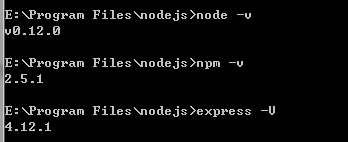
**参考：**[**http://www.cnblogs.com/Leo\_wl/p/4361289.html**](http://www.cnblogs.com/Leo_wl/p/4361289.html)

[**node.js基于express框架搭建一个简单的注册登录Web功能**](http://www.cnblogs.com/imwtr/p/4360341.html)

**这个小应用使用到了node.js  bootstrap  express  以及数据库的操作 ：使用mongoose对象模型来操作 mongodb**

**如果没了解过的可以先去基本了解一下相关概念~**

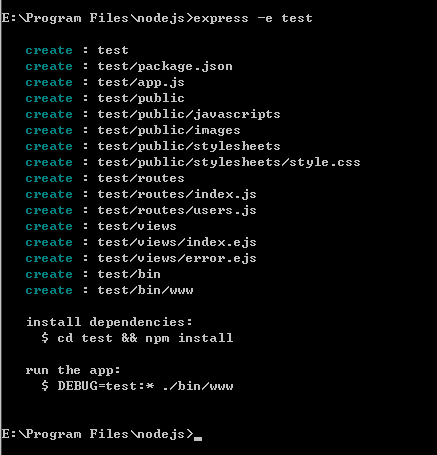
**首先注明一下版本，因为express因为版本的不同使用的方式也不同，我这算是目前最新的了吧**



**还没有装express的可以移步到**[**这里**](http://www.cnblogs.com/imwtr/p/4350282.html)**看看express框架的获取安装**

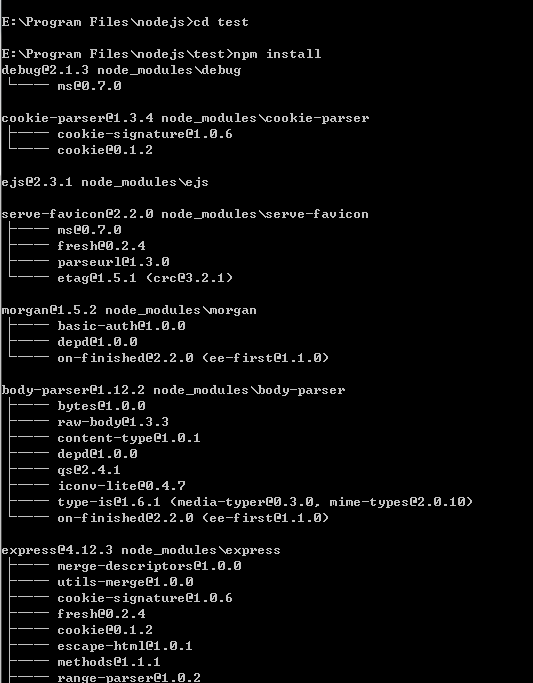
**1.简单地项目初始化**

进入你的nodejs安装路径下边，如图，然后执行命令 **express -e test**  (这里把项目名设置为test)

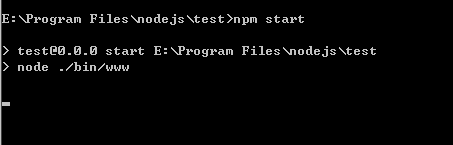


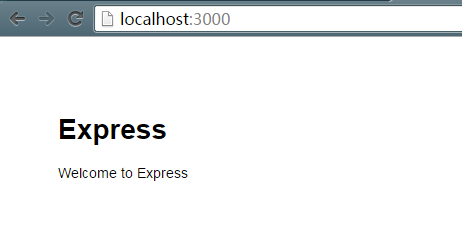
出现如上图所示，看到install dependencies没有，它说如果你想安装依赖就先进入项目test目录，然后执行 npm install安装依赖模块。

那就开始吧，网络环境差的可能安装会出错..出现很长一大串一般就行了

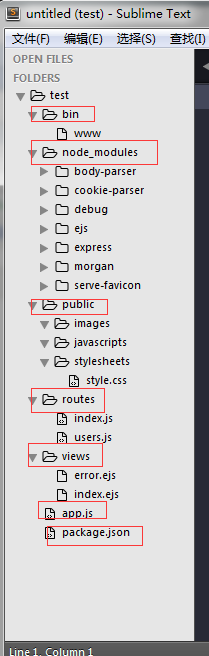
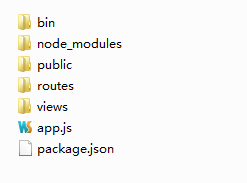


