shiro.liang.yi@gmail.com

[公司名称] | [公司地址]

日志

总结

# 优旦科技

[优旦科技 1](#_Toc514918077)

[1· 2018年4月2日星期一 6](#_Toc514918078)

[入职第一天 6](#_Toc514918079)

[2· 2018年4月3日星期二 6](#_Toc514918080)

[安装windows操作系统 6](#_Toc514918081)

[3· 2018年4月4日星期三 6](#_Toc514918082)

[安装IDE 6](#_Toc514918083)

[4· 2018年4月5日星期四 6](#_Toc514918084)

[学习Git 6](#_Toc514918085)

[5· 2018年4月6日星期五 6](#_Toc514918086)

[无任务 6](#_Toc514918087)

[6· 2018年4月7日星期六 6](#_Toc514918088)

[Git切换分支 6](#_Toc514918089)

[更新修改 6](#_Toc514918090)

[小结 6](#_Toc514918091)

[分配任务 8](#_Toc514918092)

[7· 2018年4月9日星期一 9](#_Toc514918093)

[在centos（一种linux社区发行版）上安装jdk 9](#_Toc514918094)

[安装tomcat 10](#_Toc514918095)

[安装mysql（参考：https://www.jianshu.com/p/1dab9a4d0d5f） 10](#_Toc514918096)

[Linux删除操作 15](#_Toc514918097)

[查看有哪些用户 16](#_Toc514918098)

[添加用户 16](#_Toc514918099)

[设置密码 16](#_Toc514918100)

[删除用户 17](#_Toc514918101)

[修改账号 17](#_Toc514918102)

[用户口令管理 17](#_Toc514918103)

[直接将普通用户权限提升到root 18](#_Toc514918104)

[8· 2018年4月10日星期二 18](#_Toc514918105)

[部署项目 18](#_Toc514918106)

[Elasticsearch安装 19](#_Toc514918107)

***[【注意】](#_Toc514918108)*** [20](#_Toc514918108)

[9· 2018年4月11日星期三 21](#_Toc514918109)

[Mysql 安装后修改密码 21](#_Toc514918110)

[给一个用户赋予某个文件夹（及旗下的）的权限 21](#_Toc514918111)

[关闭防火墙 21](#_Toc514918112)

[安装Kinaba 21](#_Toc514918113)

[修改当前用户下的vim编辑器颜色 21](#_Toc514918114)

[10· 2018年4月12日星期四 22](#_Toc514918115)

[1.Node与cluster 22](#_Toc514918116)

[2.document 22](#_Toc514918117)

[3.type 22](#_Toc514918118)

[新建和删除index 22](#_Toc514918119)

[11· 2018年4月13日星期五 24](#_Toc514918120)

[查询索引库下的数据 24](#_Toc514918121)

[带条件的查询索引 24](#_Toc514918122)

[一个创建索引例子的完整说明 24](#_Toc514918123)

[12· 2018年4月14日星期六 27](#_Toc514918124)

[重新梳理linux各种命令 27](#_Toc514918125)

[13· 2018年4月16日星期一 29](#_Toc514918126)

[kibana学习 29](#_Toc514918127)

[精确查询 29](#_Toc514918128)

[14· 2018年4月17日星期二 31](#_Toc514918129)

[elasticsearch java api 31](#_Toc514918130)

[15· 2018年4月18日星期三 31](#_Toc514918131)

[抓包工具：wireshark 31](#_Toc514918132)

[node.js组成模块 31](#_Toc514918133)

[谈一谈path 31](#_Toc514918134)

[node.js程序的运行方式 32](#_Toc514918135)

[node.js程序编写 32](#_Toc514918136)

[回调函数 32](#_Toc514918137)

[16· 2018年4月19日星期四 33](#_Toc514918138)

[异步编程的理解 33](#_Toc514918139)

[if-while-do while的精细理解 33](#_Toc514918140)

[node.js的模块加载 33](#_Toc514918141)

[事件驱动 33](#_Toc514918142)

[setTimeout(code，millisec)用法 34](#_Toc514918143)

[17· 2018年4月20日星期五 34](#_Toc514918144)

[最简单的文件复制 34](#_Toc514918145)

[js中三种定义变量的方式 34](#_Toc514918146)

[node.js Buffer 35](#_Toc514918147)

[buffer转换json对象 37](#_Toc514918148)

[缓冲区合并 37](#_Toc514918149)

[缓冲区比较 38](#_Toc514918150)

[拷贝缓冲区 38](#_Toc514918151)

[缓冲区裁剪和缓冲区长度 38](#_Toc514918152)

[txt字符流的拷贝 38](#_Toc514918153)

[字节流的操作（以图片为例） 40](#_Toc514918154)

[node.js的并发机制 41](#_Toc514918155)

[18· 2018年4月23日星期一 41](#_Toc514918156)

[node.js管道流 41](#_Toc514918157)

[链式流 42](#_Toc514918158)

[如何保证复制文件不被覆盖呢？ 42](#_Toc514918159)

[模块系统 43](#_Toc514918160)

[路由 43](#_Toc514918161)

[全局对象 43](#_Toc514918162)

[文件系统 44](#_Toc514918163)

[19· 2018年4月24日星期二 44](#_Toc514918164)

[get/post请求 44](#_Toc514918165)

[工具模块 44](#_Toc514918166)

[restful风格 44](#_Toc514918167)

[20· 2018年4月25日星期三 45](#_Toc514918168)

[21· 2018年4月26日星期四 45](#_Toc514918169)

[22· 2018年4月27日星期五 45](#_Toc514918170)

[element框架 45](#_Toc514918171)

[eleme form校验 46](#_Toc514918172)

[固定的elem表单格式（记） 46](#_Toc514918173)

[23· 2018年4月28日星期六 50](#_Toc514918174)

[调试装包扫码桌面程序 50](#_Toc514918175)

[24· 2018年4月30日星期一 50](#_Toc514918176)

[@RequestMapping和@ResponseBody 50](#_Toc514918177)

[25· 2018年5月2日星期三 51](#_Toc514918178)

[26· 2018年5月3日星期四 51](#_Toc514918179)

[继续开发包装扫码工具 51](#_Toc514918180)

[unshift() 51](#_Toc514918181)

[html5（注意下面所述的都是input所属的属性） 51](#_Toc514918182)

[form属性 52](#_Toc514918183)

[formaction属性 52](#_Toc514918184)

[formtarget属性 52](#_Toc514918185)

[placeholder属性 53](#_Toc514918186)

[pattern属性 53](#_Toc514918187)

[required属性 53](#_Toc514918188)

[音视频标签 53](#_Toc514918189)

[html5新增的元素 54](#_Toc514918190)

[27· 2018年5月4日星期五 54](#_Toc514918191)

[html5块级元素 54](#_Toc514918192)

[28· 2018年5月7日星期一 58](#_Toc514918193)

[数组清空 58](#_Toc514918194)

[29· 2018年5月8日星期二 58](#_Toc514918195)

[滚动条 58](#_Toc514918196)

[scrollTop 59](#_Toc514918197)

[30· 2018年5月9日星期三 60](#_Toc514918198)

[添加或删除节点(html) 60](#_Toc514918199)

[31· 2018年5月10日星期四 61](#_Toc514918200)

[git理解 61](#_Toc514918201)

[32· 2018年5月11日星期五 61](#_Toc514918202)

[nwjs应用打包 61](#_Toc514918203)

[33· 2018年5月12日星期六 61](#_Toc514918204)

[34· 2018年5月14日星期一 62](#_Toc514918205)

[35· 2018年5月15日星期二 62](#_Toc514918206)

[36· 2018年5月22日星期二 62](#_Toc514918207)

[37· 2018年5月23日星期三 62](#_Toc514918208)

[38· 2018年5月24日星期四 62](#_Toc514918209)

## 2018年4月2日星期一

### 入职第一天

填写合同，交付相关资料文件，领取电脑，并没有领取到笔记本和笔

## 2018年4月3日星期二

### 安装windows操作系统

## 2018年4月4日星期三

### 安装IDE

## 2018年4月5日星期四

### 学习Git

## 2018年4月6日星期五

### 无任务

## 2018年4月7日星期六

### Git切换分支

Git checkout [name]

### 更新修改

每次修改文件后，怎么再推送到远端仓库（或者是需要不需要，还仅是commit）

### 小结

#### 1）

任何人在使用git之前，都要提交简单的个人信息，以便git区分不同的提交者身份

$ git config –gloobal user.name “your name”

$ git config –global user.email [yourname@example.com](mailto:yourname@example.com)

#### 2）

想要开启一个项目，应该先建立一个目录，例如命名为myproject，然后所有的项目开发内容都在此目录下进行。

$ cd myproject

$ git init

$ git add .

$ git commit //这个步骤会自动进入编辑状态，要求提交者输入有关本次提交的开发信息

#### 3）

如果改进了项目源代码，并且到了开发者认为“应该再次记录开发信息”的时候，则提交“工作成果”。

$ git commit –a //这是一个偷懒的命令，相当于git add .;git commit;

但是，此处有一点需要注意，那就是git commit –a 无法把新增文件或文件夹加入进来，所以，如果你新增了文件或文件夹，那么就要老老实实的先git add . ,再git commit

#### 4）

想检查到目前为止对源码都做了哪些修改（相对于本次工作刚开始时）：

$ git diff //这个命令只在git add之前使用有效，如果已经git add了，那么此命令输出为空

$ git diff –cached //这个命令在git add之后在git commit之前有效

$ git status //这个命令在git commit之前有效，表示都有那些文件发生了改动

#### 5）

想要查看子项目开始到现在的所有开发日志

$ git log

$ git log –p //会输出非常详细的日志内容

#### 6）

开启一个试验分支（experimental），如果分支开发成功则合并到主分支（master），否则放弃该试验分支。

$ git branch experimental //创建一个试验分支，名叫experimental

$ git branch //显示当前有哪些分支，其中标注\*为当前分支

$ git checkout experimental //切换到experimental分支

（省略在此分支上的开发过程）。。。

##### 如果分支开发成功:

$ git commit –a //在experimental分支改进完代码之后用commit在此分支中进行提交

$ git checkout master //然后切换到master分支

$ git merge experimental //经证实分支开发成功（如果），将experimental分支合并到主分支上

$ git commit –a //彻底完成此次分支合并，即提交master分支

$ git branch –d experimental //因为experimental分支已经提交，所以可以安全删除此分支

##### 如果分支开发失败：

$ git checkout master

$ git branch –D experimental //由于分支被证明失败，因此使用 –D 来放弃并删除此分支

#### 7）

随时查看图形化分支信息

$ gitk

### 分配任务

一个 [root@111.230.156.186](mailto:root@111.230.156.186) 怎么去远程登录？通过git bash 命令行登录，

$ ssh [root@111.230.156.186](mailto:root@111.230.156.186)即可登录。

### 代理相关

git config --global http.proxy ‘socks5://127.0.0.1:1080’

git config –global https.proxy ‘socks5://127.0.0.1:1080’

取消代理

git config –global --unset http.proxy

git config –global --unset https.proxy

git push 或者clone出错 代码 errno 10054

解决方案：增大postBuffer

git config –global http.postBuffer 1048576000

git 初始化配置

#选择你喜欢的文本编辑器

git config –global core.editor gvim

git log详细用法：

git log –author=”liang” –p hello.py

这个命令会显示liang作者对hello.py文件所做的所有更改的差异比较（diff）

git log –online master..some-featurre

..句法是比较分支很有用的工具，这个命令显示了在some-feature分支而不在master分支的所有提交的概览



## 2018年4月9日星期一

### 在centos（一种linux社区发行版）上安装jdk

有四种方式（具体参考：https://www.linuxidc.com/Linux/2016-09/134941.htm）：

这里采用yum安装jdk

首先说明一下yum：Yum（全称为 Yellow dog Updater, Modified）是一个在Fedora和RedHat以及CentOS中的[Shell](https://baike.baidu.com/item/Shell" \t "_blank)前端软件包管理器。基于[RPM](https://baike.baidu.com/item/RPM" \t "_blank)包管理，能够从指定的服务器自动下载RPM包并且安装，可以自动处理依赖性关系，并且一次安装所有依赖的软件包，无须繁琐地一次次下载、安装

1. # yum search java|grep jdk
2. 选择版本，进行安装，这里安装jdk1.7版本（[root@localhost ~]# yum install java-1.7.0-openjdk）
3. 安装完成后默认的安装目录是在 /usr/lib/jvm/java-1.7.0-openjdk-1.7.0.75.x86\_64
4. 设置环境变量：[root@localhost ~]# vi /etc/profile 在profile文件中添加如下内容

#set java environment  
JAVA\_HOME=/usr/lib/jvm/java-1.7.0-openjdk-1.7.0.75.x86\_64  
JRE\_HOME=$JAVA\_HOME/jre  
CLASS\_PATH=.:$JAVA\_HOME/lib/dt.jar:$JAVA\_HOME/lib/tools.jar:$JRE\_HOME/lib  
PATH=$PATH:$JAVA\_HOME/bin:$JRE\_HOME/bin  
export JAVA\_HOME JRE\_HOME CLASS\_PATH PATH

1. 然后让上面的修改（即设置环境变量的内容）生效：[root@localhost java]# source /etc/profile
2. 最后一步，验证jdk是否成功安装：[root@localhost java]# java –version
3. 注意以上所有操作，可以看见都是 # ，说明是以权限高的用户操作的。
4. 注意 jdk 的多有配置都是在 /etc/profile这个文件下,如果要修改path,classpath,java\_home等，都要进入这个文件
5. **五. 卸载jdk**   
   ·找到jdk安装目录的\_uninst子目录   
   ·在shell终端执行命令./uninstall.sh即可卸载jdk。

### 安装tomcat

1. 下载：我这里提前进去到/usr/local目录（cd /usr/local），输入以下命令

wget http: //mirrors.shuosc.org/apache/tomcat/tomcat-8/v8.5.24/bin/apache-tomcat-8.5.24.tar.gz

