## Mybatis

### 批量插入

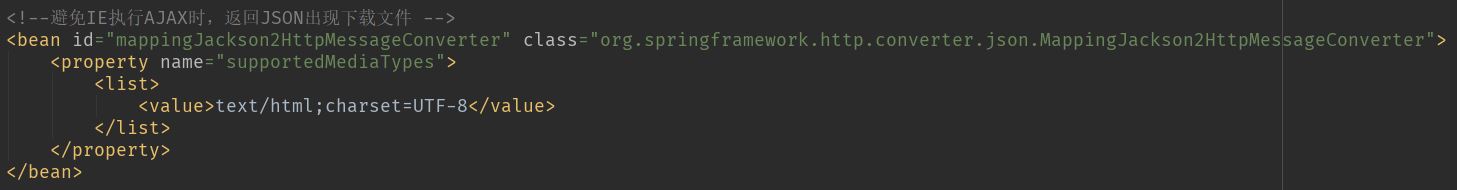


## Spring日常问题

1. java.lang.ClassNotFoundException:org.springframework.http.converter.json.MappingJacksonHttpMessageConverter .

解决方案：

由于Spring4.2.3以上版本，json解析相关方式发生变化，修改 mvc.xml 配置如下：



## jQuery控件相关

### 1.控件显示与隐藏

http://www.jb51.net/article/55696.htm document.getElementById("tr1").style.display='';

document.getElementById("tr2").style.display='none';

##### Js替换全部字符串

// var s = "LOVE LIFE ！ LOVE JAVA ...";  
// var replaceStr = "LOVE";  
// alert(s.replace(new RegExp(replaceStr,'gm'),'6'));

##### jQuery清空div

$("#existMounts").children("table").html("");//清空

##### 通过.class迭代每一个 $.each

**function** detailReason() {  
 $("#errorReason").val("");  
 $(".ceBox").each(**function** () {  
 **if** ($(**this**).is(':checked')) {  
 **if** ($("#errorReason").val().lastIndexOf($(**this**).val())) {  
 $("#errorReason").val($("#errorReason").val() + $(**this**).val());  
 }  
 }  
 })  
}

### 2.jquery id 中存在变量

$("#"+a+"\_tr").html("");

### 3.插入html 、innerHTML、append、after

$("#area\_four\_td").html("");

if(twoArea != null && twoArea != "" && twoTime != null && twoTime != "") {

$("#twoArea").html(twoArea + twoTime);

}

<tr>  
 <td align="left" style="padding-bottom: 2px">  
 <span class="spanClass">号牌号码:</span>  
 <span id="hpcode" style="font-family:隶书;font-size:20px;font-weight:bold;"></span>  
 <img style="width: 35px;height: 35px" id="cllxPicture">  
 </td>  
</tr>

//车牌号码  
**var** hpcode=document.getElementById("hpcode");  
**var** hpCodeStr='**$!{**condition.hpCode**}**'+'(**$!{**condition.cllxCn**}**)';  
hpcode.innerHTML=hpCodeStr;

### jQuery去空格

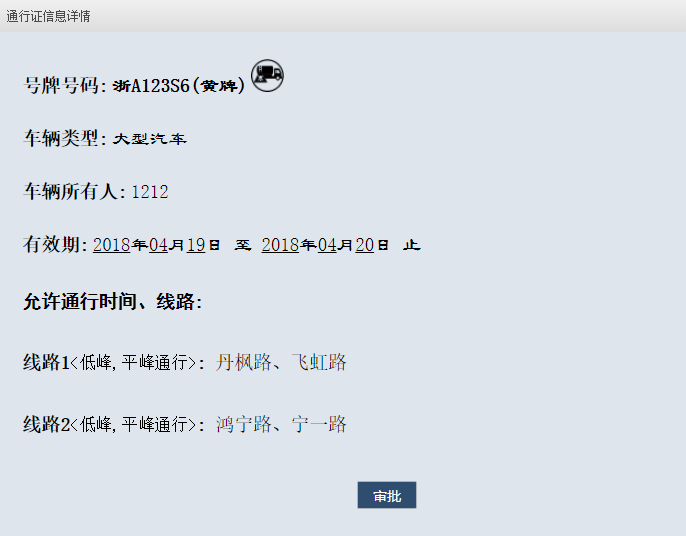
$.trim($("#flag").val())

### 4.append beofore after inner拼接HTML

**for**(**var** i=1;i<=bayArr.length;i++){  
 $("#gccXl").append("<tr id='tr"+i+"'> <td width='115px' align='right'><label>线路"+i+":</label></td><td class='ui-search-c' colspan='5'> <textarea id='roadShow"+i+"' onclick='setRailNum("+i+")' class='texa' cols='4' rows='4' >"+roadShowArr[i-1]+"</textarea></td> " +  
 "<td id='bayId"+i+"\_td'><input type='text' style='width: 1px;display: none' id='bayId"+i+"'></td>" +  
 "<td id='roadShow"+i+"\_td'><input style='width: 1px;display: none' type='text' id='roadShow"+i+"'></td>" +  
 "<td id='roadId"+i+"\_td'><input style='width: 1px;display: none' type='text' id='roadId"+i+"'></td></tr>");  
}

// function incPlus(){  
// $("#plusBotton").before($("#copyTrModel tr").eq(0).prop("outerHTML"));  
// getComboxSelect("timeCombox","--全部--");  
// }

**Inner**



<tr>  
 <td align="left" style="padding-bottom: 2px">  
 <span class="spanClass">有效期:</span>  
 <span style="font-family:隶书;font-size:20px;" id="passTime"></span>  
 </td>  
</tr>

----------------------------- JS ---------------------------

**var** passTime=document.getElementById("passTime");

**var** passTimeHtml='<span style="text-decoration:underline">'+year1+'</span>年<span style="text-decoration:underline">'+mon1+'</span>月<span style="text-decoration:underline">'+day1+'</span>日 至 ' +  
 '<span style="text-decoration:underline">'+year2+'</span>年<span style="text-decoration:underline">'+mon2+'</span>月<span style="text-decoration:underline">'+day2+'</span>日 止 ';  
passTime.innerHTML=passTimeHtml;

#### 4.1两个div一行排列

<div>

<div style="display: inline">  
 <table style="float: right">  
 <tr>  
 <td align="left" style="padding-bottom: 2px">  
 <span class="spanClass">有效期:</span>  
 <span style="font-family:隶书;font-size:20px;" id="passTime"></span>  
 </td>  
 </tr>  
 </table>  
</div>

<div style="display: inline">  
 <table style="float: right">  
 <tr>  
 <td align="left" style="padding-bottom: 2px">  
 <span class="spanClass">有效期:</span>  
 <span style="font-family:隶书;font-size:20px;" id="passTime"></span>  
 </td>  
 </tr>  
 </table>  
</div>

</div>

#### 4.2 在table末尾拼接HTML

<table class="printDiv" id="tbXl" style="margin-top:10px;" width="100%">  
  
 <tr>  
 <td align="left" style="padding-bottom: 2px">  
 <span class="spanClass">号牌号码:</span>  
 <span id="hpcode" style="font-family:隶书;font-size:20px;font-weight:bold;"></span>  
 <img style="width: 35px;height: 35px" id="cllxPicture">  
 </td>  
 </tr>  
   
 <tr >  
 <td width="350px" align="left" style="padding-top: 5px;padding-bottom: 10px" >  
 <span style="font-family:黑体;font-size:20px;font-weight:bold;">允许通行时间、线路:</span>  
 </td>  
 </tr>  
</table>

Js

//线路  
**var** roadShow="**$!{**condition.roadShow**}**";  
**var** roadShowArr= roadShow.split("#");  
**for**(**var** i=1;i<=roadShowArr.length;i++){  
 **var** show=roadShowArr[i-1];  
 $("#tbXl").append("<tr id='tr"+i+"'> <td align='left' style='padding-bottom: 10px'><span style='font-family:宋体;font-weight:bold;font-size:20px;'>线路"+i+"</span><span style='font-family:宋体;font-size:18px;'>"+"**$!{**condition.compareTimeSix**}**"+"</span><span style='font-family:宋体;font-weight:bold;font-size:20px;'>:</span><span id='xl"+i+"' style='font-family:宋体;font-size:20px;'> "+show+" </span> </td> </tr>")  
  
}

#### 4.3获得屏幕宽度 界面自适应

$(document).ready(**function** () {  
 //获得屏幕宽度 界面自适应  
 **var** scrWidth = document.documentElement.clientWidth;  
 **var** scrHeight=document.documentElement.clientHeight;  
  
 document.getElementById("body").style.height= scrHeight +"px";  
 document.getElementById("body").style.width= scrWidth +"px";  
   
 **var** nowDate = getNowFormatDate();  
 echartRun('**$!{**rc.contextPath**}**',nowDate);  
 //得到屏幕的宽度和高度  
})

### 5.Ztree

（0）当前勾选

**var** treeObj=$.fn.zTree.getZTreeObj("tree");  
 nodes=treeObj.getCheckedNodes(**true**);

判断是否选中

**if**(!treeNode.checked){

#### (1) 对之前线路的路去勾选

function clearCheck(){

$(document).ready(function () { });

//对之前线路的路去勾选

var treeObj = $.fn.zTree.getZTreeObj("tree");

var selectNodes = treeObj.getCheckedNodes(true);

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

treeObj.checkNode(selectNodes[i], false, false);

}

}

#### (3)回显树节点

//修改单条线路  
 **function** setRailNum(a){  
 **if**(confirm("确定要修改当前线路吗？")){  
 clearCheck();  
 **var** oldBayIds= $("#bayId"+a).val();  
 **var** oldBayIdArr= oldBayIds.split(",");  
// $("#roadShow"+a).val("");  
 **var** oldRoad= $("#roadId"+a).val();  
 //  
 **var** treeObj=$.fn.zTree.getZTreeObj("tree");  
 treeObj.checkAllNodes(**false**);  
 **var** nodes=treeObj.transformToArray(treeObj.getNodes());//将 zTree 使用的标准 JSON 嵌套格式的数据转换为简单 Array 格式  
 //勾选旧的  
 **for**(**var** i=0;i<oldBayIdArr.length;i++){  
 **for**(**var** m=0;m<nodes.length;m++){  
 **if**(oldBayIdArr[i]==nodes[m].id){//当已经存在的ID等于循环出的ID时候，点击选中  
 treeObj.checkNode(nodes[m], **true**, **true**);  
 }  
 }  
 }  
 //  
 $("#editRailNum").val(a);  
 //初始化完成后，增减或减少单条线路的道路的页面展示在 creeateRightPageHtml() 方法中  
 }**else**{  
 **return**;  
 }  
  
 }

#### (3)搜素树节点

搜素框

<div >  
 <input type="text" id="searchNode" onkeypress="**if** (event.keyCode == 13) searchNode();" style="margin-top: 20px;margin-left: 30px ;width: 130px;height: 20px;">  
 <img onclick="searchNode()" style="position: absolute;margin-top: 20px;height: 25px" src="**$!{**\_platform\_assets\_url**}**/picture/search.png">  
</div>

树结构

<div class="ui-layout-center" style="overflow:auto;height:100%;margin-top:5px;background:#DEE5ED">  
 <ul id="tree" class="ztree" valign=top style="overflow:auto;height:80%;margin-top:5px;background:#DEE5ED"></ul>  
</div>

搜索方法

*/\*\*  
 \* Created with IDEA  
 \*****@Author*** *zhaofennan@supcon.com  
 \*****@Date*** *2018-1-29 15:30  
 \*****@Description*** *搜索树节点  
 \*/***function** searchNode() {  
 **var** zTree = $.fn.zTree.getZTreeObj("tree");  
 //显示隐藏的节点  
 nodes = zTree.getNodesByParam("isHidden", **true**);  
 zTree.showNodes(nodes);  
  
 **var** root = zTree.getNodeByParam("level", "0");  
  
 **var** hiddenNodes = **new** Array();  
  
 //筛选出结果  
 **var** inputStr = $("#searchNode").val();  
 **if**(inputStr==""){  
 notify("请先输入要搜索到内容");  
 **return**;  
 }  
  
 filterNodes(root, inputStr, hiddenNodes);  
 zTree.hideNodes(hiddenNodes);  
}  
*/\*\*  
 \* 遍历树节点，将  
 \* 1.自身不满足搜索条件  
 \* 且  
 \* 2.其子节点不包含有满足条件的节点  
 \* 的节点加入到filterResult中  
 \*  
 \** ***@param*** *node 查询的节点  
 \** ***@param*** *inputStr 搜索条件  
 \** ***@param*** *filterResult 过滤的结果集  
 \** ***@return*** *该节点是否满足条件  
 \*/***function** filterNodes(node, inputStr, filterResult){  
 **if**(node != **null**){  
 //自身是否符合搜索条件  
 **var** selfMatch = node.name.indexOf(inputStr) > -1;  
 //子节点是否有满足的条件的节点  
 **var** childMatch = **false**;  
  