如此一来，项目初始已经完成，可以运行一下项目 npm start 看是否正常。





**ok 还算正常，下面先来基本分析一下生成的初始项目**：

之前 [**那篇文章**](http://www.cnblogs.com/imwtr/p/4350282.html) 已经说过

项目创建成功之后，生成四个文件夹，主文件app.js与配置信息文件packetage.json

**bin是项目的启动文件**，配置以什么方式启动项目，默认 npm start

**public是项目的静态文件**，放置js css img等文件

**routes是项目的路由信息文件**,控制地址路由

**views是视图文件**，放置模板文件ejs或jade等（其实就相当于html形式文件啦~)

**express这样的MVC框架模式，是一个Web项目的基本构成**。

先来看看文件信息package.json  一般项目的主要信息都会在这里产生

**[复制代码](javascript:void(0);)**

**复制代码**

**{**

**"name": "test",**

**"version": "0.0.0",**

**"private": true,**

**"scripts": {**

**"start": "node ./bin/www"**

**},**

**"dependencies": {**

**"body-parser": "~1.12.0",**

**"cookie-parser": "~1.3.4",**

**"debug": "~2.1.1",**

**"ejs": "~2.3.1",**

**"express": "~4.12.2",**

**"morgan": "~1.5.1",**

**"serve-favicon": "~2.2.0"**

**}**

**}**

**复制代码**

**[复制代码](javascript:void(0);)**

看看主文件 app.js   这是它的初始形式，这个模块还要继续导出给 bin文件夹下的www文件使用

**[复制代码](javascript:void(0);)**

**复制代码**

**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', 'ejs');**

**16**

**17 // uncomment after placing your favicon in /public**

**18 //app.use(favicon(\_\_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);**

**27**

**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 });**

**58**

**59**

**60 module.exports = app;**

**复制代码**

**[复制代码](javascript:void(0);)**

www文件内容：这里拥有着http服务器的基本配置

**http://images.cnblogs.com/OutliningIndicators/ContractedBlock.gif View Code**

再来介绍一下项目使用到的ejs模板，比如看看这个view里边的index.ejs （我们待会可以直接把它转为html，差不多的）

**[复制代码](javascript:void(0);)**

**复制代码**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<title><%= title %></title>**

**<link rel='stylesheet' href='/stylesheets/style.css' />**

**</head>**

**<body>**

**<h1><%= title %></h1>**

**<p>Welcome to <%= title %></p>**

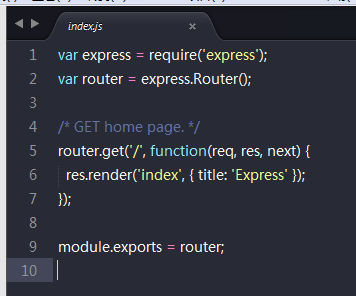
**</body>**

**</html>**

**复制代码**

**[复制代码](javascript:void(0);)**

<%= title %> 这就是ejs的使用范例，title的值通过路由routes文件夹下index.js代码传入（后面再谈）



好了，基本介绍了项目的初始情况

**2.基于初始项目的改进-- 注册登录功能**

**设计如下：**

一个初始界面（其实就是原始地址：比如 localhost:3000(index.html  路径为/ ) ，在初始界面选择登录或注册

跳进来之后会先跳进登录界面（login.html  路径为 /login)，可以选择先注册（跳转 register.html  路径为/register)

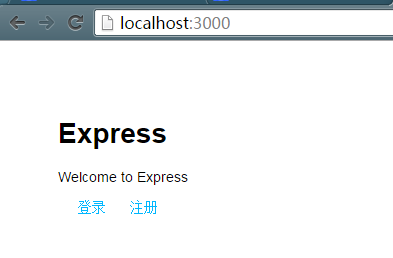
跳进注册界面后就会跳进（register.html 路径为 /register)，注册成功后就跳转登录界面(login.html  路径为 /login)

在登录界面登录成功后就跳转(home.html  路径为 /home). 在home这里还提供了注销的功能（无页面文件，它的路径为 /logout

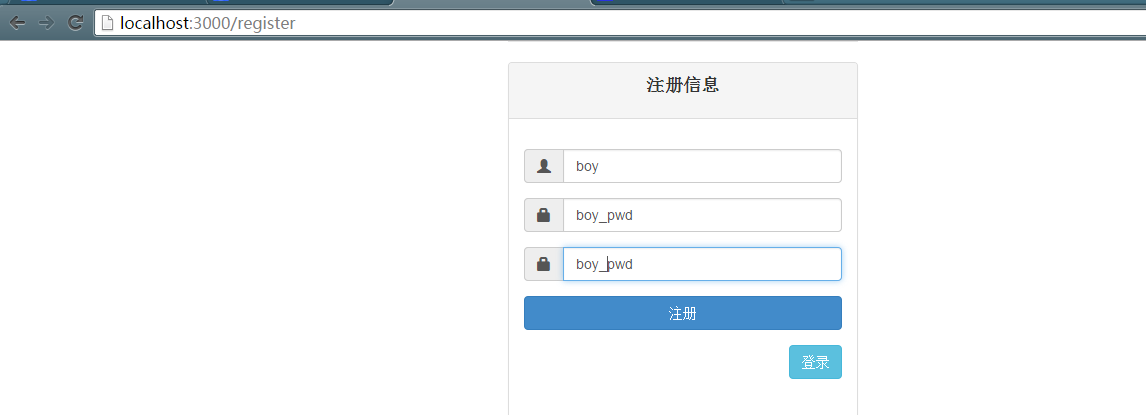
如果浏览器直接输入localhost:3000/home  要先判断是否登录成功，未登录不允许进入