2） 解压：tar zxcf apachetomcat-8.5.24.tar.gz

3） 重命名：mv apache-tomcat-8.5.24 tomcat

4） 启动tomcat： /usr/local/tomcat/bin/startup.sh

5） 查看tomcat服务详细信息：# ps -ef|grep tomcat

### 安装mysql（参考：<https://www.jianshu.com/p/1dab9a4d0d5f>）

不推荐使用wget方式安装，以为mysql依赖包比较多，难以处理，用yum

1. 先查看linux系统中是否有安装过mysql： rpm –aq | grep –i mysql;这个命令可以实现两个效果，如果没有结果，说明没有；如果有，则可以看到版本
2. 先切换到要放置mysql软件的目录（没有先创建这个文件夹，一般放在/usr/local下），这里我新建了一个目录mysql，并进入其下，cd /usr/local/mysql 。（再次重申 / 表示根目录）
3. 使用wget下载 wget <https://dev.mysql.com/get/Downloads/MySQL-5.7/mysql-5.7.21-linux-glibc2.12-x86_64.tar.gz>
4. 如果提示wget commond not found（一般不会遇到），说明你centos还没有安装wget配置。这个时候需要执行这个命令：yum –y install wget 。等待你的centos安装wget配置，当你看到这个的时候，说明安装成功。
5. 注意如果出现 yum，wget等所有操作，都会出现commond not found时，可能是断开连接了，你重新登陆一下就好了
6. 解压mysql：因为我是将安装包下载在 /root目录下，现在我执行下面这个命令将其复制到 /usr/local 下面（这个不要盲目复制，你要根据你自己的实际目录来，我这里是为了演示复制操作）。

cp mysql-5.7.21-linux-glibc2.12-x86\_64.tar.gz ../usr/local

然后进入 /usr/local目录下去解压，德奥文件夹mysql-5.7.21-linux-glibc2.12-x86\_64，将其命名为mysql，具体操作如下：

cd ../usr/local

tar –zxvf mysql-5.7.21-linux-glibc2.12-x86\_64.tar.gz

重命名一个文件mv mysql-5.7.21-linux-glibc2.12-x86\_64 mysql

1. 安装mysql：
2. 新建用户组和用户

执行以下命令：

cd mysql

groupadd mysql

useradd –r –g mysql mysql

【说明一下，useradd –r 参数表示mysql用户是系统用户，不可用于登录系统；useradd –g参数表示把mysql用户添加到mysql用户组中】

1. 创建目录并授权

mkdir data mysql-files

chown –R mysql

chgrp –R mysql

1. 初始化MySQL

bin/mysqld –initialize –user=mysql

1. 报错问题

如果你报这个错，说明你缺少包 libaio和libaio-devel

bin/mysqld: error while loading shared libraries: libaio.so.1: cannot open shared object file: No such file or directory

解决办法

这时你需要执行下面这个命令来安装

yum install libaio\*

安装好后，再次执行

bin/mysqld –initialize –user=mysql

1. 







1. 以上具体操作请参阅：<https://blog.csdn.net/bazhuayu_1203/article/details/78828641>

### Linux删除操作

Linux删除文件夹命令--rm, rmdir

linux删除目录很简单，很多人还是习惯用rmdir，不过一旦目录非空，

就陷入深深的苦恼之中，现在使用rm -rf命令即可。

直接rm就可以了，不过要加两个参数-rf

即：rm -rf 目录名字

-r 就是向下递归，不管有多少级目录，一并删除

-f 就是直接强行删除，不作任何提示的意思

删除文件夹实例：

#rm -rf /opt/real/RealPlayer

将会删除/opt/real/RealPlayer目录以及其下所有文件、文件夹

需要提醒的是：使用这个rm -rf的时候一定要格外小心，linux没有回收站的

当然，rm还有更多的其他参数和用法，man rm就可以查看了

删除文件使用实例：

rm -f /var/log/httpd/access.log

将会强制删除/var/log/httpd/access.log这个文件

### 查看有哪些用户

使用命令：

cat /etc/passwd|grep -v nologin|grep -v halt|grep -v shutdown|awk -F":" '{ print $1"|"$3"|"$4 }'|more

当然了，如果我们提前知道都有哪些用户，我们可以直接精确查看这个用户，

命令：cat /etc/passwd|grep 用户名 #这条命令可以用来精确查找某个用户。

还有查看所有组信息：

命令：cat /etc/group

当然了，查看用户组和查看用户的使用规则是一样的，同样可以具体到某一个组

命令：cat /etc/group|grep 组名，用于查找某个用户组

总结一下用户和用户组常用命令：

groups 查看当前登录用户的组内成员

groups test 查看test用户所在的组，以及组内成员

whoami 查看当前登录用户名

### 添加用户

useradd 用户名

### 设置密码

passwd 用户名 【--必须写明给谁设密码；不写默认给当前用户设置】

### 删除用户

如果一个用户的账号不再使用，可以从系统中删除。删除用户账号就是要将/etc/passwd等系统文件中的该用户记录删除，必要时还删除用户的主目录。

删除一个已有的用户账号使用userdel命令，其格式如下：

userdel 选项 用户名

常用的选项是 -r ，它的作用是把用户的主目录一起删除。

例如：

 userdel -r sam

解释：此命令删除用户sam在系统文件中（主要是/etc/passwd, /etc/shadow, /etc/group等）的记录，同时删除用户的主目录。

### 修改账号

修改用户账号就是根据实际情况更改用户的有关属性，如用户号、主目录、用户组、登录Shell等。

修改已有用户的信息使用usermod命令，其格式如下：

usermod 选项 用户名

常用的选项包括-c, -d, -m, -g, -G, -s, -u以及-o等，这些选项的意义与useradd命令中的选项一样，可以为用户指定新的资源值。

另外，有些系统可以使用选项：-l 新用户名

这个选项指定一个新的账号，即将原来的用户名改为新的用户名。

例如：

usermod -s /bin/ksh -d /home/z -g developer sam

解释：此命令将用户sam的登录Shell修改为ksh,主目录改为/home/z  用户组改为developer

### 用户口令管理

用户管理的一项重要内容是用户口令的管理。用户账号刚创建时没有口令，但是被系统锁定，无法使用，必须为其指定口令后才可以使用，即使是指定空口令。

指定和修改用户口令的Shell命令是passwd。超级用户可以为自己和其他用户指定口令，普通用户只能用它修改自己的口令。命令的格式为：

passwd 选项 用户名

可使用的选项：

-l 锁定口令，即禁用账号。

-u 口令解锁。

-d 使账号无口令。

-f 强迫用户下次登录时修改口令。

如果默认用户名，则修改当前用户的口令。

例如，假设当前用户是sam，则下面的命令修改该用户自己的口令：

$ passwd

如果是超级用户，可以用下列形式指定任何用户的口令：

# passwd sam

### 直接将普通用户权限提升到root

# chmod –R 777 /home/liangshuai

（注意，所有的用户资料信息都保存在/home文件夹下面）

### Chmod理解

chmod命令是修改权限的，但是要明白一点，就是它是修改什么权限的。要知道，这个命令只是用来修改某个文件或文件夹的权限的，即：将某一个文件夹或文件变成执行，读写等权限。将一个不可写的文件修改为可写

【注意，并没有涉及到人，只是修改的是文件的读写等权限】

### Chown理解

chown是改变文件或目录所属用户

改的时候先进入 /etc/passwd 下看看有哪些用户，然后再修改

chown 想修改成的目标用户 文件或文件夹(即目录)

## 2018年4月10日星期二

### 部署项目

通过上面命令，我们可以看到启动用户，进程id，配置等信息。对于结束服务我们可以在bin目录执行./shutdown.sh。也可以使用下面命令

[root@xiehui bin]# kill -9 5110

[root@xiehui bin]# ps -ef|grep tomcat

root 5211 4533 0 11:18 pts/0 00:00:00 grep --color=auto tomcat

kill -9后面的数字就是服务的进程号，如果想启动再次执行./shutdown.sh就可以了。再次把服务打开，然后在本机访问Linux Tomcat ，发现并不能访问，但是在Linxu内部的浏览器却是可以的，这种原因一般是防火墙的问题。关闭下防火墙就可以访问了

[root@xiehui sysconfig]# systemctl stop firewalld

因为配置的是NAT，即使本地和虚拟机不在一个网段也是可以访问的。安装好Tomcat后，在webapps下面自带一个demo

[root@xiehui apache-tomcat-8.5.8]# cd webapps/

[root@xiehui webapps]# ls

docs examples host-manager manager ROOT

[root@xiehui webapps]#

发现此时打开了部署的web项目。



部署项目只需要把把项目复制到Tomcat的Webapps下即可。这也是最简单的一种方式。如果我们想更改我们的tomcat服务接口，那么可以到tomcat下的conf目录，里面有个server.xml文件。重要的配置信息在这里

<Connector port="8080" protocol="HTTP/1.1"

connectionTimeout="20000"

redirectPort="8443" />

<Host name="localhost" appBase="webapps"

unpackWARs="true" autoDeploy="true">

在配置文件中有一个unpackWARs参数，如果设置为false即可以部署war包项目，tomcat会自动解压文件。还可以在server.xml中使用Context标签指定项目在任何位置。

<Context path="定web应用的虚拟路径名" docBase="要部署的Web应用的源路径（实际位置）" reloadable="true" />

到这里，通过自己的学习并实践将项目部署掌握了，从此部署项目So easy。应了那句话自己动手丰衣足食。   
最后再发几个常用命令

df -h 查看磁盘使用信息

vi filename 查看文件输入后进入命令行模式，命令行模式输入I进入插入模式，此时可更改文件内容，更改完成，按Esc退出编辑模式到命令行模式

: wq (输入「wq」，存盘并退出vi)

: q! (输入q!， 不存盘强制退出vi)

cat /proc/version 查看正在运行的内核版本

cat /etc/issue 显示的是发行版本信息

lsof -i:端口号 查看占用某端口的进程

ps aux | grep 进程号 根据进程ID查看进程详细信息

cat /proc/cpuinfo |grep "physical id"|sort |uniq|wc -l 物理CPU的个数

cat /proc/cpuinfo |grep "processor"|wc -l 逻辑CPU的个数

cat /proc/cpuinfo |grep "cores"|uniq 查看CPU内核数

top 查看系统负载，CPU，存储,进程等详细的使用信息

### Elasticsearch安装

1. 到官网下载好tar包（其实最好是以普通用户身份去下载安装）
2. 利用xftp工具将这个压缩包上传到linux服务器上（提前先选定好目录/usr/local下）
3. 解压，删除压缩包
4. 然后改变目录的用户权限，然后再以普通用户启动



### 【注意】

：在安装与elasticsearch相关的插件时，我们以运行elasticsearch的这个用户来安装时，会避免掉后续的许多权限问题，尽量不要以root安装。

大部分情况下需要修改配置文件中的host地址值为：0.0.0.0

可能还会进行这个操作：sudo sysctl -w vm.max\_map\_count=262144(如果报这个错误的话)

## 2018年4月11日星期三

### Mysql 安装后修改密码

答案来源（https://stackoverflow.com/questions/41645309/mysql-error-access-denied-for-user-rootlocalhost）

1. Open & Edit /etc/my.cnf
2. Add skip-grant-tables under [mysqld]
3. Restart Mysql
4. You should be able to login to mysql now using the below command mysql -u root -p
5. Run mysql> flush privileges;
6. Set new password by *ALTER USER 'root'@'localhost' IDENTIFIED BY 'NewPassword';*
7. Go back to /etc/my.cnf and **remove/comment** skip-grant-tables
8. Restart Mysql
9. Now you will be able to login with the new password mysql -u root –p

### 给一个用户赋予某个文件夹（及旗下的）的权限

首先因为是给一个用户赋予权限，因此进行操作的这个用户必须有这个授予权限的权限，因此一般是切换到root下操作：

# chown –R liangshuai usr/local/mysql #这句话的意思是授予liangshuai这个用户访问usr/local/mysql以及mysql旗下的权限。

### 关闭防火墙

sudo systemctl stop firewalld.service

### 安装Kinaba

1. 组成：Elasticsearch+logstash+kinaba+redis日志分析系统。ELK有这三部分组成，Elasticsearch是一个开源分布式搜索引擎，它的特点有：分布式，零配置，自动发现，索引自动分片，索引副本机制，restful风格接口，多数据源，自动搜索负载等。当然它集存储，搜索和分析功能；Logstash是一个开源的工具，它可以对你的日志进行收集，分析，并将其存储提供以后使用；kibana是一个开源和免费的工具，它可以为logstash和Elasticsearch提供日志分析友好的web界面，可以帮助你汇总，分析和搜索重要数据日志。
2. 组件：Logstash：logstash server端用来搜集日志；

Elasticsearch：存储各类日志；

Kinaba：web化接口用作查询和可视化日志；

Logstash Forwarder：logstash client端用来通过lumberjack网络协议发送到logstash server；

1. 工作流程：在需要收集日志的所有服务器上部署logstash，，作为logstash\_agent（logstash shipper）用于监控并过滤收集日志，将过滤后的内容发送到redis，然后通过logstash\_server将日志收集在一起交给全文搜索服务Elasticsearch，可以用Elasticsearch进行自定义搜索通过kibana来结合自定义搜索进行页面展示

### 修改当前用户下的vim编辑器颜色

直接以当前用户的身份打开: vim ~/.vimrc 文件，修改成如下代码：

colorscheme evening #（全文件就只有这一行代码即可）

## 2018年4月12日星期四

### 1.Node与cluster

1.1 Elasticsearch本质上是一个分布式数据库，允许多台服务器协同工作，每台服务器可以运行多个Elastic实例。单个Elastic实例称为一个节点（node），一组节点构成一个集群（cluster）。

1．2 Elastic会索引所有字段，经过处理后写入一个反向索引（Inverted Index），查找数据的时候，直接查找该索引。所以，elastic数据管理的顶层单位就叫做index，它是单个数据库的同义词，每个index（即数据库）的名字必须是小写。

1.3 下面的命令可以查看当前节点的所有的index

$ curl -X GET ‘http://localhost:9200/\_cat/indices?v’

### 2.document

2.1 index里面单条的记录称为document（文档），许多条document构成了一个index

2.2 document使用json格式表示，下面是一个例子

{

“user”:”张三”

“title”:”工程师”,

“desc”:”数据库管理”

}

同一个index里面的document，不要求有相同的结构（scheme），但是最好保持相同，这样有利于提高搜索效率

### 3.type

3.1 document可以分组，比如weather这个index里面，可以按城市分组（北京和上海），也可以按气候分组（晴天和雨天），这种分组就叫做type，他是虚拟的逻辑分组，用来过滤document

3.2 不同的type应该有相似的结构（scheme），举例来说，id字段不能在这个组是字符串，在另一个组是数值。这是与关系型数据库的表的一个区别，性质完全不同的数据（比如products和logs）应该存成两个index，而不是一个index里面的两个type（虽然可以做到）

