

MEAN Stack實作班02

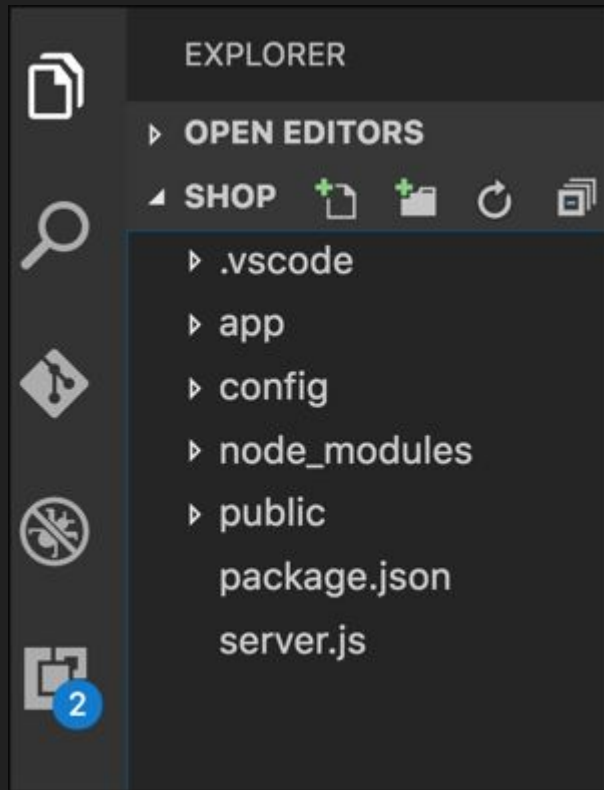
上課資源

檔案

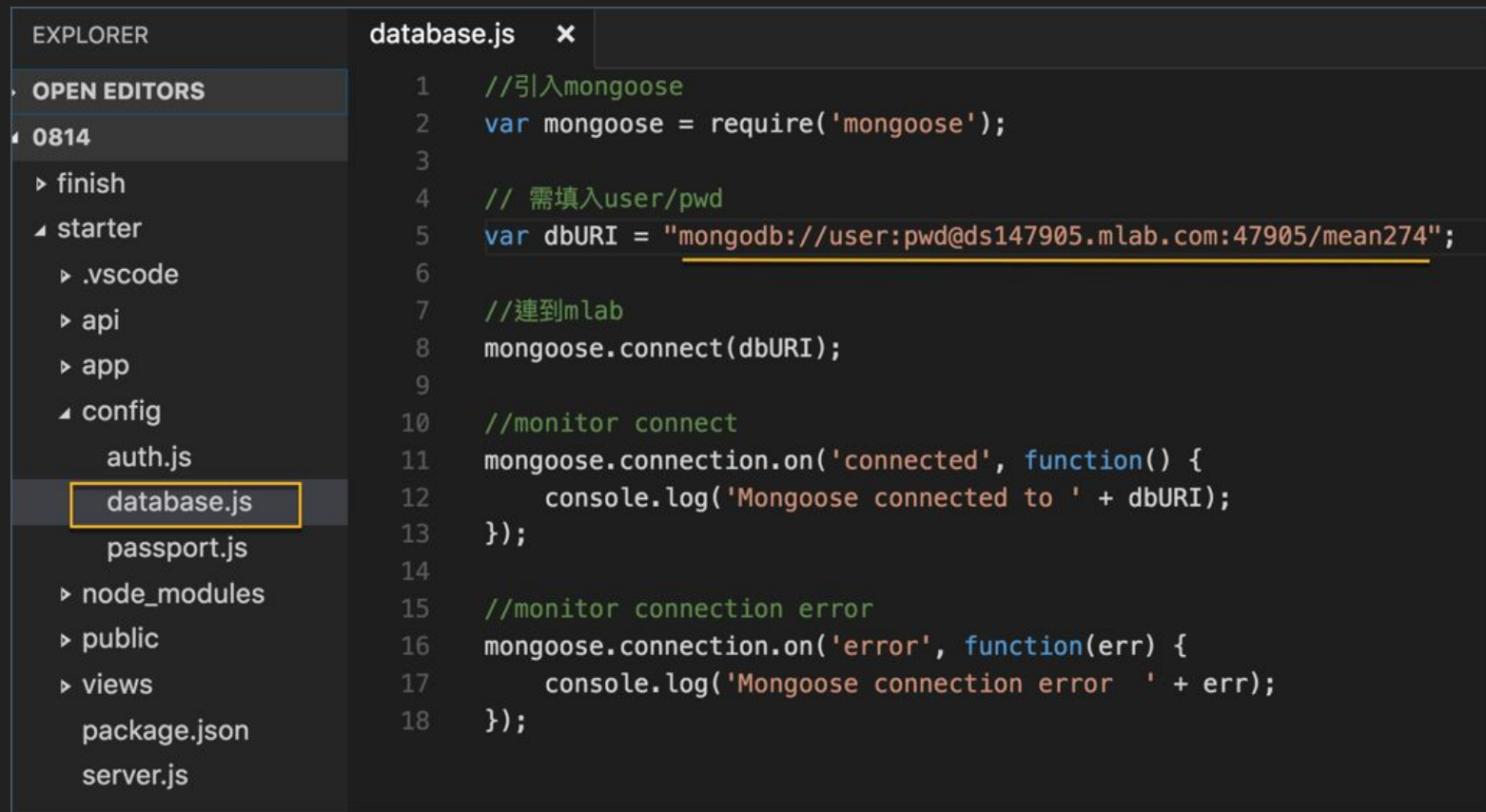
<https://www.dropbox.com/sh/mnxu5fosuhjpug0/AACoP4bMMMPOeEFYKEKPrqXpa?dl=0>

開啟shop資料夾

- server.js上按右鍵，開啟命令提示字元
- 輸入`npm install`安裝packages
- 修改`config/databast.js`內dbURI的網址及帳密
- 測試`node server.js`是否啟動



修改連到自己的mlab



EXPLORER

OPEN EDITORS

0814

- finish
- starter
 - .vscode
 - api
 - app
 - config
 - auth.js
 - database.js**
 - passport.js
 - node_modules
 - public
 - views
 - package.json
 - server.js

database.js x

```
1 //引入mongoose
2 var mongoose = require('mongoose');
3
4 // 需填入user/pwd
5 var dbURI = "mongodb://user:pwd@ds147905.mlab.com:47905/mean274";
6
7 //連到mlab
8 mongoose.connect(dbURI);
9
10 //monitor connect
11 mongoose.connection.on('connected', function() {
12     console.log('Mongoose connected to ' + dbURI);
13 });
14
15 //monitor connection error
16 mongoose.connection.on('error', function(err) {
17     console.log('Mongoose connection error ' + err);
18 });
```

nodemon與morgan

nodemon server.js

安裝nodemon:

npm install nodemon -g


```
[nodemon] restarting due to changes...  
[nodemon] starting `node server.js`  
Server is running on port 3000.....
```

morgan

```
// 在console印出收到的request  
var morgan = require('morgan');
```

```
//middleware  
// set up our express application  
app.use(morgan('dev')); // log every request to the console
```

```
Server is running on port 3000.....  
Mongoose connected to mongodb://kuolun:kuolun@ds017584.mlab.com:17584/mean_course  
GET / 304 3.857 ms - -  
GET /about 304 1.292 ms - -  
█
```



