

第二十章

前端後端整合

SERVER課程概述

SERVER課程概述

- 本次關於Server部分的課程，包含以下幾個步驟
 - 使用Node.js建立Server API
 - 建立並使用Mongo DB
 - 讓Node.js的Server與MongoDB連接
 - 讓Server API回傳MongoDB的內容

NODE.JS SERVER

範例說明

- 本範例使用Node.js為Server，MongoDB為資料庫
- 常見搭配有
 - Apache + PHP + MySQL
 - TOMCAT + JSP + MySQL
 - IIS + ASP.NET + SQLServer

什麼是NODE.JS

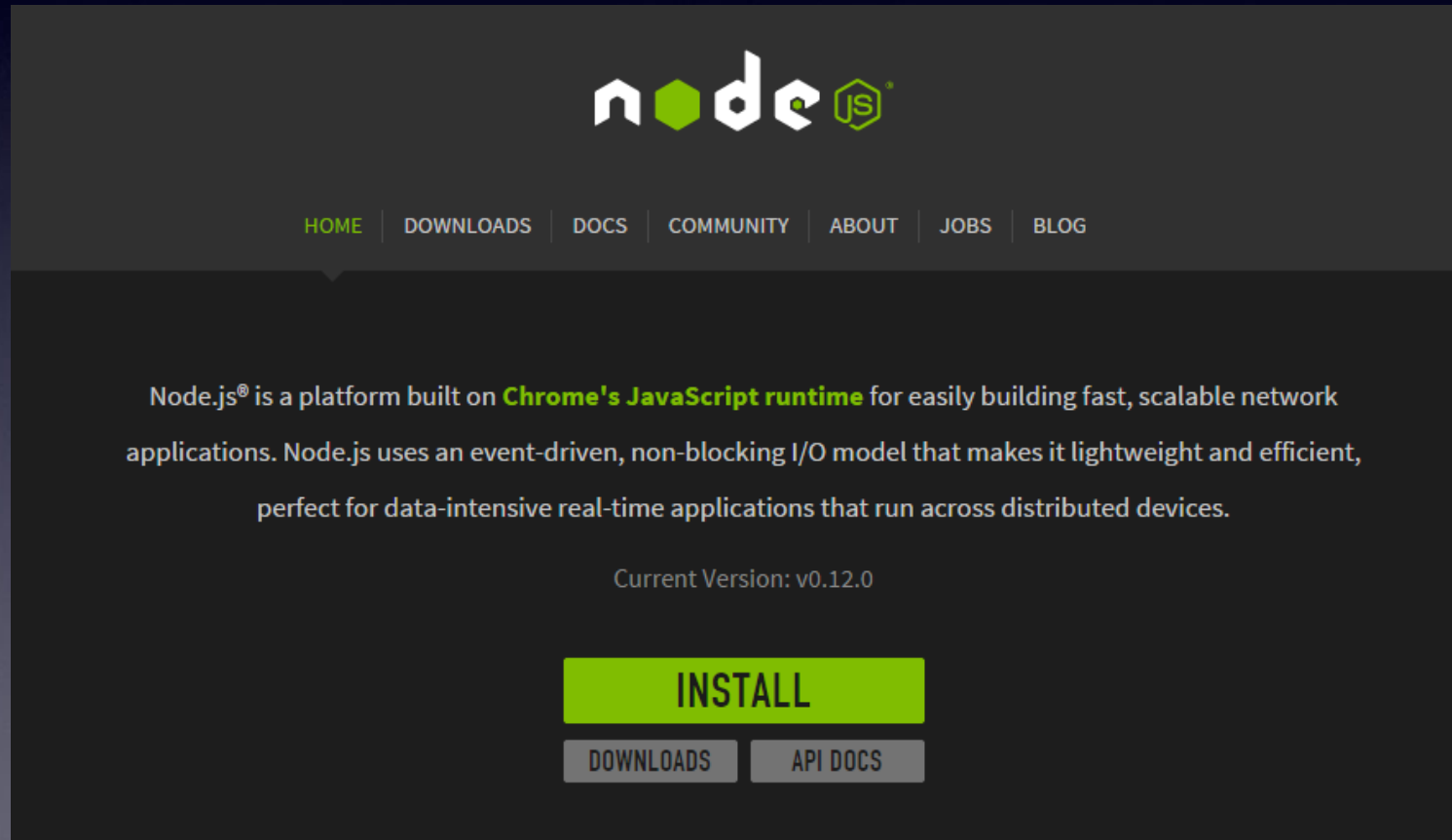
- Node.js是2009年推出
- 使用JavaScript撰寫Server後端
- 基於Google推出的v8引擎
- 為單一執行緒，環境配置容易
- 事件驅動，且使用**非同步方式**撰寫程式
- 只需要安裝Node.js，使用筆記本寫程式 (<http://nodejs.org/>)

簡單建立一個WebServer的範例

```
var express = require('express');
var app = express();
app.get('/', function(req, res){
  res.end('<html><body><H1>Hello World</H1></body></html>');
});
app.listen(5000);
```

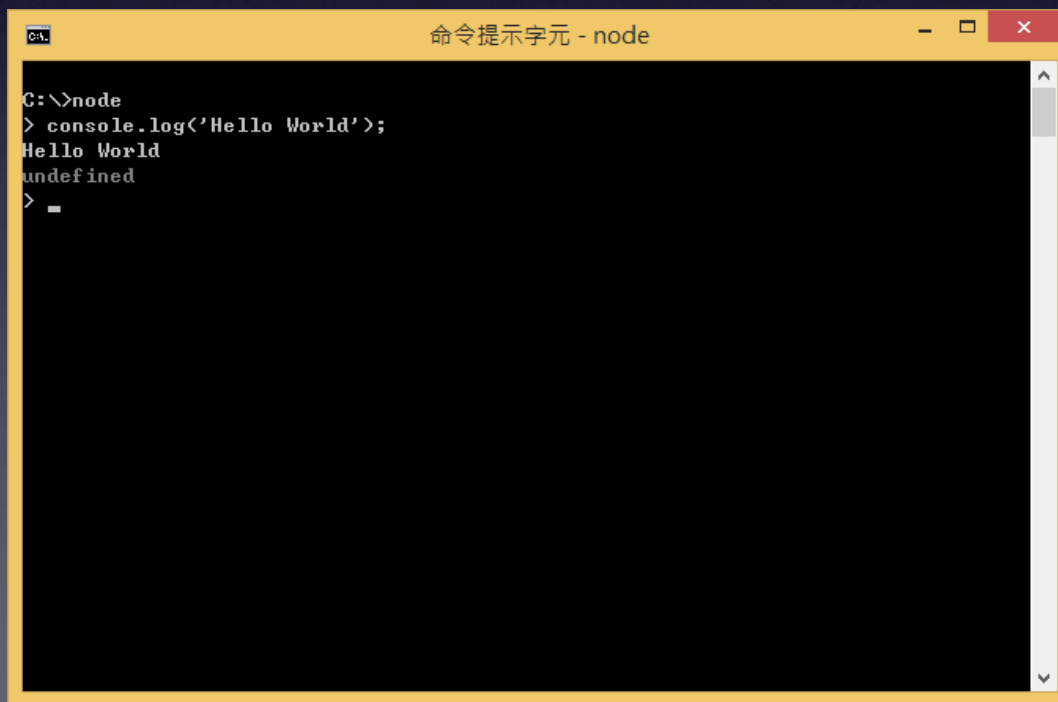
安裝NODE.JS

- 連線至<http://nodejs.org>，選擇Install
- 接下來會下載安裝檔，下載後執行



安裝NODE.JS

- 當安裝完畢，打開「命令提示字元」
- 鍵入指令node，按下enter，接著輸入
 - `console.log('Hello World');`
- 如果出現如下畫面，表示安裝成功，按CTRL+C兩次退出



```
C:\>node
> console.log('Hello World');
Hello World
undefined
> _
```


NODE.JS專案建置與設定

NODE.JS專案建置與設定



NODE.JS專案建置

對應專案
simple_api_server_step1

- 建立資料夾名叫simple_api_server
- 開啟筆記本，建立一個新檔案叫做**package.json**存在simple_api_server資料夾內
- 打入以下內容

```
{  
  "name"      : "server",  
  "version"   : "0.0.1",  
  "private"   : true,  
  "dependencies" : {  
    "express" : "*"   
  }  
}
```

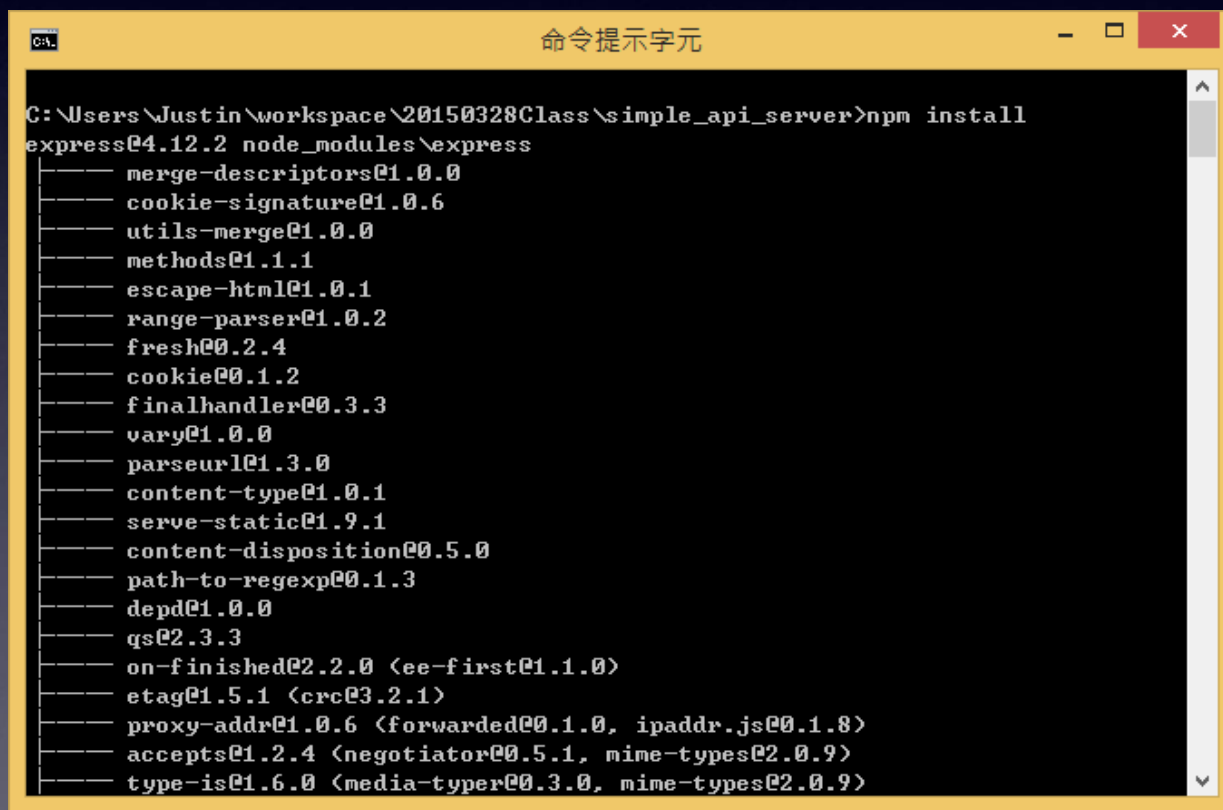
NODE.JS專案建置

```
{
  "name"      : "server",
  "version"   : "0.0.1",
  "private"   : true,
  "dependencies" : {
    "express" : "*"
  }
}
```

- package.json是每個node.js專案的設定檔
位於專案資料夾的根目錄
 - name 專案名稱
 - version 專案版本
 - private 是否為私人專案
 - **dependencies** 定義專案其他用到的函式庫(模組)

