Character Maker API

Overview

The Character Maker API is a web application developed using Express.js and EJS templating. It allows users to create, view, edit, and delete character profiles. Each character has attributes such as attack, defense, health, and description. The application uses Bootstrap for a responsive design.

1. Server Setup (`index.js`)

1.1 Dependencies

The application relies on several key modules, including Express for routing, Path for handling file paths, Method-Override for simulating HTTP methods, and UUID for generating unique identifiers for characters.

1.2 Middleware Configuration

Middleware is set up to serve static files, parse incoming request data, and enable method overriding. This setup facilitates handling various HTTP requests effectively.

1.3 View Engine Setup

The application is configured to use EJS as its templating engine, with a specified directory for views. This allows for dynamic rendering of HTML templates.

1.4 Character Data Initialization

An array of character objects is initialized, each having a unique ID and various attributes. This array serves as the in-memory data store for the application.

```
// Import required modules
const express = require('express'); // Web framework for Node.js
const app = express(); // Create an Express application
const path = require('path'); // Module for handling file paths
const methodOverride = require('method-override'); // Middleware for
supporting HTTP methods

// Serve static files from the public directory
```

```
app.use(express.static('public'));
app.use(express.static(path.join(__dirname, 'public')));

// Import UUID for generating unique IDs
const { v4: uuid } = require('uuid');

// Middleware for parsing JSON and URL-encoded data
app.use(express.json());
app.use(express.urlencoded({ extended: true }));
// Middleware for overriding HTTP methods (for PUT/PATCH/DELETE)
app.use(methodOverride('_method'));

// Set the view engine to EJS and define the views directory
app.set('view engine', 'ejs');
app.set('views', path.join(__dirname, '/views'));

// Initial array of characters with unique IDs and attributes
let characters = [
    { id: uuid(), name: 'Mary', attack: 10, defense: 20, health: 20,
description: 'I am a default female' },
    // ... other characters
];

// Route to display the character maker page with all characters
app.get('/maker', (req, res) => {
    res.render('maker', { characters });
});

// Route to show the form for creating a new character
app.get('/maker/new', (req, res) => {
    res.render('maker/new');
});

// Route to handle new character creation
app.post('/maker', (req, res) => {
    const { name, attack, defense, health, description } = req.body; //
Destructure input values
    characters.push({ name, attack, id: uuid(), defense, health,
description }); // Add new character
    res.redirect('maker'); // Redirect to the character maker page
});

// Route to display details of a specific character
app.get('/maker/:id', (req, res) => {
    const { id } = req.params; // Extract ID from request parameters
```

```
    const character = characters.find(c => c.id === id); // Find the character
by ID
    res.render('maker/id', { character }); // Render the character detail page
});

// Route to show the edit form for a specific character
app.get('/maker/:id/edit', (req, res) => {
    const { id } = req.params; // Extract ID from request parameters
    const character = characters.find(c => c.id === id); // Find the character
by ID
    res.render('maker/edit', { character }); // Render the edit form
});

// Route to handle character updates
app.patch('/maker/:id', (req, res) => {
    const { id } = req.params; // Extract ID from request parameters
    const foundCharacter = characters.find(c => c.id === id); // Find the
character
    // Update character attributes with new values
    foundCharacter.name = req.body.name;
    foundCharacter.attack = req.body.attack;
    foundCharacter.defense = req.body.defense;
    foundCharacter.health = req.body.health;
    foundCharacter.description = req.body.description;
    res.redirect(id); // Redirect to the updated character's detail page
});

// Route to delete a character
app.delete('/maker/:id', (req, res) => {
    const { id } = req.params; // Extract ID from request parameters
    characters = characters.filter(c => c.id !== id); // Remove character from
the array
    res.redirect('/maker'); // Redirect to the character maker page
});

// Start the server on port 8080
app.listen(8080, () => {
    console.log('Listening to port 8080');
});
```

2. Routing Functions

2.1 GET `/maker`

This route renders the main character overview page, displaying all existing characters by passing the character data to the `maker.ejs` template.

2.2 GET `/maker/new`

This route displays a form for creating a new character, rendering the `new.ejs` template.

2.3 POST `/maker`

This route processes form submissions to create a new character. It extracts character attributes from the request body, adds the new character to the data array, and redirects to the character overview.

2.4 GET `/maker/:id`

This route retrieves a specific character based on the ID in the URL. It finds the character in the data array and passes it to the `id.ejs` template for rendering.