VS Code熱鍵

Ctrl + X 刪除一整行

Ctrl + Z 回覆至上一個動作

alt + shift + 向下箭頭 往下repeat游標所在那一行

alt + shift + 向上箭頭 往上repeat游標所在那一行

ctrl + / 註解掉游標所在那一行

VS Code熱鍵

control + ~ 在兩個VS code視窗之間切換

ctrl + D 重複選取反白處

ctrl+ C 複製

ctrl + V 貼上

require() 的路徑參數

- 取得module.exports裡存放的物件(字串, 數值, 函數等)
- 接受module(模組)的絕對路徑和相對路徑
- 可省略module檔案的副檔名 .js
- 沒有加上前綴(如./), Node.js會去載入已安裝的核心模組或第三方模組
- 路徑可以是目錄, require會試圖載入該目錄下的index.js

```
var myModule = require('./my_module');  
var myModule = require('./my_module/index.js');
```

Routing (自定路由)

沒有Express時:

自己解析URL, 判斷URL路徑, 依照路徑不同決定要執行什麼工作

有Express後:

提供Routing機制, 可讓我們用get()或post ()方法自訂Routing和對應的處理函數

不用再自己解析URL路徑

Example

假設路徑是 <http://localhost:3000/myRoute>

```
//處理get request  
app.get('/myRoute', 處理函數);  
  
//處理post request  
app.post('/myRoute', 處理函數);
```

處理函數(callback)

request - 連線要求的資訊和方法

response - 回應連線的資訊和方法

```
function(req,res) {  
    //處理連線要求, 並回應browser  
}
```

新增Routes

app/**routes.js**

module.exports

routes.js



```
1  module.exports = function(app) {  
2  
3  };
```

Require / module.exports

Server.js

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

新增app/route.js

```
module.exports = function(app) {  
  // 建立API  
  
  // Homepage  
  app.get('/', function(req, res) {  
    res.send("Hello from route");  
  });  
};
```


加about route

```
module.exports = function(app) {  
  // 建立API  
  
  // Homepage  
  app.get('/', function(req, res) {  
    res.send("Hello from route");  
  });  
  
  //about  
  app.get('/about', function(req, res) {  
    res.send("Hello from about");  
  });  
};
```

在server.js require route.js

```
// 設定路徑  
require('./app/routes.js')(app);
```

```
/      = Root directory  
./     = Current directory  
../    = Parent of current directory  
../../ = Two directories backwards
```

app/routes.js

- 新增“*” route

```
routes.js  ×  server.js
1  module.exports = function(app) {
2      // 建立API
3
4
5      // Homepage
6  app.get('/', function(req, res) {
7      res.send("Hello from route");
8  });
9
10     //about
11  app.get('/about', function(req, res) {
12      res.send("Hello from about");
13  });
14
15     //其他沒對應到的都回傳Page not found
16  app.get("*", function(req, res){
17      res.send("Page not found");
18  });
19  };
```

測試



如果把這個route放到最前面會發生什麼事？

Route Parameters

route parameters



route params


:subredditName

對應到soccer



```
// subreddit
app.get('/r/:subredditName', function(req, res){
  console.log(req.params);
  res.send('Welcome to ' + req.params.subredditName);
});
```


多個 parameters

 https://www.reddit.com/r/soccer/comments/4xj1hd/postmatch_thread_hull_city_2_1_leicester_city/

```
app.get( '/r/subRedditName/comments/id/title' )
```

多個 parameters

```
// multi params
app.get('/r/:subRedditName/comments/:id/:title', function(req, res){
  console.log(req.params);
  res.send('Welcome to comments page!');
});
```



```
GET /r/puppets/comments/123/the_first_article 200 1.043 ms - 14
{ subRedditName: 'puppets',
  id: '123',
  title: 'the_first_article' }
```

練習時間: 10分鐘

- 建立單一params的route
- 建立multiple params的route

REST

Restful Routing

- REST: **R**epresentational **S**tate **T**ransfer
- 是一種style, 不是protocol
- 一種介於HTTP routes跟CRUD的mapping
- 大家容易follow的規則

常用的HTTP 方法(method)

- GET:請求取得指定資源
- POST:向指定資源傳送資料
- PUT:請求伺服器儲存一個資源
- DELETE:請求伺服器刪除指定資源

根據REST設計模式, 分別用於實現下面功能

- GET:讀取
- POST:新增
- PUT:更新
- DELETE:刪除

BLOG

CREATE

READ /allBlogs

UPDATE /updateBlog/:id

DESTROY /destroyBlog/:id

RESTful Routes

A table of all 7 RESTful routes

Name	Path	HTTP Verb	Purpose
Index	/dogs	GET	List all dogs
New	/dogs/new	GET	Show new dog form
Create	/dogs	POST	Create a new dog, then redirect somewhere
Show	/dogs/:id	GET	Show info about one specific dog
Edit	/dogs/:id/edit	GET	Show edit form for one dog
Update	/dogs/:id	PUT	Update a particular dog, then redirect somewhere
Destroy	/dogs/:id	DELETE	Delete a particular dog, then redirect somewhere

API

- API: application program interface, 讓 application 可互相溝通

建立RESTful API

概念

- 直接利用HTTP的命令和狀態，讓API設計者省下許多功夫
- 常用HTTP 通訊協定標準方法如下
 - GET - 讀取
 - POST - 新增
 - PUT - 更新
 - DELETE - 刪除

舉例

- DELETE /api/user/12345678
- 用HTTP通訊協定中的Status Code就可以回傳API執行結果
- 常見Status Code:
 - 200: 正常
 - 404: 找不到指定內容
 - 400: 錯誤請求
 - 401: 未認證
 - 403: 存取拒絕
 - 500: 內部錯誤

使用Express提供Restful Web API

Express對每種HTTP請求方法都設計了不同的路由綁定函數

- GET - app.get()
- POST - app.post()
- DELETE - app.delete()
- PUT - app.put()

例子

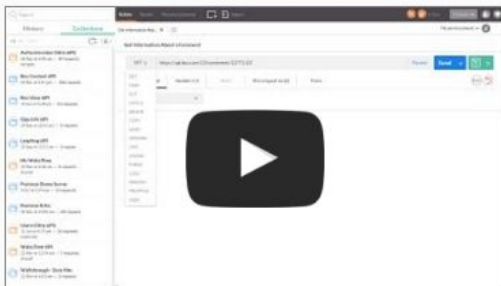
```
app.delete('/api/user/:id', function(req,res) {
  var id = req.params.id;
  //從資料庫裡刪掉user
  db.delete(id, function(err,num) {
    if(err){
      //系統出錯
      res.status(400);
      return;
    }
    if(num == 0){
      //無此使用者
      res.stauts(404);
      return;
    }
    // 刪除成功
    res.status(200);
  });
});
```

POSTMAN介紹

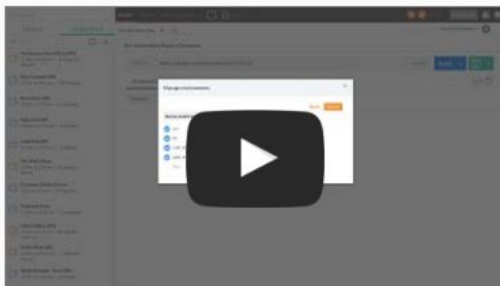
<https://www.getpostman.com/>

<https://www.getpostman.com/support>

Get Started with Postman



How to send and capture API requests using Postman



How to use Postman environments



How to use and share Postman Collections

[View Documentation](#)

[Pricing](#)[Apps](#)[Documentation](#)[Integrations](#)

POSTMAN

Modern software is built on APIs.

