## 第一章 dubbo + zookper + springMVC + redis

1.dubbo监控系统搭建

（参考：http://blog.csdn.net/u013142781/article/details/50396621/）

准备工作：（以jdk1.8为例）

1. 安装包

Dubbo dubbo-2.5.3.jar (F:\semio\)

Zookeeper zookeeper-3.4.9.tar.gz (F:\semio\)

Jdk1.8 dubbo-admin-2.5.4.war (F:\semio\)

Jdk1.7 dubbo-admin-2.5.4-SNAPSHOT.war (F:\semio\)

Tomcat(安装版) apache-tomcat-7.0.67.exe (F:\semio\)

1. 安装tomcat ，把对应版本dubbo war包复制到 tomcat\webapps\dubbo-admin-2.5.4.war ，

启动tomcat，自动解压war包，生成对应的文件夹Tomcat 7.0\webapps\dubbo-admin-2.5.4，生成后就可以把tomcat关闭，在web-info下 有dubbo.properties 文件，打开可以看到root和guest用户的密码。

3.zookeeper 解压，进入F:\semio\zookeeper-3.4.9\conf 目录下，把zoo\_sample.cfg重命名为

Zoo.cfg，内容修改为：

tickTime=2000

initLimit=10

syncLimit=5

dataDir=F:\\semio\\zookeeper-3.4.9\\data

dataLogDir=F:\\semio\\zookeeper-3.4.9\\log

clientPort=2181

启动zookeeper。

4 浏览器登录http://localhost:8088/dubbo-admin-2.5.4/，会提示输入用户名和密码，输入2步的用户，就可以登录dubbo监控系统，可以查看已发布的服务。

## 第二章 IKAnalyzer 中文分词 lucene 搜索引擎

IKAnalyzer IKAnalyzer3.2.0Stable.jar [中文分词](http://baike.baidu.com/item/%E4%B8%AD%E6%96%87%E5%88%86%E8%AF%8D/371496" \t "http://baike.baidu.com/_blank)工具包

全文检索优化的[查询分析器](http://baike.baidu.com/item/%E6%9F%A5%E8%AF%A2%E5%88%86%E6%9E%90%E5%99%A8/5837347" \t "http://baike.baidu.com/_blank)IKQueryParser

lucene lucene-core-3.6.0.jar 全文搜素引擎

两种分词方法：

第一种：

ikAnalyzer 独立于lucene的java分词实现

StringReader input = **new** StringReader(“”);

IKSegmentation ikSeg = **new** IKSegmentation(input, **true**);

// true 用智能分词 //细粒度分词

第二种：

// 创建分词对象

Analyzer analyzer = new IKAnalyzer(true);

//被分的字节流

StringReader reader = **new** StringReader(text);

//将一个字符串创建成Token流

TokenStream ts = *analyzer*.tokenStream(“”, reader);

// 重置到流的开始位置

ts.reset();

CharTermAttribute term = ts.getAttribute(CharTermAttribute.**class**);

// 遍历分词数据

System.*out.print*("IKAnalyzer把关键字拆分的结果是：");

StringBuffer b = **new** StringBuffer();

**while** (ts.incrementToken()) {

System.*out.print*("【" + term.toString() + "】");

b.append("【" + term.toString() + "】");

}

reader.close();

## 第三章 git提交本地代码到远程

本地已安装git（没有就百度教程），有自己的github账户

1. 进入git操作界面，右击桌面 点击git bash
2. 切换到要提交的项目目录下 ，pwd cd ls dir 普通dos窗口操作命令
3. Git init 把文件夹变成Git可管理的仓库
4. Git add . 提交全部文件
5. git commit -m "first commit"
6. 在远程github上创建一个responsity，把https 地址复制下
7. Git remote add origin https://github.com/louyc/myself.git(之后再提交这个项目的东西就不需要6、7了)
8. git push -u origin master 提交到远程仓库

如果你还没有一份远程代码库的本地版本（例如，如果你在另一台机器上开始工作，这台机器上还没有用过这个项目），你首先需要拷贝（clone）它。去到你的代码库想要拷贝到的文件夹下，

git **clone** https:*//your\_username@bitbucket.org/your\_username/name\_of\_remote\_repository.git*

另一方面，如果你已经在本地的项目上工作了，只是想从远程代码库上取得它最新的版本，移动到项目的根目录下，并发送：

git pull origin master

## spring 注解

1.@controller 指定控制器，支持多请求

Xml配置文件需声明：

<context:component-scan base-package=”xx.xx.controller”/>

2.@requestMapping 注释哪个类或者方法来处理请求动作

@requestMapping(value=”/hello” ....)

