# 课程目标

Solr集群

Tomcat部署

Mysql安装

整体调试

# 什么是SolrCloud

## 什么是SolrCloud

SolrCloud(solr 云)是Solr提供的分布式搜索方案，当你需要大规模，容错，分布式索引和检索能力时使用 SolrCloud。当一个系统的索引数据量少的时候是不需要使用SolrCloud的，当索引量很大，搜索请求并发很高，这时需要使用SolrCloud来满足这些需求。

SolrCloud是基于Solr和Zookeeper的分布式搜索方案，它的主要思想是使用Zookeeper作为集群的配置信息中心。

它有几个特色功能：

1）集中式的配置信息

2）自动容错

3）近实时搜索

4）查询时自动负载均衡

### zookeeper是个什么玩意？

顾名思义zookeeper就是动物园管理员，他是用来管hadoop（大象）、Hive(蜜蜂)、pig(小猪)的管理员， Apache Hbase和 Apache Solr 的分布式集群都用到了zookeeper；Zookeeper:是一个分布式的、开源的程序协调服务，是hadoop项目下的一个子项目。

### Zookeeper可以干哪些事情

**1、配置管理**

在我们的应用中除了代码外，还有一些就是各种配置。比如数据库连接等。一般我们都是使用配置文件的方式，在代码中引入这些配置文件。但是当我们只有一种配置，只有一台服务器，并且不经常修改的时候，使用配置文件是一个很好的做法，但是如果我们配置非常多，有很多服务器都需要这个配置，而且还可能是动态的话使用配置文件就不是个好主意了。这个时候往往需要寻找一种集中管理配置的方法，我们在这个集中的地方修改了配置，所有对这个配置感兴趣的都可以获得变更。比如我们可以把配置放在数据库里，然后所有需要配置的服务都去这个数据库读取配置。但是，因为很多服务的正常运行都非常依赖这个配置，所以需要这个集中提供配置服务的服务具备很高的可靠性。一般我们可以用一个集群来提供这个配置服务，但是用集群提升可靠性，那如何保证配置在集群中的一致性呢？ 这个时候就需要使用一种实现了一致性协议的服务了。Zookeeper就是这种服务，它使用Zab这种一致性协议来提供一致性。现在有很多开源项目使用Zookeeper来维护配置，比如在HBase中，客户端就是连接一个Zookeeper，获得必要的HBase集群的配置信息，然后才可以进一步操作。还有在开源的消息队列Kafka中，也使用Zookeeper来维护broker的信息。在Alibaba开源的SOA框架Dubbo中也广泛的使用Zookeeper管理一些配置来实现服务治理。

**2、名字服务**

名字服务这个就很好理解了。比如为了通过网络访问一个系统，我们得知道对方的IP地址，但是IP地址对人非常不友好，这个时候我们就需要使用域名来访问。但是计算机是不能是别域名的。怎么办呢？如果我们每台机器里都备有一份域名到IP地址的映射，这个倒是能解决一部分问题，但是如果域名对应的IP发生变化了又该怎么办呢？于是我们有了DNS这个东西。我们只需要访问一个大家熟知的(known)的点，它就会告诉你这个域名对应的IP是什么。在我们的应用中也会存在很多这类问题，特别是在我们的服务特别多的时候，如果我们在本地保存服务的地址的时候将非常不方便，但是如果我们只需要访问一个大家都熟知的访问点，这里提供统一的入口，那么维护起来将方便得多了。

**3、分布式锁**

其实在第一篇文章中已经介绍了Zookeeper是一个分布式协调服务。这样我们就可以利用Zookeeper来协调多个分布式进程之间的活动。比如在一个分布式环境中，为了提高可靠性，我们的集群的每台服务器上都部署着同样的服务。但是，一件事情如果集群中的每个服务器都进行的话，那相互之间就要协调，编程起来将非常复杂。而如果我们只让一个服务进行操作，那又存在单点。通常还有一种做法就是使用分布式锁，在某个时刻只让一个服务去干活，当这台服务出问题的时候锁释放，立即fail over到另外的服务。这在很多分布式系统中都是这么做，这种设计有一个更好听的名字叫Leader Election(leader选举)。比如HBase的Master就是采用这种机制。但要注意的是分布式锁跟同一个进程的锁还是有区别的，所以使用的时候要比同一个进程里的锁更谨慎的使用。

