Express - 9

Sessions

Introduction

Sessions are a crucial aspect of web development, allowing servers to maintain stateful information about users across multiple requests.

What are sessions ?

- A session is a way to store user-specific data on the server side
- Each session is associated with a unique identifier, typically stored in a cookie on the client side
- Sessions allow persistence of data between HTTP requests from the same client

Why use sessions ?

1. User Authentication
2. Personalization
3. Shopping Carts
4. Maintaining Application State

Implementing sessions in Express.JS

Express.js doesn't come with built-in session support, but we can use the express-session middleware.

```
npm install express-session
```

Configuring Session in Express JS

- **secret**: Used to sign the session ID cookie
- **resave**: Forces the session to be saved back to the session store
- **saveUninitialized**: Forces a session that is "uninitialized" to be saved to the store
- **cookie**: Settings for the session ID cookie

```
const express = require('express');
const session = require('express-session');

const app = express();

app.use(session({
  secret: 'your secret key',
  resave: false,
  saveUninitialized: true,
  cookie: { secure: false }
}));
```

Viewing Session Data

You can view session data by using

req.session

You can view session ID by using

req.session.id

Working of Sessions

- Client Initiates Request:

  The process begins when a client (typically a web browser) sends an HTTP request to the server.

- Session Middleware Intercepts:

  The Express.js session middleware intercepts this request before it reaches your route handlers.

- Cookie Check:

  The middleware checks if the request includes a session ID cookie.

Working of Sessions

4. Session Creation/Retrieval:

a. If no session ID cookie is found:
   i. A new session is created
   ii. A unique session ID is generated
   iii. A new session object is created in the session store
   iv. A session ID cookie is set in the response

b. If a session ID cookie is found:
   i. The session ID is extracted from the cookie
   ii. The corresponding session data is retrieved from the session store

Working of Sessions

- Session Data Attached:

  The session data (either newly created or retrieved) is attached to the request object as req.session.

- Request Handling:

  The request, now with session data attached, continues to your route handlers.

- Session Modification:

  Your application can now read from or write to req.session in your route handlers.

Working of Sessions

- Session Data Attached:

  The session data (either newly created or retrieved) is attached to the request object as req.session.

- Request Handling:

  The request, now with session data attached, continues to your route handlers.

- Session Modification:

  Your application can now read from or write to req.session in your route handlers.

Working of Sessions

- Response Preparation:

  After your route handler finishes processing, a response is prepared.

- Session Data Saved:

  If the session was modified during request processing, the session middleware saves the updated session data back to the session store.

- Cookie Sent:

  The session ID cookie is included in the response headers (if it's a new session or if the existing session was modified).

Working of Sessions

- Response Sent:

  The complete response, including any session cookies, is sent back to the client.

- Client Stores Cookie:

  The client receives the response and stores the session ID cookie.

- Subsequent Requests:

  For all subsequent requests, the client sends the session ID cookie, allowing the server to retrieve the associated session data.