**看到上诉，应该了解到：我们是通过一个路径，然后通过这个路径的解析，从而渲染出这个路径对应的模板文件，其中我们这里的模板文件为.html后缀的**

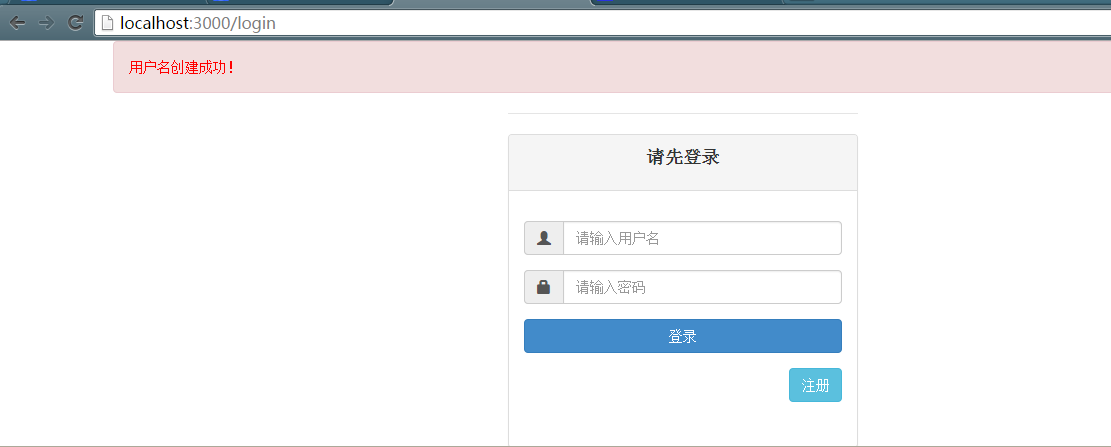
**首先展示一下基本界面形态：**

****

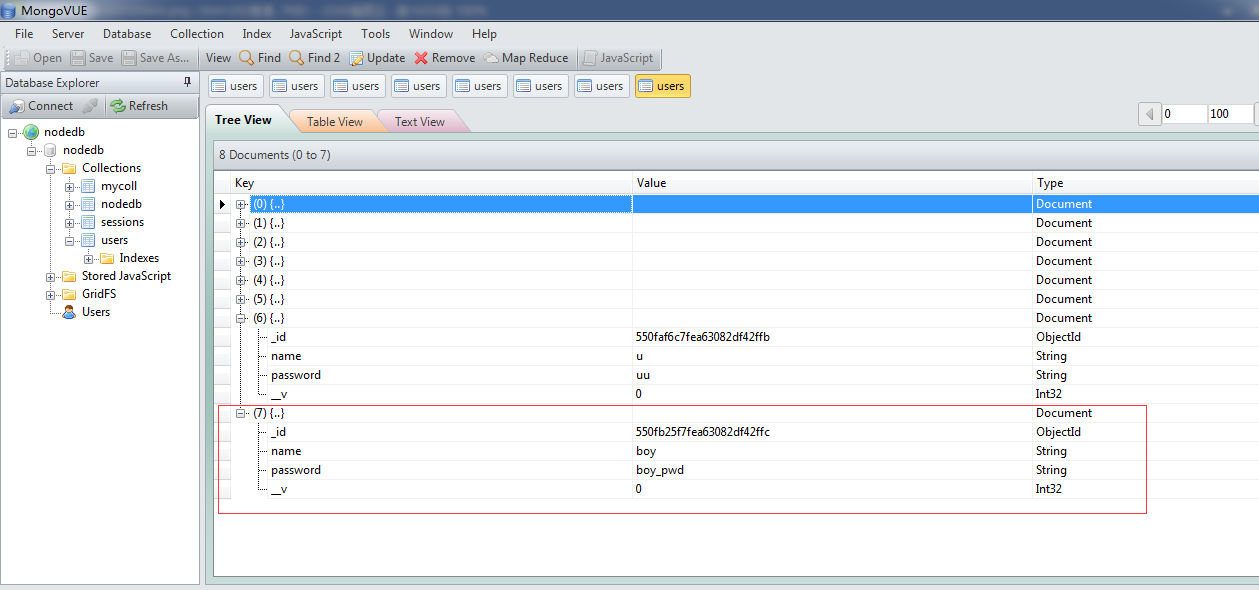
然后先注册吧，点击注册



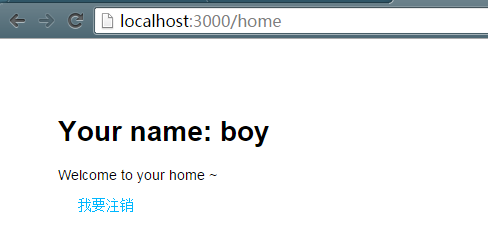
填入用户名密码，这里稍微设置了两次密码相同的判断，注册成功它会自动跳转登录界面



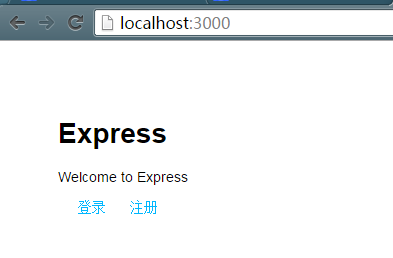
用mongoVUE看看数据的创建



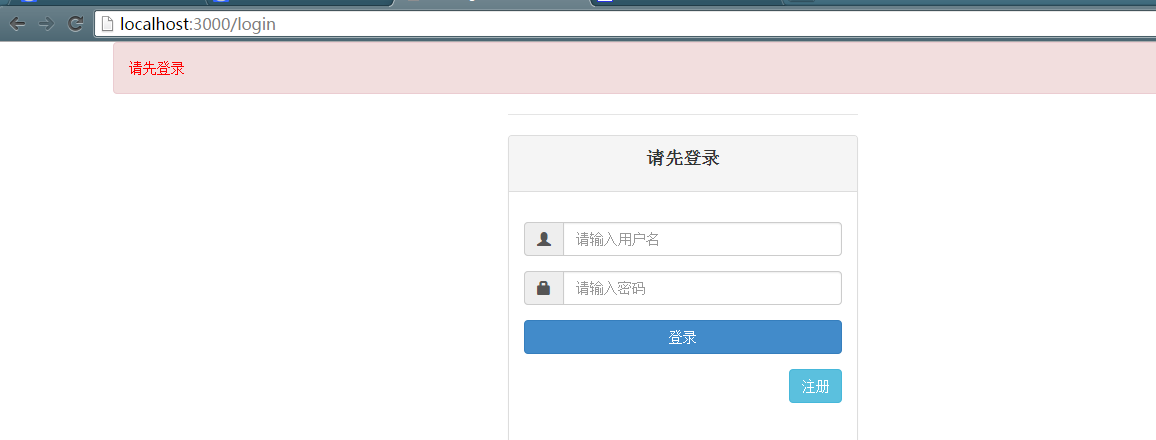
那就登录吧，登录成功跳转home界面



注销吧，注销后清除session值，然后跳转到根路径



然后试一下浏览器直接进入 home路径？ 浏览器地址输入  localhost:3000/home  回车， ok 它自动跳转到登录界面



**好现在开始解析如何构建这个小项目：**

因为我们直接使用了后缀名 .html ，所以我们要先修改一下ejs模板  ，再把原来views目录下模板文件后缀改成 .html

**[复制代码](javascript:void(0);)**

**复制代码**

**var app = express();**

**// view engine setup**

**app.set('views', path.join(\_\_dirname, 'views'));**

**app.engine("html",require("ejs").\_\_express); // or app.engine("html",require("ejs").renderFile);**

**//app.set("view engine","ejs");**

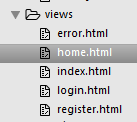
**app.set('view engine', 'html');**

**复制代码**

**[复制代码](javascript:void(0);)**

其实就是加一句再改一句。 \_\_express 和renderFile都可以，它们可以修改模板文件的类型， 不用管它是什么，它能那样用就行了

然后我们知道需要这些模板文件，那就创建它们吧



**index.html    其中 <%= title %>使用到了模板  连接<a> 直接使用了路由路径的方法**

**[复制代码](javascript:void(0);)**

**复制代码**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<title><%= title %></title>**

**<link rel='stylesheet' href='/stylesheets/style.css' />**

**<style type="text/css">**

**a{margin-left: 20px; text-decoration: none;}**

**a:hover{text-decoration: underline;}**

**</style>**

**</head>**

**<body>**

**<h1><%= title %></h1>**

**<p>Welcome to <%= title %></p>**

**<p><a href="/login">登录 </a>**

**<a href="/register"> 注册</a>**

**</p>**

**</body>**

**</html>**

**复制代码**

**[复制代码](javascript:void(0);)**

**register.html  注册方式主要是把原始 form表单 onsubmit="return false" 防止默认提交，然后在输入信息正确的情况下，通过ajax,把表单信息post到路径/register**

**然后我们就通过路由功能根据此路径来处理信息（这个跟ajax和php交互是同一个道理）**

**[复制代码](javascript:void(0);)**

**复制代码**

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<meta charset="utf-8">**

**<meta name="viewport" content="width=device-width,initial-scale=1.0">**

**<title><%= title %></title>**

**<link rel="stylesheet" href="stylesheets/bootstrap.min.css" media="screen">**

**<style type="text/css">**

**.m15{ margin: 15px;}**

**.tc{ text-align: center;font-size: 18px;font-weight: 600;}**

**</style>**

**</head>**

**<body screen\_capture\_injected="true">**

**<div class="container">**

**<%- message %>**

**<form class="col-sm-offset-4 col-sm-4 form-horizontal" role="form" method="post" onsubmit="return false">**

**<fieldset>**

**<legend></legend>**

**<div class="panel panel-default">**

**<div class="panel-heading">**

**<p class="tc">注册信息</p>**

**</div>**

**<div class="panel-body m15">**

**<div