Node.js: social login in ExpressJS using Twitter, Google, Facebook and Linkedin with Passport

In this article we're going to implement a social login system in Node.js with ExpressJS.

We'll use Passport as an authentication middleware with OAuth.

Passport inserts new methods and properties into the `request` object. `user` contains the user information taken from the authentication platform. This object is available on all routes.

Though Passport can handle OAuth callback URLs as relative, this will work only in development mode. In a production environment, callback URLs must be absolute and fully qualified in Passport.

Summary

1. Twitter
2. Google
3. Facebook
4. Linkedin

Twitter

We'll create an `.env` file with our app's OAuth credentials.

```
TWITTER_CONSUMER_KEY=your-consumer-key
TWITTER_CONSUMER_SECRET=your-consumer-secret
```

```
SESSION_SECRET=choose-a-random-string
```

We load this file at the very beginning.

```
'use strict';

require('dotenv').config();
```

Then we define our base app structure.

```
const path = require('path');
const express = require('express');
const passport = require('passport');
const { Strategy } = require('passport-twitter');
const { TWITTER_CONSUMER_KEY,
TWITTER_CONSUMER_SECRET, SESSION_SECRET } =
process.env;
const port = process.env.PORT || 3000;
const app = express();
const routes = require('./routes');
```

Now we can set up Passport.

```
passport.use(new Strategy({
    consumerKey: TWITTER_CONSUMER_KEY,
    consumerSecret: TWITTER_CONSUMER_SECRET,
    callbackURL: '/return'
  },
  (accessToken, refreshToken, profile, cb) => {
    return cb(null, profile);
}));

passport.serializeUser((user, cb) => {
```

```
  cb(null, user);
});

passport.deserializeUser((obj, cb) => {
  cb(null, obj);
});
```

`callbackURL` defines the OAuth callback URL. It must be absolute on a production environment.

Then we can initialize passport and bind it to the session.

```
app.use(require('express-session')({ secret:
SESSION_SECRET, resave: true, saveUninitialized: true
}));
app.use(passport.initialize());
app.use(passport.session());

app.use('/', routes);

app.listen(port);
```

We need to define the login, logout and callback routes. In all cases we're going to redirect back the user to the home page.

```
'use strict';

const express = require('express');
const passport = require('passport');
const router = express.Router();

router.get('/', (req, res, next) => {
    const { user } = req;
    res.render('home', { user });
```

```
});

router.get('/login/twitter',
passport.authenticate('twitter'));

router.get('/logout', (req, res, next) => {
  req.logout();
  res.redirect('/');
});

router.get('/return',
  passport.authenticate('twitter', { failureRedirect:
'/' }),
  (req, res, next) => {
    res.redirect('/');
});

module.exports = router;
```

The home page's view changes dynamically when a user is logged in.

```
<% if (!user) { %>
    <h1>Welcome!</h1>
    <p class="mt-5"><a href="/login/twitter"
class="loginBtn loginBtn--twitter">Login with
Twitter</a></p>
<% } else { %>
    <h1>Hello, <%= user.displayName %>.</h1>
    <p class="mt-5"><a href="/logout" class="btn btn-
primary">Logout</a></p>
<% } %>
```

Demo

Heroku

Source code

GitHub

Google

In the Google Developers Console you should specify the base domain and the callback URL of your app. The OAuth scope is `profile`.

We'll create an `.env` file with our app's OAuth credentials.

```
GOOGLE_CLIENT_ID=your-client-id
GOOGLE_CLIENT_SECRET=your-client-secret
SESSION_SECRET=choose-a-random-string
```

We load this file at the very beginning.

```
'use strict';

require('dotenv').config();
```

Then we define our base app structure.

```
const path = require('path');
const express = require('express');
const passport = require('passport');
const { Strategy } = require('passport-google-
oauth20');
const { GOOGLE_CLIENT_ID, GOOGLE_CLIENT_SECRET,
SESSION_SECRET } =  process.env;
const port = process.env.PORT || 3000;
```

```
const app = express();
const routes = require('./routes');
```

Now we can set up Passport.

```
passport.use(new Strategy({
    clientID: GOOGLE_CLIENT_ID,
    clientSecret: GOOGLE_CLIENT_SECRET,
    callbackURL: '/return'
  },
  (accessToken, refreshToken, profile, cb) => {
    return cb(null, profile);
}));

passport.serializeUser((user, cb) => {
  cb(null, user);
});

passport.deserializeUser((obj, cb) => {
  cb(null, obj);
});
```

`callbackURL` defines the OAuth callback URL. It must be absolute on a production environment.

Then we can initialize passport and bind it to the session.

```
app.use(require('express-session')({ secret:
SESSION_SECRET, resave: true, saveUninitialized: true
}));
app.use(passport.initialize());
app.use(passport.session());

app.use('/', routes);
```

```
app.listen(port);
```

We need to define the login, logout and callback routes. In all cases we're going to redirect back the user to the home page.

```
'use strict';

const express = require('express');
const passport = require('passport');
const router = express.Router();

router.get('/', (req, res, next) => {
    const { user } = req;
    res.render('home', { user });
});

router.get('/login/google',
passport.authenticate('google', { scope: ['profile']
}));

router.get('/logout', (req, res, next) => {
  req.logout();
  res.redirect('/');
});

router.get('/return',
  passport.authenticate('google', { failureRedirect:
'/' }),
  (req, res, next) => {
    res.redirect('/');
});
```

```
module.exports = router;
```

The home page's view changes dynamically when a user is logged in.

```
<% if (!user) { %>
    <h1>Welcome!</h1>
    <p class="mt-5"><a href="/login/google"
class="loginBtn loginBtn--google">Login with
Google</a></p>
<% } else { %>
    <h1>Hello, <%= user.displayName %>.</h1>
    <p class="mt-5"><a href="/logout" class="btn btn-
primary">Logout</a></p>
<% } %>
```

Demo

Heroku

Source code

GitHub

Facebook

In a production environment, the app must be reviewed and approved in order to work properly. Also, callback URLs must be absolute.