**4、集群管理**

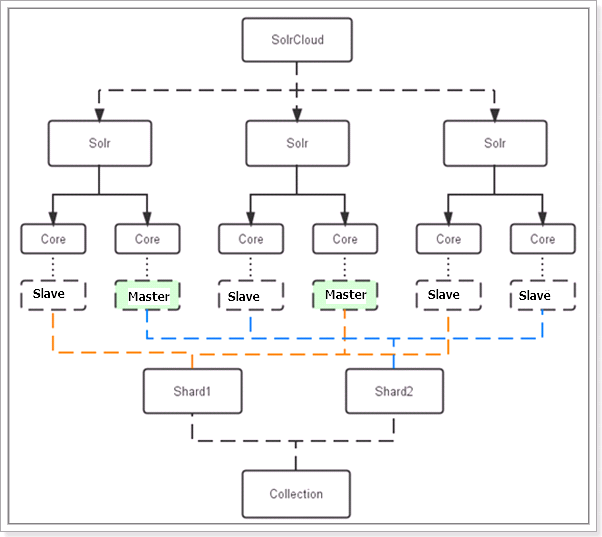
在分布式的集群中，经常会由于各种原因，比如硬件故障，软件故障，网络问题，有些节点会进进出出。有新的节点加入进来，也有老的节点退出集群。这个时候，集群中其他机器需要感知到这种变化，然后根据这种变化做出对应的决策。比如我们是一个分布式存储系统，有一个中央控制节点负责存储的分配，当有新的存储进来的时候我们要根据现在集群目前的状态来分配存储节点。这个时候我们就需要动态感知到集群目前的状态。还有，比如一个分布式的SOA架构中，服务是一个集群提供的，当消费者访问某个服务时，就需要采用某种机制发现现在有哪些节点可以提供该服务(这也称之为服务发现，比如Alibaba开源的SOA框架Dubbo就采用了Zookeeper作为服务发现的底层机制)。还有开源的Kafka队列就采用了Zookeeper作为Cosnumer的上下线管理。

## SolrCloud结构

SolrCloud为了降低单机的处理压力，需要由多台服务器共同来完成索引和搜索任务。实现的思路是将索引数据进行Shard（分片）拆分，每个分片由多台的服务器共同完成，当一个索引或搜索请求过来时会分别从不同的Shard的服务器中操作索引。

SolrCloud需要Solr基于Zookeeper部署，Zookeeper是一个集群管理软件，由于SolrCloud需要由多台服务器组成，由zookeeper来进行协调管理。

下图是一个SolrCloud应用的例子：



对上图进行图解，如下：



### 物理结构

三个Solr实例（ 每个实例包括两个Core），组成一个SolrCloud。

### 逻辑结构

索引集合包括两个Shard（shard1和shard2），shard1和shard2分别由三个Core组成，其中一个Leader两个Replication，Leader是由zookeeper选举产生，zookeeper控制每个shard上三个Core的索引数据一致，解决高可用问题。

用户发起索引请求分别从shard1和shard2上获取，解决高并发问题。

#### collection

Collection在SolrCloud集群中是一个逻辑意义上的完整的索引结构。它常常被划分为一个或多个Shard（分片），它们使用相同的配置信息。

比如：针对商品信息搜索可以创建一个collection。

collection=shard1+shard2+....+shardX

#### Core

每个Core是Solr中一个独立运行单位，提供 索引和搜索服务。一个shard需要由一个Core或多个Core组成。由于collection由多个shard组成所以collection一般由多个core组成。

#### Master或Slave

Master是master-slave结构中的主结点（通常说主服务器），Slave是master-slave结构中的从结点（通常说从服务器或备服务器）。同一个Shard下master和slave存储的数据是一致的，这是为了达到高可用目的。

