Express-Body Parsing

When handling HTTP requests, it's often necessary to parse the request body to extract and process data sent by the client. In Express.js, this is achieved through a middleware called `body-parser`.

Example Code

```
 const express = require('express');
const bodyParser = require('body-parser');

const app = express();

// Enable parsing of JSON bodies
app.use(bodyParser.json());

// Define a route to handle POST requests
app.post('/users', (req, res) => {
  const { name, age } = req.body;
  console.log(`Received request with name: ${name}, age: ${age}`);
  // Process the data...
  res.send(`User created successfully!`);
});

// Listen on port 3000
const port = 3000;
app.listen(port, () => {
  console.log(`Server listening on port ${port}...`);
});
```

In this example:

1. We import `body-parser` as a middleware.
2. We call `app.use(bodyParser.json())` to enable parsing of JSON bodies in the request.
3. We define a route for POST requests to `/users`.
4. Within the handler, we access the parsed body data using `req.body`.
5. Finally, we send a response back to the client.

How Body Parser Works

When a client sends a HTTP request with a body (e.g., JSON, form data), Express will call the specified middleware (in this case, `body-parser`) to parse and extract the data from the request.

The parsed data is then stored in the `req.body` object, which can be accessed within route handlers. This allows you to easily access and process the client-provided data in a structured way.

Common Use Cases

- Parsing JSON bodies: When working with APIs or web services that return complex data structures.
- Processing form data: For handling HTML forms where users enter text input, radio buttons, or select options.

Remember to always enable parsing for the specific type of request body you're expecting (e.g., `body-parser.json()` for JSON, `body-parser.urlencoded({ extended: true })` for form data).

Curated by Brajesh Kumar