

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: 112753138 資碩工一 周宸宇

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。

2. 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",  
  ...  
},...
```

3. 執行 npm install，此時系統會自動安裝上述 libraries 到 node_modules 底下。
4. 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。



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);`
3. (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"}`
4. 執行 `restful-server.js` (`node restful-server.js`)
5. 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.
6. 請貼上 `test-GET-tom.js` 的程式

答:

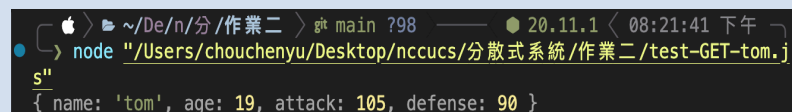
```
const client = require('node-fetch');

(async () => {
  const resp = await
client('http://localhost:3000/hogRider/tom', {
  method: 'GET'
});

  const data = await resp.json();
  console.log(data);
})();
```

7. 請貼上 `test-GET-tom.js` 執行後所印出的 `data` 內容

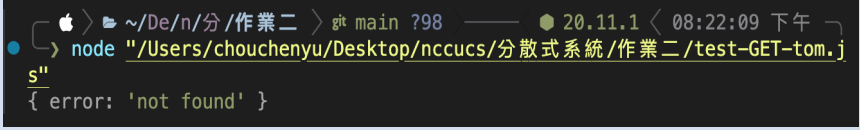
答:



```
node "/Users/chouchenyu/Desktop/nccucs/分散式系統/作業二/test-GET-tom.js"
{ name: 'tom', age: 19, attack: 105, defense: 90 }
```

8. 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。

//修改為

	<pre>const resp = await client('http://localhost:3000/hogRider/tomm'</pre> 				
Post	<p>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。</p> <p>例：</p> <table border="1"><tr><td>Client 送出</td><td>POST /hogRider Body 內容如下 { name : "mary", age : 17, attack : 99, defense : 99 }</td></tr><tr><td>Server 回應</td><td>{count:3}</td></tr></table> <p>提示:</p> <ol style="list-style-type: none">1. 使用 req.body 來取得新加入的騎士資料 let newRider=req.body;2. 使用 hogRiders.push(...)將取得的騎士資料加入儲存庫 hogRiders.push(newRider);3. 使用下列方式回傳目前騎士個數 return {count: hogRiders.length};4. 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。 <pre>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) }) });</pre>	Client 送出	POST /hogRider Body 內容如下 { name : "mary", age : 17, attack : 99, defense : 99 }	Server 回應	{count:3}
Client 送出	POST /hogRider Body 內容如下 { name : "mary", age : 17, attack : 99, defense : 99 }				
Server 回應	{count:3}				

```
const data = await resp.json();
console.log(data);
```

5. 請在下面貼上 test-POST.js 的內容:

答:

```
const client = require('node-fetch');

(async () => {
  const hogRiderData = {
    name: "mary",
    age: 17,
    attack: 99,
    defense: 99
  };

  const resp = await
client('http://localhost:3000/hogRider', {
  method: 'POST',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify(hogRiderData)
});

  const data = await resp.json();
  console.log(data);
})();
```

6. 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。

```

[
  {
    name: 'john', age: 18, attack: 100, defense: 100 },
    {
      name: 'tom', age: 19, attack: 105, defense: 90 },
    {
      name: 'mary', age: 17, attack: 99, defense: 99 }
  ]
}
```

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);`
2. 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] = ...`
3. 回傳更新後的資料:
`return hogRiders[index];`
4. 寫作 `test-PUT.js` 來驗證結果，請在下面貼上 `test-PUT.js` 的內容:

答:

```
const client = require('node-fetch');

(async () => {
  const hogRiderData = {
    name : "tom",
    age : 99,
    attack : 0,
    defense : 0
  };

  const resp = await client('http://localhost:3000/hogRider/tom', {
```

```

    method: 'PUT',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify(hogRiderData)
  });

  const data = await resp.json();
  console.log(data);
})();

```

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:

```

❯ node "/Users/chouchenyu/Desktop/nccucs/分散式系統/作業二/test-POST.js"
{ count: 3 }

❯ node "/Users/chouchenyu/Desktop/nccucs/分散式系統/作業二/test-PUT.js"
{ name: 'tom', age: 99, attack: 0, defense: 0 }

❯ node "/Users/chouchenyu/Desktop/nccucs/分散式系統/作業二/test-GET.js"
[
  { name: 'john', age: 18, attack: 100, defense: 100 },
  { name: 'tom', age: 99, attack: 0, defense: 0 },
  { name: 'mary', age: 17, attack: 99, defense: 99 }
]

```

將您的 restful-server.js 中的所有程式碼貼在下方:

答:

```

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

let john = {
  name: "john",
  age: 18,
  attack: 100,
  defense: 100
};

```

```
let tom = {
  name: "tom",
  age: 19,
  attack: 105,
  defense: 90
};

let hogRiders = [john, tom];

server.get('/hogRider', function (req, res) {
  return hogRiders;
});

server.get('/hogRider/:name', function (req, res) {
  let result = hogRiders.find(element => element.name ===
req.params.name);
  if (result) {
    res.send(result);
  } else {
    res.status(404).send({"error": "not found"});
  }
});

server.post('/hogRider', function (req, res) {
  let newHogRider = req.body;
  hogRiders.push(newHogRider);
  return {count: hogRiders.length}; // 回應目前 Hog Riders 的總數
});

server.put('/hogRider/:name', function (req, res) {
  let index = hogRiders.findIndex(element => element.name ===
req.params.name);
  if (index !== -1) {
    hogRiders[index] = req.body;
    res.send(hogRiders[index]);
  } else {
    res.status(404).send({"error": "not found"});
  }
}
```



```
});  
  
server.listen(3000, (err, address) => {  
  if (err) {  
    console.error(err);  
    process.exit(1);  
  }  
  console.log(`Server listening on ${address}`);  
});
```