#### Shard

Collection的逻辑分片。每个Shard被化成一个或者多个replication，通过选举确定哪个是Leader。

# SolrCloud搭建

本教程的这套安装是单机版的安装，所以采用伪集群的方式进行安装，如果是真正的生产环境，将伪集群的ip改下就可以了，步骤是一样的。

SolrCloud结构图如下：



## 环境准备

CentOS-6.4-i386-bin-DVD1.iso

jdk-7u72-linux-i586.tar.gz

apache-tomcat-7.0.47.tar.gz

zookeeper-3.4.6.tar.gz

solr-4.10.3.tgz

## 环境安装

### CentOs 6.4安装

略

### jdk7安装

略

### zookeeper集群安装

**第一步**：解压zookeeper，tar -zxvf zookeeper-3.4.6.tar.gz将zookeeper-3.4.6拷贝到/usr/local/solrcloud下，复制三份分别并将目录名改为zookeeper1、zookeeper2、zookeeper3

**第二步**：进入zookeeper1文件夹，创建data目录。并在data目录中创建一个myid文件内容为“1”（echo 1 >> data/myid）。

**第三步**：进入conf文件夹，把zoo\_sample.cfg改名为zoo.cfg

**第四步**：修改zoo.cfg。

修改：

dataDir=/usr/local/solrcloud/zookeeper1/data

clientPort=2181（zookeeper2中为2182、zookeeper3中为2183）

添加：

server.1=192.168.25.154:2881:3881

server.2=192.168.25.154:2882:3882

server.3=192.168.25.154:2883:3883

|  |
| --- |
| # The number of milliseconds of each tick  tickTime=2000  # The number of ticks that the initial  # synchronization phase can take  initLimit=10  # The number of ticks that can pass between  # sending a request and getting an acknowledgement  syncLimit=5  # the directory where the snapshot is stored.  # do not use /tmp for storage, /tmp here is just  # example sakes.  dataDir=/usr/local/solrcloud/zookeeper1/data  # the port at which the clients will connect  clientPort=2181  # the maximum number of client connections.  # increase this if you need to handle more clients  #maxClientCnxns=60  #  # Be sure to read the maintenance section of the  # administrator guide before turning on autopurge.  #  # http://zookeeper.apache.org/doc/current/zookeeperAdmin.html#sc\_maintenance  #  # The number of snapshots to retain in dataDir  #autopurge.snapRetainCount=3  # Purge task interval in hours  # Set to "0" to disable auto purge feature  #autopurge.purgeInterval=1  server.1=192.168.25.154:2881:3881  server.2=192.168.25.154:2882:3882  server.3=192.168.25.154:2883:3883 |

**第五步**：对zookeeper2、3中的设置做第二步至第四步修改。

zookeeper2：

myid内容为2

dataDir=/usr/local/solrcloud/zookeeper2/data

clientPort=2182

Zookeeper3：

的myid内容为3

dataDir=/usr/local/solrcloud/zookeeper3/data

clientPort=2183

**第六步**：启动三个zookeeper

/usr/local/solrcloud/zookeeper1/bin/zkServer.sh start

/usr/local/solrcloud/zookeeper2/bin/zkServer.sh start

/usr/local/solrcloud/zookeeper3/bin/zkServer.sh start

查看集群状态：

/usr/local/solrcloud/zookeeper1/bin/zkServer.sh status

/usr/local/solrcloud/zookeeper2/bin/zkServer.sh status

/usr/local/solrcloud/zookeeper3/bin/zkServer.sh status

**第七步**：开启zookeeper用到的端口，或者直接关闭防火墙。

service iptables stop

## tomcat安装

第一步：将apache-tomcat-7.0.47.tar.gz解压

tar -zxvf apache-tomcat-7.0.47.tar.gz

第二步：把解压后的tomcat复制到/usr/local/solrcloud/目录下复制四份。

/usr/local/solrcloud/tomcat1

/usr/local/solrcloud/tomcat2

/usr/local/solrcloud/tomcat3

/usr/local/solrcloud/tomcat4

第三步：修改tomcat的server.xml

vim tomcat2/conf/server.xml，把其中的端口后都加一。保证两个tomcat可以正常运行不发生端口冲突。

## solr单机部署

### 全新部署

参考solr教案单机部署方法。

## solrCloud部署

### 启动zookeeper

solrCloud部署依赖zookeeper，需要先启动每一台zookeeper服务器。

### zookeeper管理配置文件

由于zookeeper统一管理solr的配置文件（主要是schema.xml、solrconfig.xml）， solrCloud各各节点使用zookeeper管理的配置文件。

