NodeJS入门和企业级项目开发

第5天课堂笔记（本课程共8天）

# 目录

[目录](#_Toc15700)

[一、数据库](#_Toc17105)

[二、Cookie和Session](#_Toc32336)

[2.1 Cookie](#_Toc29088)

[4.2 Session](#_Toc18748)

[4.3 session](#_Toc1267)

# 一、数据库

分页，想想我们的百度百家Ajax案例，当时调用了百度的JSON，有一个参数叫做page=3，生成的JSON不一样。

这个就是分页，就是我们想寻找所有的新闻，但是是位于第3页的新闻。那么有两种做法：

1） 错误的做法： 就是讲所有的result都读取到数组，然后进行数据操作，进行分页；

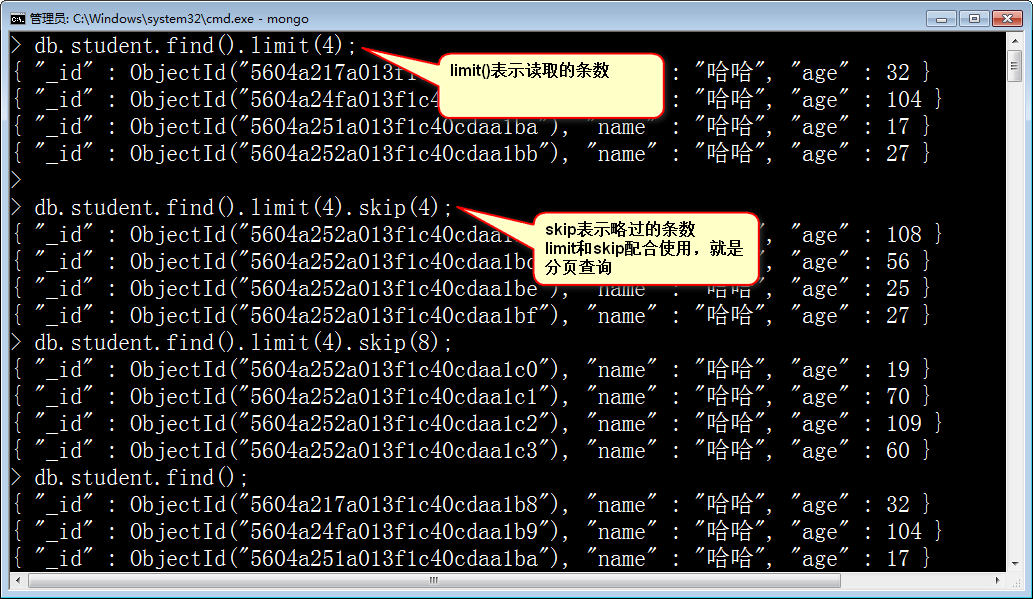
2） 正确的做法： 就是真的在数据库中，只读取这么多内容。

错误的，我们试图每次都读取全部数据，但是这样开销很大。

|  |
| --- |
| 1. var a = []; 2. db.find("student",{},function(err,result){ 3. for(var i = 10 \* page ; i < 10 \* (page + 1) ; i++){ 4. a.push(result[i]); 5. } 6. res.send(a); 7. }); |

所以，mongodb提供了傻傻的两个函数。

limit() skip()



加入，第一页是page=0。每页10条，所以当前页的查询语句

|  |
| --- |
| 1. db.student.find({}).limit(10).skip(page\*10) |

数据总数怎么得到？

shell中

|  |
| --- |
| 1. db.student.stats().count; |

ejs页面中，如果我们想使用underscore的模板，就会有模板冲突的问题。

underscore用的：

|  |
| --- |
| 1. <script type="text/template" id="moban"> 2. <div class="list-group"> 3. <a href="#" class="list-group-item active"> 4. <h4 class="list-group-item-heading">**<%= xingming %>**</h4> 5. <p class="list-group-item-text">**<%= liuyan %>**</p> 6. </a> 7. </div> 8. </script> |

ejs以为是自己的模板。所以报错，提示你没有传递xingming参数。

解决方法就是underscore源码。在源码中搜索<%即可。

# 二、Cookie和Session

## 2.1 Cookie

● HTTP是无状态协议。简单地说，当你浏览了一个页面，然后转到同一个网站的另一个页面，服务器无法认识到，这是同一个浏览器在访问同一个网站。每一次的访问，都是没有任何关系的。

那么世界就乱套了，比如我上一次访问，登陆了，下一次访问，又让我登陆，不存在登陆这事儿了。

● Cookie是一个简单到爆的想法：当访问一个页面的时候，服务器在下行HTTP报文中，命令浏览器存储一个字符串；浏览器再访问同一个域的时候，将把这个字符串携带到上行HTTP请求中。

第一次访问一个服务器，不可能携带cookie。 必须是服务器得到这次请求，在下行响应报头中，携带cookie信息，此后每一次浏览器往这个服务器发出的请求，都会携带这个cookie。

