

Sessions



Full Stack Web Development

Sessions

- HTTP is described as a stateless protocol - every new request is just as anonymous as the last.
- This sounds very unhelpful for a protocol that powers websites, where users expect to be remembered as they go to page to page.
- Cookies to the Rescue:
 - A request comes to a web application with a cookie
 - using the cookie the server can look up information about the user, either from the cookie itself or from server-side storage.
 - It can then forget all about them for a while, until the next request and the same process continues over for every request.

@hapi/cookie

- A Hapi Plugin to manage cookie access and management.
- Must be downloaded, installed and registered (like all plugins)

hapijs / cookie

Sponsor

Watch 41

Star 233

Fork 95

Code

Issues 1

Pull requests 0

Actions

Projects 0

Security

Insights

Cookie authentication plugin

227 commits

1 branch

0 packages

37 releases

36 contributors

View license

Branch: master

New pull request

Create new file

Upload files

Find file

Clone or download

hueniverse

Fix test on hapi 18

Latest commit eff4a3f 18 days ago


example	Update deps. Closes #213	9 months ago
lib	Update deps. Closes #227	18 days ago
test	Fix test on hapi 18	18 days ago
.gitignore	Cleanup	11 months ago
.travis.yml	Update deps. Closes #227	18 days ago
API.md	Move Readme to API	4 months ago
LICENSE.md	Update deps. Closes #227	18 days ago
README.md	add link to changelog	25 days ago
package.json	11.0.0	18 days ago

README.md

@hapi/cookie

Cookie authentication plugin.

cookie is part of the **hapi** ecosystem and was designed to work seamlessly with the [hapi web framework](#) and its other components (but works great on its own or with other frameworks). If you are using a different web framework and find this module useful, check out [hapi](#) – they work even better together.



cookie Installation & Registration

package.json updated

Register in server.js

npm install command

```
npm install @hapi/cookie
```

```
{
  "name": "playtime",
  "version": "0.3.0",
  "description": "A Playlist application for the HDip in Computing, WIT",
  "main": "src/server.js",
  "type": "module",
  "scripts": {
    "start": "node src/server.js",
    "dev": "./node_modules/.bin/nodemon --verbose src/server.js",
    "lint": "./node_modules/.bin/eslint . --ext .js"
  },
  "dependencies": {
    "@hapi/hapi": "^20.2.1",
    "@hapi/vision": "^6.1.0",
    "@hapi/cookie": "^11.0.2",
    "dotenv": "^10.0.0",
    "handlebars": "^4.7.7",
    "lowdb": "^3.0.0",
    "uuid": "^8.3.2",
    "joi": "^17.5.0"
  },
  "devDependencies": {
    "eslint": "^7.32.0",
    "eslint-config-airbnb-base": "^15.0.0",
    "eslint-config-prettier": "^8.3.0",
    "eslint-plugin-import": "^2.25.3",
    "nodemon": "^2.0.15",
    "prettier": "^2.5.0"
  }
}
```

```
await server.register(Cookie);
```

Setting the Cookie


- Set the cookie when a user is authenticated
- Cookie contains the logged in user ID




```
login: {  
  handler: async function (request, h) {  
    const { email, password } = request.payload;  
    const user = await db.userStore.getUserByEmail(email);  
    if (!user || user.password !== password) {  
      return h.redirect("/");  
    }  
    request.cookieAuth.set({ id: user._id });  
    return h.redirect("/dashboard");  
  },  
},
```


Cookie Configuration 1

- Set an auth 'strategy' before application starts
- Specifies range or parameters, including:
 - password for securing cookie
 - cookie name




```
server.auth.strategy("session", "cookie", {  
  cookie: {  
    name: "playtime",  
    password: "secretpasswordnotrevealedtoanyone",  
    isSecure: false,  
  },  
  redirectTo: "/",  
  validate: accountsController.validate,  
});  
server.auth.default("session");
```



Cookie Configuration 2

- By default hapi-auth-cookie will only allow the cookie to be transferred over a secure TLS/SSL connection.
- This may not be convenient during development so you can set the `isSecure` option to `false`.



```
server.auth.strategy("session", "cookie", {
  cookie: {
    name: "playtime",
    password: "secretpasswordnotrevealedtoanyone",
    isSecure: false,
  },
  redirectTo: "/",
  validate: accountsController.validate,
});
server.auth.default("session");
```

- 
- Set 'standard' as the default strategy for all routes

Validate function

```
validate: accountsController.validate,
```

- A hook function triggered when before a route guard by this strategy is to be engaged
- Has access to the cookie via session parameter
- Perform additional checks and return result of these checks

```
async validate(request, session) {  
  const user = await db.userStore.getUserById(session.id);  
  if (!user) {  
    return { isValid: false };  
  }  
  return { isValid: true, credentials: user };  
},
```


