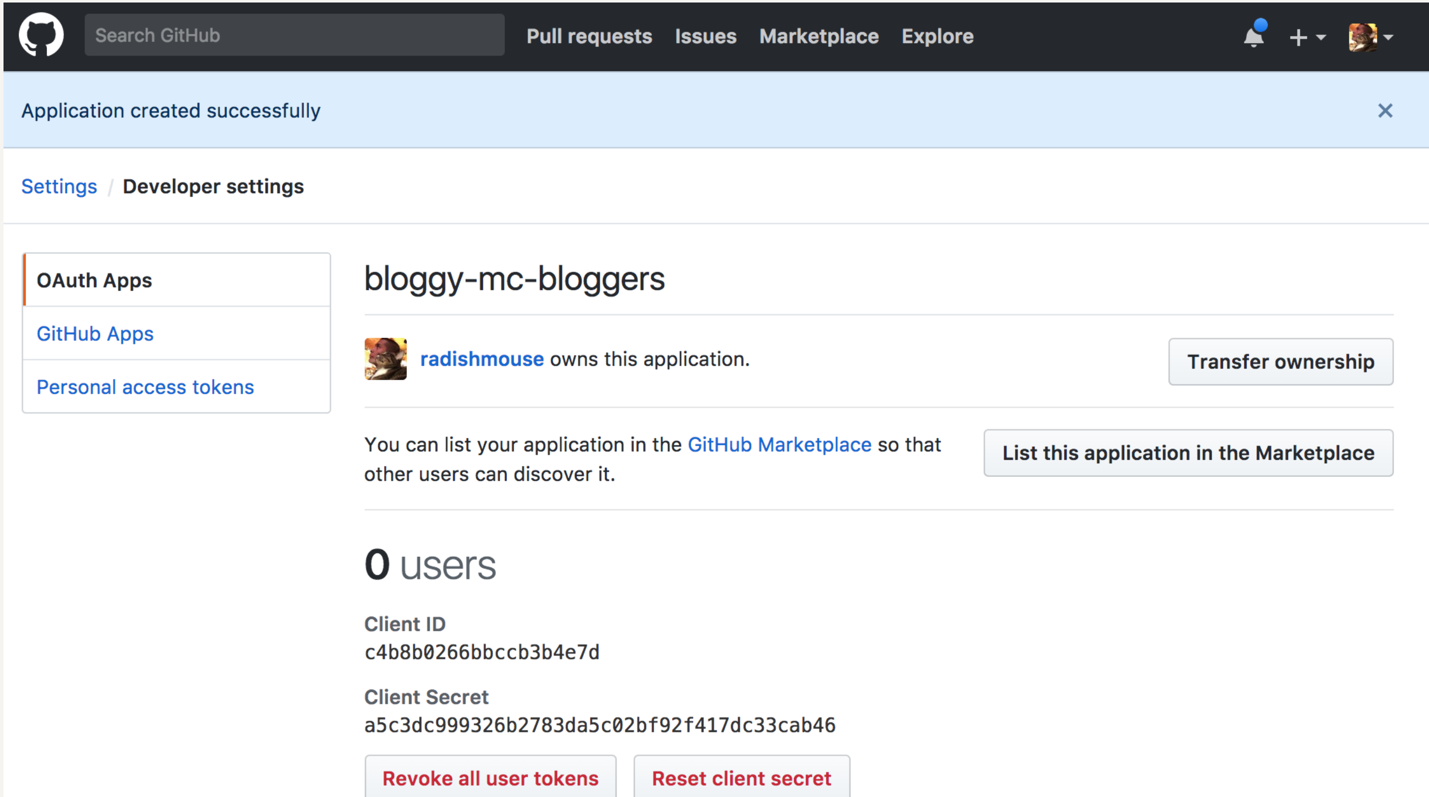
## Configure a new app in Github

## Configure a new app in Github

- https://github.com/settings/applications/new

## Configure an app in Github

## Configure an app in Github



Application created successfully

Settings / Developer settings

bloggy-mc-bloggers

radishmouse owns this application.

You can list your application in the GitHub Marketplace so that other users can discover it.