class="form-group">**

**<div class="input-group">**

**<span class="input-group-addon">**

**<span class="glyphicon glyphicon-user"></span>**

**</span>**

**<input type="text" class="form-control" id="username" name="username" placeholder="请输入用户名" required>**

**</div>**

**</div>**

**<div class="form-group">**

**<div class="input-group">**

**<span class="input-group-addon">**

**<span class="glyphicon glyphicon-lock"></span>**

**</span>**

**<input type="text" class="form-control" id="password" name="password" placeholder="请输入密码" required>**

**</div>**

**</div>**

**<div class="form-group">**

**<div class="input-group">**

**<span class="input-group-addon">**

**<span class="glyphicon glyphicon-lock"></span>**

**</span>**

**<input type="text" class="form-control" id="password1" name="password1" placeholder="请再次输入密码" required>**

**</div>**

**</div>**

**<div class="form-group">**

**<button type="submit" class="btn btn-primary btn-block" id="register1">注册</button>**

**</div>**

**<div class="form-group">**

**<button type="button" class="btn btn-info col-sm-2 col-sm-offset-10" id="login1">登录</button>**

**</div>**

**</div>**

**</div>**

**</fieldset>**

**</form>**

**</div>**

**<script type="text/javascript" src="javascripts/jquery.min.js"></script>**

**<script type="text/javascript" src="javascripts/bootstrap.min.js"></script>**

**<script type="text/javascript">**

**$(function(){**

**$("#login1").click(function(){**

**location.href = 'login';**

**});**

**$("#register1").click(function(){**

**var username = $("#username").val();**

**var password = $("#password").val();**

**var password1 = $("#password1").val();**

**if(password !== password1){**

**$("#password").css("border","1px solid red");**

**$("#password1").css("border","1px solid red");**

**}else if(password === password1){**

**var data = {"uname":username,"upwd":password};**

**$.ajax({**

**url: '/register',**

**type: 'post',**

**data: data,**

**success: function(data,status){**

**if(status == 'success'){**

**location.href = 'login';**

**}**

**},**

**error: function(data,err){**

**location.href = 'register';**

**}**

**});**

**}**

**});**

**});**

**</script>**

**</body>**

**</head>**

**</html>**

**复制代码**

**[复制代码](javascript:void(0);)**

**login.html   跟上面register.html原理差不多**

**[复制代码](javascript:void(0);)**

**复制代码**

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<meta charset="utf-8">**

