**Zookeeper**

| | | | |
|---|---|---|---|
| **笔记本：** | NoteBook | | |
| **创建时间：** | 2020/6/12 星期五 8:53 | **更新时间：** | 2020/7/8 星期三 18:09 |
| **作者：** | INVINCIBLE | | |
| **URL：** | https://blog.csdn.net/dc_726/article/details/46475633 | | |

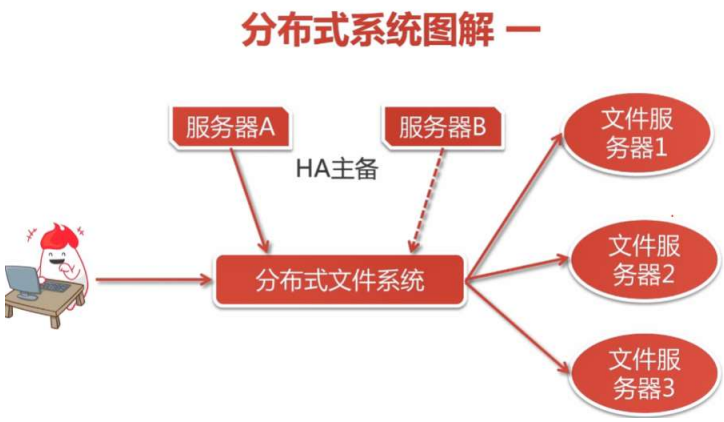
# 第一章　分布式系统概念与Zookeeper简介

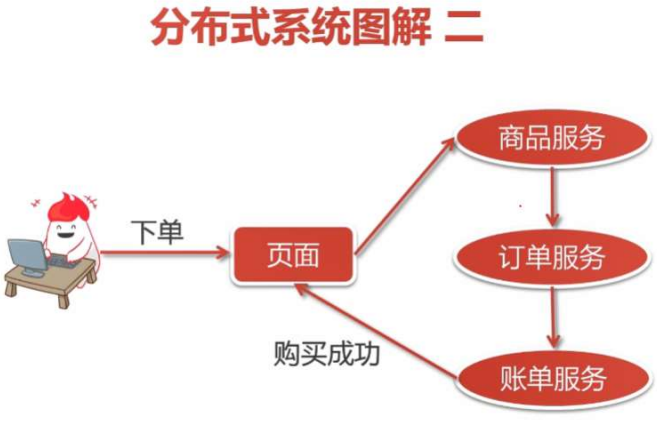
## 1.1、Zookeeper简介

中间件、提供协调服务
作用于分布式系统，发挥其优势，可以为大数据服务
支持java，提供java和c语言的客户端api

## 1.2、什么是分布式系统

很多台计算机组成一个整体，一个整体一致对外并且处理同一请求
内部的每台计算机都可以相互通信(rest/rpc)
客户端到服务端的一次请求到响应结束会历经多台计算机





## 1.3、Zookeeper的特性

一致性：　数据一致性，数据按照顺序分批入库
原子性：　事务要么成功要么失败，不会局部化

单一视图： 客户端连接集群中的任一zk节点，数据都是一致的。
可靠性： 每次对zk的操作状态都会保存在服务端
实时性： 客户端可以读取到zk服务端的最新数据

## 第二章 Zookeeper的安装

### 2.1、JDK的安装

a) 下载linux的jdk1.8 tar,上传至linux服务器
b) 解压缩jdk，配置jdk
c) 测试：java -version显示版本号

### 2.2、单机zookeeper安装

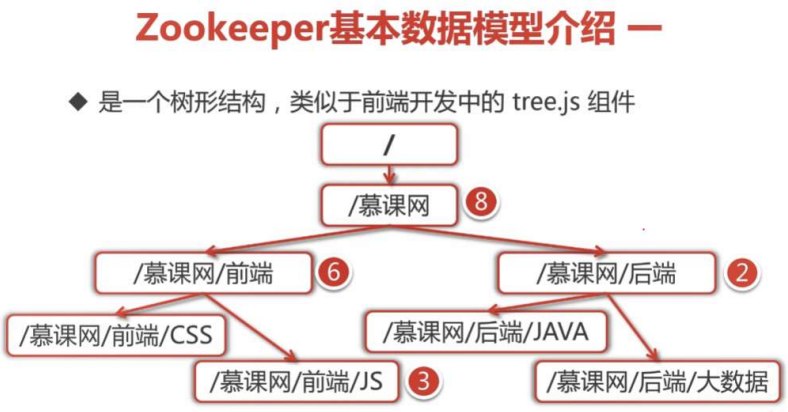
a) zookeeper下载、安装以及配置环境变量
https://www.cnblogs.com/yuefeng123/p/11805550.html