NODE.JS專案建置

- 接下來打開命令提示字元，輸入 **npm install**
- 本指令將會掃描package.json，安裝需用的函式庫

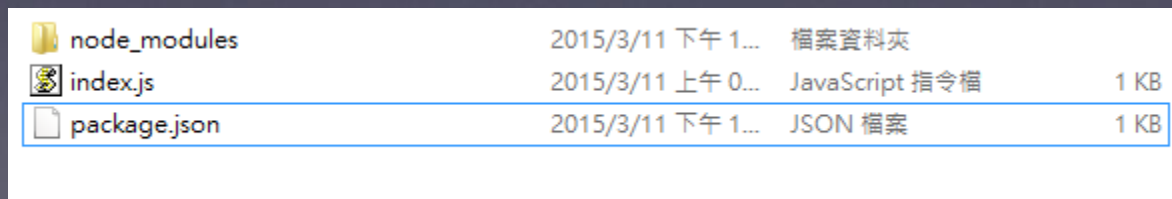





The screenshot shows a Windows Command Prompt window titled "命令提示字元". The command prompt shows the execution of `npm install` in the directory `C:\Users\Justin\workspace\20150328Class\simple_api_server`. The output lists the installed packages and their versions, including `express@4.12.2` and its dependencies.

```
C:\Users\Justin\workspace\20150328Class\simple_api_server>npm install
express@4.12.2 node_modules\express
--- merge-descriptors@1.0.0
--- cookie-signature@1.0.6
--- utils-merge@1.0.0
--- methods@1.1.1
--- escape-html@1.0.1
--- range-parser@1.0.2
--- fresh@0.2.4
--- cookie@0.1.2
--- finalhandler@0.3.3
--- vary@1.0.0
--- parseurl@1.3.0
--- content-type@1.0.1
--- serve-static@1.9.1
--- content-disposition@0.5.0
--- path-to-regexp@0.1.3
--- depd@1.0.0
--- qs@2.3.3
--- on-finished@2.2.0 <ee-first@1.1.0>
--- etag@1.5.1 <crc@3.2.1>
--- proxy-addr@1.0.6 <forwarded@0.1.0, ipaddr.js@0.1.8>
--- accepts@1.2.4 <negotiator@0.5.1, mime-types@2.0.9>
--- type-is@1.6.0 <media-typer@0.3.0, mime-types@2.0.9>
```

NODE.JS專案建置

- 指令執行完畢後，在**專案資料夾**中自動就會出現node_modules的資料夾
- 專案資料夾**下的node_modules資料夾為專案**預設**會去找尋函式庫(模組)的位置
- 隨著開發階段，可以繼續在package.json中的dependencies加入新的函式庫(模組)
- 加入後只要在**專案資料夾下鍵入指令npm install**node.js就會檢查目前專案模組狀態並安裝缺少模組



 node_modules	2015/3/11 下午 1...	檔案資料夾	
 index.js	2015/3/11 上午 0...	JavaScript 指令檔	1 KB
 package.json	2015/3/11 下午 1...	JSON 檔案	1 KB

NODE.JS伺服器程式

NODE.JS伺服器

對應專案
simple_api_server_step1

- 打開筆記本，輸入以下內容，並存檔在專案資料夾下，檔名為index.js

```
var express = require('express');
var app = express();

app.get('/', function(request, response) {
    response.status(200).send(' <html><body><H1>Hello World</H1></body></html> ');
    response.end();
});

app.get('/api/test', function(request, response) {
    var ret = {
        msg : 'Hello World',
        status : 0
    }
    response.status(200).send(JSON.stringify(ret));
    response.end();
});

app.listen(5000);
```


NODE.JS伺服器

```
var express = require('express');  
var app = express();
```

使用模組express

```
app.get('/', function(request, response) {  
    response.status(200).send(' <html><body><H1>Hello World</H1></body></html> ');  
    response.end();  
});
```

```
app.get('/api/test', function(request, response) {  
    var ret = {  
        msg : 'Hello World',  
        status : 0  
    }  
    response.status(200).send(JSON.stringify(ret));  
    response.end();  
});
```

```
app.listen(5000);
```

NODE.JS伺服器

```
var express = require('express');  
var app = express();
```

建立express實體

```
app.get('/', function(request, response) {  
    response.status(200).send(' <html><body><H1>Hello World</H1></body></html> ');  
    response.end();  
});  
  
app.get('/api/test', function(request, response) {  
    var ret = {  
        msg : 'Hello World',  
        status : 0  
    }  
    response.status(200).send(JSON.stringify(ret));  
    response.end();  
});  
  
app.listen(5000);
```

NODE.JS伺服器

```
var express = require('express');
var app = express();

app.get('/', function(request, response) {
    response.status(200).send(' <html><body><H1>Hello World</H1></body></html> ');
    response.end();
});

app.get('/api/test', function(request, response) {
    var ret = {
        msg : 'Hello World',
        status : 0
    }
    response.status(200).send(JSON.stringify(ret));
    response.end();
});

app.listen(5000);
```

啟動伺服器，聆聽port 5000

NODE.JS伺服器

若request的網址是主機
要處理的邏輯

```
var express = require('express');  
var app = express();
```

```
app.get('/', function(request, response) {  
    response.status(200).send(' <html><body><H1>Hello World</H1></body></html> ');  
    response.end();  
});
```

```
app.get('/api/test', function(request, response) {  
    var ret = {  
        msg : 'Hello World',  
        status : 0  
    }  
    response.status(200).send(JSON.stringify(ret));  
    response.end();  
});
```

```
app.listen(5000);
```

NODE.JS伺服器

```
var express = require('express');
var app = express();

app.get('/', function(request, response) {
    response.status(200).send(' <html><body><H1>Hello World</H1></body></html> ');
    response.end();
});

app.get('/api/test', function(request, response) {
    var ret = {
        msg : 'Hello World',
        status : 0
    }
    response.status(200).send(JSON.stringify(ret));
    response.end();
});

app.listen(5000);
```

若request的網址是主機/api/test
要處理的邏輯

NODE.JS伺服器

```
var express = require('express');
var app = express();

app.get('/', function(request, response) {
    response.status(200).send('<html><body><H1>Hello World</H1></body></html>');
    response.end();
});

app.get('/api/test', function(request, response) {
    var ret = {
        msg : 'Hello World',
        status : 0
    }
    response.status(200).send(JSON.stringify(ret));
    response.end();
});

app.listen(5000);
```

回覆request
html內容

NODE.JS伺服器

```
var express = require('express');
var app = express();

app.get('/', function(request, response) {
    response.status(200).send(' <html><body><H1>Hello World</H1></body></html> ');
    response.end();
});

app.get('/api/test', function(request, response) {
    var ret = {
        msg : 'Hello World',
        status : 0
    }
    response.status(200).send(JSON.stringify(ret));
    response.end();
});

app.listen(5000);
```

回覆request json
內容

啟動SERVER

啟動SERVER

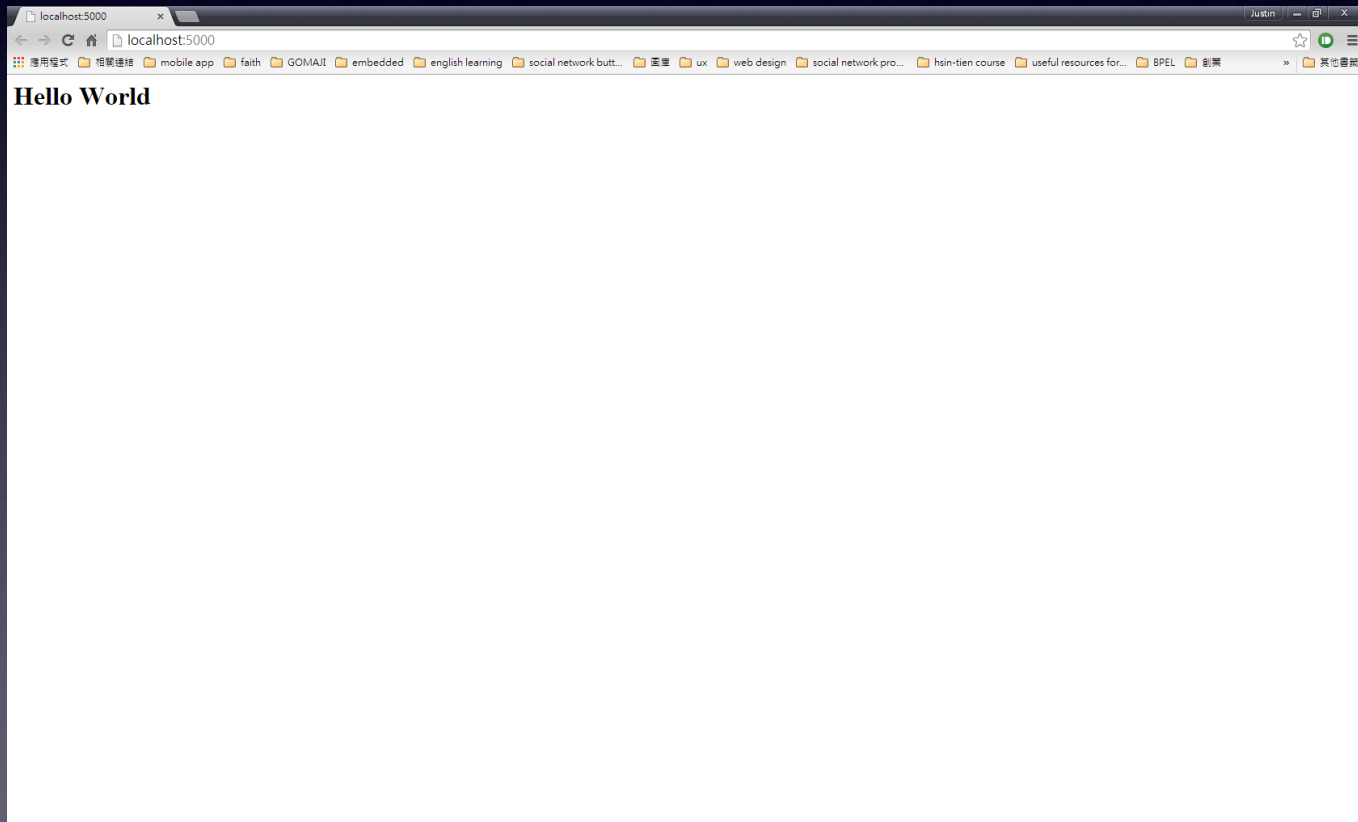
- 打開命令提示字元，切換到專案資料夾下
- 輸入指令`node index.js`
- 指令輸入後畫面會停下，只剩游標在閃



```
命令提示字元 - node index.js
C:\Users\Justin\workspace\20150328Class\simple_api_server>node index.js
```

