

iOS 雲端服務開發

第四週

課程大綱

- 記錄檔管理
- 登入認證
- 密語
- 權限管理
- 註冊確認信
- Apple Push Notification

記錄檔管理 log4js

- [https://github.com/nomiddlename/
log4js-node](https://github.com/nomiddlename/log4js-node)
- 為什麼需要記錄檔
- appender
 - file
 - console
 - SMTP

安裝 log4js

#手動安裝

```
~$ npm install log4js
```



or add to package.json

使用 log4js

```
var log4js = require('log4js');
var logger = log4js.getLogger();

logger.trace('Entering cheese testing');
logger.debug('Got cheese.');
logger.info('Cheese is Gouda.');
logger.warn('Cheese is quite smelly.');
logger.error('Cheese is too ripe!');
logger.fatal('Cheese was breeding ground for listeria.');
```

```
[2013-06-21 12:13:27.622] [TRACE] [default] - Entering cheese testing
[2013-06-21 12:13:27.625] [DEBUG] [default] - Got cheese.
[2013-06-21 12:13:27.625] [INFO] [default] - Cheese is Gouda.
[2013-06-21 12:13:27.625] [WARN] [default] - Cheese is quite smelly.
[2013-06-21 12:13:27.625] [ERROR] [default] - Cheese is too ripe!
[2013-06-21 12:13:27.625] [FATAL] [default] - Cheese was breeding ground for listeria.
```

記錄層級

```
logger.setLevel(log4js.levels.WARN);
```

- 設定記錄層級：log4js.levels
- ALL, TRACE, DEBUG, INFO, WARN, ERROR, FATAL, OFF

記錄層級

```
var log4js = require('log4js');
var logger = log4js.getLogger();

logger.setLevel(log4js.levels.WARN);
logger.trace('Entering cheese testing');
logger.debug('Got cheese.');
logger.info('Cheese is Gouda.');
logger.warn('Cheese is quite smelly.');
logger.error('Cheese is too ripe!');
logger.fatal('Cheese was breeding ground for listeria.');
```

appender

- file, console, smtp

```
var log4js = require('log4js');
//console log is loaded by default, so you won't normally need to do
this
//log4js.loadAppender('console');
log4js.loadAppender('file');
//log4js.addAppender(log4js.appenders.console());
log4js.addAppender(log4js.appenders.file('cheese.log'), 'cheese');

var logger = log4js.getLogger('cheese');
logger.setLevel(log4js.levels.ERROR);

logger.trace('Entering cheese testing');
logger.debug('Got cheese.');
logger.info('Cheese is Gouda.');
logger.warn('Cheese is quite smelly.');
logger.error('Cheese is too ripe!');
logger.fatal('Cheese was breeding ground for listeria.');
```

log4js 設定

• log4js.configure(configurableObj)

```
var log4js = require('log4js');
log4js.configure({
  appenders: [
    { type: 'console' },
    { type: 'file', filename: 'logs/cheese.log', category: 'cheese' }
  ]
});

var logger = log4js.getLogger('cheese');
logger.setLevel(log4js.levels.WARN);

logger.trace('Entering cheese testing');
logger.debug('Got cheese.');
logger.info('Cheese is Gouda.');
logger.warn('Cheese is quite smelly.');
logger.error('Cheese is too ripe!');
logger.fatal('Cheese was breeding ground for listeria.');
```

configurableObj

```
{  
    appenders: [  
        {  
            "type": "file", //appender type  
            "absolute": false, //file path type  
            "filename": "log_file.log",  
            "maxLogSize": 20480,  
            "backups": 3, //number of logs to keep  
            "category": "absolute-logger"  
        },  
        {  
            "type": "console"  
        }  
],  
    replaceConsole: true,  
    levels: { //set log level by category  
        "absolute-logger": log4js.levels.WARN,  
        "[default)": log4js.levels.ERROR  

```

外部設定檔

```
#logger.json

{
  "appenders": [
    {
      "type": "file",
      "absolute": false,
      "filename": "log_file.log",
      "maxLogSize": 20480,
      "backups": 3,
      "category": "absolute-logger"
    },
    {
      "type": "console"
    }
  ],
  "replaceConsole": true,
  "levels": {
    "[default)": "ERROR"
  }
}
```