We'll create an `.env` file with our app's OAuth credentials.

```
FACEBOOK_CLIENT_ID=your-client-id
FACEBOOK_CLIENT_SECRET=your-client-secret
```

```
SESSION_SECRET=choose-a-random-string
```

We load this file at the very beginning.

```
'use strict';

require('dotenv').config();
```

Then we define our base app structure.

```
const path = require('path');
const express = require('express');
const passport = require('passport');
const { Strategy } = require('passport-facebook');
const { FACEBOOK_CLIENT_ID, FACEBOOK_CLIENT_SECRET,
SESSION_SECRET } =  process.env;
const port = process.env.PORT || 3000;
const app = express();
const routes = require('./routes');
```

Now we can set up Passport.

```
passport.use(new Strategy({
    clientID: FACEBOOK_CLIENT_ID,
    clientSecret: FACEBOOK_CLIENT_SECRET,
    callbackURL: '/return'
  },
  (accessToken, refreshToken, profile, cb) => {
    return cb(null, profile);
}));

passport.serializeUser((user, cb) => {
  cb(null, user);
```

```
});

passport.deserializeUser((obj, cb) => {
  cb(null, obj);
});
```

`callbackURL` defines the OAuth callback URL. It must be absolute on a production environment.

Then we can initialize passport and bind it to the session.

```
app.use(require('express-session')({ secret:
SESSION_SECRET, resave: true, saveUninitialized: true
}));
app.use(passport.initialize());
app.use(passport.session());

app.use('/', routes);

app.listen(port);
```

We need to define the login, logout and callback routes. In all cases we're going to redirect back the user to the home page.

```
'use strict';

const express = require('express');
const passport = require('passport');
const router = express.Router();

router.get('/', (req, res, next) => {
    const { user } = req;
    res.render('home', { user });
});
```

```
router.get('/login/facebook',
passport.authenticate('facebook'));

router.get('/logout', (req, res, next) => {
  req.logout();
  res.redirect('/');
});

router.get('/return',
  passport.authenticate('facebook', {
failureRedirect: '/' }),
  (req, res, next) => {
    res.redirect('/');
});

module.exports = router;
```

The home page's view changes dynamically when a user is logged in.

```
<% if (!user) { %>
    <h1>Welcome!</h1>
    <p class="mt-5"><a href="/login/facebook"
class="loginBtn loginBtn--facebook">Login with
Facebook</a></p>
<% } else { %>
    <h1>Hello, <%= user.displayName %>.</h1>
    <p class="mt-5"><a href="/logout" class="btn btn-
primary">Logout</a></p>
<% } %>
```

Source code

GitHub

Linkedin

In the Linkedin developers section you should specify the callback URL for your app. In Passport you also need to specify the OAuth scopes during setup.

We'll create an `.env` file with our app's OAuth credentials.

```
LINKEDIN_API_KEY=your-client-id
LINKEDIN_SECRET_KEY=your-client-secret
SESSION_SECRET=choose-a-random-string
```

We load this file at the very beginning.

```
'use strict';

require('dotenv').config();
```

Then we define our base app structure.

```
const path = require('path');
const express = require('express');
const passport = require('passport');
const { Strategy } = require('passport-linkedin-
oauth2');
const { LINKEDIN_API_KEY, LINKEDIN_SECRET_KEY,
SESSION_SECRET } =  process.env;
const port = process.env.PORT || 3000;
```

```
const app = express();
const routes = require('./routes');
```

Now we can set up Passport.

```
passport.use(new Strategy({
    clientID: LINKEDIN_API_KEY,
    clientSecret: LINKEDIN_SECRET_KEY,
    callbackURL: '/return',
    scope: ['r_emailaddress', 'r_liteprofile'],
    state: true
  },
  (accessToken, refreshToken, profile, cb) => {
    return cb(null, profile);
}));

passport.serializeUser((user, cb) => {
  cb(null, user);
});

passport.deserializeUser((obj, cb) => {
  cb(null, obj);
});
```

`callbackURL` defines the OAuth callback URL. It must be absolute on a production environment.

Then we can initialize passport and bind it to the session.

```
app.use(require('express-session')({ secret:
SESSION_SECRET, resave: true, saveUninitialized: true
}));
app.use(passport.initialize());
app.use(passport.session());
```

```
app.use('/', routes);

app.listen(port);
```

We need to define the login, logout and callback routes. In all cases we're going to redirect back the user to the home page.

```
'use strict';

const express = require('express');
const passport = require('passport');
const router = express.Router();

router.get('/', (req, res, next) => {
    const { user } = req;
    res.render('home', { user });
});

router.get('/login/linkedin',
passport.authenticate('linkedin'));

router.get('/logout', (req, res, next) => {
  req.logout();
  res.redirect('/');
});

router.get('/return',
  passport.authenticate('linkedin', {
failureRedirect: '/', successRedirect: '/' }),
  (req, res, next) => {
    res.redirect('/');
});
```

```
module.exports = router;
```

The home page's view changes dynamically when a user is logged in.

```
<% if (!user) { %>
    <h1>Welcome!</h1>
    <p class="mt-5"><a href="/login/linkedin"
class="loginBtn loginBtn--linkedin">Login with
Linkedin</a></p>
<% } else { %>
    <h1>Hello, <%= user.displayName %>.</h1>
    <p class="mt-5"><a href="/logout" class="btn btn-
primary">Logout</a></p>
<% } %>
```

Demo

Heroku

Source code

GitHub