2.5 GET `/maker/:id/edit`

This route renders the edit form for a specific character, pre-filling the form fields with the character's current attributes for user convenience.

2.6 PATCH `/maker/:id`

This route updates a character's attributes based on submitted form data. It merges the updated values into the found character object and redirects to the character's detailed view.

2.7 DELETE `/maker/:id`

This route removes a character from the array based on the ID. It filters out the deleted character and redirects back to the main overview.

3. EJS Templates

3.1. partials/head.ejs

- **Purpose:** Sets up the HTML document's head section.
- **Key Elements:**
    - **Meta Tags:** Includes character encoding (UTF-8) and viewport settings for responsive design.
    - **Title:** Specifies the page title as "Character Maker."
    - **Stylesheets:** Links to `reset.css` for CSS resets and Bootstrap CSS for responsive layout and components.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8"> <!-- Character encoding -->
    <meta name="viewport" content="width=device-width, initial-scale=1.0"> <!-
- Responsive design -->
    <title>Character Maker</title> <!-- Page title -->

    <link rel="stylesheet" href="reset.css"> <!-- Include reset.css for CSS
normalization -->
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH"
crossorigin="anonymous"> <!-- Bootstrap CSS -->
</head>

<body class="d-flex flex-column min-vh-100 bg-light" style="background:
linear-gradient(to bottom right, #f0f8ff, #e6f7ff);"> <!-- Body styles -->
    <nav class="navbar bg-primary" data-bs-theme="dark"> <!-- Navigation bar -
->
        <div class="container-fluid">
            <span class="navbar-brand mb-0 h1">Character Maker</span> <!--
Brand name -->
        </div>
    </nav>
```

3.2. partials/footer.ejs

- **Purpose:** Contains the footer for the web application.
- **Key Elements:**

- **Background Color:** The footer has a primary color scheme to match the navigation bar.
- **Text:** Displays a credit message ("mycko made this") centered within the footer.
- **Scripts:** Includes the Bootstrap JavaScript bundle for enabling interactive components, ensuring proper functionality of any dynamic UI elements.

```
<footer class="bg-primary mt-auto"> <!-- Footer with background color -->
    <p class="text-light fw-bold text-center pt-1">mycko made this</p> <!--
Footer text -->
</footer>

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min
.js" integrity="sha384-
YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz"
crossorigin="anonymous"></script> <!-- Bootstrap JS -->
</body>
</html>
```

3.3. views/maker.ejs

- **Purpose:** Displays a grid of character cards representing all characters.
- **Key Elements:**
  - **Responsive Grid Layout:** Utilizes Bootstrap's grid system to create a flexible layout that adapts to different screen sizes.
  - **Character Cards:** For each character, a card displays:
    - **Image:** A generated avatar for the character.
    - **Name:** The character's name as the card title.
    - **Link:** Each card links to the character's detail view for easy access.
  - **Add New Character Card:** A dedicated card that links to the form for creating a new character, encouraging user engagement.

```
<%- include('partials/head') %> <!-- Include head partial -->
<div class="container-fluid"> <!-- Main container -->
    <div class="row justify-content-center"> <!-- Centered row -->
        <% for (c of characters) { %> <!-- Loop through characters -->
            <div class="col-xl-2 col-lg-3 col-md-4 col-sm-6 col-12 mt-3 mx-2
mb-3"> <!-- Responsive column -->
                <a href="/maker/<%= c.id %>" class="text-decoration-none"> <!
- Link to character details -->
                    <div class="card shadow-sm h-100 rounded" style="cursor:
pointer;"> <!-- Character card -->
```

```
                        <img
src="https://api.dicebear.com/9.x/adventurer/svg?seed=<%= c.id %>"
class="card-img-top rounded-top" alt="<%= c.name %>"> <!-- Character avatar --
>
                        <div class="card-body text-center"> <!-- Card body -->
                            <h5 class="card-title"><%= c.name %></h5> <!--
Character name -->
                        </div>
                    </div>
                </a>
            </div>
        <% } %>

        <div class="col-xl-2 col-lg-3 col-md-4 col-sm-6 col-12 mt-3 mx-2 mb-
3"> <!-- Add new character card -->
            <a href="/maker/new" class="text-decoration-none"> <!-- Link to
create a new character -->
                <div class="card shadow-sm h-100 rounded" style="cursor:
pointer;">
                    <div class="card-body text-center d-flex flex-column
justify-content-center align-items-center" style="height: 100%;"> <!-- Card
body -->
                        <h5 class="card-title">Add a New Character</h5> <!--
Add character title -->
                    </div>
                </div>
            </a>
        </div>
    </div>
</div>