外部設定檔

```
#logger.js
```

```
var log4js = require('log4js');
log4js.configure("logger.json");
//log4js.configure("logger.json", {cwd: "外部設定檔目錄"});
```



```
var logger = log4js.getLogger("absolute-logger" );

logger.trace('Entering cheese testing');
logger.debug('Got cheese.');
logger.info('Cheese is Gouda.');
logger.warn('Cheese is quite smelly.');
logger.error('Cheese is too ripe!');
logger.fatal('Cheese was breeding ground for listeria.');
```

SMTP APPENDER

```
log4js.configure({
  "appenders": [
    {
      "type": "smtp",
      "recipients": "logfilerecipient@logging.com",
      "subject": "log mail subject", //信件主旨，預設為第一個事件訊息
      "sendInterval": 60, //發信間隔
      "transport": "SMTP",
      "SMTP": {
        "host": "smtp.gmail.com",
        "secureConnection": true,
        "port": 465,
        "auth": {
          "user": "someone@gmail",
          "pass": "*****"
        },
        "debug": true //顯示nodemailer的debug訊息
      },
      "category": "mailer"
    }
  ]
});
```



注意：須額外安裝nodemailer套件

進階設定

```
{  
  "replaceConsole": true,  
  "appenders": [  
    {  
      "type": "console"  
    },  
    {  
      "type": "logLevelFilter", // This is a recursive appender,  
                                // filters log messages and  
                                // sends them to itsown appender.  
      "level": "ERROR", // Include only error logs.  
      "appender": { // the filter's appender, smtp  
        "type": "smtp",  
        "recipients": "you@example.com",  
        "sender": "sender@serverhost.com",  
        "sendInterval": 60,  
        "transport": "SMTP",  
        "SMTP": {  
          "host": "localhost", // Other SMTP options here.  
          "port": 25  
        }  
      }  
    }  
  ]  
}
```

- 以console做log，ERROR層級以上訊息會額外發信

整合express

```
var log4js = require('log4js');
log4js.configure({
  appenders: [
    { type: 'console' },
    { type: 'file', filename: 'cheese.log', category: 'cheese' }
  ]
});
var logger = log4js.getLogger('cheese');
logger.setLevel('INFO');

var app = require('express').createServer();
app.configure(function() {
  app.use(log4js.connectLogger(logger, { level: log4js.levels.INFO }));
});
app.get('/', function(req,res) {
  res.send('hello world');
});
app.listen(5000);
```

messageBoard 整合

- package.json 加入log4js, nodemailer
- 設定檔
- 連接express logger
- 紀錄錯誤狀況

```
#config/log4js.json
{
  "replaceConsole": true,
  "appenders": [
    {
      "type": "console",
      "layout": {
        "type": "pattern",
        "pattern": "%[%d{yyyyMMdd hh:mm:ss} %p %] - %m"
      }
    },
    {
      "type": "file",
      "filename": "app.log",
      "maxLogSize": 20480,
      "backups": 3,
      "layout": {
        "type": "pattern",
        "pattern": "[%d{yyyyMMdd hh:mm:ss} %p ] - %m"
      }
    },
    {
      "type": "logLevelFilter",
      "level": "FATAL",
      "appender": {
        "type": "smtp",
        "recipients": "",
        "sendInterval": 5,
        "transport": "SMTP",
        "SMTP": {
          "host": "smtp.gmail.com",
          "secureConnection": true,
          "port": 465,
          "auth": {
            "user": "",
            "pass": ""
          }
        }
      }
    }
  ],
  "levels": {
    "[default)": "ERROR"
  }
}
```

連接express logger

```
#app.js

var log4js = require('log4js');

log4js.configure('config/log4js.json');
var logger = log4js.getLogger();

...

app.use(express.favicon());
app.use(log4js.connectLogger(logger, { level: 'auto' }));
app.use(express.cookieParser());
```

紀錄錯誤狀況

```
#routes/user.js
var log4js = require('log4js');
var logger = log4js.getLogger();

...
exports.register = function(req, res) {
  User.create({
    name: req.body.name,
    email: req.body.email,
    password: req.body.password,
    description: req.body.description
  }, function(err, user) {
    if (err) {
      logger.error(err);
      res.send(400, "Invalid data format or duplicate user
name");
      return;
    }
    console.info("Created user:", user.name);
    res.send("successfully created user:" + user.name);
  });
}
```