Postman helps you develop APIs faster.



Chrome App



Mac App

Go ahead and download our apps, they're free!

google回傳response body (payload)

The screenshot shows the Chrome DevTools Network tab with a request to `http://www.google.com`. The 'Body' tab is selected, and the response is displayed in HTML format. A yellow dashed arrow points from the 'Body' tab to the 'No Auth' dropdown menu.

Authorization: No Auth

Body: Cookies Headers (12) Tests

Status: 200 OK Time: 535 ms

Pretty Raw Preview HTML

```
1 <!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="zh-TW"><head><meta
  content="/logos/doodles/2016/2016-doodle-fruit-games-day-6-5753948142043136-hp.gif"
  itemprop="image"><link href="/images/branding/product/ico/googleg_lodp.ico" rel="shortcut
  icon"><meta content="2016 Doodle 水果遊戲就在 g.co/fruit #GoogleDoodle" property="og
  :description"><meta content="http://www.google.com/logos/doodles/2016/2016-doodle-fruit
  -games-day-6-5753948142043136-thp.png" property="og:image"><meta content="400" property="og
  :image:width"><meta content="132" property="og:image:height"><meta content="origin" id
  ="mref" name="referrer"><title>Google</title><script>(function(){window.google={kEI:'gp
  -qV_GDAYKs0AS-3YPwCw',kEXPI:'750222,750448,1351501,3300015,3300100,3300134,3300161,3312779
  ,3313265,3313270,3700315,3700389,4029815,4031109,4032677,4036509,4036527,4038012,4038214
  ,4038394,4039268,4040135,4041776,4043041,4043492,4045096,4045293,4045841,4046043,4046835
  ,4046837,4046904,4047140,4047454,4047593,4048347,4048980,4049063,4050281,4050714,4050750
  ,4051887,4052304,4056126,4056682,4058016,4059817,4061155,4061666,4061922,4061980,4062203
  ,4062724,4063220,4063879,4064168,4064702,4064796,4065787,4065793,4065873,4065919,4066654
```

跟看原始碼一樣

```
<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="zh-TW"><head><meta charset="utf-8"><title>2016 Doodle 水果遊戲就在 g.co/fruit #GoogleDoodle</title><link href="/images/branding/product/2016/doodle-fruit-games-day-6-5753948142043136-hp.gif" itemprop="image"><link href="/images/branding/product/2016/doodle-fruit-games-day-6-5753948142043136-hp.gif" itemprop="image"><meta content="2016 Doodle 水果遊戲就在 g.co/fruit #GoogleDoodle" property="og:description"><meta content="http://www.google.com/logos/doodles/2016/2016-doodle-fruit-games-day-6-5753948142043136-hp.gif" property="og:image"><meta content="http://www.google.com/logos/doodles/2016/2016-doodle-fruit-games-day-6-5753948142043136-hp.gif" property="og:image:width"><meta content="132" property="og:image:height"><script>(function(){window.google={kEI:'W5-qV-XYCMyx0ATb4qLwAQ',kEXPI:'750222,1351501,3300015,3300100,3300134,3300161,3312779,3313269,36509,4036527,4038012,4038214,4038394,4039268,4041776,4043041,4043492,4045096,4045293,4045304,4045305,4045306,4045307,4045308,4045309,4045310,4045311,4045312,4045313,4045314,4045315,4045316,4045317,4045318,4045319,4045320,4045321,4045322,4045323,4045324,4045325,4045326,4045327,4045328,4045329,4045330,4045331,4045332,4045333,4045334,4045335,4045336,4045337,4045338,4045339,4045340,4045341,4045342,4045343,4045344,4045345,4045346,4045347,4045348,4045349,4045350,4045351,4045352,4045353,4045354,4045355,4045356,4045357,4045358,4045359,4045360,4045361,4045362,4045363,4045364,4045365,4045366,4045367,4045368,4045369,4045370,4045371,4045372,4045373,4045374,4045375,4045376,4045377,4045378,4045379,4045380,4045381,4045382,4045383,4045384,4045385,4045386,4045387,4045388,4045389,4045390,4045391,4045392,4045393,4045394,4045395,4045396,4045397,4045398,4045399,4045400,4045401,4045402,4045403,4045404,4045405,4045406,4045407,4045408,4045409,4045410,4045411,4045412,4045413,4045414,4045415,4045416,4045417,4045418,4045419,4045420,4045421,4045422,4045423,4045424,4045425,4045426,4045427,4045428,4045429,4045430,4045431,4045432,4045433,4045434,4045435,4045436,4045437,4045438,4045439,4045440,4045441,4045442,4045443,4045444,4045445,4045446,4045447,4045448,4045449,4045450,4045451,4045452,4045453,4045454,4045455,4045456,4045457,4045458,4045459,4045460,4045461,4045462,4045463,4045464,4045465,4045466,4045467,4045468,4045469,4045470,4045471,4045472,4045473,4045474,4045475,4045476,4045477,4045478,4045479,4045480,4045481,4045482,4045483,4045484,4045485,4045486,4045487,4045488,4045489,4045490,4045491,4045492,4045493,4045494,4045495,4045496,4045497,4045498,4045499,4045500,4045501,4045502,4045503,4045504,4045505,4045506,4045507,4045508,4045509,4045510,4045511,4045512,4045513,4045514,4045515,4045516,4045517,4045518,4045519,4045520,4045521,4045522,4045523,4045524,4045525,4045526,4045527,4045528,4045529,4045530,4045531,4045532,4045533,4045534,4045535,4045536,4045537,4045538,4045539,4045540,4045541,4045542,4045543,4045544,4045545,4045546,4045547,4045548,4045549,4045550,4045551,4045552,4045553,4045554,4045555,4045556,4045557,4045558,4045559,4045560,4045561,4045562,4045563,4045564,4045565,4045566,4045567,4045568,4045569,4045570,4045571,4045572,4045573,4045574,4045575,4045576,4045577,4045578,4045579,4045580,4045581,4045582,4045583,4045584,4045585,4045586,4045587,4045588,4045589,4045590,4045591,4045592,4045593,4045594,4045595,4045596,4045597,4045598,4045599,4045600,4045601,4045602,4045603,4045604,4045605,4045606,4045607,4045608,4045609,4045610,4045611,4045612,4045613,4045614,4045615,4045616,4045617,4045618,4045619,4045620,4045621,4045622,4045623,4045624,4045625,4045626,4045627,4045628,4045629,4045630,4045631,4045632,4045633,4045634,4045635,4045636,4045637,4045638,4045639,4045640,4045641,4045642,4045643,4045644,4045645,4045646,4045647,4045648,4045649,4045650,4045651,4045652,4045653,4045654,4045655,4045656,4045657,4045658,4045659,4045660,4045661,4045662,4045663,4045664,4045665,4045666,4045667,4045668,4045669,4045670,4045671,4045672,4045673,4045674,4045675,4045676,4045677,4045678,4045679,4045680,4045681,4045682,4045683,4045684,4045685,4045686,4045687,4045688,4045689,4045690,4045691,4045692,4045693,4045694,4045695,4045696,4045697,4045698,4045699,4045700,4045701,4045702,4045703,4045704,4045705,4045706,4045707,4045708,4045709,4045710,4045711,4045712,4045713,4045714,4045715,4045716,4045717,4045718,4045719,4045720,4045721,4045722,4045723,4045724,4045725,4045726,4045727,4045728,4045729,4045730,4045731,4045732,4045733,4045734,4045735,4045736,4045737,4045738,4045739,4045740,4045741,4045742,4045743,4045744,4045745,4045746,4045747,4045748,4
```