请求参数不一样可以设置多个请求处理方法

1. 参数绑定注解

a.@requestParam 将指定的请求参数赋值给方法中的形参

@requestParam(value=“loginname”,required=true,defaultValue=”admin”)

b.@pathVariable 获取请求url中的动态参数

C.@requestHeader 将请求的头信息区数据映射到功能处理方法的参数上

d.@CookieValue 将请求的cookie数据映射到功能处理方法的参数上

e.@SessionAttributes 允许有选择的指定model属性转存到httpSession中

f.@modelAttribute 将请求参数绑定到model对象中

## Js html

<c:forEach items="${nodes}" var="node">

<c:out value="${node.name}"></c:out>

</c:forEach>

<span>返回值：</span>${node}

1. js直接获取modelandview中数据

var name = '${node}';

2.js遍历modelandview中list集合

var nodes = new Array();

var testlist = '${node}';

var json = eval("(" + testlist + ")");

alert("changdu::" + json.length);

for (var i = 0; i < json.length; i++) {

var obj = eval('(' + json[i] + ')');

nodes.push(obj);

}

Html 页面中直接获取url中信息

<script type="text/javascript">

// 根据参数名称获取value

var name = GetRequest("name");

$("#loginname").html(name);

function GetRequest(param) {

var url = location.search; //获取url中"?"符后的字串

var theRequest = new Object();

if (url.indexOf("?") != -1) {

var str = url.substr(1);

strs = str.split("&");

for (var i = 0; i < strs.length; i++) {

if (strs[i].split("=")[0].indexOf(param) >= 0) {

return strs[i].split("=")[1];

}

}

}

return null;

}

</script>

## linux搭建svn

1. 安装svn服务器

sudo apt-get install subversion

svnserve --version

1. 配置版本库

[root@singledb ~]# mkdir /u02/svn

[root@singledb ~]# svnadmin create /var/svn/davesvn

--davesvn为版本库名称

库下会生成3个文件：

[root@singledb conf]# ls

authz  passwd  svnserve.conf

1. 修改配置文件

修改svnserve.conf 如下：（每行开头空格一定要去掉，不然会报错）

anon-access = read

auth-access = write

password-db = passwd

authz-db = authz

修改password如下：

[users]

username = password

修改authz 文件如下：

[/]

\* = rw

1. 启动svn

启动：svnserve -d -r /var/svn/davesvn

查看svn：ps -ef | grep svn

关闭svn：kill -9 对应端口号

E:\lyc\eclipse-workspace\semioe-analysis\bin\src\main\webapp\alian\images\images-fullscreen\5.jpg

## linux mysql firewalld

ps -ef|grep mysql

usr/bin/mysql 是指：mysql的运行路径   
var/lib/mysql 是指：mysql数据库文件的存放路径   
usr/lib/mysql 是指：mysql的安装路径

Mysql：

systemctl start mysqld.service

添加端口：

firewall-cmd --add-port=3306/tcp

firewall-cmd --add-port=20881/tcp

firewall-cmd --add-port=2181/tcp

firewall-cmd --add-port=8090/tcp

firewall-cmd --add-port=8443/tcp

firewall-cmd --add-port=8005/tcp

firewall-cmd --add-port=8009/tcp

firewall-cmd --add-port=8080/tcp

查看端口：firewall-cmd --zone=public --list-ports

ps aux|grep getCimiss-surf.jar

在dubbo.properties 文件里，添加 dubbo.protocol.host=[物理地址](http://www.baidu.com/s?wd=%E7%89%A9%E7%90%86%E5%9C%B0%E5%9D%80&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YdPHmsnhckuWwBuAcsPjmz0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnWfYnH6YnW6dPH0kP1TYPHcY" \t "https://zhidao.baidu.com/question/_blank)