Annotating Routes

- All routes are now 'guarded' by default, cookie based authentication mechanism
- Any attempt to visit a route will be rejected unless valid cookie detected.
- Some routes need to be available (to signup or login for instance)
- These routes must specifically disable auth mechanism



```
server.auth.default("session");
```



```
export const accountsController = {  
  index: {  
    auth: false,  
    handler: function (request, h) {  
      return h.view("main", { title: "Welcome to Playlist" });  
    },  
  },  
  showSignup: {  
    auth: false,  
    handler: function (request, h) {  
      return h.view("signup-view", { title: "Sign up for Playlist" });  
    },  
  },  
  signup: {  
    auth: false,  
    handler: async function (request, h) {  
      const user = request.payload;  
      await db.userStore.addUser(user);  
      return h.redirect("/");  
    },  
  },  
  showLogin: {  
    auth: false,  
    handler: function (request, h) {  
      return h.view("login-view", { title: "Login to Playlist" });  
    },  
  },  
},
```

Using the Cookie/Session - **request.auth.credentials**

- The validate function will have set the 'credentials'

```
async validate(request, session) {
  const user = await db.userStore.getUserById(session.id);
  if (!user) {
    return { valid: false };
  }
  return { valid: true, credentials: user };
},
```



```
export const dashboardController = {
  index: {
    handler: async function (request, h) {
      const loggedInUser = request.auth.credentials;
      const playlists = await db.playlistStore.getUserPlaylists(loggedInUser._id);
      const viewData = {
        title: "Playtime Dashboard",
        user: loggedInUser,
        playlists: playlists,
      };
      return h.view("dashboard-view", viewData);
    },
  },
},
```

- We can now use credentials in the controllers

Logout

- Cookie deleted
- Any attempt to access protected routes rejected

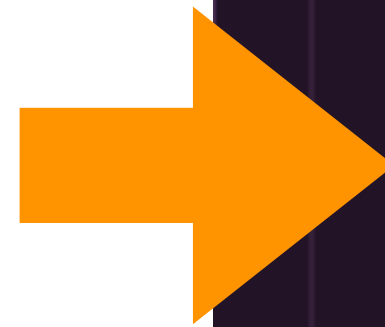


```
logout: {  
  handler: function (request, h) {  
    request.cookieAuth.clear();  
    return h.redirect("/");  
  },  
},
```

Redirects

If route
protected, and
cookie deleted/
timed out

Redirect to main



```
server.auth.strategy("session", "cookie", {  
  cookie: {  
    name: process.env.cookie_name,  
    password: process.env.cookie_password,  
    isSecure: false,  
  },  
  redirectTo: "/",  
  validateFunc: accountsController.validate,  
});  
server.auth.default("session");
```


Cookies can
be Inspected
in Browser

Playtime Dashboard

×

+

←

→

↻

🏠

localhost:3000/dashboard

🔗

📄

☆

f?

🌈

⚙️

E

Update

⋮

📄

📄

📄

Playtime

Dashboard

About

Logout

Playlist Title

Enter playlist title

Add Playlist

🔍📄 | Elements | Console | Sources | Network | Performance | Memory | Application | Security | Lighthouse

🔧 Service Workers

📄 Storage

Storage

▶ 📄 Local Storage

▶ 📄 Session Storage

📄 IndexedDB

📄 Web SQL

▼ 📄 Cookies

📄 http://localhost:3000

📄 Trust Tokens

Cache

📄 Cache Storage

📄 Back-forward Cache

Background Services

🔄 Filter

☒ Only show cookies with an issue

Name	Value	Do...	Path	Exp...	Size	Htt...	Sec...	Sa...	Sam...	P...
playlist	Fe26.2**cceac9ca53351caf4d47776...	loca...	/	Ses...	278	✓		Strict		Me...

Cookie Value

☐ Show URL decoded

Fe26.2**cceac9ca53351caf4d477762a8847fd124d630d745c74ed8d92ff48e2eac52d5*bMhHgUgACT_YtQE3m1Qd3Q*r-kyda0sYsB5O4MUMuJFuMCZ5pOz17cAgsACdQL9fKUjqFBldEjRZnuzJzJcLS8u**f4b424939d3dfc7e7ea4e8c56fa6828c4f70df5d63ebce8ef6cfe7e923afc708*UnPsj1q5svh-BovLqDKksCTwmGMsiwiY1UKV1mR8ETU

Sessions



Full Stack Web Development