Header

Body Cookies **Headers (12)** Tests

alt-svc → quic=":443"; ma=2592000; v="36,35,34,33,32,31,30"

alternate-protocol → 443:quic

cache-control → private, max-age=0

content-encoding → gzip

content-type → text/html; charset=UTF-8

date → Wed, 10 Aug 2016 03:29:06 GMT

expires → -1

p3p → CP="This is not a P3P policy! See <https://www.google.com/support/accounts/answer/151657?hl=en> for more info."

server → gws

status → 200

x-frame-options → SAMEORIGIN

x-xss-protection → 1; mode=block

status code

Cookies Headers (12) Tests

Status: 200 OK Time: 535 ms

→ quic="443"; ma=2592000; v="36,35,34,33,32,31,30"

te-protocol → 443:quic

control → private, max-age=0


t-encoding → gzip

t-type → text/html; charset=UTF-8

Wed, 10 Aug 2016 03:29:06 GMT

→ -1

CP="This is not a P3P policy! See <https://www.google.com/support/accounts/answer/151657?hl=en> for more info."

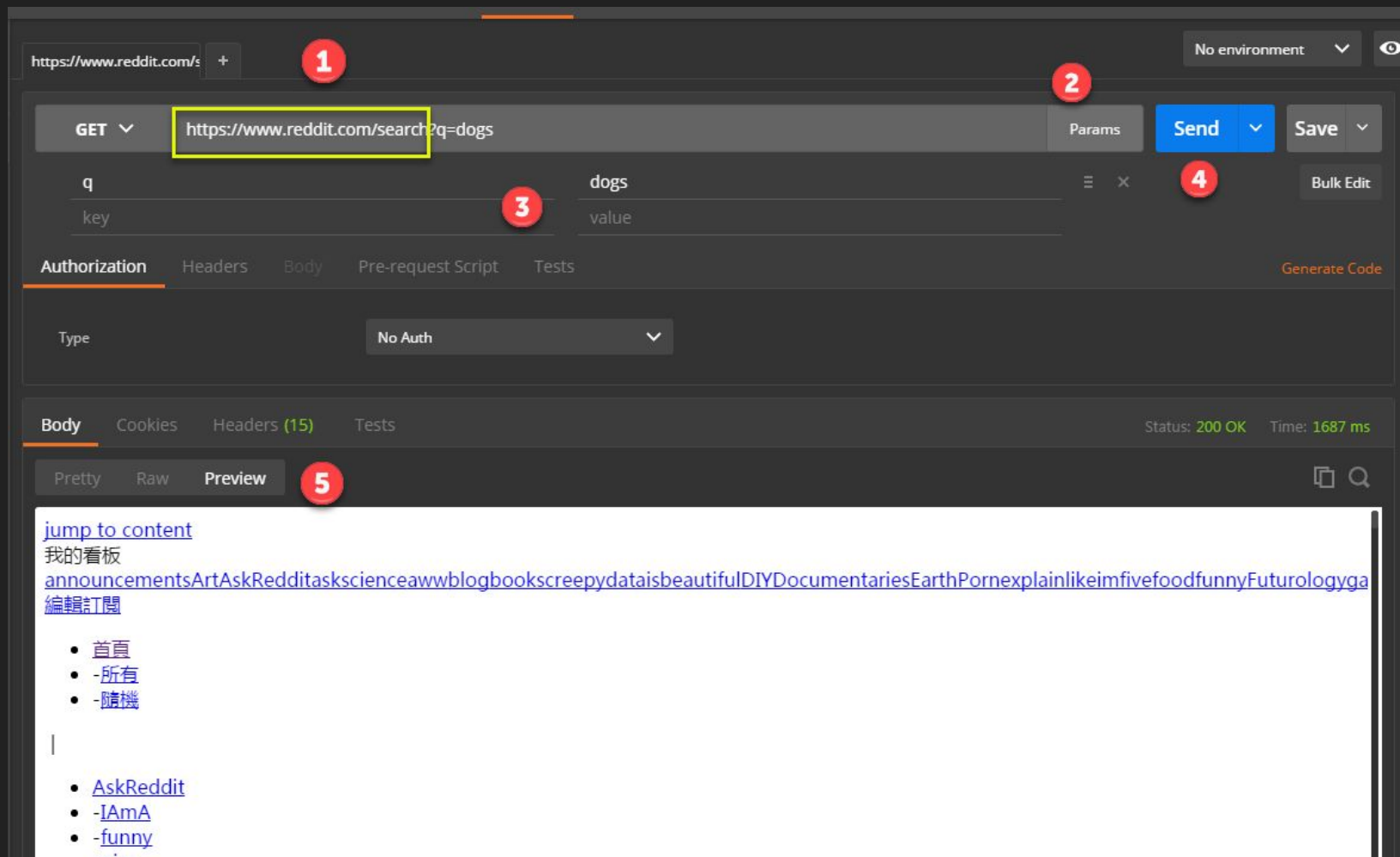


Query String

Browser的search bar
只能發GET



`https://www.reddit.com/search?q=keyboard&name=kuolun`



DEMO

localhost:3000 +

GET ▼

localhost:3000

Params

key

value

Authorization

Headers

Body

Pre-request Script

Tests

Type

No Auth ▼

Body

Cookies

Headers (6)


Tests

Pretty

Raw

Preview

HTML ▼



i 1 Hello from route

練習: 10分鐘

- 練習使用postman對網站發出request
- 練習對自己的server發出request

建立購物車的產品目錄

server.js 引入body-parser 模組

app.use

=>使用middleware

bodyParser.json()

