# Dubbo&&zookeeper

## Dubbo介绍



**节点角色说明：**

* **Provider:** 暴露服务的服务提供方。
* **Consumer:** 调用远程服务的服务消费方。
* **Registry:** 服务注册与发现的注册中心。
* **Monitor:** 统计服务的调用次调和调用时间的监控中心。
* **Container:** 服务运行容器。

**调用关系说明：**

* 0. 服务容器负责启动，加载，运行服务提供者。
* 1. 服务提供者在启动时，向注册中心注册自己提供的服务。
* 2. 服务消费者在启动时，向注册中心订阅自己所需的服务。
* 3. 注册中心返回服务提供者地址列表给消费者，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。
* 4. 服务消费者，从提供者地址列表中，基于软负载均衡算法，选一台提供者进行调用，如果调用失败，再选另一台调用。
* 5. 服务消费者和提供者，在内存中累计调用次数和调用时间，定时每分钟发送一次统计数据到监控中心。

## Dubbo使用方法

1. 在提供方和消费方都添加dubbo和zookeeper的依赖

dubbo中依赖了spring，但是版本比较低，而且项目中本来就有spring的依赖，所以为了统一版本，在导入dubbo的依赖的时候剔除掉对spring的传递依赖。

|  |
| --- |
| <!-- dubbo相关 -->  <dependency>  <groupId>com.alibaba</groupId>  <artifactId>dubbo</artifactId>  <exclusions>  <exclusion>  <groupId>org.springframework</groupId>  <artifactId>spring</artifactId>  </exclusion>  <exclusion>  <groupId>org.jboss.netty</groupId>  <artifactId>netty</artifactId>  </exclusion>  </exclusions>  </dependency>  <dependency>  <groupId>org.apache.zookeeper</groupId>  <artifactId>zookeeper</artifactId>  </dependency>  <dependency>  <groupId>com.github.sgroschupf</groupId>  <artifactId>zkclient</artifactId>  </dependency> |

1. 在spring配置文件中添加dubbo的约束，并注册服务，引用服务

提供方：

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans xmlns=*"http://www.springframework.org/schema/beans"*  xmlns:context=*"http://www.springframework.org/schema/context"* xmlns:p=*"http://www.springframework.org/schema/p"*  xmlns:aop=*"http://www.springframework.org/schema/aop"* xmlns:tx=*"http://www.springframework.org/schema/tx"*  xmlns:dubbo=*"http://code.alibabatech.com/schema/dubbo"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.2.xsd*  *http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.2.xsd*  *http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-4.2.xsd http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-4.2.xsd*  *http://code.alibabatech.com/schema/dubbo http://code.alibabatech.com/schema/dubbo/dubbo.xsd*  *http://www.springframework.org/schema/util http://www.springframework.org/schema/util/spring-util-4.2.xsd"*>  <context:component-scan base-package=*"cn.e3mall.service"*/>    <!-- dubbo发布服务 -->  <!-- 提供方应用信息，用于计算依赖关系 -->  <dubbo:application name=*"e3-manager"*/>  <dubbo:registry protocol=*"zookeeper"* address=*"192.168.12.3:2181"*/>  <!-- 用dubbo协议在20880端口暴露服务 -->  <dubbo:protocol name=*"dubbo"* port=*"20880"*/>  <!-- 声明需要暴露的服务接口 -->  <dubbo:service interface=*"cn.e3mall.service.ItemService"* ref=*"itemServiceImpl"*></dubbo:service>    </beans> |

消费方：

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans xmlns=*"http://www.springframework.org/schema/beans"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns:p=*"http://www.springframework.org/schema/p"*  xmlns:context=*"http://www.springframework.org/schema/context"*  xmlns:mvc=*"http://www.springframework.org/schema/mvc"*  xmlns:dubbo=*"http://code.alibabatech.com/schema/dubbo"*  xsi:schemaLocation=*"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.2.xsd*  *http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-4.2.xsd*  *http://code.alibabatech.com/schema/dubbo http://code.alibabatech.com/schema/dubbo/dubbo.xsd*  *http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.2.xsd"*>  <context:component-scan base-package=*"cn.e3mall.controller"* />  <mvc:annotation-driven />  <bean  class=*"org.springframework.web.servlet.view.InternalResourceViewResolver"*>  <property name=*"prefix"* value=*"/WEB-INF/jsp/"* />  <property name=*"suffix"* value=*".jsp"* />  </bean>    <!-- 引用dobbo服务 -->  <dubbo:application name=*"e3-manager-web"*/>  <dubbo:registry protocol=*"zookeeper"* address=*":2181"*/>  <dubbo:reference interface=*"cn.e3mall.service.ItemService"* id=*"itemService"*></dubbo:reference>    </beans> |

