Node.js入门和企业级项目开发

第4天课堂笔记

讲师：邵山欢

日期：2017年11月7日

# 复习

我们现在学习数据库的目的就是让你面试的时候，能够有一套有完整API接口的作品。前端React、Vue。

**数据库：数据的存储 + 一套数据操作的API。**

比如我们用txt文件**模拟**数据库：

[

{"id":10001,"name":"小明","age":12},

{"id":10002,"name":"小红","age":12},

{"id":10003,"name":"小刚","age":13}

]

现在的需求是：更改id为10002的age为16。需要用遍历的方法，看看哪个项的id是10002，改变这项之后重新写全部的数组。

如果是真实数据库，此时不需要遍历直接写一条语句就可以更改：

Student.update({"id":10002} , {"$set" : {"age" : 16}} , function(){

});

NoSQL和SQL的不同一定要知道：没有字段的限制，每个条目和每个条目可以有不同的字段，每个字段可以有不同的类型。

[

{"id":10001,"name":"小明","age":12},

{"id":10002,"name":"小红","age":12},

{"id":10003,"name":"小刚","age":13,"sex":"男"},

{"id":"CR10001","name":"小名","age":13,"sex":"男"}

]

MongoDB的使用，如何安装的？绿色软件，直接解压缩，设置环境变量。能够使用4个CMD命令：

|  |  |
| --- | --- |
| mongod | mongod --dbpath c:\database 开机 |
| mongo | 管理数据库的，进入REPL环境 |
| mongoimport | mongoimport -d 数据库名字 -c 集合名字 文件的名字.txt --drop |
| mongoexport | mongoexport -d 数据库名字 -c 集合名字 -o 文件的名字.txt |

数据库和NodeJS的连接，有原生的方法，不要求会了。只要会Mongoose。

**两步走：创建schema和model → CRUD操作。**

第一步，创建schema和model：

var mongoose = require("mongoose");

var schema = new mongoose.Schema({

"name" : String ,

"color" : String ,

"age" : Number ,

"pinzhong" : {

"type" : String ,

"default" : "中型犬"

}

});

module.exports = mongoose.model("Dog" , schema);

第二步，写CRUD操作：

var mongoose = require("mongoose");

mongoose.connect("mongodb://localhost/cwgl\_system");

var Dog = require("./models/Dog.js");

Dog.create({

"name" : "小白" ,

"color" : "白色" ,

"age" : 2

},(err)=>{

console.log(err);

});

Express中静态化一个资源文件夹

app.use(express.static("www"));

现在的套路是nodejs制作JSON、JSONP接口 ， 静态资源文件夹里面的文件负责页面的样式，用Ajax请求接口的数据。

formidable要熟悉：

app.post("/tijiao" , function(req,res){

//只要是post请求、delete等等，一定要用formidable来处理请求的参数

var form = new formidable.IncomingForm();

form.parse(req , function(err , fields , files){

//往数据库中存一个数据，存的数据来自POST请求的参数

Survey.create({

"timu1" : fields.timu1 ,

"timu2" : fields.timu2 ,

"timu3" : fields.timu3 ,

"date" : new Date()

},function(err){

res.json({"result" : err ? -1 : 1});

});

});

});

**编程实际上就是多个东西配合使用：**

● express的中间件

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

});

● formidable

var form = new formidable.IncomingForm();

form.parse(req , (err , fields , files)=>{

});

● 数据库

Dog.create({

},(err)=>{

});

结合起来就是这样：

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

var form = new formidable.IncomingForm();

form.parse(req , (err , fields , files)=>{

Dog.create({

name : fields.name ,

age : fields.age

},(err)=>{

})

});

});

# 模板引擎

如果要使用模板引擎，要做4个事情：

1. 安装依赖，npm install --save ejs
2. 设置默认模板引擎 app.set("view engine" , "ejs");
3. 在views文件夹中创建一个.ejs后缀的页面，就是模板
4. 在express的中间件中用res.render()来呈递视图，语法就是res.render(模板文件名字 , {字典});