下面的命令可以列出每个index所包含的type

$ curl ‘localhost:9200/\_mapping?pretty=true’

### 新建和删除index

4.1 新建index，可以直接向Elastic服务器发出put请求，下面的例子是新建一个名叫weather的index。

$ curl –X PUT ‘localhost:9200/weather’

4.2 服务器返回一个json对象，里面的acknowledged字段表示操作成功

{

"acknowledged":true,

"shards\_acknowledged":true

}

* 1. 然后，我们发出delete请求，删除这个index

$ curl -X DELETE 'localhost:9200/weather'

* 1. 在kibana中新建索引：

这个是索引

PUT /shakespeare

{

"mappings": {

"doc": {

这个是type

"properties": {

"speaker": {"type": "keyword"},

"play\_name": {"type": "keyword"},

属性（即字段吧）

"line\_id": {"type": "integer"},

"speech\_number": {"type": "integer"}

}

}

}

}

新建索引用的是put命令，添加数据用post或者get，查询记录用get

然后给新建的这个索引添加数据

这个叫filed（字段）

post shakespeare/doc

{

"speaker":"李四",

"play\_name":"李四",

这个是integer类型，不需要加双引号

"line\_id":9,

"speech\_number":2

}

## 2018年4月13日星期五

### 查询索引库下的数据

这是type

get /bjsxt/test/\_search

{

"query":{

"match\_all": {}

}

这个是索引库

}

### 带条件的查询索引

|  |
| --- |
| get /bjsxt/test/\_search?q=name:zengyue  {  "query":{  "match\_all": {}  }  }  这里的q是一个固定的写法，想带什么参数就跟到后面 |

### 一个创建索引例子的完整说明

#### 1.以下写法是基于终端的：

|  |
| --- |
| curl -XPUT http://localhost:9200/shakespeare -d '  {   "mappings" : {    "\_default\_" : {     "properties" : {      "speaker" : {"type": "string", "index" : "not\_analyzed" },      "play\_name" : {"type": "string", "index" : "not\_analyzed" },      "line\_id" : { "type" : "integer" },      "speech\_number" : { "type" : "integer" }     }    }   }  }  '; |

这个例子中mapping（映射）指定了数据集下列特质：

1. speaker字段是不可拆分的字符串，在这个filed（字段）中的字符串被视为一个单独的单元，即使在这个filed（字段）中有多个单词。
2. 这同样适用于play\_name字段。
3. line\_id和speech\_number字段是整数。
4. -d 代表是一条document，会在put和post命令（插入数据或是修改）中出现

#### 2.映射

日志数据需要映射，通过将geo\_point类型应用于这些字段，，将日志中的latitude（纬度）/longitude（经度）对标记为地理位置。

使用以下命令监理日志geo\_point mapping（映射）

|  |
| --- |
| curl -XPUT http://localhost:9200/logstash-2015.05.18 -d'  {  "mappings":{  "log":{  "properties":{  "geo":{  "properties":{  "coordinates":{  "type":"geo\_point"  }  }  }  }  }  }  }  ';  curl -XPUT http://loocalhost:9200/logstash-2015.05.19 -d '  {  "mappings":{  "log":{  "properties":{  "geo":{  "properties":{  "coordinates":{  "type":"geo\_point"  }  }  }  }  }  }  }  ';  curl -XPUT http://localhost:9200/logstash-2015.05.20 -d '  {  "mappings": {  "log": {  "properties": {  "geo": {  "properties": {  "coordinates": {  "type": "geo\_point"  }  }  }  }  }  }  }  '; |

## 2018年4月14日星期六

### 重新梳理linux各种命令

#### 1.shell

shell是系统的用户界面，提供了用户与内核进行交互操作的一种接口。它接受用户输入的命令并把它送入内核去执行。实际上shell是一个命令解释器，它解释由用户输入的命令并且把它们送到内核，不仅如此，shell有自己的编程语言用于对命令的编辑，它允许用户编写由shell命令组成的程序，shell编程语言具有普通编程语言的很多特点，比如它也有循环结构和分支结构等，用这种编程语言编写的shell程序与其他应用程序具有同样的效果。

#### 2.vi和vim

直接使用vi打开文件，如果文件存在打开存在文件；如果不存在的话，将以我们指定的参数作为文件名。输入q！是强制退出，同时没有保存当前文件内容。如果文件没有进行修改，或者已经保存完毕也可以使用q退出。保存退出使用wq，保存强制退出wq！

#### 3.新建用户

使用shell命令建立用户有两个命令‘useradd’和‘adduser’，它们是一样的，使用的情况是这样的，因为linux系统中用户必须归属于某个组，所以如果新建用户时，不指定组命令就会建立一个与用户名称一模一样的组，将新建的用户放进去。如果新建用户时指定了组，用户就归属于指定的组。

我们练习一个新建用户时指定组的例子： # useradd –g hello yes ：新建了一个用户yes，并且将它的组指定为已经存在的hello组。

#### 4.echo

echo功能：显示文字，常用参数:-n –e

#### 5.file

file：判断文件类型：file [参数] 文件名

#### 6.chmod

功能：改变权限命令。

常用参数：（1=执行权 ，2=可写，4=可读，1+2=3执行和可写，以此类推）

提示：chmod 参数 文件或目录名,这个命令是用来修改文件的权限的

#### 7.chgrp

功能：改变文件或目录的所属组

常用参数

提示：chgrp 所属组 文件或目录名

修改的时候进入 /etc/group 看看有什么组

#### 8.chown

功能：改变文件或目录所属用户

提示： chown 所属用户 文件或目录名

修改的时候进入 /etc/passwd这个文件下看看有什么用户

#### 9.tar

功能：主要是归档和解档功能

常用参数： -c-v-f-z-x

提示：tar 【主选项和辅选项】 文件或目录

#### 10.whoami

显示自身用户名称

#### 11.top

显示程序进程 top pid

#### 12.ps

功能：显示程序命令

#### 13. whereis

可以使用 whereis xxx 命令，来查找 xxx 程序的安装位置，详见 搜索。

## 2018年4月16日星期一

### kibana学习

#### DSL语言

所有的linux涉及权限的都可以用chmod和chown两个命令搞定，操作是：先用chmod命令将这个文件修改成所有者可执行（一旦是可执行就包括读写了），组用户可读；第二步用chown命令将这个文件的所有者赋予目标人物即可，截至目前个人理解是这个（有待验证）。

sudo只是强行提升一个人的身份：将这个人以root（以root为例）身份去做一些操作。

在dsl语言下，match\_all不能与条件查询连用，会报错，因为match\_all本身意思就是不带条件的查询所有。

\_score是按照文档与查询的相关度的相关度的一个搜索结果。

-d是一个document记录

### 精确查询

#### term查询

我们首先来看最为常用的 term 查询， 可以用它处理数字（numbers）、布尔值（Booleans）、日期（dates）以及文本（text）。一定要注意term是包含操作，而不是等值操作

GET /my\_store/products/\_search

{

注意‘{’不能放在第一行

"query" : {

"constant\_score" : {

"filter" : {

"term" : {

"price" : 20

}

}

}

}

}

我们用 constant\_score 将 term 查询转化成为过滤器

#### 组合过滤器

bool就是复合过滤器，一个bool过滤器由三部分组成

{

"bool" : {

"must" : [],

"should" : [],

"must\_not" : [],

}

}

must

所有的语句都 必须（must） 匹配，与 AND 等价。

must\_not

所有的语句都 不能（must not） 匹配，与 NOT 等价。

should

至少有一个语句要匹配，与 OR 等价。

就这么简单！ 当我们需要多个过滤器时，只须将它们置入 bool 过滤器的不同部分即可。

note: 一个 bool 过滤器的每个部分都是可选的（例如，我们可以只有一个 must 语句），而且每个部分内部可以只有一个或一组过滤器。

#### range范围查询

range 查询可同时提供包含（inclusive）和不包含（exclusive）这两种范围表达式，可供组合的选项如下：

gt: > 大于（greater than）

lt: < 小于（less than）

gte: >= 大于或等于（greater than or equal to）

lte: <= 小于或等于（less than or equal to）

range 查询同样可以应用在日期字段上：

"range" : {

"timestamp" : {

"gt" : "2014-01-01 00:00:00",

"lt" : "2014-01-07 00:00:00"

}

}

## 2018年4月17日星期二

### elasticsearch java api

大体上有两种写法：

现在版本的java api是调用的这个方法：Java High Level REST Client方式；具体参考这个链接：

<https://artifacts.elastic.co/javadoc/org/elasticsearch/elasticsearch/6.2.3/org/elasticsearch/action/admin/indices/create/CreateIndexResponse.html?is-external=true>

## 2018年4月18日星期三

nwjs的项目运行步骤：--先进入到nwjs项目所在的位置—必须进入到这个项目的层级目录中然后安装npm install 【注意这个只需要安装一次】---然后在当前项目下面运行nw.exe . 命令【注意这里有一点很重要，就是nw.exe命令前面必须进入正确的nwjs安装目录下面，但是dos命令执行目录一直是在当前项目下】（其实比较好的一种方式是配置一下nwjs的path路径，这是我实验出来的，这样就不需要每次记那些nwjs的安装路径了）

### 抓包工具：wireshark

文件后缀名是 .msi表示这是一个windows安装包

文件后缀名是 .exe表示这是一个二进制文件包

文件后缀名是 .tar.gz 表示这是一个linux 系统版本的文件

### node.js组成模块

node.js应用由这几部分组成：\*.引入required模块：我们使用require指令来载入node.js模块\*.创建服务器：服务器可以监听客户端的请求，类似于Apache nginx等http服务器

\*.接收请求预想请求 服务器很容易创建，客户端可以使用浏览器或终端发送http请求，服务器接收请求后返回响应数据

### 谈一谈path

当我们在桌面点击一个应用程序时，系统首先会在当前用户的目录下去找这个命令，若是没有，则会进入path中继续去找，直到找到或者报错说找不到；因此我们在path中配置的命令就是相当于是一个快捷方式，直接将某个应用程序的安装位置告诉了path，然后再输入命令时（注意这里我自测的结果是不区分命令的大小写）系统就会找的到。

### node.js程序的运行方式

必须用node.js程序去打开，应该有两种方式：一种命令行方式；另一种客户端（目前没接触还不会），命令行的方式前提是我们必须在path中配置了node.js的路径，然后找到并进入要运行的目标文件，以node 前缀加文件名的方式打开运行即可。

### node.js程序编写

var x = require(‘y’)

x(options,callback)大概就是这种模式吧。

变量声明需要使用var关键字，

console.log（“hello world”）这个语句运行后会输出hello world和undefined原因是hello world是动作输出结果。而console.log这个语句本身是没有值的，是undefined

### 回调函数

所谓的回调函数是一种异步执行方式：node.js异步编程的直接体现就是回调，异步编程依托于回调来实现，比如可以一边读取文件，一边执行其他命令（这句话的更确切理解是一边读取文件，一边执行这个读取文件所在语句的下一条命令），在文件读取完成后，我们将文件内容作为回调函数的参数返回，大大提高了node.js的性能，可以处理大量的并发请求。



一旦input.txt读取完成，回调函数function（）就会被调用，这时整个已经读取完成的txt文本就会作为回调函数的参数（即data字段）返回，这样在执行代码时就没有阻塞或等待文件i/o操作。

input.txt一开始读取时，console.log（“程序执行结束”就开始运行了），两个命令是并发的

特别重要的一点：这里判断是异步编程是因为fs调用了异步方法readFile（），而不是调用的readFileSync（）这个方法（这是一个同步方法），而绝不是因为看到readFile（）方法里面有一个回调函数，要特别理解这句话“异步编程依托于回调来实现，但不能说使用了回调后程序就异步化了。”

## 2018年4月19日星期四

### 异步编程的理解

|  |
| --- |
| var fs = require("fs");  fs.readFile("input.txt",function(err,data){  if(err) return console.error("err----> "+err);  console.log("input---> "+data.toString());  });  var out = fs.readFileSync('output.txt');  console.log("output---> "+out.toString()); |

这段代码的执行，经过测试，一定是在output.txt这个文件完全操作完，function（err，data）这个回调函数才会执行（但是readFile（）方法是和readFileSync（）同时执行的），并不会出现回调函数抢占下面这个函数的情况出现。

### if-while-do while的精细理解

if：if代码块是如果条件满足，那么执行，不满足不执行，单独来说不具备循环调用；

while：当条件满足时就一直在while代码块里执行，除非条件不满足才会跳出来；

do-while：是一个关于先做再判断，还是先判断再做的问题。

### node.js的模块加载

node.js的模块加载：在node.js中，有一个很强大的地方就是模块加载系统，在node.js中模块和文件是一 一对应的（每一个文件都被视作为一个独立的模块），这个文件可能是js代码，json或c/c++扩展，因为我们知道js本身就是c的一个扩展。

而加载模块需要的一个命令就是：require（）这个module6之前制霸的命令。现在新版本有其它的命令了好像。

### 事件驱动

1）要引入events（事件）模块:var events = require(‘events’)

2）创建事件驱动对象：var eventEmitter = new events.EventEmitter()

3）创建这个事件驱动程序索要执行的动作

4）绑定事件：eventEmitter.on(‘data’,function(){});

5）触发事件：eventEmitter.emit

data是你绑定的事件，这个绝对不能随便写，这个实例化对象有这个事件才可以去绑定;以后绑定事件时都是同样的道理

6）我把EventEmitter类叫做事件驱动类

### setTimeout(code，millisec)用法

setTimeout(code,millisec)方法用于在指定的毫秒数之后调用函数或计算表达式

code：必须，要调用的函数后（这里指得是调用setTimeout()函数后）要执行的js代码

millisec：必须，在执行代码前需要等待的毫秒数

这句话表明code可以是一个函数或者可以计算的计算表达式

## 2018年4月20日星期五

### 最简单的文件复制

|  |
| --- |
| private String fileWritePath = “C:\\writer.txt”;  privater String fileReadPath = “C:\\reader.txt”;  FileReader fr = new FileReader();  FileWriter fw = new FileWriter();  int len;  len = fr.read();  while(len != -1){  fw.write(len);  len = fr.read();  }  fr.close();  fw.close(); |

### js中三种定义变量的方式

1）const定义的变量不可以修改，而且必须初始化

|  |
| --- |
| 1 const b = 2;//正确  2 // const b;//错误，必须初始化  3 console.log('函数外const定义b：' + b);//有输出值  4 // b = 5;  5 // console.log('函数外修改const定义b：' + b);//无法输出 |