## zookeeper注册中心

Zookeeper是Apacahe Hadoop的子项目，是一个树型的目录服务。dubbo官方推荐使用zookeeper注册中心。zookeeper就是一个java项目。

zookeeper作为dubbo的注册中心，相当于服务目录，提供者和消费者只在启动的时候与注册中心交互。zookeeper不转发请求，压力较小。

**Linux上安装zookeeper：**

1、在zookeeper根目录中创建data文件夹

2、修改zoo\_sample.cfg改名为zoo.cfg，并修其中的属性：

dataDir=/usr/local/zookeeper/data

启动：./zkServer.sh start

关闭：./zkServer.sh stop

查看状态：./zkServer.sh status

**注意：需要关闭防火墙。（centos7之前）**

service iptables stop

永久关闭修改配置开机不启动防火墙：

chkconfig iptables off

如果不能成功启动zookeeper，需要删除data目录下的zookeeper\_server.pid文件。

## Dubbo监控中心

将dubbo-admin.war放入到linux的Tomcat中，启动Tomcat

访问：<http://192.168.0.102/dubbo-admin>

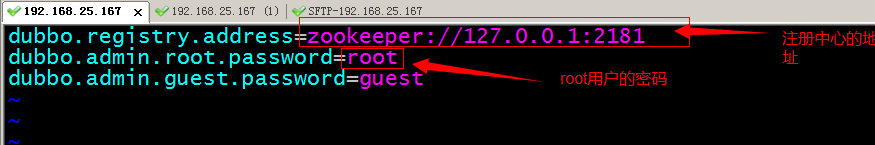
用户名：root

密码：root

如果监控中心和注册中心在同一台服务器上，可以不需要任何配置。

如果不在同一台服务器，需要修改配置文件：

/root/apache-tomcat-7.0.47/webapps/dubbo-admin/WEB-INF/dubbo.properties



## soa下debug及超时时长设置

debug的时候，不指示代码执行到哪一行。（未解决）

超时时长设置：在注册服务（发布方）的代码中加：timeout属性，单位ms

|  |
| --- |
| <dubbo:service interface=*"cn.e3mall.service.ItemService"* ref=*"itemServiceImpl"* timeout=*"600000"*/> |

# PageHeper

由于使用mybatis逆向工程生成的mapping，不能分页，所以在这里使用pageHeper插件。该插件目前支持Oracle,Mysql,MariaDB,SQLite,Hsqldb,PostgreSQL六种数据库分页。

**使用方法：**

1. 导入依赖

官方提供的代码对逆向工程支持的不好，使用参考资料中的pagehelper-fix。

|  |
| --- |
| <dependency>  <groupId>com.github.pagehelper</groupId>  <artifactId>pagehelper</artifactId>  <version>${pagehelper.version}</version>  </dependency> |

1. 在Mybatis配置xml中配置拦截器插件

|  |
| --- |
| <plugins>  <!-- com.github.pagehelper为PageHelper类所在包名 -->  <plugin interceptor="com.github.pagehelper.PageHelper">  <!-- 设置数据库类型 Oracle,Mysql,MariaDB,SQLite,Hsqldb,PostgreSQL六种数据库-->  <property name="dialect" value="mysql"/>  </plugin>  </plugins> |

1. 代码使用

|  |
| --- |
| **public** **class** PageHelperTest {  @Test  **public** **void** testPageHelper() {  ApplicationContext applicationContext = **new** ClassPathXmlApplicationContext("classpath:spring/applicationContext-dao.xml");  TbItemMapper itemMapper = applicationContext.getBean(TbItemMapper.**class**);    //执行sql语句前设置分页信息  PageHelper.*startPage*(1, 10);    //执行查询  TbItemExample example = **new** TbItemExample();  List<TbItem> list = itemMapper.selectByExample(example);    //取分页信息  PageInfo<TbItem> pageInfo = **new** PageInfo<>(list);  System.***out***.println(pageInfo.getTotal());  System.***out***.println(pageInfo.getPages());  System.***out***.println(list);  }    } |

# nginx

## 简介

Nginx是一款高性能的http 服务器/反向代理服务器及电子邮件（IMAP/POP3）代理服务器。官方测试nginx能够支支撑5万并发链接，并且cpu、内存等资源消耗却非常低，运行非常稳定。

Nginx是用C语言开发的。由于linux不同版本下nginx不同，所以nginx的linux版只有源码，需要我们自己在服务器上编译，。编译前需要第三方的开发包先装好。

**应用场景：**

1. http服务器。Nginx是一个http服务可以独立提供http服务。可以做网页静态服务器。
2. 虚拟主机。可以实现在一台服务器虚拟出多个网站。例如个人网站使用的虚拟主机。
3. 反向代理，负载均衡。当网站的访问量达到一定程度后，单台服务器不能满足用户的请求时，需要用多台服务器集群可以使用nginx做反向代理。并且多台服务器可以平均分担负载，不会因为某台服务器负载高宕机而某台服务器闲置的情况。

## nginx安装

### 要求的安装环境

（1）需要安装gcc的环境。yum install gcc-c++

（2）第三方的开发包。

* PCRE

PCRE(Perl Compatible Regular Expressions)是一个Perl库，包括 perl 兼容的正则表达式库。nginx的http模块使用pcre来解析正则表达式，所以需要在linux上安装pcre库。