我们学习ejs模板引擎，首先安装依赖

cnpm install --save express ejs

|  |
| --- |
| ┣ views  ┃ ┣ shouye.ejs  ┣ app.js |

shouye.ejs：

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

</head>

<body>

<div class="wrap">

<h1>好**<%=xinqing%>**啊！今天我买了**<%=dongxi%>**，花了**<%=qian%>**元！</h1>

</div>

</body>

</html>

app.js:

var express = require("express");

var app = express();

//设置默认模板引擎为ejs

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

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

//我们现在的res有的功能是：

//res.send()、res.json()、res.jsonp()、res.sendFile()

//现在多了一个res.render()表示使用模板页面

//不需要加上views文件夹，因为模板引擎默认就是放在views文件夹中的

//也不需要加上.ejs后缀

res.render("shouye" , {

"xinqing" : "高兴",

"dongxi" : "苹果叉",

"qian" : 8000

});

});

app.listen(3000);

一些注意事项：

① views文件夹可以改变，使用语句

app.set("views" , "templates");

这样我们所有的.ejs文件都要放到 templates文件夹中了。

② 拓展名必须是.ejs，render的时候不需要写.ejs

res.render("shouye" , {

"xinqing" : "高兴",

"dongxi" : "苹果叉",

"qian" : 8000

});

③ 可以使用一些for循环和if语句：

<ul>

<% for(var i = 0 ; i < ouxiang.length ; i++){ %>

<li><%= ouxiang[i] %></li>

<% } %>

</ul>

<% %>表示for循环、if语句；

<%= %>表示输出

对应的字典就必须是数组：

res.render("shouye" , {

"xinqing" : "高兴",

"dongxi" : "苹果叉",

"qian" : 8000 ,

"ouxiang" : **["鹿晗","王源","王俊凯","胡歌"]**

});

再比如做一个年份选择的下拉列表：

<select name="" id="">

<% for(var i = 1930 ; i <= 2017 ; i++){ %>

<option value="<%= i %>"><%= i %></option>

<% } %>

</select>

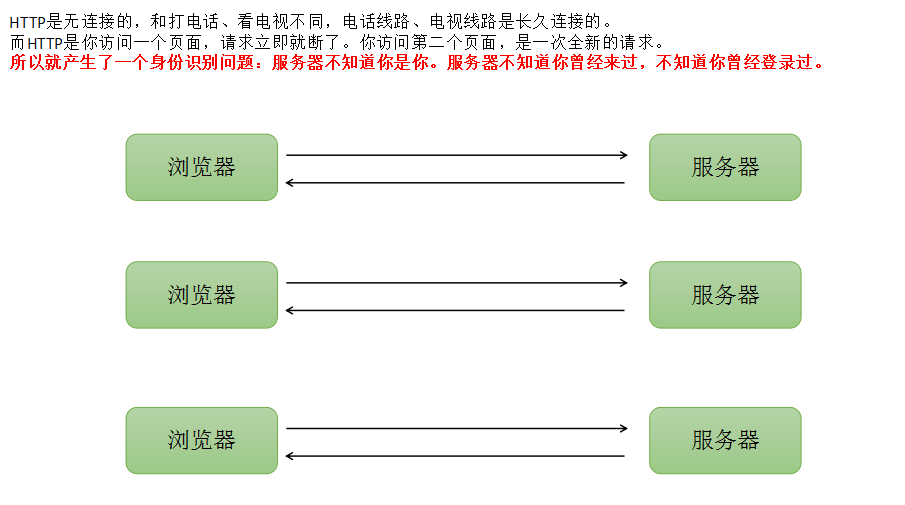
还有其他的模板引擎pug（原名叫做Jade） : https://www.npmjs.com/package/pug