使用者認證 passportjs

- <http://passportjs.org/>
- 安裝
- 設定 Strategy
- 認證 authenticate

安裝 passport

#手動安裝

```
~$ npm install passport
```



or add to package.json

Strategy

- 登入方式 ex: FB, OAuth2, 本機登入等
- LocalStrategy
- LocalAPIStrategy

LocalStrategy 安裝

#手動安裝

```
~$ npm install passport-local
```

package.json

```
{  
  "name": "MessageBoard",  
  "version": "0.0.1",  
  "dependencies": {  
    ...  
    "passport-local": "*"  
  }  
}
```

Local Strategy

```
#/util/auth.js
var passport = require('passport')
, LocalStrategy = require('passport-local').Strategy;

passport.use(new LocalStrategy(function(username, password, done) {
  User.findOne({ name: username }, function(err, user) {
    if (err) { return done(err); }
    if (!user) {
      return done(null, false, {
        message: "Invalid username or password"
      });
    }
    user.comparePassword(password, function(err, isMatch) {
      if (err) { return done(err); }
      if (isMatch) { return done(null, user); }
      } else {
        return done(null, false, {
          message: "Invalid username or password"
        });
      }
    });
  });
}));

exports.ensureAuthenticated = function(req, res, next) {
  if (req.isAuthenticated()) { return next(); }
  return res.redirect("/logout");
};
```

Local Strategy

Express 設定

```
#app.js
var passport = require('passport');
var auth = require('./util/auth');

app.use(passport.initialize());
app.use(passport.session());

app.post('/messages', auth.ensureAuthenticated, message.createMessage);
app.get('/mymessage', auth.ensureAuthenticated, message.myMessage);
app["delete"]('/messages/:id', auth.ensureAuthenticated, message.removeMessage);
```

Local Strategy

登入&登出

```
#routes/user.js
exports.login = function(req, res, next) {
  passport.authenticate("local", function(err, user, info) {
    if (err) { return next(err); }
    if (!user) {
      req.session.messages = [info.message];
      return res.redirect("/login");
    }
    req.logIn(user, function(err) {
      if (err) { return next(err); }
      return res.redirect("/");
    });
  })(req, res, next);
};

exports.logout = function(req, res) {
  req.logout();
  res.redirect('/');
}
```

Local Strategy - View

```
#views/login.jade
extends layout

block content
  form(action="/login", method="POST")
    .control-group
      label.control-label(for='username') Username
      .controls
        input(type="text",name="username")

    .control-group
      label.control-label(for='password') Password
      .controls
        input(type="password",name="password")
    .control-group
      .controls
        if message
          .alert.alert-error
            button.close(type="button", data-dismiss="alert") &times;
            =message
    .control-group
      .controls
        input.btn.btn-primary(type="submit", value="登入")
```

LocalAPIStrategy

- for REST API > 不使用session
- 登入方式同localstrategy
- 登入後會獲得一個apikey，存在cookie中，往後連線使用apikey做認證
- USER model需要多一個apikey欄位
 - 亂數產生apikey
- 簡單的solution，建議使用oauth2

LocalAPIStrategy 安裝

#手動安裝

```
~$ npm install https://github.com/fifin/passport-localapikey/tarball/master
```

package.json

```
{  
  "name": "MessageBoard",  
  "version": "0.0.1",  
  "dependencies": {  
    ...  
    "passport-localapikey": "https://github.com/fifin/passport-localapikey/tarball/master"  
  }  
}
```

Local API Strategy

```
#models/user.js

var userSchema = new Schema({
  ...,
  apikey: {
    type: String,
    required: false,
    unique: true
  },
  description: String
});
```

Local API Strategy

```
#models/user.js -- continued

userSchema.pre('save', function(next) {
  var user;

  user = this;
  if (!user.isModified("password")) {
    return next();
  }
  return bcrypt.genSalt(SALT_WORK_FACTOR, function(err, salt) {
    if (err) {
      return next(err);
    }
    user.apikey = generateAPIKey();
    bcrypt.hash(user.password, salt, function(err, hash) {
      if (err) {
        return next(err);
      }
      user.password = hash;
      next();
    });
  });
});
```

