Raspberry Pi with Python

- Node.js workshop1

고 강 태
010-8269-3535
gangtai.goh@gmail.com
@gtko
http://goo.gl/9V3H6
http://www.facebook.com/gangtai.goh

- VERSION 1.1
- Release Date : 2015.

Book api

Node version manager
nvm
nodist
express.js
books api

express.js

express.js

express 모듈은 express 모듈 혹은 명령 라인으로 사용할 수 있다.

1. Express Generator

expres 명령을 이용해서 프로젝트를 생성하는 모듈로서 글로벌로 설치합니다.

```
$ npm install -g express-generator
```

2. Express module

- package.js
- require("express") 이용

Express generator

```
$ express -help
  Usage: express [options]
  Options:
    -h, --help          output usage information
    -V, --version       output the version number
    -s, --sessions      add session support
    -e, --ejs           add ejs engine support (defaults to jade)
    -J, --jshtml        add jshtml engine support (defaults to
jade)
    -c, --css <engine>  add stylesheet <engine> support (less|
stylus) (defaults to plain css)
    -f, --force         force on non-empty directory
```

hello express

```
$ mkdir nodejs
$ cd nodejs
$ express helloexpress
   create : helloexpress
  ...
```

그리고 프로젝트 폴더로 이동해 필요한 모듈을 설치합니다.

```
$ cd helloexpress
$ npm install
```

설치가 마무리되면 다음 같이 실행합니다.

```
$ npm start

or
$ node app.js
```

브라우저에서 http://IP_ADDRESS:3000 접속

app.js

```
1  var express = require('express');
2  var path = require('path');
3  var favicon = require('serve-favicon');
4  var logger = require('morgan');
5  var cookieParser = require('cookie-parser');
6  var bodyParser = require('body-parser');
7
8  var routes = require('./routes/index');
9  var users = require('./routes/users');
10
11 var app = express();
12
13 // view engine setup
14 app.set('views', path.join(__dirname, 'views'));
15 app.set('view engine', 'jade');
16
17 // uncomment after placing your favicon in /public
18 //app.use(favicon(path.join(__dirname, 'public', 'favicon.ico')));
19 app.use(logger('dev'));
20 app.use(bodyParser.json());
21 app.use(bodyParser.urlencoded({ extended: false }));
22 app.use(cookieParser());
23 app.use(express.static(path.join(__dirname, 'public')));
24
25 app.use('/', routes);
26 app.use('/users', users);
```

http://localhost:3000/
http://localhost:3000/users

app.js

```
28  // catch 404 and forward to error handler
29  app.use(function(req, res, next) {
30    var err = new Error('Not Found');
31    err.status = 404;
32    next(err);
33  });
34
35  // error handlers
36
37  // development error handler
38  // will print stacktrace
39  if (app.get('env') === 'development') {
40    app.use(function(err, req, res, next) {
41      res.status(err.status || 500);
42      res.render('error', {
43        message: err.message,
44        error: err
45      });
46    });
47  }
48
49  // production error handler
50  // no stacktraces leaked to user
51  app.use(function(err, req, res, next) {
52    res.status(err.status || 500);
53    res.render('error', {
54      message: err.message,
55      error: {}
56    });
57  });
```

routes

index.js

```
1  var express = require('express');
2  var router = express.Router();
3
4  /* GET home page. */
5  router.get('/', function(req, res, next) {
6    res.render('index', { title: 'Express' });
7  });
8
9  module.exports = router;
```

users.js

```
1  var express = require('express');
2  var router = express.Router();
3
4  /* GET users listing. */
5  router.get('/', function(req, res, next) {
6    res.send('respond with a resource');
7  });
8
9  module.exports = router;
```

books api

books OpenAPI

GET /api/books -------- 모든 목록
GET /api/books/{ID} -------- {ID} 도서 정보
POST /api/books
PUT /api/books
DELETE /api/books