将上边部署的solr单机的conf拷贝到/home/solr下。

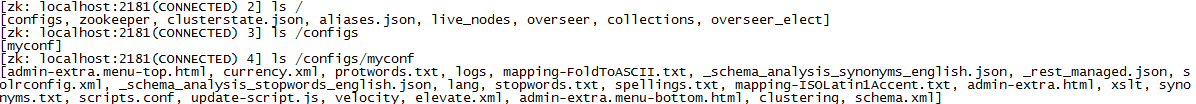
执行下边的命令将/home/solr/conf下的配置文件上传到zookeeper（此命令为单条命令，虽然很长o(╯□╰)o）。此命令在solr-4.10.3/example/scripts/cloud-scripts/目录下：

|  |
| --- |
| ./zkcli.sh -zkhost 192.168.25.154:2181,192.168.25.154:2182,192.168.25.154:2183 -cmd upconfig -confdir /usr/local/solrcloud/solrhome1/collection1/conf -confname myconf |

登陆zookeeper服务器查询配置文件：

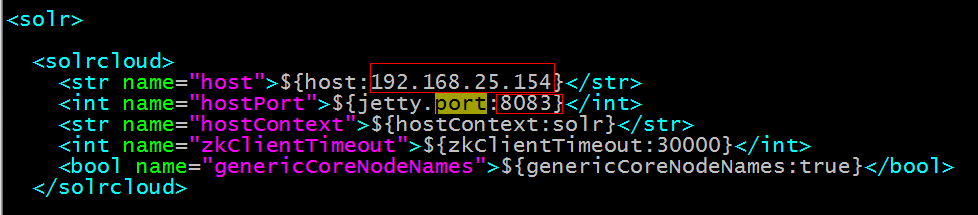
cd /usr/local/zookeeper/bin/

./zkCli.sh



### 修改SolrCloud监控端口

修改每个solrhome的solr.xml文件。将host改成虚拟机ip地址，port改成对应的tomcat的端口号。



### 每一台solr和zookeeper关联

修改每一台solr的tomcat 的 bin目录下catalina.sh文件中加入DzkHost指定zookeeper服务器地址：

JAVA\_OPTS="-DzkHost=192.168.25.154:2181,192.168.25.154:2182,192.168.25.154:2183"