Local API Strategy

```
#models/user.js -- continued

generateAPIKey = function() {
  var chars;

  chars = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890";
  return generateRandomToken(32, chars);
};

generateRandomToken = function(len, chars) {
  var charLen, i, token, x;

  charLen = chars.length;
  token = '';
  x = 0;
  while (x < len) {
    i = Math.floor(Math.random() * charLen);
    token += chars.charAt(i);
    x++;
  }
  return token;
};
```

LocalAPIStrategy

```
#util/auth.js

passport.use(new LocalAPIKeyStrategy(function(apikey, done) {
  User.findOne({
    apikey: apikey
  }, function(err, user) {
    if (err) { return done(err); }
    if (!user) { return done(null, false); }
    return done(null, user);
  });
}));

exports.authenticateAPIKey = authenticateAPIKey = function(req, res, next)
{
  passport.authenticate('localapikey', function(err, user, info) {
    if (err || !user) { return res.send(403); }
    return req.logIn(user, function(err) {
      if (err) {
        return next(err);
      }
    });
  })(req, res, next);
  return next();
};
```

Local Strategy

• Express 設定

```
#app.js

app.use('/api', auth.authenticateAPIKey); //一定在router之前
app.use(app.router);

//REST API
app.post('/apiLogin', user.apiLogin);
app.get('/apiLogout', user.apiLogout);
app.get('/api/messages', message.index);
```

Local Strategy

登入&登出

```
#routes/user.js
exports.apiLogin = function(req, res, next) {
  passport.authenticate("local", function(err, user, info) {
    if (err) { return next(err); }
    if (!user) { return res.send(403); }
    res.cookie('apikey', user.apikey, {
      maxAge: 86409000,
      httpOnly: true
    });
    return res.end();
  })(req, res, next);
};

exports.apiLogout = function(req, res, next) {
  res.clearCookie('apikey');
  res.end();
};
```

密語功能

- 訊息類別： 公開、密語
- 密語目標
- 僅傳送者與接收者能看到密語

修改 Message Model

```
#models/message.js

var mongoose = require('mongoose');
var Schema = mongoose.Schema;
var User = mongoose.model('User');

var messageSchema = new Schema({
  user: String,
  text: String,
  date: {type: Date, default: Date.now},
  type: {
    type: String,
    trim: true,
    enum: {
      "public", "private"
    }
  }
  target: String
});

var Message = mongoose.model('Message', messageSchema);
```

修改 Message View

```
#views/index.js

form(action="/messages", method="POST")
  select(name="type")
    option(value="public" selected) 公開
    option(value="private") 密語
  input#targetInput.hide(type="text", length="80", name="target",
placeholder="密語對象")
  input(type="text", length="140", name="text")
  input(type="submit", value="發送")
```

修改 Message View

```
#views/index.js

ul.messages
each message in messages
li
  form(action='/messages/#{message._id}', method='POST')
    input(type="submit", value="remove") 移除
    input(type="hidden", name="_method", value="delete")
  span.name
    if message.type == 'private' && message.user == user.name
      [密語] 
      給 
      a(href='/messages?name=#{message.target}')= message.target
    else if message.type == 'private' && message.target == user.name
      [密語] 
      a(href='/messages?name=#{message.user}')= message.user
      &nbsp;說
    else
      [公開] 
      a(href='/messages?name=#{message.user}')= message.user
  span.text= message.text
  span.date.pull-right= message.date
```

Message Controller

新增訊息

```
#routes/message.js

exports.createMessage = function(req, res) {
  Message.create({
    user: req.user.name,
    text: req.body.text,
    type: req.body.type,
    target: req.body.target
  }, function(err, message) {
    if (err) {
      logger.error(err);
      res.send(500, "Message creation error");
      return;
    }
    res.redirect(req.get('Referrer'));
  })
};
```

Message Controller

- 公開訊息列表

```
#routes/message.js

exports.index = function(req, res) {
  var pageSize = 10;

  var query = { $or: [ {type:"public"} ]};
  if ( req.user ) {
    query.$or.push({user: req.user.name});
    query.$or.push({target: req.user.name});
  }

  if ( req.query.name ) {
    query.user = req.query.name;
  }

  Message.find(query)...
```

Message Controller

- 我的訊息

```
#routes/message.js

exports.myMessage = function(req, res) {
  var pageSize = 10;

  Message.find({
    $or: [ {user: req.user.name}, {target: req.user.name} ]
  })
  ...
}
```