Hosts 10.161.170.84 iZ23x0tmgg7Z #将这个内网IP改成公网IP即

java -jar /usr/local/provider/semioe-manager-provider.jar &

**Chmod -R(包括子文件夹) 777 filename**

**Dubbo控制台 修改root 密码**

发现在/${user}/dubbo.properties 还有一份配置，dubbo-admin优先读取这份配置，删掉它。

## Redis服务查看、开启、关闭:

a.通过ps -ef|grep redis命令查看Redis进程

b.开启Redis服务操作通过/etc/init.d/redis\_6379 start命令，也可通过（service redis\_6379 start）

c.关闭Redis服务操作通过/etc/init.d/redis\_6379 stop命令，也可通过（service redis\_6379 stop）

基本操作 redis-cli

redis 127.0.0.1:6379> info #查看server版本内存使用连接等信息   
redis 127.0.0.1:6379> client list #获取客户连接列表   
redis 127.0.0.1:6379> client kill 127.0.0.1:33441 #终止某个客户端连接   
redis 127.0.0.1:6379> dbsize #当前保存key的数量   
redis 127.0.0.1:6379> save #立即保存数据到硬盘   
redis 127.0.0.1:6379> bgsave #异步保存数据到硬盘   
redis 127.0.0.1:6379> flushdb #当前库中移除所有key   
redis 127.0.0.1:6379> flushall #移除所有key从所有库中   
redis 127.0.0.1:6379> lastsave #获取上次成功保存到硬盘的unix时间戳   
redis 127.0.0.1:6379> monitor #实时监测服务器接收到的请求   
redis 127.0.0.1:6379> slowlog len #查询慢查询日志条数   
(integer) 3   
redis 127.0.0.1:6379> slowlog get #返回所有的慢查询日志，最大值取决于slowlog-max-len配置   
redis 127.0.0.1:6379> slowlog get 2 #打印两条慢查询日志   
redis 127.0.0.1:6379> slowlog reset #清空慢查询日志信息

## Mysql

SELECT u.count\_date,

GROUP\_CONCAT(u.province,":",u.sum) AS provinces

FROM

(select DATE\_FORMAT(create\_time, '%Y-%m-%d') as count\_date,

SUBSTRING\_INDEX(SUBSTRING\_INDEX(u.address,' ',1),'"',-1) as province,

sum(1) as sum

from api\_user\_info u

where

DATE\_FORMAT(create\_time, '%Y-%m-%d') &gt;= DATE\_FORMAT(#{startDate,jdbcType=VARCHAR}, '%Y-%m-%d')

and DATE\_FORMAT(create\_time, '%Y-%m-%d') &lt;= DATE\_FORMAT(#{endDate,jdbcType=VARCHAR}, '%Y-%m-%d')

and in\_use=1

group by count\_date,province

)u

GROUP BY

u.count\_date

## Linux 启动jar

**[Linux](http://lib.csdn.net/base/linux" \o "Linux知识库" \t "http://blog.csdn.net/qq_30739519/article/details/_blank)** 运行jar包命令如下：

方式一：

1. java -jar shareniu.jar

特点：当前ssh窗口被锁定，可按CTRL + C打断程序运行，或直接关闭窗口，程序退出

那如何让窗口不锁定？

方式二

1. java -jar shareniu.jar &

&代表在后台运行。

特定：当前ssh窗口不被锁定，但是当窗口关闭时，程序中止运行。

继续改进，如何让窗口关闭时，程序仍然运行？

方式三

nohup **[Java](http://lib.csdn.net/base/java" \o "Java 知识库" \t "http://blog.csdn.net/qq_30739519/article/details/_blank)** -jar shareniu.jar &