**<meta name="viewport" content="width=device-width,initial-scale=1.0">**

**<title><%= title %></title>**

**<link rel="stylesheet" href="stylesheets/bootstrap.min.css" media="screen">**

**<style type="text/css">**

**.m15{ margin: 15px;}**

**.tc{ text-align: center;font-size: 18px;font-weight: 600;}**

**</style>**

**</head>**

**<body screen\_capture\_injected="true">**

**<div class="container">**

**<%- message %>**

**<form class="col-sm-offset-4 col-sm-4 form-horizontal" role="form" method="post" onsubmit="return false">**

**<fieldset>**

**<legend></legend>**

**<div class="panel panel-default">**

**<div class="panel-heading">**

**<p class="tc">请先登录</p>**

**</div>**

**<div class="panel-body m15">**

**<div class="form-group">**

**<div class="input-group">**

**<span class="input-group-addon">**

**<span class="glyphicon glyphicon-user"></span>**

**</span>**

**<input type="text" class="form-control" id="username" name="username" placeholder="请输入用户名" required>**

**</div>**

**</div>**

**<div class="form-group">**

**<div class="input-group">**

**<span class="input-group-addon">**

**<span class="glyphicon glyphicon-lock"></span>**

**</span>**

**<input type="text" class="form-control" id="password" name="password" placeholder="请输入密码" required>**

**</div>**

**</div>**

**<div class="form-group">**

**<button type="submit" class="btn btn-primary btn-block" id="login0">登录</button>**

**</div>**

**<div class="form-group">**

**<button type="button" class="btn btn-info col-sm-2 col-sm-offset-10" id="register0">注册</button>**

**</div>**

**</div>**

**</div>**

**</fieldset>**

**</form>**

**</div>**

**<script type="text/javascript" src="javascripts/jquery.min.js"></script>**

**<script type="text/javascript" src="javascripts/bootstrap.min.js"></script>**

**<script type="text/javascript">**

**$(function(){**

**$("#register0").click(function(){**

**location.href = 'register';**

**});**

**$("#login0").click(function(){**

**var username = $("#username").val();**

**var password = $("#password").val();**

**var data = {"uname":username,"upwd":password};**

**$.ajax({**

**url:'/login',**

**type:'post',**

**data: data,**

**success: function(data,status){**

**if(status == 'success'){**

**location.href = 'home';**

**}**

**},**

**error: function(data,status){**

**if(status == 'error'){**

**location.href = 'login';**

**}**

**}**

**});**

**});**

**});**

**</script>**

**</body>**

**</head>**

**</html>**

**复制代码**

**[复制代码](javascript:void(0);)**

**最后是 home.html里头的 user.name 就是使用ejs模板通过session.user来获取user对象，这里user有name和password的属性**

**[复制代码](javascript:void(0);)**

**复制代码**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<title><%= title %></title>**

**<link rel='stylesheet' href='/stylesheets/style.css' />**

**<style type="text/css">**

**a{margin-left: 20px; text-decoration: none;}**

**a:hover{text-decoration: underline;}**

**</style>**

**</head>**

**<body>**

**<h1>Your name: <%- user.name %></h1>**

**<p>Welcome to your home ~</p>**

**<p><a href="/logout">我要注销 </a>**

**</p>**

**</body>**

**</html>**

**复制代码**

**[复制代码](javascript:void(0);)**

模板文件就是这些，接下来给主文件 app.js增加路由配置，让浏览器访问到路径后得以被解析

