PassportJS

INTRODUCTION

Passport is Express-compatible authentication middleware for Node.js.

Passport's sole purpose is to authenticate requests, which it does through an extensible set of plugins known as strategies. Passport does not mount routes or assume any particular database schema, which maximizes flexibility and allows application-level decisions to be made by the developer. The API is simple: you provide Passport a request to authenticate, and Passport provides hooks for controlling what occurs when authentication succeeds or fails.

How To Install

```
$ npm install passport
```

Usage

Strategies

Passport uses the concept of strategies to authenticate requests. Strategies can range from verifying username and password credentials, delegated authentication using OAuth (for example, via Facebook or Twitter), or federated authentication using OpenID.

Before authenticating requests, the strategy (or strategies) used by an application must be configured.

```
passport.use(new LocalStrategy(
  function(username, password, done) {
    User.findOne({ username: username }, function (err, user) {
      if (err) { return done(err); }
      if (!user) { return done(null, false); }
      if (!user.verifyPassword(password)) { return done(null, false); }
      return done(null, user);
    });
  }
));
```

There are 480+ strategies. Find the ones you want at: passportjs.org

Sessions

Passport will maintain persistent login sessions. In order for persistent sessions to work, the authenticated user must be serialized to the session, and deserialized when subsequent requests are made.

Passport does not impose any restrictions on how your user records are stored. Instead, you provide functions to Passport which implements the necessary serialization and deserialization logic. In a typical application, this will be as simple as serializing the user ID, and finding the user by ID when deserializing.

```
passport.serializeUser(function(user, done) {
  done(null, user.id);
});

passport.deserializeUser(function(id, done) {
  User.findById(id, function (err, user) {
    done(err, user);
  });
});
```

Middleware

To use Passport in an Express or Connect-based application, configure it with the required `passport.initialize()` middleware. If your application uses persistent login sessions (recommended, but not required), `passport.session()` middleware must also be used.

```
var app = express();
app.use(require('serve-static')(__dirname + '/../../public'));
app.use(require('cookie-parser')());
app.use(require('body-parser').urlencoded({ extended: true }));
app.use(require('express-session')({ secret: 'keyboard cat', resave:
true, saveUninitialized: true }));
app.use(passport.initialize());
app.use(passport.session());
```

Authenticate Requests

Passport provides an `authenticate()` function, which is used as route middleware to authenticate requests.

```
app.post('/login',
  passport.authenticate('local', { failureRedirect: '/login' }),
  function(req, res) {
    res.redirect('/');
  });
```

Strategies
Passport has a comprehensive set of over 480 authentication strategies covering social networking, enterprise integration, API services, and more.

Search all strategies
There is a Strategy Search at passportjs.org

Name: PassportJS Authentication Library For NodeJs

Description: Passport's sole purpose is to authenticate requests, which it does through an extensible set of plugins known as strategies. Passport does not mount routes or assume any particular database schema, which maximizes flexibility and allows application-level decisions to be made by the developer. The API is simple: you provide Passport a request to authenticate, and Passport provides hooks for controlling what occurs when authentication succeeds or fails.

Library Language - Javascript
Github Link - https://github.com/jaredhanson/passport
Author - Jarred Hanson
Version - 0.6.0