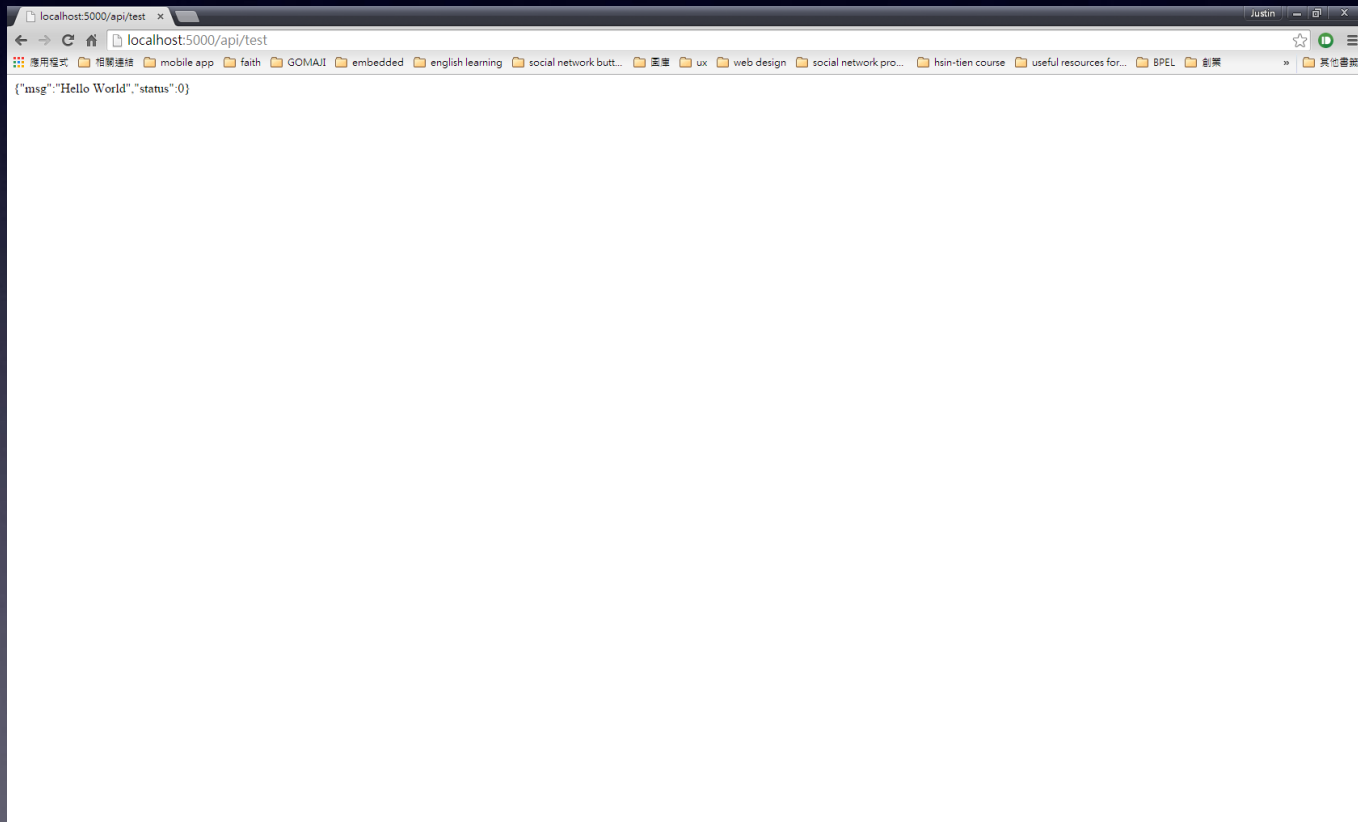
啟動SERVER

- 打開瀏覽器，在網址列輸入http://localhost:5000
- 將會看到剛剛程式撰寫的HTML



啟動SERVER

- 打開瀏覽器，輸入網址<http://localhost:5000/api/test>將會看到剛剛程式回覆的JSON



作業3

- 新增一個Server(你可以使用別的Server語言)，包含三個API
 - 查詢
 - 使用GET
 - 暫時回傳Query is under construction
 - 新增
 - 使用GET，要能接3個參數，包含**title, desc, owner, time**
 - 暫時回傳Insert title=..., desc=..., owner=..., time=...
 - 上述...表示GET代入的參數
 - 刪除
 - 使用GET，要接受1個參數id
 - 暫時回傳Remove id=...
 - 上述...表示GET代入的參數
- 參考資料<http://fred-zone.blogspot.tw/2012/02/nodejs-express-getpost-request.html>

使用MONGODB

使用MONGODB



什麼是MONGODB

- NoSQL Database, Document-Oriented Database
- 本次使用免費服務MongoLab
- 完全使用JSON當作儲存的文件內容

RDBMS	MongoDB
Table	Collection
Row	Document
Column	Field
Primary Key	_id

什麼是MONGODB

- Node.js尋找MongoDB collection內容的範例

取得items collection

```
var items = db.collection('items');
```

```
items.find({'name' : 'Books'}).toArray(function (err, docs) {  
  console.log(JSON.stringify(docs));  
});
```

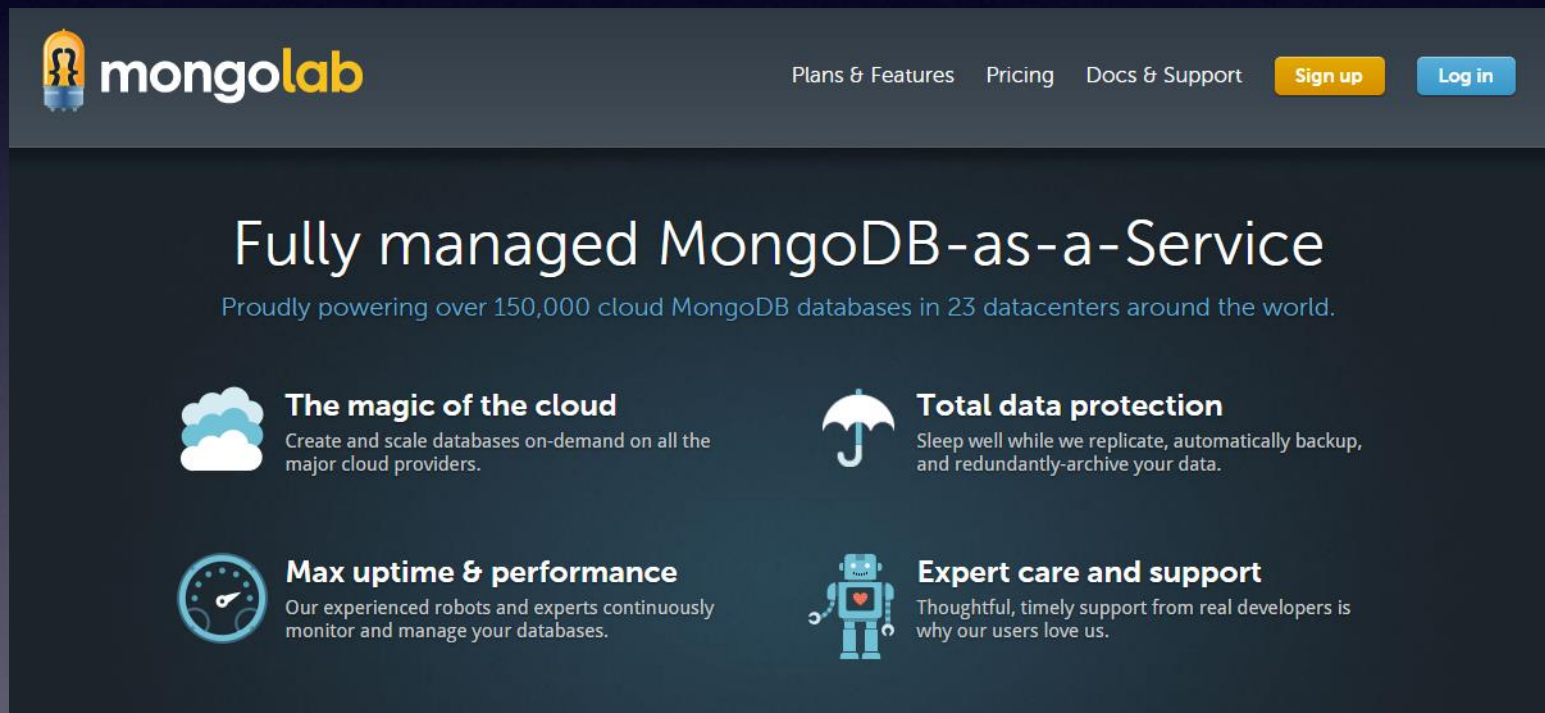
回傳的結果陣列

表示找尋collection中
Name為Books的資料

申請MONGODB帳號

申請MONGOLAB帳號

- 連線到<https://mongolab.com/>
- 點選Sign Up

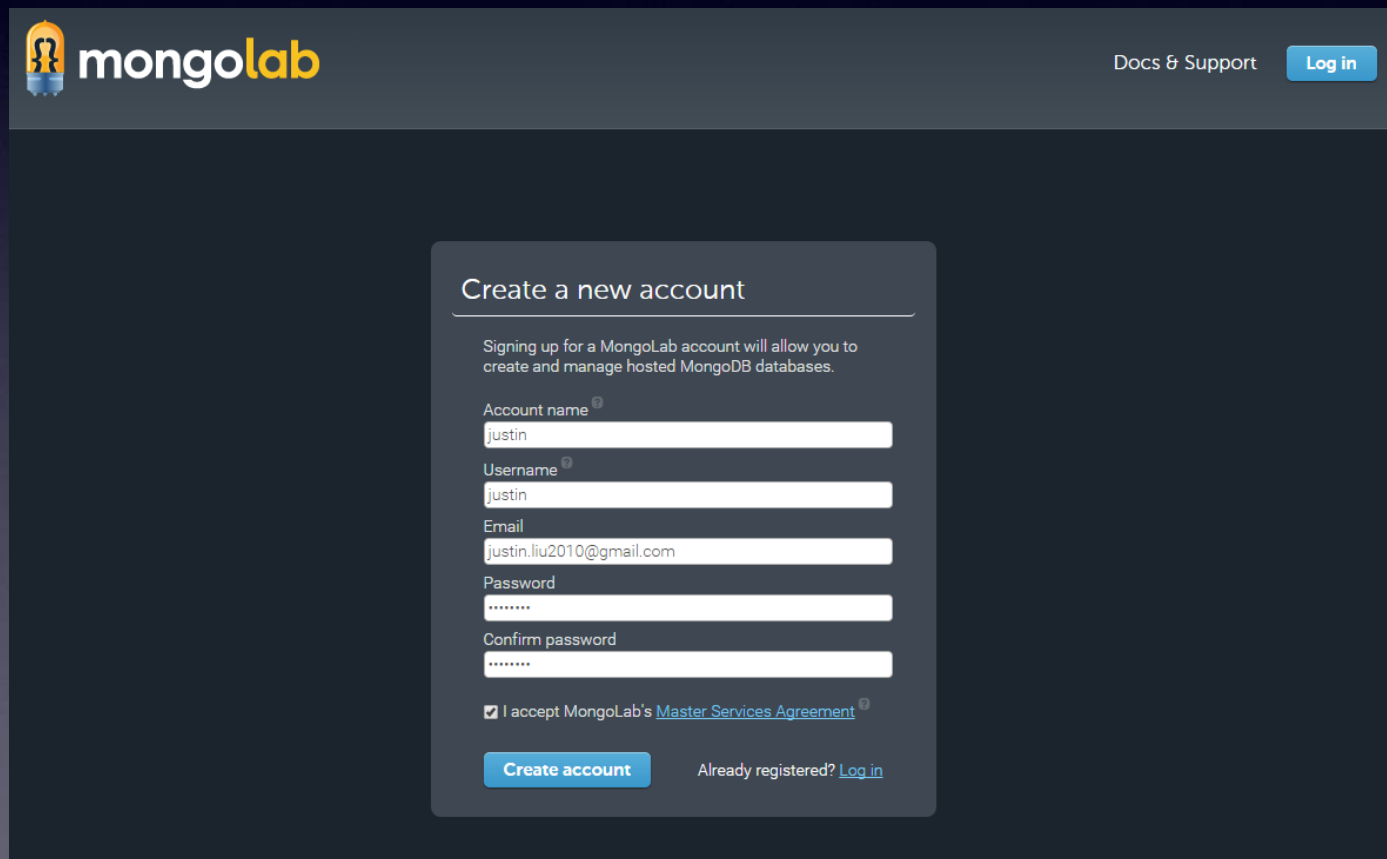


The screenshot shows the Mongolab website homepage. At the top is a dark navigation bar with the Mongolab logo on the left and links for 'Plans & Features', 'Pricing', 'Docs & Support', 'Sign up' (in a yellow button), and 'Log in' (in a blue button). The main content area has a dark background. The headline reads 'Fully managed MongoDB-as-a-Service' in large white text, followed by a sub-headline 'Proudly powering over 150,000 cloud MongoDB databases in 23 datacenters around the world.' Below this are four feature sections, each with an icon and text:


- The magic of the cloud** (cloud icon): Create and scale databases on-demand on all the major cloud providers.
- Total data protection** (umbrella icon): Sleep well while we replicate, automatically backup, and redundantly-archive your data.
- Max uptime & performance** (clock icon): Our experienced robots and experts continuously monitor and manage your databases.
- Expert care and support** (robot icon): Thoughtful, timely support from real developers is why our users love us.