**[复制代码](javascript:void(0);)**

**app.use('/', routes); // 即为为路径 / 设置路由**

**app.use('/users', users); // 即为为路径 /users 设置路由**

**app.use('/login',routes); // 即为为路径 /login 设置路由**

**app.use('/register',routes); // 即为为路径 /register 设置路由**

**app.use('/home',routes); // 即为为路径 /home 设置路由**

**app.use("/logout",routes); // 即为为路径 /logout 设置路由**

**[复制代码](javascript:void(0);)**

app.use是一个中间件的用法，这里的routes看初始项目的那句代码，就是引用了routes文件夹下的index.js模块

**var routes = require('./routes/index');**

**var users = require('./routes/users');**

**所以待会我们还得继续修改完善index.js（我这里是直接把所有路径的处理方法全部放到index.js中，实际做的时候可以考虑细分出模块）**

**这里先不说index.js，因为还有很多更宽泛的工作没弄**

**1.注册登录，所以我们得需要数据库**

这里使用到了mongodb . 据我所知mongodb主要有两种使用方法，**这里使用了其中的一种：使用 mongoose**

　　Mongoose是MongoDB的一个对象模型工具，是基于node-mongodb-native开发的MongoDB nodejs驱动，可以在异步的环境下执行。

同时它也是针对MongoDB操作的一个对象模型库，封装了MongoDB对文档的的一些增删改查等常用方法，让NodeJS操作Mongodb数据库变得更加灵活简单。

我们通过Mongoose去创建一个“集合”并对其进行增删改查，就要用到它的三个属性：Schema(数据属性模型)、Model、Entity

**这里简单介绍一下，更详细的用法可以自行查阅~**

**Schema** —— 一种以文件形式存储的数据库模型骨架，无法直接通往数据库端，也就是说它不具备对数据库的操作能力，仅仅只是数据库模型在程序片段中的一种表现，可以说是数据属性模型(传统意义的表结构)，又或着是“集合”的模型骨架。

**比如定义一个Schema：**

**[复制代码](javascript:void(0);)**

**复制代码**

**var mongoose = require("mongoose");**

**var TestSchema = new mongoose.Schema({**

**name : { type:String },//属性name,类型为String**

**age : { type:Number, default:0 },//属性age,类型为Number,默认为0**

**time : { type:Date, default:Date.now },**

**email: { type:String,default:''}**

**});**

**复制代码**

**[复制代码](javascript:void(0);)**

**Model** —— 由Schema构造生成的模型，除了Schema定义的数据库骨架以外，还具有数据库操作的行为，类似于管理数据库属性、行为的类。

**比如定义一个Model：**

**var db = mongoose.connect("mongodb://127.0.0.1:27017/test");**

**// 创建Model**

**var TestModel = db.model("test1", TestSchema);**

**Entity** —— 由Model创建的实体，使用save方法保存数据，Model和Entity都有能影响数据库的操作，但Model比Entity更具操作性。

**比如定义一个Entity**：

**[复制代码](javascript:void(0);)**

**复制代码**

**var TestEntity = new TestModel({**

**name : "Lenka",**

**age : 36,**

**email: "lenka@qq.com"**

**});**

**console.log(TestEntity.name); // Lenka**

**console.log(TestEntity.age); // 36**

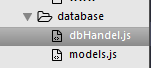
**复制代码**

**[复制代码](javascript:void(0);)**

基本就介绍到这里

因为我们要使用数据库，那就来创建它。使用的就是上述的方法

**首先，在项目根目录下建立一个database文件夹，建立文件 models.js  然后建立model处理文件 dbHandel.js**



写入文件 models.js  一个user集合，里面有name和password属性

**[复制代码](javascript:void(0);)**

**module.exports = {**

**user:{**

**name:{type:String,required:true},**

**password:{type:String,required:true}**

**}**

**};**

**[复制代码](javascript:void(0);)**

写入文件 dbHandel.js  里边主要是获取 Schema 然后处理获取 model ，最后就是返回一个model了（提供其他文件对model的操作 -- Entity是使用）