nohup 意思是不挂断运行命令,当账户退出或终端关闭时,程序仍然运行

当用 nohup 命令执行作业时，缺省情况下该作业的所有输出被重定向到nohup.out的文件中，除非另外指定了输出文件。

方式四

nohup java -jar semioe-manager-provider.jar >/usr/local/provider/out.log 2>&1 &

nohup java -jar semioe-provider.jar >/usr/local/provider/account/out.log 2>&1 &

解释下 >temp.txt

command >out.file

command >out.file是将command的输出重定向到out.file文件，即输出内容不打印到屏幕上，而是输出到out.file文件中。

可通过jobs命令查看后台运行任务

1. jobs

那么就会列出所有后台执行的作业，并且每个作业前面都有个编号。  
如果想将某个作业调回前台控制，只需要 fg + 编号即可。

1. fg 23

查看某端口占用的线程的pid

netstat -nlp |grep :9181

## Vue npm

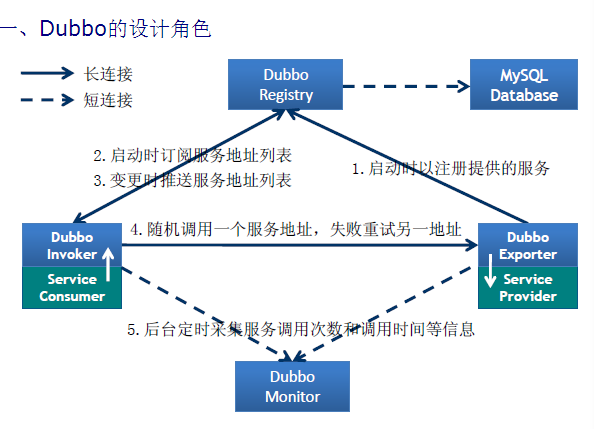
打包：

Npm install

Npm run build

## Dubbo 底层原理

Dubbo是阿里巴巴开源的一个分布式服务框架，致力于提供高性能和透明化的RPC远程服务调用方案，以及SOA服务治理方案



服务提供者 provider 提供服务，在注册中心注册服务

注册中心 zookper 通过注册中心 管理服务并展示出来，会把对应服务信息的具体ip返回消费者，如果有变更，注册中心将基于长连接推送变更数据给消费者

服务消费者 consumer 在注册中心订阅自己需要的服务

监控中心 监管所有服务 调用方 提供方 统计服务的调用次调和调用时间的监控中心。

1：注册服务

就是将服务装载容器中，然后准备注册服务。和Spring中启动过程类似，spring启动时，将bean装载进容器中的时候，首先要解析bean。所以dubbo也是先读配置文件解析服务。

详情看：

http://blog.csdn.net/chao\_19/article/details/51764150

## Dubbo超时机制

<dubbo:provider timeout=*"5000"* />

服务器响应慢或者网络原因 dubbo自身的超时重试机制 可能导致数据重复

经常发生场景： 发送邮件 账户注册

解决方案：

1：去掉dubbo自带的重试机制

<dubbo:provider delay="-1" timeout="6000"  retries="0"/>

或者根据实际情况 自己评估设定超时时间

2：业务代码必须放在服务端 客户端只做数据校验和服务调用，不涉及业务处理

## Zookper 理解

分布式协调服务

Zookper 和 dubbo关系

http://blog.csdn.net/daiqinge/article/details/51282874

## IO和NIO

区别：

1：NIO 增加一个角色 selector selectionKey

2：阻塞服务改为非阻塞服务：传统io流读取read()时，碰到无数据时，线程会阻塞被占用，NIO读取空时直接返回0，线程不阻塞

一个开饭店的例子：

IO：每个顾客有自己专门的服务员，当顾客少时没压力，当顾客越来越多，相应的服务员也越来越多，服务员就是一个线程，顾客点完餐，就餐时，服务员就会等待，浪费资源

NIO: 服务员时流动的，当顾客有需求时，分配服务员去完成，完成后就回到前台等待其他顾客的需求，这样就节约了资源。

NIO中，客户端创建一个连接后，先要将连接注册到Selector，相当于客人进入餐厅后，告诉前台你要用餐，前台会告诉你你的桌号是几号，然后你就可能到那张桌子坐下了，SelectionKey就是桌号。当某一桌需要服务时，前台就记录哪一桌需要什么服务，比如1号桌要点菜，2号桌要结帐，服务员从前台取一条记录，根据记录提供服务，完了再来取下一条。这样服务的时间就被最有效的利用起来了。

https://www.cnblogs.com/xiaoxi/p/6576588.html

## Spring

7个模块

1. Spring Core 核心容器

Core封装包是框架的最基础部分，提供IOC和依赖注入特性。这里的基础概念是BeanFactory,提供了对Factory模式的经典实现来消除对程序性单例模式的需要，并真正允许你从程序逻辑中分离出依赖关系和配置