 **var** children = node.children;  
 **if**(children != undefined){  
 **for**(index in children){  
 childMatch = filterNodes(children[index], inputStr, filterResult) || childMatch;  
 }  
 }  
  
 //自身不满足搜索条件 且其子节点不包含有满足条件的节点  
 **if**(!selfMatch && !childMatch){  
 filterResult.push(node);  
 }  
  
 **return** selfMatch || childMatch;  
 }**else**{  
 **return true**;  
 }  
}

##### js去掉相同节点

//去除重复的要素信息  
**function** removeSameData(parNodes){  
 **for**(**var** i=0;i<parNodes.length;i++){  
 **for**(**var** j=i+1;j<parNodes.length;j++){  
 **if**(parNodes[i].id==parNodes[j].id){  
 parNodes.splice(i, 1);  
 i-=1;  
 **break**;  
 }  
 }  
 }  
 **return** parNodes;  
}

##### 转换为节点数组

**var** treeObj=$.fn.zTree.getZTreeObj("tree");  
treeObj.checkAllNodes(**false**);  
**var** nodes=treeObj.transformToArray(treeObj.getNodes());

##### 清空相同的节点

// function removeSameData(parNodes){  
// for(var i=0;i<parNodes.length;i++){  
// for(var j=i+1;j<parNodes.length;j++){  
// if(parNodes[i].id==parNodes[j].id){  
// parNodes.splice(i, 1);  
// i-=1;  
// break;  
// }  
// }  
// }  
// return parNodes;  
// }

## Kendo UI 相关

### 1.多选框数据回显

**var** isSpecial = $("#isSpecial").val();  
$("#area\_four").kendoMultiSelect({  
 autoClose: **false**,  
 autoBind: **true**,  
 //filter:"contains",  
 dataTextField: "name",  
 dataValueField: "code",  
 //index:-1,  
 dataSource: {  
 transport: {  
 read: {  
 dataType: "json",  
 url: "**$!{**rc.contextPath**}**/passcardInfo/selectAreaTime",  
 data: {  
 cardId: cllx,  
 isSpecial:isSpecial  
 }  
 }  
 }  
 }  
});  
  
$("#area\_four").data("kendoMultiSelect").value(["**$!{**condition.compareTimeSix**}**"]);

## H5属性

### 页面跳转

###### window.location.href="$!{rc.contextPath}/Oa/home";

//在ifram中跳转页面

###### $("#iframe").attr("src","$!{rc.contextPath}/Oa/home");

### attr 设置属性

$("#aa").attr({width:"50",height:"80"});

$("#iframe").attr("src","**$!{**rc.contextPath**}**/Oa/home");

<input id="checkbox1" type="checkbox" >

<input id="checkbox2" type="checkbox”>

$("#checkbox1").attr("checked")

$("#checkbox2").attr("checked")

//名称

**if**($("input[name='checkList.carNumFlag']").is(':checked')){

head+='<th>车牌号</th>'

}

#### Js下一张图片（src）

**var** element = document.getElementById("picPath");  
element.src = prevLine.picPath;

#### 事件函数

$(document).ready(function(){ //文档准备函数

});

$(“#a”).click(function(){

});

// js中设置css

$("p").css("background-color","red");



鼠标事件



###### 隐藏控件

document.getElementById("a").style.display='none'；

//通过名称

$("p").hide();

$("p").show();

//渐变隐藏

$("p").hide(1000);

//隐藏与显示切换

$("button").click(function(){

$("p").toggle(1000);

});

淡入

$("button").click(function(){

$("#div1").fadeIn();

$("#div2").fadeIn("slow");

$("#div3").fadeIn(3000);

});

## Linux常用操作命令

cd /home

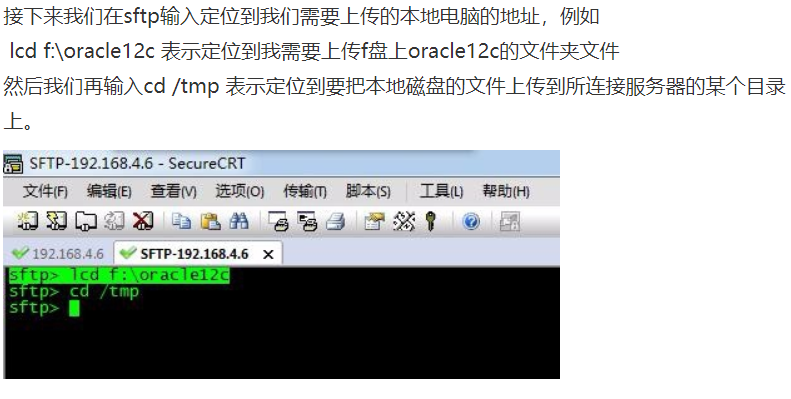
rm -rf catalina.out

get catalina.out

get catalina.2018-05-22.log

cd /home/tomcat/logs/

### SFTP命令Linux服务器上传 下载









### Linux常用命令：

链接 https://www.cnblogs.com/ccy1106/p/6637661.html

ls　　        显示文件或目录

     -l           列出文件详细信息l(list)

     -a          列出当前目录下所有文件及目录，包括隐藏的a(all)

mkdir         创建目录

     -p           创建目录，若无父目录，则创建p(parent)

cd               切换目录

touch          创建空文件

echo            创建带有内容的文件。

cat              查看文件内容

cp                拷贝

mv               移动或重命名

rm               删除文件

     -r            递归删除，可删除子目录及文件

     -f            强制删除

find              在文件系统中搜索某文件

wc                统计文本中行数、字数、字符数

grep             在文本文件中查找某个字符串

rmdir           删除空目录

tree             树形结构显示目录，需要安装tree包

pwd              显示当前目录

ln                  创建链接文件

more、less  分页显示文本文件内容

head、tail    显示文件头、尾内容

ctrl+alt+F1  命令行全屏模式

**系统管理命令**

stat              显示指定文件的详细信息，比ls更详细

who               显示在线登陆用户

whoami          显示当前操作用户

hostname      显示主机名

uname           显示系统信息

top                动态显示当前耗费资源最多进程信息

ps                  显示瞬间进程状态 ps -aux

du                  查看目录大小 du -h /home带有单位显示目录信息

df                  查看磁盘大小 df -h 带有单位显示磁盘信息

ifconfig          查看网络情况

ping                测试网络连通

netstat          显示网络状态信息

man                命令不会用了，找男人  如：man ls

clear              清屏

alias               对命令重命名 如：alias showmeit="ps -aux" ，另外解除使用unaliax showmeit

kill                 杀死进程，可以先用ps 或 top命令查看进程的id，然后再用kill命令杀死进程。

**打包压缩相关命令**

gzip：

bzip2：

tar:                打包压缩

     -c              归档文件

     -x              压缩文件

     -z              gzip压缩文件

     -j              bzip2压缩文件

     -v              显示压缩或解压缩过程 v(view)

     -f              使用档名

例：

tar -cvf /home/abc.tar /home/abc              只打包，不压缩

tar -zcvf /home/abc.tar.gz /home/abc        打包，并用gzip压缩

tar -jcvf /home/abc.tar.bz2 /home/abc      打包，并用bzip2压缩

当然，如果想解压缩，就直接替换上面的命令  tar -cvf  / tar -zcvf  / tar -jcvf 中的“c” 换成“x” 就可以了。

**关机/重启机器**

shutdown

     -r             关机重启

     -h             关机不重启

     now          立刻关机

halt               关机

reboot          重启

## Velocity 做权限管理

1. 后台封装menuArr权限数组 : 当前用户包含的权限数组

String permission= "user:userManage/add;user:userManage/delete;user:userManage/edit";  
String a []=permission.split(";");  
model.put("menuArr",a);

1. 前台循环权限判断是否包含 显示当前的权限

**#foreach**(**$**a **in $!{**menuArr**}**)  
 **#if**("user:userManage/delete"==**$**a)  
 <li>  
 <a onclick="userOnline();" class="img">  
 <p><a>员工之家</a></p>  
 </li>  
 **#end  
#end**

## 搭建zookeeper和kafka 集群

参考资料：

https://blog.csdn.net/jiecxy/article/details/52348120

https://www.cnblogs.com/luotianshuai/p/5206662.html

<http://orchome.com/454>

常用命令：https://blog.csdn.net/tototuzuoquan/article/details/73432256

### 第一步:准备工具

3台linux服务器 jdk1.8 securCRT

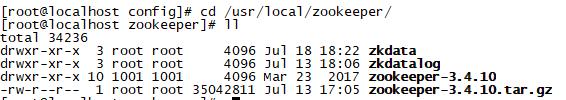
Zookeeper3.4.10 kafka\_2.11-1.1.0

### 第二步：安装zookeeper

（1）安装包放到 /usr/local/zookeeper 文件夹下，减压

（2）创建相关文件夹

cd /usr/local/zookeeper/zookeeper-3.4.10/bin;./zkServer.sh stop

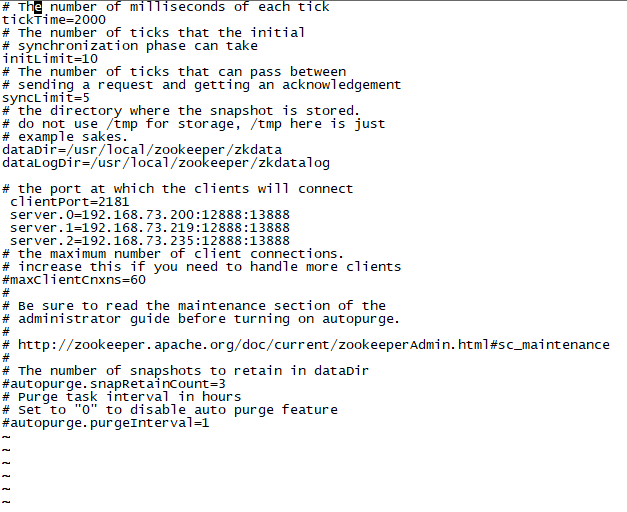


在zkData里创建myid文件012

（3）修改配置文件



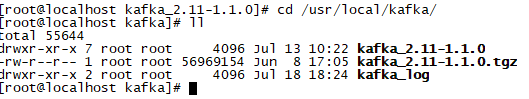
改动内容：



### 第三步： kafka 安装

（1）安装包放到 /usr/local/kafka 文件夹下，减压

（2）创建相关文件夹



（3）修改配置文件



改动内容：

broker.id=1

zookeeper.connect=192.168.73.200:2181,192.168.73.219:2181,192.168.73.235:2181

delete.topic.enable=true

num.partitions=1

log.dirs=/usr/local/kafka/kafka\_log

log.retention.hours=168

log.retention.check.interval.ms=300000

message.max.byte=5242880

#生产者发布内容的监听

listeners = PLAINTEXT://192.168.73.219:9092

host.name=192.168.73.219

注：内容123也要创建，具体看连接资料

同样的方式安装另外两台服务器

新建topic：

bin/kafka-topics.sh --create --topic topicname --replication-factor 1 --partitions 1 --zookeeper localhost:2181

查看topic：

bin/kafka-topics.sh --list --zookeeper localhost:2181

查看top详情：

sh bin/kafka-topics.sh --topic test --describe --zookeeper hadoop11:2181

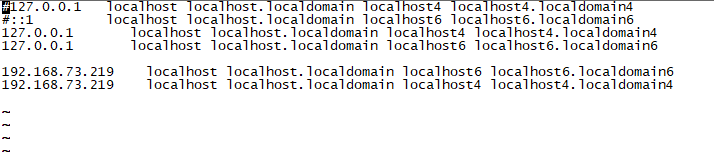
修改partition数量：

bin/kafka-topics.sh --zookeeper cdh1:2181 -alter --partitions 15 --topic wz\_bayonet

### 第四步：修改hosts



修改内容：



127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4

127.0.0.1 localhost localhost.localdomain localhost6 localhost6.localdomain6

192.168.73.219 localhost localhost.localdomain localhost6 localhost6.localdomain6

192.168.73.219 localhost localhost.localdomain localhost4 localhost4.localdomain4

### 第五步：先启动zookeeper 再启动kafka

创建topic 创建 producer 创建consummer

具体操作命令见 资料

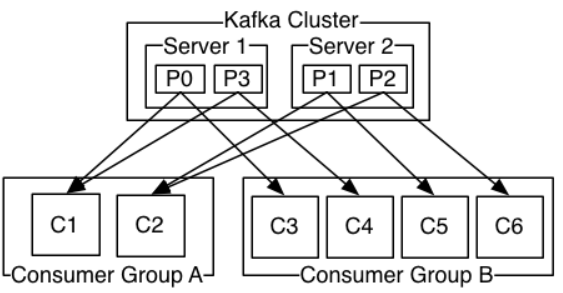
关闭要先kafka 再 zookeeper

### 第六步：java代码编写demo

参考：<https://blog.csdn.net/zo_184/article/details/52476307>

<https://blog.csdn.net/hekf2010/article/details/78982163>

## KAFKA工作原理



每个分区拥有一个Leader节点，和零或多个Follower。Leader处理该分区所有的读写请求，Follower复制Leader数据。如果Leader节点宕机，将会有一个Follower节点自动的转化为Leader。每个节点成为其部分分区的Leader，并成为剩余分区的Follower，这样整个集群的负载将比较均衡。