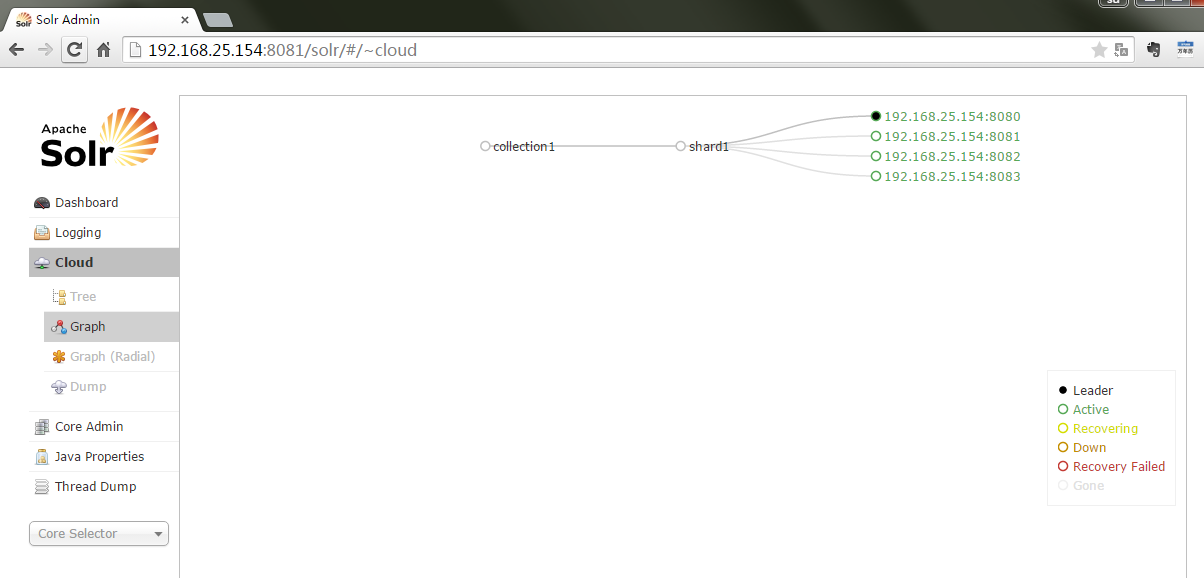
（可以使用vim的查找功能查找到JAVA\_OPTS的定义的位置，然后添加）

### 启动所有的solr服务

启动每一台solr的tomcat服务。

### 访问solrcloud

访问任意一台solr，左侧菜单出现Cloud：

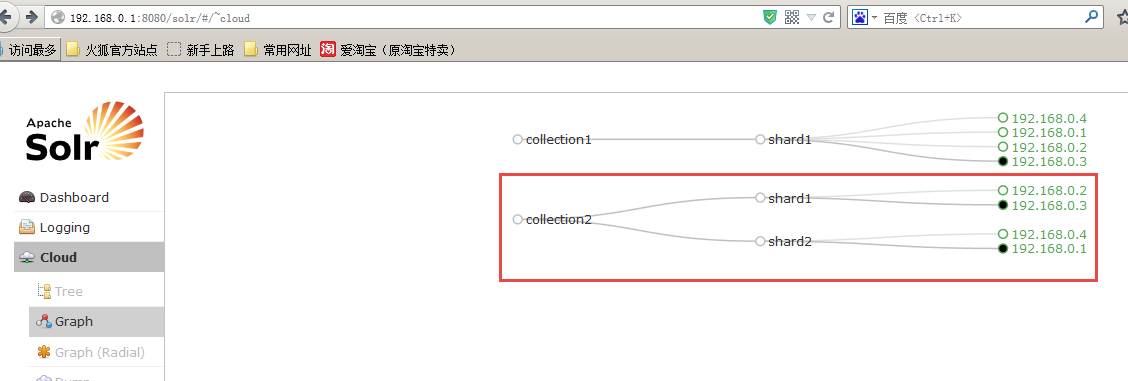


### SolrCloud集群配置

上图中的collection1集群只有一片，可以通过下边的方法配置新的集群。

如果集群中有四个solr节点创建新集群collection2，将集群分为两片，每片两个副本。

http://192.168.25.154:8080/solr/admin/collections?action=CREATE&name=collection2&numShards=2&replicationFactor=2



删除集群命令；

http://192.168.25.154:8080/solr/admin/collections?action=DELETE&name=collection1

执行后原来的collection1删除，如下：



更多的命令请参数官方文档：apache-solr-ref-guide-4.10.pdf

### 启动solrCloud注意

启动solrCloud需要先启动solrCloud依赖的所有zookeeper服务器，再启动每台solr服务器。

# solrJ访问solrCloud

**public** **class** SolrCloudTest {

// zookeeper地址

**private** **static** String *zkHostString* = "192.168.25.154:2181,192.168.25.154:2182,192.168.25.154:2183";

// collection默认名称，比如我的solr服务器上的collection是collection2\_shard1\_replica1，就是去掉“\_shard1\_replica1”的名称

**private** **static** String *defaultCollection* = "collection1";

// cloudSolrServer实际

**private** CloudSolrServer cloudSolrServer;

// 测试方法之前构造 CloudSolrServer

@Before

**public** **void** init() {

cloudSolrServer = **new** CloudSolrServer(*zkHostString*);

cloudSolrServer.setDefaultCollection(*defaultCollection*);

cloudSolrServer.connect();

}

// 向solrCloud上创建索引

@Test

**public** **void** testCreateIndexToSolrCloud() **throws** SolrServerException,

IOException {

SolrInputDocument document = **new** SolrInputDocument();

document.addField("id", "100001");

document.addField("title", "李四");

cloudSolrServer.add(document);

cloudSolrServer.commit();