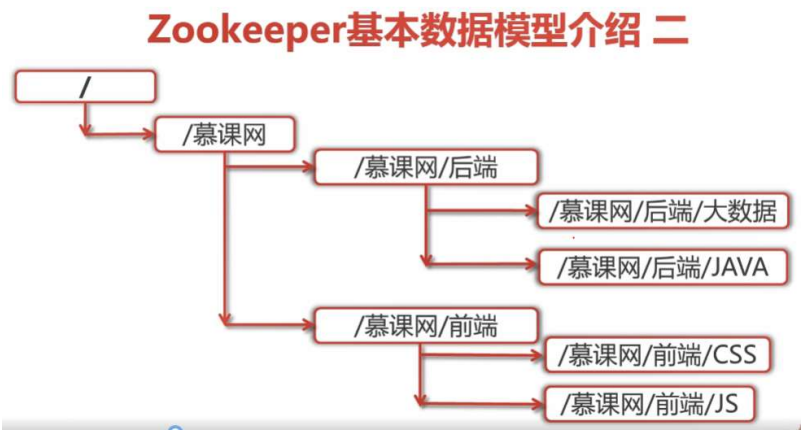
b) zookeeper文件夹主要目录介绍
bin： 主要的一些运行命令
conf： 存放配置文件，其中我们需要修改zk.cfg
contrib： 附加的一些功能
dist-maven：mvn编译后的目录
docs： 文档
lib： 需要依赖的jar包
recipes： 案例demo源码
src： 源码

c) zookeeper配置文件介绍，运行zk
zoo.cfg配置：
tickTime： 用于计算的时间单元。比如session超时：N*tickTime
initLimit： 用于集群，允许从节点连接并同步到master节点的初始化连接时间，以tickTime的倍数来表示。 syncLimit： 用于集群，master主节点与从节点之间发送消息，请求和应答时间长度(心跳机制)
dataDir： 必须配置
dataLogDir：日志目录。如果不配置将会和DataDir共用一个目录。
clientPort： 连接服务器的端口，默认2181
启动zookeeper：
cd 到zookeeper/bin目录下，./zkServer.sh start启动zookeeper，./zkServer.sh 查看帮助

## 第三章 Zookeeper基本数据模型介绍

### 3.1、是一个树形结构，类似于前端开发中的tree.js组件






a、zk的数据模型也可以理解为linux/unix的文件目录：/usr/local/...
b、每一个节点都称之为znode，他可以有子节点，也可以有数据
c、每个节点分为临时节点和永久节点，临时节点在客户端断开后消失
d、每个zk节点都有各自的版本号，可以通过命令行来显示节点信息
e、每当节点数据发生变化，那么该节点的版本号会累加（乐观锁）
f、删除/修改过时节点，版本号不匹配则会报错
g、每个zk节点存储的数据不宜过大，几k即可
h、节点可以设置权限acl，可以通过权限来限制用户的访问

### 3.2、Zookeeper数据模型基本操作

**a、客户端连接**

cd到zookeeper安装bin目录下，运行zkServer.sh启动zookeeper
同目录下运行zkClient.sh 连接zookeeper
输入help回车可查询相关命令

**b、查看znode结构**

ls：查看当前目录结构 (通常结构：/zookeeper/quota)

**c、关闭客户端连接**

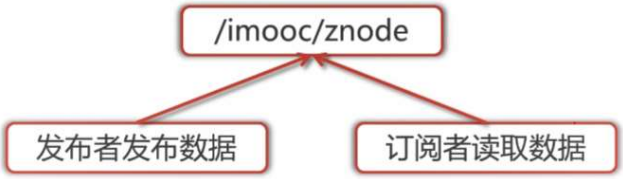
ctrl+c 即可断开当前连接

### 3.3、zk的作用体现

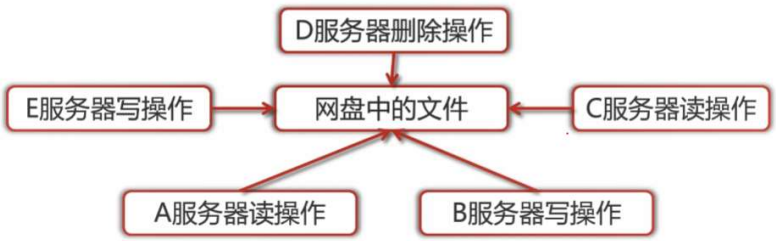
**master节点选举**，主节点挂了以后，从节点就会接手工作，并且保证这个节点是唯一的，
这也是首脑模式，从而保证我们的集群是高可用的。

