## Node.js微博项目

### 技术框架介绍

#### nodejs

基础核心开发语言。

#### Express

一个简洁而灵活的node.js Web应用框架，提供一系列强大的特性帮助我们创建各种Web应用。

#### Mongodb

用于存储在Web应用当中产生的一些数据，包括注册的用户、添加的分类、发表的文章以及对文章的评论

#### 第三方模块&中间件

bodyParser：解析post请求数据

cookies：读写cookie

swig： 模块解析引擎

mongoose：操作mongodb数据

markdown：markdown语法解析生成模块

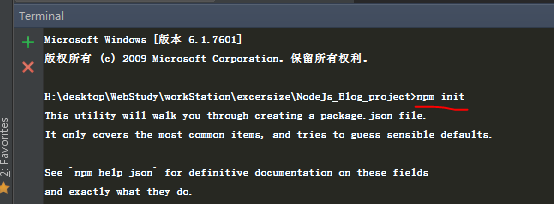
......

### 项目初始化

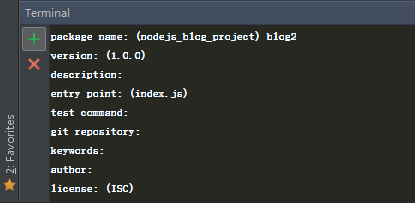
#### 在webstorm指定目录中新建一个项目

#### 在webstorm自带的Terminal中安装第三方模块

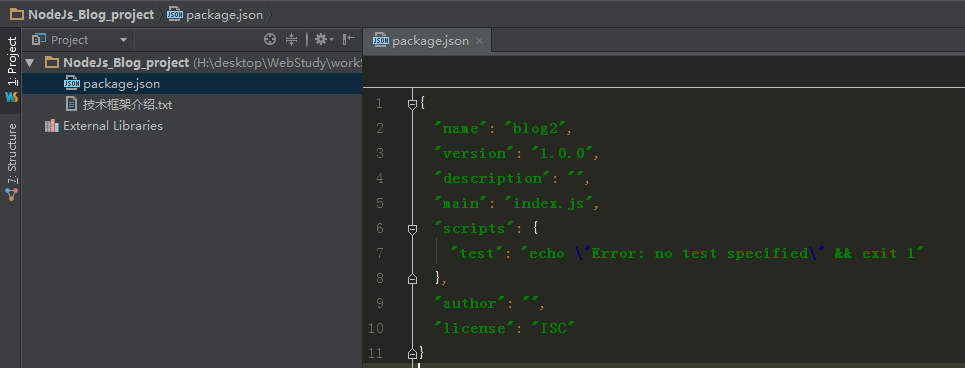
##### 使用npm命令：npm init对项目进行初始化



输入之后会它会让我们填写一些相关的信息，生成一个package.json的文件，接下来的交互内容可以暂时不填。最后输入”yes”



回车后，就会在项目目录创建一个package.json的文件：

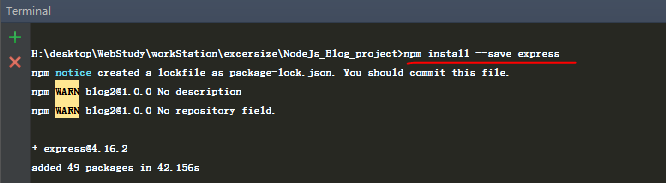


后期需要更改json文件配置的，直接打开它在上面改就行

##### 安装项目所需第三方框架和模块包

* 安装express核心模块

npm install –save express



* 安装上文所述第三方模块&中间件

npm install –save body-parser

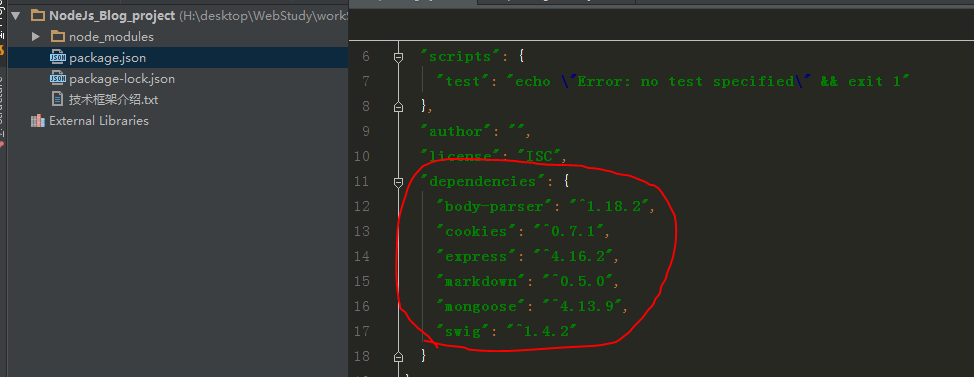
npm install –save cookies

npm install –save swig

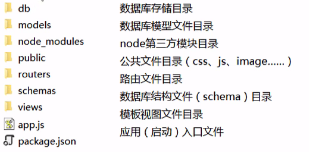
npm install –save mongoose

npm install –save markdown

安装完成后，package的json文件会新添加安装好的第三方模块以及它的需求版本



#### 项目结构分析



#### 在app.js中初始化项目

##### 创建应用、监听端口

var express = require(‘express’);