2）var定义的变量可以修改，如果不初始化会输出undefined，不会报错

|  |
| --- |
| 1 var a = 1;  2 // var a;//不会报错  3 console.log('函数外var定义a：' + a);//可以输出a=1  4 function change(){  5 a = 4;  6 console.log('函数内var定义a：' + a);//可以输出a=4  7 }  8 change();  9 console.log('函数调用后var定义a为函数内部修改值：' + a);//可以输出a=4 |

3）let是块级作用域，函数内部使用let定义后，对函数外部没有影响

|  |
| --- |
| 1 let c = 3;  2 console.log('函数外let定义c：' + c);//输出c=3  3 function change(){  4 let c = 6;  5 console.log('函数内let定义c：' + c);//输出c=6  6 }  7 change();  8 console.log('函数调用后let定义c不受函数内部定义影响：' + c);//输出c=3 |

### node.js Buffer

#### 创建buffer缓冲区

buffer库为node.js提供操作i/o的功能。创建buffer对象使用buffer.from（）接口，

buffer类是专门用来存放二进制数据的缓冲区,注意每一个Buffer实例都对应于V8堆内存之外的一块原始内存，我们可以理解为每一个buffer对象（实例）都是一个整数数组。



[,这种写法表示第二个参数是可选的



#### 写入buffer缓冲区

buf.write(string[,offset[,length[,encoding):根据encoding的字符编码写入string到buf中的offset位置，length参数是写入的字节数，这个方法的返回值是返回实际写入的字节数大小，如果buf的空间不够，则只会写入部分字符串。注意buffer实例是一个数组。

|  |
| --- |
| **var *buf*** = Buffer.alloc(3); **var *len*** = ***buf***.write(**"01234567899876543210"**,0,5,**"utf8"**); **console**.log(**"写入的字节数："**+***len***); |

#### 从缓冲区读取数据

buf.toString(encoding,start,end);

|  |
| --- |
| **var *buf*** = Buffer.alloc(10); ***buf***.write(**"fmaomiaoemrjajfemofajf"**); **var *data*** = ***buf***.toString(**"utf8"**,0,10); **console**.log(**"buf中的数据："**+***data***); |

|  |
| --- |
| **var *buf*** = Buffer.alloc(26); **for**(**var *i*** = 0 ; ***i*** < 26 ; ***i***++){  ***buf***[***i***] = ***i*** + 97 ; } **console**.log(***buf***.toString(**'utf8'**));*//abcdefghijklmnopqrstuvwxyz* **console**.log(***buf***.toString(**'utf8'**,0,10));*//abcdefghij* **console**.log(***buf***.toString(**'utf8'**,5));*//fghijklmnopqrstuvwxyz* **console**.log(***buf***.toString(**''**,1,3));*//bc* **console**.log(***buf***.toString(**undefined**,1,3));*//bc* |

### buffer转换json对象

|  |
| --- |
| **const *buf*** = Buffer.from([1,2,3,4,5]); **const *json*** = ***JSON***.stringify(***buf***); *//JSON.stringify() 方法用于将 JavaScript 值转换为 JSON 字符串。* **console**.log(***json***); *//{"type":"Buffer","data":[1,2,3,4,5]}* **const *copy*** = ***JSON***.parse(***json***,**function**(){  **console**.log(**"1"**); }); |

### 缓冲区合并

|  |
| --- |
| **var *buffer1*** = Buffer.from(**'菜鸟教程 '**); **console**.log(***buffer1***);*//<Buffer e8 8f 9c e9 b8 9f e6 95 99 e7 a8 8b 20>* **var *buffer2*** = Buffer.from((**'www.runoob.com'**)); **var *buffer3*** = Buffer.concat([***buffer1***,***buffer2***]); **console**.log(**"buffer3 内容: "** + ***buffer3***.toString());*//buffer3 内容: 菜鸟教程 www.runoob.com* |

### 缓冲区比较

|  |
| --- |
| **var *buf\_1*** = **new** Buffer.from(**"buffer对象1"**); **var *buf\_2*** = **new** Buffer.from([**'buffer字符串数组对象2'**]); **var *res*** = ***buf\_2***.compare(***buf\_1***); **console**.log(***res***);*//-1;小于0表示buf\_2在buf\_1之前* |

### 拷贝缓冲区

buf.copy(targetBuffer[, targetStart[, sourceStart[, sourceEnd]]])

|  |
| --- |
| var buf1 = Buffer.from('abcdefghijkl');  var buf2 = Buffer.from('RUNOOB');  //将 buf2 插入到 buf1 指定位置上  buf2.copy(buf1, 2);  console.log(buf1.toString());//abRUNOOBijkl |

### 缓冲区裁剪和缓冲区长度

|  |
| --- |
| **var *buf\_1*** = **new** Buffer.from([0,9,0,7,1]); **var *res*** = ***buf\_1***.slice(0,3); **console**.log(***res***);*//<Buffer 00 09 00>* **console**.log(***buf\_1***.length);*//5* |

### txt字符流的拷贝

|  |
| --- |
| **import** java.io.**FileReader**; **import** java.io.**FileWriter**; **import** java.io.IOException;  public **class** IO {   private **String fileReader** = **"D:\\Documents\\GitHub\\udanTech\\linux.txt"**;  private **String fileWriter** = **"C:\\Users\\Administrator\\Desktop\\liangshuai\_20180402\\node.js.example\\fileWriter.txt"**;  private **FileReader fr** ;  private **FileWriter fw** ;  public **void** copy() {  **try**{  **fr** = **new FileReader**(**fileReader**);  **fw** = **new FileWriter**(**fileWriter**);  int **length**;  **length** = **fr**.read();  **while**(**length** != -1) {  **fw**.write(**length**);  **length** = **fr**.read(); *//这行代码相当于是游标，一直把游标往后移动（每次移动一个字符），如果没有这行代码，length是一个固定的值喽* }*//如果将while换成if，这块代码就只执行一次了，注意在此认真理解while和if，do while的区别* **fr**.close();  **fw**.close();  }**catch**(IOException e) {  e.getMessage();  }**finally** {   }  } } |

#### 自动插入数据到txt

|  |
| --- |
| **public** **class** CreateFileChar {  **public** **static** **void** main(String[] args) **throws** Exception {  String path1 = "C:\\Users\\Administrator\\Desktop\\liangshuai\_20180402\\node.js.example\\output.txt";  FileWriter fw = **new** FileWriter(path1);  **for**(**int** i=0;i<=100;i++) {  fw.write(i+"\n");  }  fw.close();  }  } |

### 字节流的操作（以图片为例）

|  |
| --- |
| **import** java.io.**File**; **import** java.io.FileInputStream; **import** java.io.FileOutputStream;  public **class** CopyPhotoByte {   public static *void* main(String[] **args**) **throws Exception**{  **String src** = **"C:\\1.jpg"**;  **String cpname** = **src**.substring(3, **src**.**length**());  **String tar** = **"E:\\Users\\Administrator\\Desktop\\liangshuai\_20180402\\node.js.example"**;   *//判断tar的路径对不对（即完全存在不在）* **File file** = **new File**(**tar**);  **if**(!**file**.exists()) {  *//如果上述目录不存在，则创建这个目录* **System**.**out**.println(1);  **file**.mkdirs();*//注意这里若是多级目录调用mkdirs()，一级目录用mkdir()* }  FileInputStream **fis** = **new** FileInputStream(src); FileOutputStream **fos** = **new** FileOutputStream(tar+**"\\"**+cpname);  byte[] **b** = **new** byte[1024]; int **len** ; **len** = **fis**.read(**b**); *//这两种方式while都行 /\*while(len != -1) {  fos.write(b);  len = fis.read(b);  }\*/* **while**(**len** != -1) {  **fos**.write(**b**,0,**len**);  **len** = **fis**.read(**b**); } **fis**.close(); **fos**.close(); } } |

### node.js的并发机制

注意：

1）node.js是单进程单线程应用程序，但是通过事件和回调函数支持并发，所以性能非常高

2）node.js的每一个API都是异步的，并作为一个独立线程运行，使用异步函数调用，并处理并发

3）可以这样理解，以后在node.js中见到事件（或者eventEmitter的子类，这个子类最常见的有fs,net,http,只要是支持事件响应的核心模块都是其子类）和回调函数基本都可以认为是异步并发

## 2018年4月23日星期一

### node.js管道流

管道流提供了一个输出流到输入流的机制，通常我们用于从一个流中获取数据并将数据传递到另外一个流中。

|  |
| --- |
|  |

如上面的图示所示，我们把文件比作装水的桶，而水就是文件里面的内容，我们用一根管子（pipe即输出流需要调用这个方法从而写入输入流中）链接两个桶使得水从一个桶流入另一个桶，这样就实现了大文件的复制过程。【通常是这么写的：readStream.pipe(writeStream)】

### 链式流

链式流是通过链接输出流到另外一个流并创建多个流操作链的机制，链式流一般用于管道操作。

其实链式流更像是一种（或者说就是）链式反应一样，把一个输出流通过管道流进行相应的处理再创建多个操作流，这一系列流称为一个链式流。例如文件的解压缩。（zlib是一个压缩库）

### 如何保证复制文件不被覆盖呢？

在实例化写入流时，添加参数：**var** writeStream = fs.createWriteStream(**'3.txt'**,{**'flags'**:**'a'**});即可

|  |
| --- |
| */\*\*  \* Created by Administrator on 2018/4/23.  \*/ /\* \* 合并两个文件的内容到第三份文件中,尝试不用块级作用域修饰【读取1.txt和2.txt文件到3.txt中】 \* \*/* **var** data = **''**; **var** data1 = **''**; **var** data2 = **''**; **var** fs = require(**'fs'**);  **var** readStream3 = fs.createReadStream(**'input.txt'**); readStream3.on(**'data'**,**function**(d){  data += d; }); */\*readStream3.on('end',function(){  writeStream.writes(data,'utf8'); });\*/* readStream3.on(**'err'**,**function**(err){  **console**.log(err.stack); });  **var** readStream1 = fs.createReadStream(**'1.txt'**); readStream1.on(**'data'**,**function**(d){  data1 += d; }); */\*readStream1.on('end',function(){  writeStream.writes(data1,'utf8'); });\*/* readStream1.on(**'err'**,**function**(err){  **console**.log(err.stack); }); **console**.log(**"1.txt读取完毕"**);  **var** readStream2 = fs.createReadStream(**'2.txt'**); readStream2.on(**'data'**,**function**(d){  data2 += d; }); readStream2.on(**'end'**,**function**(){  *//怎么保证让他只写入一次而不重复写入，就是放在最后执行的地方再写入即可  writes*(data+data1+data2); }); readStream2.on(**'err'**,**function** (err) {  **console**.log(err.stack); }); **console**.log(**"2.txt读取完毕"**);  **var** *writes* = **function**() {  **var** writeStream = fs.createWriteStream(**'3.txt'**,{**'flags'**:**'a'**});  writeStream.write(data+data1+data2,**'utf8'**);  writeStream.end();  writeStream.on(**'error'**, **function** (error) {  **console**.log(error.stack);  });  writeStream.on(**'finish'**, **function** () {  **console**.log(**"写入完毕"**);  }); } |

### 模块系统

node.js中文件和模块是一一对应的，一个node.js文件就是一个模块

通过require来引入一个模块，exports是该模块公开的一个接口。模块—》接口—》方法

### 路由

完全很懵，没有理解了

### 全局对象

\_\_filename表示当前正在执行的脚本的文件名（注意是英文两个下划线） ，它将输出文件所在位置的绝对路径

\_\_dirname表示当前执行脚本所在的目录

setTimeout(cb,ms)全局函数在指定的毫秒数（ms）后执行指定函数（cb）。:setTimeout()只执行一次指定函数，返回一个代表定时器的句柄值

clearTimeout（t）全局函数用于停止一个之前通过setTimeout()创建的定时器，参数t是通过setTimeout()函数创建的定时器

如下实例

|  |
| --- |
| function printHello(){  console.log( "Hello, World!");  }  // 两秒后执行以上函数  var t = setTimeout(printHello, 2000);  // 清除定时器  clearTimeout(t);  执行后，相当于是没有了printHello()函数的调用了，就更不会有输出了 |

setInterval(cb,ms)全局函数表示在指定的毫秒数（ms）后执行指定函数（cb），可以使用clearInterval(t)函数来清楚定时器，这个方法会不停地调用目标函数（cb），直到clearInterval(t)被调用或窗口被关闭

util.inherits继承只能实现对象原型属性的继承

### 文件系统

文件系统（fs）模块中的方法均有异步和同步版本，例如读取文件内容的函数有异步的 fs.readFile() 和同步的 fs.readFileSync()

## 2018年4月24日星期二

### get/post请求

### 工具模块

工具模块

os模块

path模块

net模块

dns模块

domain模块

提供基本的系统操作函数

提供了处理和转换文件路径的工具

用于底层的网络通信。提供了服务端和客户端的的操作

用于解析域名

简化异步代码的异常处理，可以捕捉处理try catch无法捕捉的

### restful风格

get ---用于获取数据

put ---用于更新数据

delete ---用于删除数据

post ---用于添加数据



## 2018年4月25日星期三

编写包装扫码工具

svg标签是可缩放矢量图形标签

## 2018年4月26日星期四

## 2018年4月27日星期五

### element框架

1.引入脚本

2.引入样式

3.el的标签元素，就是在原有的html基础上加上了el前缀

4.

|  |
| --- |
| <el-from ref=’form’ :module=’form’ label-width=’90px’>  <el-form-item label=’名字’ prop=’name’>  <el-input v-module=’form.name’></el-input>  </el-form-item>  </el-form>  <script>  var vm = new Vue({  el:’#...’,  data:{},  methods:{}  });  </script> |