这是Kafka消息分区造成的，你可以去了解一下Kafka是如何分区的，就知道原因了。

问题原因可能是：你的所有消息的Key都是一样的,使用默认的Partitioner: hash(key)%numPartitions,这样每次的partion num都是一样的，所以数据都落到一个分区了。

而同一consumer grop并行消息，也是按照分区来分配的，因为只有一个分区上有数据，所以有一个consumer始终拿不到消息。

解决办法：1.自定义分区函数。2.消息散列为不同的key

### Kafka性能优化

<https://blog.csdn.net/qq_20641565/article/details/59746101>

<https://blog.csdn.net/baidu_39530785/article/details/82796507>

partition优化：

http://www.cnblogs.com/zhengchunyuan/p/9761767.html

## 文件、文件夹监听

### commons-io

### Jnotify

http://cybrc.iteye.com/blog/1900042

## Hadoop+hbase集群

注意二者版本兼容

Hbase地址： <http://archive.apache.org/dist/hbase/>

清除旧公钥：

ssh-keygen -R 192.168.73.219 <https://blog.csdn.net/nahancy/article/details/51052127>

cp ~/.ssh/id\_rsa.pub ~/.ssh/192.168.73.200\_id\_rsa.pub

scp ~/.ssh/192.168.72.200\_id\_rsa.pub 192.168.73.235:~/.ssh/

cat ~/.ssh/id\_rsa.pub >> ~/.ssh/authorized\_keys

cat ~/.ssh/192.168.73.200\_id\_rsa.pub >> ~/.ssh/authorized\_keys

cat ~/.ssh/192.168.73.219\_id\_rsa.pub >> ~/.ssh/authorized\_keys

scp ~/.ssh/authorized\_keys 192.168.73.200:~/.ssh

scp ~/.ssh/authorized\_keys 192.168.73.219:~/.ssh

### 集群免密码登录靠谱：

<https://www.linuxidc.com/Linux/2017-03/141296.htm>

ssh-keygen -t rsa 产生两个文件

将产生的.pub文件复制到 cat id\_rsa.pub >> authorized\_keys 所有节点都要执行

在slaver操作：将master的.pub复制到自己的.ssh目录下，

scp root@cdh1:~/.ssh/id\_rsa.pub ~/.ssh/cdh1\_rsa.pub

将复制过来的master\_rsa.pub添加进auth…文件

cat ~/.ssh/master\_rsa.pub >> ~/.ssh/authorized\_keys

然后,在master节点上 scp [grid@master:~/.ssh/id\_rsa.pub ~/.ssh/master\_rsa.pub](mailto:grid@master:~/.ssh/id_rsa.pub%20~/.ssh/master_rsa.pub)

scp root@cdh1:~/.ssh/id\_rsa.pub ~/.ssh/cdh1\_rsa.pub

产生master\_rsa.pub；

并将其加入到自己的 author\_...

所有节点：ssh localhost master节点：ssh 每个slaver

OK，master可以免密码登录slaver1

核心就是 master的id\_rsa.pub必须在slaver的author…里

### Hadoop+hbase集群搭建

<https://blog.csdn.net/h952520296/article/details/79629195> 这个比较好

https://blog.csdn.net/loopeng1/article/details/79012055

<https://blog.csdn.net/reblue520/article/details/70888850>

### 商业版CDH5搭建

<https://blog.csdn.net/u010077555/article/details/78340141>

<https://www.cnblogs.com/zhangleisanshi/p/7575579.html>

#### cdh5报错

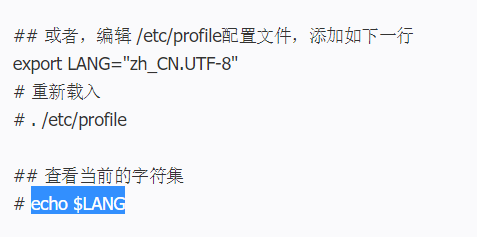
1.测试数据库时：Unexpected error.Unable to verify database connection…

原因参考：<https://blog.csdn.net/qq_15758463/article/details/78424429>

2．Logon denied for user/password. Able to find the database server and database, but logon request was rejected

 解决办法：mysql> grant all on hive.\* TO ‘root’@’主机名’ IDENTIFIED BY ‘123456’;   
mysql> flush privileges;

### linux中文乱码解决



### Linux常用命令

#export LANG=en\_US

#source /etc/profile

service vsftpd status 、、、、ftp

ps -ef|grep tomcat7.0.88

kill -9 30085

ps -ef|grep tomcat7.0.88|awk '{print $2}'|xargs kill -9

ps -ef|grep tomcat7.0.90|awk '{print $2}'|xargs kill -9

tail -f catalina.out

tcpdump -i p8p1 port 8080 -n -s 0 -w data.cap -v ifconfig 抓包

## 抓包

抓包：tcpdump -i lo port 8080 -n -s 0 -w data.cap -v ifconfig

tcpdump -i any port 8080 -n -s 0 -w data.cap –v

## webservice 服务端和客户端开发

### base64与图片流DataHandlar相互转换：

**if**(firstPic!=**null**){

**byte**[] readBytesFromStream1;

**try** {

readBytesFromStream1 = IOUtils.*readBytesFromStream*(firstPic.getInputStream());

String encodeBase64String1 = Base64.*encodeBase64String*(readBytesFromStream1);

//// ##########start: 转回DataHandlar #######################################  
//// BASE64Decoder decode = new BASE64Decoder();  
//// byte[] bytes = decode.decodeBuffer(encodeBase64String1);  
//// DataSource source = new ByteDataSource(bytes);  
//// DataHandler reFirst = new DataHandler(source);

### 服务端开发(CXF+SSM)：

#### Spring4.0.5+ SpringCXF2.7.10

参考：https://blog.csdn.net/u013305082/article/details/51887562?utm\_source=blogxgwz0

1. 倒jar包
2. <!-- webService:CXF Dependencies -->  
   <dependency>  
    <groupId>org.apache.cxf</groupId>  
    <artifactId>cxf-rt-frontend-jaxws</artifactId>  
    <version>${cxf.version}</version>  
   </dependency>  
   <dependency>  
    <groupId>org.apache.cxf</groupId>  
    <artifactId>cxf-rt-transports-http</artifactId>  
    <version>${cxf.version}</version>  
   </dependency>  
   <dependency>  
    <groupId>org.apache.cxf</groupId>  
    <artifactId>cxf-rt-transports-http-jetty</artifactId>  
    <version>${cxf.version}</version>  
   </dependency>  
   <dependency>  
    <groupId>org.apache.cxf</groupId>  
    <artifactId>cxf-rt-ws-security</artifactId>  
    <version>${cxf.version}</version>  
   </dependency>  
   <dependency>  
    <groupId>org.apache.cxf</groupId>  
    <artifactId>cxf-rt-ws-policy</artifactId>  
    <version>${cxf.version}</version>  
   </dependency>  
   <dependency>  
    <groupId>org.apache.cxf</groupId>  
    <artifactId>cxf-bundle-jaxrs</artifactId>  
    <version>${cxf.version}</version>  
   </dependency>  
   <dependency>  
    <groupId>javax.ws.rs</groupId>  
    <artifactId>jsr311-api</artifactId>  
    <version>1.1.1</version>  
   </dependency>  
   <dependency>  
    <groupId>commons-httpclient</groupId>  
    <artifactId>commons-httpclient</artifactId>  
    <version>3.0</version>  
   </dependency>  
   <!-- webService:End of CXF Dependencies -->

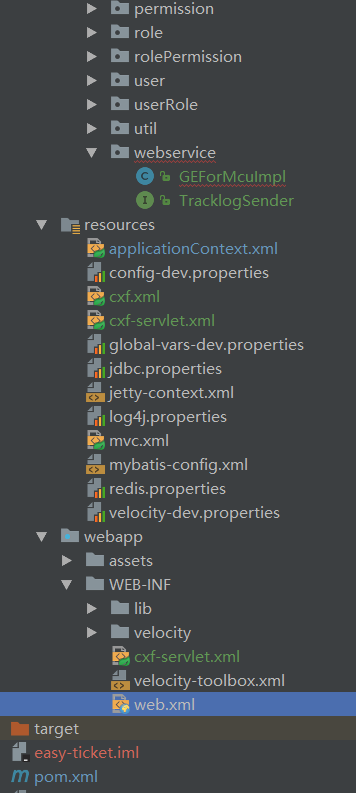
2两个配置文件

1. Spring配置文件

1. 服务端编码见参考链接
2. 图片结构



#### Spring4.2.3+SpringCXF3.1.8

1.接口和实现类



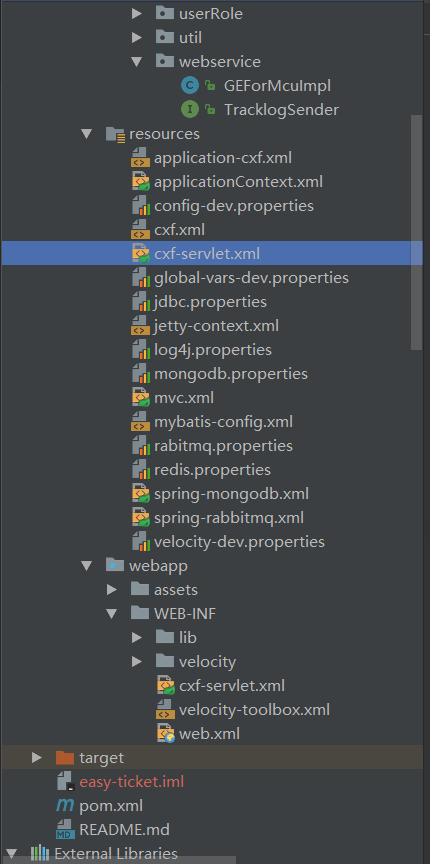
2配置文件和web.xml



1. 架包POM



1. 整体结构;



### 客户端测试：

<https://www.cnblogs.com/charlypage/p/9162631.html>

cxf方式：

1.通过Apache cxf 的wsdl2java生成

1).先下载apace cxf包，配置环境变量，和java 的环境变量的配置方式一样。

在环境变量中配置CXF\_HOME 值为D:\cxf\apache-cxf-3.0.0,在PATH中加入%CXF\_HOME%\bin

验证环境变量有无配置成功，cmd 窗口，输入  wsdl2java

2）根据可用wsdl文件生成cxf客户端代码

wsdl2java -encoding utf8 -p com.pubinfo.controller -d D:\ <http://122.224.82.77:8090/webws/webservice/Tracklog?wsdl>

java调用：

String a = new TracklogImplService().getTracklogImplPort().writeTracklogData((String) map.get("deviceId"),(String)map.get("agencyId"))；

## 远程连接专网

先登录：33.112.8.72 3377 megaeyes+16#zwpt

然后：ssh -p 3377 33.112.9.24 megaeyes+16#zwpt

ssh -p 22 33.115.245.3

远程部署：

scp -P 3377 easy\_ticket.war [root@33.112.9.24:/usr/local/tomcat-9.0.11/webapps/](mailto:root@33.112.9.24:/usr/local/tomcat-9.0.11/webapps/)

## Oracle数据库

Pivot

### 数据库事物并发机制：acid

参考：<https://blog.csdn.net/justloveyou_/article/details/70312810>

原子性：原子性是指事务包含的所有操作要么全部成功，要么全部失败回滚

一致性：一致性是指事务执行的结果必须使数据库从一种一致性状态变到另一种一致性状态（转账）

隔离性：**隔离性是指并发执行的事务之间不能相互影响**

**持久性：**持久性是指一个事务一旦被提交了，那么对数据库中的数据的改变就是永久性的

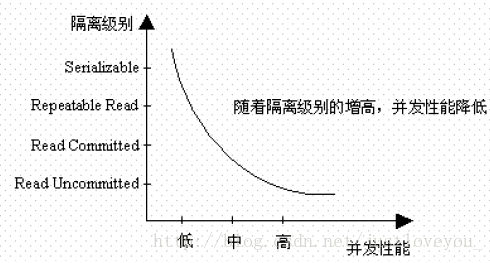
脏读：**脏读是指在一个事务处理过程中读取了另一个事务未提交的数据（读取未提交）**

