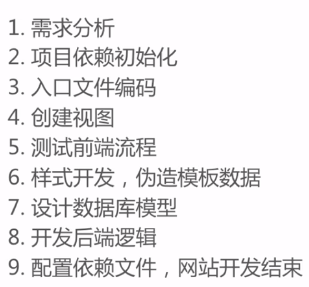
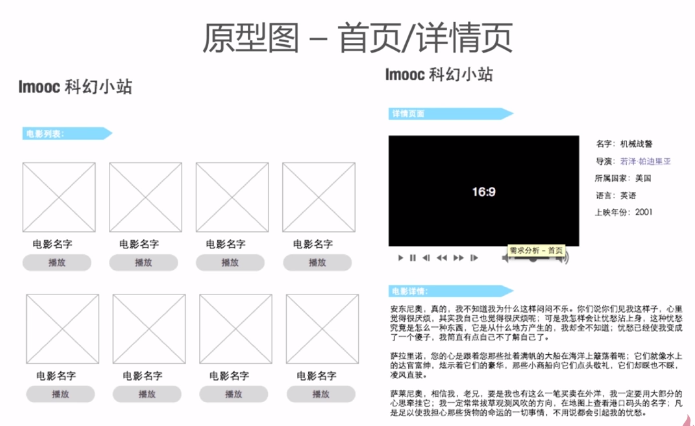
Nodsjs+mangodb建站攻略（一期）