**[复制代码](javascript:void(0);)**

**复制代码**

**var mongoose = require('mongoose');**

**var Schema = mongoose.Schema;**

**var models = require("./models");**

**for(var m in models){**

**mongoose.model(m,new Schema(models[m]));**

**}**

**module.exports = {**

**getModel: function(type){**

**return \_getModel(type);**

**}**

**};**

**var \_getModel = function(type){**

**return mongoose.model(type);**

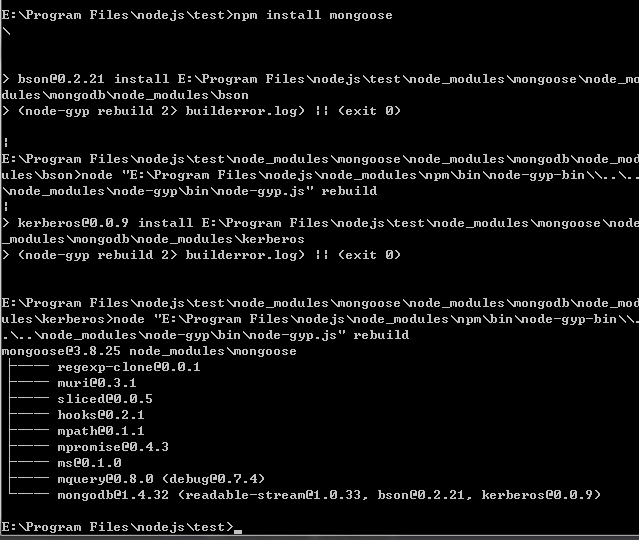
**};**

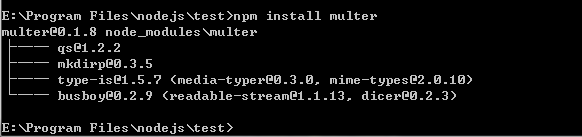
**复制代码**

**[复制代码](javascript:void(0);)**

建立好基本文件后我们就在app.js中调用使用它：要使用multer和mongoose模块

**项目没有，所以我们要安装**





**app.js中加上**

**[复制代码](javascript:void(0);)**

**复制代码**

**var multer = require('multer');**

**var mongoose = require('mongoose');**

**global.dbHandel = require('./database/dbHandel');**

**global.db = mongoose.connect("mongodb://localhost:27017/nodedb");**

**// 下边这里也加上 use(multer())**

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

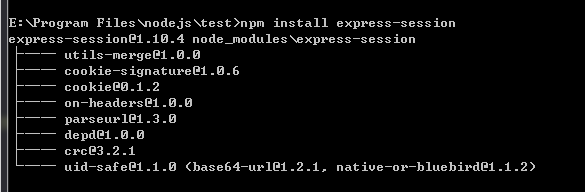
**app.use(multer());**

**app.use(cookieParser());**

**复制代码**

**[复制代码](javascript:void(0);)**

**2.因为我们使用到了session（比如进入home的时候判断session值是否为空），所以需要express-session 模块**



**然后在app.js中引用它并作初始设置：**

**[复制代码](javascript:void(0);)**

**复制代码**

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

**var app = express();**

**app.use(session({**

**secret: 'secret',**

**cookie:{**

**maxAge: 1000\*60\*30**

**}**

**}));**

**app.use(function(req,res,next){**

**res.locals.user = req.session.user; // 从session 获取 user对象**

**var err = req.session.error; //获取错误信息**

**delete req.session.error;**

**res.locals.message = ""; // 展示的信息 message**

**if(err){**

**res.locals.message = '<div class="alert alert-danger" style="margin-bottom:20px;color:red;">'+err+'</div>';**

**}**

**next(); //中间件传递**

**});**

**复制代码**

**[复制代码](javascript:void(0);)**

好现在想想我们还剩下什么：

数据库已经提供出model接口给我们使用（给它填数据）

已经初始化了路径处理

初始化了session信息 数据库配置等

页面模板也已经做完

**所以剩下的就是路径处理的部分：去routes目录下 修改index.js吧**

**/  路径**

**/\* GET index page. \*/**

**router.get('/', function(req, res,next) {**

**res.render('index', { title: 'Express' }); // 到达此路径则渲染index文件，并传出title值供 index.html使用**

**});**

**/login 路径**

**[复制代码](javascript:void(0);)**

**复制代码**

**/\* GET login page. \*/**

**router.route("/login").get(function(req,res){ // 到达此路径则渲染login文件，并传出title值供 login.html使用**

**res.render("login",{title:'User Login'});**

**}).post(function(req,res){ // 从此路径检测到post方式则进行post数据的处理操作**

**//get User info**

**//这里的User就是从model中获取user对象，通过global.dbHandel全局方法（这个方法在app.js中已经实现)**

**var User = global.dbHandel.getModel('user');**

**var uname = req.body.uname; //获取post上来的 data数据中 uname的值**

**User.findOne({name:uname},function(err,doc){ //通过此model以用户名的条件 查询数据库中的匹配信息**

**if(err){ //错误就返回给原post处（login.html) 状态码为500的错误**

**res.send(500);**

**console.log(err);**

**}else if(!doc){ //查询不到用户名匹配信息，则用户名不存在**

**req.session.error = '用户名不存在';**

**res.send(404); // 状态码返回404**

**// res.redirect("/login");**

**}else{**

**if(req.body.upwd != doc.password){ //查询到匹配用户名的信息，但相应的password属性不匹配**

**req.session.error = "密码错误";**

**res.send(404);**

**// res.redirect("/login");**

**}else{ //信息匹配成功，则将此对象（匹配到的user) 赋给session.user 并返回成功**