**统一配置文件管理**，即只需要部署一台服务器，则可以把相同的配置文件同步更新到其他所有服务器，
此操作在计算中用的特别多（假设修改了redis统一配置）

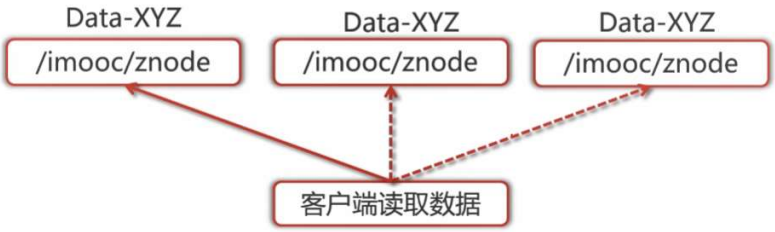

**发布与订阅**，类似消息队列MQ（amq,rmq...）,dubbo发布者把数据存在znode上，订阅者会读取这个数据




**提供分布式锁**，分布式环境中不同进程之间争夺资源，类似于多线程中的锁




**集群管理**，集群中保证数据的强一致性




## 第4章　Zookeeper客户端命令

### 4.1、zkClient.sh

启动客户端

### 4.2、ls与ls2命令

ls查看当前目录，ls2查看当前目录以及状态信息

### 4.3、get与stat命令

stat查看当前目录状态信息，get获取当前节点（目录）数据和状态信息

```
cZxid = 0x0                                  //当前节点编号
ctime = Thu Jan 01 08:00:00 CST 1970         //当前节点创建时间
mZxid = 0x0                                  //修改后的节点编号
mtime = Thu Jan 01 08:00:00 CST 1970         //当前节点修改时间
pZxid = 0x0                                  //子节点编号
cversion = -1                                //子节点版本
aclVersion = 0                               //当前节点权限版本
ephemeralOwner = 0x0                         //节点类型标识，0x0为永久节点、其他为临时节点
dataLength = 0                               //当前节点数据长度
numChildren = 1                              //子节点数量
```

### 4.4、create命令

```
create -e /xxx/xxx data     创建临时节点
create -s /xxxx      data     创建永久节点
```

### 4.5、set命令

```
set  /path  data                设置节点数据
set  /path   data  version      根据节点版本设置节点数据（乐观锁）
```

### 4.6、delete命令

```
delete   /path                  删除节点数据
delete   /path  version         根据节点版本删除节点（乐观锁）
```

### 4.7、Session的基本原理

1、客户端与服务端之间的连接存在会话
2、每个会话都会可以设置一个超时时间
3、心跳结束，session则过期
4、session过期，则临时节点znode会被抛弃
5、心跳机制：客户端向服务端的ping包请求

### 4.8、watcher机制

1、针对每个节点的操作，都会有一个监督者 -->watcher
2、当监控的某个对象（znode）发生了变化，则出发watcher事件
3、zk中watcher是一次性的，触发后立即销毁
4、父节点，子节点增删改都能够触发其watcher
5、针对不同类型的操作，触发的watcher事件也不同：
   a、（子）节点创建事件
   b、（子）节点删除事件

c、（子）节点数据变化事件

**watcher命令行**

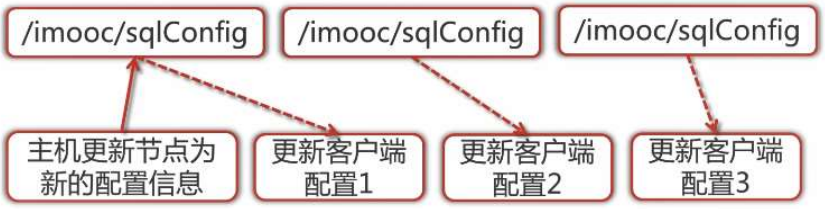
a、通过get path [watch]设置watcher
b、父节点增删改操作触发watcher
c、子节点增删改操作触发watcher

