1、/\*  
\* nodejs  
\* nodejs不是一种语言，也不是第三方框架，是一个平台  
\* 1、nodejs是一个让javascript运行在服务器端的开发平台  
\* 2、nodejs基于chrome v8 引擎  
\* 3、遵循commonJS规范  
\* 4、异步式i/o  
\*  
\* 一、nodejs常用命令行工具  
\* >node -e 直接解析js代码  
\* >node -v 版本  
\* >node -help 命令清单  
\* node 回车 .help 命令清单  
\* \*/  
  
/\*  
\* 二、http服务器  
\* nodejs提供了http模块  
\* \*/  
var http=require("http");//引入模块  
//创建服务  
var server=http.createServer(function(req,res){  
 //响应头  
 res.writeHead(200,{"Content-Type":"text/html"});  
 //响应内容  
 res.write("<h2>hello student!!</h2>");  
 //响应结束  
 res.end("<p>haha game</p>")  
});  
//监听端口  
server.listen(333);  
  
/\*  
\* 三、文件系统  
\* \*/  
//同步式读取文件  
var fs=require("fs");//引入文件系统模块  
var data=fs.readFileSync("testcontext.txt","utf-8");  
console.log(data);//先出  
console.log("dddddddd");//后出  
testcontext.txt内容：adsfadsfadf

//异步式读取文件  
fs.readFile("testcontext.txt","utf-8",function(err,data){  
 if(err){  
 console.log(err);  
 }else{  
 console.log(data);//后出  
 }  
});  
console.log("aaaaaa");//先出  
  
/\*  
\* 首先将异步式i/o请求发送给操作系统，然后立即返回执行后面的语句，执行完后，进入事件循环监听  
\* 当fs接收到i/o请求完毕会主动调用回调函数完成后续工作  
\*  
\* 同步则阻塞等待，直至完成，才继续后面的内容  
\* \*/  
  
/\*  
\* 四、node模块和包  
\* node模块:  
\* module.js  
\* getModule01.js  
\* getModule02.js  
\*  
\* package 包：  
\* commonJS规范的包具有如下特性：  
\* 1、package.json 在包的顶层目录下  
\* 2、二进制文件在lib目录下  
\* 3、js代码在lib目录下  
\* 4、文档在doc目录下  
\* 5、单元测试在test目录下  
\* \*/

Module01.js内容：

//1  
var name;  
exports.setName=function(myname){  
 name=myname;  
};  
exports.funsayname=function(){  
 console.log("my name is:"+name);  
};  
//2  
function funame(){  
 var name;  
 this.setName=function(myname){  
 name=myname;  
 };  
 this.funsayname=function(){  
 console.log("my name is:"+name);  
 };  
}  
module.exports=funame;

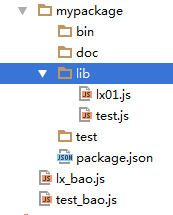
getModule01.js内容：

//1  
var mymodule=require("./module01");  
mymodule.setName("zs");  
  
var mymodule1=require("./module01");  
mymodule1.setName("ls");  
mymodule1.funsayname();//ls  
  
mymodule.funsayname();//ls

getModule02.js内容：

//2  
var module01=require("./module01");  
var m1=new module01();  
m1.setName("zs");  
  
var m2=new module01();  
m2.setName("ls");  
m2.funsayname();//ls  
  
m1.funsayname();//zs

package包



package.json内容:

{  
 "main":"./lib/test.js"  
}

test\_bao.js内容：

//调用包  
var mupag=require("./mypackage");  
mupag.say();

test.js内容：

exports.say=function(){  
 console.log("test ... package...");  
};

3、额外：

前台 接口 后台（后台传给前台接口）

接口文件：

登录模块：

文件：login.php

输入：username 用户名

password 密码

输出：字符串

成功：登录成功

失败：登录失败及错误

注册模块。。。。。

Mongodb

mongodb：

是一种非关系型数据库（No-SQL）

利用面向对象的思想，在mongdb数据库中每条记录都是一个Document对象

优势：对数据的持久操作无需开发人员再手动编写sql语句，直接调用某些方法即可实现CRUD（增删改查）操作

1. D:\mongodb-win32-i386-3.0.7\mongodb-win32-i386-3.0.7\bin>mongod.exe -dbpath

E:\mongodb43\db

启动数据库服务器

2）启动数据库操作界面

mongo.exe

3）创建数据库

>use test43

4）查看数据库

show dbs

5）给数据库添加文档（集合）并添加记录

db.persons.insert({name:'zs',age:23})

6）查询指定文档所有数据

db.persons.find()

7）删除文档中的数据

db.persons.remove({name:'zs'})

