Next.js + Passport.js Example

Author: Vercel

Version: v1 (original)

Source:
https://github.com/vercel/next.js/tree/canary/examples/with-passport

Compiled on: 2022-09-22 14:22

Directory:

```
.
├── README.md
├── components
│   ├── layout.js
│   ├── header.js
│   └── form.js
├── .gitignore
├── package.json
├── .env
├── lib
│   ├── hooks.js
│   ├── user.js
│   ├── auth.js
│   ├── auth-cookies.js
│   └── password-local.js
└── pages
    ├── profile.js
    ├── index.js
    ├── signup.js
    ├── login.js
    └── api
        ├── user.js
        ├── logout.js
        ├── signup.js
        └── login.js
```

Passport.js Example

This example show how to use Passport.js with Next.js. The example features cookie based authentication with username and password.

The example shows how to do a login, signup and logout; and to get the user info using a hook with SWR.

A database is not included. You can use any database you want and add it in this file.

The login cookie is httpOnly, meaning it can only be accessed by the API, and it's encrypted using @hapi/iron for more security.

Deploy your own

Deploy the example using Vercel:

How to use

Execute create-next-app with npm, Yarn, or pnpm to bootstrap the example:

bash npx create-next-app --example with-passport with-passport-app

bash yarn create next-app --example with-passport with-passport-app

bash pnpm create next-app --example with-passport with-passport-app

Deploy it to the cloud with Vercel (Documentation).

```
# Secrets like the one below should go into `.env.local` instead to avoid pushing
# them to a repository, this is an exception for the sake of the example
TOKEN_SECRET="this-is-a-secret-value-with-at-least-32-characters"
```

```
import Head from 'next/head'
import Header from './header'

const Layout = (props) => (
  <>
    <Head>
      <title>With Cookies</title>
    </Head>

    <Header />

    <main>
      <div className="container">{props.children}</div>
    </main>

    <style jsx global>{`
      *,
      *::before,
      *::after {
        box-sizing: border-box;
      }
      body {
        margin: 0;
        color: #333;
        font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto,
          'Helvetica Neue', Arial, Noto Sans, sans-serif, 'Apple Color Emoji',
          'Segoe UI Emoji', 'Segoe UI Symbol', 'Noto Color Emoji';
      }
      .container {
        max-width: 42rem;
        margin: 0 auto;
        padding: 2rem 1.25rem;
      }
    `}</style>
  </>
)

export default Layout
```

```
import Link from 'next/link'
import { useUser } from '../lib/hooks'

const Header = () => {
  const user = useUser()

  return (
    <header>
      <nav>
        <ul>
          <li>
            <Link href="/">
              <a>Home</a>
            </Link>
          </li>
          {user ? (
            <>
              <li>
                <Link href="/profile">
                  <a>Profile</a>
                </Link>
              </li>
              <li>
                <a href="/api/logout">Logout</a>
              </li>
            </>
          ) : (
            <li>
              <Link href="/login">
                <a>Login</a>
              </Link>
            </li>
          )}
        </ul>
      </nav>
      <style jsx>{`
        nav {
          max-width: 42rem;
          margin: 0 auto;
          padding: 0.2rem 1.25rem;
        }
        ul {
          display: flex;
          list-style: none;
          margin-left: 0;
          padding-left: 0;
        }
        li {
          margin-right: 1rem;
        }
        li:first-child {
          margin-left: auto;
        }
        a {
          color: #fff;
          text-decoration: none;
        }
        header {
          color: #fff;
          background-color: #333;
        }
      `}</style>
    </header>
  )
}

export default Header
```

```
import Link from 'next/link'

const Form = ({ isLogin, errorMessage, onSubmit }) => (
  <form onSubmit={onSubmit}>
    <label>
      <span>Username</span>
      <input type="text" name="username" required />
    </label>
    <label>
      <span>Password</span>
      <input type="password" name="password" required />
    </label>
    {!isLogin && (
      <label>
        <span>Repeat password</span>
        <input type="password" name="rpassword" required />
      </label>
    )}

    <div className="submit">
      {isLogin ? (
        <>
          <Link href="/signup">