有兴趣的同学自己研究。

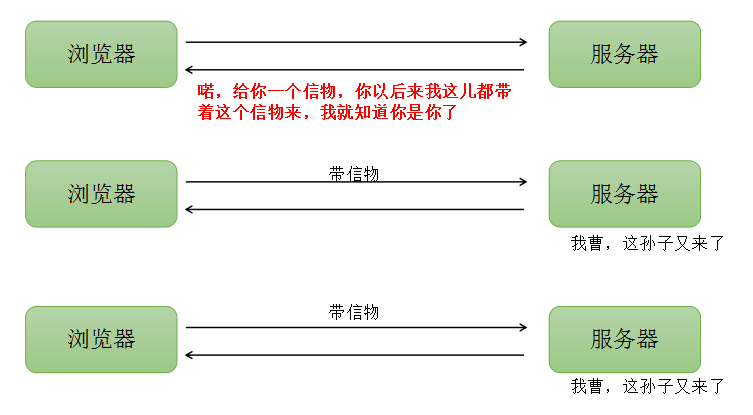
# cookie和session

## 3.1 cookie

HTTP是无连接的，所以产生了身份识别问题：



工程师是神奇的物种，解决问题的思路往往非常简单：



**老外喜欢用“曲奇饼干”（cookie）当做信物，所以cookie就是上图中的信物。**

|  |
| --- |
| 服务器会下发一个Set-Cookie的下行报文字段，今后每一次访问这个服务器的时候，浏览器都要携带Cookie上行报文上去。这样服务器就知道你是你了。 |

express中使用cookie需要安装一个依赖cookie-parser

cnpm install --save cookie-parser

设置cookie：

res.cookie('visited', visitedArr, { maxAge: 86400 });

读取cookie：

**//提前装好cookie-parser**

var cookieParser = require('cookie-parser');

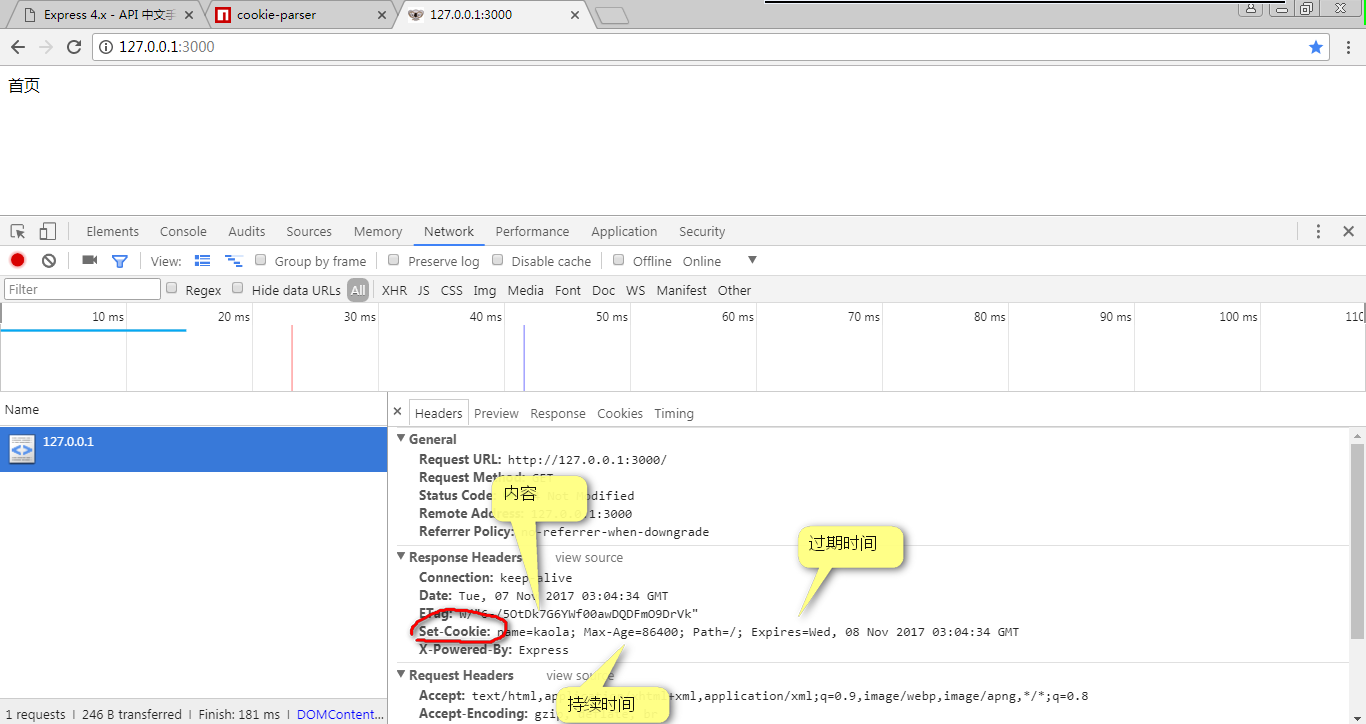
app.use(cookieParser());