=>回傳只parse json的
middleware

```
// 引入body-parser 模組
```

```
var bodyParser = require('body-parser');
```

```
app.use(bodyParser.json());
```

```
app.use(bodyParser.urlencoded({  
  extended: true  
}));
```

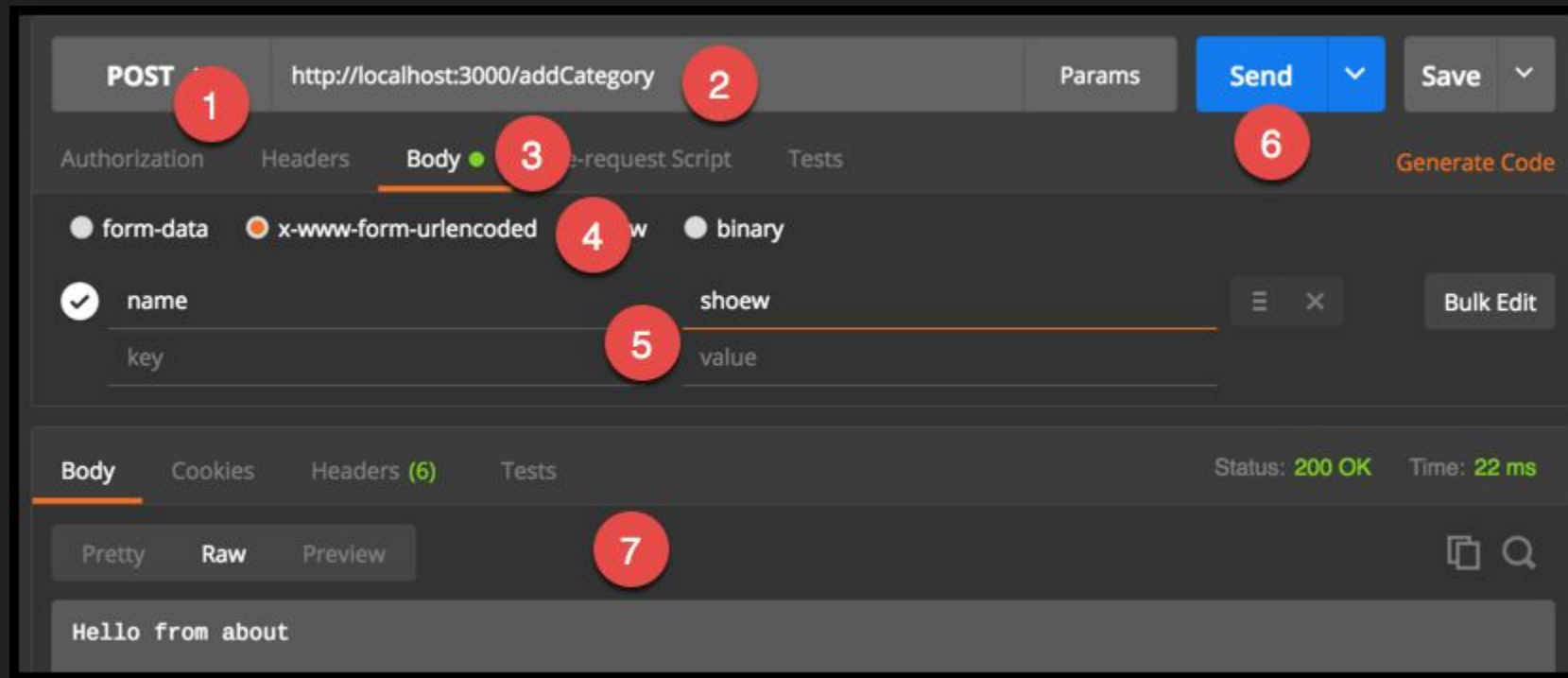
建立 新增category的route (app/routes.js)

/addCategory

POST

```
routes.js x  routes.js  database.js  database.js  api.j
1  module.exports = function(app) {
2      // 建立API
3      //取得category model
4      var Category = require('./models/category');
5
6      // 新增目錄
7      app.post('/addCategory', function(req, res) {
8          var category = new Category();
9          category.name = req.body.name;
10
11          category.save(function(err, category) {
12              res.json({
13                  category: category
14              });
15          });
16      })
17  }
```

postman測試



localhost:3000/addCategory

MEAN app

POST

localhost:3000/addCategory

Params

Send

key

value

Authorization

Headers

Body

Pre-request Script

Tests

☐ form-data

☒ x-www-form-urlencoded

☐ raw

☐ binary



name

telephone

key

value

Body

Cookies

Headers (6)

Tests

Status: 200 OK

Pretty

Raw

Preview

JSON



```
1 {  
2   "category": {  
3     "__v": 0,  
4     "name": "telephone",  
5     "_id": "57afd940a5d1540b59cf610d"  
6   }  
7 }
```

mlab 會多一筆telephone的類別

- 產生categories collection
- 產生一筆telephone document

Collections	Users	Stats
Collections		
NAME	DOCUMENTS	
categories	1	

Documents

-- Start new search --

All Documents

Display mode: ☒ list ☐ table ([edit table view](#))

records / page 10

```
{
  "_id": {
    "$oid": "57afd940a5d1540b59cf610d"
  },
  "name": "telephone",
  "__v": 0
}
```

設計取得所有目錄的route

Demo

取得所有目錄

Category.find() - 傳入query條件去找

```
//取得所有目錄for dropdown
app.get('/categories/all', function(req, res) {
  //空{}代表傳回categories下所有document
  Category.find({}, function(error, categories) {
```


GET ▾

localhost:3000/categories/all

key

value

Authorization

Headers

Body

Pre-request Script

Tests

Type

No Auth ▾

Body

Cookies

Headers (6)

Tests

Pretty

Raw

Preview

JSON ▾



```
1 {  
2   "categories": [  
3     {  
4       "_id": "57afd940a5d1540b59cf610d",  
5       "name": "telephone",  
6       "__v": 0  
7     }  
8   ]  
9 }
```