0 users

Client ID
c4b8b0266bbccb3b4e7d

Client Secret
a5c3dc999326b2783da5c02bf92f417dc33cab46

## Configuring Passport in your Node.js app

## Configure the Github Strategy (auth.js)

```
const passport = require('passport');
const GitHubStrategy = require('passport-github').Strategy;
const session = require('express-session');
const cookieParser = require('cookie-parser')


const setupAuth = (app) => {
  // We accept the express `app` as an arg.
  // It will be created somewhere else; we
  // are just adding middleware to it.
};

module.exports = setupAuth;
```

## Add the cookieParser middleware

```
const setupAuth = (app) => {
  // #1 set up cookie middleware
  app.use(cookieParser());

};
```

# Add the session middleware

```
const setupAuth = (app) => {
  // #1 set up cookie middleware
  app.use(cookieParser());

  // #2 set up session middleware
  app.use(session({
    secret: 'whatever',
    resave: true,
    saveUninitialized: true
  }));

};
```

# Configure the GithubStrategy

```
const setupAuth = (app) => {
  // #1 set up cookie middleware
  // #2 set up session middleware
  // #3 set up passport strategy
  passport.use(new GithubStrategy({
    clientID: "c4b8b0266bbccb3b4e7d",
    clientSecret: "a5c3dc999326b2783da5c02bf92f417dc33cab46",
    callbackURL: "http://localhost:3000/github/auth"
  }, (accessToken, refreshToken, profile, done) => {
    // Will be filled in later
  }));
};
```

## Tell passport how to keep track of a user by id

```
const setupAuth = (app) => {
  // #1 set up cookie middleware
  // #2 set up session middleware
  // #3 set up passport strategy

  // #4 call passport.serializeUser
  // This configures how passport turns a user object
  // into something it can track in a session.
  passport.serializeUser(function(user, done) {
    done(null, user.id);
  });

  // #5 call passport.serializeUser
  // This configures how passport checks what's in the
  // session to see if the login is still valid.
  passport.deserializeUser(function(id, done) {
    done(null, id);
  });
};
```

## Add the passport and passport session middleware

```
const setupAuth = (app) => {
  // #1 set up cookie middleware
  // #2 set up session middleware
  // #3 set up passport strategy
  // #4 call passport.serializeUser
  // #5 call passport.serializeUser

  // #6 initialize passport middleware and register it with express
  app.use(passport.initialize());

  // #7 start passport's session management middleware and
  // register it with express
  app.use(passport.session());
};
```

# Register our login, logout, and auth routes

```
const setupAuth = (app) => {
  // Steps #1-7

  // #8 register our login, logout, and auth routes
  app.get('/login', passport.authenticate('github'));
  app.get('/logout', function(req, res, next) {
    req.logout();
    res.redirect('/');
  });

  app.get('/github/auth',
    passport.authenticate('github', { failureRedirect: '/login' }),
    (req, res) => {
      res.redirect('/');
    }
  );
};
```

## Connect GithubStrategy to Sequelize User

```
const User = require('./models/user');
const setupAuth = (app) => {
  // Steps #1-2
  // #3 set up passport strategy
  passport.use(new GithubStrategy({
    // clientID, clientSecret, callbackURL
  }, (accessToken, refreshToken, profile, done) => {
    User.findOrCreate({where: {
      githubid: profile.id
    }}).then(result => {
      // `findOrCreate` returns an array
      // The actual user instance is the 0th element in the array
      return done(null, result[0]);
    })
    .catch(done)
  }));
  // Setps #4-8
};
```