"express hellobookapi" 로 생성 템플릿을 생성하고 내부 내용을 수정해 간다.

app.js

```
1  var express = require('express');
2  var path = require('path');
3  var favicon = require('serve-favicon');
4  var logger = require('morgan');
5  var cookieParser = require('cookie-parser');
6  var bodyParser = require('body-parser');
7
8  var routes = require('./routes/index');
9  var users = require('./routes/users');
10 // book api
11 var books = require('./routes/books');
12
```

```
21 app.use(logger('dev'));
22 app.use(bodyParser.json());
23 app.use(bodyParser.urlencoded({ extended: false }));
24 app.use(cookieParser());
25 app.use(express.static(path.join(__dirname, 'public')));
26
27 app.use('/', routes);
28 app.use('/users', users);
29 // uri prefix
30 app.use('/api', books);
31
```

routes/books.js

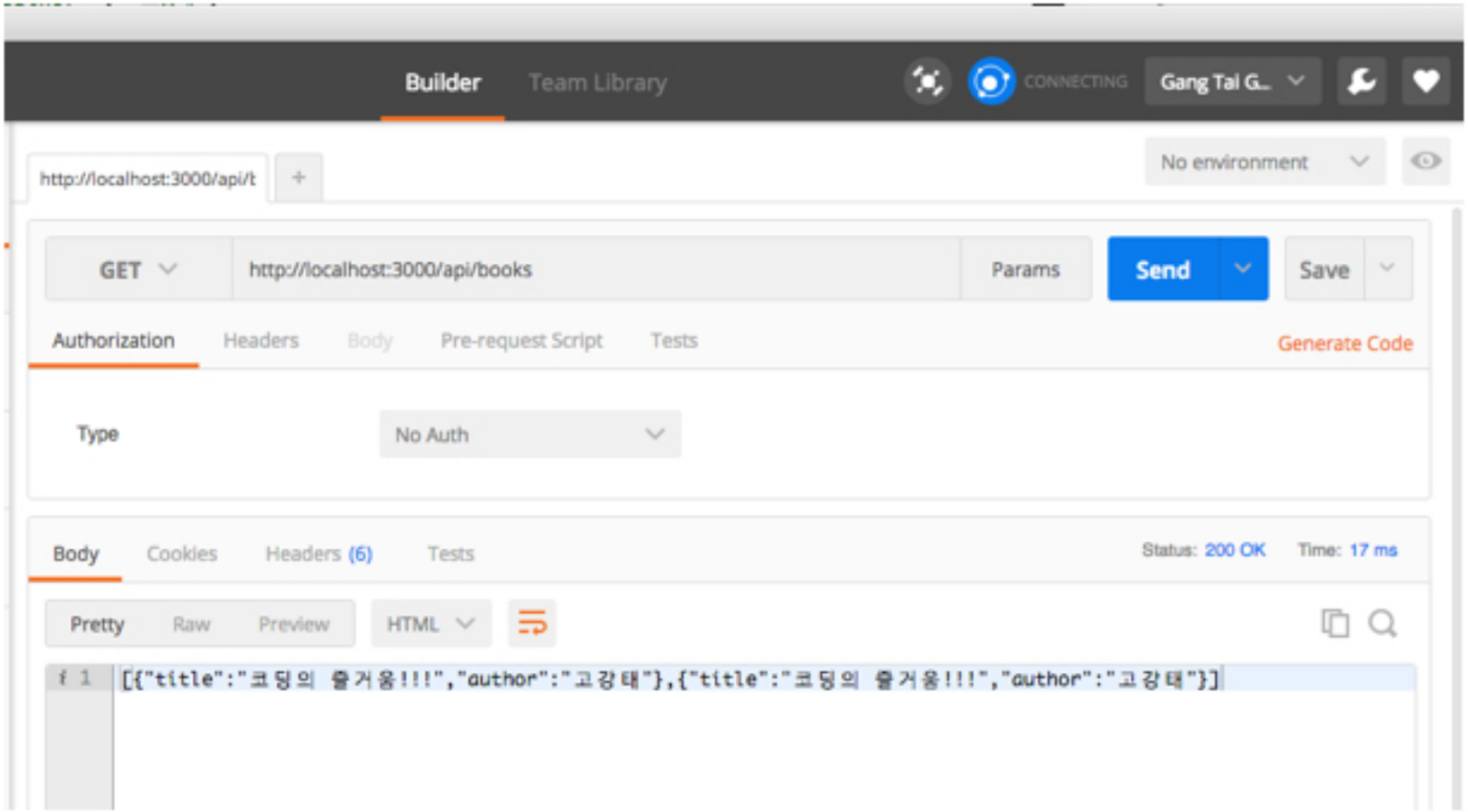
```
1  var express = require('express');
2  var router = express.Router();
3
4  var books = [{
5      "title": "코딩의 즐거움!!!",
6      "author" : "고강태",
7      },
8      {
9      "title": "코딩의 즐거움!!!",
10     "author" : "고강태",
11     }];
12
13 // API /api/books
14 router.route('/books')
15     .get( function(req, res) {
16         console.log( 'GET /api/books' );
17         res.send(JSON.stringify(books));
18     })
19     ;
20 // API /api/books/:id
21 router.route('/books/:id')
22     .get( function(req, res) {
23         console.log( 'GET /api/books/:id' );
24         res.send('Request by ID:' + req.params.id);
25     })
26     ;
27
28 module.exports = router;
```