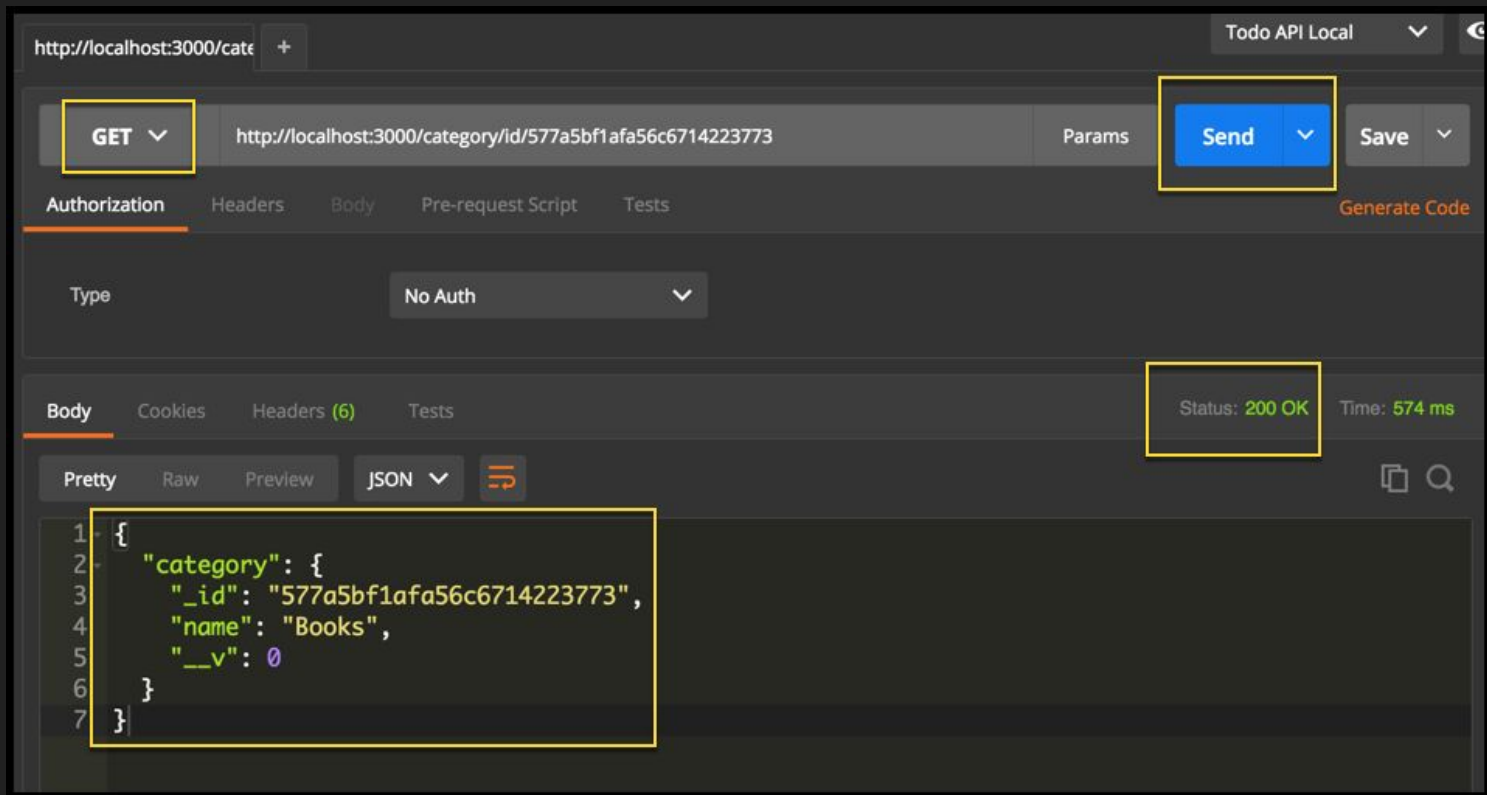
用id取得單一目錄

Demo

Category.findOne - 傳入id去找

```
//用id找出特定目錄
app.get('/category/id/:id', function(req, res) {
  // 用Category Model去找data
  // 對應的collection為categories
  Category.findOne({
    _id: req.params.id
  }, function(error, category) {
```

練習：用POSTMAN測試



GET ▼

localhost:3000/category/id/57afd940a5d1540b59cf610d

key

value

Authorization

Headers

Body

Pre-request Script

Tests

Type

No Auth ▼

Body

Cookies

Headers (6)

Tests

Pretty

Raw

Preview

JSON ▼



```
1 - {  
2 -   "category": {  
3     "_id": "57afd940a5d1540b59cf610d",  
4     "name": "telephone",  
5     "__v": 0  
6   }  
7 }
```

練習15分鐘

完成3個route API及測試 並新增3個目錄

get /category/id/:id

get /categories/all

post /addCategory

建立Fake Data

介紹faker.js

<http://marak.github.io/faker.js/>



- commerce
 - color
 - department
 - productName
 - price
 - productAdjective
 - productMaterial
 - product

用法

npm install faker

- internet
 - avatar
 - email
 - exampleEmail
 - userName
 - protocol
 - url
 - domainName
 - domainSuffix
 - domainWord
 - ip
 - userAgent
 - color
 - mac
 - password

- name
 - firstName
 - lastName
 - findName
 - jobTitle
 - prefix
 - suffix
 - title
 - jobDescriptor
 - jobArea
 - jobType

Node.js

– raw 5

```
var faker = require('faker');

var randomName = faker.name.findName(); // Rowan Nikolaus
var randomEmail = faker.internet.email(); // Cassandra.Haley@erich.biz
var randomCard = faker.helpers.createCard(); // random contact card containing many properties
```


新增api/api.js

```
// 完整路徑 localhost:3000/api/xxxxxxx(要建立products的目錄名)
var router = require('express').Router();

var faker = require('faker');
var Category = require('../app/models/category');
var Product = require('../app/models/product');
```

修改server.js

```
//faker模組  
var apiRoutes = require('./api/api');
```

```
//設定subroute for faker  
app.use('/api', apiRoutes);
```

```
// 設定路徑  
require('./app/routes.js')(app);
```

```
router.get('/:name', function(req, res, next) {  
  // DB需要先有對應的category存在，才能建立product  
  Category.findOne({  
    name: req.params.name  
  }, function(err, category) {  
    if (err) return next(err);  
    // 設定要建立幾筆product  
    for (var i = 0; i < 10; i++) {  
      var product = new Product();  
      product.category = category._id;  
      //check faker api  
      product.name = faker.commerce.productName();  
      product.price = faker.commerce.price();  
      product.image = faker.image.image();  
      product.save();  
    }  
    // 都建立完之後再回傳json  
    res.json({  
      message: 'Success'  
    });  
  });  
});
```

```
//create product schema  
var schema = Schema({  
  category: {  
    type: Schema.Types.ObjectId,  
    ref: 'Category'  
  },  
  name: String,  
  price: Number,  
  image: String  
});
```

http://localhost:3000/api/ +

GET ▼

http://localhost:3000/api/telephone

key

value

Authorization

Headers

Body

Pre-request Script

Tests

Type

No Auth ▼

Body

Cookies

Headers (6)

Tests

Pretty

Raw

Preview

JSON ▼




```
1 - {  
2   "message": "Success"  
3 }
```

mlab

- 會產生products collections
- 產生10筆product documents

Collections	Users	Stats
Collections		
NAME	DOCUMENTS	
categories	1	
products	10	



練習時間 15分鐘

- 建立api.js
- 調整server.js
- 用postman建立3個類別下的products

建立Product相關的routes

/products/:id

/product/:id

/productsall/

routes.js

```
//取得product model  
var Product = require('./models/product');
```


/products/:id 取得某一目錄下所有products

```
//id對應category，取得某一目錄下所有products
app.get('/products/:id', function(req, res, next) {
  Product
    .find({
      category: req.params.id
    })
    // 將category path替換成對應的資料
    .populate('category')
    .exec(function(err, products) {
      if (err) return next(err);
      // 取到資料就回傳json
      res.json({
        products: products
      });
    });
});
```

```
//create product schema
var schema = Schema({
  category: {
    type: Schema.Types.ObjectId,
    ref: 'Category'
  },
  name: String,
  price: Number,
  image: String
});
```