**//中间件中**

app.get("/:city" , function(req,res){

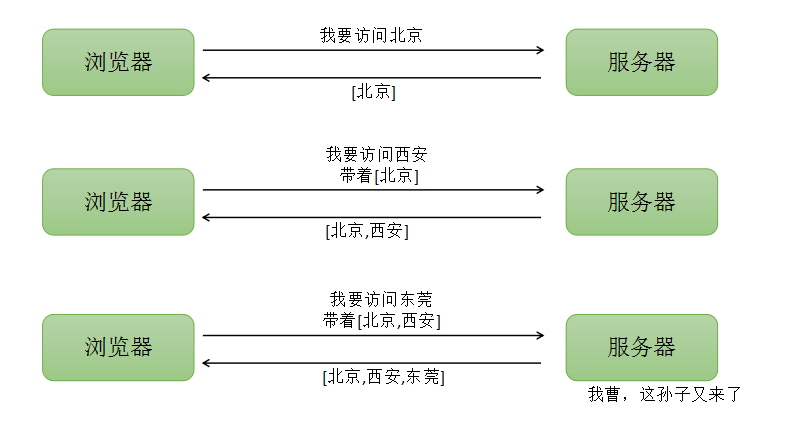
req.cookies.visited;

});



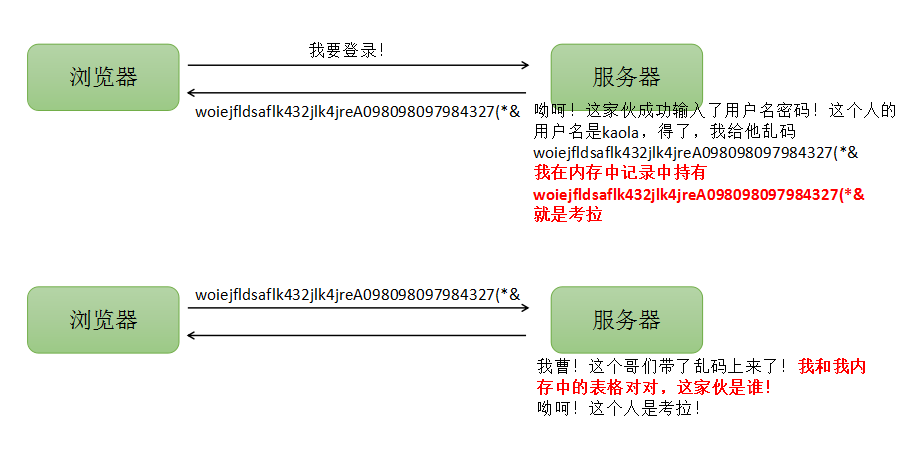
浏览器在2011年之前，如果想往硬盘中存储东西，唯一可以的办法就是cookie。

2011年本地存储诞生了，localStorage诞生了，在2011年之前，cookie扮演了很多本地存储的角色。



## 3.2 session

刚才我们制作了cookie的案例，发现服务器每次下发的cookie是有意义的文字。**session的机理是下发一个随机乱码，服务器记录下这个随机乱码的持有者的情况**。

