第2次隨堂題目-隨堂-QZ2

學號：112111108
姓名：詹羽庭

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

第一部分：建立 MIME 類型模組（utils/mimeTypes.js）

分將原本寫在主程式 `2b.js` 中的 MIME 類型對照表抽取為獨立模組，以避免主程式過於冗長，並提升程式的可讀性與可維護性。
透過集中管理副檔名與 Content-Type 的對應關係，後續若需要新增或修改檔案類型，只需調整此模組即可。

程式碼（utils/mimeTypes.js）：

```
import path from "path";

// MIME 類型對照表
// Key 為副檔名，Value 為瀏覽器辨識用的 Content-Type
const contentTypes = {
  ".html": "text/html; charset=utf-8",
  ".ejs": "text/html; charset=utf-8",
  ".js": "text/javascript; charset=utf-8",
  ".css": "text/css; charset=utf-8",
  ".json": "application/json; charset=utf-8",
  ".png": "image/png",
  ".jpg": "image/jpeg",
  ".jpeg": "image/jpeg",
  ".gif": "image/gif",
  ".svg": "image/svg+xml",
  ".ico": "image/x-icon",
};

/**
 * getContentType
 * 根據副檔名回傳對應的 MIME 類型
 *
 * @param {string} extname 副檔名（.css）或檔名（style.css）
 * @returns {string} 對應的 Content-Type
 */
export function getContentType(extname) {
  // 若傳入的是檔名，先取出副檔名
  const ext = extname.startsWith(".")
    ? extname
    : path.extname(extname);
```

```
  // 若找不到對應類型，回傳預設 text/plain
  return contentTypes[ext] || "text/plain; charset=utf-8";
}
```

第二部分：建立模板渲染模組（utils/templateRenderer.js）

將原本分散於主程式中的 EJS 模板渲染流程抽取為獨立模組，統一負責讀取模板、渲染資料與回傳 HTML。 這樣主檔案只需呼叫函式即可完成頁面渲染，不必關心實作細節。 **程式碼（utils/templateRenderer.js）**：

```
import fs from "fs/promises";
import ejs from "ejs";

/**
 * renderTemplate
 * 統一處理 EJS 模板讀取與渲染
 *
 * @param {ServerResponse} res HTTP 回應物件
 * @param {string} filePath EJS 檔案路徑
 * @param {object} data 傳遞給模板的資料（預設為空物件）
 * @param {number} statusCode HTTP 狀態碼（預設 200）
 */
export async function renderTemplate(
  res,
  filePath,
  data = {},
  statusCode = 200
) {
  try {
    // 讀取 EJS 模板檔案（UTF-8 文字檔）
    const template = await fs.readFile(filePath, "utf8");

    // 使用 EJS 將資料渲染成 HTML
    const html = ejs.render(template, data);

    // 設定回應標頭並回傳 HTML
    res.writeHead(statusCode, { "Content-Type": "text/html; charset=utf-8" });
    res.end(html);
  } catch (err) {
    // 若檔案讀取或渲染失敗，回傳 500 錯誤
    res.writeHead(500, { "Content-Type": "text/html; charset=utf-8" });
    res.end(`500 - 伺服器錯誤：無法渲染模板<br>${err.message}`);
  }
}

/**
 * render404
 * 專門處理 404 錯誤頁面
 *
 * @param {ServerResponse} res HTTP 回應物件
 */
export async function render404(res) {
```

```
  // 使用既有 renderTemplate，統一渲染流程
  await renderTemplate(res, "./index3.ejs", {}, 404);
}
```

第三部分：建立靜態文件處理模組（utils/staticFileHandler.js）

將靜態文件的處理邏輯抽取成獨立模組。 模組會負責讀取檔案、設定正確的 Content-Type，並在檔案不存在時自動顯示 404 錯誤頁面。 **程式碼（utils/staticFileHandler.js）**：

```
import fs from "fs/promises";
import path from "path";

// 引入 MIME 類型處理模組
import { getContentType } from "./mimeTypes.js";

// 引入 404 頁面渲染函式
import { render404 } from "./templateRenderer.js";

/**
 * handleStaticFile
 * 處理靜態文件請求
 *
 * @param {ServerResponse} res HTTP 回應物件
 * @param {string} filePath 靜態檔案實體路徑
 */
export async function handleStaticFile(res, filePath) {
  try {
    // 讀取靜態檔案（不指定編碼，支援圖片等二進位檔）
    const content = await fs.readFile(filePath);

    // 取得檔案副檔名
    const ext = path.extname(filePath);

    // 依副檔名取得正確的 Content-Type
    const contentType = getContentType(ext);

    // 回傳靜態檔案內容
    res.writeHead(200, { "Content-Type": contentType });
    res.end(content);
  } catch (err) {
    // 若檔案不存在，顯示 404 頁面
    if (err.code === "ENOENT") {
      await render404(res);
      return;
    }

    // 其他錯誤則回傳 500
    res.writeHead(500, { "Content-Type": "text/plain; charset=utf-8" });
    res.end("500 - 伺服器錯誤：讀取靜態檔案失敗");
  }
}
```

第四部分：重構主檔案（2b-refactored.js）

將主檔案重構為只負責路由判斷，並保留使用 switch 語句處理路由。 主檔案不再包含任何檔案讀取、模板渲染或 MIME 判斷的實作細節，而是交由前面完成的模組處理。 **程式碼（2b-refactored.js）**：

```
import http from "http";
import path from "path";
import { fileURLToPath } from "url";

// 引入模板渲染與靜態檔案處理模組
import { renderTemplate } from "./utils/templateRenderer.js";
import { handleStaticFile } from "./utils/staticFileHandler.js";

// ESM 環境下自行取得 __dirname
const __filename = fileURLToPath(import.meta.url);
const __dirname = path.dirname(__filename);

// 建立 HTTP 伺服器
http
  .createServer(async (req, res) => {
    // 只取路徑部分，忽略 query string
    const urlPath = (req.url || "/").split("?")[0];

    // 使用 switch 處理路由（題目要求）
    switch (urlPath) {
      case "/":
        // 根目錄 → 渲染 index.ejs
        await renderTemplate(res, path.join(__dirname, "index.ejs"), {
          message: "您好 xxx",
        });
        break;

      case "/calculator":
        // /calculator → 渲染 index2.ejs
        await renderTemplate(res, path.join(__dirname, "index2.ejs"));
        break;

      default:
        // 其他路徑 → 視為靜態檔案處理
        await handleStaticFile(res, path.join(__dirname, urlPath));
        break;
    }
  })
  .listen(3000, () => {
    console.log("伺服器已啟動：http://localhost:3000");
  });
```