<%- include('partials/footer') %> <!-- Include footer partial -->
```

3.4. views/maker/id.ejs

- **Purpose:** Displays detailed information about a specific character.
- **Key Elements:**
    - **Character Avatar:** Displays a circular avatar generated based on the character's ID.
    - **Character Details:** Shows various attributes:
        - **Name:** Highlighted prominently.
        - **Attack, Defense, Health, Description:** Each attribute is presented with a label for clarity.

- **Action Buttons:**
  - **Edit Button:** Directs to the edit page for the character.
  - **Delete Button:** Sends a DELETE request to remove the character from the list.
  - **Back to Home Button:** Returns the user to the character overview page.

```
<%- include('../partials/head') %> <!-- Include head partial -->
<div class="container-fluid mt-4"> <!-- Main container -->
    <div class="card mx-auto" style="max-width: 600px;"> <!-- Character card -
->
        <div class="card-body d-flex flex-column flex-md-row align-items-
center"> <!-- Card body -->
            <div class="text-center mb-3 mb-md-0"> <!-- Avatar section -->
                <img src="https://api.dicebear.com/9.x/adventurer/svg?seed=<%=
character.id %>" alt="Character Avatar" class="img-fluid rounded-circle"
style="width: 150px; height: 150px;"> <!-- Avatar image -->
            </div>

            <div class="ms-md-4"> <!-- Character details section -->
                <h5 class="card-title text-center text-md-start"><%=
character.name %></h5> <!-- Character name -->
                <p class="card-text mb-1"><strong>Attack:</strong> <%=
character.attack %></p> <!-- Attack value -->
                <p class="card-text mb-1"><strong>Defense:</strong> <%=
character.defense %></p> <!-- Defense value -->
                <p class="card-text mb-1"><strong>Health:</strong> <%=
character.health %></p> <!-- Health value -->
                <p class="card-text"><strong>Description:</strong> <%=
character.description %></p> <!-- Description -->
            </div>
        </div>

        <div class="d-flex flex-column flex-md-row justify-content-center mt-
3"> <!-- Action buttons -->
            <form action="/maker/<%= character.id %>/edit" class="mb-2 me-md-
2"> <!-- Edit form -->
                <button class="btn btn-success w-100 w-md-auto">Edit</button>
<!-- Edit button -->
            </form>
            <form action="/maker/<%= character.id %>?_method=DELETE"
method="post" class="mb-2 me-md-2"> <!-- Delete form -->
                <button class="btn btn-danger w-100 w-md-auto">Delete</button>
<!-- Delete button -->
            </form>
            <form action="/maker" method="get"> <!-- Back form -->
```

```
                <button class="btn btn-primary w-100 w-md-auto">Back to
Home</button> <!-- Back button -->
            </form>
        </div>
    </div>
</div>

<%- include('../partials/footer') %> <!-- Include footer partial -->
```

3.5. views/maker/edit.ejs

- **Purpose:** Provides a form for editing an existing character's attributes.
- **Key Elements:**
    - **Form Setup:** Uses the PATCH method to update character details.
    - **Pre-filled Input Fields:** Each input is populated with the current character's values for easy editing:
        - **Character Name:** Text input limited to 8 characters.
        - **Attack, Defense, Health:** Number inputs with a maximum value of 100.
        - **Description:** Text input limited to 25 characters.
    - **Character Avatar:** Displays the character's avatar to visually represent the character while editing.
    - **Save and Cancel Buttons:**
        - **Save Button:** Submits the form to update the character.
        - **Cancel Button:** Navigates back to the character's detail view without saving changes.

```
<%- include('../partials/head') %> <!-- Include head partial -->

<form action="/maker/<%= character.id %>?_method=PATCH" method="post"> <!--
Form for editing a character -->
    <div class="container-fluid"> <!-- Main container -->
        <div class="row justify-content-center m-3"> <!-- Centered row -->
            <div class="col-lg-8 col-md-10 col-sm-12 mb-4"> <!-- Responsive
column -->
                <section class="mb-2"> <!-- Character name input -->
                    <label for="name" class="form-label">Character
Name:</label>
                    <input type="text" class="form-control" id="name"
name="name" placeholder="name" maxlength="8" required value="<%=
character.name %>"> <!-- Input field with default value -->
                </section>
```