8）修改文档中的数据

db.persons.update({name:'zx'},{$set:{name:'jx'}})

mongodb的shell内置了javascript引擎，可直接执行js代码

load('test01.js')

load('E:/mongodb43/db/test01.js')

test01.js中：

1）循环插入数据

for(var i=1;i<10;i++){

db.persons.insert({name:'zs'+i,age:23+i});

}

2）全部name改成ls，age改成56

db.persons.find().forEach(function(item){

item.name='ls';

item.age=56;

db.persons.save(item);

})

test01.js

/1/print("hello");

/2/db.persons.find();

/3/

for(var i=1;i<10;i++){

db.persons.insert({name:'zs'+i,age:23+i});

}

/4/

db.persons.find().forEach(function(item){

item.name='ls';

item.age=56;

db.persons.save(item);

})

/5/

var i=0;

db.persons.find().forEach(function(item){

if(i<5){

item.name='zwww';

}

item.age=56+i;

db.persons.save(item);

i++;

})



5、

mongoDB非关系型（no-sql）与关系型数据库对比（mysql\sqlserver\oracle\db2\sysbase）

集合：是一组mongodb文档

文档：一组键值对

\_id：由12个字节的16进制数组成，保证每个文档的唯一性

关系型数据库 mongodb

表 table(二维表) 集合collection

表中一行数据 row行 文档document

字段（列） column列 key值

主外键 pk(主键)fk(外键) 主键自动生成\_id

mongoDB非关系型（no-sql）优点：

文件的动态模式--指在相同集合中的文档，不需要具有相同的字段或结构，都可以动态加入）

高可扩展性

高性能

高可用性

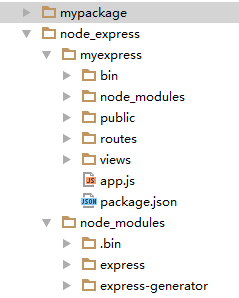
mongoDB非关系型（no-sql）缺点：

不支持“事务”

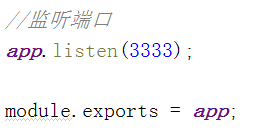
不支持join操作（复合查询）

所以，在需要用到的事务或复杂查询时，mongoDB非关系型（no-sql）不适用

6、express



app.js添加：



实现了nodejs mongoDB express 连接

7、实现node-express-mongodb连接

（node--框架express--非关系型数据库mongodb）

一共打开三个cmd.exe

第一个cmd.exe：

首先建一个文件夹，如db。当作数据库，利用D:\mongodb-win32-i386-3.0.7\mongodb-win32-i386-3.0.7\bin>mongod.exe -dbpath E:\mongodb43\db启动数据库服务器，出现27017，就表示数据库服务器开启成功。

第二个cmd.exe：

利用D:\mongodb-win32-i386-3.0.7\mongodb-win32-i386-3.0.7\bin>

mongod.exe启动数据库操作界面，use test43创建数据库，

db.persons.insert({name:'zs',age:23})给数据库添加文档（集合、表）并添加记录，db.persons.find()查询指定文档所有数据。

第三个cmd.exe：

E:\nodejsProject\_43\node\_express>express ejs zhporject建立项目（文件夹），在package.json中"dependencies":{}中添加mongodb包："mongodb":"\*"，在app.js里设定监听接口app.listen(1111)，

E:\nodejsProject\_43\node\_express\myexpress>npm install下载package.json中引用的包，（NODEjs的小基友--supervisor每次修改代码后会自动重启，引用是E:\nodejsProject\_43\node\_express>npm install -g supervisor）下载成功后用supervisor运行app.js

E:\nodejsProject\_43\node\_express\myexpress>supervisor app.js。

至此就把node-express-mongodb连接起来了。

之后在项目中：

//加载mongodb模块  
var mongodb = require('mongodb');  
//连接数据库  
var server=new mongodb.Server('127.0.0.1','27017',{auto\_reconnect:true});  
var db=new mongodb.Db("test43",server,{safe:true});

## 在view文件夹中建立ejs格式的前台页面，主要通过routes文件夹中的路由进行映射。

routes路由：index.js

var express = require('express');  
var router = express.Router();  
//加载mongodb模块  
var mongodb=require("mongodb");  
//连接数据库  
var server=new mongodb.Server('127.0.0.1','27017',{auto\_reconnect:true});  
var db=new mongodb.Db("testdb",server,{safe:true});  
  
/\* GET home page. \*/  
router.get('/', function(req, res, next) {  
 res.render('adduser', {});  
});  
  
router.get('/insertuser', function(req, res, next) {  
 db.open(function(err,db){  
 if(!err){  
 db.collection("users",function(err,coll) {  
 //插入数据  
 coll.insert({uname: req.query.username, age:req.query.userage},  
 function (err, docs) {  
 if(!err){  
 console.log(docs);  
 db.close();  
 res.send('插入成功');  
 }else{  
 res.send('插入失败');  
 }  
  
 });  
 });  
 }  
 });  
});  
module.exports = router;