**分散式系統  
Lab: RESTful Web Services**

請務必填寫學號系級姓名，以免成績登錄錯誤。

學號系級姓名:

請依問題與提示在指定區域回答問題，並依規定時間內上傳至moodle。

1. 使用您的開發工具，建立一個名為「restlab」的空專案新增一個package.json。

提示: 若您的開發工具不支援產生package.json，可透過命令列: npm init來產生。

1. 請在package.json中新增dependencies區段，其中加入下列libraries:

* "fastify": "^3.11.0"
* "node-fetch": "^2.6.1"

提示:

. . . ,

"dependencies": {

"fastify": "^3.11.0",

. . .

}, . . .

1. 執行npm install，此時系統會自動安裝上述libraries到node\_modules底下。
2. 新增一個檔案:restful-server.js，參考以下說明，完成一個簡單的RESTful Server。

|  |  |
| --- | --- |
| 部落中需要更多的支援而調動了許多野豬騎士。  我們需要存取目前野豬騎士(hogRider)的資訊，基於給定的樣板restfulServer.js，完成一個野豬騎士的RESTful Server。 | http://cdn2.ettoday.net/images/1573/1573541.jpg |

提示: 下面請依先後次序操作會比較好寫

操作1: 加入下列敘述匯入fastify函式庫

const server = require('fastify')();

操作2: 寫作野豬騎士儲存庫: 我們將以一個陣列(array)來儲存所有野豬騎士的資料，首先新增二名野豬騎士:john, tom，並將它們加入野豬騎士儲存庫陣列(hogRiders):

let john = {

name: "john",

age: 18,

attack: 100,

defense: 100

};

let tom = {

name: "tom",

age: 19,

attack: 105,

defense: 90

};

let hogRiders = [john, tom];

操作3: 完成以下功能:

|  |  |
| --- | --- |
| Method | 功能說明 |
| Get | 透過/hogRider/:name取得某位野豬騎士的資料  例如：   |  |  | | --- | --- | | Client送出 | GET /hogRider/john | | Server回應 | {  name : "john",  age : 18,  attack : 100,  defense : 100  } | | 提示: server.get(‘/hogRider’)與其測試用程式(test-GET.js)已經實作完成，請同學參考此實作完成接下來的部份。   1. 使用req.params.name可以取得:name的內容 2. 使用hogRiders的find方法取得hogRider陣列中name屬性為req.params.name的物件，存到result中:   let result = hogRiders.find(element => element.name === req.params.name);   1. (使用if-else) 檢查result的內容，如果result是truty，就回傳result，不然就回傳下列錯誤訊息:   {“error”:”not found”}   1. 執行restful-server.js (node restful-server.js) 2. 請參考test-GET.js程式實作一個新的test-GET-tom.js程式來測試<http://localhost:3000/hogRider/tom的結果>. 3. 請貼上test-GET-tom.js的程式   答:   1. 請貼上test-GET-tom.js執行後所印出的data內容   答:   1. 修改test-GET-tom.js程式，尋找一個不存在的人，例如: http://localhost:3000/hogRider/mary，測試看看是否輸出第3步驟的內容({“error”:”not found”})。如果不能正確輸出，代表restful-server.js中，有關本小題的程式碼有誤。 | | |
| Post | 依據上小題的範例，在server.post(‘/hogRider’, …)的內容，實作新增(POST)野豬騎士的功能，server回應目前騎士數量。  例：   |  |  | | --- | --- | | Client送出 | POST /hogRider  Body內容如下  {  name : "mary",  age : 17,  attack : 99,  defense : 99  } | | Server回應 | {count:3} | | 提示:   1. 使用req.body來取得新加入的騎士資料   let newRider=req.body;   1. 使用hogRiders.push(…)將取得的騎士資料加入儲存庫   hogRiders.push(newRider);   1. 使用下列方式回傳目前騎士個數   return {count: hogRiders.length};   1. 依給定的程式(如下)，寫作一個新的 test-POST.js程式來測試正確性。這個程式新增一個mary騎士。程式主體結構同上小題給的範例，以下只列出(async () => {…})()中的內容。   const resp = await client('http://localhost:3000/hogRider', {  method: 'POST',  headers: {  'Content-Type': 'application/json'  },  body: JSON.stringify({  name: "mary",  … (請依題目要求加上其它屬性)…  })  });  const data = await resp.json();  console.log(data);   1. 請在下面貼上test-POST.js的內容:   答:   1. 使用test-GET.js會向url http://localhost:3000/hogRider下達GET，可用來列出所有騎士資料，觀察mary是否順利新增。 | | |
| Put | 更新騎士資料   |  |  | | --- | --- | | Client送出 | PUT /hogRider/tom，會將tom的資料取代為body中的資料  Body如下  {  name : "tom",  age : 99,  attack : 0,  defense : 0  } | | Server回應 | (更新後的騎士資料)  {  name : "tom",  age : 99,  attack : 0,  defense : 0  } | |  | | |
| 提示:   1. 使用hogRiders的findIndex函式找到要更新的資料的索引，存在index變數中:   let index = hogRiders.findIndex(element => element.name === req.params.name);   1. 使用req.body取得新的騎士資料，並將新騎士資料更新到正確的陣列索引位置:   提示: hogRiders[index] = …   1. 回傳更新後的資料:   return hogRiders[index];   1. 寫作test-PUT.js來驗證結果，請在下面貼上test-PUT.js的內容:   答: | |

|  |
| --- |
| 請重啟restful-server.js，之後依序執行test-POST.js🡪test-PUT.js🡪test-GET.js將輸出結果貼在下方:  答: |

|  |
| --- |
| 將您的restful-server.js中的所有**程式碼**貼在下方:  答: |