申請MONGOLAB帳號

- 依序填好申請資料，按下Create account
- 接下來去申請時填寫的email做確認



The screenshot shows the MongoDB Lab website's account creation interface. At the top left is the MongoDB Lab logo, and at the top right are links for 'Docs & Support' and a 'Log in' button. The main content area is titled 'Create a new account' and includes a brief explanation of the account's purpose. Below this, there are input fields for 'Account name', 'Username', 'Email', 'Password', and 'Confirm password'. The 'Account name' and 'Username' fields contain the text 'justin', and the 'Email' field contains 'justin.liu2010@gmail.com'. The 'Password' and 'Confirm password' fields are masked with asterisks. A checkbox at the bottom indicates acceptance of the 'Master Services Agreement'. A 'Create account' button is positioned at the bottom left of the form, and a link for 'Already registered? Log in' is at the bottom right.

 mongolab

[Docs & Support](#) [Log in](#)

Create a new account

Signing up for a MongoLab account will allow you to create and manage hosted MongoDB databases.

Account name [?]
justin

Username [?]
justin

Email
justin.liu2010@gmail.com

Password

Confirm password

☒ I accept MongoLab's [Master Services Agreement](#) [?]

[Create account](#) Already registered? [Log in](#)

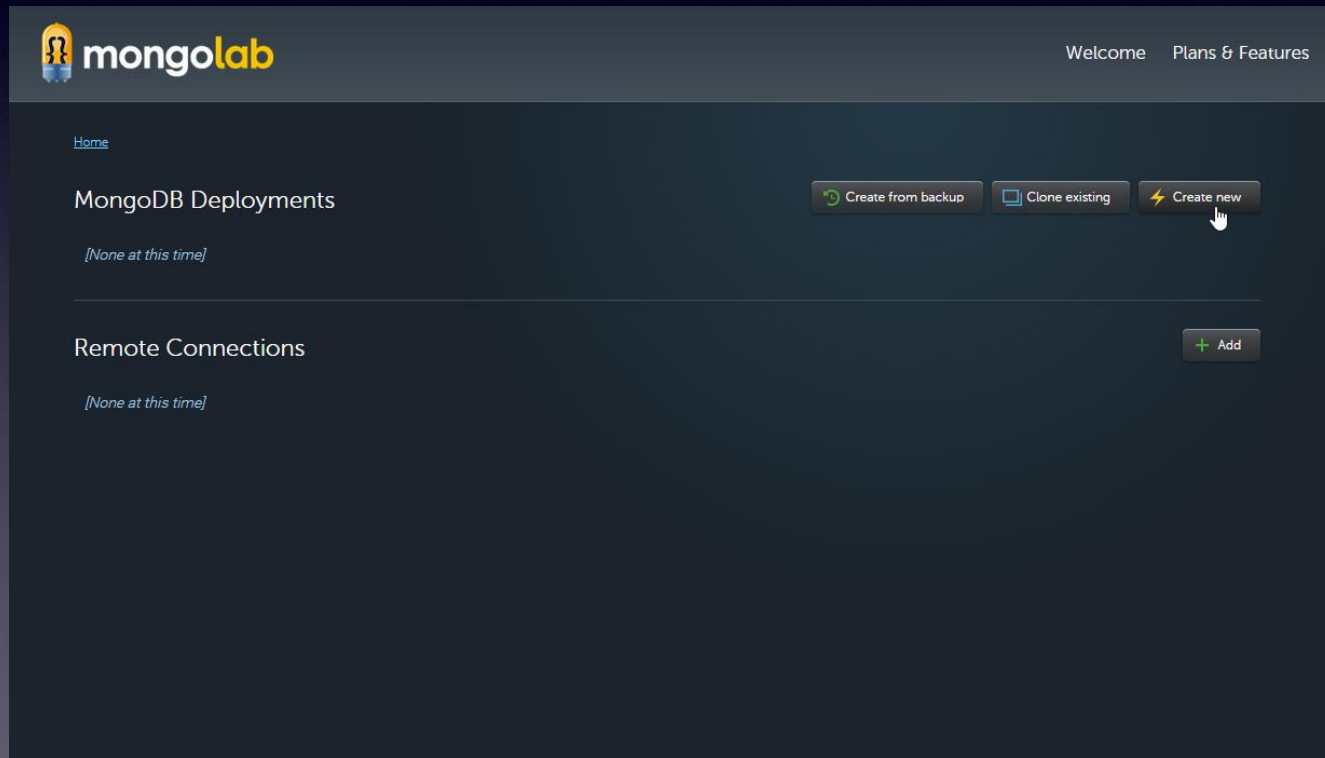
MONGODB資料建立流程

- 建立資料庫
 - 建立使用者
 - 建立Collection(資料表)
 - 建立資料
-
- 資料庫就像Excel檔
 - 資料庫 = Excel的檔案
 - Collection資料表 = 打開Excel檔案內的工作表
 - 資料 = Excel每筆紀錄

建立MONGO資料庫

建立MONGODB資料庫

- 登入MongoLab，按下create new，來建立資料庫



建立MONGODB資料庫

- 此處都保持預設值，「除了」Plan的部分選擇Single-node，且選擇FREE的方案

mongolab Welcome

[Home](#)

Create new subscription

Fill out this form to create a brand-new MongoDB deployment in the cloud location of your choice.
Alternatively, you can [clone an existing deployment](#).

Cloud provider:

Location: Amazon's US East (Virginia) Region (us-east-1)

Plan ([view pricing page](#)):

Single-node Replica set cluster

These plan(s) are perfect for development/testing/staging environments as well as for utility instances that do not require high-availability.

Standard Line

The most economical plans for production applications running on AWS. Plans come standard with 2 data nodes plus an arbiter node.

<input checked="" type="radio"/> Sandbox (shared, 0.5 GB)	FREE
<input type="radio"/> M3 Single-node (7.5 GB, 120 GB SSD block storage)	\$420
<input type="radio"/> M4 Single-node (15 GB, 240 GB SSD block storage)	\$835
<input type="radio"/> M5 Single-node (34.2 GB, 480 GB SSD block storage)	\$1310

建立MONGODB資料庫

- 頁面往下滑動，在Database name的地方寫上資料庫名稱
- 確認Price是\$0/month
- 最後按下最下方的create new MongoDB deployment

Standard Line

The most economical plans for production applications running on AWS. Plans come standard with 2 data nodes plus an arbiter node.

<input checked="" type="radio"/> Sandbox (shared, 0.5 GB)	FREE
<input type="radio"/> M3 Single-node (7.5 GB, 120 GB SSD block storage)	\$420
<input type="radio"/> M4 Single-node (15 GB, 240 GB SSD block storage)	\$835
<input type="radio"/> M5 Single-node (34.2 GB, 480 GB SSD block storage)	\$1310
<input type="radio"/> M6 Single-node (68.4 GB, 700 GB SSD block storage)	\$2045

High Storage Line

Plans which offer a higher storage-to-RAM ratio than those in the Standard line and are geared towards applications that need to store large amounts of data but have more modest performance requirements. Plans come standard with 2 data nodes plus an arbiter node.

MongoDB version: 2.6.x ▼

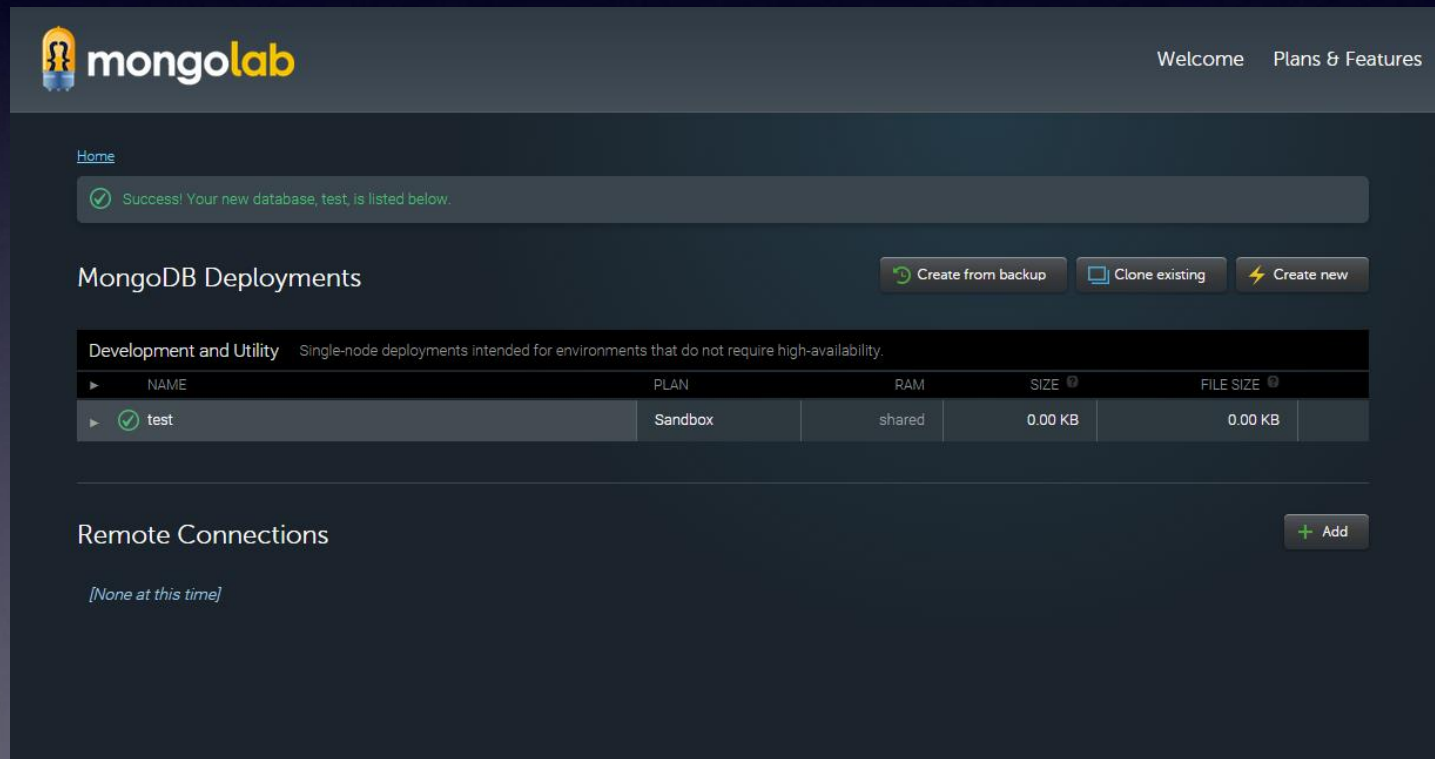
Database name:

Price:
\$0 / month

Create new MongoDB deployment

建立MONGODB資料庫

- 接下來頁面會出現你新建立的資料庫
- 可以點一下資料庫進入資料庫的儲存內容頁面



The screenshot shows the MongoDB Lab interface. At the top, the 'mongolab' logo is on the left, and 'Welcome Plans & Features' is on the right. Below the header, a green success message states: 'Success! Your new database, test, is listed below.' The main section is titled 'MongoDB Deployments' and includes three buttons: 'Create from backup', 'Clone existing', and 'Create new'. Below these buttons is a table for 'Development and Utility' deployments. The table has columns for NAME, PLAN, RAM, SIZE, and FILE SIZE. A single deployment named 'test' is listed with a 'Sandbox' plan, 'shared' RAM, and '0.00 KB' for both size and file size. At the bottom, there is a 'Remote Connections' section with an 'Add' button and the text '[None at this time]'.

Home

Success! Your new database, test, is listed below.

MongoDB Deployments

Create from backup Clone existing Create new

Development and Utility Single-node deployments intended for environments that do not require high-availability.

