DISSECTING

MVC

Model View Controller

Exploring Models

- **File Involved:** post.js
- **Purpose:** The model manages the data and business logic of the application. Here, it is responsible for storing and manipulating blog post data.
- **Explanation:**
  - The Post model is an object that contains methods to access and manage blog posts.
  - Methods:
    - all(): Retrieves all posts.
    - findById(id): Finds a post by its ID.
    - create(title, content): Creates a new post by adding it to the in-memory posts array.
- **Role in MVC:** The model acts as the data layer, handling all data-related logic, such as fetching, creating, and updating posts.

models\post.js

models\post.js

```
const Post = {
    all: () => posts,
    findById: (id) => posts.find(post => post.id === parseInt(id)),
    create: (title, content) => {
      const newPost = { id: posts.length + 1, title, content };
      posts.push(newPost);
      return newPost;
    }
};

module.exports = Post;
```

Exploring Views

- **Files Involved:** index.ejs, post.ejs
- **Purpose:** Using EJS (Embedded JavaScript) templates to dynamically generate HTML for displaying posts.
- **Explanation:**
  - **Index.ejs:**
    - Displays a list of all posts and includes a form to add a new post.
    - Loops through the posts array (provided by the controller) and generates a link for each post, linking to its detailed view.
  - **Post.ejs:**
    - Displays an individual post, including its title and content.
    - Includes a link to return to the list of all posts.
- **Role in MVC:** The view represents the presentation layer, showing data to the user and allowing them to interact with the app (e.g., through forms).

views\index.ejs

views\index.ejs

```
<body>
  <h1>Welcome to the Simple Blog</h1>
  <h2>All Posts</h2>
  <ul>
    <% posts.forEach(post => { %>
      <li><a href="/posts/<%= post.id %>"><%= post.title %></a></li>
    <% }) %>
  </ul>
  <h3>Add a New Post</h3>
  <form action="/posts/create" method="POST">
    <label>Title:</label><br>
    <input type="text" name="title" required><br>
    <label>Content:</label><br>
    <textarea name="content" required></textarea><br>
    <button type="submit">Submit</button>
  </form>
</body>
```

views\post.ejs

views\post.ejs

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title><%= post.title %></title>
</head>
<body>
  <h1><%= post.title %></h1>
  <p><%= post.content %></p>
  <a href="/">Back to All Posts</a>
</body>
</html>
```

Exploring Controllers

- **File Involved:** postController.js
- **Purpose:** It handles user input, processes requests, and determines what data should be sent to the view.
- **Explanation:**
  - The postController contains functions that define the behavior of the app when certain routes are accessed.
  - **Methods:**
    - **getAllPosts:** Calls Post.all() to retrieve all posts and passes them to the index.ejs view to render.
    - **getPostById:** Uses Post.findById(req.params.id) to get a specific post based on the ID in the URL, then passes it to the post.ejs view.
    - **createPost:** Takes data from the form (title and content), creates a new post by calling Post.create, and redirects the user to the homepage.
- **Role in MVC:** The controller acts as the logic layer, handling requests and determining which view should display the data.

controllers\postController.js

controllers\postController.js