            <a>I don't have an account</a>
          </Link>
          <button type="submit">Login</button>
        </>
      ) : (
        <>
          <Link href="/login">
            <a>I already have an account</a>
          </Link>
          <button type="submit">Signup</button>
        </>
      )}
    </div>

    {errorMessage && <p className="error">{errorMessage}</p>}

    <style jsx>{`
      form,
      label {
        display: flex;
        flex-flow: column;
      }
      label > span {
        font-weight: 600;
      }
      input {
        padding: 8px;
        margin: 0.3rem 0 1rem;
        border: 1px solid #ccc;
        border-radius: 4px;
      }
      .submit {
        display: flex;
        justify-content: flex-end;
        align-items: center;
        justify-content: space-between;
      }
      .submit > a {
        text-decoration: none;
      }
      .submit > button {
        padding: 0.5rem 1rem;
        cursor: pointer;
        background: #fff;
        border: 1px solid #ccc;
        border-radius: 4px;
      }
      .submit > button:hover {
        border-color: #888;
      }
      .error {
        color: brown;
        margin: 1rem 0 0;
      }
    `}</style>
  </form>
)

export default Form
```

```
import { useEffect } from 'react'
import Router from 'next/router'
import useSWR from 'swr'

const fetcher = (url) =>
  fetch(url)
    .then((r) => r.json())
    .then((data) => {
      return { user: data?.user || null }
    })

export function useUser({ redirectTo, redirectIfFound } = {}) {
  const { data, error } = useSWR('/api/user', fetcher)
  const user = data?.user
  const finished = Boolean(data)
  const hasUser = Boolean(user)

  useEffect(() => {
    if (!redirectTo || !finished) return
    if (
      // If redirectTo is set, redirect if the user was not found.
      (redirectTo && !redirectIfFound && !hasUser) ||
      // If redirectIfFound is also set, redirect if the user was found
      (redirectIfFound && hasUser)
    ) {
      Router.push(redirectTo)
    }
  }, [redirectTo, redirectIfFound, finished, hasUser])

  return error ? null : user
}
```

```
import crypto from 'crypto'
import { v4 as uuidv4 } from 'uuid'

/**
 * User methods. The example doesn't contain a DB, but for real applications you must use a
 * db here, such as MongoDB, Fauna, SQL, etc.
 */

const users = []

export async function createUser({ username, password }) {
  // Here you should create the user and save the salt and hashed password (some dbs may have
  // authentication methods that will do it for you so you don't have to worry about it):
  const salt = crypto.randomBytes(16).toString('hex')
  const hash = crypto
    .pbkdf2Sync(password, salt, 1000, 64, 'sha512')
    .toString('hex')
  const user = {
    id: uuidv4(),
    createdAt: Date.now(),
    username,
    hash,
    salt,
  }

  // This is an in memory store for users, there is no data persistence without a proper DB
  users.push(user)

  return { username, createdAt: Date.now() }
}

// Here you should lookup for the user in your DB
export async function findUser({ username }) {
  // This is an in memory store for users, there is no data persistence without a proper DB
  return users.find((user) => user.username === username)
}

// Compare the password of an already fetched user (using `findUser`) and compare the
// password for a potential match
export function validatePassword(user, inputPassword) {
  const inputHash = crypto
    .pbkdf2Sync(inputPassword, user.salt, 1000, 64, 'sha512')
    .toString('hex')
  const passwordsMatch = user.hash === inputHash
  return passwordsMatch
}
```

```javascript
import Iron from '@hapi/iron'
import { MAX_AGE, setTokenCookie, getTokenCookie } from './auth-cookies'

const TOKEN_SECRET = process.env.TOKEN_SECRET

export async function setLoginSession(res, session) {
  const createdAt = Date.now()
  // Create a session object with a max age that we can validate later
  const obj = { ...session, createdAt, maxAge: MAX_AGE }
  const token = await Iron.seal(obj, TOKEN_SECRET, Iron.defaults)

  setTokenCookie(res, token)
}

export async function getLoginSession(req) {
  const token = getTokenCookie(req)

  if (!token) return

  const session = await Iron.unseal(token, TOKEN_SECRET, Iron.defaults)
  const expiresAt = session.createdAt + session.maxAge * 1000

  // Validate the expiration date of the session
  if (Date.now() > expiresAt) {