5. :module=’’ 这个有点类似于声明，目前感觉只写在大标签中，即form，table等中

v-module=’’ 这个是将此处的参数绑定到上面的module中（值传递）

[6. @click](mailto:6.@click) <==> v-on:click

### eleme form校验

Form 组件提供了表单验证的功能，只需要通过 rules 属性传入约定的验证规则，并 Form-Item 的 prop 属性设置为需校验的字段名即可

### 固定的elem表单格式（记）

1）不带校验规则的form表单（即form中没有:rules属性，form-item中没有prop指令）

|  |
| --- |
| <!doctype **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**> </**head**> <**body**> <**script src="../public/javascripts/vue.js"**></**script**> <**script src="../public/javascripts/index.js"**></**script**> <**div id="app2"**>  <**el-form :inline="true" :model="formInline" class="demo-form-inline"**>  <**el-form-item label="审批人"**>  <**el-input v-model="formInline.user" placeholder="审批人"**></**el-input**>  </**el-form-item**>  <**el-form-item label="活动区域"**>  <**el-select v-model="formInline.region" placeholder="活动区域"**>  <**el-option label="区域一" value="shanghai"**></**el-option**>  <**el-option label="区域二" value="beijing"**></**el-option**>  </**el-select**>  </**el-form-item**>  <**el-form-item**>  <**el-button type="primary" @click="onSubmit('formInline')"**>查询</**el-button**>  <**el-button type="danger" @click="reset('formInline')"**>取消</**el-button**>  </**el-form-item**>  </**el-form**> </**div**>  <**script**>  **var *vm*** = **new** Vue({  **el**: **'#app2'**,  data(){  **return**{  **formInline**:{  **user**:**''**,  **region**:**''** }  }  },  **methods**: {  onSubmit:**function**(){  ***console***.log(**'submit'**);  },  reset:**function**(){  ***console***.log(**new** Date().toLocaleString());  }  }  }); </**script**> </**body**> </**html**>  一定要注意data（）的写法，里面一定包含一个return作为初始化 |

2）带校验规则的

|  |
| --- |
| <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>Title</**title**>  <**script src="../public/javascripts/vue.js"**></**script**>  <**script src="../public/javascripts/index.js"**></**script**> </**head**> <**body**> <**div id="app3"**>  <**el-form :model="ruleForm" :rules="rules" ref="ruleForm" label-width="100px" class="demo-ruleForm"**>  <**el-form-item label="活动名称" prop="name"**>  <**el-input v-model="ruleForm.name"**></**el-input**>  </**el-form-item**>  <**el-form-item label="活动区域" prop="region"**>  <**el-select v-model="ruleForm.region" placeholder="请选择活动区域"**>  <**el-option label="区域一" value="shanghai"**></**el-option**>  <**el-option label="区域二" value="beijing"**></**el-option**>  </**el-select**>  </**el-form-item**>  <**el-form-item label="活动时间" required**>  <**el-col :span="11"**>  <**el-form-item prop="date1"**>  <**el-date-picker type="date" placeholder="选择日期" v-model="ruleForm.date1"  style="width**: 100%;**"**></**el-date-picker**>  </**el-form-item**>  </**el-col**>  <**el-col class="line" :span="2"**>-</**el-col**>  <**el-col :span="11"**>  <**el-form-item prop="date2"**>  <**el-time-picker type="fixed-time" placeholder="选择时间" v-model="ruleForm.date2"  style="width**: 100%;**"**></**el-time-picker**>  </**el-form-item**>  </**el-col**>  </**el-form-item**>  <**el-form-item label="即时配送" prop="delivery"**>  <**el-switch v-model="ruleForm.delivery"**></**el-switch**>  </**el-form-item**>  <**el-form-item label="活动性质" prop="type"**>  <**el-checkbox-group v-model="ruleForm.type"**>  <**el-checkbox label="美食/餐厅线上活动" name="type"**></**el-checkbox**>  <**el-checkbox label="地推活动" name="type"**></**el-checkbox**>  <**el-checkbox label="线下主题活动" name="type"**></**el-checkbox**>  <**el-checkbox label="单纯品牌曝光" name="type"**></**el-checkbox**>  </**el-checkbox-group**>  </**el-form-item**>  <**el-form-item label="特殊资源" prop="resource"**>  <**el-radio-group v-model="ruleForm.resource"**>  <**el-radio label="线上品牌商赞助"**></**el-radio**>  <**el-radio label="线下场地免费"**></**el-radio**>  </**el-radio-group**>  </**el-form-item**>  <**el-form-item label="活动形式" prop="desc"**>  <**el-input type="textarea" v-model="ruleForm.desc"**></**el-input**>  </**el-form-item**>  <**el-form-item**>  <**el-button type="primary" @click="submitForm('ruleForm')"**>立即创建</**el-button**>  <**el-button @click="resetForm('ruleForm')"**>重置</**el-button**>  </**el-form-item**>  </**el-form**> </**div**> <**script**>  **var *vm*** = **new** Vue( {  **el**: **'#app3'**,  data() {  **return** {  **ruleForm**: {  **name**: **''**,  **region**: **''**,  **date1**: **''**,  **date2**: **''**,  **delivery**: **false**,  **type**: [],  **resource**: **''**,  **desc**: **''** },  **rules**: {  **name**: [  {**required**: **true**, **message**: **'请输入活动名称'**, **trigger**: **'blur'**},  {**min**: 3, **max**: 5, **message**: **'长度在 3 到 5 个字符'**, **trigger**: **'blur'**}  ],  **region**: [  {**required**: **true**, **message**: **'请选择活动区域'**, **trigger**: **'change'**}  ],  **date1**: [  {**type**: **'date'**, **required**: **true**, **message**: **'请选择日期'**, **trigger**: **'change'**}  ],  **date2**: [  {**type**: **'date'**, **required**: **true**, **message**: **'请选择时间'**, **trigger**: **'change'**}  ],  **type**: [  {**type**: **'array'**, **required**: **true**, **message**: **'请至少选择一个活动性质'**, **trigger**: **'change'**}  ],  **resource**: [  {**required**: **true**, **message**: **'请选择活动资源'**, **trigger**: **'change'**}  ],  **desc**: [  {**required**: **true**, **message**: **'请填写活动形式'**, **trigger**: **'blur'**}  ]  }  };  },  **methods**: {  submitForm(formName) {  **this**.$refs[formName].validate((valid) => {  **if** (valid) {  alert(**'submit!'**);  } **else** {  ***console***.log(**'error submit!!'**);  **return false**;  }  });  },  resetForm(formName) {  **this**.$refs[formName].resetFields();  }  }  }); </**script**> </**body**> </**html**> |

## 2018年4月28日星期六

### 调试装包扫码桌面程序

1）解决了页面加载完成后，显示登录页面

注意prop属性目前只能在form-item上，注意理解代码含义

## 2018年4月30日星期一

### @RequestMapping和@ResponseBody

在方法上若只添加@RequestMapping返回值会解析为跳转路径（ModelAndView解析器，这个注解会将http请求直接映射到被controller标注的方法上）

再加上@ResponseBody后返回结果不会仅仅只被解析为跳转路径，有两个作用：1）使用该注解会读取request请求的body部分数据，使用系统默认配置的HttpMessageConverter进行解析，然后把相应的数据绑定到要返回的对象上；2）再把HttpMessageConverter返回的对象数据绑定到controller中方法的参数上

## 2018年5月2日星期三

开发包装扫码工具

## 2018年5月3日星期四

### 继续开发包装扫码工具

在Vue中，this的指向经常发生改变，所以，一定要记得让这样写 \_self=this;用\_self来代替this，这样不会发生跨域的影响

### unshift()

.unshift（）方法可向数组的开头添加一个或更多元素，并返回新的长度

.unshift（）方法不创建新的数组，而是直接修改原有的数组，所以该方法会改变数组的长度

.如果要把一个或多个元素添加到数组的尾部，请使用push（）方法

### html5（注意下面所述的都是input所属的属性）

|  |
| --- |
| HTML5 增加了多个新的输入类型：   * color * date * datetime * datetime-local * email * month * number * range * search * tel * time * url * week |

html新增了许多输入类型内容，如时间等，限制数字输入等，很方便

maxlegth属性规定输入字段允许的最大长度（字符位数）

### form属性

form属性，而不是form表单，该属性规定<input>元素所属的一个或多个表单

tips:如需引用一个以上的表单，请使用空格分隔的表单id列表，输入字段位于html表单之外（但仍属于表单）

实例：

|  |
| --- |
| <form action="action\_page.php" id="form1">  First name: <input type="text"name="fname"><br>  <input type="submit" value="Submit">  </form>  Last name: <input type="text" name="lname"form="form1"> |

### formaction属性

formaction属性规定当提交表单时处理该输入控件的文件的url

formaction睡醒覆盖<form>元素的action属性（即点击后，处理路径不再是action中的了）

formaction属性适用于type=“submit”以及type=“image”

示例如下：

<form action="action\_page.php">

First name: <input type="text" name="fname"><br> Last name: <input type="text" name="lname"><br>

<input type="submit" value="Submit"><br>

<input type="submit" formaction="demo\_admin.asp" value="Submit as admin">

</form>

当点击Submit as admin时，跳转页面是demo\_admin.asp了，

只有点击Submit时，才会跳转到action\_page.php（交给form的action处理）

### formtarget属性

input是使用这个属性之后，会覆盖掉form表单中的target属性，会规定重新的目标页面打开的方式（新开窗口等）

### placeholder属性

placeholder属性是用来描述一个输入框的预期值的提示，该提示在输入框是空的时候，自动显示

它适用于以下输入类型:text、search、url、tel、email以及password

如: <input type="text" name="fname" placeholder="First name">

### pattern属性

pattern属性规定用来检查<input/>元素值的正则表达式

如：<input type="text" name="country\_code" pattern="[A-Za-z]{3}" title="Three letter country code">

### required属性

required属性是布尔值，如果设置，则规定在提交表单之前必须填写输入字段

required属性适用于以下输入类型:text、search、url、tel、email、password、date pickers、number、checkbox、radio、file

Username: <input type="text" name="usrname" required>

### 音视频标签

<!DOCTYPE html>

<html>

<body>

<video width="420" controls>

<source src="../resources/videos/movie.mp4" type="video/mp4">

<source src="../resources/videos/msg.wav" type="video/ogg">

Your browser does not support the video tag.

</video>

</body>

</html>

这个video标签运行后，效果如下：可以用来播放视频和音频等资源，这个标签还提供下载功能，个人觉得必须掌握

个人感觉很厉害的一个标签

### html5新增的元素

增加了一些特别有趣的新特性：

新的语义元素，比如<header>,<footer>,<article>,<section>

强大的图像支持，<canvas>和<svg>

强大的图像支持<video>和<audio>

强大的新API，比如用本地存储取代cookie、

## 2018年5月4日星期五

### html5块级元素

#### 向html添加新元素

html5定义了8个新的语义的html元素，都是块级元素

你可以通过浏览器trick向html添加任何新元素

本例向html添加了一个名为<myHero>的新元素，并为其定义display样式

<!DOCTYPE html>

<html>

<head>

<title>Creating an HTML Element</title>

<script>document.createElement("myHero")</script>

<style>

myHero {

display: block;//这个表明是块级元素

background-color: #ddd;//ddd是灰颜色

padding: 50px;

font-size: 30px;

}

</style>

</head>

<body>

<h1>My First Heading</h1>

<p>My first paragraph.</p>

<myHero>My First Hero</myHero>

</body>

</html>

#### <nav>元素

这个元素是h5新增的，用来将具有导航性质的链接划分在一起，nav元素是一个可以用来作为页面导航的链接组；其中的导航元素链接到其他页面或当前页面的其他部分，并不是所有的链接组都要被放进<nav>元素；例如，在页脚中通常会有一组链接，包括服务条款，版权声明等，这时使用<footer>元素是最恰当的，而不需要<nav>元素

#### <style>样式书写格式

class的样式直接是以 ‘.’+‘class名’开头，然后在{}中写样式

元素样式直接以元素名开头

具体看下面这个完整的html例子：

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="UTF-8">  <title>HTML5 Skeleton</title>  <style>  /\*solid是实线的意思;1px是宽度;0.8em是一个相对当前浏览器字体的大小;如果用了px就不能用浏览器的自动放大字体了;  inline表示显示方式为内联式，元素前后没有换行符;margin是外边距\*/  body{font-family: verdana,sans-serif;font-size: 0.8em;}  header,nav,section,article,footer{border: 1px solid grey;margin: 5px;padding: 8px;}  nav ul li{display: inline;margin: 5px;}  </style>    </head>  <body>  <header>HTML5 Skeleton</header>    <nav>  <ul>  <li><a href="01.html">01.html</a></li>  <li><a href="02.html">02.html</a></li>  <li><a href="03.html">03.html</a></li>  </ul>  </nav>    <section>  <h1>famous cities</h1>  <article>  <h2>london</h2>  <p>London is the capital city of England. It is the most populous city in the United Kingdom,  with a metropolitan area of over 13 million inhabitants.</p>  </article>    <article>  <h2>paris</h2>  <p>>Paris is the capital and most populous city of France.</p>  </article>    <article>  <h2>tokyo</h2>  <p>Tokyo is the capital of Japan, the center of the Greater Tokyo Area,  and the most populous metropolitan area in the world.</p>  </article>    </section>    <footer>  <p>© 2014 w3School. All rights reserved.</p>  </footer>    </body>  </html> |

## 2018年5月7日星期一

### 数组清空

数组的清空是清空为一个空的数组即：[]

因此只要将其设置成 this.table = [];即可

## 2018年5月8日星期二

### 滚动条

|  |
| --- |
| <div id="app1">  <span>演示滚动条始终定位到底端</span>  //overflow属性：当内容溢出当前区域时解决的方案  <div id="app2" style="overflow：auto;width: 400px;height: 200px;background-color: darkgray;"> </div>  <input type="button" value="点击插入一行文字" onclick="add()" />  </div>    <script>  function add(){  var div = document.getElementById('app2');  //注意，div中的html在innerHtml元素中，每次都是在先前的innerHtml基础上追加  div.innerHTML = div.innerHTML + 'time= '+new Date().toLocaleString()+'<br/>';  div.scrollTop = div.scrollHeight;  };  </script> |

### scrollTop

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="UTF-8">  <title></title>  </head>  <body>  <div style="width:200px;height:200px;background-color:#999999;overflow:auto;" id="out">  <div style="width:100px;height:300px;background-color:#FFFF00;" id="inner">  这些文字显示在内层元素中。  </div>  </div>  <p>scrollTop值是：<span id="show"></span></p>  <script type="text/javascript">  var out = document.getElementById("out");  var show\_ele = document.getElementById("show");  out.onscroll = function(){  show\_ele.innerHTML = out.scrollTop;  };  //使用匿名函数或是声明函数，若是声明函数需要这样调用  /\*\*  \* out.onscroll = readVal;  \* function readVal(){  \* show\_ele.innerHTML = out.scrollTop;  \* }  \*/  //注册onscroll事件处理函数。当拖动滚动条时，会产生onscroll事件  //onscroll事件的处理函数  //页面加载完成后，执行一次此函数。显示最初的scrollTop值，此时的值为0  </script>  </body>  </html> |