**yum install -y pcre pcre-devel**

注：pcre-devel是使用pcre开发的一个二次开发库。nginx也需要此库。

* zlib

zlib库提供了很多种压缩和解压缩的方式，nginx使用zlib对http包的内容进行gzip，所以需要在linux上安装zlib库。

**yum install -y zlib zlib-devel**

* openssl

OpenSSL 是一个强大的安全套接字层密码库，囊括主要的密码算法、常用的密钥和证书封装管理功能及SSL协议，并提供丰富的应用程序供测试或其它目的使用。

nginx不仅支持http协议，还支持https（即在ssl协议上传输http），所以需要在linux安装openssl库。

**yum install -y openssl openssl-devel**

### 安装步骤

第一步：把nginx的源码包上传到linux系统

第二步：解压缩

[root@localhost ~]# tar zxf nginx-1.8.0.tar.gz

第三步：使用configure命令创建一makeFile文件。

./configure \

--prefix=/usr/local/nginx \

--pid-path=/var/run/nginx/nginx.pid \

--lock-path=/var/lock/nginx.lock \

--error-log-path=/var/log/nginx/error.log \

--http-log-path=/var/log/nginx/access.log \

--with-http\_gzip\_static\_module \

--http-client-body-temp-path=/var/temp/nginx/client \

--http-proxy-temp-path=/var/temp/nginx/proxy \

--http-fastcgi-temp-path=/var/temp/nginx/fastcgi \

--http-uwsgi-temp-path=/var/temp/nginx/uwsgi \

--http-scgi-temp-path=/var/temp/nginx/scgi

**注意：启动nginx之前，上边将临时文件目录指定为/var/temp/nginx，需要在/var下创建temp及nginx目录**

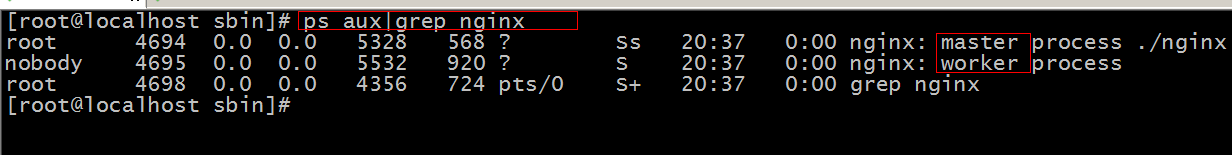
[root@localhost sbin]# mkdir /var/temp/nginx/client -p

第四步：make

第五步：make install

### 启动nginx

进入sbin目录

[root@localhost sbin]# ./nginx   


关闭nginx：

[root@localhost sbin]# ./nginx -s stop

推荐使用：

[root@localhost sbin]# ./nginx -s quit

重启nginx：

（1）先关闭后启动。

（2）刷新配置文件：

[root@localhost sbin]# ./nginx -s reload

查看进程：ps aux|grep

aux：所有用户终端

必须启动两个进程：master、worker才可以用

### 4、访问nginx



默认是80端口。

## 配置虚拟主机

就是在一台服务器启动多个网站。

如何区分不同的网站：

（1）域名不同

（2）端口不同

### 通过端口区分不同虚拟主机

Nginx的配置文件：

/usr/local/nginx/conf/nginx.conf

|  |
| --- |
| #user nobody;  worker\_processes 1;  #error\_log logs/error.log;  #error\_log logs/error.log notice;  #error\_log logs/error.log info;  #pid logs/nginx.pid;  events {  worker\_connections 1024;  }  http {  include mime.types;  default\_type application/octet-stream;  #log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '  # '$status $body\_bytes\_sent "$http\_referer" '  # '"$http\_user\_agent" "$http\_x\_forwarded\_for"';  #access\_log logs/access.log main;  sendfile on;  #tcp\_nopush on;  #keepalive\_timeout 0;  keepalive\_timeout 65;  #gzip on;  一个server节点就是一个虚拟主机  server {  listen 80;  server\_name localhost;  #charset koi8-r;  #access\_log logs/host.access.log main;  Html是nginx安装目录下的html目录  location / {  root html;  index index.html index.htm;  }  }  }  server {  listen 81;  server\_name localhost;  #charset koi8-r;  #access\_log logs/host.access.log main;  location / {  root html-81;  index index.html index.htm;  }  }  } |