**watcher事件类型**

1、创建父节点触发：NodeCreated（stat path watch）
2、修改父节点数据触发：NodeDataChanged（get path watch）
3、删除父节点触发：NodeDeleted（get path watch）
4、ls为父节点设置watcher，创建子节点触发：NodeChildrenChanged
5、ls为父节点设置watcher，删除子节点触发：NodeChildrenChanged
6、ls为父节点设置watcher，修改子节点不触发事件

**watcher使用场景**

统一资源配置




## 4.9、ACL（access control lists）权限控制

针对节点可以设置相关读写等权限，目的为了保障数据安全性。
权限permissions可以指定不同的权限范围以及角色。

**ACL命令行**

getAcl： 获取某个节点的acl权限信息。
setAcl： 设置某个节点的acl权限信息。
addauth: 输入认证授权信息，注册时输入明文密码（登录），
但是在zk的系统里，密码是以加密的形式存在的。

**ACL构成**

zk的acl通过[scheme:id:permissions]来构成权限列表
scheme：代表采用的某种权限机制
world：world下面只有一个id，即只有一个用户，也就是anyone，
那么组合的写法就是world:anyone:[permissions]
auth：代表认证登录，需要注册用户有权限就可以，形式为
auth:user:password:[permisisions]
digest：需要对密码加密才能访问，组合形式为
digest：username：BASE64（SHA1(password)）:[permissions]

◆ 简而言之，auth 与 digest的区别就是，前者明文，后者密文
setAcl /path auth:lee:lee:cdrwa
与
setAcl /path digest:lee:BASE64(SHA1(password))cdrwa
是等价的，在通过
addauth digest lee:lee 后都能操作指定节点的权限

ip：当设置为ip指定的IP地址，此时限制ip进行访问，比如ip:192.168.1.1:[permisssions]
super：代表超级管理员，拥有所有的权限
a.修改zkServer.sh 增加super管理员
b.重启zkServer.sh

id：代表允许访问的用户
permissions：权限组合字符串（权限字符串缩写 crdwa）
CREATE： 创建子节点
READ： 获取节点/子节点
WRITE： 设置节点数据
DELETE： 删除子节点
ADMIN： 设置权限

**ACL常见使用场景**

开发/测试环境分离，开发者无权操作测试库的节点，只能看
生产环境上控制指定ip的服务可以访问相关节点，防止混乱

## 4.10、zk四字命令 Four Letter Words

zk可以通过它自身提供的简写命令来和服务器进行交互
需要使用到nc命令，安装：yum install nc
echo [command] | nc [ip] [port]

**ZooKeeper Commands: The Four Letter Words**

ZooKeeper responds to a small set of commands. Each command is composed of four letters. You issue the commands to ZooKeeper via telnet or nc, at the client port.

Three of the more interesting commands: "stat" gives some general information about the server and connected clients, while "srvr" and "cons" give extended details on server and connections respectively.

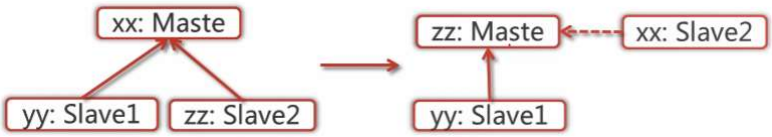
- *conf* : **New in 3.3.0:** Print details about serving configuration.
- *cons* : **New in 3.3.0:** List full connection/session details for all clients connected to this server. Includes information on numbers of packets received/sent, session id, operation latencies, last operation performed, etc...
- *crst* : **New in 3.3.0:** Reset connection/session statistics for all connections.
- *dump* : Lists the outstanding sessions and ephemeral nodes. This only works on the leader.
- *envi* : Print details about serving environment
- *ruok* : Tests if server is running in a non-error state. The server will respond with imok if it is running. Otherwise it will not respond at all. A response of "imok" does not necessarily indicate that the server has joined the quorum, just that the server process is active and bound to the specified client port. Use "stat" for details on state wrt quorum and client connection information.
- *srst* : Reset server statistics.
- *srvr* : **New in 3.3.0:** Lists full details for the server.
- *stat* : Lists brief details for the server and connected clients.
- *wchs* : **New in 3.3.0:** Lists brief information on watches for the server.
- *wchc* : **New in 3.3.0:** Lists detailed information on watches for the server, by session. This outputs a list of sessions(connections) with associated watches (paths). Note, depending on the number of watches this operation may be expensive (ie impact server performance), use it carefully.
- *dirs* : **New in 3.5.1:** Shows the total size of snapshot and log files in bytes
- *wchp* : **New in 3.3.0:** Lists detailed information on watches for the server, by path. This outputs a list of paths (znodes) with associated sessions. Note, depending on the number of watches this operation may be expensive (ie impact server performance), use it carefully.
- *mntr* : **New in 3.4.0:** Outputs a list of variables that could be used for monitoring the health of the cluster.