开发步骤



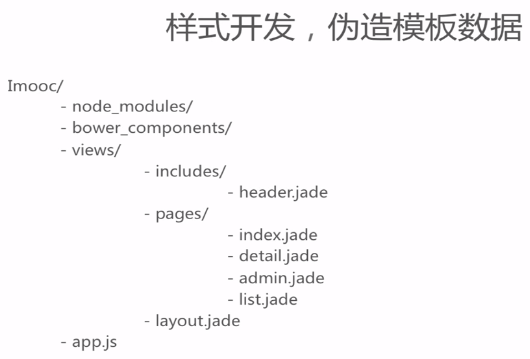




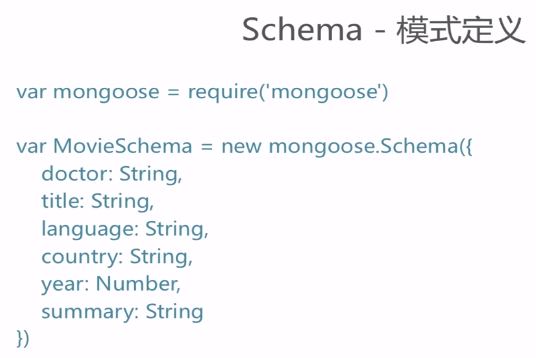


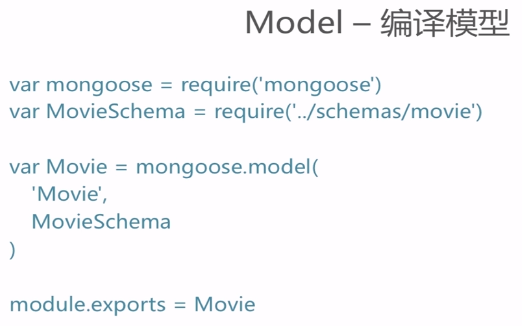


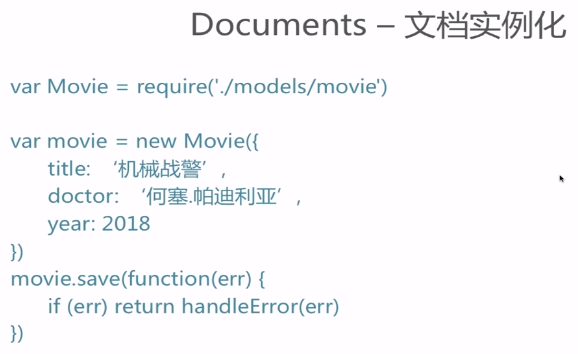


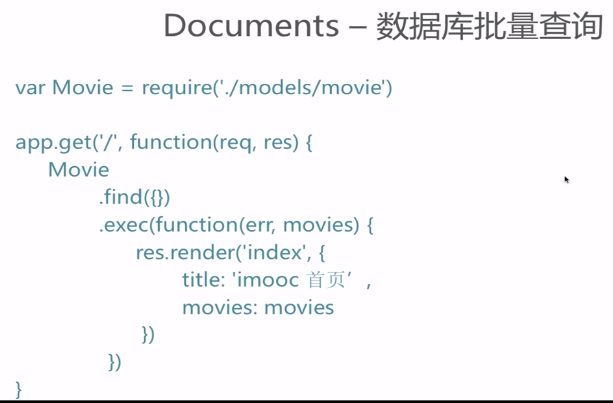




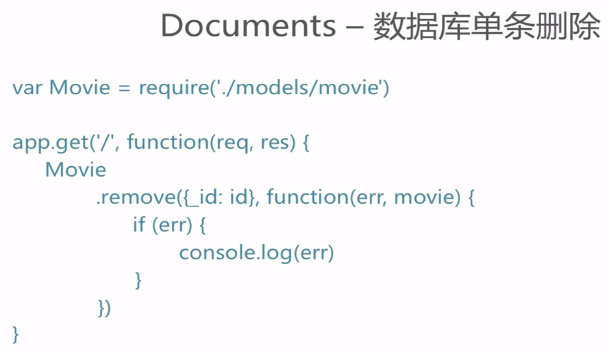


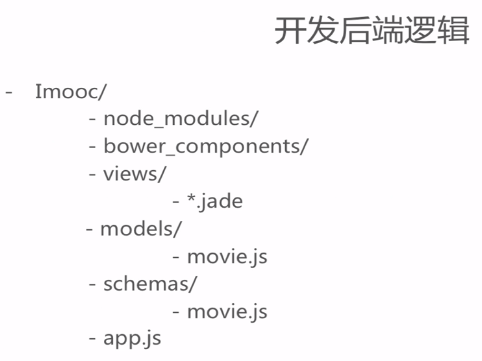












Me :

**安装**

首先假定你已经安装了 [Node.js](https://nodejs.org/)，接下来为你的应用创建一个目录，然后进入此目录并将其作为当前工作目录。

$ mkdir myapp

$ cd myapp

通过 npm init 命令为你的应用创建一个 package.json 文件。 欲了解 package.json 是如何起作用的，请参考 [Specifics of npm’s package.json handling](https://docs.npmjs.com/files/package.json)。

$ npm init

此命令将要求你输入几个参数，例如此应用的名称和版本。 你可以直接按“回车”键接受默认设置即可，下面这个除外：

entry point: (index.js)

键入 app.js 或者你所希望的名称，这是当前应用的入口文件。如果你希望采用默认的 index.js 文件名，只需按“回车”键即可。

接下来安装 Express 并将其保存到依赖列表中：

$ npm install express --save

如果只是临时安装 Express，不想将它添加到依赖列表中，只需略去 --save 参数即可：

$ npm install express

安装pug

install pug-cli –save-dev

安装moment

Moment.js是一个javascript日起处理类库，用于解析，检验，操作，以及显示日期的工具，支持多语言，网站http://momentjs.cn

**安装方法**

bower install moment --save # bower  
npm install moment --save   # npm  
Install-Package Moment.js   # NuGet  
spm install moment --save   # spm  
meteor add momentjs:moment  # meteor

**日期格式化**

moment().format('MMMM Do YYYY, h:mm:ss a'); // 二月 5日 2017, 12:09:10 中午  
moment().format('dddd');                    // 星期日  
moment().format("MMM Do YY");               // 2月 5日 17  
moment().format('YYYY [escaped] YYYY');     // 2017 escaped 2017  
moment().format();                          // 2017-02-05T12:09:10+08:00