重新加载配置文件

[root@localhost nginx]# sbin/nginx -s reload

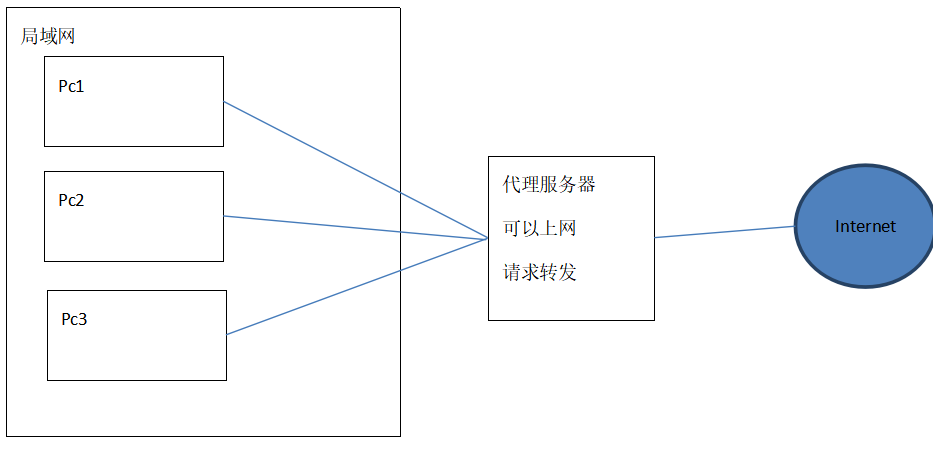
### 通过域名区分不同虚拟主机

|  |
| --- |
| #user nobody;  worker\_processes 1;  #error\_log logs/error.log;  #error\_log logs/error.log notice;  #error\_log logs/error.log info;  #pid logs/nginx.pid;  events {  worker\_connections 1024;  }  http {  include mime.types;  default\_type application/octet-stream;  #log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '  # '$status $body\_bytes\_sent "$http\_referer" '  # '"$http\_user\_agent" "$http\_x\_forwarded\_for"';  #access\_log logs/access.log main;  sendfile on;  #tcp\_nopush on;  #keepalive\_timeout 0;  keepalive\_timeout 65;  #gzip on;  server {  listen 80;  server\_name localhost;  #charset koi8-r;  #access\_log logs/host.access.log main;  location / {  root html;  index index.html index.htm;  }  }  server {  listen 81;  server\_name localhost;  #charset koi8-r;  #access\_log logs/host.access.log main;  location / {  root html-81;  index index.html index.htm;  }  }  server {  listen 80;  server\_name www.taobao.com;  #charset koi8-r;  #access\_log logs/host.access.log main;  location / {  root html-taobao;  index index.html index.htm;  }  }  server {  listen 80;  server\_name www.baidu.com;  #charset koi8-r;  #access\_log logs/host.access.log main;  location / {  root html-baidu;  index index.html index.htm;  }  }  } |