NAME	PLAN	RAM	SIZE	FILE SIZE
test	Sandbox	shared	0.00 KB	0.00 KB

Remote Connections

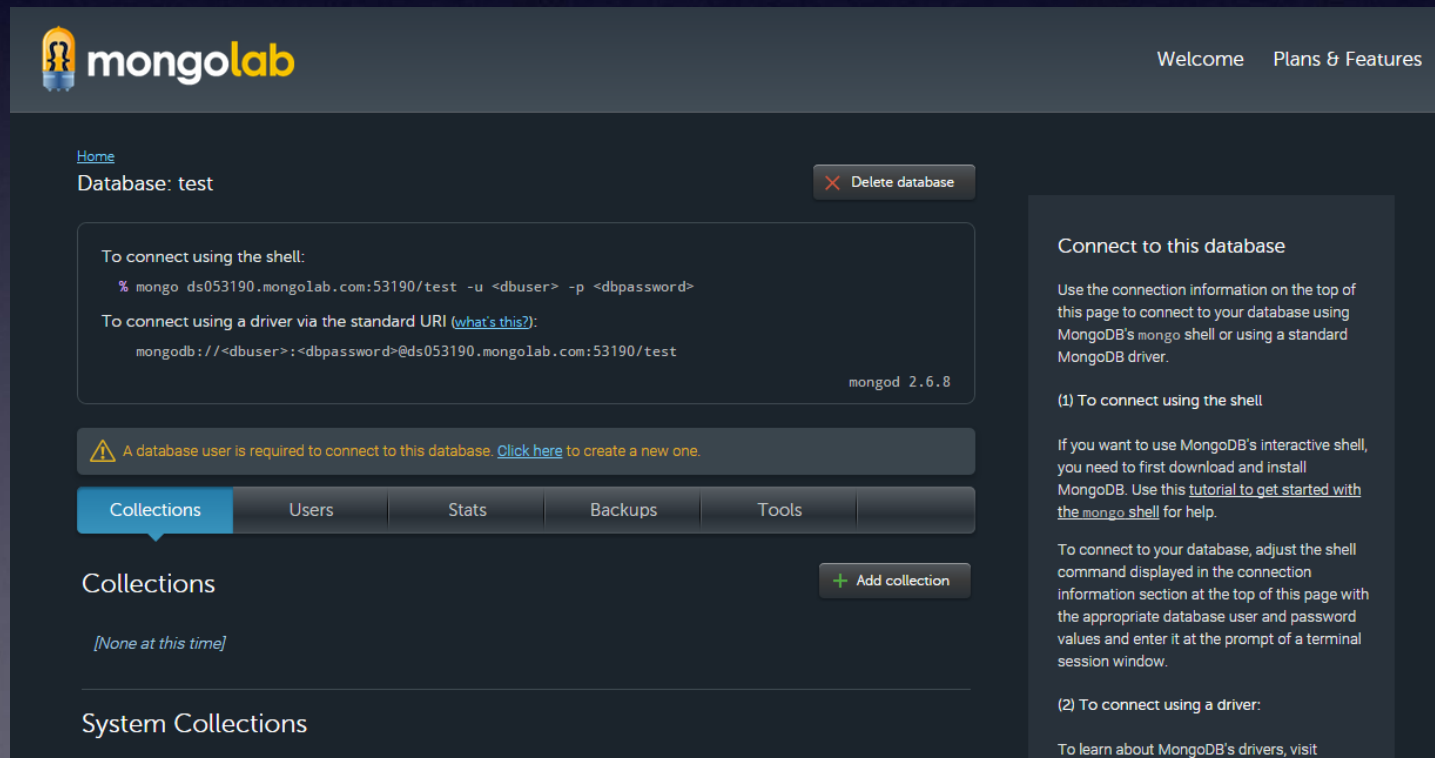
+ Add

[None at this time]

建立MONGODB資料庫使用者

建立MONGODB資料庫使用者

- **Delete database** 會刪除目前資料庫
- 按下 **Click here** 增加資料庫的使用者，以便我們之後寫程式能夠有權限來存取資料庫



The screenshot shows the Mongolab web interface for a database named 'test'. At the top, there's a 'Welcome' message and a link to 'Plans & Features'. Below the database name, there's a 'Delete database' button. A section titled 'To connect using the shell:' provides a command: `% mongo ds053190.mongolab.com:53190/test -u <dbuser> -p <dbpassword>`. Another section titled 'To connect using a driver via the standard URI (what's this?):' provides a URI: `mongodb://<dbuser>:<dbpassword>@ds053190.mongolab.com:53190/test`. A warning message states: 'A database user is required to connect to this database. Click here to create a new one.' Below this, there's a navigation bar with tabs for 'Collections', 'Users', 'Stats', 'Backups', and 'Tools'. The 'Collections' tab is active, showing a message '[None at this time]'. There's also an 'Add collection' button. On the right side, there's a 'Connect to this database' section with instructions on how to connect using the shell or a driver. The version 'mongod 2.6.8' is displayed at the bottom right of the connection information section.

建立MONGO資料庫使用者

- 在以下畫面輸入帳號、密碼(重複兩次)
- 不用勾選Make read-only，按下Create

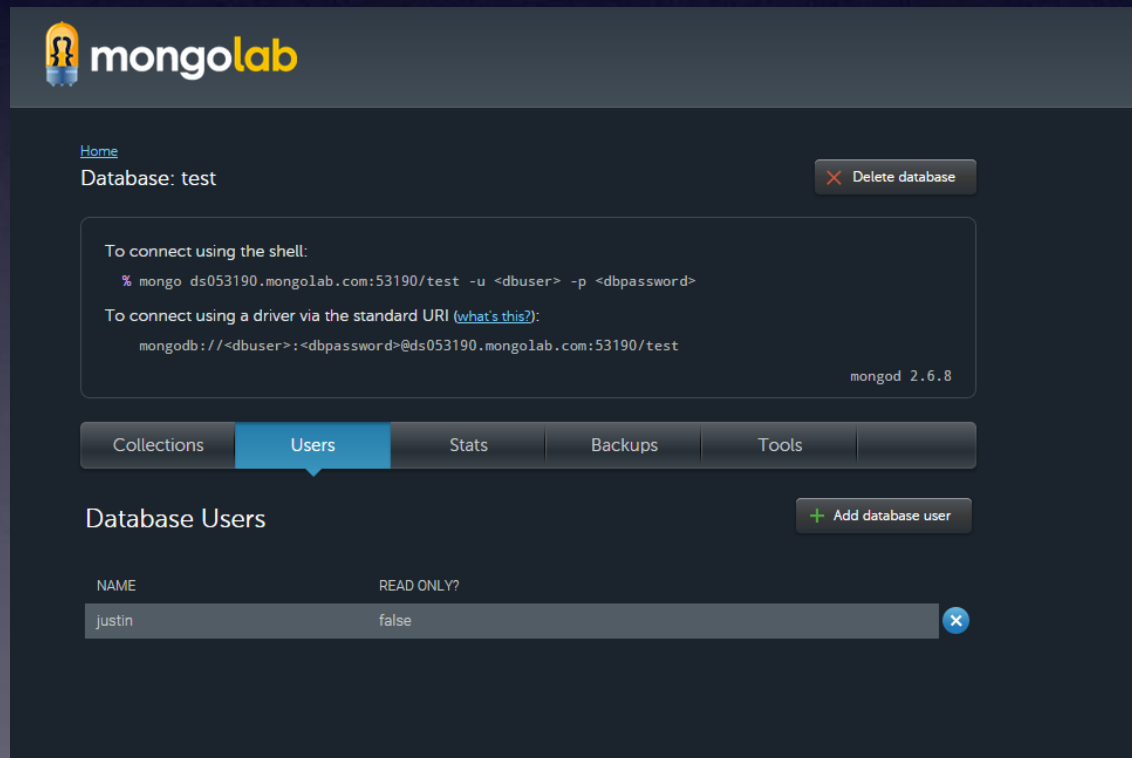
The image shows a screenshot of the MongoDB Atlas interface. In the foreground, a modal dialog box titled "Add new database user" is open. It contains the following fields and options:

- Database username***: A text input field containing the value "justin".
- Database password***: A password input field with masked characters (dots).
- Confirm password***: A second password input field with masked characters (dots).
- ☐ **Make read-only**: An unchecked checkbox.
- Buttons**: "Cancel" and "Create" buttons at the bottom.

The background shows the MongoDB Atlas console with a "Connect to this database" section and a "Backups" tab.

建立MONGO資料庫使用者

- 建立成功後會自動切換到Users的標籤頁
你可以看到剛剛建立的使用者
- 可以點選Add database user增加新的使用者



建立MONGODB資料表

建立MONGODB資料表

- 切換回Collections的標籤頁
- 按下Add collection增加collection (Table)

mongolab Welcome Plans & Features

Home Database: test ✕ Delete database

To connect using the shell:
`% mongo ds053190.mongolab.com:53190/test -u <dbuser> -p <dbpassword>`

To connect using a driver via the standard URI ([what's this?](#)):
`mongodb://<dbuser>:<dbpassword>@ds053190.mongolab.com:53190/test`

mongod 2.6.8

⚠ A database user is required to connect to this database. [Click here](#) to create a new one.

Collections Users Stats Backups Tools

Collections + Add collection

[None at this time]

System Collections

Connect to this database

Use the connection information on the top of this page to connect to your database using MongoDB's `mongo` shell or using a standard MongoDB driver.

(1) To connect using the shell

If you want to use MongoDB's interactive shell, you need to first download and install MongoDB. Use this [tutorial](#) to get started with the `mongo` shell for help.

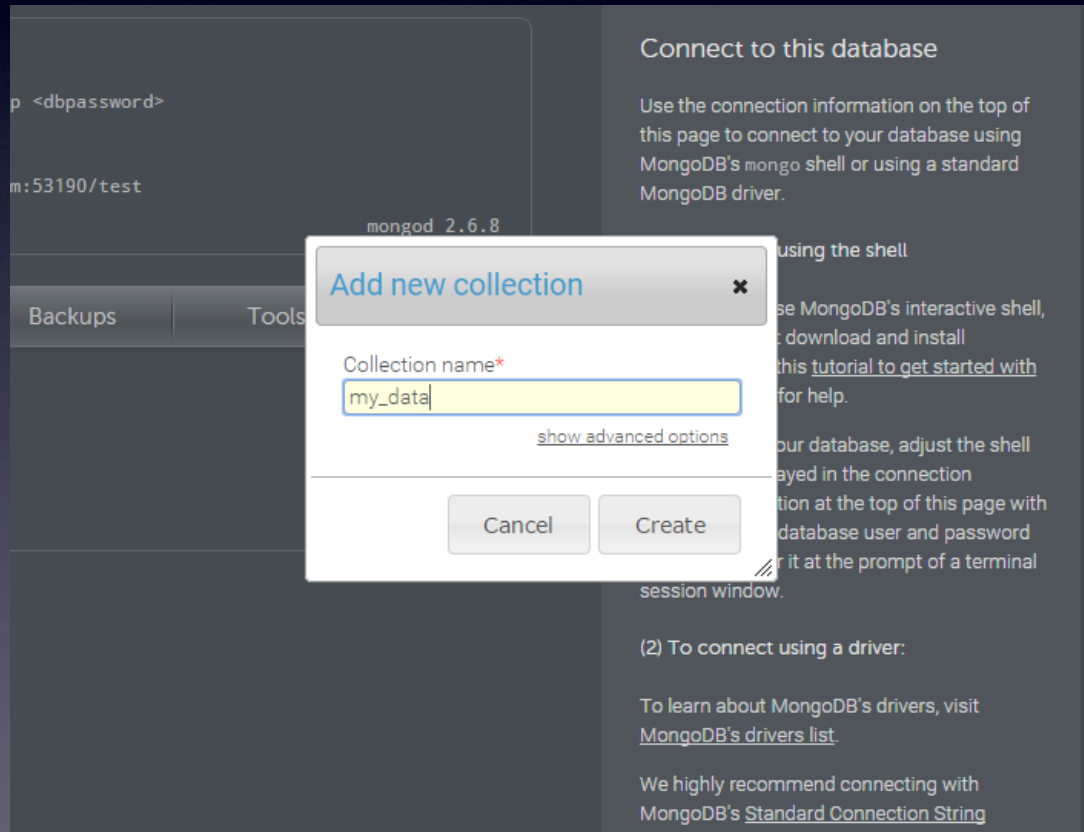
To connect to your database, adjust the shell command displayed in the connection information section at the top of this page with the appropriate database user and password values and enter it at the prompt of a terminal session window.