    throw new Error('Session expired')
  }

  return session
}
```

```
import { serialize, parse } from 'cookie'

const TOKEN_NAME = 'token'

export const MAX_AGE = 60 * 60 * 8 // 8 hours

export function setTokenCookie(res, token) {
  const cookie = serialize(TOKEN_NAME, token, {
    maxAge: MAX_AGE,
    expires: new Date(Date.now() + MAX_AGE * 1000),
    httpOnly: true,
    secure: process.env.NODE_ENV === 'production',
    path: '/',
    sameSite: 'lax',
  })

  res.setHeader('Set-Cookie', cookie)
}

export function removeTokenCookie(res) {
  const cookie = serialize(TOKEN_NAME, '', {
    maxAge: -1,
    path: '/',
  })

  res.setHeader('Set-Cookie', cookie)
}

export function parseCookies(req) {
  // For API Routes we don't need to parse the cookies.
  if (req.cookies) return req.cookies

  // For pages we do need to parse the cookies.
  const cookie = req.headers?.cookie
  return parse(cookie || '')
}

export function getTokenCookie(req) {
  const cookies = parseCookies(req)
  return cookies[TOKEN_NAME]
}
```

```
import Local from 'passport-local'
import { findUser, validatePassword } from './user'

export const localStrategy = new Local.Strategy(function (
  username,
  password,
  done
) {
  findUser({ username })
    .then((user) => {
      if (user && validatePassword(user, password)) {
        done(null, user)
      } else {
        done(new Error('Invalid username and password combination'))
      }
    })
    .catch((error) => {
      done(error)
    })
})
```

```
import { useUser } from '../lib/hooks'
import Layout from '../components/layout'

const Profile = () => {
  const user = useUser({ redirectTo: '/login' })

  return (
    <Layout>
      <h1>Profile</h1>
      {user && (
        <>
          <p>Your session:</p>
          <pre>{JSON.stringify(user, null, 2)}</pre>
        </>
      )}

      <style jsx>{`
        pre {
          white-space: pre-wrap;
          word-wrap: break-word;
        }
      `}</style>
    </Layout>
  )
}

export default Profile
```

```
import { useUser } from '../lib/hooks'
import Layout from '../components/layout'

const Home = () => {
  const user = useUser()

  return (
    <Layout>
      <h1>Passport.js Example</h1>

      <p>Steps to test the example:</p>

      <ol>
        <li>Click Login and enter a username and password.</li>
        <li>
          You'll be redirected to Home. Click on Profile, notice how your
          session is being used through a token stored in a cookie.
        </li>
        <li>
          Click Logout and try to go to Profile again. You'll get redirected to
          Login.
        </li>
      </ol>

      {user && (
        <>
          <p>Currently logged in as:</p>
          <pre>{JSON.stringify(user, null, 2)}</pre>
        </>
      )}

      <style jsx>{`
        li {
          margin-bottom: 0.5rem;
        }
        pre {
          white-space: pre-wrap;
          word-wrap: break-word;
        }
      `}</style>
    </Layout>
  )
}

export default Home
```

```js
import { useState } from 'react'
import Router from 'next/router'
import { useUser } from '../lib/hooks'
import Layout from '../components/layout'
import Form from '../components/form'

const Signup = () => {
  useUser({ redirectTo: '/', redirectIfFound: true })

  const [errorMsg, setErrorMsg] = useState('')

  async function handleSubmit(e) {
    e.preventDefault()

    if (errorMsg) setErrorMsg('')

    const body = {
      username: e.currentTarget.username.value,
      password: e.currentTarget.password.value,
    }

    if (body.password !== e.currentTarget.rpassword.value) {
      setErrorMsg(`The passwords don't match`)
      return
    }

    try {
      const res = await fetch('/api/signup', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify(body),
      })
      if (res.status === 200) {
        Router.push('/login')
      } else {
        throw new Error(await res.text())
      }
    } catch (error) {
      console.error('An unexpected error happened occurred:', error)
      setErrorMsg(error.message)
    }
  }

  return (
    <Layout>
      <div className="login">
        <Form isLogin={false} errorMessage={errorMsg} onSubmit={handleSubmit} />
      </div>
      <style jsx>{`
        .login {
          max-width: 21rem;
          margin: 0 auto;
          padding: 1rem;
          border: 1px solid #ccc;
          border-radius: 4px;
        }
      `}</style>
    </Layout>
  )
}