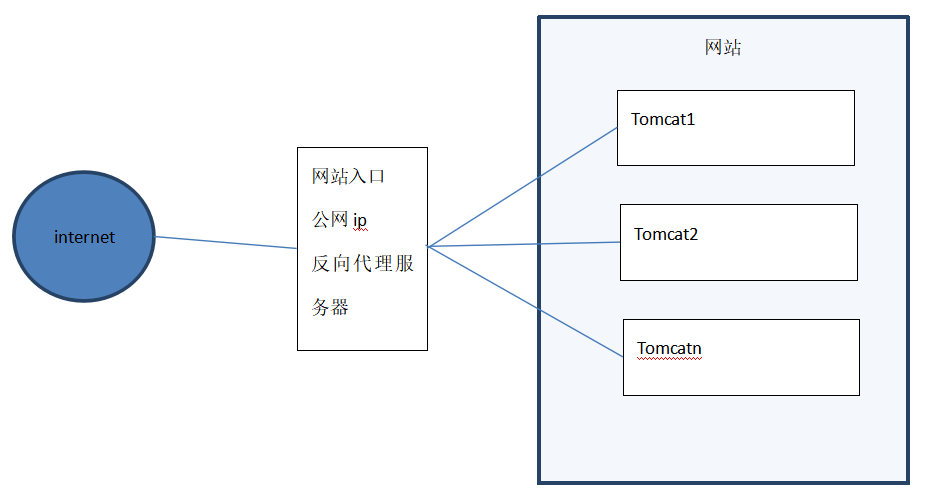
## 反向代理

### 什么是反向代理

正向代理



反向代理：



反向代理服务器决定哪台服务器提供服务。

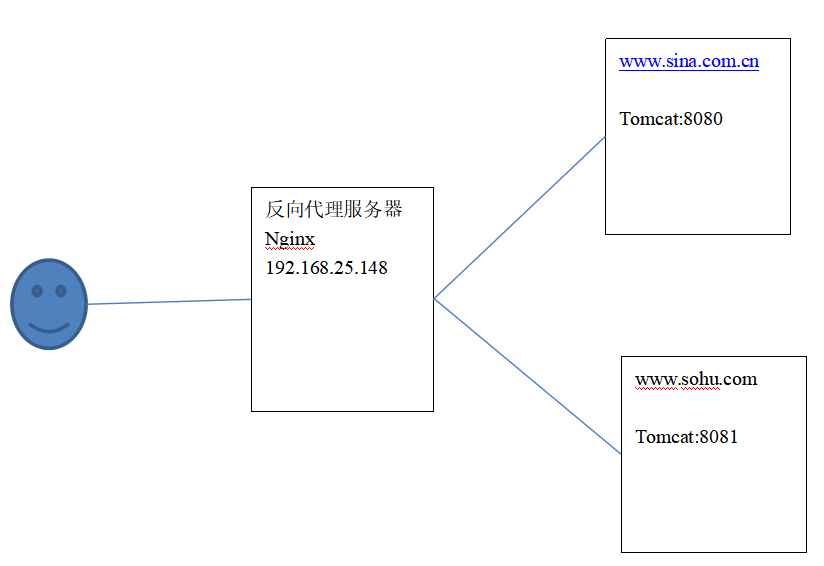
返回代理服务器不提供服务器。也是请求的转发。

### nginx实现反向代理

两个域名指向同一台nginx服务器，用户访问不同的域名显示不同的网页内容。

两个域名是www.sian.com.cn和www.sohu.com

nginx服务器使用虚拟机192.168.101.3



第一步：安装两个tomcat，分别运行在8080和8081端口。

第二步：启动两个tomcat。

第三步：反向代理服务器的配置

|  |
| --- |
| upstream tomcat1 {  server 192.168.25.148:8080;  }  server {  listen 80;  server\_name www.sina.com.cn;  #charset koi8-r;  #access\_log logs/host.access.log main;  location / {  proxy\_pass http://tomcat1;  index index.html index.htm;  }  }  upstream tomcat2 {  server 192.168.25.148:8081;  }  server {  listen 80;  server\_name www.sohu.com;  #charset koi8-r;  #access\_log logs/host.access.log main;  location / {  proxy\_pass http://tomcat2;  index index.html index.htm;  }  } |

第四步：nginx重新加载配置文件

第五步：配置域名

在hosts文件中添加域名和ip的映射关系

192.168.25.148 www.sina.com.cn

192.168.25.148 www.sohu.com

## 负载均衡

如果一个服务由多条服务器提供，需要把负载分配到不同的服务器处理，需要负载均衡。

upstream tomcat2 {

server 192.168.25.148:8081;

server 192.168.25.148:8082;

}

可以根据服务器的实际情况调整服务器权重。权重越高分配的请求越多，权重越低，请求越少。默认是都是1

|  |
| --- |
| upstream tomcat2 {  server 192.168.25.148:8081;  server 192.168.25.148:8082 weight=2;  } |

## ngin高可用

要实现nginx的高可用，需要实现备份机。

### 1、什么是负载均衡高可用

nginx作为负载均衡器，所有请求都到了nginx，可见nginx处于非常重点的位置，如果nginx服务器宕机后端web服务将无法提供服务，影响严重。

为了屏蔽负载均衡服务器的宕机，需要建立一个备份机。主服务器和备份机上都运行高可用（High Availability）监控程序，通过传送诸如“I am alive”这样的信息来监控对方的运行状况。当备份机不能在一定的时间内收到这样的信息时，它就接管主服务器的服务IP并继续提供负载均衡服务；当备份管理器又从主管理器收到“I am alive”这样的信息时，它就释放服务IP地址，这样的主服务器就开始再次提供负载均衡服务。

### 2、keepalived+nginx实现主备

#### （1）什么是keepalived

keepalived是集群管理中保证集群高可用的一个服务软件，用来防止单点故障。

Keepalived的作用是检测web服务器的状态，如果有一台web服务器死机，或工作出现故障，Keepalived将检测到，并将有故障的web服务器从系统中剔除，当web服务器工作正常后Keepalived自动将web服务器加入到服务器群中，这些工作全部自动完成，不需要人工干涉，需要人工做的只是修复故障的web服务器。

#### （2）keepalived工作原理

keepalived是以VRRP协议为实现基础的，VRRP全称Virtual Router Redundancy Protocol，即虚拟路由冗余协议。

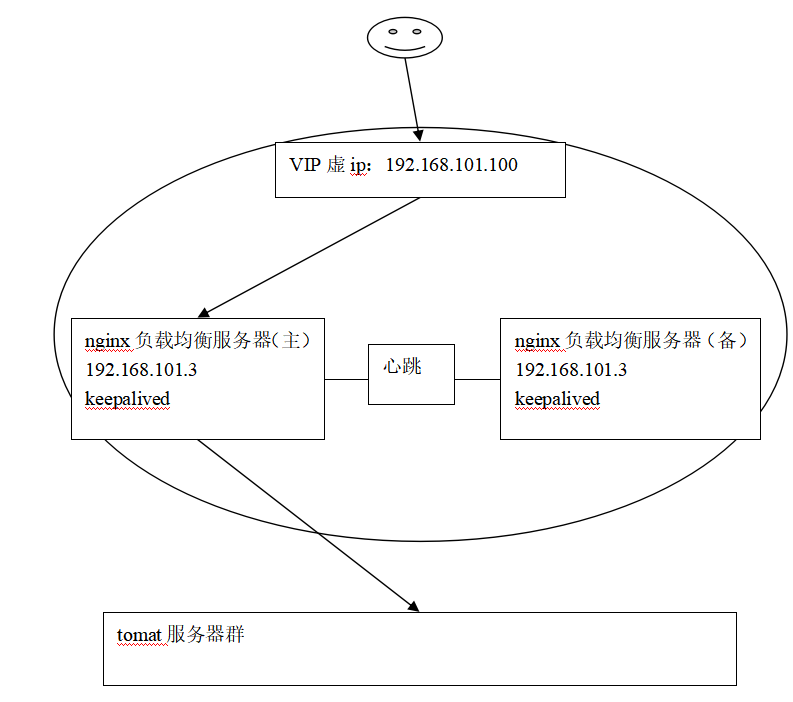
虚拟路由冗余协议，可以认为是实现路由器高可用的协议，即将N台提供相同功能的路由器组成一个路由器组，这个组里面有一个master和多个backup，master上面有一个对外提供服务的vip（VIP = Virtual IP Address，虚拟IP地址，该路由器所在局域网内其他机器的默认路由为该vip），master会发组播，当backup收不到VRRP包时就认为master宕掉了，这时就需要根据VRRP的优先级来选举一个backup当master。这样的话就可以保证路由器的高可用了。