幻读：**幻读是事务非独立执行时发生的一种现象，即在一个事务读的过程中，另外一个事务可能插入了新数据记录，影响了该事务读的结果（）**

不可重复读：**对于数据库中的某个数据，一个事务范围内多次查询却返回了不同的数据值（转账与消费）**

### 数据库的事务隔离级别





## Mysql数据库

### mysql5.6安装

<https://blog.csdn.net/changyinling520/article/details/80963015>

1. 下载解压 2.添加用户用户组 3…..顺序操作 4.创建my.cnf

关键步骤：初始化 注册启动服务 建立快捷键

<https://www.cnblogs.com/guoyh/p/8125268.html>



# ln -s /usr/local/mysql/bin/mysql /usr/bin

创建用户和连接

mysql和mysqldump出现command not found 问题解决

1、查找mysql安装路径

find / -name mysql

通常mysql安装路径在:/usr/local/mysql/bin/mysql

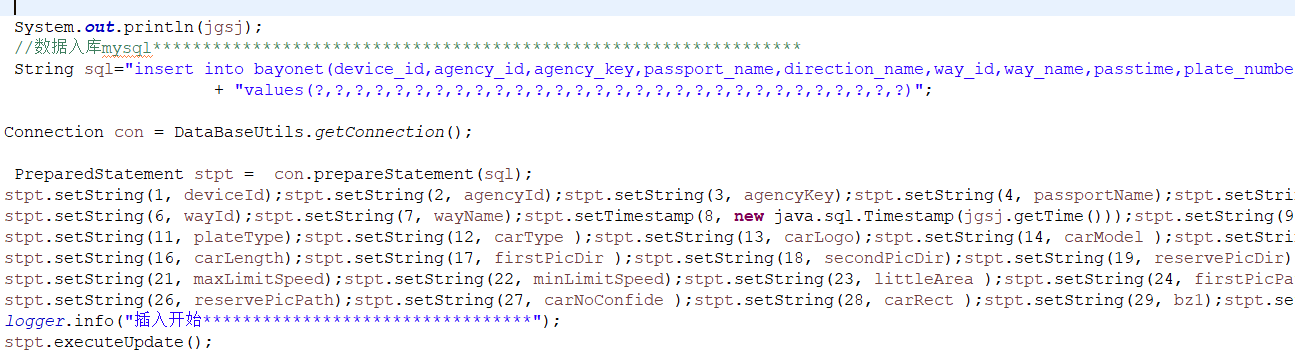
2、mysql:command not found建立软连接

ln -s  /usr/local/mysql/bin/mysql  /usr/bin

3、mysqldump:command not found 建立软连接

ln -s  /usr/local/mysql/bin/mysqldump  /usr/bin

#### mysql插入数据有年月日没时分秒：



## FTP （vsftpd）安装，路径指定

https://blog.csdn.net/firewolf1758/article/details/73822082

## mongodb安装教程

<https://blog.csdn.net/A_Gorilla/article/details/80957374>

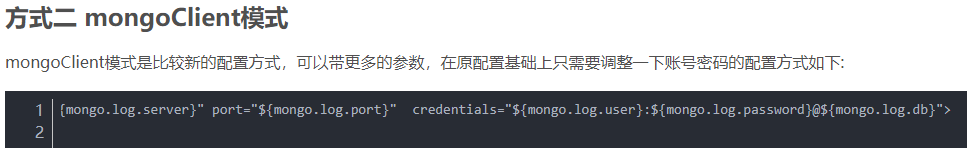
注意：mongodb.config配置文件多加 bind\_ip=0.0.0.0 不然客户端（robot3T）无法远程连接。

### 遇到的坑：

（1）Id自增 ：见demo

（2）权限验证 ：配置文件中用户名或密码错了

配置参数用户名，密码，数据库顺序颠倒



### mongodb创建数据库授权

<https://wenku.baidu.com/view/04204fad52d380eb63946d0c.html>

db.createUser( {user: "admin",pwd: "admin123",roles: [ { role: "userAdminAnyDatabase", db: "admin" } ]})

db.createUser( {user: "gxd",pwd: "gxd",roles: [ { role: "dbOwner", db: "gxd" } ]})

db.grantRolesToUser（"gxd",[{role:"dbOwner",db:"gxd"}]）

### 语法语句

#### 增删改查：

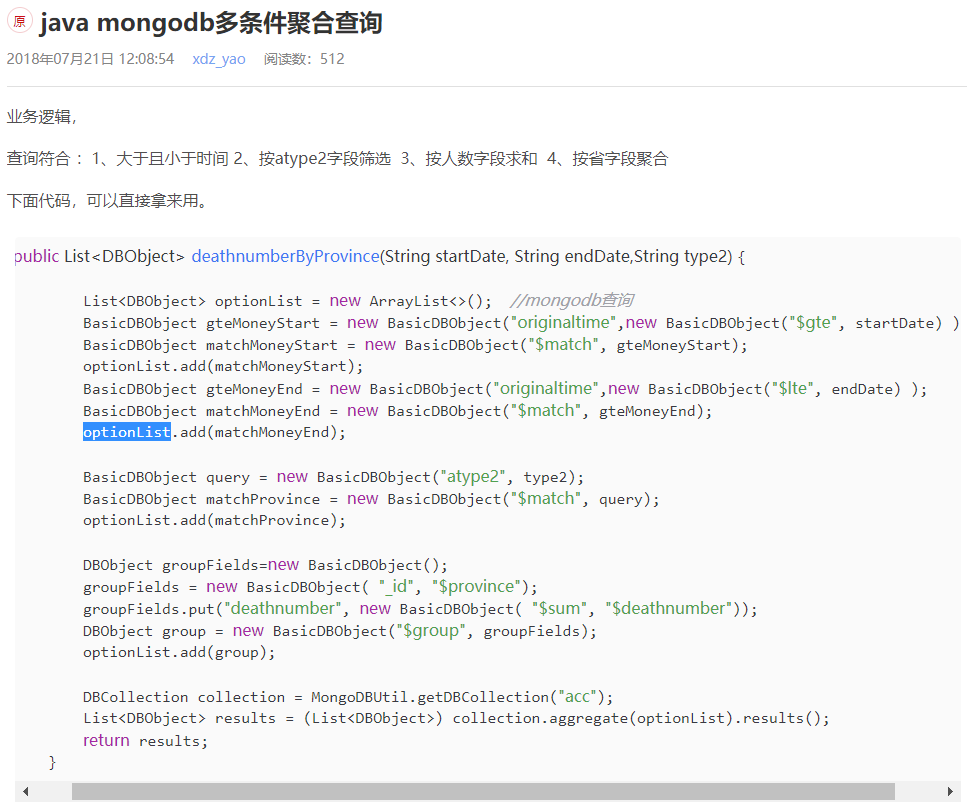
<https://www.jianshu.com/p/941a9ec2a93c>

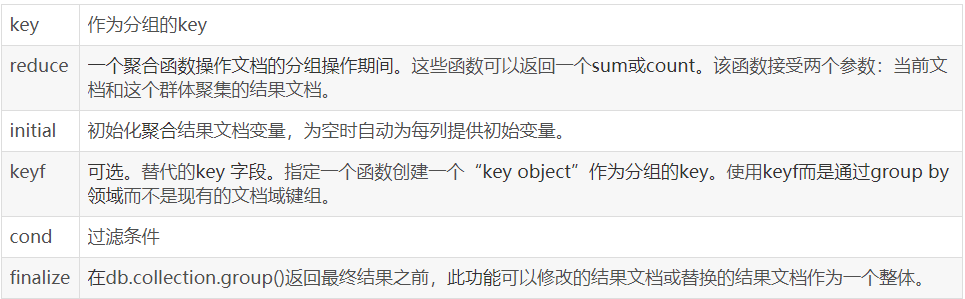


@Override  
 public List<Vehicle> findByPage(Vehicle condition) {  
  
 DBObject fields = new BasicDBObject();  
 DBObject query = new BasicDBObject();  
 query.put("plateColor", "5");  
// query.put("ctime", (new BasicDBObject("$gte", condit)).append("$lt", eTime));  
 List<DBObject> list = new ArrayList<>();  
 List<Vehicle> vehicleList=new ArrayList<>();  
 DBCursor dbCursor=mongoTemplate.getCollection("vehicle").find(query,fields).skip((Integer.*parseInt*(condition.getPageNo())-1)\*10).limit(10);  
 while (dbCursor.hasNext()){  
 DBObject object = dbCursor.next();  
 Vehicle vehicle= JSONObject.*parseObject*(object.toString(),Vehicle.class);  
 vehicleList.add(vehicle)  
 System.*out*.println(object);  
 }  
 return vehicleList;  
 }

<https://www.cnblogs.com/wslook/p/9275861.html>

某个字段的数量聚合





#### Group 、where 、limit

参考：<https://blog.csdn.net/mahongb/article/details/81275011>

BasicDBObject cond = new BasicDBObject();  
cond.put("countDay","2019-01-15");  
DBObject match = new BasicDBObject("$match", cond);  
  
DBObject fields = new BasicDBObject("plateColor", 1);  
DBObject project = new BasicDBObject("$project", fields);  
  
DBObject \_group = new BasicDBObject("plateColor", "$plateColor");  
  
//相当于 group by recommendName，将分组后的数据封装为键值为\_id的键值对  
DBObject groupFields = new BasicDBObject("\_id", \_group);  
  
//相当于求出分组之下的和  
groupFields.put("count", new BasicDBObject("$sum", 1));  
//分组、排序、限制个数  
DBObject group = new BasicDBObject("$group", groupFields);  
  
DBObject limit = new BasicDBObject("$limit", 10);  
  
DBObject sort = new BasicDBObject("$sort", new BasicDBObject("count", -1));  
  
  
DBCollection aa=mongoTemplate.getCollection("vehicle");  
AggregationOutput output = aa.aggregate(match,project,group,sort,limit);