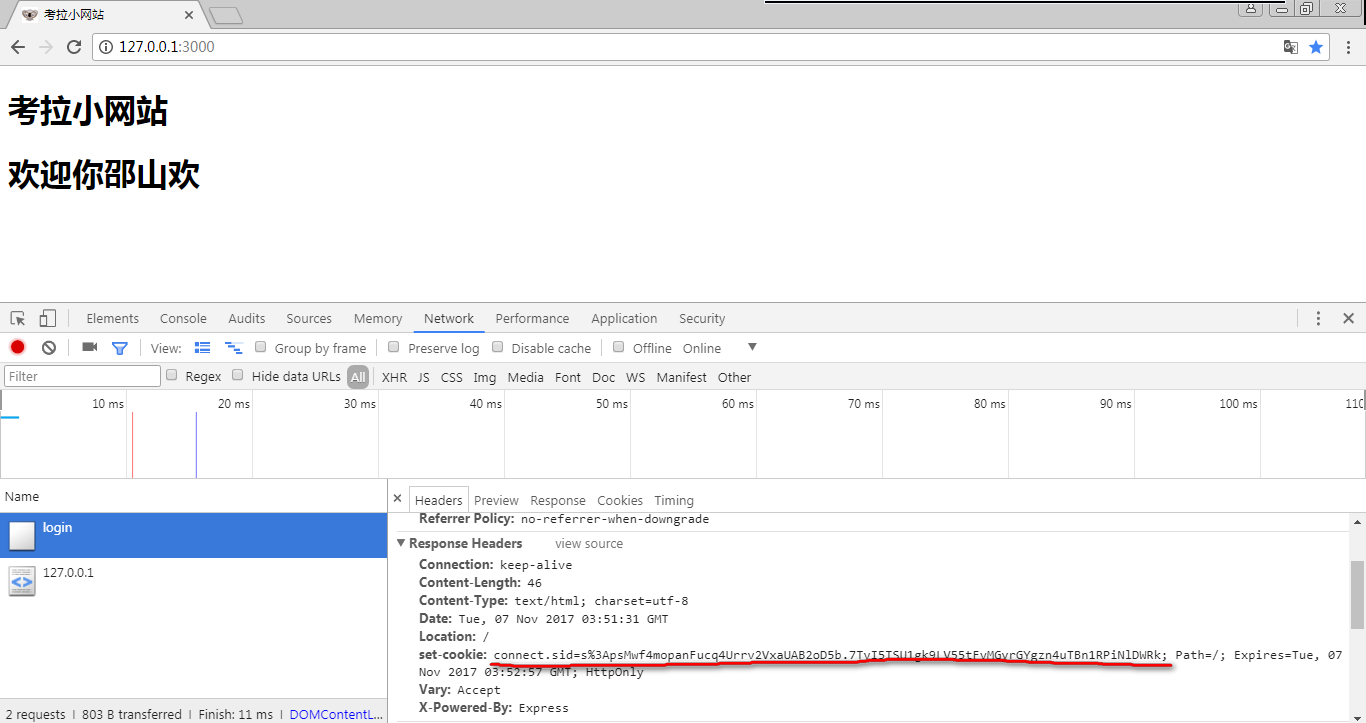


session的使用是对程序员是透明的，程序员不用刻意的设置session，Set-Cookie和Cookie就已经设置好了。

session在express中的使用，需要npm包：express-session。

cnpm install --save express express-session ejs

登录成功后，浏览器会下发随机乱码：



对于服务器来说，它只需要记住：

**携带s%3ApsMwf4mopanFucq4Urrv2VxaUAB2oD5b.7TvI5TSU1gk9LV55tFvMGyrGYgzn4uTBn1RPiNlDWRk**

**的人login是true了，yonghuming是邵山欢。**

//登录

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

var form = new formidable.IncomingForm();

form.parse(req , function(err , fields){

if(fields.mima = "123123"){

//下发session

**req.session.login = true;**

**req.session.yonghuming = fields.yonghuming;**

res.redirect("/");

}

});

});

首页：

//首页

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

//呈递模板，把登录信息也带上去：

res.render("shouye" , {

login : **req.session.login** ,

yonghuming : **req.session.yonghuming** ,

anlian : **req.session.anlian**

});

});

# 问答平台项目

## 4.1 基本文件夹的结构

创建项目文件夹，安装依赖：

cnpm intsall --save express mongoose formidable cookie-parser express-session ejs