keepalived主要有三个模块，分别是core、check和VRRP。core模块为keepalived的核心，负责主进程的启动、维护以及全局配置文件的加载和解析。check负责健康检查，包括常见的各种检查方式。VRRP模块是来实现VRRP协议的。

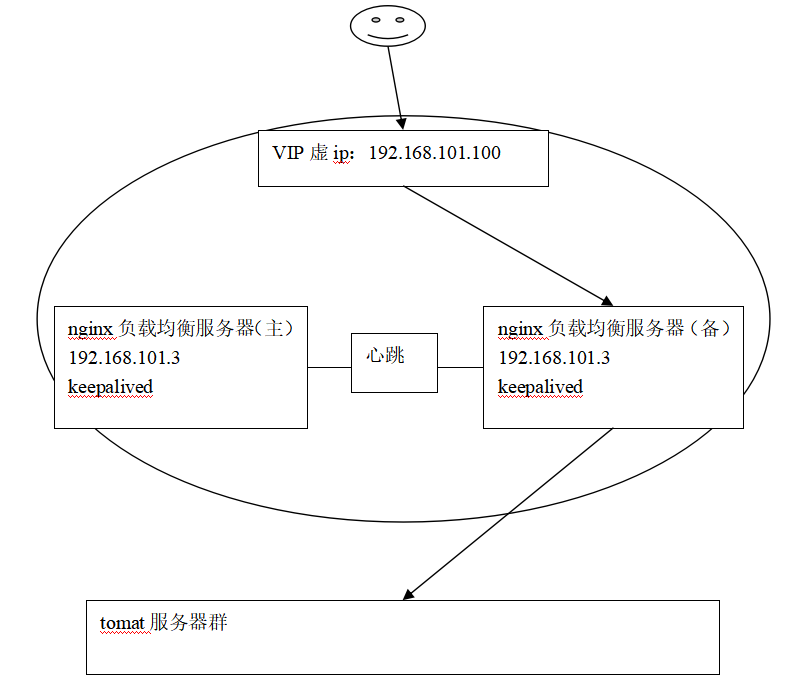
详细参考：Keepalived权威指南中文.pdf

#### （3）keepalived+nginx实现主备过程

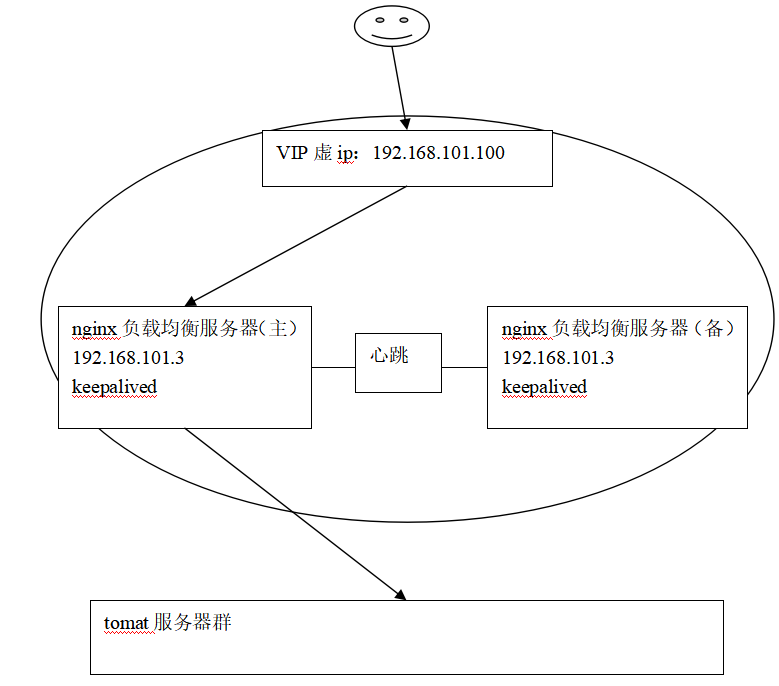
##### 初始状态



##### 主机宕机



##### 3）主机恢复



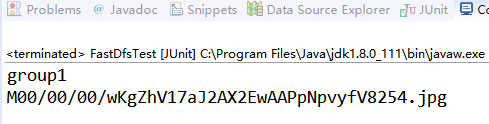
# FastDFS

1. 安装配置
2. 上传图片
3. 项目导入fastdfs\_client
4. 书写配置文件：

|  |
| --- |
| tracker\_server=192.168.25.133:22122 |

|  |
| --- |
| **import** org.csource.fastdfs.ClientGlobal;  **import** org.csource.fastdfs.StorageClient;  **import** org.csource.fastdfs.StorageServer;  **import** org.csource.fastdfs.TrackerClient;  **import** org.csource.fastdfs.TrackerServer;  **import** org.junit.Test;  **public** **class** FastDfsTest {  @Test  **public** **void** fastDfsTest() **throws** Exception {  //创建一个配置文件。文件名任意。内容就是tracker服务器的地址。  //使用全局对象加载配置文件。  ClientGlobal.*init*("E:\\eclipse-workspace\\e3-manager-web\\src\\main\\resources\\fastdfs\\client.conf");  //创建一个TrackerClient对象  TrackerClient trackerClient = **new** TrackerClient();  //通过TrackClient获得一个TrackerServer对象  TrackerServer trackerServer = trackerClient.getConnection();  //创建一个StorageServer的引用，可以是null  StorageServer storageServer = **null**;  //创建一个StorageClient，参数需要TrackerServer和StrorageServer  StorageClient storageClient = **new** StorageClient(trackerServer, storageServer);  //使用StorageClient上传文件。  String[] string = storageClient.upload\_file("H:\\娱乐\\壁纸图片\\01.jpg", "jpg", **null**);  **for**(String str: string) {  System.***out***.println(str);  }  }    } |

返回字符串：



访问：

<http://192.168.25.133/group1/M00/00/00/wKgZhV17aJ2AX2EwAAPpNpvyfV8254.jpg>

# KindEditor

# redis

# shorCloud

# actionMQ

# freemaker

# jsonp

# mycat

# e3mall

开启项目时注意：防火墙有没有关，ip有没有变

## 商品列表显示

1、静态资源引入

因为springmvc配置了拦截除jsp以外的所有资源，所以要配置资源映射