## RabbitMQ

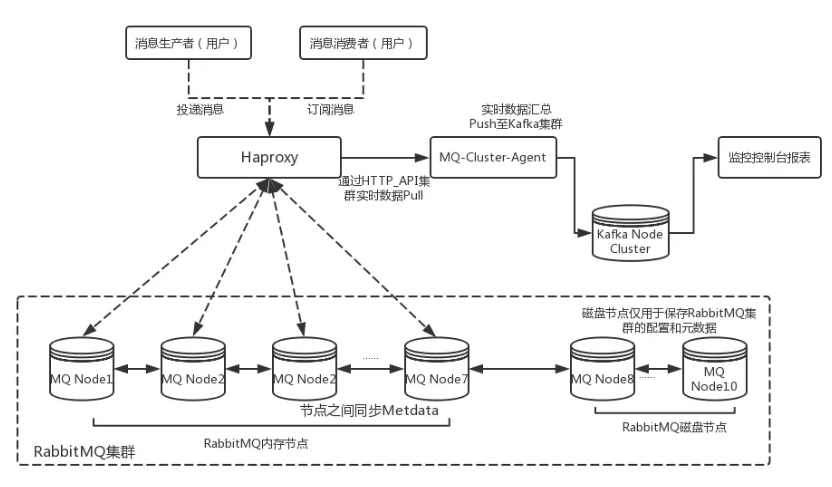
### 原理：



<https://www.cnblogs.com/dwlsxj/p/RabbitMQ.html>

### 集群原理和搭建





### 集群搭建

<http://blog.51cto.com/11134648/2155934>

<https://my.oschina.net/genghz/blog/1840262>

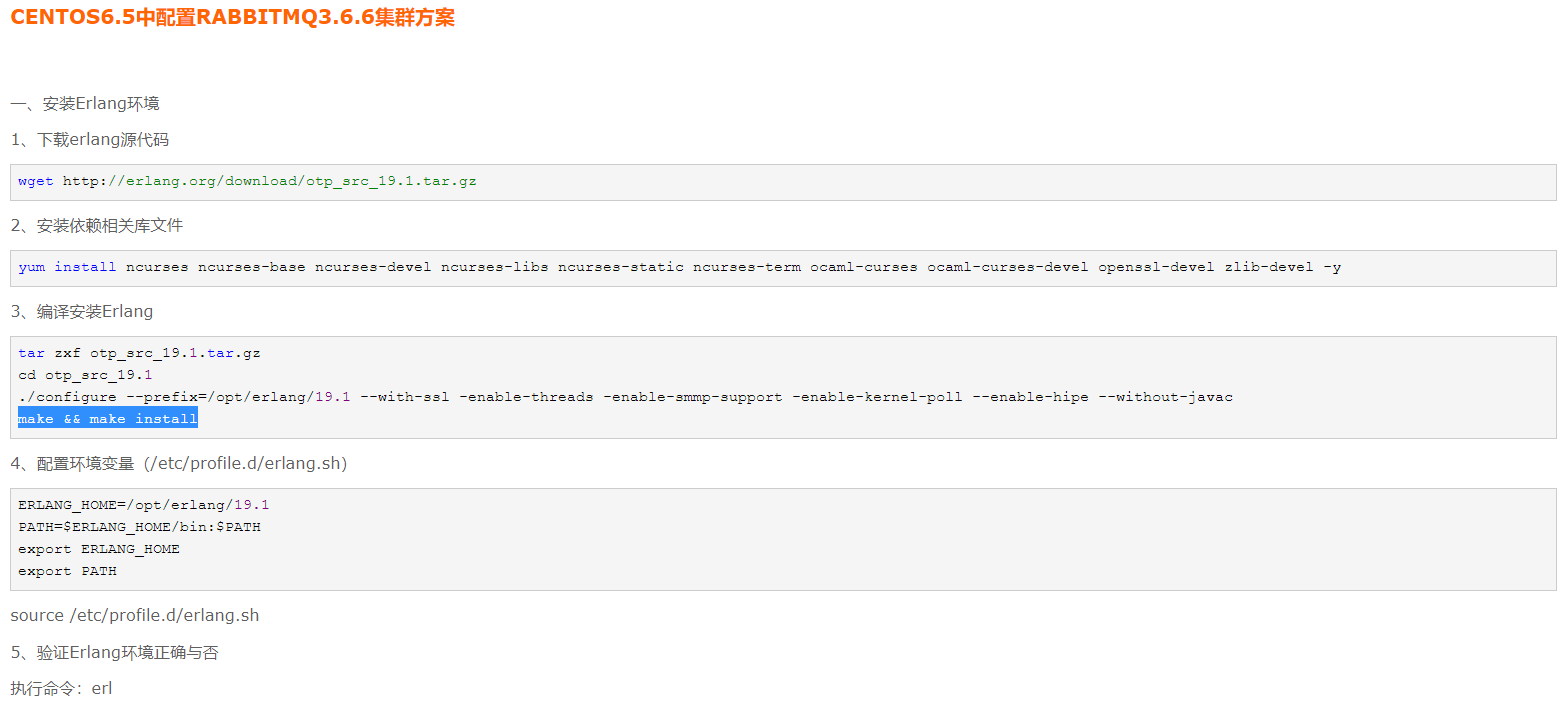
#### linux下安装RabbitMQ3.6.15

<https://www.cnblogs.com/taoge007/p/6208552.html>（安装erlang和rb）

<https://www.cnblogs.com/kevingrace/p/8012792.html>（详细集群搭建）

<http://blog.51cto.com/11134648/2155934> （破坏测试）

<https://blog.csdn.net/liaomin416100569/article/details/78507200> 负载均衡





### 遇到bug

* org.springframework.amqp.AmqpIOException: java.io.IOException //默认端口：5672

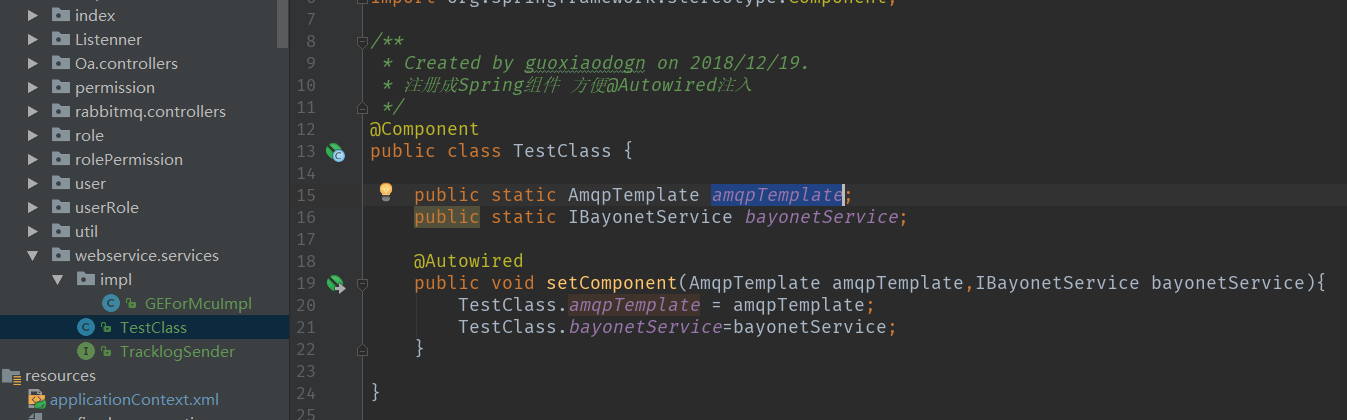
<https://blog.csdn.net/xiaqingxue930914/article/details/81226153>

* .erlang.cookie编辑之后一定要source保存，如果.tar.gz减压包安装，文件在/home目录，如果.rpm安装，位置在/var/lib/rabbitmq/下。
* 一定要按集群搭建步骤进行。
* rabbitMq+Spring注入amqpTemplate为null

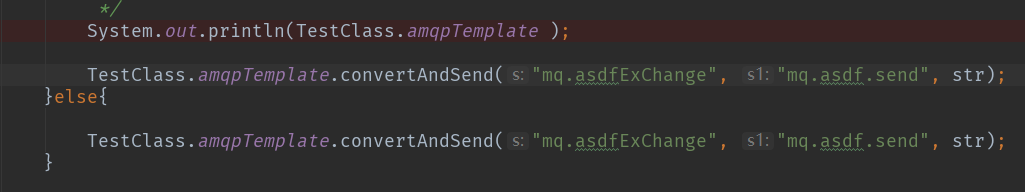
原因：要把注入对象定义成Spring组件，再静态调用

解决方案：<https://blog.csdn.net/RogueFist/article/details/79575665>





调用：



### RabbitMQ集群和消息传递确认机制

<https://blog.csdn.net/everlasting_188/article/details/51098917>

### SSM整合RabbitMQ

<https://blog.csdn.net/LZMLZMLZM268/article/details/82667500>

ack参考：<https://blog.csdn.net/asdfsadfasdfsa/article/details/79671097>

生产者：

消费者：

配置文件：

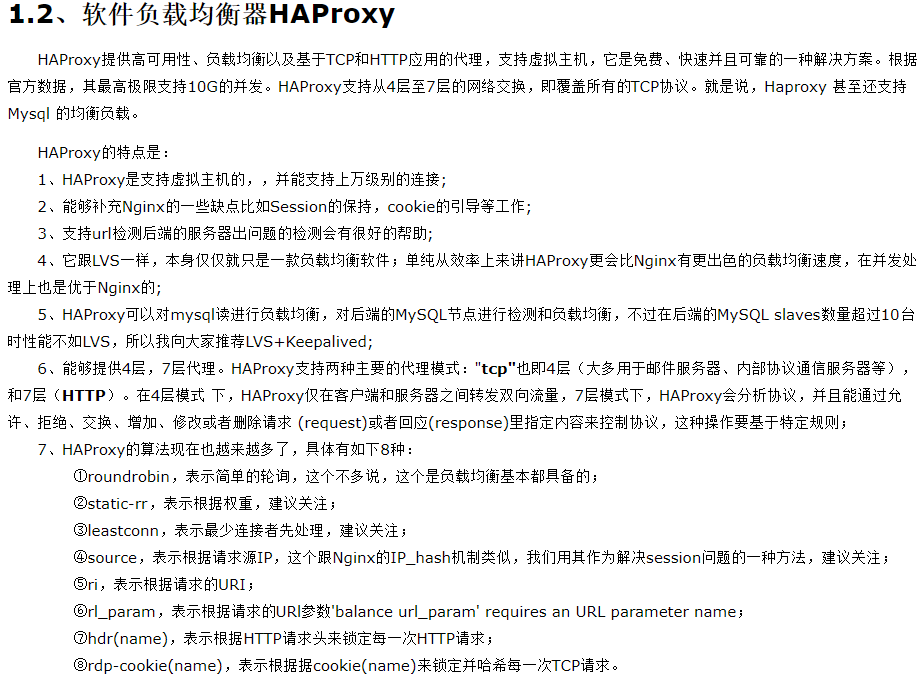
静态获取amqpTemplate: 

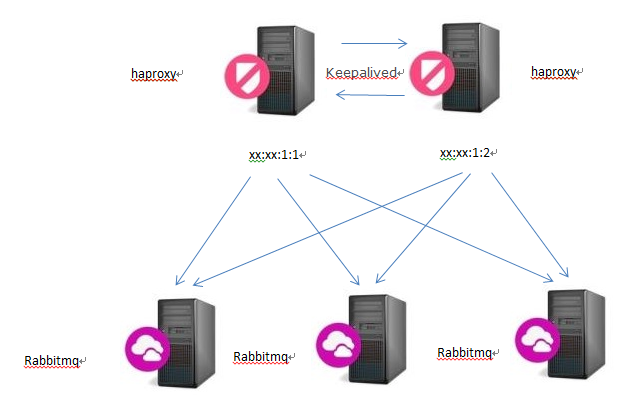
服务器部署文件：先erlang,后安装搭建

### RabbitMQ负载均衡方案：

1. Haproxy+keepalived构建高可用的rabbitmq统一入口，及基本的负载均衡服务。

参考：<https://blog.csdn.net/qq_34021712/article/details/72634167>





1. 基于keepalived+nginx部署强健的高可用7层负载均衡方案

## RocketMQ

### 理论：

#### RocketMQ概念：

##### Topic



Topic表示消息类别，一类消息。第一级消息类型。

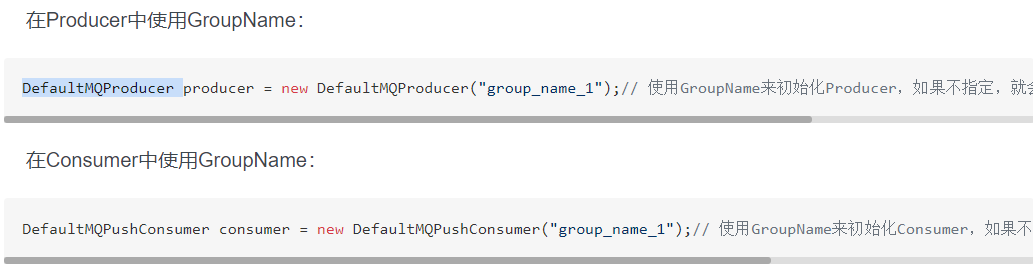
##### Tags



Tags为一类消息（topic）下的二级分类，比如交易信息下的交易创建，交易完成。

##### Group

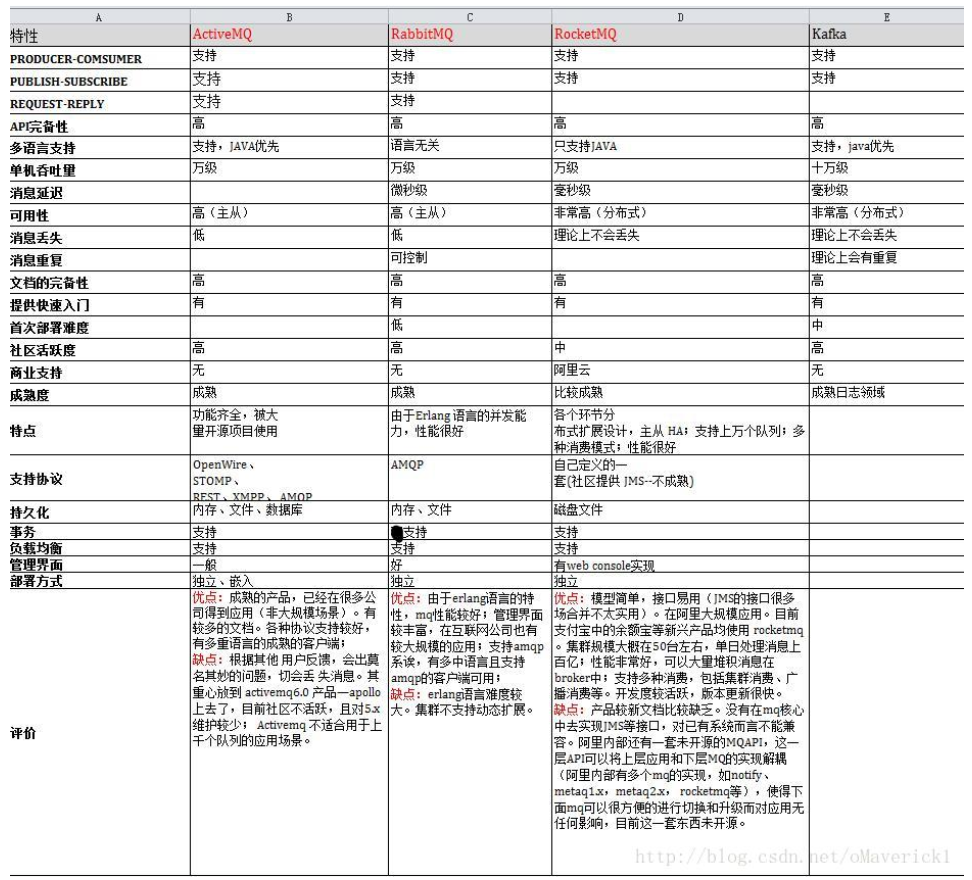
在集群HA状态下，当一个生产者down掉，事务回滚，会联系另外一个生产者处理，保持业务可进行。