}

// 搜索索引

@Test

**public** **void** testSearchIndexFromSolrCloud() **throws** Exception {

SolrQuery query = **new** SolrQuery();

query.setQuery("\*:\*");

**try** {

QueryResponse response = cloudSolrServer.query(query);

SolrDocumentList docs = response.getResults();

System.*out*.println("文档个数：" + docs.getNumFound());

System.*out*.println("查询时间：" + response.getQTime());

**for** (SolrDocument doc : docs) {

ArrayList title = (ArrayList) doc.getFieldValue("title");

String id = (String) doc.getFieldValue("id");

System.*out*.println("id: " + id);

System.*out*.println("title: " + title);

System.*out*.println();

}

} **catch** (SolrServerException e) {

e.printStackTrace();

} **catch** (Exception e) {

System.*out*.println("Unknowned Exception!!!!");

e.printStackTrace();

}

}

// 删除索引

@Test

**public** **void** testDeleteIndexFromSolrCloud() **throws** SolrServerException, IOException {

// 根据id删除

UpdateResponse response = cloudSolrServer.deleteById("zhangsan");

// 根据多个id删除

// cloudSolrServer.deleteById(ids);

// 自动查询条件删除

// cloudSolrServer.deleteByQuery("product\_keywords:教程");

// 提交

cloudSolrServer.commit();

}

}

# Mysql5.6的安装

## Mysql的卸载

### 查看是否有mysql软件：

rpm -qa|grep mysql

### 卸载mysql

yum remove mysql mysql-server mysql-libs mysql-common

rm -rf /var/lib/mysql

rm /etc/my.cnf

查看是否还有mysql软件，有的话继续删除。

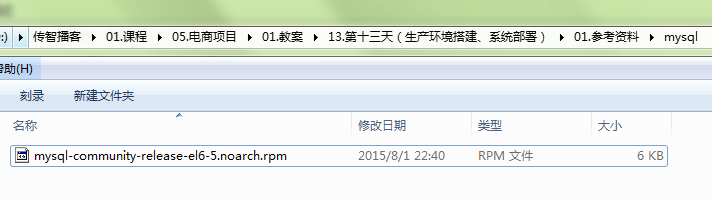
软件卸载完毕后如果需要可以删除mysql的数据库：/var/lib/mysql

## 下载rpm包

要使用yum 安装mysql，要使用mysql的yum仓库，先从官网下载适合你系统的仓库

<http://dev.mysql.com/downloads/repo/yum/>

我们是centos6.4对应的rpm包为：mysql-community-release-el6-5.noarch.rpm



## 安装仓库列表

yum localinstall mysql-community-release-el6-5.noarch.rpm

### 安装mysql

yum install mysql-community-server

### 启动mysql

sudo service mysqld start

### 设置root用户密码

mysql数据库安装完以后只会有一个root管理员账号，但是此时的root账号还并没有为其设置密码，在第一次启动mysql服务时，会进行数据库的一些初始化工作，在输出的一大串信息中，我们看到有这样一行信息 ：

/usr/bin/mysqladmin -u root password 'new-password'　　// 为root账号设置密码

### mysql远程连接授权：

GRANT ALL PRIVILEGES ON \*.\* TO 'myuser'@'%' IDENTIFIED BY 'mypassword' WITH GRANT OPTION;

注意：'myuser'、'mypassword' 需要替换成实际的用户名和密码。

## Rpm安装法

如果没有网络环境可以使用参考资料中的mysql-rpm文件夹下的mysql安装包来安装。

[root@bogon mysql-rpm]# rpm -ivh mysql-community-\*

安装后，启动服务、设置密码、远程授权后既可以使用。

# 工程部署

## 域名规划

|  |  |
| --- | --- |
| **系统** | **域名** |
| Taotao-portal | www.taotao.com |
| Taotao-rest | rest.taotao.com |
| Taotao-search | search.taotao.com |
| Taotao-sso | sso.taotao.com |
| Taotao-order | order.taotao.com |
| Taotao-manager | manager.taotao.com |

测试域名方法：

修改host文件。

## 环境准备

CentOS6.4

Jdk1.7以上

Tomcat7

## 部署步骤

使用maven实现tomcat的热部署。

### 第一步：安装tomcat

先在CentOS中安装jdk，然后解压tomcat。

### 第二步：在tomcat中配置用户权限

我们需要实现热部署,自然就需要通过maven操作tomcat,所以就需要maven取得操作tomcat的权限,现在这一步就是配置tomcat的可操作权限.