**相对时间**

moment("20111031", "YYYYMMDD").fromNow(); // 5 年前  
moment("20120620", "YYYYMMDD").fromNow(); // 5 年前  
moment().startOf('day').fromNow();        // 12 小时前  
moment().endOf('day').fromNow();          // 12 小时内  
moment().startOf('hour').fromNow();       // 9 分钟前

**日历时间**

moment().subtract(10, 'days').calendar(); // 2017年1月26日  
moment().subtract(6, 'days').calendar();  // 本周一中午12点09  
moment().subtract(3, 'days').calendar();  // 本周四中午12点09  
moment().subtract(1, 'days').calendar();  // 昨天中午12点09分  
moment().calendar();                      // 今天中午12点09分  
moment().add(1, 'days').calendar();       // 明天中午12点09分  
moment().add(3, 'days').calendar();       // 下周三中午12点09  
moment().add(10, 'days').calendar();      // 2017年2月15日

**多语言支持**

moment().format('L');    // 2017-02-05  
moment().format('l');    // 2017-02-05  
moment().format('LL');   // 2017年2月5日  
moment().format('ll');   // 2017年2月5日  
moment().format('LLL');  // 2017年2月5日中午12点09分  
moment().format('lll');  // 2017年2月5日中午12点09分  
moment().format('LLLL'); // 2017年2月5日星期日中午12点09分  
moment().format('llll'); // 2017年2月5日星期日中午12点09分

安装body-parser

npm install body-parser –save-dev

\_id 是mongodb的默认主键。TypeError: Cannot read property '\_id' of undefined。我的出现这个错误的原因是app.use(bodyParser.urlencoded({extended:false})); 把extended:false改为extended:true。就可以了。

安装mongoose

npm install mongoose –save-dev/

**Plugging in your own Promises Library**

New in Mongoose 4.1.0

While mpromise is sufficient for basic use cases, advanced users may want to plug in their favorite [ES6-style promises library](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise) like [bluebird](https://www.npmjs.com/package/bluebird), or just use native ES6 promises. Just set mongoose.Promise to your favorite ES6-style promise constructor and mongoose will use it.

Mongoose tests with ES6 native promises, [bluebird](https://www.npmjs.com/package/bluebird), and [q](https://www.npmjs.com/package/q). Any promise library that exports an ES6-style promise constructor should work in theory, but theory often differs from practice. If you find a bug, open [an issue on GitHub](https://github.com/Automattic/mongoose/issues)

// Use native promises

mongoose.Promise = global.Promise;

assert.equal(query.exec().constructor, global.Promise);

// Use bluebird

mongoose.Promise = require('bluebird');

assert.equal(query.exec().constructor, require('bluebird'));

// Use q. Note that you \*\*must\*\* use `require('q').Promise`.

mongoose.Promise = require('q').Promise;

assert.ok(query.exec() **instanceof** require('q').makePromise);

F盘创建 data/db文件夹

命令行进入F盘 ，输入 mongod启动服务

新开一个命令行窗口 直接输入mongo

use users;

db.user.insert({})

db.user.find()

db.users.count({userid:'mldn',password:'hello'}) ;

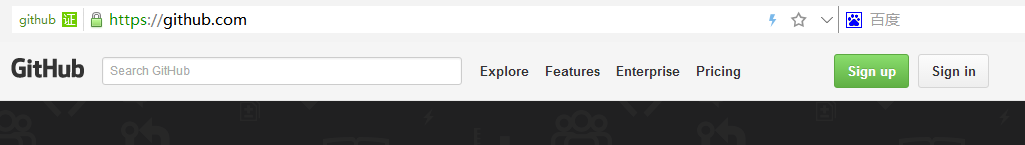
存储在app.locals中的这些键值对一般是公共模板方法或者公共模板变量，express提供了这样的机制，便于公共数据和方法在模板中的使用，而无需每次render手动传入，这种思想值得我们设计代码框架时学习