http://localhost:3000/api/books

http://localhost:3000/api/books/1

Postman

수정후 **npm start**로 시작한다. 기본 3000번 포트를 사용해서 Postman을 사용해서 api 호출을 해본다.



expressjs api

http://expressjs.com/en/api.html

app.get(name)

Returns the value of `name` app setting, where `name` is one of strings in the app settings table. For example:

```
app.get('title');
// => undefined

app.set('title', 'My Site');
app.get('title');
// => "My Site"
```

app.get(path, callback [, callback ...])

Routes HTTP GET requests to the specified path with the specified callback functions. For more information, see the routing guide.

You can provide multiple callback functions that behave just like middleware, except these callbacks can invoke `next('route')` to bypass the remaining route callback(s). You can use this mechanism to impose pre-conditions on a route, then pass control to subsequent routes if there's no reason to proceed with the current route.

```
app.get('/', function (req, res) {
  res.send('GET request to homepage');
});
```

/api/sensors 전환 -1

앞서 helloexpress 처럼 새로운 express app을 생성한다.
"express hellosensors"

그리고 다음 사례를 참고해 "/api/sensors" API를 완성한다.

http://jungchul.tistory.com/304

books api

Post
Put
Delete

POST /api/books

```
12  var book = {
13      "title": "",
14      "author" : "",
15      "year" : 0
16  };
17  // API /api/books
18  router.route('/books')
19      .get( function(req, res) {
20          console.log( 'GET /api/books' );
21          res.send(JSON.stringify(books));
22      })
23      .post( function(req, res) {
24          console.log("Adding a Book: " + req.body.title );
25
26          book.title = req.body.title;
27          book.author = req.body.author;
28          book.year = req.body.year;
29          return res.send( book);
30      })
31      ;
```

book object

POST
http://localhost:3000/api/books

PUST, DELETE /api/books

```
32  // API /api/books/:id
33  router.route('/books/:id')
34      .get( function(req, res, next) {
35          console.log( 'GET /api/books/:id' );
36          res.send('Request by ID:' + req.params.id);
37      })
38      .put( function(req, res) {
39          book.title = req.body.title;  // update
40          return res.json({ title: book.title, message: 'Book updated!' });
41      })
42      .delete(function(req, res) {
43          book.title = req.body.title;  // update
44          return res.json( {message:"ID("+req.params.id+") Successfully deleted!"} );
45      })
46      ;
47
```