项目的基本文件结构：

|  |
| --- |
| ┣ models  ┣ views  ┣ controllers  ┣ www  ┃ ┣ js  ┃ ┣ css  ┃ ┣ images  ┣ app.js |

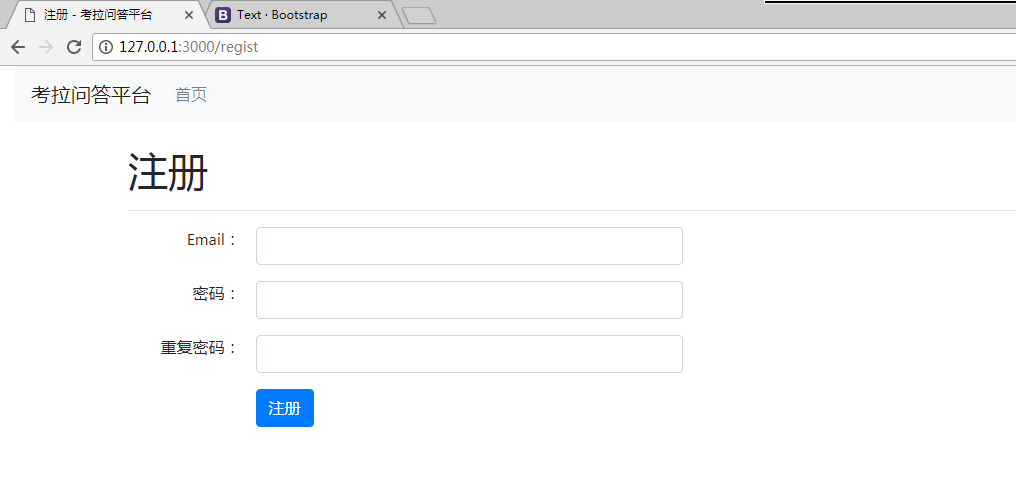
## 4.2 注册业务

开通一个业务，有三个事儿要做：

1. 开路由，罗列一个app.get("/regist" , \*\*\*\*\*);
2. 开模板引擎，在views文件夹中创建一个同名的.ejs结尾的模板文件;
3. 思考模板引擎中的字典，用res.render()呈递

路由：

|  |  |  |
| --- | --- | --- |
|  |  |  |
| /regist | GET | 显示注册页面 |
| /regist | POST | 执行注册 |
| /regist | CHECKOUT | 验证用户名是否被占用 |



**一定要注意一个行业操守：不能将用户的密码的明文直接保存在数据库中。**

CSDN有一次被黑了，结果用户的密码都泄露了，考拉老师的百度贴吧、QQ、网易邮箱都用的一个密码，全完蛋。

**我们不能防止被黑，但是我们可以不让用户的密码泄露。黑客只能得到加密之后的密码。**

这里介绍一个加密的东西MD5或者SHA256：这些加密都是不可逆的加密，不能从密码翻译为明文。常用于校验信息的正确性。

在线加密网站：http://tool.oschina.net/encrypt?type=2

我们来看SHA256：

|  |  |
| --- | --- |
| 明文 | 密码 |
| 妈妈说不管你有多长的文字要加密，一律是64位 | 57f146775795fc42689b98cb9e756f7347efd4cfe6626b0a9aeea8c6de58eec7 |
| 妈妈讲不管你有多长的文字要加密，一律是64位 | b56c5f6e489fbd69df47bf21ae9dcaa9771094a6e8224a379eeae6fa96490b5e |
| 我爱你 | c0ad5411b19cfcba9d674d21411a970159f6ae4e180831ddd6a91797be547752 |
| 你爱我 | f3a5bb9836b59f01cf001bc70c95318fea4cf43a1b148a8b5344aea07b34a279 |

注册的原理：服务器的表格中，存储用户名，和加密之后的密码。当用户登录的时候，再次将用户登录填写的密码加密，和数据库的加密的密码进行比对，如果正确了说明用户密码填写正确。

NodeJS中有一个原生模块叫做crypto，可以实现SHA256、MD5加密。