權限控管

- /login, /logout, /register, / : 公開
- /mymessage, POST /message : 登入使用者
- DELETE /messages/:id : 擁有者 & admin
- /users, DELETE /users/:id : admin
- /users/:id, /users/:id/edit, PUT /users/:id: 擁有者 & admin

權限控管

- 增加admin權限

```
#models/user.js

var userSchema = new Schema({
  ...
  admin: {
    type: Boolean,
    required: true,
    default: false
  }
});
```

權限控管

修改註冊頁面

```
#view/register.jade

form(action="/register", method="POST")
  ....
  .control-group
    label.control-label(for='description') Description
    .controls
      input(type="text", length="140", name="description")
  .control-group
    label.control-label(for='admin') Admin
    .controls
      select(name="admin")
        option(value="true") 是
        option(value="false") 否
  ....
```

權限控管

- /mymessage, POST /message : 登入使用者

```
#app.js

app.post('/messages', auth.ensureAuthenticated, message.createMessage);
app.get('/mymessage', auth.ensureAuthenticated, message.myMessage);
```

權限控管

- DELETE /messages/:id : 擁有者 & admin

```
#routes/message.js

exports.removeMessage = function(req, res) {
  var query = { _id: req.params.id, };
  if (req.user && !req.user.admin)
    query.user = req.user.name;

  Message.remove(query, function(err, messages) {
    if (err || !messages ) {
      console.log(err);
      res.send(500, "Cant remove such message");
      return;
    }
    res.redirect(req.get('Referrer'));
  });
};
```

權限控管

• /users, DELETE /users/:id : admin

```
#app.js
app.get('/users', auth.ensureAdmin, user.list);
app["delete"]('/users/:name', auth.ensureAdmin, user.remove);

#util.auth.js
...
exports.ensureAdmin = ensureAdmin = function(req, res, next) {
  if (req.user && req.user.admin === true) {
    return next();
  } else {
    return res.send(403);
  }
};
```

權限控管

- /users/:id, /users/:id/edit, PUT /
users/:id: 擁有者 & admin

```
#app.js
app.get('/users/:name', auth.ensureSelfOrAdmin, user.detail);
app.get('/users/:name/edit', auth.ensureSelfOrAdmin, user.editView);
app.put('/users/:name', auth.ensureSelfOrAdmin, user.edit);

#util.auth.js
...
exports.ensureSelfOrAdmin = function(req, res, next) {
  name = req.params.name;
  if ( req.user && (req.user.admin === true || req.user.name === name) ) {
    return next();
  } else {
    return res.send(403);
  }
};
```

權限控管

- 其他 library
 - [https://github.com/OptimalBits/
node_acl](https://github.com/OptimalBits/node_acl)
 - <https://github.com/chakrit/simple-acl>

註冊確認信 nodemailer

- <https://github.com/andris9/Nodemailer>
- 安裝
- 設定
- 整合註冊流程

註冊確認信 nodemailer

#手動安裝

```
~$ npm install nodemailer
```

or add to package.json

設定傳輸協定

```
// var transport = nodemailer.createTransport("SMTP", {smtp_options});  
var smtpTransport = nodemailer.createTransport("SMTP", {  
  service: "Gmail",  
  auth: {  
    user: "gmail.user@gmail.com",  
    pass: "userpass"  
  },  
  //host: "smtp.google.com"      //not needed with service  
  //"secureConnection": true,   //not needed with service  
  //"port": 465,                //not needed with service  
  //"debug": true               //output client and server message to console  
});
```

發信

```
var mailOptions = {
  from: "Fred Foo ✓ <foo@blurdybloop.com>", // sender address
  to: "bar@blurdybloop.com, baz@blurdybloop.com", // list of receivers
  subject: "Hello ✓", // Subject line
  text: "Hello world ✓", // plaintext body
  html: "<b>Hello world ✓</b>" // html body
}

// send mail with defined transport object
smtpTransport.sendMail(mailOptions, function(error, response){
  if(error){ console.log(error); }
  else{ console.log("Message sent: " + response.message); }
  smtpTransport.close();
});
```