/productsall/

取得所有product

```
//取得所有product
app.get('/productsall/', function(req, res) {
  //空{}代表傳回Category下所有document
  Product.find({})
    .populate('category')
    .exec(function(error, products) {
      if (error) {
        return res.status(500).
          json({
            error: error.toString()
          });
      }
      res.json({
        products: products
      });
    });
});
```

/product/:id

id對應到的product

```
//用id找特定product
app.get('/product/:id', function(req, res) {
  Product.findById({
    _id: req.params.id
  }, function(err, product) {
    if (err) return next(err);
    //回傳json
    res.json({
      product: product
    });
  })
});
```

DEMO

練習

- 建立product相關3個routes
- 用postman測試

GET

http://localhost:3000/productsall

Authorization

Headers

Body

Pre-request Script

Tests

Type

No Auth

Body

Cookies

Headers (6)

Tests

Pretty

Raw

Preview

JSON



```
1 {  
2   "products": [  
3     {  
4       "_id": "578b461223dd391e9032422c",  
5       "image": "http://goo.gl/p95H3Y",  
6       "price": 215,  
7       "name": "Kindle fire",  
8       "category": {  
9         "_id": "5785f1c5a18925d41b2842a8",  
10        "name": "cars",  
11        "__v": 0  
12      }  
    ]  
  }
```

GET ▾

http://localhost:3000/products/579426f565cb84805f36cc90

Param

Body

Cookies

Headers (6)

Tests

Pretty

Raw

Preview

JSON ▾



```
1 - {
2 -   "products": [
3 -     {
4 -       "_id": "579436be54b24ee15fbb2d95",
5 -       "image": "http://lorempixel.com/640/480/technics",
6 -       "price": 842,
7 -       "name": "Refined Soft Mouse",
8 -       "category": {
9 -         "_id": "579426f565cb84805f36cc90",
10 -        "name": "tv",|
11 -        "__v": 0
12 -      }
```

GET ▾

http://localhost:3000/products/57afd940a5d1540b59cf610d

body

Cookies

Headers (6)

Tests

Pretty

Raw

Preview

JSON ▾



```
1 {  
2   "products": [  
3     {  
4       "_id": "57afe24df46597a862644229",  
5       "image": "http://lorempixel.com/640/480/sports",  
6       "price": 245,  
7       "name": "Incredible Granite Gloves",  
8       "category": {  
9         "_id": "57afd940a5d1540b59cf610d",  
10        "name": "telephone",  
11        "__v": 0  
12      },  
13      "__v": 0  
14    },  
15  ]  
}
```


Robomongo介紹

<https://robomongo.org/>



Robomongo

設定連線

Connection Settings

Connection Authentication SSH Advanced

Name: 1

Choose any connection name that will help you to identify this connection.

Address: 2 : 3

Specify host and port of MongoDB server. Host can be either IPv4, IPv6 or domain name.

! Test Cancel Save

設定帳密及DB

Connection Settings

Connection Authentication SSH Advanced

☒ Perform authentication

Database 1

The admin database is unique in MongoDB. Users with normal access to the admin database have read and write access to **all databases**.

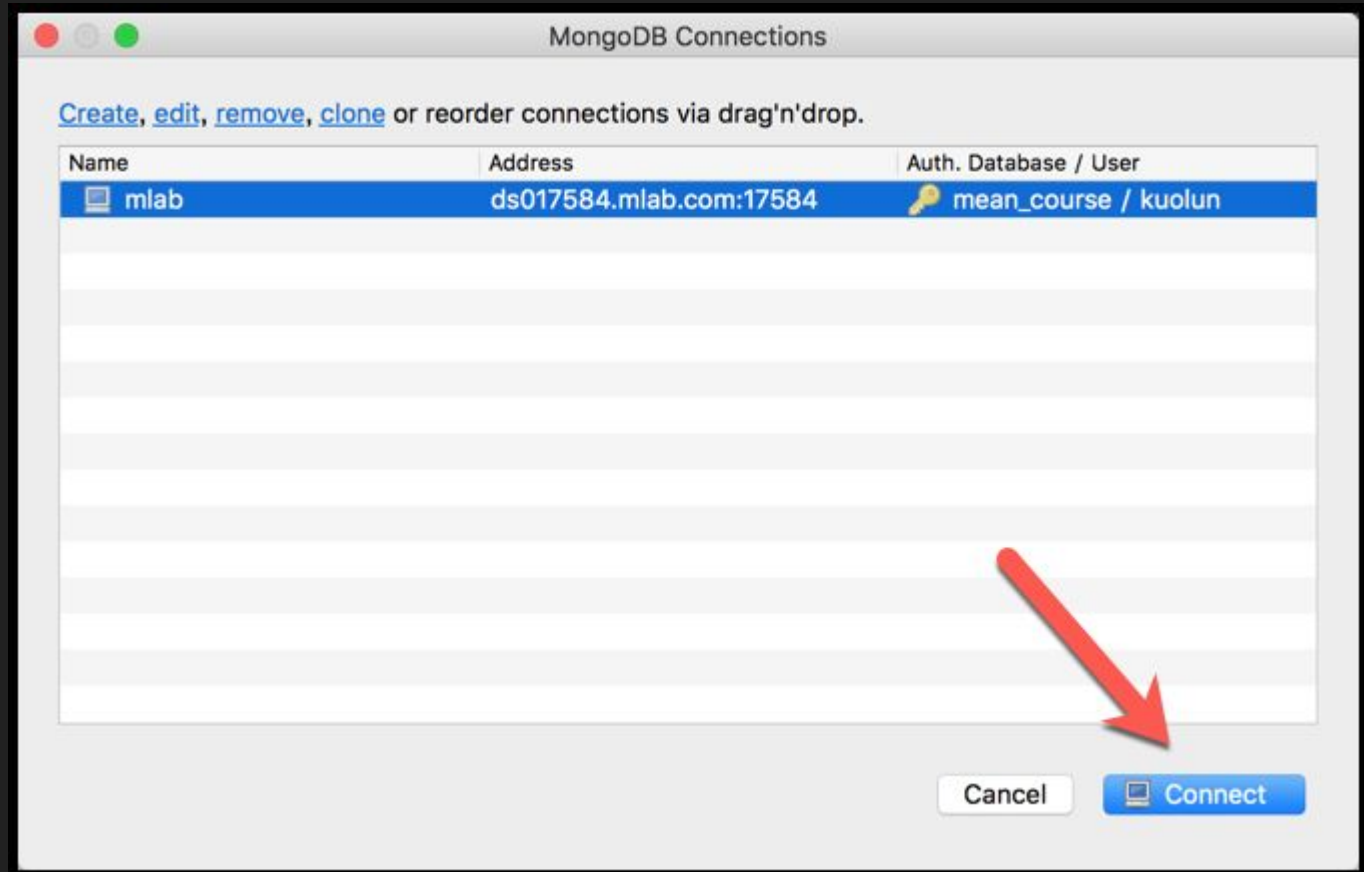
User Name

Password 2

Auth Mechanism 4

3

connect



The screenshot displays the MongoDB Compass interface. On the left, the database structure is shown with 'mean_course' expanded, revealing 'Collections (9)'. The 'categories' collection is highlighted. The main panel shows the query `db.getCollection('categories').find({})` and the resulting document for the first category.