```

```
export default Signup
```

```
import { useState } from 'react'
import Router from 'next/router'
import { useUser } from '../lib/hooks'
import Layout from '../components/layout'
import Form from '../components/form'

const Login = () => {
  useUser({ redirectTo: '/', redirectIfFound: true })

  const [errorMsg, setErrorMsg] = useState('')

  async function handleSubmit(e) {
    e.preventDefault()

    if (errorMsg) setErrorMsg('')

    const body = {
      username: e.currentTarget.username.value,
      password: e.currentTarget.password.value,
    }

    try {
      const res = await fetch('/api/login', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify(body),
      })
      if (res.status === 200) {
        Router.push('/')
      } else {
        throw new Error(await res.text())
      }
    } catch (error) {
      console.error('An unexpected error happened occurred:', error)
      setErrorMsg(error.message)
    }
  }

  return (
    <Layout>
      <div className="login">
        <Form isLogin errorMessage={errorMsg} onSubmit={handleSubmit} />
      </div>
      <style jsx>{`
        .login {
          max-width: 21rem;
          margin: 0 auto;
          padding: 1rem;
          border: 1px solid #ccc;
          border-radius: 4px;
        }
      `}</style>
    </Layout>
  )
}

export default Login
```

```
import { getLoginSession } from '../../lib/auth'
import { findUser } from '../../lib/user'

export default async function user(req, res) {
  try {
    const session = await getLoginSession(req)
    const user = (session && (await findUser(session))) ?? null

    res.status(200).json({ user })
  } catch (error) {
    console.error(error)
    res.status(500).end('Authentication token is invalid, please log in')
  }
}
```

```
import { removeTokenCookie } from '../../lib/auth-cookies'

export default async function logout(req, res) {
  removeTokenCookie(res)
  res.writeHead(302, { Location: '/' })
  res.end()
}
```

```
import { createUser } from '../../lib/user'

export default async function signup(req, res) {
  try {
    await createUser(req.body)
    res.status(200).send({ done: true })
  } catch (error) {
    console.error(error)
    res.status(500).end(error.message)
  }
}
```

```js
import passport from 'passport'
import nextConnect from 'next-connect'
import { localStrategy } from '../../lib/password-local'
import { setLoginSession } from '../../lib/auth'

const authenticate = (method, req, res) =>
  new Promise((resolve, reject) => {
    passport.authenticate(method, { session: false }, (error, token) => {
      if (error) {
        reject(error)
      } else {
        resolve(token)
      }
    })(req, res)
  })

passport.use(localStrategy)

export default nextConnect()
  .use(passport.initialize())
  .post(async (req, res) => {
    try {
      const user = await authenticate('local', req, res)
      // session is the payload to save in the token, it may contain basic info about the user
      const session = { ...user }

      await setLoginSession(res, session)

      res.status(200).send({ done: true })
    } catch (error) {
      console.error(error)
      res.status(401).send(error.message)
    }
  })
```