项目上传到github

**一、创建github repository(仓库)**

1-1 登录github

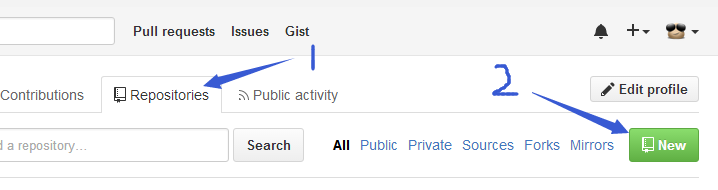
github的官方网址：https://github.com ，如果没有账号，赶紧注册一个。



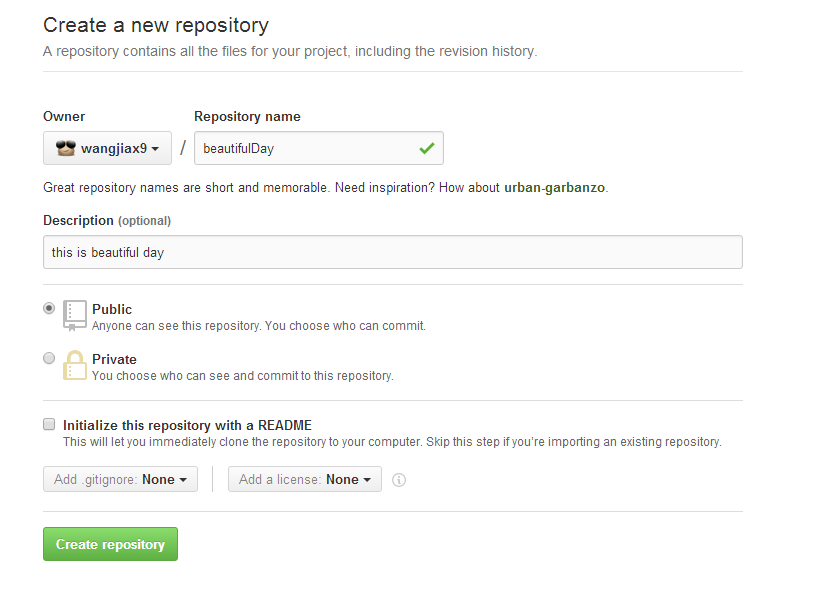
点击Sign in进入登录界面，输入账号和密码登入github。

1-2 创建repository(仓库)