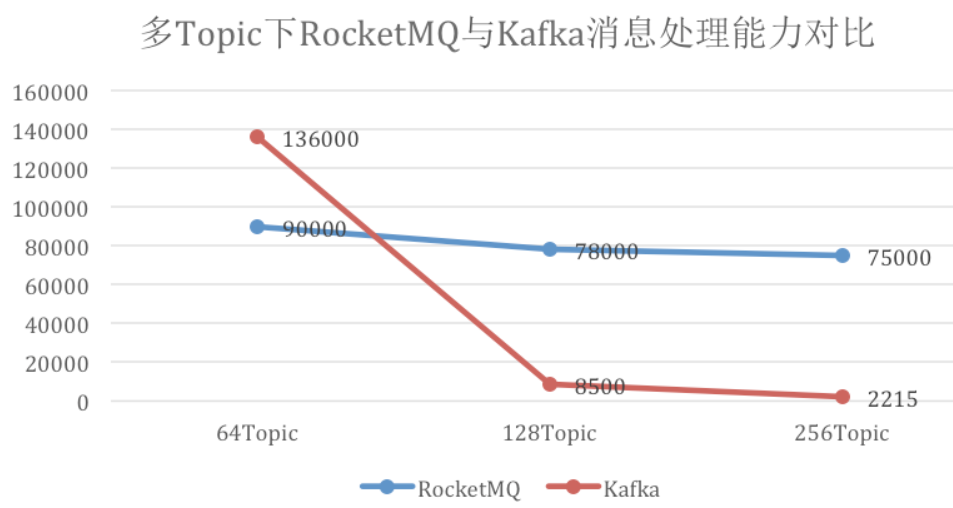
#### Kafka、RabbitMQ、RocketMQ对比

Kafka和rocketmq会自动实现负载均衡，rabbitMQ需要借助ha-proxy和keep-alive实现

<https://blog.csdn.net/jason201710/article/details/78983908>



应用场景：



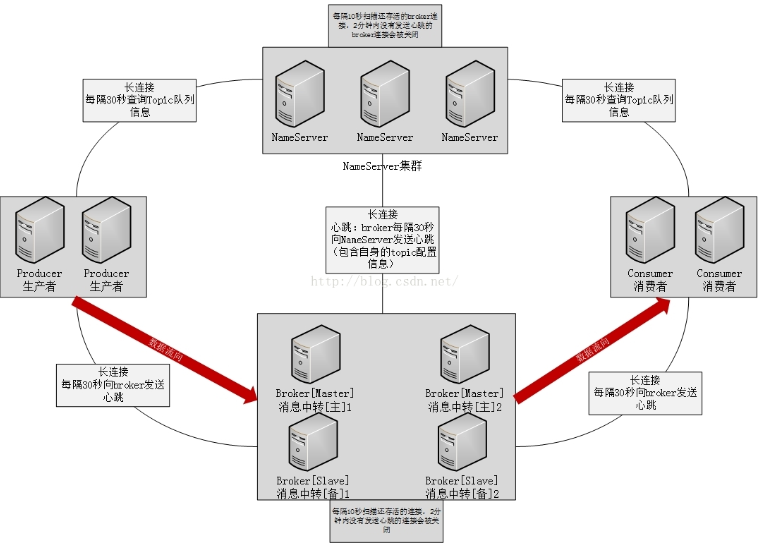
随着topic数量增加，kafka性能明显下降，rocketmq趋于稳定。

因为kafka的每个topic、每个分区都对应一个物理文件，当topic数量增加时，消息分散的落盘策略会导致磁盘IO竞争激烈成为kafka的瓶颈；而rocketmq只保存在同一个物理文件，topic和分区数对于rocketmq只是概念上的划分，所以topic数的增加对rocketmq的性能不会造成太大影响。

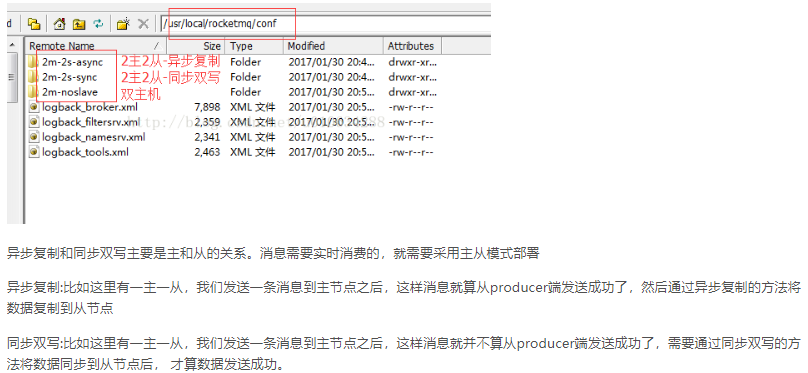
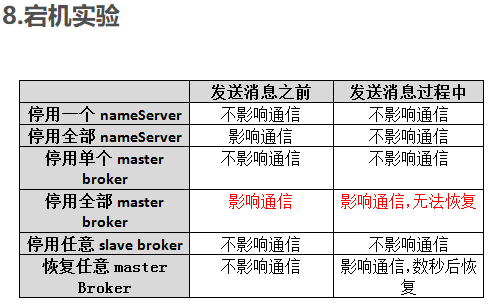
#### RocketMQ核心原理

参考：<https://blog.csdn.net/jeffwxz/article/details/49633383>

<https://blog.csdn.net/zengxiaosen/article/details/78896820>





集群模式：

1. 单个master：风险较高，一旦broker宕机，导致整个服务不可以。
2. 多master：

优点：配置简单，单台master宕机，消息不会丢（异步刷盘丢失少量数据，同步一条不丢），性能很高。

缺点：单台宕机，这台机器上未被消费的消息在机器恢复之前不可被订阅，消息实时性受到影响。

1. 多master多slaver,异步复制模式

Master-Slaver成对存在，HA采用异步复制，主备有短暂消息延时，毫秒级。

优点：即使磁盘损坏，消息丢失非常少，且消息实时性不会受到影响，

缺点：当宕机同时磁盘损坏，会丢失少量消息。

1. 多master多slaver，同步双写模式

多对master-slaver，HA采用同步双写模式，主备都写成功，返回成功。

优点：数据和服务都有备份，可靠性高，消息无延迟。

缺点：性能比异步复制略低10%，单条消息性能开销较大，主备无法自动切换。

Broker和Slaver是通过指定相同的brokerName来配对，master的brokerId必须是0，slaver的brokerId必须大于0。另外，一个mater下可以挂载多个slaver，通过指定不同的slaverId来指定。

### 集群搭建

参考：<https://www.jianshu.com/p/e5b7a0b4803b> （2master-noslaver模式）

<https://blog.csdn.net/qq_32711825/article/details/78579660> （2m-2s-async模式）

#### 配置参数说明













### 集群消费和广播消费

多个消费者是否消费同一消息。

### Java编码

参考：<https://blog.csdn.net/u013278314/article/details/82497132>

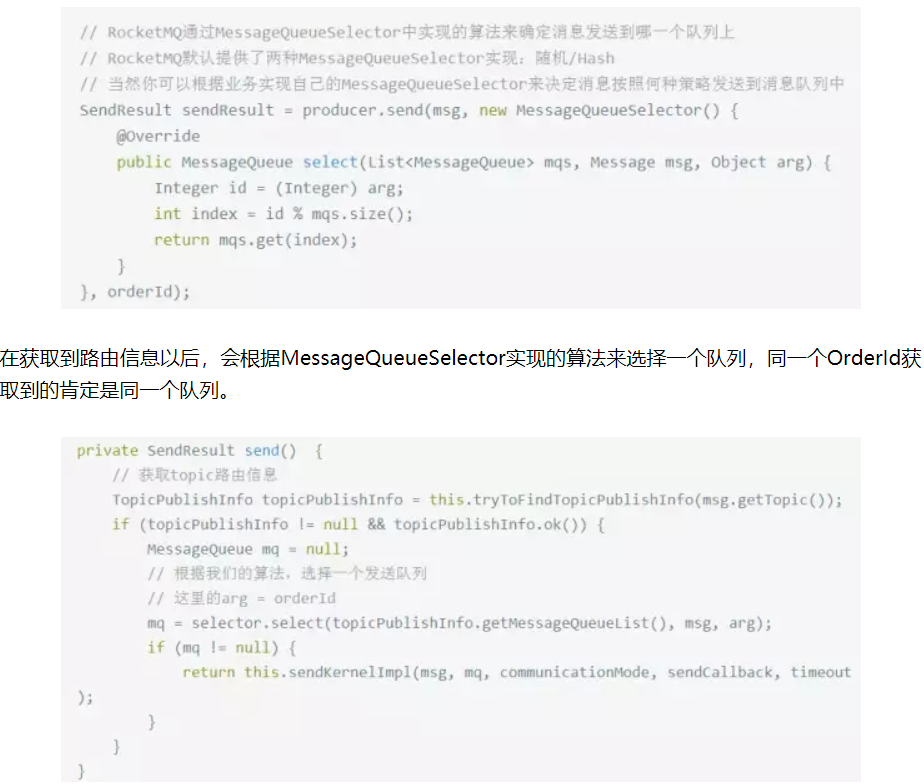




### 消息的顺序消费，重复消费

源码如何实现发送顺序消息：

Rocketmq通过轮询的方式确定消息被发送到哪一个队列（负载均衡策略），比如订单号相同的消息被先后发送到同一队列中。



重复消费问题：

重负是由于网络不可达造成的，重复的几率很小，未来不影响吞吐，将交给业务系统去重。

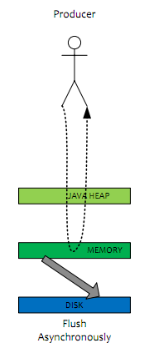
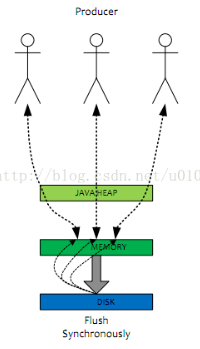
### 普通消息，顺序消息，事物消息，批量消费，消费模式，消息重试，刷盘方式