**req.session.user = doc;**

**res.send(200);**

**// res.redirect("/home");**

**}**

**}**

**});**

**});**

**复制代码**

**[复制代码](javascript:void(0);)**

**/register 路径**

**[复制代码](javascript:void(0);)**

**复制代码**

**/\* GET register page. \*/**

**router.route("/register").get(function(req,res){ // 到达此路径则渲染register文件，并传出title值供 register.html使用**

**res.render("register",{title:'User register'});**

**}).post(function(req,res){**

**//这里的User就是从model中获取user对象，通过global.dbHandel全局方法（这个方法在app.js中已经实现)**

**var User = global.dbHandel.getModel('user');**

**var uname = req.body.uname;**

**var upwd = req.body.upwd;**

**User.findOne({name: uname},function(err,doc){ // 同理 /login 路径的处理方式**

**if(err){**

**res.send(500);**

**req.session.error = '网络异常错误！';**

**console.log(err);**

**}else if(doc){**

**req.session.error = '用户名已存在！';**

**res.send(500);**

**}else{**

**User.create({ // 创建一组user对象置入model**

**name: uname,**

**password: upwd**

**},function(err,doc){**

**if (err) {**

**res.send(500);**

**console.log(err);**

**} else {**

**req.session.error = '用户名创建成功！';**

**res.send(200);**

**}**

**});**

**}**

**});**

**});**

**复制代码**

**[复制代码](javascript:void(0);)**

**/home  路径**

**[复制代码](javascript:void(0);)**

**复制代码**

**/\* GET home page. \*/**

**router.get("/home",function(req,res){**

**if(!req.session.user){ //到达/home路径首先判断是否已经登录**

**req.session.error = "请先登录"**

**res.redirect("/login"); //未登录则重定向到 /login 路径**

**}**

**res.render("home",{title:'Home'}); //已登录则渲染home页面**

**});**

**复制代码**

**[复制代码](javascript:void(0);)**

**/logout  路径**

**[复制代码](javascript:void(0);)**

**/\* GET logout page. \*/**

**router.get("/logout",function(req,res){ // 到达 /logout 路径则登出， session中user,error对象置空，并重定向到根路径**

**req.session.user = null;**

**req.session.error = null;**

**res.redirect("/");**

**});**

**[复制代码](javascript:void(0);)**

**当然了，把所有路径的处理放在同一个index.js事实上有点糟糕，可以考虑分着写：（这里提供一种思路分出模块）**

比如一个home.js模块里边：

**[复制代码](javascript:void(0);)**

**复制代码**

**module.exports = function ( app ) {**

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

**req.session.user = null;**

**req.session.error = null;**

**res.redirect('/');**

**});**

**}**

**复制代码**

**[复制代码](javascript:void(0);)**

从而只需要在index.js模块里边引用即可

**module.exports = function ( app ) {**

**require('./logout')(app);**

**};**

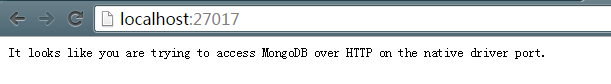
在app.js模块中再引用一下就可以（routes目录下index.js是默认文件，所以可以省略index)

**require('./routes')(app);**

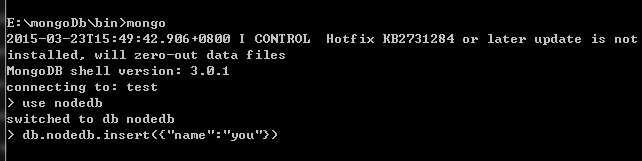
**3.好了，一个简单的注册登录功能已经完成了，启动项目吧**

（注意：因为要使用到mongodb数据库，所以要先开启数据库服务，不然无法访问，因为我们使用了nodedb 这个数据库，所以最后也要先在mongodb中创建它，不然也有可能出错 未安装数据库的可以看看  [**这篇**](http://www.cnblogs.com/imwtr/p/4351311.html)  ，检测数据库服务是否开启：浏览器打开localhost:27017 就能访问 ，然后给数据库添加nodedb吧）

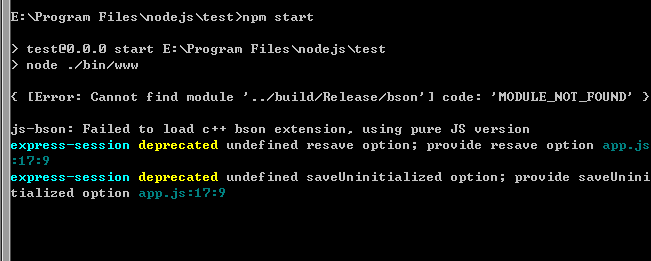
服务开启



初始化nonedb可以类似这样

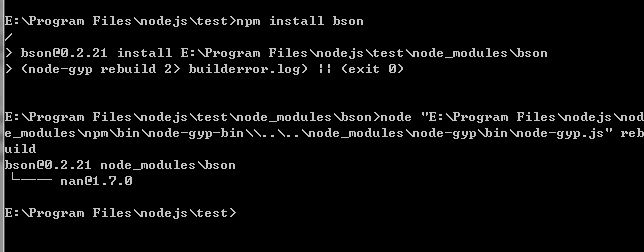


**启动项目，npm start**



上面那个bson错误的不用管它..我也不知咋处理，听说可以直接 **npm install bson 或者 npm update** 就行

但我试了貌似没什么效果



好了，项目已经打开，浏览器输入 localhost:3000 访问吧 （期间可以自己查看mongodb数据库里边nodedb --> user 数据的改动，使用mongoVUE或者命令查看）

需要代码的可移步至Github：  https://github.com/imwtr/nodejs\_express\_login\_register

------------------------------  原创作品，转载还请注明  -------------------------------