登录后可以看到有repository选项卡



下面是创建仓库信息，只有名字是必填项，现在我创建了一个仓库叫：beautifulDay



创建成功后，可以看到自己的仓库地址，如此，我的远程免费的仓库就创建了。它还介绍了github仓库的常用指令。这个指令需要在本地安装git客户端。

　　git init //把这个目录变成Git可以管理的仓库

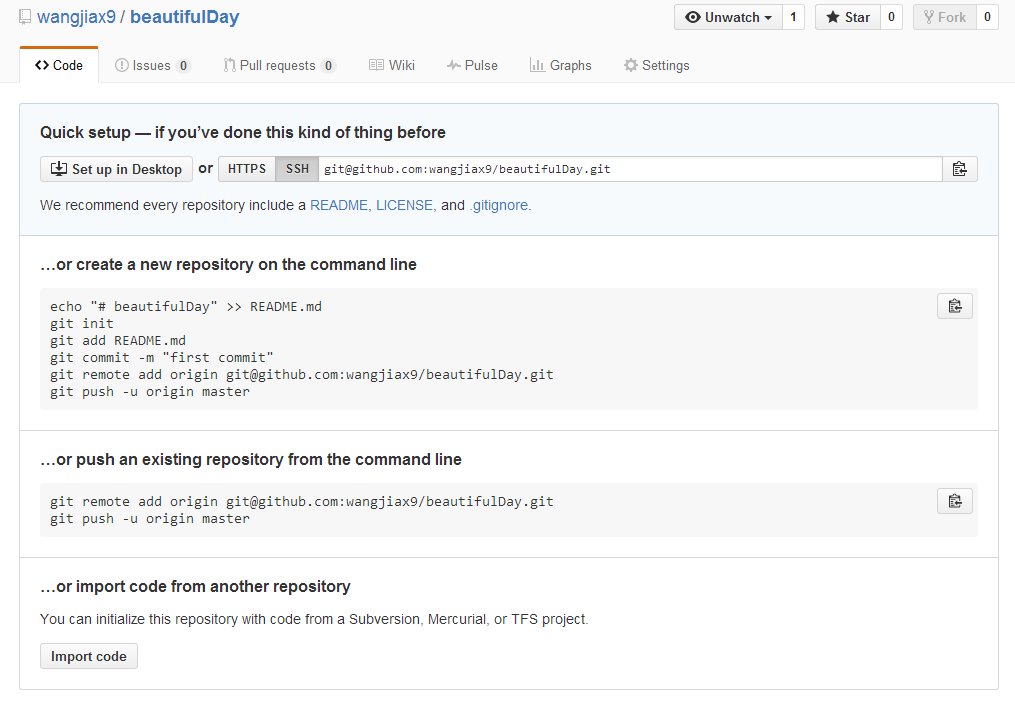
　　git add README.md //文件添加到仓库

　　git add . //不但可以跟单一文件，还可以跟通配符，更可以跟目录。一个点就把当前目录下所有未追踪的文件全部add了

　　git commit -m "first commit" //把文件提交到仓库

　　git remote add origin git@github.com:wangjiax9/practice.git //关联远程仓库

　　git push -u origin master //把本地库的所有内容推送到远程库上



**三、为Github账户设置SSH key**

众所周知ssh key是加密传输。

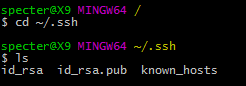
加密传输的算法有好多，git使用rsa，rsa要解决的一个核心问题是，如何使用一对特定的数字，使其中一个数字可以用来加密，而另外一个数字可以用来解密。这两个数字就是你在使用git和github的时候所遇到的public key也就是公钥以及private key私钥。

其中，公钥就是那个用来加密的数字，这也就是为什么你在本机生成了公钥之后，要上传到github的原因。从github发回来的，用那公钥加密过的数据，可以用你本地的私钥来还原。

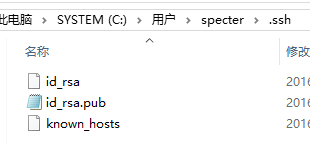
如果你的key丢失了，不管是公钥还是私钥，丢失一个都不能用了，解决方法也很简单，重新再生成一次，然后在github.com里再设置一次就行

3-1 生成ssh key

首先检查是否已生成密钥 cd ~/.ssh，ls如果有3个文件，则密钥已经生成，id\_rsa.pub就是公钥



也可以打开我的电脑C:\Users\specter\.ssh 里面找到

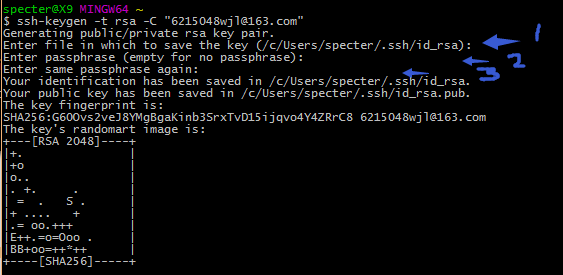


如果没有生成，那么通过$ ssh-keygen -t rsa -C “6215048wjl@163.com”来生成。

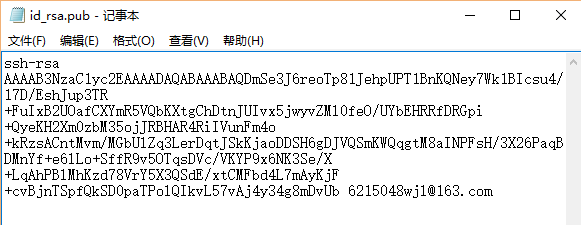
1）是路径确认，直接按回车存默认路径即可

2）直接回车键，这里我们不使用密码进行登录, 用密码太麻烦;