發信

```
var nodemailer = require("nodemailer");

// create reusable transport method (opens pool of SMTP connections)
var smtpTransport = nodemailer.createTransport("SMTP",{
  service: "Gmail",
  auth: {
    user: "gmail.user@gmail.com",
    pass: "userpass"
  }
});

// setup e-mail data with unicode symbols
var mailOptions = {
  from: "Fred Foo ✓ <foo@blurdybloop.com>", // sender address
  to: "bar@blurdybloop.com, baz@blurdybloop.com", // list of receivers
  subject: "Hello ✓", // Subject line
  text: "Hello world ✓", // plaintext body
  html: "<b>Hello world ✓</b>" // html body
}

// send mail with defined transport object
smtpTransport.sendMail(mailOptions, function(error, response){
  if(error){ console.log(error); }
  else{ console.log("Message sent: " + response.message); }
  smtpTransport.close();
});
```

留言板註冊流程

- 註冊帳號
- 系統創造帳號資料 & 驗證碼，設定帳號為尚未啓用
- 寄發驗證碼至使用者信箱
- 點擊email內含之link啓用帳號或是於app中輸入驗證碼啓用帳號
- 正確的話即設定帳號為啓用

package.json

```
{  
  "name": "MessageBoard",  
  "version": "0.0.1",  
  "dependencies": {  
    "express": "*",  
    "jade": "*",  
    "mongoose": "*",  
    "bcrypt": "*",  
    "log4js": "*",  
    "nodemailer": "*",  
    "passport": "*",  
    "passport-local": "*",  
    "passport-localapikey": "https://github.com/fifin/passport-localapikey/tarball/master",  
    "underscore": "*"  
  }  
}
```

User Model

- 多出啓用、驗證碼兩個欄位
- 創建使用者時亂數產生驗證碼

```
#models/user.js

var userSchema = new Schema({
  ...
  token: String,
  enabled: { type: Boolean, required: true }
});

userSchema.pre('save', function(next) {
  ...
  user.apikey = generateAPIKey();
  if (user.enabled) { user.token = null; }
  else { user.token = generateValidationToken(); }
  bcrypt.hash(user.password, salt, function(err, hash) {
    ...
  });
  ...
});

generateValidationToken = function() {
  var chars;

  chars = "1234567890";
  return generateRandomToken(8, chars);
};
```

Mailer Config

寄信設定檔

```
#config/mailer.json

{
  "smtpOption": {
    "service": "Gmail",
    "auth": {
      "user": "",
      "pass": ""
    },
    "debug": true
  },
  "mailOption": {
    "subject": "MessageBoard Confirmation Mail",
    "template": "confirmation",
    "generateTextFromHTML": true
  }
}
```

信件範本

```
#views-mails/confirmation.html

<html>
  <head>
    <title>MessageBoard Confirmation Mail</title>
  </head>
  <body>
    <p>
      Hi {{name}},
    </p>
    <p>
      非常歡迎您註冊本程式，請點擊以下連結啓動您的帳號：  

      <a href="http://{{host}}/validate?token={{token}}">http://
{{host}}/validate?token={{token}}</a> <br/>
      或是登入App後輸入您的驗證碼： {{token}}
    </p>
    <p>
      App Team
    </p>
  </body>
</html>
```

Emailer

```
#util/emailer.js

var emailer = require("nodemailer");
var fs = require("fs");
var _ = require("underscore");
var config = require("../config/mail");

var Emailer = function (options, data) {
  this.options.mailOption = _.extend(config.mailOption, options) || {};
  console.log(this.options);
  this.data = data || {};
  this.attachments = [];
}

Emailer.prototype.options = config;
Emailer.prototype.send = function(callback) {
  var attachments, html, messageData, transport;

  this.options.mailOption.html = this.getHtml();
  transport = this.getTransport();
  return transport.sendMail(this.options.mailOption, callback);
};

Emailer.prototype.getTransport = function() {
  return emailer.createTransport("SMTP", this.options.smtpOption);
};

Emailer.prototype.getHtml = function() {
  var encoding, templateContent, templatePath;
  var templateName = this.options.mailOption.template;

  templatePath = "./views/emails/" + templateName + ".html";
  templateContent = fs.readFileSync(templatePath, "utf8");
  return _.template(templateContent, this.data, {
    interpolate: /\{\{\{(.+?)\}\}\}/g
  });
};

exports = module.exports = Emailer;
```