(2) To connect using a driver:

To learn about MongoDB's drivers, visit

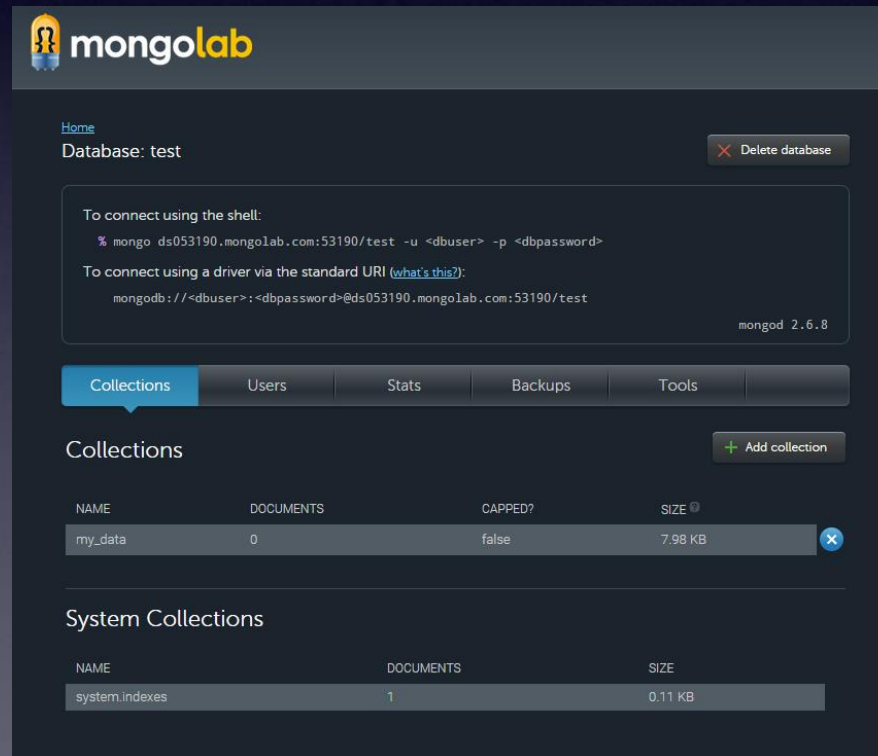
建立MONGODB資料表

- 在Collection name輸入資料表的名稱
- 輸入完畢後按下Create



建立MONGODB資料表

- Name顯示資料表的名稱
- Document顯示目前資料表中的資料數目
- 可以點選資料表進入Mongo資料內容頁



The screenshot shows the MongoDB Lab web interface. At the top, the 'mongolab' logo is visible. Below it, the 'Database: test' is selected, with a 'Delete database' button. A section for connecting to the shell or via a standard URI is provided. A navigation bar includes 'Collections', 'Users', 'Stats', 'Backups', and 'Tools'. The 'Collections' tab is active, showing a table with one collection: 'my_data' with 0 documents and a size of 7.98 KB. Below this, 'System Collections' are listed, including 'system.indexes' with 1 document and a size of 0.11 KB.

Database: test Delete database

To connect using the shell:
`% mongo ds053190.mongolab.com:53190/test -u <dbuser> -p <dbpassword>`

To connect using a driver via the standard URI ([what's this?](#)):
`mongodb://<dbuser>:<dbpassword>@ds053190.mongolab.com:53190/test`

mongod 2.6.8

Collections Add collection

NAME	DOCUMENTS	CAPPED?	SIZE
my_data	0	false	7.98 KB

System Collections

NAME	DOCUMENTS	SIZE
system.indexes	1	0.11 KB

建立MONGODB資料表的資料

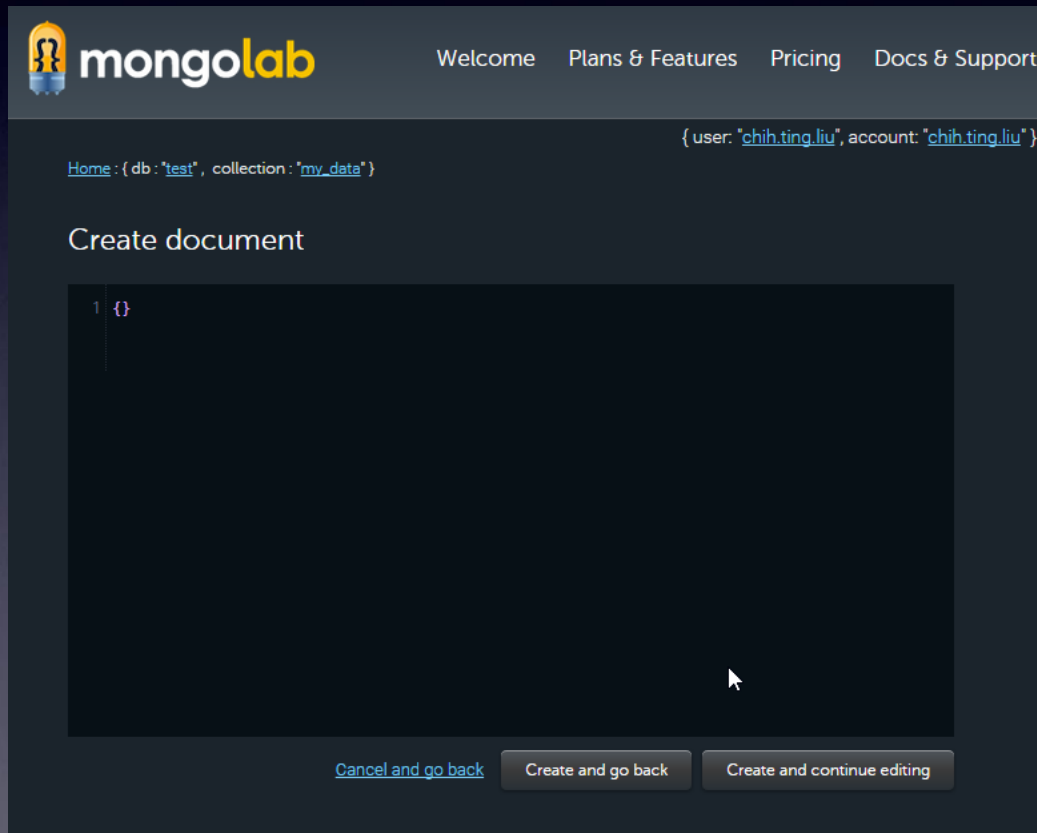
建立MONGODB資料表的資料

- Documents標籤頁會顯示目前資料表中的資料
- 按下Add document可以新增資料



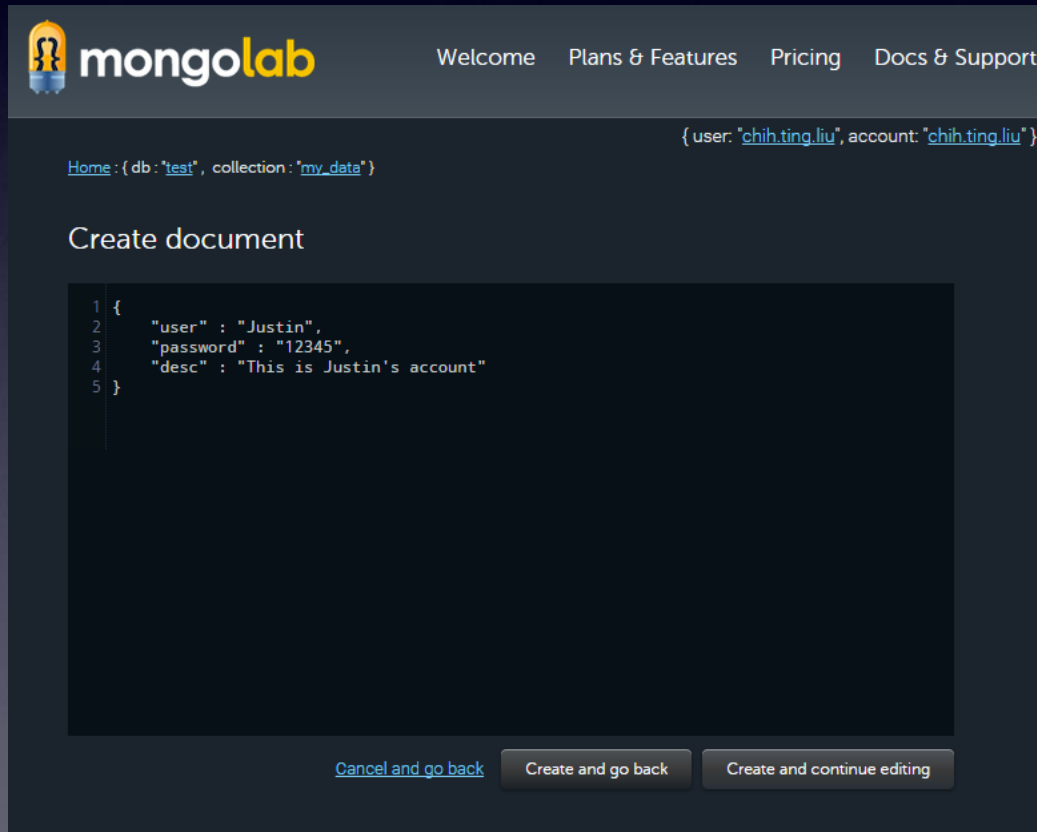
建立MONGODB資料表的資料

- MongoDB是接受JSON為資料儲存格式，所以可以在編輯區輸入資料



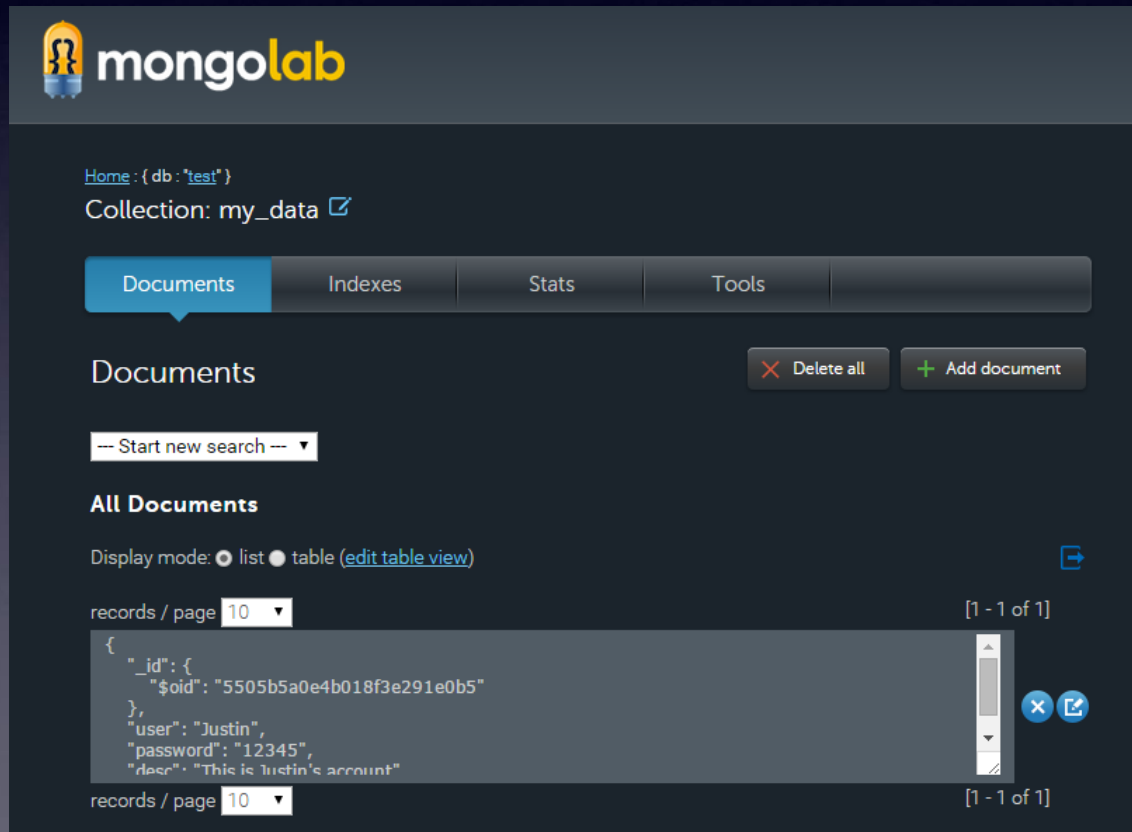
建立MONGODB資料表的資料

- 輸入完畢後可以按下 **Create and go back** 回到前頁
也可以按下 **Create and continue editing** 繼續編輯