1. Spring Context 应用上下文

构建于core封装包基础上的context封装包，提供了一种框架式的对象访问方法，类似JNDI注册器，context封装包的特性来自于Beans封装包，并添加了对国际化的支持，事件传播，资源装载的方式和context的透明创建，通过servlet容器

1. Spring DAO JDBC和DAO模块

提供JDBC抽象层，消除冗长的jdbc编码，并且提供了一种更好的声明性事物管理方法

1. Spring ORM 对象实体映射

提供了常用的对象、关系映射的集成层，包括hibernate、ibatis。可以混合使用所有spring提供的特性进行对象、关系映射

1. Spring AOP AOP模块

提供了面向方面的编程实现，可以定义拦截器和切点，减弱代码的功能耦合

1. Spring Web Web模块

提供了基础的针对web开发的集成特性，多文件上传，可以是spring和其他框架结合

1. Spring Web MVC MVC模块

提供了web应用的model-view-controller实现，并还可以借助spring框架其他特性

BeanFactory ApplicationContext 对比

applicationContext是从BeanFactory接口派生出来的，包含beanFacory的所有功能，候着无法支持spring的许多插件，aop、web等，前者是一种面向框架的方式工作以及对上下文分层实现，同时包括：

国际化的消息访问，messageSource

资源访问，url、文件

事件传播

载入多个上下文

对web应用的支持：

beanFactory以编程的方式被创建

ApplicationContext 能以声明方式创建（contextLoader），亦可以用编程方式创建

Spring bean 5种作用域

Singleton：ioc容器中只存在一个共享的bean实例

Prototype：对应多个实例，每次请求（调用getBean()或者将该bean注入到其他bean中）都会创建一个新的实例

Request：针对web的http请求（一个请求），每次http请求都会有各自的bean实例，仅在基于web的上下文中

Session：针对web的http session（一次会话），每次http请求都会有各自的bean实例，仅在基于web的上下文中

gloable session：全局的web域，类似于servlet中的application

单例Beans是线程安全的？不完全对，bean涉及状态修改的不安全用prototype作用域

Spring bean生命周期：

1. 实例化一个Bean，也就是我们通常说的new

2. 按照Spring上下文对实例化的Bean进行配置，也就是IOC注入

3. 如果这个Bean实现了BeanNameAware接口，会调用它实现的setBeanName(String beanId)方法，此处传递的是Spring配置文件中Bean的ID

4. 如果这个Bean实现了BeanFactoryAware接口，会调用它实现的setBeanFactory()，传递的是Spring工厂本身（可以用这个方法获取到其他Bean）

5. 如果这个Bean实现了ApplicationContextAware接口，会调用setApplicationContext(ApplicationContext)方法，传入Spring上下文，该方式同样可以实现步骤4，但比4更好，以为ApplicationContext是BeanFactory的子接口，有更多的实现方法

6. 如果这个Bean关联了BeanPostProcessor接口，将会调用postProcessBeforeInitialization(Object obj, String s)方法，BeanPostProcessor经常被用作是Bean内容的更改，并且由于这个是在Bean初始化结束时调用After方法，也可用于内存或缓存技术

7. 如果这个Bean在Spring配置文件中配置了init-method属性会自动调用其配置的初始化方法

8. 如果这个Bean关联了BeanPostProcessor接口，将会调用postAfterInitialization(Object obj, String s)方法

注意：以上工作完成以后就可以用这个Bean了，那这个Bean是一个single的，所以一般情况下我们调用同一个ID的Bean会是在内容地址相同的实例

9. 当Bean不再需要时，会经过清理阶段，如果Bean实现了DisposableBean接口，会调用其实现的destroy方法

10. 最后，如果这个Bean的Spring配置中配置了destroy-method属性，会自动调用其配置的销毁方法