/api/sensors 전환 - 2

"express hellosensors" 에 POST, PUT, DELETE 를 다음 사례를 참고해 "/api/sensors" API를 완성한다.

http://jungchul.tistory.com/304

books renderer

jade

Jade Template engine

Jade Template engine
- express 프레임워크 기본 웹 템플릿 엔진
- HTML 태그보다 상당히 심플한 형태의 마크업
- 자동으로 HTML을 생성해 주는 역할
- http://jade-lang.com/

```
doctype 5
html(lang="en")
  head
    title= pageTitle
    script(type='text/javascript')
      if (foo) {
         bar()
      }
  body
    h1 Jade - node template engine
    #container
      if youAreUsingJade
        p You are amazing
      else
        p Get on it!
```

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Jade</title>
    <script type="text/javascript">
      if (foo) {
           bar()
      }
    </script>
  </head>
  <body>
    <h1>Jade - node template engine</
h1>
    <div id="container">
      <p>You are amazing</p>
    </div>
  </body>
</html>
```

routes/books.js

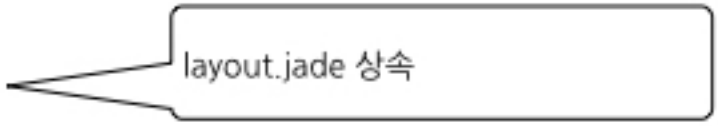
```
17  // API /api/books
18  router.route('/books')
19      .get( function(req, res) {
20          console.log( 'GET /api/books' );
21          //res.send(JSON.stringify(books));
22          res.render('books', { "title" : "GET /api/books",
23                                "books" : books});
24      })
25      .post( function(req, res) {
```

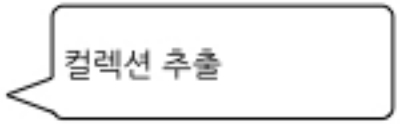
response 객체의 render에
books.jade 전달

http://expressjs.com/en/guide/routing.html

views/books.jade

```
1  extends layout
2
3  block content
4    h1= title
5    p Welcome to #{title}
6    ul
7    each val, index in books
8      li= index + ': ' + val.title
9
```





http://jade-lang.com/reference/iteration/

nodemon 과 forever

Runtime editing
Run as daemon

nodemon 이란?

노드에서 사이트를 개발할 경우에는 매번 노드를 재시작해야 합니다. 그런데 Remy Sharp가 제공하는 nodemon을 이용하면 이런 경우를 피할 수 있습니다.

- https://github.com/remy/nodemon

설치:

```
$[sudo] npm install -g nodemon
```

nodemon 사용

nodemon을 설치후 앱을 다음 같이 시작할 수 있다.

```
$ nodemon app.js
1 May 09:22:12 - [nodemon] v0.6.18
1 May 09:22:12 - [nodemon] watching: /Users/gtko/workspace_web/
Node_Lecture/WebContent/express_mysite
1 May 09:22:12 - [nodemon] starting `node app.js`
Express server listening on port 3000
```

이 앱을 개발하는 동안 app.js 관련 소스가 수정되면 자동으로 반영된다.

forever-monitor

forever-monitor는 시스템에 별도의 프로세서를 가지고,
'forever' 명령으로 노드 앱을 실행하면 차일드 프로세스로 실행해서 쉘을 나온 후에도 계속 실행할 수 있다.

```
$ [sudo] npm install forever -g
```

사용

```
usage: forever [start | stop | stopall | list | cleanlogs] [options] SCRIPT [script options]

  options:
    start          start SCRIPT as a daemon
    stop           stop the daemon SCRIPT
    stopall        stop all running forever scripts
    list           list all running forever scripts
    cleanlogs      [CAREFUL] Deletes all historical forever log files
```