[stat] 查看zk的状态信息，以及是否mode
[ruok] 查看当前zkserver是否启动，返回imok
[dump] 列出未经处理的会话和临时节点
[conf] 查看服务器配置
[cons] 展示连接到服务器的客户端信息
[envi] 环境变量
[mntr] 监控zk健康信息
[wchs] 展示watch的信息
[wchc] 与 [wchp] session与watch及path与watch信息

# 第5章　选举模式与Zookeeper集群安装

## 5.1 zookeeper集群搭建

zk集群，主从节点，心跳机制（选举模式）




**zookeeper伪分布式集群搭建注意点：**

配置数据文件 myid 1/2/3 对应server.1/2/3
通过./zkCli.sh -server [ip]:[port] 检测集群是否配置成功

```
[root@wang conf]# cat zoo.cfg
tickTime=2000
initLimit=10
syncLimit=5
dataDir=/usr/local/zookeeper-3.4.14-02/dataDir
dataLogDir=/usr/local/zookeeper-3.4.14-02/dataLogDir
clientPort=2182

server.1=192.168.0.105:2888:3888
server.2=192.168.0.105:2889:3889
server.3=192.168.0.105:2890:3890
[root@wang conf]#
```

每个zookeeper配置目录下配置server.x=ip:message port:leader port
并修改clientPort、dataDir、dataLogDir的值，最后在dataDir下创建myid文件
文件内容对应server.x中x的值。依次启动各个zookeeper即可。

**zookeeper真实集群搭建**

需要注意：环境变量的配置，ip配置不同，端口号可以相同
集群测试，选举测试

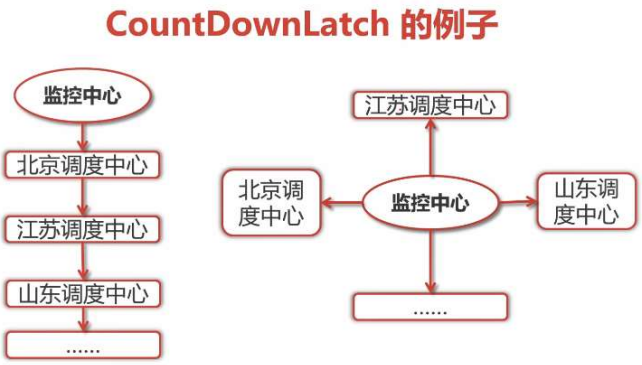
# 第6章　使用ZooKeeper原生Java API进行客户端开发

## 6.1　zookeeper原生Java Api使用

会话连接与恢复
节点的增删改查
watch与acl操作
https://blog.csdn.net/ouzhuangzhuang/article/details/86680258

## 6.2　CountDownLatch

它是一个计数器
多用于线程，可以暂停也可以继续
await() 阻塞当前线程，countDown()




### 6.2 原生Java api节点查询

获取节点数据
获取子节点数据
判断节点是否存在

## 第7章 Apache Curator客户端的使用

### 7.1 常用的zk java客户端

**zk原生api**
超时重连，不支持自动，需要手动操作
Watch注册一次后会失效
不支持递归创建节点
**zkclient**
**Apache curator**
Apache的开源项目
解决watcher的注册一次就失效
Api更加简单易用
提供更多解决方案并且实现简单：比如分布式锁
提供常用的Zookeeper工具类
编程风格更爽

### 7.2 客户端Curator使用

**会话连接与关闭**
**节点的增删改查**
https://www.jianshu.com/p/db65b64f38aa
**watch与acl的相关操作**
Curator提供了三种Watcher(Cache)来监听结点的变化：
Path Cache：
监视一个路径下
1）孩子结点的创建、
2）删除，
3）以及结点数据的更新。
产生的事件会传递给注册的PathChildrenCacheListener。
Node Cache：
监视一个结点的创建、更新、删除，并将结点的数据缓存在本地。
Tree Cache：
Path Cache和Node Cache的“合体”，监视路径下的创建、更新、删除事件，并缓存路径下所有孩子结点的数据