在tomcat的安装目录下,修改conf / tomcat-user.xml文件,在<tomcat-users> 节点下面增加如下配置:

|  |
| --- |
| <role rolename="manager-gui" />  <role rolename="manager-script" />  <user username="tomcat" password="tomcat" roles="manager-gui, manager-script"/> |

### 第四步：修改pom文件

在project中添加插件,以及maven中配置的server,

现在maven已经拥有操作tomcat的权限了,但是这两者之间想要通信的话还需要一个桥梁,那就是在maven中配置tomcat插件.

修改项目的pom.xml文件,在<build> 节点下面增加如下配置:

|  |
| --- |
| <build>  <plugins>  <!-- 配置Tomcat插件 -->  <plugin>  <groupId>org.apache.tomcat.maven</groupId>  <artifactId>tomcat7-maven-plugin</artifactId>  <configuration>  <port>8081</port>  <path>/</path>  <url>http://192.168.25.136:8080/manager/text</url>  <username>tomcat</username>  <password>tomcat</password>  </configuration>  </plugin>  </plugins>  </build> |

### 第五步：设置部署命令

一般使用搜是在eclipse中,可以右键点击需要部署的项目,Run as -> Run configurations -> maven build -> 右键 new,这样配置一个新的maven命令具体配置命令方法:

初次部署可以使用 "**tomcat7:deploy**" 命令

如果已经部署过使用 "**tomcat7:redeploy**" 命令

### 附相关错误及解决办法

1.Connection refused错误

报错信息如下：

[ERROR]Failed to execute goal org.apache.tomcat.maven: tomcat7-maven-plugin: 2.0- SNAPSHOT: deploy (default-cli) on project helloworld: Cannot invoke Tomcat manager: Connection refused: connect -> [Help 1]

原因：未启动Tomcat服务器

解决办法：先启动Tomcat服务器再选择Run

2. 401错误

报错信息如下:

[ERROR] Failed to execute goal org.apache.tomcat.maven:tomcat7-maven-plugin: 2.0-SNAPSHOT:deploy (default-cli) on project helloworld: Cannot invoke Tomcat manager: Server returned HTTP response code: 401 for URL: http://localhost:8080/manager/text/deploy?path=%2Fhelloworld -> [Help 1]

原因：权限问题

解决办法在$CATALINA\_BASE/conf/tomcat-users.xml，

如D:\apache-tomcat-7.0.34\conf\tomcat-users.xml文件中添加权限

<role rolename=”manager”/>

<user username=”admin” password=”admin” roles=”manager”/>

修改pom.xml文件，在<configuration> </configuration>中添加

<username>admin</username> <password>admin</password>

3.403错误

报错信息如下：

[ERROR] Failed to execute goal org.apache.tomcat.maven:tomcat7-maven-plugin: 2.0-SNAPSHOT:deploy (default-cli) on project helloworld: Cannot invoke Tomcat manager: Server returned HTTP response code: 403 for URL: http://localhost:8080/manager/html/deploy?path=%2Fhelloworld -> [Help 1]

原因：产生该问题有可能因为两个原因，具体参见解决办法

解决办法：

1）如果使用的是Tomcat 7，需要修改pom.xml中部署的url地址，将<url>http://localhost:8080/manager</url>改<url>http://localhost:8080/manager/text</url>

2）给tomcat用户权限分配上，需要同时具备manager-gui和manager-script权限，我在遇到该问题时，就是忘了分配manager-script权限。

正确的conf/tomcat-users.xml配置应为：

<tomcat-users>

<role rolename="manager-gui"/>

<role rolename="manager-script"/>

<user username="admin” password="admin" roles="manager-gui, manager-script"/>

</tomcat-users>