## 2018年5月9日星期三

### 添加或删除节点(html)

|  |
| --- |
| <body>  <!--  首先明白：若是向html dom中添加新元素，你必须首先创建该元素(元素节点)，然后再向一个已存在的元素追加该元素  -->  <div id="div1">  <p id="p1">111</p>  <p id="p2">222</p>  <p id="p3">333</p>  </div>    <!--<script>  //操作步骤：1.先创建一个新的元素  var cp = document.createElement('p');  //2.然后由domcument创建一个文本节点(如果需要在这个元素中添加文本的话,这一步就是必须的)  var cpNode = document.createTextNode('这是一个文本的节点');  //3.然后向你创建的元素追加这个文本节点  cp.appendChild(cpNode);    //4.最后，必须向一个已有的元素追加这个新元素  document.getElementById('p2').appendChild(cp);  </script>-->  <script>  //删除节点的话，只能从一个 父节点中删除子节点  document.getElementById('div1').removeChild(document.getElementById('p1'));  </script>  </body> |

## 2018年5月10日星期四

### git理解

总的来说是先在本地建立一个本地仓库，再将本地仓库中的内容推送到远端服务器上

1.进入你想要进的一个磁盘目录下比如c:/work

2.然后在这个work目录下，建立一个images文件夹用来存放你的目标图像

3.然后打开git bash，进入到work目录下，初始化(git init)这个目录，这样就在work下面创建了一个本地仓库

4.将你的目标文件夹（images）添加入本地仓库中:git add . ,然后再从缓存库中提交到本地仓库中：git commit –m “”

5.然后将你的本地仓库与远端服务器关联起来，操作如下:①登录github，复制你的目标仓库地址；②执行指令：git remote add origin <https://github.com/shiro-liang-yi/uDantech.git>

6.代码合并：git pull –rebase origin master 【注意：pull = fetch + merge】

7.执行上面代码后，可以看到在本地多了一个README.md的文件

8.最后一步，上传本地仓库到远端服务器: git push –u origin master

【需要注意的是git不能管理空的文件夹，必须有文件才能管理】

设置git的url:

git remote set-url --push origin 目标url

git remote set-url –fetch origin 目标url

查看本地仓库的关联远程地址: git remote –v

## 2018年5月11日星期五

### nwjs应用打包

## 2018年5月12日星期六

继续packageScan

## 2018年5月14日星期一

## 2018年5月15日星期二

## 2018年5月22日星期二

## 2018年5月23日星期三

在centos上搭建了一个ftp服务（账号：ftpuser，密码：123），要求能上传、下载、覆盖文件

## 2018年5月24日星期四



在这个上面安装jdk、tomcat、elasticsearch

## 2018年5月25日星期五

### Mysql服务在centos上并且只提供内网地址访问，centos提供内网和公网，

如何通过本地的sql客户端工具去访问呢？

1. 如下操作即可：





## 2018年5月26日星期六

在linux上导入sql语句，所有的linux都可以用xshell工具登陆，然后点击新建传输工具，如果有xftp的话，就会自动打开其

以mysql –u root –p –h 172.17.0.4进入mysql，创建或者切换到数据库，使用source 来加载sql脚本文件

### Mysl在linux端的常用命令

1、MySQL常用命令

create database name; 创建数据库

use databasename; 选择数据库

drop database name 直接删除数据库，不提醒

show tables; 显示表

describe tablename; 表的详细描述

select 中加上distinct去除重复字段

mysqladmin drop databasename 删除数据库前，有提示。

显示当前mysql版本和当前日期

select version(),current\_date;

2、修改mysql中root的密码：

shell>mysql -u root -p

mysql> update user set password=password(”xueok654123″) where user=’root’;

mysql> flush privileges //刷新数据库

mysql>use dbname； 打开数据库：

mysql>show databases; 显示所有数据库

mysql>show tables; 显示数据库mysql中所有的表：先use mysql；然后

mysql>describe user; 显示表mysql数据库中user表的列信息）；

3、grant

创建一个可以从任何地方连接服务器的一个完全的超级用户，但是必须使用一个口令something做这个

mysql> grant all privileges on \*.\* to user@localhost identified by ’something’ with

增加新用户

格式：grant select on 数据库.\* to 用户名@登录主机 identified by “密码”

GRANT ALL PRIVILEGES ON \*.\* TO monty@localhost IDENTIFIED BY ’something’ WITH GRANT OPTION;

GRANT ALL PRIVILEGES ON \*.\* TO monty@”%” IDENTIFIED BY ’something’ WITH GRANT OPTION;

删除授权：

mysql> revoke all privileges on \*.\* from root@”%”;

mysql> delete from user where user=”root” and host=”%”;

mysql> flush privileges;

创建一个用户custom在特定客户端it363.com登录，可访问特定数据库fangchandb

mysql >grant select, insert, update, delete, create,drop on fangchandb.\* to custom@ it363.com identified by ‘ passwd’

重命名表:

mysql > alter table t1 rename t2;

4、mysqldump

备份数据库

shell> mysqldump -h host -u root -p dbname >dbname\_backup.sql

恢复数据库

shell> mysqladmin -h myhost -u root -p create dbname

shell> mysqldump -h host -u root -p dbname < dbname\_backup.sql

如果只想卸出建表指令，则命令如下：

shell> mysqladmin -u root -p -d databasename > a.sql

如果只想卸出插入数据的sql命令，而不需要建表命令，则命令如下：

shell> mysqladmin -u root -p -t databasename > a.sql

那么如果我只想要数据，而不想要什么sql命令时，应该如何操作呢？

　　 mysqldump -T./ phptest driver

其中，只有指定了-T参数才可以卸出纯文本文件，表示卸出数据的目录，./表示当前目录，即与mysqldump同一目录。如果不指定driver 表，则将卸出整个数据库的数据。每个表会生成两个文件，一个为.sql文件，包含建表执行。另一个为.txt文件，只包含数据，且没有sql指令。

5、可将查询存储在一个文件中并告诉mysql从文件中读取查询而不是等待键盘输入。可利用外壳程序键入重定向实用程序来完成这项工作。

例如，如果在文件my\_file.sql 中存放有查

询，可如下执行这些查询：

例如，如果您想将建表语句提前写在sql.txt中:

mysql > mysql -h myhost -u root -p database < sql.txt

具体更加详细的内容请参考这篇文章：<https://blog.csdn.net/ithomer/article/details/5131863>

## 2018年5月28日星期一

### Maven理解

#### pom.xml

Maven项目最核心的文件就是pom.xml文件，pom即project-object-model

在pom.xml文件中，除了文档开头的xml文档声明外，其实最大的节点就是project，它是一级节点

下面依次有<modelVersion>，对于maven2和maven3来说，这里的版本就是4.0.0，这个版本指的是maven的版本

<groupId>是机构或者组织的域名逆序

<artifactId>是组织机构下面项目的唯一标识符，通常一个组织下面会有许多的项目

<packaging>是项目的打包方式，可选值有jar、war、pom等等

<version>是这个项目的版本号，snapshot是快照版本，表明还处于开发中

<name>项目的名称

<url>项目的地址

剩下的就是<dependencies>依赖了

当然还有其他的节点，视情况而定，有时候并不需要

比如

|  |
| --- |
| <!—这个是配置代理服务器 -🡪  <proxies>   <proxy>   <!—代理的唯一定义符，用来区分不同的代理元素-->   <id>myproxy</id>   <!--该代理是否是激活的那个。true则激活代理。当我们声明了一组代理，而某个时候只需要激 活一个代理的时候，该元素就可以派上用处。 -->       <active>true</active>       <!--代理的协议。 协议://主机名:端口，分隔成离散的元素以方便配置。 -->       <protocol>http</protocol>       <!--代理的主机名。协议://主机名:端口，分隔成离散的元素以方便配置。 -->       <host>proxy.somewhere.com</host>       <!--代理的端口。协议://主机名:端口，分隔成离散的元素以方便配置。 -->       <port>8080</port>       <!--代理的用户名，用户名和密码表示代理服务器认证的登录名和密码。 -->       <username>proxyuser</username>       <!--代理的密码，用户名和密码表示代理服务器认证的登录名和密码。 -->       <password>somepassword</password>       <!--不该被代理的主机名列表。该列表的分隔符由代理服务器指定；例子中使用了竖线分隔符，使用逗号分隔也很常见。 -->       <nonProxyHosts>\*.google.com|ibiblio.org</nonProxyHosts>  </proxy>  </proxies> |

## 2018年5月29日星期二

### Idea中import projects 和open的区别

import projects可以导入非idea环境下的工程，比如eclipse、maven工程

而 open打开的直接是idea工程

### Maven工程plugins 和 dependencis报错

解决方案：单独在项目所在的目录下，运行cmd，手动通过maven构建：maven clean

Maven install –DskipTests

当然前提是maven的环境变量配置了

在maven的settings.xml中配置了<localRepository>E:\maven-respository</localRepository>

这样，所有的依赖jar等都下载到了本地仓库中

看是否能够构建成功

然后再用idea打开即可

### ssh端口转发

ssh端口转发的含义：比如远程有一个服务端机器，本地有一个客户端，原先即是本地客户端直接与远程服务端通过特定的端口进行通讯

而现在是：两者之间不直接进行通讯，本地客户端的请求经由ssh客户端的特定端口发出，然后ssh客户端与ssh远程服务端进行响应，ssh服务端再与服务端通信，这样整体上感觉是在二者之间建立了一个隧道通信。

因此ssh端口转发可以实现一些防火墙的端口限制访问等

具体实现例子：ssh -4 –L 1026:172.17.0.4:3306 [root@118.25.64.20](mailto:root@118.25.64.20)

含义如下： -4 表示这里是ipv4协议

-L是参数

1026：是本地端口（即你想要转发的目的端口，一般取1024~65535之间的端口）

172.17.0.4：是远程主机

3306：是远程主机端口，即你想转发的目标端口

root@:即以root身份登陆，在ssh协议通信中，以某某身份登陆就是某某@主机

118.25.64.20：是ssh的主机名

整体格式：ssh -L <local port>:<remote host>:<remote port> <SSH hostname>

【ssh是一种通信协议，而这种通信协议最大的一个特点就是传输安全，或者说通信安全，因此就涉及到安全密钥的问题，即公钥、私钥；而在与服务器端的通信中，我们一般都是采用ssh协议】

### Maven工程运行

推荐这种方式【手动构建】：

在当前项目的目录下，打开cmd，运行mvn clean；mvn install –DskipTests 这样打包成一个war包了

然后切换回idea中直接运行即可

Jetty服务器默认主机就是localhost:8080

## 2018年5月30日星期三

### Mysql自增长理解

如果是刚开始建表时，设置auto\_increment=100;那么，自增长的起始值就是100

如果是已有表，我们查询该表的信息时，显示出来的auto\_increment是此时的主键字段索引的指向位置，并不是在创建表的时候设置的auto\_increment的值。

### 数据库或表的创建

1. 表

create table if not exists tablename();

1. 数据库

create database if not exists database1;

1. 从模板表创建表

create table if not exists tt like cyx;

【注意，这里是exists，不是exits，谨记】

1. 如果存在删除已有

drop table if exists user;

### mysql limit

limit子句用于强制select语句返回指定的记录数，limit接受一个或两个数字参数，参数必须是一个整数：第一个参数指定第一条返回记录行的偏移量；第二个参数指定返回记录行的最大数目（因为有可能数据量没有这么多），初始记录行的偏移量是0，

例子： select \* from user limit 5,10;

这条语句的执行结果是，返回10条数据（如果有这么多的数据的话），第一条的行偏移量是5，即返回的第一条数据的下标应该是从6开始，到下标15，总共10条数据

## 2018年5月31日星期四

### nexus私服的搭建

1. 到官网下载包含Jetty容器的bundle包，下载地址：<https://www.sonatype.com/download-oss-sonatype>
2. 解压安装包：会得到两个包（以nexus-2.14.8-01-bundle.zip为例）：nexus-2.14.8-01、sonatype-work
3. 进入D:\uDan\_software\_computer\nexus-2.14.8-bundle\nexus-2.14.8-01\bin\jsw\windows-x86-64（这里是我的解压目录，视具体情况而定），点击install-nexus.bat安装成为windows服务，此时可能会报错：wrapper | OpenSCManager failed - 拒绝访问。 (0x5)；这是由于权限不够的原因造成的，只需要退出，然后以管理员的身份去运行cmd，再进入目标目录执行即可
4. 然后访问<http://loclahost:8081/nexus/>,这就是私服 **（**如果访问不了，可能是nexus服务没启动，启动即可**）**
5. 进入后会看到这个界面：
6. 点击右上角的login in ，默认账号admin，密码admin123
7. 点击左边的Respository，可以看到一系列仓库类型，我们一般使用3个，分别是Releases，Snapshots和3rd party，这3个仓库分别是装稳定版、开发版和第三方jar包，截图如下：
8. 举例，手动上传一个jar包到Releases仓库里里面：



9）上面是手动方式，另外一种是利用mvn直接发布到私服上面来，需要修改2个地方，第一是在settings.xml文件的servers节点下加入，这是配置私服的密码和仓库，配合pom文件中的仓库地址就构成了完整的访问私服的要素，账号和密码之所以在settings.xml中设置，是由于settings.xml文件时本地的，而pom.xml文件是公共的，不安全，所以放在settings.xml中

|  |
| --- |
| <server>  <id>nexus-releases</id>  <username>admin</username>  <password>admin123</password>  </server>  <server>  <id>nexus-snapshots</id>  <username>admin</username>  <password>admin123</password>  </server> |

第二是在pom.xml 中增加，下面的id和settings.xml中的id必须要一样

|  |
| --- |
| <distributionManagement>  <repository>  <id>nexus-release</id>  <name>Nexus Release Repository</name>  <url>http://localhost:8081/nexus/content/repositories/release/</url>  </repository>  <snapshotRepository>  <id>nexus-snapshots</id>  <name>Nexus Snapshot Repository</name>  <url>http://localhost:8081/nexus/content/repositories/snapshots/</url>  </snapshotRepository>  </distributionManagement> |

10）那么上面的私服的仓库的url从什么地方获取呢？如下图：

11）新建一个mvn项目，

1、运行source：jar在target下生成源码jar包；

2、运行deploy，class的jar包发布到私服；

3、运行source：jar deploy，连同源码一起发布到私服

12）完成，在其他项目中可以使用上面的jar包了（具体详情请参考：<https://www.cnblogs.com/dreamroute/p/5440419.html>这篇文章）