**大事务 = 小事务 + 异步**

**刷盘方式：**

**刷盘就是把消息持久化，从内存刷到commitlog日志**

**同步刷盘：消息到达内存之后，必须刷到commitlog之后才算成功，返回producer数据已经发送成功。**



**异步刷盘：**消息到达内存之后返回producer数据发送成功，然后再写入commitlog日志。

RocketMQ底层存储依靠consume Quene和commitlog共同完成，consumeQuene指定位置，commitlog存储元数据，包含消息体。

主要有commitlog的存在，即使consumequeue数据丢失，任然可以恢复出来。

Consumequeue，记录消息位置，以便consume快速通过consumequeue找到commitlog中的数据。

## Redis和SpringCache缓存

### Linux下安装redis3.2

<https://blog.csdn.net/lwj_199011/article/details/51423729>

### 使用案例



## SpringCloud

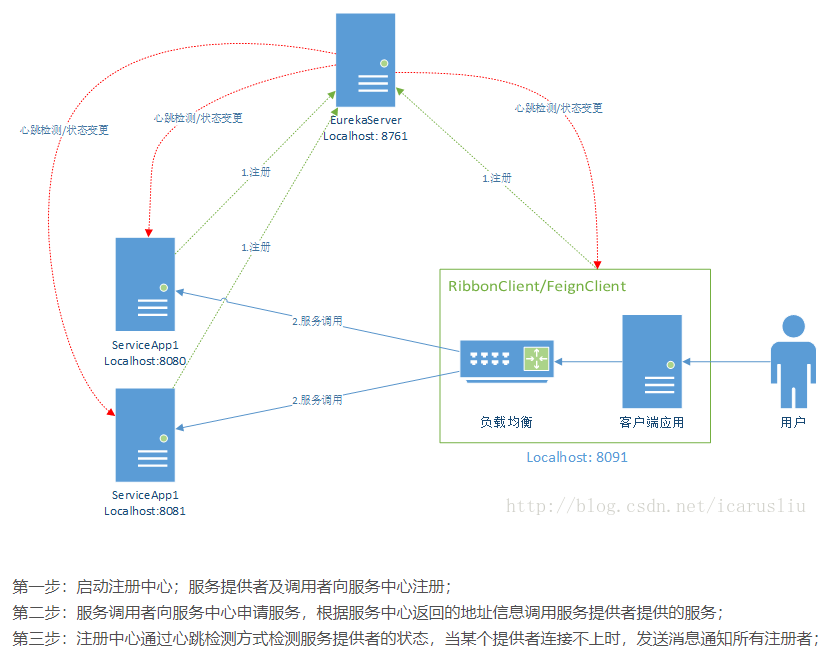
### 理论

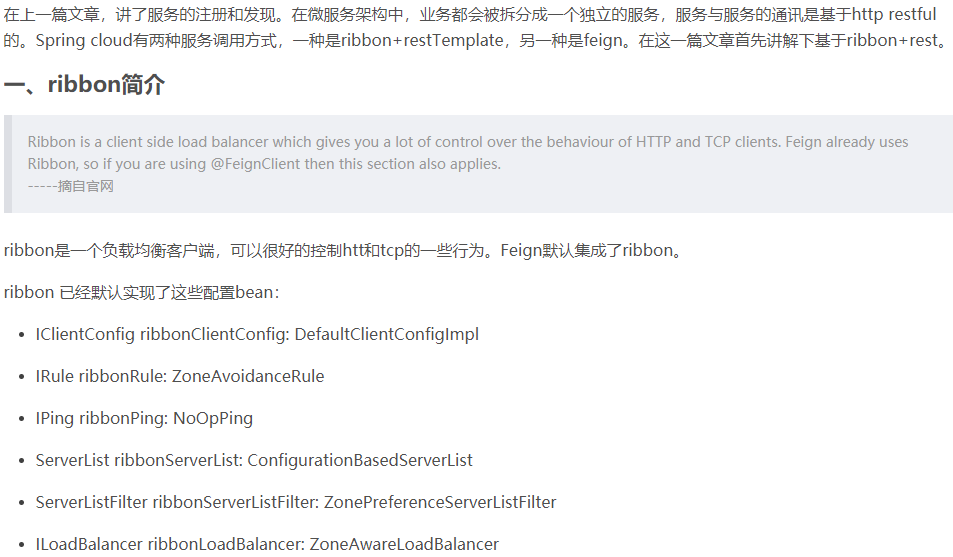
#### Dubbo和Springcloud对比

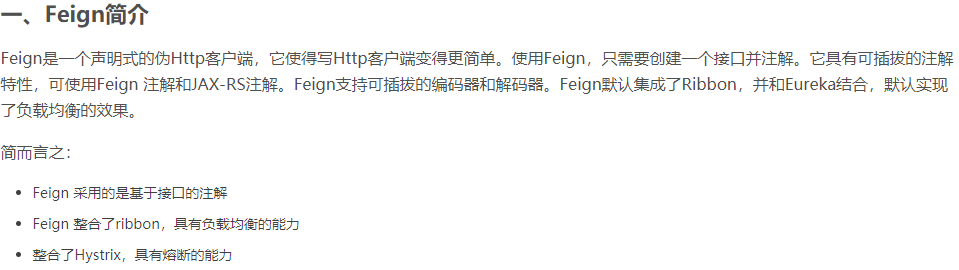




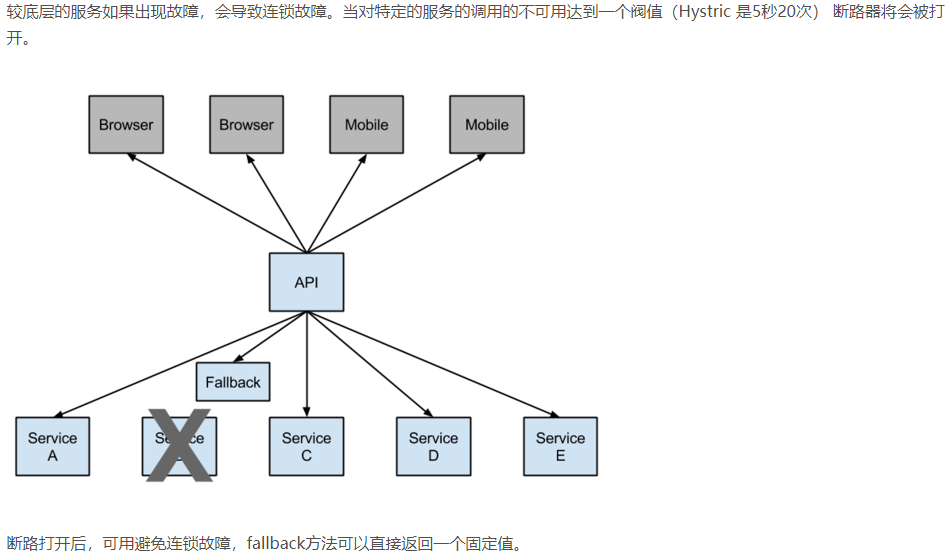
Springcloud与dubbo ：<https://blog.csdn.net/qq_33524158/article/details/80873184>

Springcloud详细教程： <https://blog.csdn.net/forezp/article/details/81040925> 

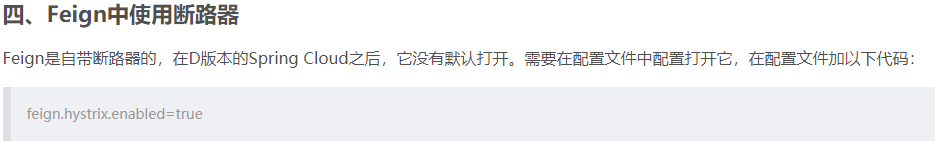




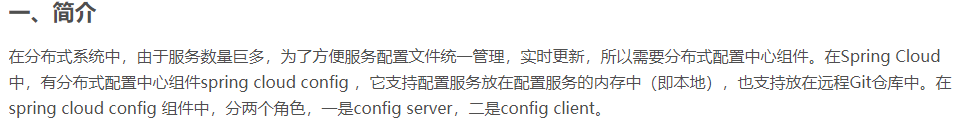
#### 断路器模式：

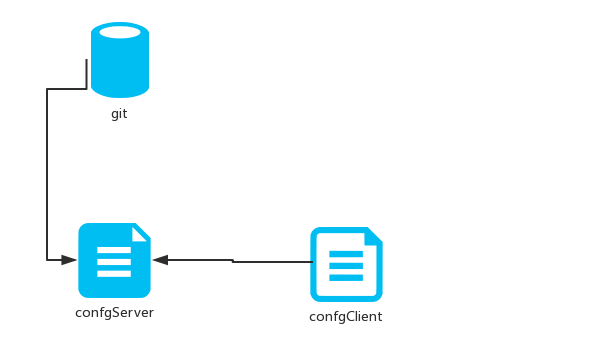






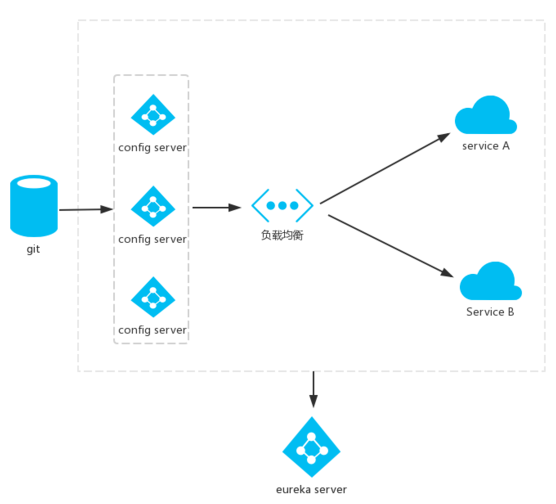
#### 分布式配置中心





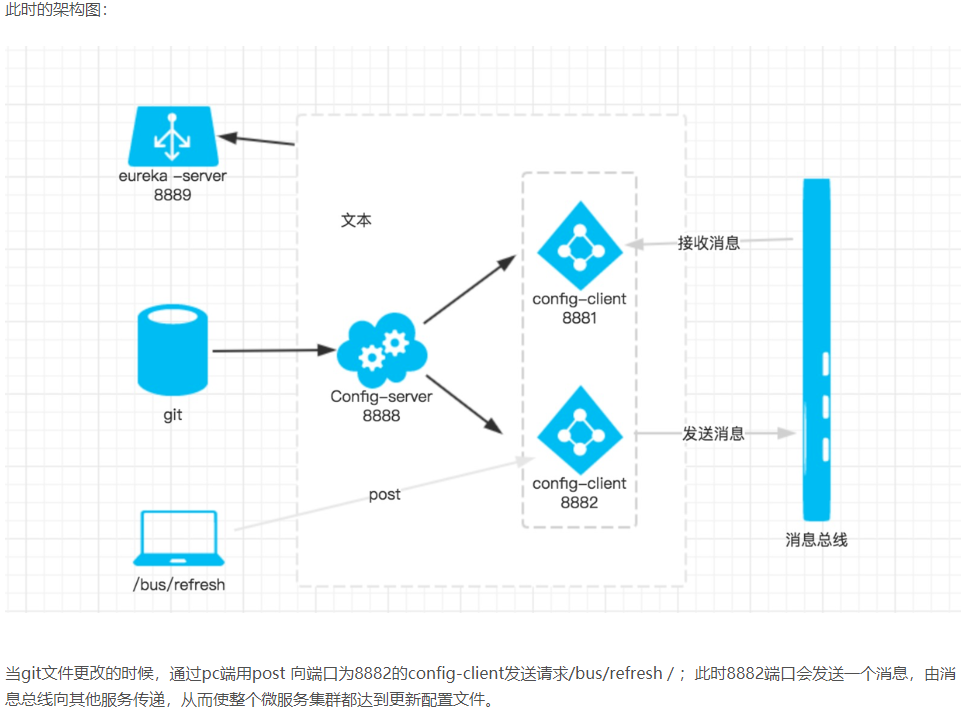
#### 高可用的分布式配置中心(Spring Cloud Config)

将config-server和config-client到注册中心注册作为eureka-client，利用eureka-server的负载均衡特性，当注册多个config-server时候，实现高可用和负载均衡。



#### 消息总线(Spring Cloud Bus)

1广播配置文件 2服务间通讯 3用于监控



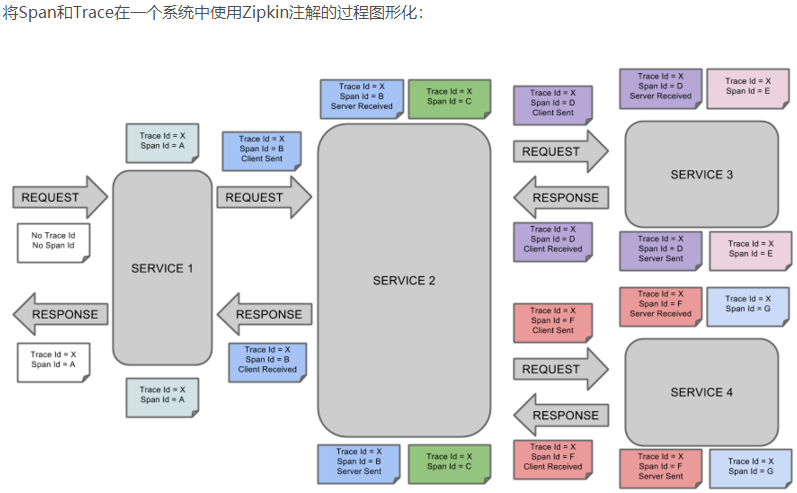
#### 服务链路追踪(Spring Cloud Sleuth)

概念术语：

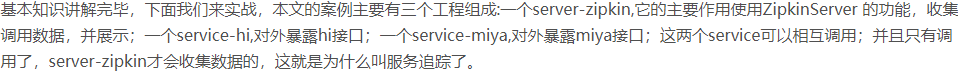
Span：基本工作单元。如client向server发送RPC请求，server对client进行回复。这是一个基本单元。Span通过64位id唯一标识，span还有其他数据信息，比如摘要、时间戳事件、关键值注释(tags)、span的ID、以及进度ID(通常是IP地址)。

span在不断的启动和停止，同时记录了时间信息，当你创建了一个span，你必须在未来的某个时刻停止它。

Trace：许多Span构成的树状结构，例如分布式大数据项目需要创建一个trace。



##### Zipkin-Server(收集调用数据并展示)

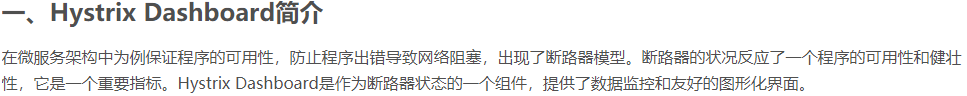


#### 高可用的服务注册中心

服务注册与发现，其中服务注册中心Eureka Server，是一个实例，当成千上万个服务向它注册的时候，它的负载是非常高的，这在生产环境上是不太合适的。

Eureka通过运行多个实例，使其更具有高可用性。事实上，这是它默认的熟性，你需要做的就是给对等的实例一个合法的关联serviceurl。

#### 断路器监控(Hystrix Dashboard)



#### 断路器聚合监控(Hystrix Turbine)

在项目中，多组断路器监控的组合，提供良好的可视化界面。

#### Spring Cloud Gateway

二代网关框架，替代了zuul，网关常见的功能有：路由转发，权限校验，限流控制等作用。

### 遇到问题：

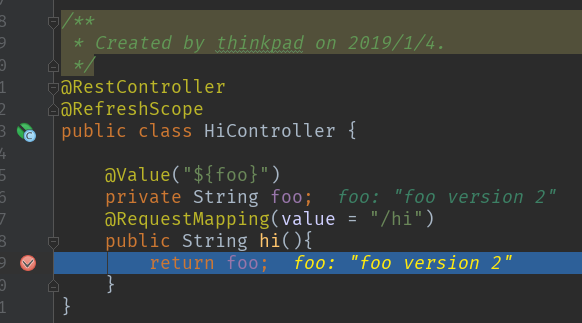
1. [SpringBoot2.X整合SpringCloud出现@EnableFeignClients](mailto:SpringBoot2.X整合SpringCloud出现@EnableFeignClients)找不到类

解决方案：<https://blog.csdn.net/lieyanhaipo/article/details/80480409>

<https://blog.csdn.net/alinyua/article/details/80070890>

1. SpringCloud git仓库取配置的config-server运行报错：取不到配置数据

单独写一个类



### 基于springcloud分布式微服务的分布式事务管理

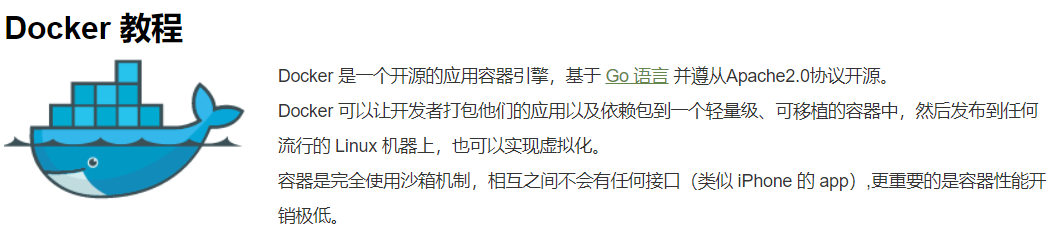
<http://blog.51cto.com/14048134/2312473>

## Nginx

反向代理、正向代理、负载均衡、HTTP服务器(包含动静分离)

## Quartz

## Docker



## Elasticsearch

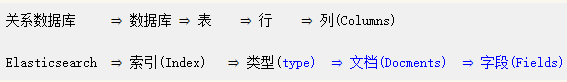
Elasticsearch 是一个分布式可扩展的实时搜索和分析引擎,一个建立在全文搜索引擎 Apache Lucene(TM) 基础上的搜索引擎.