```
const Post = require('../models/post');
const getAllPosts = (req, res) => {
  const posts = Post.all();
  res.render('index', { posts });
};
const getPostById = (req, res) => {
  const post = Post.findById(req.params.id);
  if (!post) {
    return res.status(404).send('Post not found');
  }
  res.render('post', { post });
};
const createPost = (req, res) => {
  const { title, content } = req.body;
  const newPost = Post.create(title, content);
  res.redirect('/');
};
module.exports = { getAllPosts, getPostById, createPost };
```

Routing

- **File Involved:** app.js
- **Purpose:** Routing is essential in MVC for directing requests to the correct controller function based on the URL.
- **Explanation:**
  - The routes in app.js map URLs to controller functions:
    - **GET /** → getAllPosts (displays all posts)
    - **GET /posts/:id** → getPostById (displays a specific post)
    - **POST /posts/create** → createPost (adds a new post and redirects)

app.js

app.js

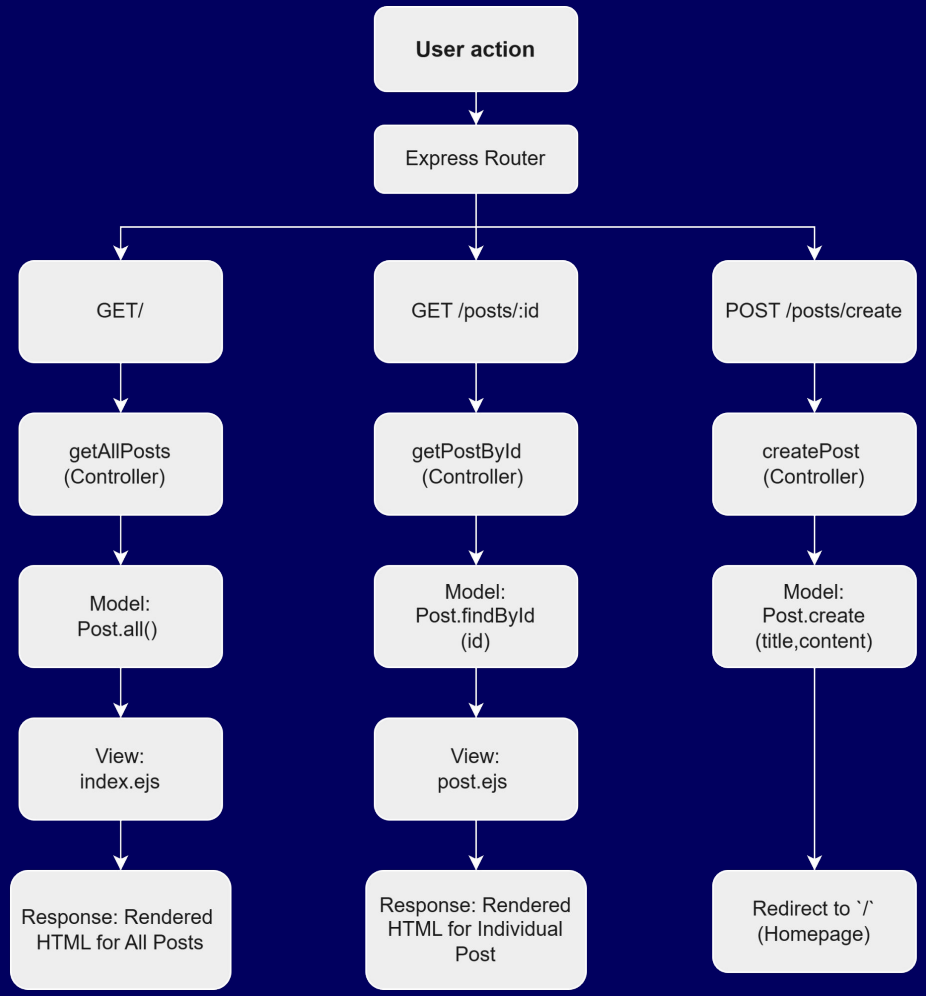
```javascript
const express = require('express');
const postController = require('./controllers/postController');

const app = express();

app.get('/', postController.getAllPosts);
app.get('/posts/:id', postController.getPostById);
app.post('/posts/create', postController.createPost);

app.listen(3000, () => {
  console.log(`Server running on http://localhost:3000`);
});
```

THE BIG PICTURE

Flow of MVC app

1. **User Request:** The user interacts with the app (e.g., viewing or creating posts).
2. **Express Router:** Routes the request to the appropriate controller.
3. **Controller:** Fetches or updates data using the model and sends it to the view.
4. **Model:** Manages the data (e.g., fetching or adding posts).
5. **View:** Renders HTML based on the data.
6. **Response:** Sends the rendered HTML back to the user.

This structure ensures clear **separation of concerns**, simplifying maintenance and extension.



THANK YOU