补充：我们可以使用proxy代理仓库，来代理公共的远程仓库【这个参考<https://www.cnblogs.com/luotaoyeah/p/3791966.html>文章】

### Druid

Druid首先是一个数据库连接池，Druid是目前最好的数据库连接池（有待验证），类似于dbcp、c3p0等，

但是它不仅仅是一个数据库连接池，它包含四个部分：

Druid是一个JDBC组件，包含三个部分：1）基于Filter – Chain模式的插件体系

2）DruidDataSource高效可管理的数据库连接池

3）SQLParser

### fluentd(td-agent)日志收集系统

fluentd是一个日志收集系统，它的特点在于其各部分均是可定制化的，你可以通过简单的配置，将日志收集到不同的地方

1. 下载压缩包：<https://td-agent-package-browser.herokuapp.com/3/redhat/7/x86_64>
2. 将下载下来的td-agent-3.1.1-0.el7.x86\_64.rpm包放在：D:\uDan\_software\_computer\linux\_software下
3. 通过xshell登陆服务器，点击新建文件传输，自动打开xftp（如果已安装了的话，没有请先安装），
4. 在 /usr/local 下新建文件夹 fluented，将rpm包传输到这里
5. 解压缩：因为是rpm格式的，因此需要转化为cpio格式解压，使用 rpm2cpio xxx.rpm | cpio –div 这条命令即可，这条命令包含转换成cpio格式的，并且解压缩了
6. 然后进入解压得到的包的 /usr/local/fluented/opt/td-agent/etc/（这里以我本次安装为例），运行：td-agent start
7. 运行： [root@VM\_0\_9\_centos etc]# ./init.d/td-agent start
   * 1. [root@VM\_0\_9\_centos etc]# ./init.d/td-agent stop
     2. [root@VM\_0\_9\_centos etc]# ./init.d/td-agent restart

### rpm方式安装

直接下载rpm包，然后运行 rpm -ivh --force td-agent-3.1.1-0.el7.x86\_64.rpm(这个是文件名)即可

### ping、telnet、curl的关系

ping：测试远端主机是否来连通？ 例如： ping 118.25.64.20 【注意，它是测试主机连通性的，没有端口哟】

telnet：实现对远程主机的操控，既然是操控，那就一定有进入的端口喽，例如 telnet 118.25.64.20 8080【注意，主机名和端口之间没有冒号哟，记不住的话，就输入telnet –h好喽】

curl：它是将远程主机的内容下载到本地进行浏览，并不能对远程主机直接进行操控，因此它也是有端口的哦

## 2018年6月4日星期一

## 2018年6月5日星期二

### 这两天git上传遇到的问题解决

#### Git lfs

1）执行

|  |
| --- |
| $ git push origin master  Counting objects: 4, done.  Delta compression using up to 4 threads.  Compressing objects: 100% (4/4), done.  Writing objects: 100% (4/4), 5.36 MiB | 29.85 MiB/s, done.  Total 4 (delta 2), reused 0 (delta 0) |

到这一步，后面一直在等待，好久都没有反应，好长时间后可能会报错、也有可能会提交成功，怎么解决呢？？？

如果后续报错提示 ‘errno 10054’，那么你需要设置http上传缓冲区大小了： git config –global http.postBuffer 1000M\*1024\*1024 【注意这里的数值自己拿计算器算一下，再填入那个数字后就ok了】

1. 如果上面提交成功了，但是你嫌时间实在是太长了，怎么解决？？？

可能有两方面原因：① github本身访问不太稳定，可以理解，毕竟服务器在国外嘛

② 你所上传的文件太大了

这里我们着重解决的时第二种情况：

用 git lfs 命令 【Large File Storage】

具体怎么用呢？

假设我现在需要提交一个zip【名字是：Free Download Manager.zip】的压缩文件，其大小是58.3M

|  |
| --- |
| $ git lfs track "Free Download Manager.zip" |

先用这个命令去跟踪这个文件，后面的git add . git commit等操作正常就行了

## 2018年6月6日星期三

### WebSocket

1. 是一种基于tcp的应用层传输协议
2. 一次握手，服务器主动推送信息到客户端，真正意义上的实时更新
3. 传统的实现实时推送信息的方式一般有两种，一种是ajax轮询，一种是流，但是这两种都是模拟的实时推送，会有许多网络，效率，阻塞等问题。

## 2018年6月7日星期四（请假半天，13：54来的）

### 线程

线程时程序执行流的最小单元，一个标准的线程由线程Id，当前指令指针(pc)，寄存器集合和堆栈组成；线程是进程中的一个实体（说到线程就不得不提进程），其状态由就绪、阻塞、运行三种。

线程是程序中一个单一的顺序控制流程，进程内有一个相对独立的、可调度的执行单元，是系统独立调度和分派CPU的基本单位指令运行时的程序的调度单位；在单个程序中同时运行多个线程完成不同的工作，称为多线程

### 死锁

首先呢，想一个实际问题：某人去银行用银行卡取钱（原卡内共有5000￥），他想取出2000，可是就在作人员给他操作时，他的老婆拿着该银行卡存折也去取钱了，也想取2000，并且也在操作中，这时问题出现了，他俩都操作成功了，并且银行卡最后还有3000￥的余额，这显然是不对的啦，怎么解决呢？？？

我们想，应该在某一方操作这张卡的时候呢，给它加一把锁，其他的人必须等待其操作完才可以继续操作（这里，其他人与目标用户都相当于一个线程）

那么怎么加锁呢？加什么锁呢？其实就是使用Synchronized同步代码块或者同步方法，使用Synchronized的就是同步锁。只不过可以使用在代码块或者方法中，同步锁（即Synchronized修饰的）其实是一种保护锁，同步锁会确保在某个时刻内只有一个线程被允许执行特定的代码块）

但是Synchronized使用的不恰当的话，就出现问题了，什么呢？？？死锁喽！！！怎么出现的呢？？？Synchronized嵌套啦，即Synchronized修饰的方法中还有Synchronized修饰的方法，这时就出现了两个线程都把各自的资源锁住了（只要调用了Synchronized修饰的方法或代码块，那么这个线程就会给这个对象加一把锁），并且都在等待别人释放自己所需要的那部分资源，如此下去，死锁产生了。

怎么解决死锁或者说避免呢？？？一个比较好的思路是给线程设置优先级，或者释放所有的锁，回退操作

其实有同步锁（Synchronized）就会有异步锁（asynchronized），而同步的概念就是共享某个资源，异步的概念就是独立

总结来说：就是一个线程如果是访问A对象的带有Synchronized修饰的方法的话，那么其他线程如果在这时也访问该对象的这个带有Synchronized的方法的话，那么就必须要等待，直至前一个释放资源

### maven项目目录结构



## 2018年6月8日星期五

### nutz学习

它也是一个web方面的框架，不同于ssh与ssm的框架

它其实是与hibernate比较类似的，具体的后续实践过再总结（个人现在也是比较懵圈）

### @At

这个其实有点类似于springMvc中的requestMapping（纯属个人的理解）

### Dao

nutz框架中的Dao类，是对数据库操作的所有封装，我们只需要调用这个类就行了

## 2018年6月9日星期六

### web项目启动后怎么知道访问路径

这个问题的关键是要知道两个问题：

1. 访问的端口

* -- 如果是采用的tomcat服务器，那么在配置tomcat的时候会让你选择端口，填入你选的即可
* -- 如果是采用的jetty的话，那么其默认的端口也是8080，可以到该工程项目的pom文件中去寻找

1. 本项目在服务器端的部署路径

* 以jetty为例，在pom文件中，如果配置了如下：
* 

<contextPath/>个人理解是将该工程部署到webapps下面，并用 ‘/’代替了部署的工程的名；即你只需要启动jetty后在8080端口后面加上 ‘/’即表示你已经进入该公工程项目了，你就可以访问该工程项目的web-inf下面的任意页面了

* 关于tomcat的部署路径后续再说？？？

### 使用jetty作服务器的配置

使用jetty作服务器的话，首先要配置，它的配置不像tomcat那样，而是直接点击 Edit Configuration，然后点击 ‘+’号，选择maven，然后在Command line项填入jetty即可出现下拉选项卡选中即可。

## 2018年6月11日星期一

继续nutz的学习

### @At

标注有这个注解的类，表示这是一个带有入口方法的类；同时只有带有@At注解的类才是模块类

### @Ok

这个应该是指定方法的返回值类型，总之是跟返回值、交互值有关联的注解内容

这里如果是@Ok("json:full")

表明是以json视图交互的，full是JsonFormat的一中内置格式的缩写

full – 不忽略空值，换行，key带双引号，新版jquery兼容

compact -- 不换行，忽略空值，key不带双引号，新版jquery不兼容

默认 -- 忽略空值，换行，key不带双引号，新版jquery不兼容

### @Fail

见名知意，这是入口方法或适配器抛出异常才会执行的配置

@Fail(“jsp:jsp.500”)

含义就是 内部重定向到 /WEB-INF/jsp/500.jsp页面

打开web.xml，加入如下配置

<error-page>

<error-code>500</error-code>

<location>/WEB-INF/jsp/500.jsp</location>

</error-page>

然后在WebContent/WEB-INF/jsp/ 下新疆爱你一个文件叫 500.jsp 写入内容即可

## 2018年6月12日星期二

### Nutz验证码

获取验证码的方法一般返回值类型是BufferedImage，因为其返回的是一个图像

一般都用第三方jar包：Captcha

## 2018年6月13日星期三

### nutz继续学习

#### @Param(“..”)

两个点号是按对象属性一一设置

### Springboot学习

#### starters

Spring boot 为我们提供了简化企业级开发绝大多数场景的starter pom（启动器），只要引入了相应场景的 starter pom，相关技术的绝大部分配置将会消除（自动配置），从而简化我们开发；业务中我们就会使用到spring boot为我们自动配置的bean

spring-boot-dependencies里面定义了jar包的版本

#### 入口类和@SpringBootApplication

1. 程序从main方法开始运行
2. 使用SpringApplication.run()加载主程序类
3. 主程序类需要标注@SpringBootApplication
4. @EnableAutoConfiguration是核心注解
5. @Import导入所有的自动配置场景
6. @AutoConfigurationPackage定义默认的包扫描规则
7. 程序启动扫描加载主程序类所在的包以及下面所有 子包的组件

#### 自动配置

xxxAutoConfiguration

--Spring Boot中存在大量的这些类，这些类的作用就是帮我们进行自动配置

--它会将这个场景需要的所有组件都注册到容器中，并配置好

--他们在类路径下的META-INF/spring.factories文件中

--spring-boot-autoconfigure-1.5.9.RELEASE.jar中包含了所有的场景的自动配置类代码

--这些自动配置类是Spring Boot进行自动配置的精髓

#### Spring Boot配置

主要从以下三个方面来详细说明：配置文件、加载顺序、配置原理

--配置文件

Spring Boot使用一个全局的配置文件

。application.properties

。application.yml

配置文件放在src/main/resources目录或者类路径 /config 下

.yml 是YAML(YAML Ain’t Markup Language)语言的文件，以数据为中心，比json、xml等更适合做配置文件

全局配置文件的可以对一些默认配置值进行修改

--加载顺序

Spring boot启动会扫描以下位置的application.properties或者application.yml文件作为spring boot的默认配置文件

- file:./config/

- file:./

- classpath:/config/

- classpath:/

- 以上是按照优先级从高到低的顺序，所有位置的文件都会被加载，高优先级配置内容会覆盖低优先级配置内容

- 我们也可以通过配置spring.config.location来改变默认配置

外部配置加载顺序

Spring boot支持多种外部配置方式



--自动配置原理

可以查看HttpEncodingAutoConfiguration

通用模式

-- xxxAutoConfiguration：自动配置类

-- xxxProperties：属性配置类

-- yml/properties文件中能配置的值就来源于【属性配置类】

几个重要注解

-- @Bean

-- @Conditional

-- debug=trrue查看详细的自动配置报告



#### Spring Boot与日志

-- 日志框架

市场上存在非常多的日志框架。JUL（java.util.logging），JCL（Apache

Commons Logging），Log4j，Log4j2，Logback、SLF4j、jboss-logging等

Spring Boot在框架内容部使用JCL，spring-boot-starter-logging采用了

slf4j+logback的形式，Spring Boot也能自动适配（jul、log4j2、logback） 并

简化配置



-- 日志配置

-- 全局常规设置（格式、路径、级别）

-- 指定日志配置文件位置

-- 切换日志框架



#### Spring Boot与web开发

【从这三方面Thymeleaf、web定制、容器定制】

-- web自动配置规则

1. WebMvcAutoConfiguration
2. WebMvcProperties
3. ViewResolver自动配置
4. 静态资源自动映射
5. Formatter与Converter自动配置
6. HttpMessageConverter自动配置
7. 静态首页
8. favicon
9. 错误处理

-- Thymeleaf模板引擎介绍

Thymeleaf是一款用于渲染XML/XHTML/HTML5内容的模板引擎。类似JSP，Velocity，FreeMaker等，它也可以轻易的与Spring MVC等WEB框架进行集成作为Web应用的模板引擎，与其它模板引擎相比，Thymeleaf最大的特点是能够直接在浏览器中打开并正确显示模板页面，而不需要启动整个Web应用

Spring Boot推荐使用Thymeleaf、FreeMaker等后现代的模板引擎技术；一旦导入相关依赖，会自动配置ThymeleafAutoConfiguration、FreeMarkerAutoConfiguration

-- 整合Thymeleaf

1. 导入starter-thymeleaf
2. template文件夹下创建模板文件
3. 测试页面&取值
4. 基本配置

基本语法

表达式

-- #{...} : 国际化消息

-- ${...} ：变量取值

-- \*{...} ：当前对象/变量取值

-- @{...} : url表达式

-- ~{...} ： 片段引用

-- 内置对象/共用对象

判断/遍历

-- th:if

-- th:unless

-- th:each

-- th:switch、th:case

th:属性

### Docker

Docker是一个开源的应用容器引擎，基于Go语言并遵从Apache2.0协议开源

Docker可以让开发者打包他们的应用以及依赖到一个轻量级、可移植的容器中，然后发布到任何流行的Linux机器上，也可以实现虚拟化

Docker支持将软件编译成一个镜像，然后在这个镜像中各种软件做好配置，将镜像发布出去，其他使用者可以直接使用这个镜像，运行中的这个镜像称为容器，容器启动是非常快速的，类似windows里面的ghost操作系统，安装好后什么都有了

