**Explanation:**

**Server Setup (index.js):**

**Express Setup: Sets up an Express app with necessary middleware for static files and form data.**

**Blog Storage (blogs Array): Initializes an array to store blogs.**

**Middleware (viewDate): Middleware to extract title and content from the request body and store in the blogs array.**

**View Page (view.ejs):**

**Display Blogs: Iterates through each blog in the blogs array and displays their titles and content.**

**Update Form: For each blog, includes a form with text input and textarea for updating the title and content.**

**Delete Form: For each blog, includes a form with a delete button.**

**Update and Delete Routes (index.js):**

**Update Route: Handles POST requests to the "/update/:id" endpoint.**

**Extracts updated title and content from the request body.**

**Finds the blog in the array with the matching ID and updates its title and content.**

**Renders the view with the updated blogs array.**

**Delete Route: Handles POST requests to the "/delete/:id" endpoint.**

**Extracts the blog ID from the route parameter.**

**Filters out the blog with the specified ID from the array.**

**Renders the view with the updated blogs array.**

**Notes:**

**The code uses Express for server setup, EJS for rendering views, and basic HTML forms for user input.**

**Update and delete functionality is implemented with route handlers that modify the blogs array.**

**Appropriate error handling is included to handle cases where blogs or specific blogs are not found.**

**Forms on the view page are structured to send POST requests to the respective update or delete routes with the appropriate blog ID.**

# Update and Delete Commands

1. Update Route (/update/:id):

Route Definition:

Handles POST requests to update a blog.

Extracts updated title and content from the request body.

Extracts the blog ID from the route parameter.

Searches for the corresponding blog in the blogs array.

Update Operation:

If the blog is found, updates its title and content.

Renders the view.ejs template with the updated blogs array.

If the blog is not found, renders the view.ejs template with an error message.

2. Delete Route (/delete/:id):

Route Definition:

Handles POST requests to delete a blog.

Extracts the blog ID from the route parameter.

Uses the filter function to create a new array excluding the blog with the specified ID.

Deletion Operation:

Checks if any blog was deleted by comparing the length of the updated array with the original blogs array.

Renders the view.ejs template with the updated blogs array if a blog was deleted.

If the blog is not found, renders the view.ejs template with an error message.

3. View Page Integration (views/view.ejs):

Update Form (/update/:id):

Generates a form for each blog with input fields for updated title and content.

The form has a POST action pointing to the /update/:id route.

Delete Form (/delete/:id):

Generates a form for each blog with a button for deletion.

The form has a POST action pointing to the /delete/:id route.

Summary:

Update (/update/:id):

Handles POST requests to update a blog.

Extracts data, identifies the blog, updates it, and renders the updated view.

Delete (/delete/:id):

Handles POST requests to delete a blog.

Extracts data, filters out the specified blog, renders the updated view, or shows an error if the blog is not found.

View Integration (views/view.ejs):

Generates forms in the view for updating and deleting blogs, with actions pointing to the respective routes.

How req.params.id works

req.params.id; how is this works

ChatGPT

In Express.js, req.params is an object containing properties mapped to the named route parameters. When a route is defined with a parameter, like :id in /update/:id, Express extracts the value of that parameter from the URL and stores it in req.params.

For example:

javascript

Copy code

// Route definition

app.post('/update/:id', (req, res) => {

const blogId = req.params.id;

// ... rest of the code

});

In this route handler, req.params.id is used to retrieve the value of the id parameter from the URL. If you have a URL like /update/123, then req.params.id would be '123'. The value is always a string, so if you need it as a number, you may want to parse it:

javascript

Copy code

const blogId = parseInt(req.params.id, 10);

Now, blogId will be a number (123 in the example). This parameter is useful when you want to work with dynamic values in your routes. It allows you to create more flexible route handlers that can respond to different values for id in the URL.

const blogToUpdate = blogs.find(blog => blog.id == blogId); explain this

blogs: This assumes you have an array named blogs containing objects, where each object represents a blog. Each blog object is expected to have an id property.

find method: The find method is an array method in JavaScript. It is used to search for an element in the array that satisfies the provided testing function. Once the first element is found, it stops and returns that element.

blog => blog.id == blogId: This is an arrow function used as the testing function for find. It checks if the id property of a blog object is equal (==) to the blogId variable.

blog is a parameter representing each element in the blogs array during the iteration.

blog.id refers to the id property of the current blog in the iteration.

blog.id == blogId checks if the id of the current blog is equal to the blogId we obtained from req.params.id.

blogToUpdate: This variable holds the result of the find operation. If a blog with an id matching blogId is found in the blogs array, blogToUpdate will be set to that blog object. If no matching blog is found, blogToUpdate will be undefined.

So, in summary, this line of code is searching for a blog in the blogs array that has the same id as the one extracted from the route parameters (blogId). The found blog (if any) is stored in the blogToUpdate variable.