```
                <section class="mb-2"> <!-- Character attack input -->
                    <label for="attack" class="form-label">Attack:</label>
                    <input type="number" id="attack" name="attack"
placeholder="attack" class="form-control" max="100" required value="<%=
character.attack %>"> <!-- Input field with default value -->
                </section>

                <section class="mb-2"> <!-- Character defense input -->
                    <label for="defense" class="form-label">Defense:</label>
                    <input type="number" id="defense" name="defense"
placeholder="defense" class="form-control" max="100" required value="<%=
character.defense %>"> <!-- Input field with default value -->
                </section>

                <section class="mb-2"> <!-- Character health input -->
                    <label for="health" class="form-label">Health:</label>
                    <input type="number" id="health" name="health"
placeholder="health" class="form-control" max="100" required value="<%=
character.health %>"> <!-- Input field with default value -->
                </section>

                <section class="mb-2"> <!-- Character description input -->
                    <label for="description" class="form-
label">Description</label>
                    <input type="text" name="description" id="description"
class="form-control" placeholder="type here.." maxlength="25" required
value="<%= character.description %>"> <!-- Input field with default value -->
                </section>
            </div>

            <div class="col-lg-4 col-md-6 col-sm-12 d-flex justify-content-
center align-items-center mb-4"> <!-- Character avatar -->
                <img
src="https://api.dicebear.com/9.x/adventurer/svg?seed=<%=character.id %>"
class="img-fluid rounded" alt="Character Avatar"> <!-- Avatar image -->
            </div>
        </div>

        <div class="row justify-content-center"> <!-- Centered row for buttons
-->
            <div class="col-lg-4 col-md-6 col-sm-12 d-flex justify-content-
between mb-3"> <!-- Responsive column -->
                <button class="btn btn-success w-100 me-2">Save</button> <!--
Save button -->
                <a href="/maker/<%= character.id %>" class="btn btn-danger w-
100">Cancel</a> <!-- Cancel button -->
            </div>
```

```
        </div>
    </div>
</form>

<%- include('../partials/footer') %> <!-- Include footer partial -->
```

3.6. views/maker/new.ejs

- **Purpose:** Displays a form for creating a new character.
- **Key Elements:**
    - **Form Setup:** Uses the POST method to submit new character data.
    - **Input Fields:** Each field is required and has specific attributes:
        - **Character Name:** Text input limited to 8 characters.
        - **Attack, Defense, Health:** Number inputs, each required and capped at 100.
        - **Description:** Text input limited to 25 characters.
    - **Submit Button:** Submits the form to create a new character.

```
<%- include('../partials/head') %> <!-- Include head partial -->

<form action="/maker" method="post"> <!-- Form for creating a new character --
>
    <div class="container mt-4"> <!-- Main container -->
        <div class="row justify-content-center"> <!-- Centered row -->
            <div class="col-md-6 col-lg-4"> <!-- Responsive column -->
                <div class="d-flex flex-column"> <!-- Flex container for form
inputs -->

                    <section class="mb-3"> <!-- Character name input -->
                        <label for="name" class="form-label">Character
Name:</label>
                        <input type="text" class="form-control" id="name"
name="name" placeholder="name" maxlength="8" required> <!-- Input field -->
                    </section>

                    <section class="mb-3"> <!-- Character attack input -->
```

```
                        <label for="attack" class="form-label">Attack:</label>
                        <input type="number" id="attack" name="attack"
placeholder="attack" class="form-control" max="100" required> <!-- Input field
-->
                    </section>

                    <section class="mb-3"> <!-- Character defense input -->
                        <label for="defense" class="form-
label">Defense:</label>
                        <input type="number" id="defense" name="defense"
placeholder="defense" class="form-control" max="100" required> <!-- Input
field -->
                    </section>

                    <section class="mb-3"> <!-- Character health input -->
                        <label for="health" class="form-label">Health:</label>
                        <input type="number" id="health" name="health"
placeholder="health" class="form-control" max="100" required> <!-- Input field
-->
                    </section>

                    <section class="mb-3"> <!-- Character description input --
>
                        <label for="description" class="form-
label">Description:</label>
                        <input type="text" name="description" id="description"
class="form-control" placeholder="type here.." maxlength="25" required> <!--
Input field -->
                    </section>

                    <button class="btn btn-success w-100">Submit</button> <!--
Submit button -->


                </div>
            </div>
        </div>
    </div>
</form>

<%- include('../partials/footer') %> <!-- Include footer partial -->
```

4. Public Directory

The public directory includes a CSS file (reset.css) that resets default styling across browsers to ensure a consistent appearance of elements.