3）直接回车键

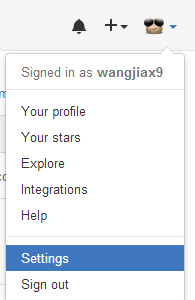


生成成功后，去对应目录C:\Users\specter\.ssh里（specter为电脑用户名，每个人不同）用记事本打开id\_rsa.pub，得到ssh key公钥



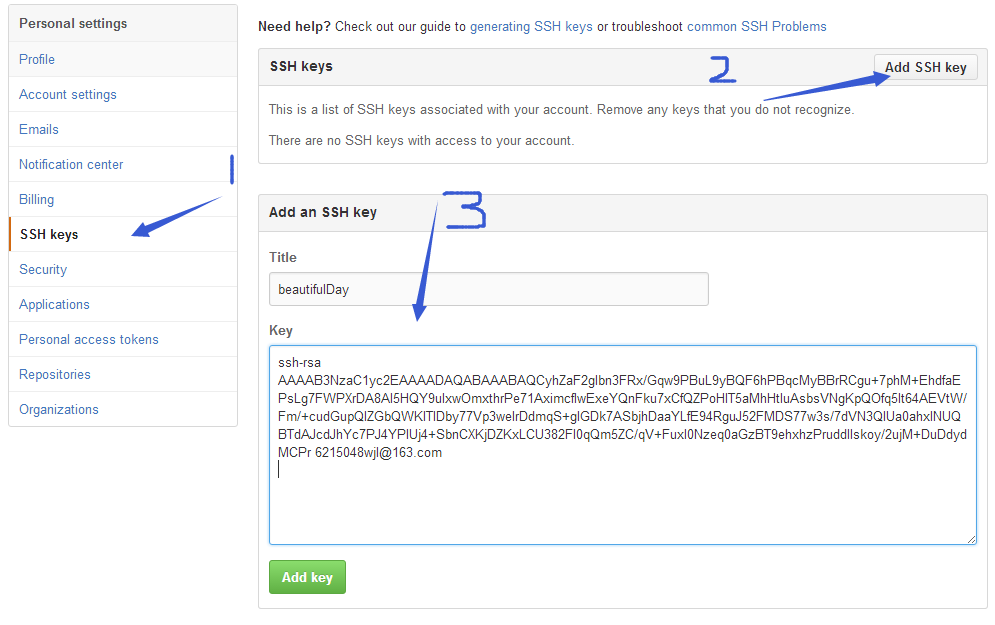
3-2 为github账号配置ssh key

切换到github，展开个人头像的小三角，点击settings

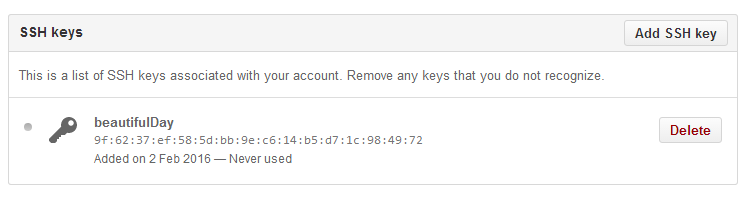


然后打开SSH keys菜单， 点击Add SSH key新增密钥，填上标题，跟仓库保持一致吧，好区分。

接着将id\_rsa.pub文件中key粘贴到此，最后Add key生成密钥吧。



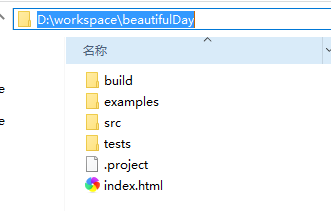
如此，github账号的SSH keys配置完成。



**四、上传本地项目到github**

4-1 创建一个本地项目

我这创建了几个空文件夹和一个文件及一个项目配置文件，好多前端项目都这样搭架构，我也追随潮流哈。



4-2 建立本地仓库

再来复习一下创建新仓库的指令：

　　git init //把这个目录变成Git可以管理的仓库

　　git add README.md //文件添加到仓库

　　git add . //不但可以跟单一文件，还可以跟通配符，更可以跟目录。一个点就把当前目录下所有未追踪的文件全部add了

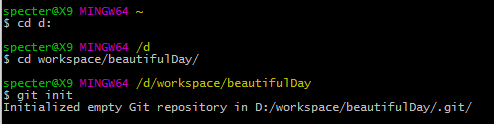
　　git commit -m "first commit" //把文件提交到仓库