var app = express();

app.listen(8081);

通过express创建app应用=>等同于Node.js中的http.createServer()

应用创建完后，运行app.js文件，就可以在页面输入localhost:8081访问了

##### 分析用户请求网页的过程

用户通过url访问web应用，如http://localhost:8081

Web后端根据用户访问的URL处理不同的业务逻辑

##### 处理请求输出

* **路由绑定**

通过app.get()或app.post()等方法可以把一个url路径和一个或N个函数进行绑定：

app.get(‘/’,function(req,res,next){})

* **req**: request对象-保存客户端请求相关的一些数据🡪 http.request
* **res**: response对象-服务端输出对象，提供了一些服务端输出相关的一些方法🡪http.response
* **next:**方法，用于执行下一个和路径匹配的函数

【注意】

* + 当用户访问的路径满足了get方法或者post方法时，对应的app.get()或app.post()函数将会被执行
  + 同一个路径可以绑定多个function(req,res,next) 这样的函数，可通过next方法，处理下一个行为
  + req对象，包括客户端的IP、客户端的请求方式以及客户端传送过来的数据等
  + res对象，包括服务端输出的send(string)方法、setHeaders方法等方法
* **内容回应**

通过res.send(string)发送内容至客户端

#### **前后端分离—使用模板**

模板的使用：使得后端逻辑和页面表现分离—前后端分离。

##### 模板配置

var swig = require(‘swig’);

app.engine(‘html’,swig.renderFile);//定义模板引擎，使用swig.renderFile方法解析后缀为html的文件

app.set(‘view’,’/views’);//设置模板存放目录

app.set(‘view engine’,’html’);//注册模板引擎

swig.setDefault(cache:false);

#### 静态文件托管

前端的一些css、js、img、html文件有些是不需要动态改变的，只要原封不动发给客户端就好，这些统称为静态文件，针对这些静态文件，在项目中使用app.use进行统一托管。 设置静态文件托管目录---在public目录下划分并存放好相关的静态资源文件

app.use(‘/public’,express.static(\_\_dirname+’/public’));

### 项目模块划分

根据功能进行模块划分：

* + 前台模块
  + 后台管理模块
  + API模块 – 通过ajax调用的接口

使用app.use()进行模块划分

app.use(‘/admin’,require(‘./routers/admin’));//后台管理模块

app.use(‘/api’,require(‘./routers/api’));//api管理模块

app.use(‘/’,require(‘./routers/main’));//前台展示模块

#### 前台模块--路由+模板

main模块

/ 首页

/view 内容页

api模块

/ 首页

/register 用户注册

/login 用户登录

/comment 评论获取

/comment/post 评论提交

#### 后台管理模块--路由+模板



#### 功能开发顺序

##### 功能模块开发排序

* + **用户**

前台展示以及后台管理的这些数据都是由后台添加数据的时候产生的，开发过程中需要一个管理员用户，进行登录，登录后才可以进入到后台添加数据，添加完后台的功能之后，再反过来显示前台。

包括注册、登录以及管理员的认证等

* + **栏目**

分为后台管理和前台展示这两个内容。先做好栏目的后台管理（在后台可以对栏目进行添加、编辑和删除），只有这些数据产生之后，才能再返过头去添加前台展示，通过调用的方式，动态展示栏目细节。