**锁：**
包括共享锁、共享可重入锁、读写锁等。
**选举：**
Leader选举算法。
**Barrier：**
阻止分布式计算直至某个条件被满足的“栅栏”，可以看做JDK Concurrent包中 Barrier的分布式实现。
**缓存：**
前面提到过的三种Cache及监听机制。
**持久化结点：**
连接或Session终止后仍然在Zookeeper中存在的结点。
**队列：**
分布式队列、分布式优先级队列等。

**分布式锁：**

分布式编程时，比如最容易碰到的情况就是应用程序在线上多机部署，于是当多个应用同时访问某一资源时，就需要某种机制去协调它们。例如，现在一台应用正在rebuild缓存内容，要临时锁住某个区域暂时不让访问；又比如调度程序每次只想一个任务被一台应用执行等等。
下面的程序会启动两个线程t1和t2去争夺锁，拿到锁的线程会占用5秒。运行多次可以观察到，有时是t1先拿到锁而t2等待，有时又会反过来。Curator会用我们提供的lock路径下创建子结点作为全局锁，这个子节点名称类似：[_c_64e0811f-9475-44ca-aa36-c1db65ae5350-lock-0000000005]，每次获得锁时会生成这种串，释放锁时清空数据。

**Leader选举：**

当集群里的某个服务down机时，我们可能要从slave结点里选出一个作为新的master，这时就需要一套能在分布式环境中自动协调的Leader选举方法。Curator提供了LeaderSelector监听器实现Leader选举功能。同一时刻，只有一个Listener会进入takeLeadership()方法，说明它是当前的Leader。注意：当Listener从takeLeadership()退出时就说明它放弃了“Leader身份”，这时Curator会利用Zookeeper再从剩余的Listener中选出一个新的Leader。autoRequeue()方法使放弃Leadership的Listener有机会重新获得Leadership，如果不设置的话放弃了的Listener是不会再变成Leader的。

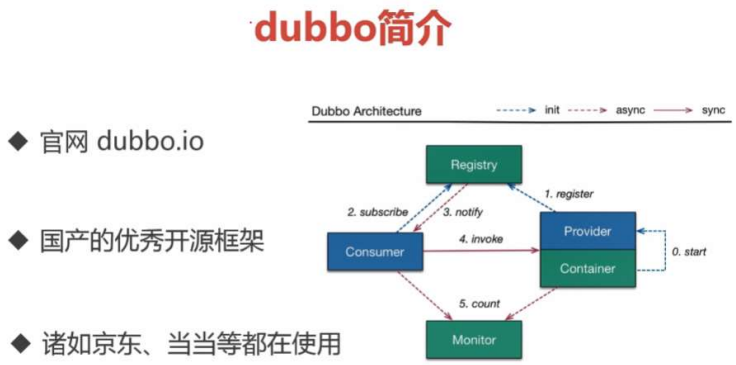
参考：

https://blog.csdn.net/dc_726/article/details/46475633

http://www.throwable.club/2018/12/16/zookeeper-curator-usage/#Zookeeper%E5%AE%A2%E6%88%B7%E7%AB%AFCurator%E4%BD%BF%E7%94%A8%E8%AF%A6%E8%A7%A3