特点

● cookie是不加密的，用户可以自由看到；

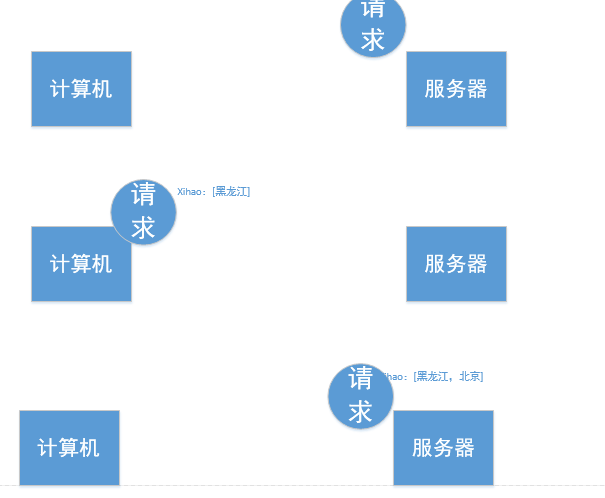
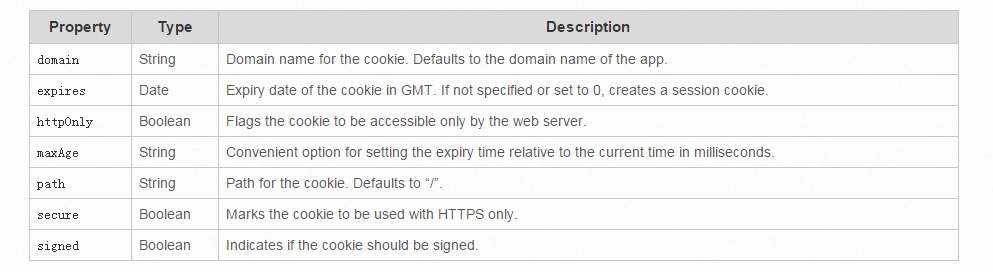
● 用户可以删除cookie，或者禁用它

● cookie可以被篡改

● cookie可以用于攻击

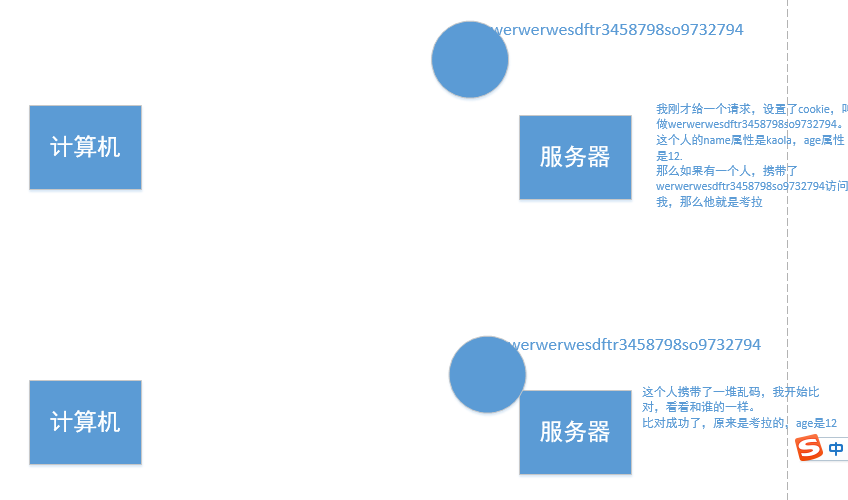
● cookie存储量很小。未来实际上要被localStorage替代，但是后者IE9兼容。

express中的cookie，你肯定能想到。 res负责设置cookie， req负责识别cookie。



## 4.2 Session

会话。 Session不是一个天生就有的技术，而是依赖cookie。



session依赖cookie，当一个浏览器禁用cookie的时候，登陆效果消失； 或者用户清除了cookie，登陆也消失。

session比cookie不一样在哪里呢？ session下发的是乱码，并且服务器自己缓存一些东西，下次浏览器的请求带着乱码上来，此时与缓存进行比较，看看是谁。

所以，一个乱码，可以对应无限大的数据。

任何语言中，session的使用，是“机理透明”的。他是帮你设置cookie的，但是足够方便，让你感觉不到这事儿和cookie有关。

## 4.3 session

|  |
| --- |
| 1. var session = require("express-session"); 2. app.use(session({ 3. secret: 'keyboard cat', 4. resave: false, 5. saveUninitialized: true 6. })) 7. app.get("/",function(req,res){ 8. if(**req.session.login == "1"**){ 9. res.send("欢迎" + **req.session.username**); 10. }else{ 11. res.send("没有成功登陆"); 12. } 13. }); 14. app.get("/login",function(req,res){ 15. **req.session.login = "1"**; //设置这个session 16. req.session.username = "考拉"; 17. res.send("你已经成功登陆"); 18. }); |

加密使用的是MD5加密。

1 C4CA4238A0B923820DCC509A6F75849B

2 C81E728D9D4C2F636F067F89CC14862C

我爱北京天安没和陈伟：

97AFBA2EE71DE6A78B61149A391A378F

我爱北京天安没与陈伟：

CCB721C1E26067BBF4BB49DFDBFA0C9E

我爱北京天安没和陈猥：

C3D110D2441EF9323D3F1E6082E708F8

不管你加密多大的东西，哪怕10M文字，都会加密为32位的字符串，就是密码。并且神奇的，数学上能够保证，哪怕你更改1个文字，都能大变。所以MD5也能用于比对版本。

MD5是数学上，不能破解的。 不能反向破解。

也就是说，C4CA4238A0B923820DCC509A6F75849B 没有一个函数，能够翻译成为1的。

但是，有的人做数据库，就是把1~999999所有数字都用MD5加密了，然后进行了列表，所以有破解的可能。