建立MONGODB資料表的資料

- 在下方畫面可以看到剛剛新增的資料
- 資料內容會多一筆_id，那是MongoDB為資料自動加上的索引值，如同關聯式資料庫的主鍵

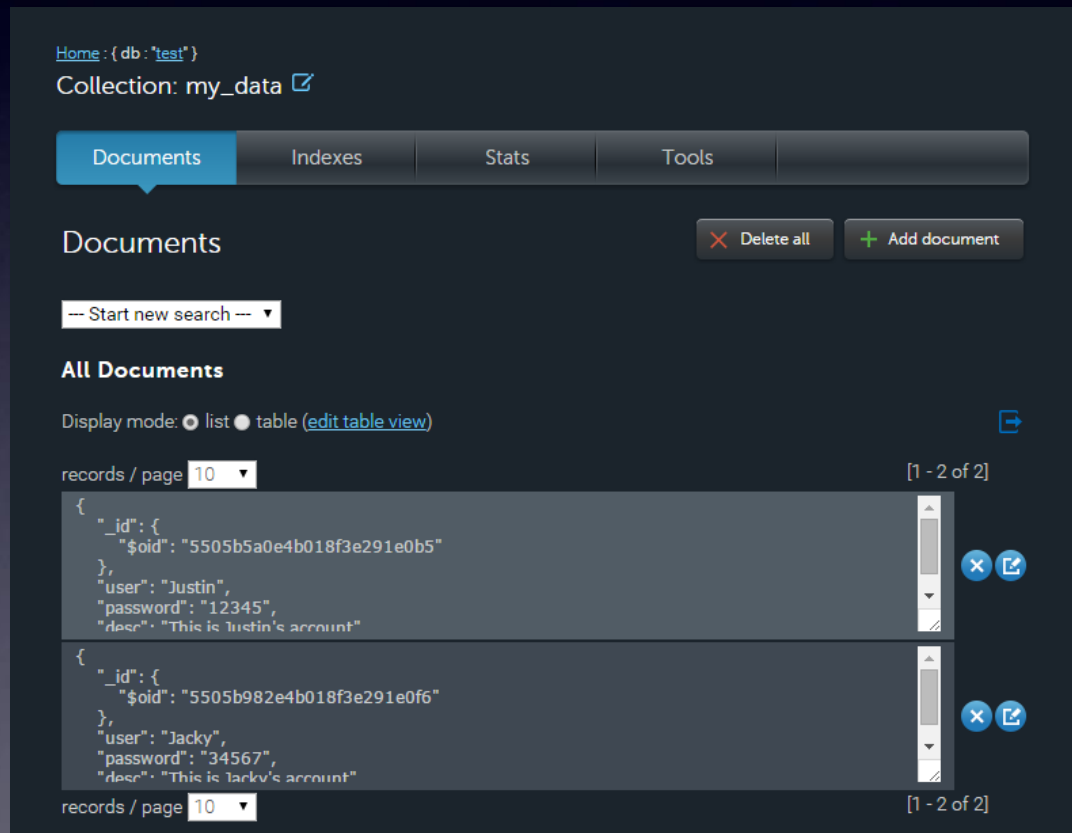


建立MONGODB資料表的資料

- MongoDB的資料表內的資料，_id不能重複
- 新增資料時，若不是很確定，不要寫_id
讓MongoDB在新增資料時自動產生
- MongoDB沒有規定每筆資料都需要有相同的格式
- 但為了管理方便，實作時仍然會給予存在同個資料表內的資料有統一的格式
- 記得新增資料時，不要自己寫上_id的內容，除非你很確定你在做什麼
- 現在請試著新增第二筆資料

建立MONGODB資料表的資料

- 現在新增好了第二筆資料，接下來要說明MongoDB的尋找資料



作業4

- 在MongoLab新增一個Database稱為todolist
- Collection稱為Todos
- 新增一個可以讀寫Collection的使用者
- 新增兩筆資料在其中，資料包含
 - 代辦事項名稱 (title)
 - 細節說明 (desc)
 - 代辦事項時間的秒數 (time)
 - 可以由這網址得到<http://www.epochconverter.com/>
 - 建立者 (owner)
- 資料請照上述格式，但可以隨便建立

連接MONGODB

連接MONGODB



連接MONGODB

對應專案
simple_api_server_step2

- 首先要取得MongoDB的位置，位於MongoDB網站Database頁面最上方
- `mongodb://<dbuser>:<dbpassword>@ds053190.mongolab.com:53190/test`

The screenshot shows the MongoDB Atlas web interface for a database named 'test'. At the top, there's a 'Home' link and a 'Delete database' button. Below this, a section titled 'To connect using the shell:' provides a command: `% mongo ds053190.mongolab.com:53190/test -u <dbuser> -p <dbpassword>`. Another section titled 'To connect using a driver via the standard URI (what's this?):' shows the connection string `mongodb://<dbuser>:<dbpassword>@ds053190.mongolab.com:53190/test` highlighted with a yellow box. Below these sections is a navigation bar with tabs for 'Collections', 'Users', 'Stats', 'Backups', and 'Tools'. The 'Collections' tab is selected, showing a table of collections. The table has columns for 'NAME', 'DOCUMENTS', 'CAPPED?', and 'SIZE'. There is one collection named 'my_data' with 2 documents, not capped, and a size of 8.20 KB. An 'Add collection' button is located to the right of the table.

Database: test Delete database

To connect using the shell:

```
% mongo ds053190.mongolab.com:53190/test -u <dbuser> -p <dbpassword>
```

To connect using a driver via the standard URI (what's this?):

```
mongodb://<dbuser>:<dbpassword>@ds053190.mongolab.com:53190/test
```

mongod 2.6.8

Collections Users Stats Backups Tools

Collections + Add collection

NAME	DOCUMENTS	CAPPED?	SIZE ?
my_data	2	false	8.20 KB

連接MONGODB

- `mongodb://<dbuser>:<dbpassword>@ds053190.mongolab.com:53190/test`
- <dbuser>填入建立Database時的帳號名稱
- <dbpassword>填入帳號的密碼
- 見章節「[建立Mongo資料庫使用者](#)」

連接MONGODB

- 與MongoDB相連接，Node.js需要用套件mongodb
- 所以修改serve的package.json，加入mongodb

```
{
  "name"      : "todo",
  "version"   : "0.0.1",
  "private"   : true,
  "dependencies" : {
    "express" : "*",
    "mongodb"  : "*"
  }
}
```

- 輸入完畢，記得使用npm install來安裝
見章節「[NODE.JS專案建置](#)」

連接MONGODB

```
var express = require('express');  
var mongodb = require('mongodb');  
var app = express();
```

使用模組mongodb

```
var mongodbURL =  
'mongodb://justin:justin12345@ds053190.mongolab.com:53190/test';  
  
mongodb.MongoClient.connect(mongodbURL, function(err, db) {  
    if (err) {  
        console.log(err);  
    } else {  
        console.log('connection success');  
    }  
});  
  
app.get('/', function(request, response) {  
    .....  
});
```

連接MONGODB

```
var express = require('express');  
var mongodb = require('mongodb');  
var app = express();
```

將MongoDB的位置在Server
程式碼中以一個變數儲存

```
var mongodbURL =  
'mongodb://justin:justin12345@ds053190.mongolab.com:53190/test';
```

```
mongodb.MongoClient.connect(mongodbURL, function(err, db) {  
  if (err) {  
    console.log(err);  
  } else {  
    console.log('connection success');  
  }  
});
```

```
app.get('/', function(request, response) {  
  .....  
});
```


連接MONGODB

```
var express = require('express');  
var mongodb = require('mongodb');  
var app = express();
```

```
var mongodbURL =  
'mongodb://justin:justin12345@ds053190.mongolab.com:53190/test';
```

```
mongodb.MongoClient.connect(mongodbURL, function(err, db) {  
  if (err) {  
    console.log(err);  
  } else {  
    console.log('connection successful');  
  }  
});
```

使用mongodb.MongoClient
的方法connect()進行連線

```
app.get('/', function(request, response) {  
  .....  
});
```

連接MONGODB

```
var express = require('express');  
var mongodb = require('mongodb');  
var app = express();
```

```
var mongodbURL =  
'mongodb://justin:justin12345@ds053190.mongolab.com:53190/test';
```

```
mongodb.MongoClient.connect(mongodbURL, function(err, response) {  
  if (err) {  
    console.log(err);  
  } else {  
    console.log('connection success');  
  }  
});
```

若回傳的參數有error，用
console.log()印出錯誤內容

若沒有錯誤表示連線成功，印出
connection success

搜尋MONGODB

搜尋MONGODB



搜尋MONGODB

對應專案
simple_api_server_step2

- 現在我們要讓原本的API `/api/test`回傳MongoDB的test中的my_data collection的資料，該怎麼做？
- 修改Server端程式碼

搜尋MONGODB

建立一個全域變數myDB

```
.....  
var myDB;  
mongodb.MongoClient.connect(mongodbURL, function(err, db) {  
  if (err) {  
    console.log(err);  
  } else {  
    myDB = db;  
    console.log('connection success');  
  }  
});
```

在mongoDB連線成功後，留住db物件

```
.....  
app.get('/api/test', function(request, response) {  
  var collection = myDB.collection('my_data');  
  collection.find({}).toArray(function(err, docs) {  
    if (err) {  
      response.status(406).end();  
    } else {  
      response.type('application/json');  
      response.status(200).send(docs);  
      response.end();  
    }  
  });  
});  
});
```

搜尋MONGODB

```
.....
var myDB;
mongodb.MongoClient.connect(mongodbURL, function(err, db) {
    if (err) {
        console.log(err);
    } else {
        myDB = db;
        console.log('connection success');
    }
});
.....
app.get('/api/test', function(request, response) {
    var collection = myDB.collection('my_data');
    collection.find({}).toArray(function(err, docs) {
        if (err) {
            response.status(406).end();
        } else {
            response.type('application/json');
            response.status(200).send(docs);
            response.end();
        }
    });
});
```

使用myDB的方法
collection('my_data')取得
my_data這個collection

搜尋MONGODB

```
.....
var myDB;
mongodb.MongoClient.connect(mongodbURL, function(err, db) {
    if (err) {
        console.log(err);
    } else {
        myDB = db;
        console.log('connection success');
    }
});

.....
app.get('/api/test', function(request, response) {
    var collection = myDB.collection('my_data');
    collection.find({}).toArray(function(err, docs) {
        if (err) {
            response.status(400).end();
        } else {
            response.type('application/json');
            response.status(200).send(docs);
            response.end();
        }
    });
});
```

使用collection的方法find()取得資料表內的內容，{}表示取得全部資料

搜尋MONGODB

```
.....
var myDB;
mongodb.MongoClient.connect(mongodbURL, function(err, db) {
    if (err) {
        console.log(err);
    } else {
        myDB = db;
        console.log('connection success');
    }
});
.....
app.get('/api/test', function(request, response) {
    var collection = myDB.collection('my_data');
    collection.find({}).toArray(function(err, docs) {
        if (err) {
            response.status(406).end();
        } else {
            response.type('application/json');
            response.status(200).send(docs);
            response.end();
        }
    });
});
```