註冊

```
#routes/user.js

exports.register = function(req, res) {
  User.create({
    ...
    res.send("successfully created user:" + user.name);

    var options = { to: user.email };

    var data = {
      name: user.name,
      token: user.token,
      host: req.headers.host
    };

    var mailer = new Mailer(options, data);
    mailer.send(function(err, result) {
      if(err) {
        return logger.error(err);
      }
    });
  });
}
```

驗證

```
#routes/user.js

exports.validate = function(req, res) {
  var token = req.query.token;
  return User.findOneAndUpdate({
    token: token
  }, {
    enabled: true,
    token: null
  }, function(err, user) {
    if (err) {
      return res.send(500);
    }
    if (!user) {
      return res.send(403);
    }
    req.session.messages = ["User: " + user.name + " validated, please
login."];
    return res.redirect('/login')
  });
};
```

權限設定

```
#util/auth.js

exports.ensureAuthenticated = ensureAuthenticated = function(req, res, next) {
  if (req.isAuthenticated() && req.user.enabled) {
    ...
  };
}

exports.ensureAdmin = ensureAdmin = function(req, res, next) {
  if (req.user && req.user.enabled && req.user.admin === true) {
    ...
  };
}

exports.ensureSelfOrAdmin = function(req, res, next) {
  name = req.params.name;
  if (req.user && req.user.enabled && (req.user.admin === true ||
  req.user.name === name)) {
    ...
  };
}
```

express routes

```
#app.js  
  
app.get('/validate', user.validate);
```

APN

- <https://github.com/argon/node-apn>
- 安裝
- 整合messageBoard

APN 安裝

#手動安裝

```
~$ npm install apn
```

or add to package.json

整合 messageBoard

- 登入時傳入UUID
- 密語時如果target有UUID的話則傳送apn訊息

User Model

```
#models/user.js

var userSchema = new Schema({
  ...
  UUID: String
});
```

Login Page

```
form(action="/login", method="POST")
  .control-group
    label.control-label(for='username') Username
    .controls
      input(type="text",name="username")

  .control-group
    label.control-label(for='password') Password
    .controls
      input(type="password",name="password")
  .control-group
    label.control-label(for='UUID') UUID
    .controls
      input(type="text",name="UUID")
```

Login Controller

```
#routes/user.js

exports.login = function(req, res, next) {
  passport.authenticate("local", function(err, user, info) {
    ...
    return req.logIn(user, function(err) {
      if (err) {
        return next(err);
      }
      user.UUID = req.body.UUID;
      user.save(function(err) {
        if(err) {
          logger.error(err);
        }
        res.redirect("/");
      });
    });
  })(req, res, next);
};
```

Message Controller

```
#routes/message.js

var apn = require('../util/apnhandler');
...

exports.createMessage = function(req, res) {
  Message.create({
    }, function(err, message) {
      if (err) {
        logger.error(err);
        res.send(500, "Message creation error");
        return;
      }

      if (message.type === "private") {
        User.findOne({ name: message.target }, function(err, user) {
          if(err || !user) {
            logger.error(err);
            res.send(500, "error");
            return;
          }
          if (user.UUID) {
            apn.notification(user.UUID, message.text);
          }
          res.redirect(req.get('Referrer'));
        });
      } else
        res.redirect(req.get('Referrer'));
    })
};

};
```

apn handler

```
#util/apnhandler.js
```

```
var options = {  
    cert: 'config/cert.pem',  
    key: 'config/key.pem',  
    gateway: 'gateway.sandbox.push.apple.com', /* gateway address */  
    port: 2195,  
    errorCallback: function(err) {logger.error(err)}, /*Callback when error occurs*/  
};  
  
var apnsConnection = new apns.Connection(options);  
  
exports.notification = function(token, message) {  
    var device;  
    if (util.isArray(token)) {  
        device = [];  
        token.forEach(function(e) {  
            device.push(new apns.Device(e));  
        });  
    } else {  
        device = new apns.Device(token);  
    }  
    var note = new apns.Notification();  
    note.expiry = Math.floor(Date.now() / 1000) + 3600; // Expires 1 hour from now.  
    note.badge = 1;  
    note.sound = "ping.aiff";  
    note.alert = message;  
    note.payload = {'messageFrom': 'Caroline'};  
  
    apnsConnection.pushNotification(note, device);  
}
```

產生pem檔

```
$ openssl x509 -in cert.cer -inform DER -outform PEM -out cert.pem  
$ openssl pkcs12 -in key.p12 -out key.pem -nodes
```