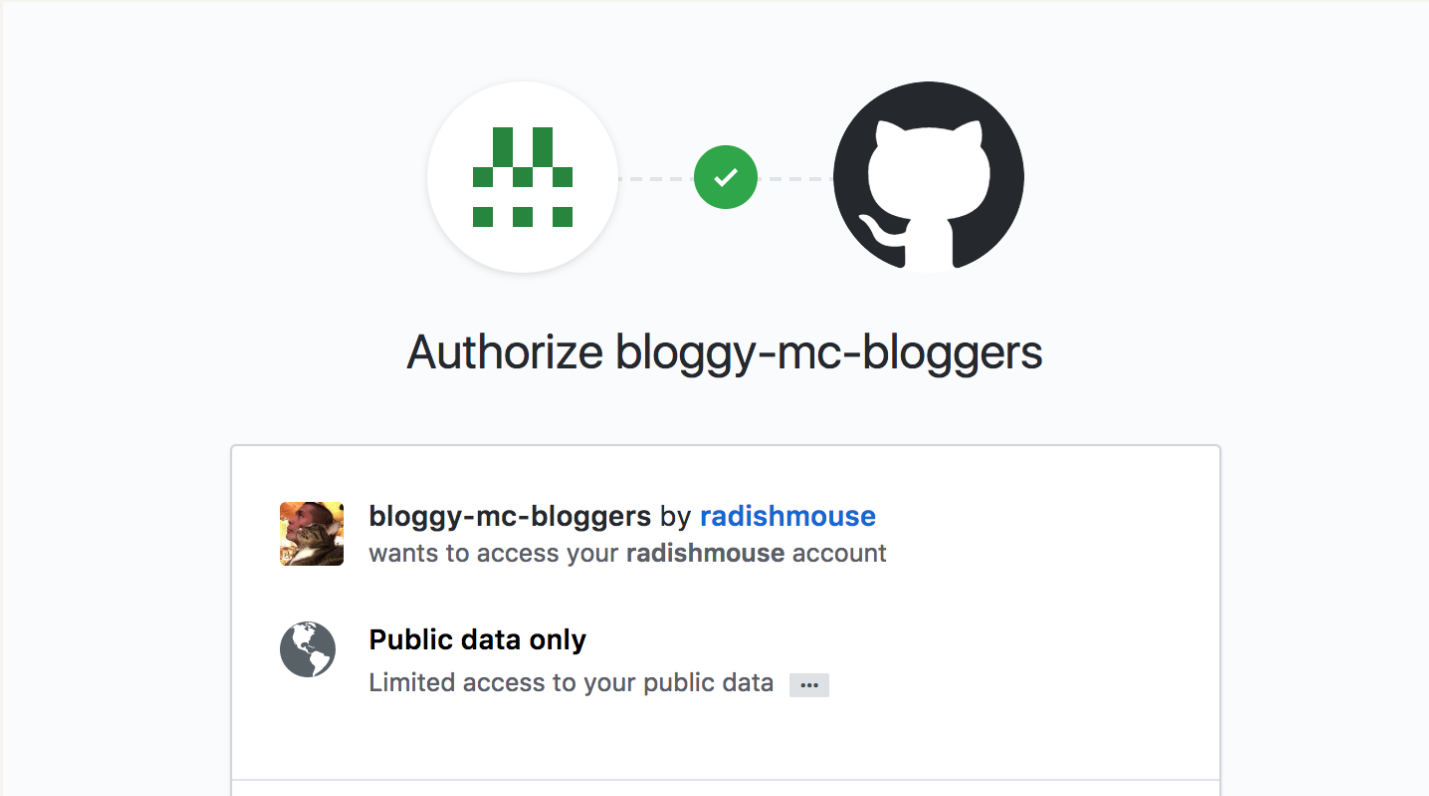
## Add a method for checking for a user

```
const setupAuth = (app) => {
  // Steps #1-8
};
const ensureAuthenticated = (req, res, next) => {
  if (req.isAuthenticated()) {
    return next();
  }
  // denied. redirect to login
  res.redirect('/login');
};
module.exports = setupAuth;
module.exports.ensureAuthenticated = ensureAuthenticated;
```

# Use our auth module (index.js)

```
const express = require('express');
const bodyParser = require('body-parser');
const expressHbs = require('express-handlebars');
const setupAuth = require('./auth');
const app = express();
app.use(bodyParser.urlencoded({ extended: false }));
app.engine('.hbs', expressHbs({defaultLayout: 'layout', extname: '.hbs'}));
app.set('view engine', '.hbs');
setupAuth(app);
```

## The /login route



Authorize bloggy-mc-bloggers

**bloggy-mc-bloggers** by **radishmouse**
wants to access your **radishmouse** account

**Public data only**
Limited access to your public data

## Protect a Route (routes/posts.js), Example 1

```
router.route('/blog/new')
  .get((req, res) => {
    res.render('blog-form');
  })
```

## Becomes

```
const ensureAuthenticated = require('../auth').ensureAuthenticated;
router.route('/blog/new')
  .get(ensureAuthenticated,
    (req, res) => {
      res.render('blog-form');
    })
```

## Protect a Route (routes/posts.js), Example 2

```
router.route('/blog')
  .post((req, res) => {
    Post.create({
      // ...
    }).then(newPost => {
      res.redirect(`/blog/${newPost.id}`);
    });
  })
```

## Becomes

```
const ensureAuthenticated = require('../auth').ensureAuthenticated;
router.route('/blog')
  .post(ensureAuthenticated,
    (req, res) => {
      Post.create({
        // ...
      }).then(newPost => {
        res.redirect(`/blog/${newPost.id}`);
      });
```

## Protecting a Router

```
const ensureAuthenticated = require('../auth').ensureAuthenticated;
router.all('*', ensureAuthenticated);

// All other routes stay the same
```

## Code available at:

- https://github.com/radishmouse/bloggy-mc-bloggers
- `radishmouse/passport` branch