|  |
| --- |
| <!-- 添加静态资源映射 -->  <mvc:resources location=*"/css/"* mapping=*"/css/\*\*"*></mvc:resources>  <mvc:resources location=*"/js/"* mapping=*"/js/\*\*"*></mvc:resources> |

静态资源跳转

|  |
| --- |
| @Controller  **public** **class** PageController {  @RequestMapping("/")  **public** String accessIndex() {  **return** "index";  }    @RequestMapping("/{page}")  **public** String accessPage(@PathVariable String page) {  **return** page;  }    } |

问题：分页列表中使用pageHelper，e3-manager-web抛异常

原因：e3-manager-web中没有加pageHelper依赖

e3-manager-web调用e3-manager-service中的服务，返回值为分页对象，分页对象中有list。

该list是pageHelper封装的，封装的不是ArrayList，而是pageHelper自己定义的数据类型Page。

不过Page继承了ArrayList。

## 添加商品——加载商品类别Tree

## 添加商品——上传图片（fastDFS）

1、springmvc上传图片注意：

需要jar包：commons-io、fileupload

springmvc中配置多媒体解析器

2、上传图片：浏览器兼容性问题

springmvc中@responseBody的意思是直接响应浏览器，不经过视图解析器；如果返回值有对象，就把对象转成json，且返回值类型content-type:application/json。

如果返回值为string，默认content-type:text/plain

uEditor：百度的富文本编辑器

ckeditor

jquery提供的方法$(form).serialize()

虚拟机修改ip后不重启生效：service network restart

伪图形界面：setup命令

user DHCP \*表示动态获取ip

## 添加商品

dubbo调用服务默认为3次，3次都失败后就停止调用

# Git

1. git安装
2. tortoiseGit安装,汉化包下载安装
3. 向本地仓库中添加、修改文件
4. 删除还原文件，删除并保留本地副本
5. 图标显示问题（禁用微软自带的OneDriver，然后重启）
6. 把文件或文件名添加到忽略列表，.gitignore也要添加到本地仓库中去
7. 创建远程仓库
   1. 在本地创建密钥对
   2. 在github上添加公钥
   3. 使用命令、小乌龟把分支上传到远程仓库
   4. 两种传输方式：ssh、https
8. 从远程仓库推送、拉去文件
9. 解决版本冲突
10. 在linux上搭建私有git服务器
11. 从远程仓库中下载上传文件
12. 创建、合并、删除分支
13. 在idea中使用git

git命令：

创建本地仓库：git init --bare（参数表示不带工作目录）

创建密钥对：ssh-keygen -t rsa 默认保存在当前Windows用户文件夹下的.ssh文件夹下

连接远程仓库：git remote add origin git@github.com:299413619/reop1.git

把分支上传到远程仓库：git push -u origin master

把远程仓库的内容克隆到本地：git clone <https://github.com/2994413619/repo3.git>

# SpringBoot

1. 快速入门
2. springboot热部署
3. yml文件语法
   1. 普通数据
   2. 对象数据
   3. map对象数据
   4. 数组数据
4. 配置文件数据映射到java代码中