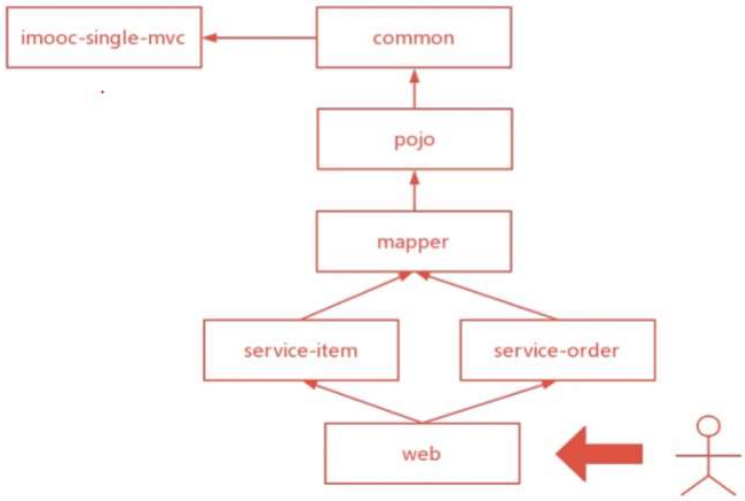
# 第8章　Dubbo入门与重构服务

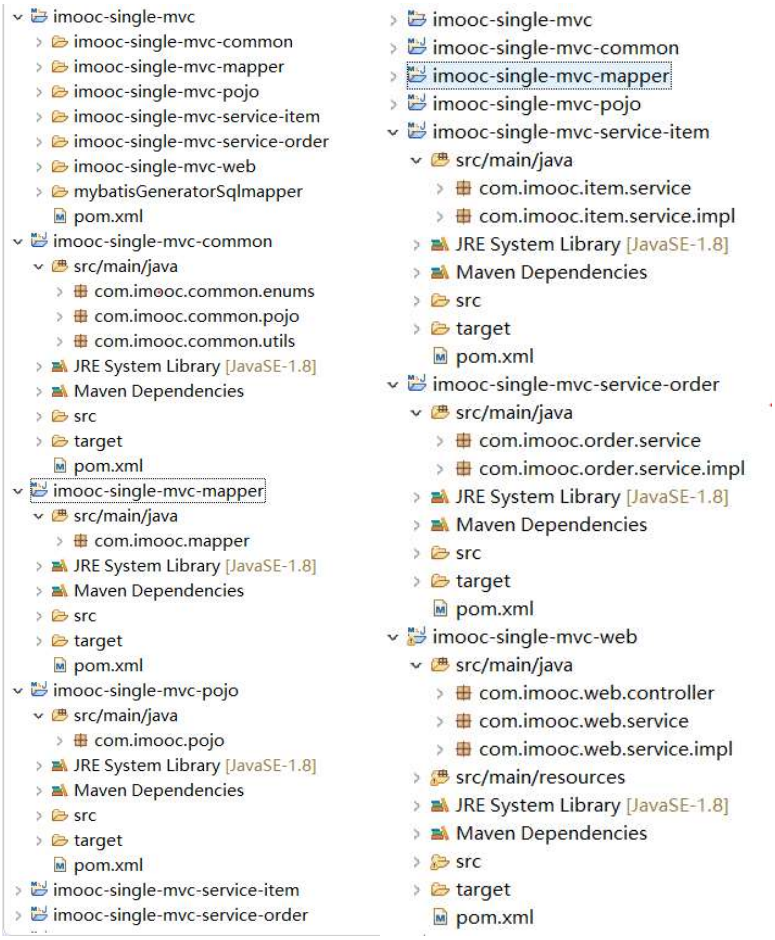
## 8.1 Dubbo简介




**最大程度进行解耦，降低系统耦合性**
**生产者/消费者模式**
**zk注册中心，admin监控中心，协议支持**

## 8.2 单层到分层模式代码演示






### 8.3 重构商品服务并抽取抽象工程，暴露商品服务

1、创建聚合工程，聚合所有dubbo子项目，引入dubbo相关依赖
2、创建imooc-dubbo-common项目，引入相关依赖，保留src目录
3、创建imooc-dubbo-item-api项目，创建抽象接口ItemService，引入相关pojo
4、a、创建imooc-dubbo-item项目，创建抽象接口ItemService的实现类，引入Mapper
b、引入imooc-dubbo-item-api、dubbo、spring、mybatis、mysql、连接池等依赖。
c、引入mybatis配置文件，数据库配置文件db.properties，spring配置文件(dao.xml,
service.xml,transaction.xml 注意删除spring-mvc相关配置)，log4j.properties。
d、引入dubbo配置文件dubbo-provider.xml(提示需要引入xsd，dubbo项目下载查找引入path)

### 8.4 运行dubbo的三种方式

**a、Tomcat容器启动**

修改service打包方式为war，引入maven tomcat插件，maven clean tomcat7:run 启动

**b、Main主线程运行**

1、dubbo官方网站 http://dubbo.apache.org/en-us/docs/user/quick-start.html
拷贝启动类至src目录

```
public class Provider {
     public static void main(String[] args) throws Exception {
System.setProperty("java.net.preferIPv4Stack", "true");
         ClassPathXmlApplicationContext context = new ClassPathXmlApplicationContext(new String[]
{"META-INF/spring/dubbo-demo-provider.xml"}); context.start();
        System.out.println("Provider started.");
        System.in.read(); // press any key to exit
    }
}
```