作用：

* 分布式实时文件存储，并将每一个字段都编入索引，使其可以被搜索。
* 实时分析的分布式搜索引擎。
* 可以扩展到上百台服务器，处理PB级别的结构化或非结构化数据。

Elasticsearch是面向文档型数据库，一条数据在这里就是一个文档，用JSON作为文档序列化的格式

存储结构：



## Tomcat配置显示web图片

<https://blog.csdn.net/qq_25302451/article/details/78907188>

## JSON相关

### Json转换

#### Map转换json：

String str= JSONUtils.*toJSONString*(map);

#### Json转map

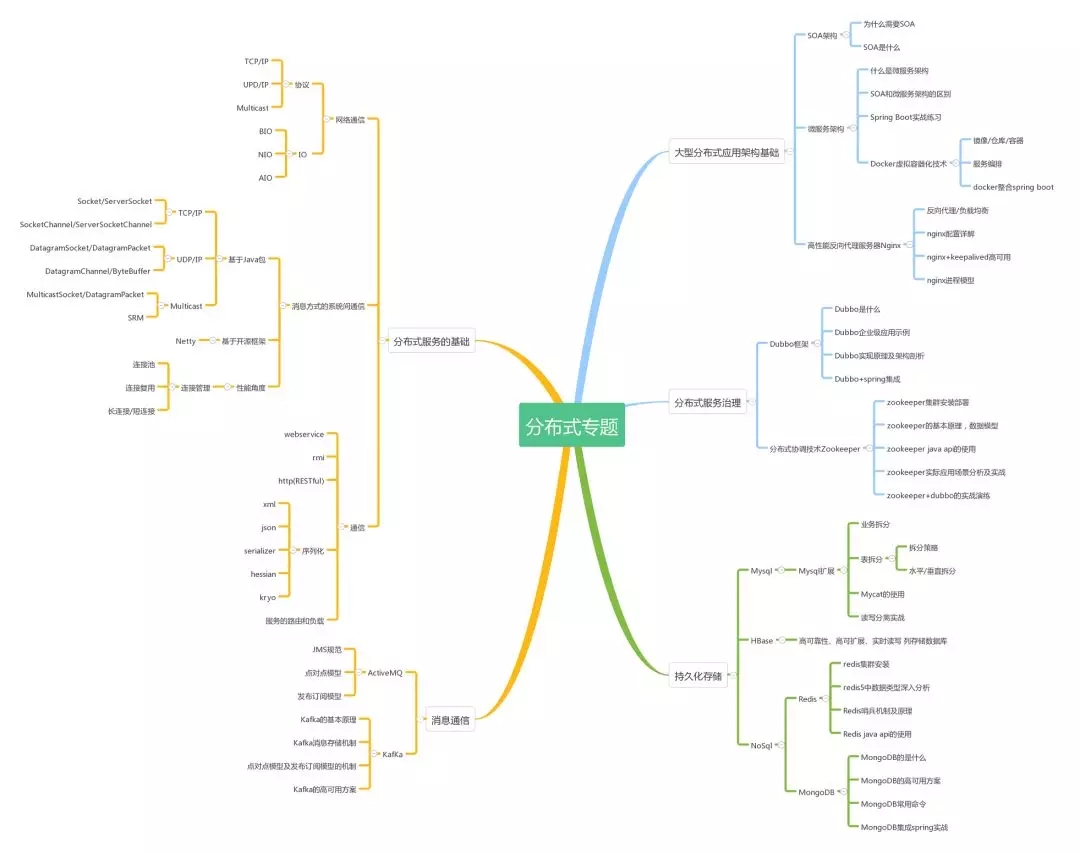
#### DBObject 转换为 jsonObject

@Override  
 public List<Vehicle> findByPage(Vehicle condition) {  
  
 DBObject fields = new BasicDBObject();  
 DBObject query = new BasicDBObject();  
 query.put("plateColor", "5");  
// query.put("ctime", (new BasicDBObject("$gte", condit)).append("$lt", eTime));  
 List<Vehicle> vehicleList=new ArrayList<>();  
 DBCursor dbCursor=mongoTemplate.getCollection("vehicle").find(query,fields).skip((Integer.*parseInt*(condition.getPageNo())-1)\*10).limit(10);  
 while (dbCursor.hasNext()){  
 DBObject object = dbCursor.next();  
 JSONObject jsonObject= (JSONObject) JSONObject.*toJSON*(object);  
 jsonObject.remove("\_id");  
 jsonObject.remove("\_class");  
  
 Vehicle vehicle= JSONObject.*parseObject*(jsonObject.toString(),Vehicle.class);  
 vehicleList.add(vehicle);  
 }  
 return vehicleList;  
 }

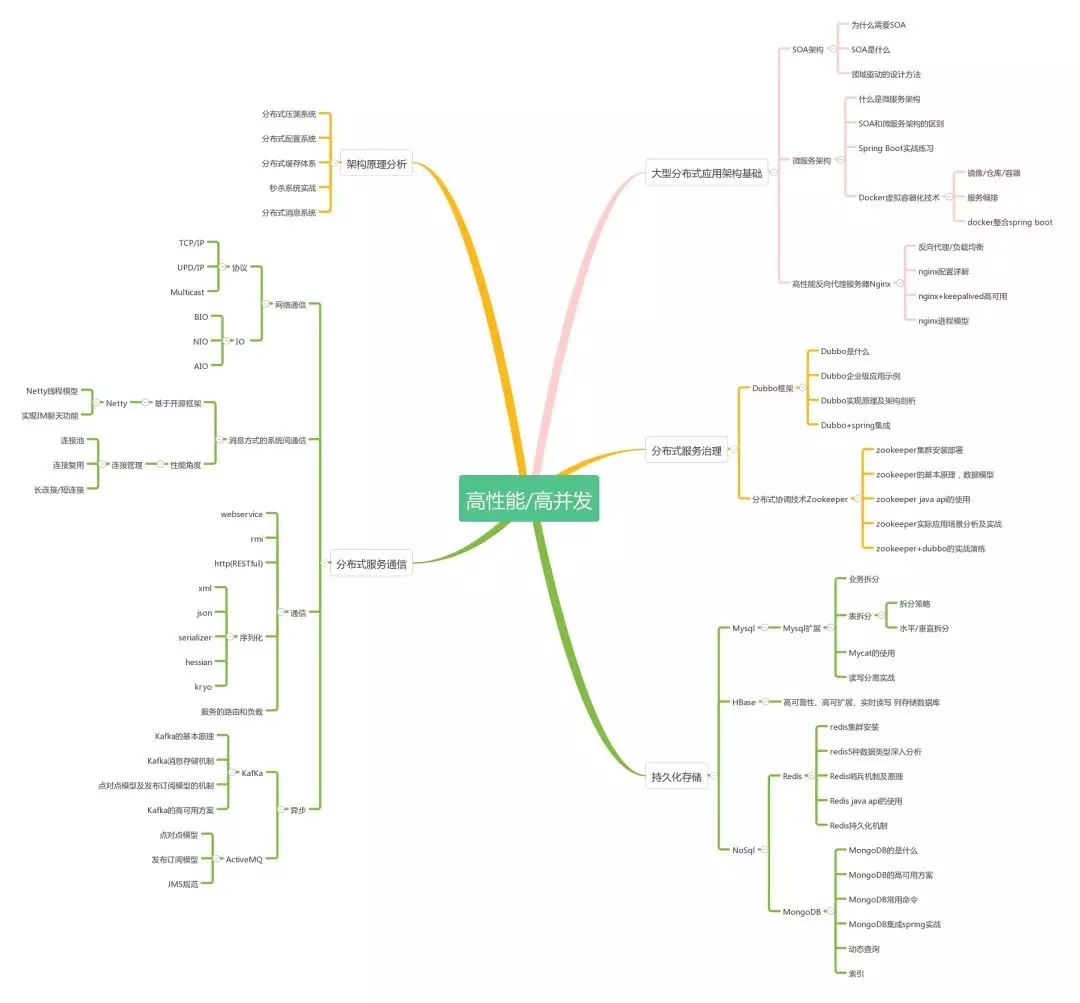
#### Json数组取值



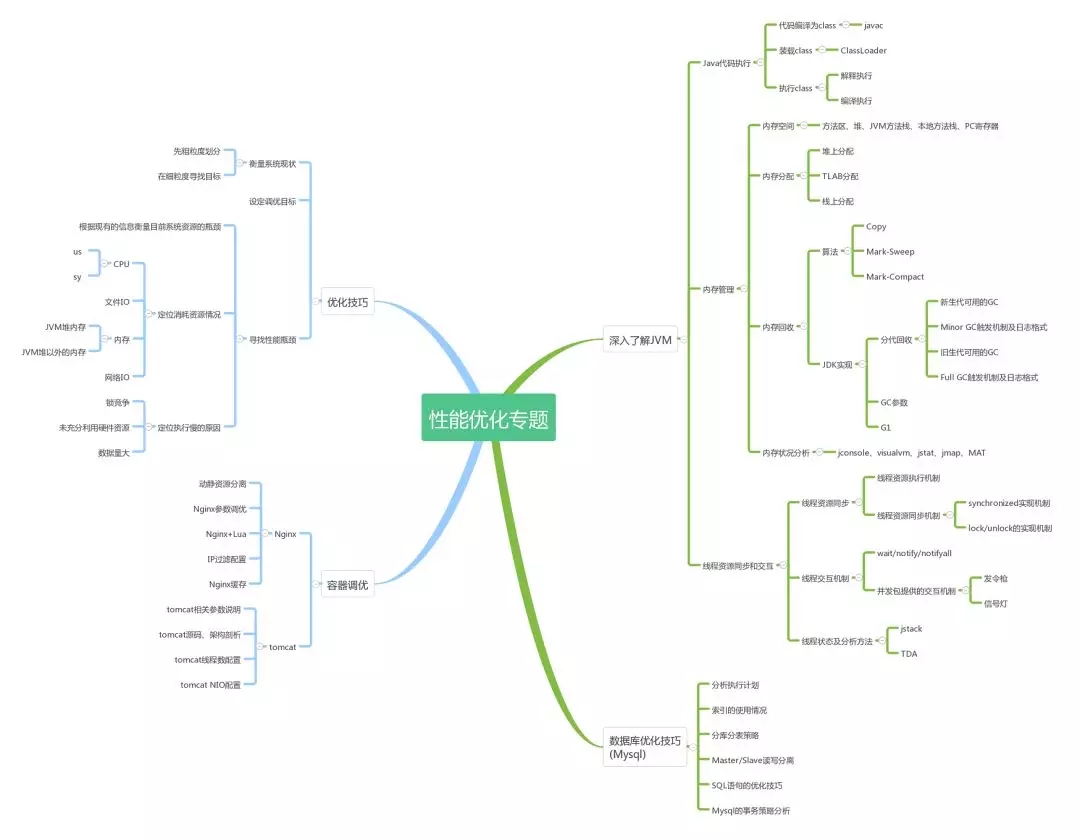
## 分布式专题



## 高性能、高并发



## 性能优化专题



## 版本

Mongodb必须Spring4.2+ rabbitMQ 必须Spring4.2.3+

Spring4.2+ 必须 cxf 3.0+