# 实验9：express+mongodb制作blog

实验背景

利用已掌握的知识，完成实例。如图所示：

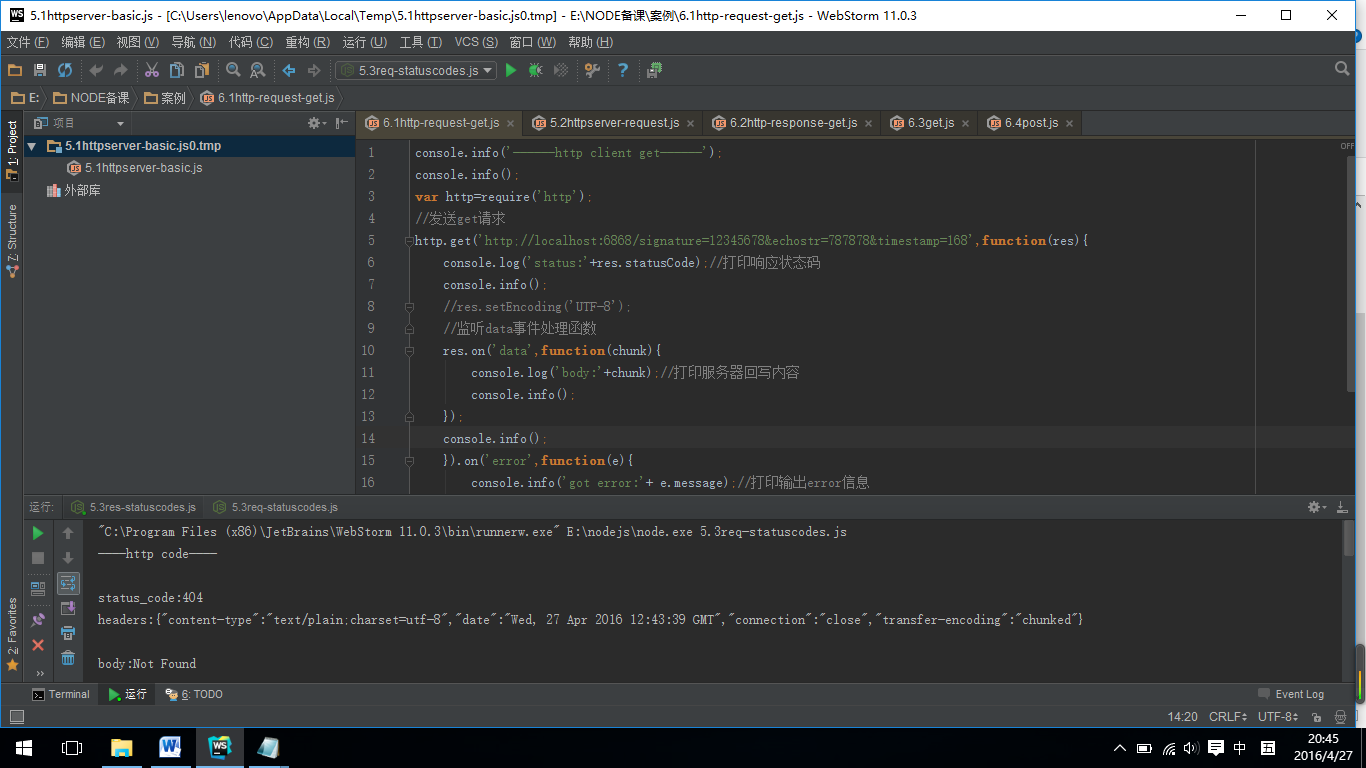


图1-1

实验目的

1．掌握express

2．掌握mongodb

3．掌握通过express+mongodb制作blog

实验分析

本次实验要完成搭建一个简单的具有多人注册、登录、发表文章、登出功能的博客。

实验步骤

1、结构设计

未登录：主页左侧导航显示 home、login、register，右侧显示已发表的文章、发表日期及作者。  
登陆后：主页左侧导航显示 home、post、logout，右侧显示已发表的文章、发表日期及作者。  
用户登录、注册、发表成功以及登出后都返回到主页。

**①未登录**：

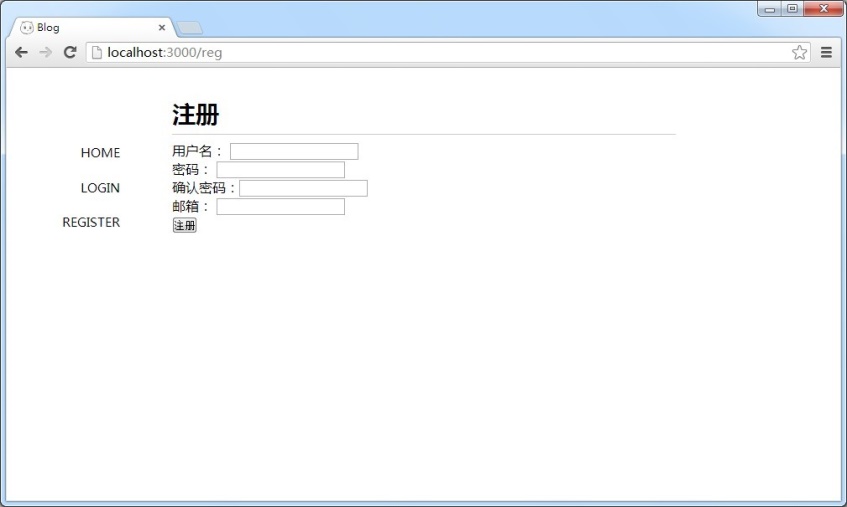
主页：



登录页：



注册页：



**②登录后**：

主页:



发表页：



**注意**：没有登出页，当点击 LOGOUT 后，退出登陆并返回到主页。

2、**路由规划**

我们已经把设计的构想图贴出来了，接下来的任务就是完成路由规划了。路由规划，或者说控制器规划是整个网站的骨架部分，因为它处于整个架构的枢纽位置，相当于各个接口之间的粘合剂，所以应该优先考虑。

根据构思的设计图，作以下路由规划：

/ ：首页

/login ：用户登录

/reg ：用户注册

/post ：发表文章

/logout ：登出

我们要求 /login 和 /reg 只能是未登录的用户访问，而 /post 和 /logout 只能是已登录的用户访问。左侧导航列表则针对已登录和未登录的用户显示不同的内容。

修改 index.js 如下：

module.exports = function(app) {

app.get('/', function (req, res) {

res.render('index', { title: '主页' });

});

app.get('/reg', function (req, res) {

res.render('reg', { title: '注册' });

});

app.post('/reg', function (req, res) {

});

app.get('/login', function (req, res) {

res.render('login', { title: '登录' });

});

app.post('/login', function (req, res) {

});

app.get('/post', function (req, res) {

res.render('post', { title: '发表' });

});

app.post('/post', function (req, res) {

});

app.get('/logout', function (req, res) {

});

};

3、**使用mongodb数据库**

①安装MongoDB

安装 MongoDB 很简单,去官网下载对应系统的 MongoDB 压缩包即可。解压后将文件夹重命名为 mongodb，并在 mongodb 文件夹里新建 blog 文件夹作为我们博客内容的存储目录。进入到 bin 目录下：运行：

.mongod --dbpath ../blog/

以上命令的意思是:设置blog 文件夹作为我们工程的存储目录并启动数据库。

②连接MongoDB

数据库虽然安装并启动成功了，但我们需要连接数据库后才能使用数据库。怎么才能在 Node.js 中使用 MongoDB 呢？我们使用官方提供的 node-mongodb-native 驱动模块，打开 package.json，在 dependencies 中添加一行：

"mongodb": "1.4.15"

③更新依赖的模块

然后运行 npm install 更新依赖的模块，稍等片刻后 mongodb 模块就下载并安装完成了。

接下来在工程的根目录中创建 settings.js 文件，用于保存该博客工程的配置信息，比如数据库的连接信息。我们将数据库命名为 blog，因为数据库服务器在本地，所以 settings.js 文件的内容如下：

module.exports = {

cookieSecret: 'myblog',

db: 'blog',

host: 'localhost',

port: 27017

};

其中 db 是数据库的名称，host 是数据库的地址，port是数据库的端口号，cookieSecret 用于 Cookie 加密与数据库无关，我们留作后用。

接下来在根目录下新建 models 文件夹，并在 models 文件夹下新建 db.js ，添加如下代码：

var settings = require('../settings'),

Db = require('mongodb').Db,

Connection = require('mongodb').Connection,

Server = require('mongodb').Server;

module.exports = new Db(settings.db, new Server(settings.host, settings.port),

{safe: true});

其中通过 new Db(settings.db, new Server(settings.host, settings.port), {safe: true}); 设置数据库名、数据库地址和数据库端口创建了一个数据库连接实例，并通过 module.exports 导出该实例。这样，我们就可以通过 require 这个文件来对数据库进行读写了。

打开 app.js，在 var routes = require('./routes/index'); 下添加：

var settings = require('./settings');

4、会话支持

express 也提供了会话中间件，默认情况下是把用户信息存储在内存中，但我们既然已经有了 MongoDB，不妨把会话信息存储在数据库中，便于持久维护。为了使用这一功能，我们需要借助 express-session 和 connect-mongo 这两个第三方中间件，在package.json 中添加：

"express-session": "1.9.1",

"connect-mongo": "0.4.1"

注意：如报"error setting ttl index on collection : sessions"错误，把"mongodb"&"connect-mongo"版本号更到最新。

运行npm install安装模块,打开app.js，添加以下代码：

var session = require('express-session');

var MongoStore = require('connect-mongo')(session);

app.use(session({

secret: settings.cookieSecret,

key: settings.db,//cookie name

cookie: {maxAge: 1000 \* 60 \* 60 \* 24 \* 30},//30 days

store: new MongoStore({

db: settings.db,

host: settings.host,

port: settings.port

})

}));

注意： connect-mongo 最新版需要改成如：

store: new MongoStore({

url: 'mongodb://localhost/blog'

})

扩展练习

1.思考并实现，完成案例《6.2 http-response-get》并实现测试

参考学习资料：详见案例《6.2 http-response-get》源文件

实验思考

1.思考并实现，计数器案例。

参考代码：