### Spring Data

#### Spring Data简介

Spring Data项目的目的是为了简化构建基于Spring框架应用的数据访问技术，包括非关系数据库、Map-Reduce框架、云数据服务等等；另外也包含对关系数据库的访问支持

#### Spring Data包含多个子项目：



#### Spring Data特点

Spring Data为我们提供使用统一的API来对数据访问层进行操作；这主要是Spring Data Commons项目来实现的。Spring Data Commons让我们在使用关系型或非关系型数据访问技术时都基于Spring提供的统一标准，标准包含了CRUD（创建、获取、更新、删除）、查询、排序和分页的相关操作

#### 统一的Repository接口

Repository<T,ID extends Serializable>：统一接口

RevisionRepository<T,ID extends Serializable,N extends Number & Comparable<N> >：基于乐观锁机制

CrudRepository<T,ID extends Serializable>：基于CRUD操作

PagingAndSortingRepository<T,ID extends Serializable>：基于crud及分页

#### 提供数据访问模板类 xxxTemplate

如： MongoTemplate、RedisTemplate等

#### JPA与Spring Data

1. JpaRepository基本功能

编写接口继承JpaRepository既有crud又有分页等基本功能

1. 定义符合规范的方法命名

在接口中只需要声明符合规范的方法，即拥有对应的功能



1. @Query自定义查询，定制查询SQL
2. Specifications查询（Spring Data JPA支持JPA2.0的Criteria查询）

## 2018年6月14日星期四

### nutz手册

#### Nutz Mvc

Nutz Mvc是要和一个web服务器（比如tomcat）一起工作的，它存在的意义就是要把一个标准的HTTP请求，转发到某一个Java函数中

它的特点是：

。帮你做参数的分析（当然你可以虽易扩展）

。如果你愿意，你可以不用View来渲染HTTP输出流

。 提供几个基本的渲染方式（比如JSP、JSON）

。除此以外，不内置更多的功能

每一种HTTP请求，标识就是一个URL，而每一个Java函数怎样和一个URL关联呢？？？

最直接的办法就是在函数上声明一个注解，这个注解，在Nutz Mvc中，叫@At

整个过程如下：  




声明了@At的函数被称为入口函数

任何一个请求，都会经过四道工序

A -- 过滤 你通过@Filters注解可以为你的入口函数定义任意多的过滤器

B --适配 这个过程将HTTP输入流转换成入口函数的参数数组

。默认的，它认为输入流是传统的名值对方式

。更多的适配方式请看[关于适配器](https://nutzam.github.io/nutz/)

。文件上传也是一种适配方式，请参看UploadAdaptor

C --调用 调用入口函数，你在里面需要调用相关的业务层代码

。如果你的业务比较复杂，为了解耦合，你可能需要Ioc容器的帮助

D --渲染 根据入口函数的返回，渲染HTTP Response

。如果返回是个View ，这用这个View来渲染null（null？是的，你没看错，这种情况View接口第三个参数会是null）

.你可以用org.nutz.mvc.view.ViewWrapper将你的返回对象以及要返回的视图组合在一起返回，ViewWrapper也是一个View

。否则用函数的@Ok注解声明的View来渲染入口函数的返回对象

。如果你的函数处理过程中抛出了异常，用@Fail注解声明的View来渲染异常对象

。返回值会保存在request的attr中，名字是obj

#### Nutz Mvc的module（主模块）

我们怎么把一个Nutz Mvc挂载到tomcat中呢？？？或者说（我个人理解来说）我们这个项目怎么和Nutz Mvc发生关系呢？？？

其实我们只需要在web.xml中声明一个 Filter 即可，让其来过滤

如下图所示：



我们配置了一个过滤器nutz，然后并且给出了它的映射方式（其实就是具体它的运作方式）

其实，任何一个类都可以作为主模块，只要你将其配置在web.xml中（就像上图中的MainModule），这样的设计主要是为了方便一些非JSP/Servlet标注的web服务器整合Nutz mvc框架

在主模块中，你可以声明如下的注解：

。@Modules -- 声明应用的所有子模块

。@IocBy -- 设置应用所采用的Ioc容器

。@SetupBy -- 应用启动以及关闭时的额外处理

。@Views -- 自定义的扩展视图

。@Localization -- 应用的本地化字符串设定

。 -- 自定义Session实现

。所有入口函数上支持的注解

。声明在主模块的注解，将作为所有入口函数的默认配置

。在子模块中的定义更为优先

#### 子模块

任何类都可以作为子模块，只要通过@Module注解声明到主模块上即可

就像这样：



子模块中任何函数，只要是public的，且不是static的，都可以作为入口函数；标记入口函数的方法是在其上标注注解@At

在入口函数上，你可以声明如下注解：

。@At -- 入口函数对应的URL

。@Ok -- 成功视图

。@Fail -- 失败视图

。@AdaptBy -- HTTP参数适配方式

。@Filter -- 过滤器

。@Encoding --输入输出编码

只有标注了@At注解的函数才被认为是入口函数，例如：

@At(“/my/abc”)

public void someFunc(){}

你也可以为该函数声明多个url

@At({“/my/abc”,”/my/xyz”})

public void someFunc(){}

#### URL映射

为什么需要详细描述URL映射？？？

Nutz.Mvc 的核心任务就是将 HTTP 请求的 URL 映射到某一个入口函数，如果你看完了 Nutz.Mvc 概述你大概应该知道，映射的配置信息是通过注解 @At 来设置的，@At 注解也非常简单，配置起来应该没有什么障碍。但是，依然在某些时候，你会在你的应用出现 404 错误，为了能让你在编写项目时，心里非常有底，这里将详细的解释一下JSP/Servlet 以及 Nutz.Mvc 映射部分的工作原理，在你遇到讨厌的 404 时，只要通读本文，相信就能找到问题的症结。

什么是URL？？？

任何URL都由如下部分组成

http://www.myapp.com/app/module/action.suffix

。http:// --协议，也可以是https://

。www.myapp.com -- 域名或者IP地址，由DNS服务器负责转发

。/app -- Context的path，这个匹配的server.xml中每个<context>的path属性，由HTTP服务器负责转发

。/module/action.suffix -- 从这里以后的匹配将交给相应的Context的web.xml，由HTTP服务器根据web.xml来转发

因此我们主要研究就是 /module/action.suffix的部分是如何转发的、

web.xml中的映射 -url-pattern？？？

通常，在web.xml中，我们可以声明各种HttpServlet子类，为了能让某一个子类接受URL，就需要配置映射，众所周知你可以通过<servlet-mapping>，为你的Servlet增加一组至多组的配置

|  |
| --- |
| <servlet-mapping>  <servlet-name>MyServletName</servlet-name>  <url-pattern>/\*</url-pattern>  </servlet-mapping> |

#### 适配器

首先来了解一下什么是适配器？？？

其实啊，适配器的作用就是将一个HTTP请求参数转换成一个函数参数，而执行这个过程的对象就被称为适配器了。怎么样？是不是并不像想象的那样难理解啊（那是因为老哥我付出很大的努力好嘛）

怎么使用适配器呢？？？

在每一个入口函数上，你都可以通过注解@AdaptBy来声明如何适配HTTP参数，当然你没有必要在每一个入口函数上都声明，在子模块类上声明，或者在整个应用的主模块类上声明均可。默认的，如果你什么也不写，nutz Mvc（~~此处为什么会是它而不是其它呢？因为关乎HTTP请求的这一块都是由它负责的嘛，啊哈哈哈哈~~）会采用org.nutz.mvc.adaptor.PairAdaptor（也就是名值对的方式）来适配你的HTTP参数

### 全匹配 /\*

转发到本Context的任何请求都会调用这个Servlet

如果请求为：<http://localhost:8080/testweb/abc/getlist.nut>

调用request对象不同方法将会返回的值

|  |  |
| --- | --- |
| req.getRequestURL() | “http://localhost:8080/testweb/abc/getlist.nut” |
| req.getRequestURI() | “/testweb/abc/getlist.nut” |
| req.getPathInfo() | “/abc/getlist.nut” |
| req.getServletPath() | “” |

### 搞清楚这几个注解的作用范围



### 一般乱码的解决方案

要尽量减少出现乱码，我个人认为要做到五码合一：ide（eclipse/idea）、页面（jsp/其它模板引擎）、应用服务器（tomcat等）、源码（java源码及周边文件）、数据库编码

1）将eclipse设置成utf8

打开eclipse安装目录下的eclipse.ini，在最末尾新增一行：

-Dfile.encoding=UTF-8

修改之后，重启即可

1. jsp页面编码

<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>

1. tomcat编码

打开tomcat安装目录下的 bin\setenv.bat，该文件通常不存在，新建之，添加如下内容

set JAVA\_OPTS=-Dfile.encoding=UTF-8

打开 conf\server.xml，在8080端口所述的Connector节点，添加URIEncoding，可解决大部分GET请求中文乱码的问题

URIEncoding=”UTF-8”

1. 源码的编码

通常情况下，文件本身的编码，取决于新建文件时ide或project的编码

另外一个隐藏的编码，是maven/ant编译java源文件时使用的编码

Maven的配置如下：

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

1. 数据库编码

通常来说，如果其它4码解决了，那大部分情况下是mysql会出现编码问题

Mysql有4个编码

Server characterset： utf-8 //服务器字节集

Db characterset： utf-8 //当前数据集字节集

Client characterset： utf-8 //客户端字节集

Conn characterset： utf-8 //当前连接的字节集

修改my.ini的mysqld小节，设置服务器字符集，是最佳解决方式

[mysqld]

Character-set-server = utf-8

然而，对于现有系统，全局修改的风险比较大，所以，可以在客户端解决，即jdbcurl上做配置

J

jdbc:mysql://127.0.0.1/nutzdemo?userUnicode=true&characterEncoding=UTF-8

## 2018年6月15日星期五

我们现在使用的是nutz wiki框架，这个框架底层包含了nutz、nutz mvc、nutz dao等

所以此后就是学习nutz wiki了

它整体的结构是一种按照模块来划分的，所有的直接与web交互的控制器全都放在web包下面，

然后一个功能就是一个module，更加的解耦合

目前遇到的几个问题：

1. 动作链到底是干什么的，有什么作用？？？
2. nutz mvc只是nutz wiki框架中的一部分，那么nutz呢？这个又是什么？？？怎么理解？？？
3. 动作链的配置文件是不是都是JJSON格式的文件，文件名后缀是不是只能是.js呢？？？

个人理解如上问题：

1. 动作链其实是nutz mvc（我暂且称为前端请求控制器吧）处理HTTP请求的一个方式：
   1. 书中原话如下：对于一个http请求的处理实际上是由一系列子处理构成的，比如根据映射找到入口函数、比如为入口函数生成调用参数、比如调用入口函数，我们希望这一系列子处理可以根据url的不同而不同，我们也希望子处理是可配置的，因为它能让你的http映射处理具备更大的灵活性
   2. 总的来说，用我自己的话来讲，首先动作链是不是依赖于nutz mvc呢？？？貌似是的，嗯确实是的。那么我们知道，nutz mvc是处理http请求与类，方法之间的映射关系的，那么问题来了，这些处理都是由一个个子处理构成的，我们当然希望这些子处理有比较高的可复用性或者说灵活性（匹配性上比较高），那么，我就引入了动作链这个怪东西了（原谅我，啰嗦了半天才降到关键处；但是没办法，编程就是这样，一个内容嵌套一个内容，要想讲清楚这个东东，就要清楚它的原理所在，编程一定要做到这一点，告诫同仁）
   3. 那么动作链里面到底是什么内容呢？？？首先它一定是一个xxx.js命名的文件，而且格式一定是JSON格式的，它里面会有好多实现了不同的处理器接口的实现类，这些实现类均是运用于那个http请求映射过程中，你也可以写自定义的动作链文件，当然这样会覆盖默认的动作链配置文件。在每个入口函数里，你可以通过注解@Chain来指定你要采用哪个动作链，当然如果你没有指定，那么nutz Mvc默认认为你是用@Chain（“default”）来处理这个入口函数的【特别注意一点；在Nutz Mvc加载时，它就会为每个入口函数创建URL映射，即把一个URL映射到一个动作链实例上，所以动作链的实例，是在加载时就被创建了；所以如果你自己实现了动作链工厂，请保证工厂生成的每个动作链是线程安全的】
2. 关于nutz是什么的这个问题，我想应该是比较难以回答得了。首先你要明白，nutz Mvc以及后来比较新的nutz boot等都是nutz的内容，他们都是基于nutz的一个开发方向，只是侧重点不同而已（纯属个人理解），像我们现在将要使用的nutz wiki框架一样，它是以nutz为基础的，也就是原理是一样的，但是可能更加全面一些，因为后来的东西总会对已有的内容做更好的完善并丰富之，目前而言，你只需要学习好nutz wiki框架就行了，因为它比nutz要更加实用，而且更丰富
3. 这个问题，我在第一个问题里面相信已经做了比较详细的叙述了，这里就不再赘述了，请绕道第一问

## 2018年6月19日星期二

### nutz视图

视图的任务就是将入口函数的返回值（一个java对象）渲染到HTTP响应流中

现在nutz mvc自带的主要视图有：



### Jsp视图

无论入口函数返回什么对象，只要它的Ok返回值是jsp视图类型的话，它都会将其保存到request的“obj”属性中，比如



那么，你在test.jsp页面里，通过



或者用JSTL的EL



都会直接输出字符串“this is test”

### Raw视图

视图的实现类为：org.nutz.mvc.viewRawView

该视图将数据对象直接写入HTTP响应

ContentType支持几种缩写



使用方式



将方法的返回值直接写入流，数据对象可以是如下类型：



## 2018年6月20日星期三

Nutz之Ioc容器

一个mvc框架可以通过Ioc接口同一个Ioc容器挂接，挂接的方式很简单：在主模块上声明@IocBy

那么牵涉到一个问题：怎么知道其是一个主模块呢？？？

这个就要看web.xml配置文件了

按照手册来看：只需要声明了modules参数即可，然后让其指向一个类即可

需要注意：在主模块上声明@Inject是没有意义的

通过@IocBy为整个应用声明了Ioc容器，那么如何使用呢？实际上，你的每一个模块都可以来自容器，只要你在模块上声明了@Inject

如果你生明了这个朱捷，nutz Mvc构造你的这个模块的时候，会通过ioc容器获取，而不直接调用默认的构造函数了