Query: `db.getCollection('categories').find({})`

Result: categories (0.196 sec.)

Key	Value
(1) ObjectId("577a5bf1afa56c6714223...")	{ 3 fields }
(2) ObjectId("5785f18ba18925d41b28...")	{ 3 fields }
(3) ObjectId("5785f1c5a18925d41b28...")	{ 3 fields }
(4) ObjectId("579426f565cb84805f3...")	{ 3 fields }
_id	ObjectId("579426f565cb84805f36c...")
name	tv
_v	0

mlab (1)

- mean_course
 - Collections (9)
 - System
 - categories
 - Indexes
 - cats
 - dogs
 - objectlabs-system
 - objectlabs-system.admin...
 - products
 - Indexes
 - test
 - users
 - Functions
 - Users
- mlab (1)
 - mean_course

* db.getCollection('products').find({category: ObjectId("579426f565cb84805f36cc90")})

mlab ds017584.mlab.com:17584 mean_course

db.getCollection('products').find({category:ObjectId("579426f565cb84805f36cc90")})

products 0.197 sec.

Key	Value	Type
(1) ObjectId("579436be54b24ee15fbb2d95") { 6 fields }		Object
_id	ObjectId("579436be54b24ee15fbb2d95")	ObjectId
image	http://lorempixel.com/640/480/technics	String
price	842	Int32
name	Refined Soft Mouse	String
category	ObjectId("579426f565cb84805f36cc90")	ObjectId
_v	0	Int32
(2) ObjectId("579436be54b24ee15fbb2d95") { 6 fields }		Object
(3) ObjectId("579436be54b24ee15fbb2d95") { 6 fields }		Object
(4) ObjectId("579436be54b24ee15fbb2d95") { 6 fields }		Object
(5) ObjectId("579436be54b24ee15fbb2d95") { 6 fields }		Object
(6) ObjectId("579436be54b24ee15fbb2d95") { 6 fields }		Object
(7) ObjectId("579436be54b24ee15fbb2d95") { 6 fields }		Object
(8) ObjectId("579436be54b24ee15fbb2d95") { 6 fields }		Object
(9) ObjectId("579436be54b24ee15fbb2d95") { 6 fields }		Object
(10) ObjectId("579436be54b24ee15fbb2d95") { 6 fields }		Object

Query 的2種使用方式 Callback或 exec

callback:

```
User.findOne({ name: 'daniel' }, function (err, user) {  
  //  
});
```

exec:

```
User  
  .findOne({ name: 'daniel' })  
  .exec(function (err, user) {  
    //  
  });
```

PUT method: Updating data

- 類似POST method
- 取得form data並去update 現有的document

Recap 步驟

1. 找到相關的document
2. 對instance做一些變動
3. Save document
4. send a JSON response

觀念複習

Mongoose model 直接對應到MongoDB的document

當你的query找到document A

你就拿到model instance

對這個instance做異動然後save

Mongoose就會update document A

```
Product.findById(req.params.productid)
.exec(
  function(err,product){
    product.name = req.body.name;
    product.save( function(err, product) {
      if(err) {
        //deal with err
      }else{
        //deal with success}
      }
    }
  )
}
```

Select

```
Product.findById(req.params.productid)
```

```
.select( ' name price' )
```

```
.exec(
```

```
.select( ' -name -price' )
```

Session and Cookie

Stateless

HTTP protocol是 **Stateless** 連線

也就是client與server不會一直保持連線狀態

通常 HTTP以一個頁面request為單位

當client取得頁面後, 就不再與server端溝通

下次的頁面request會被視為全新獨立的連線

Session 的機制

以去飲料店買飲料為例

點了飲料以後，店員要處理一下

所以給你一張號碼牌

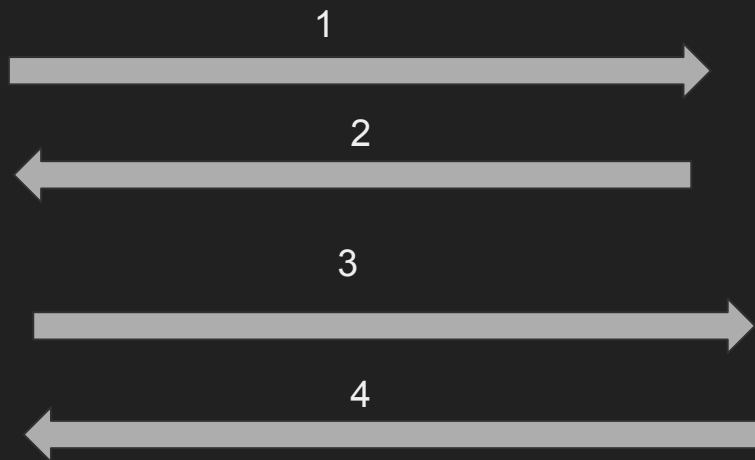
等下回來時，店員就知道你是誰

Session觀念

可用cookie來當成號碼牌

利用瀏覽器上的 **cookie** 達成
(現今的瀏覽器預設都支援 Cookie)

- 利用瀏覽器對同一網域(domain)的頁面, 都會傳送同一份cookie到Server的特性
- cookie內可存放一些資料, Server便可由此得知此次連線與上次連線的關聯性



集中式Session

- Server產生的key會在DB建一筆資料
- 可存放使用者名稱，帳號等
- 缺點：多台機器如何共用session要處理

Cookie-based Session

- 使用cookie存放session的資料
- 不發key, 直接把資料存在client端
- 配合加解密機制
- 缺點: cookie有4KB大小限制

混合式

前兩種一起用

敏感資料放server端

Express session 實作

Express要啟用session要裝2個模組

- cookie-parser
- express-session

cookie session要裝

- cookie-session

Demo

```
// demo  
var cookieParser = require('cookie-parser');  
var cookieSession = require('cookie-session');
```

```
// demo  
app.use(cookieParser());  
app.use(cookieSession({  
  name: 'kuolun-session',  
  keys: ['hello']  
}));
```

回家作業

- 於mlab上建立好3個目錄
- 用faker.js幫3個目錄各增加10個產品

module.exports (補充)

- `require('./config/database.js')` 會去執行database.js內每一行
- `var dbURI = require('./config/database.js');`
dbURI會被assign為database.js內, 有被module.exports出來的物件或function
但因為都沒有寫module.exports
所以console.log(dbURI)會是空物件
- A檔案require B檔案進來, 只會執行B檔案內容, 無法存取B檔案內的東西
要存取, 必須去寫module.exports=要export出來的東西