@Value

@ConfigurationProperties

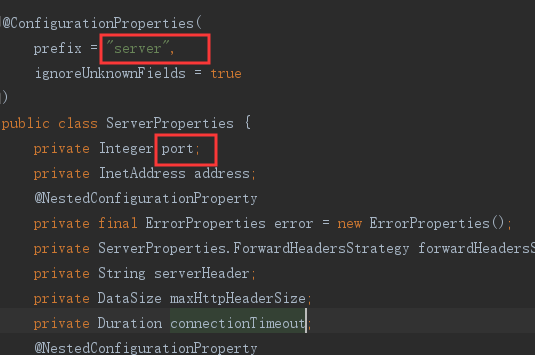
5、SpringBoot集成mybatis

@SpringBootApplication源码分析：

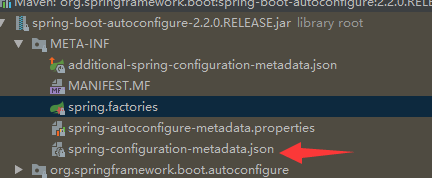
该注解上的@EnableAutoConfiguration注解表示是否开始自动配置

@EnableAutoConfiguration注解上的@Import({AutoConfigurationImportSelector.class})表示加载配置

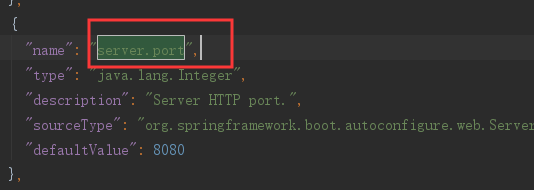
AutoConfigurationImportSelector类中加载该包下的META-INF/spring.factories.文件，该文件中是完整包名的键值对，其中ServletWebServerFactoryAutoConfiguration类上@EnableConfigurationProperties({ServerProperties.class})中ServerProperties类用来存储的加载配置属性。



他加载的配置文件为META-INF/spring.factories.同级目录下的json文件，该文件位置：



文件中属性名就是上面截图中的server.属性名



该json中的默认配置，我们可以写配置文件覆盖。

但覆盖的配置文件写在哪里，如何加载？

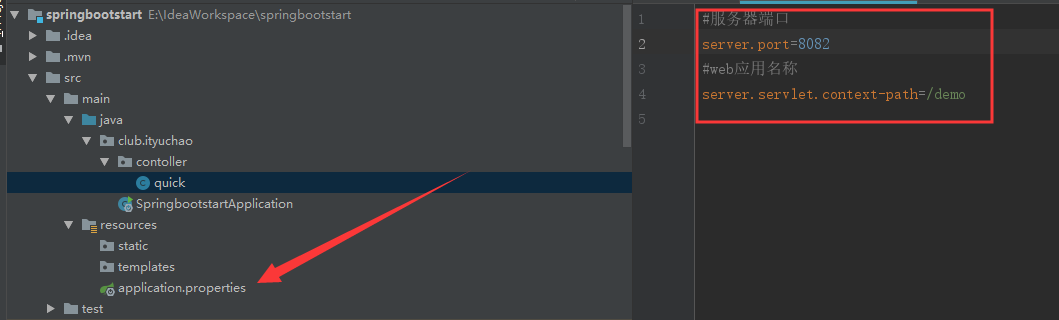
我们的springboot文件都必须有该标签：

|  |
| --- |
| <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>2.2.0.RELEASE</version>  <relativePath/> <!-- lookup parent from repository --> </parent> |

打开spring-boot-starter-parent，我们可以看到加载的配置文件：



例如：修改访问端口和项目名称：



# SpringCloud

## 一、系统架构演变

### 1、集中式架构

缺点：

1. 单点故障
2. 并发数低
3. 代码耦合度高
4. 不方便扩展

好处：

维护简单

### 2、水平切分

web、service、mapper

### 3、垂直拆分，功能拆分

优点：

1. 方便水平扩展
2. 方便单独优化
3. 解耦合
4. 提高并发

缺点：

增加维护成本

重复开发

### 4、没有注册中心的调用

调用关系错综复杂

维护困难

### 5、SOA（面向服务开发）

用注册中心解决了调用关系错综复杂的问题

### 6、微服务

微服务特点：

1. 单一职责：
2. 微：相比SOA，微服务的粒度更小
3. 面向服务：每个服务对外提供一个rest风格的接口API，对内不限制实现服务的技术
4. 自治：
   1. 团队独立
   2. 技术独立
   3. 前后端分离
   4. 数据库分离
   5. 独立部署

springcloud发展：

dubbo12年停更，17年恢复，并共享给Apache，如果dubbo没有停更几年，中国根本没有springcloud的事

## 二、服务调用方式

RPC和HTTP

1）RPC（传输层）：Remote Produce Call远程过程调用，类似还有RMI。自定义数据格式，基于原生TCP通信，速度快，效率高，早期的WebService，现在热门的dubbo，都是RPC的典型代表。

2）Http：缺点是消息封装臃肿，优势是对服务的提供和调用方都没有任何技术限制，只有灵活，更符合微服务理念

Rest风格，可以通过http协议来实现

## 三、http客户端工具

HttpClient

OKHttp

URLConnection

spring-boot-start-web中有Jackson的包，该工具中，ObjectMapper可以进行序列化和反序列化

反序列化方法：readValue(jsonString,class);

序列化：writeValueAsString(Object);