使用toArray()將資料轉成陣列
function的docs是轉成陣列後的結果

搜尋MONGODB

```
.....  
var myDB;  
mongodb.MongoClient.connect(mongodbURL, function(err, db) {  
    if (err) {  
        console.log(err);  
    } else {  
        myDB = db;  
        console.log('connection success');  
    }  
});
```

```
.....  
app.get('/api/test', function(request, response) {  
    var collection = myDB.collection('test');  
    collection.find({}).toArray(function(err, docs) {
```

```
        if (err) {  
            response.status(406).end();  
        } else {
```

```
            response.type('application/json');  
            response.status(200).send(docs);  
            response.end();  
        }  
    });  
});
```

轉陣列過程若有err，回傳給錯誤碼406
此為Http協定狀態碼

Http協定狀態碼：
<http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html>

搜尋MONGODB

```
.....
var myDB;
mongodb.MongoClient.connect(mongodbURL, function(err, db) {
    if (err) {
        console.log(err);
    } else {
        myDB = db;
        console.log('connection success');
    }
});

.....
app.get('/api/test', function(request, response) {
    var collection = myDB.collection('my_data');
    collection.find({}).toArray(function(err, docs) {
        if (err) {
            response.status(400).end();
        } else {
            response.type('application/json');
            response.status(200).send(docs);
            response.end();
        }
    });
});
```

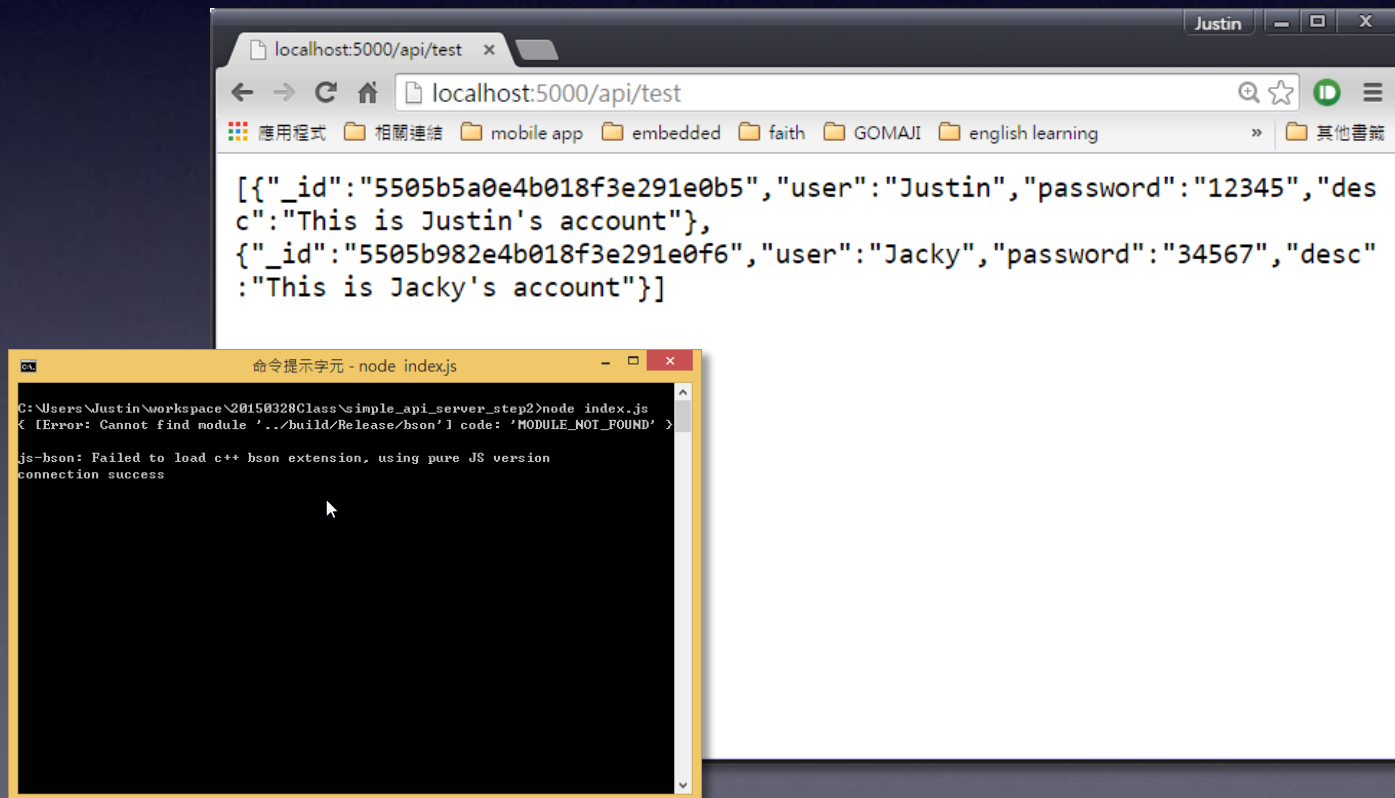
沒有錯誤回傳狀態碼200並附帶著資料
因為MongoDB存的資料就是JSON
所以不用特別轉換

搜尋MONGODB

- MongoDB的搜尋方式非常多，請參考以下連結
 - <http://docs.mongodb.org/manual/reference/method/db.collection.find/>
 - [http://fecbob.pixnet.net/blog/post/38494441 - %E6%B7%BA%E8%AB%87mongodb%E4%B8%AD%E5%B9%BE%E7%A8%AE%E4%B8%8D%E5%90%8C%E6%9F%A5%E8%A9%A2%E6%96%B9%E6%B3%95](http://fecbob.pixnet.net/blog/post/38494441-%E6%B7%BA%E8%AB%87mongodb%E4%B8%AD%E5%B9%BE%E7%A8%AE%E4%B8%8D%E5%90%8C%E6%9F%A5%E8%A9%A2%E6%96%B9%E6%B3%95)

搜尋MONGODB

- 現在來啟動Server，可以參見章節「[啟動Server](#)」
- 啟動後在瀏覽器做測試，網址輸入
<http://localhost:5000/api/test>看看結果



MONGODB其他操作

- Insert

- <http://mongodb.github.io/node-mongodb-native/markdown-docs/insert.html>
- e.g. `collection.insert({"name":"1234"})`

- Remove

- <http://stackoverflow.com/questions/20699671/removing-documents-from-a-mongodb-collection-from-node-js>
- e.g. `collection.delete({"_id":"John"})`

作業5

- 連接作業3的Server API與作業4建立的MongoDB todolist
 - 查詢
 - 回傳todolist中Todos collection全部結果
 - 新增
 - 依照API Server的GET參數，組成如同作業4的JSON格式存到Todos的資料庫中
 - 刪除
 - 依照GET的參數，刪除Todos中對應的_id
 - 刪除比較特別，使用ObjectID，請參考<http://mongodb.github.io/node-mongodb-native/api-bson-generated/objectid.html>
- 相關資料
 - <http://mongodb.github.io/node-mongodb-native/api-generated/collection.html>

APP與SERVER連接

APP與SERVER連接



APP與SERVER連接

對應專案
NetworkCommunication_step5

- 原本的程式碼需要改變的地方
- 假設連線的是模擬器，連線的網址要改成
 - Android官方模擬器：<http://10.0.2.2:5000/api/test>
 - Genymotion：<http://10.0.2.3:5000/api/test>

```
public class MainActivity extends Activity {  
  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        String request = new StringRequest(Request.Method.GET,  
            "http://10.0.2.2:5000/api/test", mResponseListener, mErrorListener);  
        NetworkManager.getInstance(this).request(null, request);  
    }  
    .....  
}
```

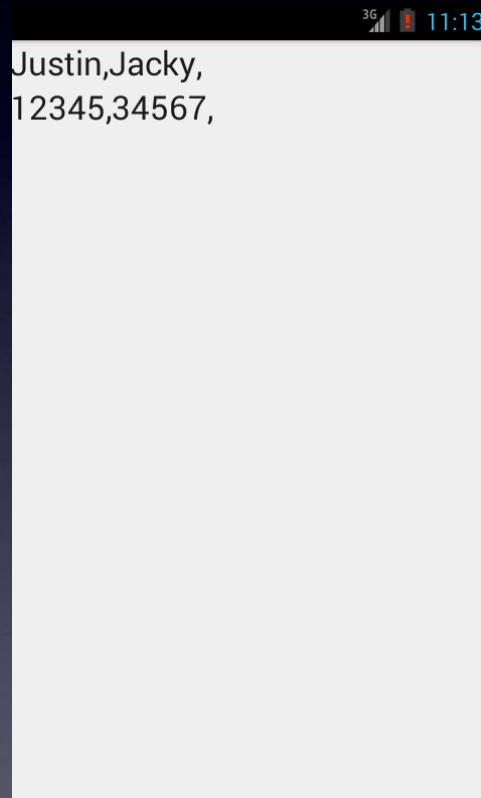
APP與SERVER連接

- 在Response的Listener部分，要依照回傳的JSON進行剖析

```
private Listener<String> mResponseListener = new Listener<String>() {  
    public void onResponse(String string) {  
        try {  
            JSONArray ary = new JSONArray(string);  
            StringBuilder users = new StringBuilder();  
            StringBuilder passwords = new StringBuilder();  
            for (int i = 0; i < ary.length(); i++) {  
                JSONObject json = ary.getJSONObject(i);  
                String user = json.getString("user");  
                users.append(user);  
                users.append(",");  
                String password = json.getString("password");  
                passwords.append(password);  
                passwords.append(",");  
            }  
            .....  
        }  
    }  
};
```

APP與SERVER連接

- 執行結果



作業6

- 建立一個代辦事項App，共分三個頁面
 - 第一頁：可以選擇新增代辦事項或是瀏覽代辦事項
 - 第二頁：瀏覽或刪除代辦事項
 - 第三頁：新增代辦事項
- 新增的代辦事項要連接作業5的MongoDB
- 查詢代辦事項也要與MongoDB連結
- 刪除代辦事項，也要從MongoDB中刪除

作業6

- 操作選擇頁，可以選擇功能2項
- 點下功能後切換到所屬的 Activity



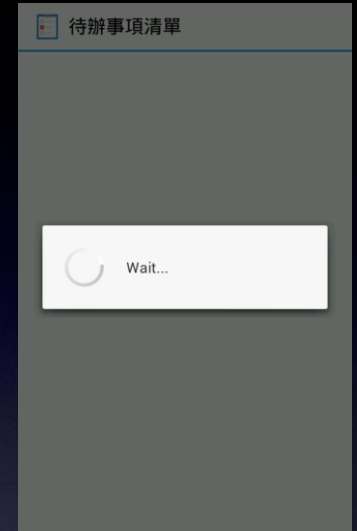
作業6

- 新增待辦事項頁
- 輸入建立者、事項、說明
- 按下新增按鈕時，將資訊都傳至Server API新增資料
 - 建立者→ owner
 - 事項→ title
 - 說明→ desc
 - 時間→ time
- 本頁注意事項，文字上傳到DB時要使用URL Encode編碼，避免產生亂碼
- 目前時間可以使用System.currentTimeMillis()取得



作業6

- 待辦事項清單頁
- 進入後由Server API負責將MongoDB的結果回傳，回傳的結果以ListView呈現
- ListView的每一行都必須包含title, time, owner
- 本頁注意事項
 - 請練習使用BaseAdapter建立客製化的ListView呈現方式
 - 客製化的Layout可使用LayoutInflater產生
 - 由數字轉變成可讀的時間，可以使用SimpleDateFormat



作業6

- 待辦事項清單頁
- 點選ListView的任何一行會出現AlertDialog
- AlertDialog會呈現出點下項目的desc
- AlertDialog包含關閉和刪除的按鈕
- 按下刪除，藉由Server API到MongoDB刪除資料
- 刪除成功後更新ListView