　　git remote add origin git@github.com:wangjiax9/practice.git //关联远程仓库

　　git push -u origin master //把本地库的所有内容推送到远程库上

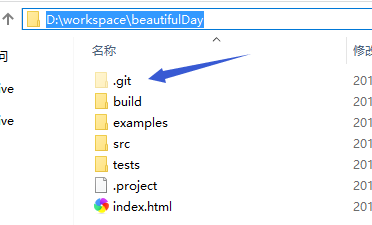
首先，进入到beautifulDay项目目录，还记得创建仓库成功后的那个页面吧，指令都在呢。

然后执行指令：git init



初始化成功后你会发现项目里多了一个隐藏文件夹.git

这个目录是Git用来跟踪管理版本库的，没事千万不要手动修改这个目录里面的文件，不然改乱了，就把Git仓库给破坏了。



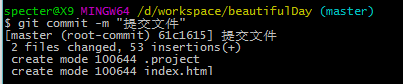
接着，将所有文件添加到仓库

执行指令：git add .

http://images2015.cnblogs.com/blog/872853/201602/872853-20160203180016429-652781698.png

然后，把文件提交到仓库，双引号内是提交注释。

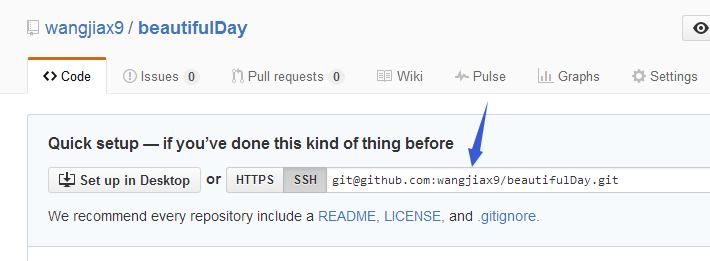
执行指令：git commit -m "提交文件"



如此本地仓库建立好了。

4-3 关联github仓库

到github beautifulDay仓库复制仓库地址



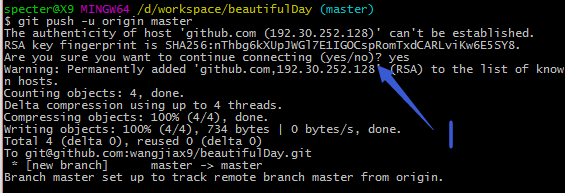
然后执行指令：git remote add origin git@github.com:wangjiax9/beautifulDay.git

http://images2015.cnblogs.com/blog/872853/201602/872853-20160203180530663-1895341072.png