* + **内容**
  + **评论**

##### 编码排序

* + **涉及到数据保存，所以通过Scheme定义设计数据存储结构**

以用户模块开发为例，因此首先定义用户数据的存储结构，即在数据库当中的存储结构，设计好数据库结构后，就可以写功能逻辑了。

* + **功能逻辑（后端的）**

比如用户注册、用户登录、管理员的验证如何实现？然后配合着页面展示

* + **页面展示**

#### 用户注册

##### UserSchema结构设计

在users.js中定义结构---使用mongoose模块

##### 注册界面

##### 注册逻辑

使用ajax方式实现注册

api接口编号

### 数据库的使用

#### 开启mongodb数据库

mongod –dbpath=数据文件保存路径

https://www.mongodb.com

mongodb快速入门链接：

http://mongodb.github.io/node-mongodb-native/2.2/quick-start/quick-start/

#### mongodb服务端开启

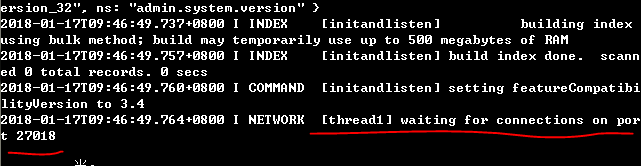
在安装的bin目录下，服务端的执行文件：mongod.exe；客户端的执行文件：mongo.exe

在Windows的cmd下，切换到mongodb安装的bin目录下，输入mongod命令开启服务端：

mongod --dbpath=H:\desktop\WebStudy\workStation\excersize\NodeJs\_Blog\_project\db --port=27018

**dbpath：**指当前数据库存储的路径，在本项目中，数据库目录文件在项目工程目录下的db目录中

**port：**指定的端口数据，默认情况下是27017，如果该端口被占用，可指定其他的；没被占用这个参数可以不设定；



键入该命令后，mongodb的服务端就开启了，接下来用mongo的客户端与它连接

#### mongodb客户端开启

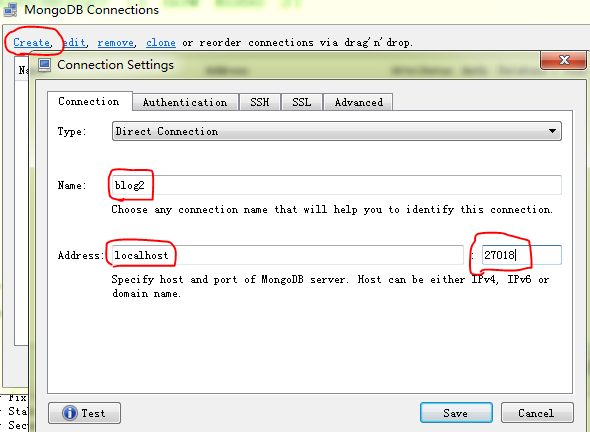
可以使用bin目录下的mongo.exe开启mongodb客户端，这里使用可视化管理工具启动。

使用可视化管理工具：Robomongo0.9.0-RC7

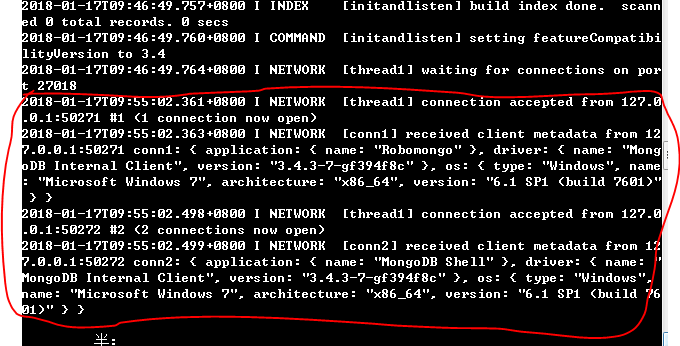
官网下载：https://robomongo.org/download

mongodb安装贴士：

https://www.cnblogs.com/xuanmanstein/p/7991589.html



点击save后点击connection，在MongoDB服务端就会有相应的连接响应，此时数据库就已经连接成功了。



#### 数据保存

使用mongoose操作数据库

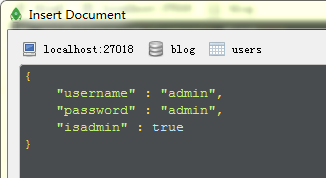
http://mongoosejs.com/

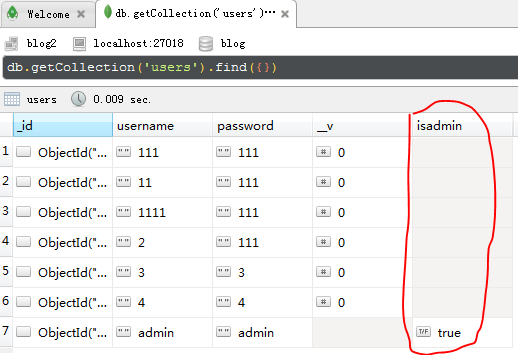
#### 创建model

通过Schema创建模型类

mongoose.model(‘模型类名称’,Schema);

添加数据库字段：





接收前端提交过来的表单：

bodyParser—用于接收post提交过来的数据，这样就可以直接通过router.post中function里req.body属性就可以得到提交过来的数据