第2次隨堂題目-隨堂-QZ2

> 學號：112111119 (學號和姓名都要寫)
> 姓名：賴俊升

本份文件包含以下主題：(至少需下面兩項，若是有多者可以自行新增)

- [x] 說明內容

說明程式與內容

1. 第一部分 建立 MIME 類型模組

函式getContentType(extname)

Ans:

檔案位置： `utils/mimeTypes.js`

此模組負責定義檔案副檔名與 Content-Type 的對應關係，將資料與邏輯分離，方便統一管理支援的檔案類型。

```
const contentTypes = {
    '.html': 'text/html; charset=utf-8',          // HTML 網頁文件
    '.ejs': 'text/html; charset=utf-8',           // EJS 模板（渲染後輸出為 HTML）
    '.js': 'text/javascript; charset=utf-8',      // JavaScript 腳本文件
    '.css': 'text/css; charset=utf-8',            // CSS 樣式表文件
    '.json': 'application/json',                  // JSON 資料格式
    '.png': 'image/png',                          // PNG 圖片格式
    '.jpg': 'image/jpg',                          // JPG/JPEG 圖片格式
    '.gif': 'image/gif',                          // GIF 動畫圖片
    '.svg': 'image/svg+xml',                      // SVG 向量圖形
    '.ico': 'image/x-icon'                        // 網站 favicon 圖示
  };
```

函式名稱：getContentType 根據檔案的副檔名回傳對應的 MIME 類型

```
export function getContentType(extname) {
  return contentTypes[extname] || contentTypes['default']; // 如果找不到指定的副檔名，則回傳預設的 'application/octet-stream'
}
```

2. 第二部分 建立模板渲染模組

函式renderTemplate(res, filePath, data)

Ans:

檔案位置：`utils/templateRenderer.js`

此模組封裝了 EJS 模板引擎的渲染邏輯與 HTTP 回應處理，包含成功渲染與錯誤處理 (404/500) 。

```
import { readFile } from 'fs'; // Node.js 檔案系統模組
import { render } from 'ejs';   // EJS 模板引擎
```

函式名稱：renderTemplate 讀取 EJS 模板檔案，渲染內容後發送給客戶端

```
export function renderTemplate(res, filePath, data = {}) { // 默認資料為空物件
  readFile('.' + filePath, 'utf8', (err, template) => { // 讀取 EJS 模板文件
    if (err) { // 讀取錯誤
      res.writeHead(500, { 'Content-Type': 'text/html; charset=utf-8' }); // 伺服器錯誤
      res.end(`錯誤：無法讀取模板文件 - ${err.message}`); // 回傳錯誤訊息
      return;
    }

    try { // 嘗試渲染模板
      const html = render(template, data); // 使用 EJS 渲染模板
      res.writeHead(200, { 'Content-Type': 'text/html; charset=utf-8' }); // 成功回應
      res.end(html);  // 回傳渲染後的 HTML
    } catch (renderErr) {   // 渲染錯誤
      res.writeHead(500, { 'Content-Type': 'text/html; charset=utf-8' }); // 伺服器錯誤
      res.end(`模板渲染錯誤：${err.message}`); // 回傳錯誤訊息
    }
  });
}
```

函式名稱：render404 處理 404 錯誤回應

```
export function render404(res) { // 使用 renderTemplate 函式渲染 404 頁面
  renderTemplate(res, '/index3.ejs', { message: '頁面不存在' }); // 傳入錯誤訊息
}
```

3. 第三部分：建立靜態文件處理模組

函數handleStaticFile(res, filePath)

Ans: 檔案位置： `utils/staticFileHandler.js`

此模組負責處理非模板類的靜態資源請求 (如 CSS, 圖片, 客戶端 JS)，並整合了前述的 MIME 類型判斷與 404 錯誤處理 。

```
import { readFile } from 'fs'; // Node.js 檔案系統模組
import { extname } from 'path'; // Node.js 路徑模組
import { getContentType } from './mimeTypes.js'; // 正確匯入  getContentType
import { render404 } from './templateRenderer.js';  // 正確匯入 render404
```

函式名稱：handleStaticFile 處理所有靜態檔案 (CSS, JS, 圖片等) 的讀取與回應

```
export function handleStaticFile(res, filePath) { // 處理靜態檔案請求
  const fullPath = '.' + filePath; // 假設靜態檔案位於專案根目錄下
  const ext = extname(filePath); // 取得檔案副檔名
  const contentType = getContentType(ext); // 使用匯入的 getContentType 函式取得
MIME 類型

  readFile(fullPath, (err, content) => { // 讀取檔案內容
    if (err) { // 發生錯誤（例如檔案不存在）
      render404(res);  // 直接使用匯入的 render404
      return;
    }

    res.writeHead(200, { 'Content-Type': contentType }); // 設定成功回應標頭
    res.end(content); // 回傳檔案內容
  });
}
```

4. 第四部分：重構主檔案

引入必要的模組

Ans: 檔案位置： `2b-refactored.js`

主程式專注於伺服器的建立與路由分發 (Routing)，將具體實作邏輯委派給上述模組處理，使程式碼結構清晰易維護 。

```
import { createServer } from 'http'; // Node.js HTTP 模組
import { renderTemplate } from './utils/templateRenderer.js'; // 正確匯入
renderTemplate
import { handleStaticFile } from './utils/staticFileHandler.js'; // 正確匯入
handleStaticFile
```

switcht程式

```
createServer((req, res) => { // 建立 HTTP 伺服器
  let filePath = ''; // 用於存放要渲染的模板路徑
  let data = {}; // 用於存放傳遞給模板的資料

  // 路由邏輯：使用 switch 處理明確路徑
```

```
  switch (req.url) { // 根據請求的 URL 進行路由處理
    case '/': // 根路徑
      filePath = '/index.ejs'; // 指定要渲染的 EJS 模板路徑
      renderTemplate(res, filePath, data); // 呼叫渲染函式並傳遞資料
      return;

    case '/calculator': // 計算機路徑
      filePath = '/index2.ejs'; // 指定計算機模板路徑
      renderTemplate(res, filePath); // 呼叫渲染函式（無需額外資料）
      return;

    default: // 預設情況
      // 其他路徑 → 視為靜態文件
      handleStaticFile(res, req.url); // 呼叫靜態文件處理函式
      return;
  }
```

啟動路由

```
.listen(3000, () => {
  // 在終端機（控制台）輸出訊息，告知開發者伺服器已啟動
  // 使用者可以透過瀏覽器訪問 http://localhost:3000 來查看網站
  console.log('伺服器已啟動！請訪問 http://localhost:3000');
  console.log('可用路由：');
  console.log('  - http://localhost:3000');
  console.log('  - http://localhost:3000/calculator');
  console.log('  - 其他路徑將顯示 404 錯誤頁面');
});
```