4-4 上传本地代码

执行指令：git push -u origin master

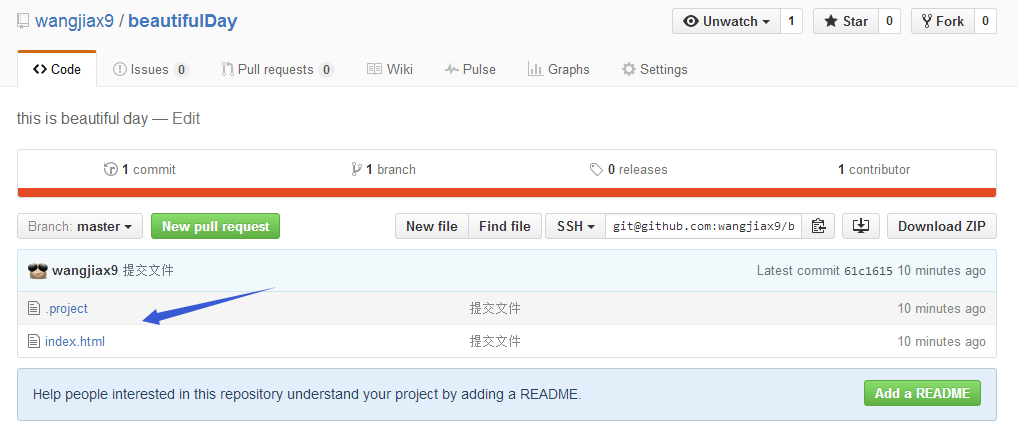
1）敲一个：yes， 然后回车



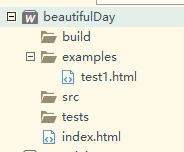
到此，本地代码已经推送到github仓库了，我们现在去githubt仓库看看。

咦！奇怪了，我的目录呢？这个坑突然冒出来是不是印象很深刻呢~

注意咯：git是不能管理空的文件夹的，文件夹里必须有文件才能add



好，我们来试一下，我在examples里新建了一个test1.html文件

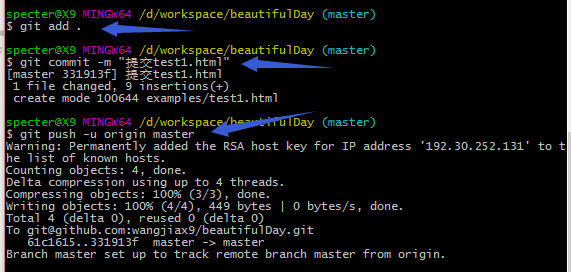


执行指令添加文件->提交文件->推送文件

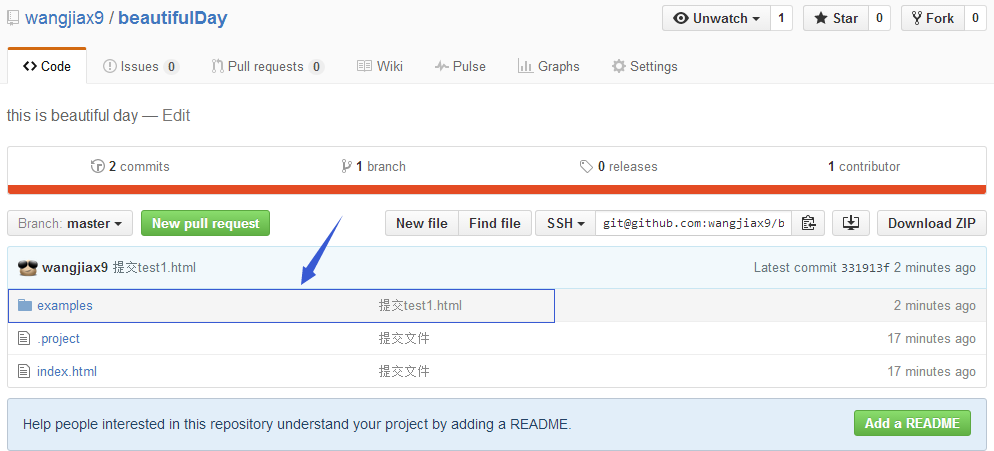
git add .

git commit -m "提交test1.html"

git push -u origin master



然后刷新一个github，你会看到，examples文件夹出来了。



打开examples文件夹，test1.html也在里面。

Nodsjs+mangodb建站攻略（二期）

日志打印

npm install morgan –save-dev

文件上传

npm install connect-multiparty –save-dev

npm install gulp –save-dev

// 引入 gulp  
var gulp = *require*('gulp');  
  
// 引入组件  
var uglify = *require*('gulp-uglify');  
  
// 合并，压缩文件  
gulp.task('scripts', function() {  
 gulp.src('app/\*/\*.js')  
 .pipe(uglify())  
 .pipe(gulp.dest('dist'));  
});  
  
gulp.task('default', function(){  
 gulp.watch('app/\*/\*.js', function(){  
 gulp.run('scripts');  
 });  
});