2、resource目录下建立代码引用配置文件
3、启动上上述该类

**c、dubbo内置main打包jar**

```
    <!-- 通过dubbo内置main打包jar -->
     <build>
         <finalName>imooc-dubbo-item-service</finalName>
         <resources>
              <resource>
                  <targetPath>${project.build.directory}/classes</targetPath>
                  <directory>src/main/resources</directory>
                  <filtering>true</filtering>
```

```
                <includes>
                     <include>**/*.xml</include>
                     <include>**/*.properties</include>
                </includes>
            </resource>
            <resource>
                <targetPath>${project.build.directory}/classes/META-INF/spring</targetPath>
                <directory>src/main/resources/spring</directory>
                <filtering>true</filtering>
                <includes>
                     <include>spring-context.xml</include>
                </includes>
            </resource>
        </resources>
        <pluginManagement>
            <plugins>
                <plugin>
                     <groupId>org.eclipse.m2e</groupId>
                     <artifactId>lifecycle-mapping</artifactId>
                     <version>1.0.0</version>
                     <configuration>
                          <lifecycleMappingMetadata>
                              <pluginExecutions>
                                   <pluginExecution>
                                        <pluginExecutionFilter>
                                            <groupId>org.apache.maven.plugins</groupId>
                                            <artifactId>maven-dependency-plugin</artifactId>
                                            <versionRange>[2.0,)</versionRange>
                                            <goals>
                                                 <goal>copy-dependencies</goal>
                                            </goals>
                                        </pluginExecutionFilter>
                                        <action>
                                            <ignore  />
                                        </action>
                                   </pluginExecution>
                              </pluginExecutions>
                          </lifecycleMappingMetadata>
                     </configuration>
                </plugin>
            </plugins>
        </pluginManagement>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-jar-plugin</artifactId>
                <configuration>
                     <classesDirectory>target/classes/</classesDirectory>
                     <archive>
                          <manifest>
                              <mainClass>com.alibaba.dubbo.container.Main</mainClass>
                              <useUniqueVersions>false</useUniqueVersions>
                              <addClasspath>true</addClasspath>
                              <classpathPrefix>lib/</classpathPrefix>
                          </manifest>
                          <manifestEntries>
                              <Class-Path>.</Class-Path>
                          </manifestEntries>
                     </archive>
                </configuration>
            </plugin>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-dependency-plugin</artifactId>
                <executions>
                     <execution>
                          <id>copy-dependencies</id>
                          <phase>package</phase>
                          <goals>
                              <goal>copy-dependencies</goal>
                          </goals>
                          <configuration>
                              <type>jar</type>
                              <includeTypes>jar</includeTypes>
                              <useUniqueVersions>false</useUniqueVersions>
                              <outputDirectory>
                                   ${project.build.directory}/lib
                              </outputDirectory>
                          </configuration>
                     </execution>
                </executions>
            </plugin>
        </plugins>
    </build>
```

将上述代码拷贝至pom.xml，修改打包方式为jar，在父项目运行maven install，在service项目target目录下看到生成的jar包，在该目录下cmd运行jar包即可。

### 8.5 重构订单服务

参考重构商品服务

### 8.6 消费方对订单服务以及商品服务的引用

a、imooc-dubbo-web引入imooc-dubbo-item-api、imooc-dubbo-order-api、imooc-dubbo-common、Spring、SpringMVC、Dubbo、Zookeeper、Jsp等相关依赖。

b、src编写相关代码

c、引入相关资源文件（Spring、SpringMVC、web.xml）

d、先启动imooc-dubbo-item-api、imooc-dubbo-order-api两个provider，然后启动imooc-dubbo-web。

e、浏览器访问测试

### 8.7 dubbo监控服务

dubbo提供了监控服务，启动并且运行后可以对服务提供方以及消费者进行监控，查看各自的被调用的次数以及成功失败的次数，还有并发数。

下载dubbo源码并对dubbo进行编译

提取监控服务的安装包运行

### 8.8 dubbo 2.6.0的使用

dubbo在经历5年后正式恢复更新

稳定版本2.5.3

截止目前最新版2.6.0

（注意：dubbo2.6.0自带spring和servlet-api依赖，与项目引入依赖冲突，需要排除 ）

```
2020-07-08 14:03:28,461 [localhost-startStop-1]  [com.alibaba.dubbo.qos.server.Server.start(Server.java:102)] -
[ERROR]  [DUBBO] qos-server can not bind  localhost:33333, dubbo version: 2.6.0, current host:  127.0.0.1
java.net.BindException: Address already in use: bind
```

上述异常请参考：http://dubbo.apache.org/zh-cn/docs/user/references/qos.html

## 第9章　分布式锁

### 9.1 分布式锁简介

分布式锁的概念　　最终数据的不一致


目的：数据的最终一致性

Curator和Spring整合

分布式锁流程图