# 一、准备工作

## 1、到官方网站下载：Openfire（版本3.6.4）、Spark（版本2.5.8）、Smack（版本3.1.0），其中Spark只能使用SVN下载，源码的文件夹分别对应Openfire、Spark和Smack。

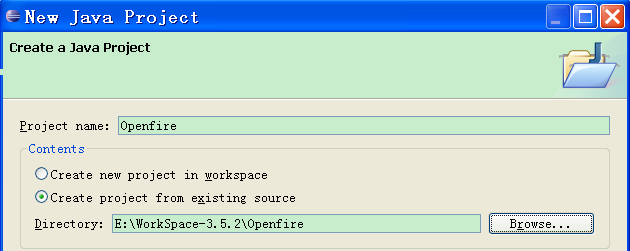
## 2、网址：<http://www.igniterealtime.org/downloads/source.jsp>，界面显示如下图：



# 二、创建Java Project项目——Openfire

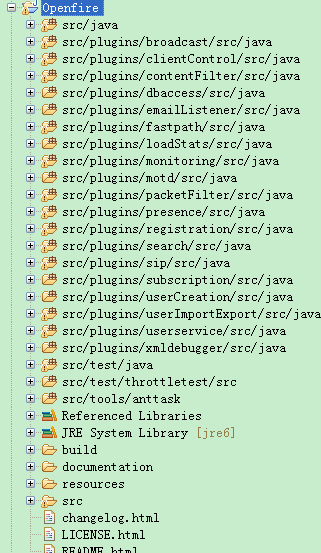
## 1、使用Eclipse-3.5.2

## 2、选择File——New——Project——Java Project。输入项目工程名Openfire，Contents中选择 "Create project from existiing source",然后把Openfire文件所在的文件夹加进去。如下图所示：



## 3、点击“Finish”，完成。

## 4、在Eclipse的Package Explorer中的显示如下图所示：



# 三、生成、编译、运行Openfire

## 1、生成Openfire

1.1点击Window——Show View——Ant；

1.2右击Ant面板,选择Add Buildfiles；

1.3展开Openfire——build文件夹,选择build.xml,点击"OK"；

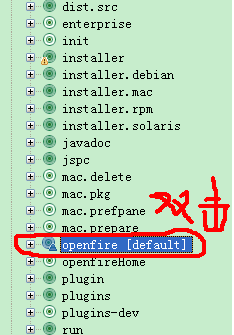
1.4在Ant面板,展开Spark,双击"release"；

1.5等一段时间,会提示"Build Successful"；

1.6生成target、work文件夹，重新编译时，要手工删除此文件夹。

注意：如果没有显示“Build Successful”，则再次双击“release”.

如下图所示：



## 2、配置Openfire运行环境

2.1点击Run——Run Configurations

2.2 点击“Main”

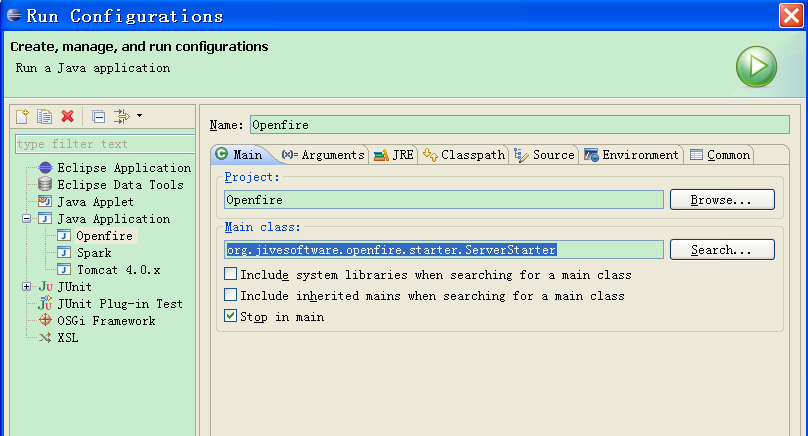
Name填写：Openfire

Project：Openfire（选择Openfire工程）

Main class：org.jivesoftware.openfire.starter.ServerStarter（启动类）

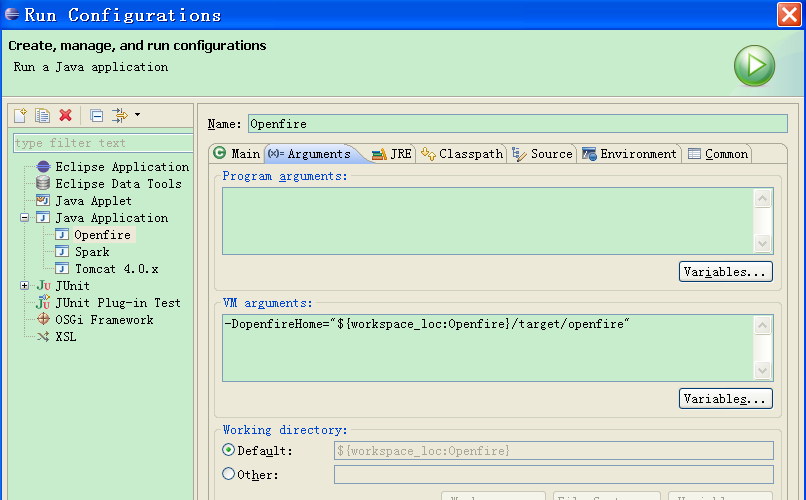
勾选Stop in main

如下图：



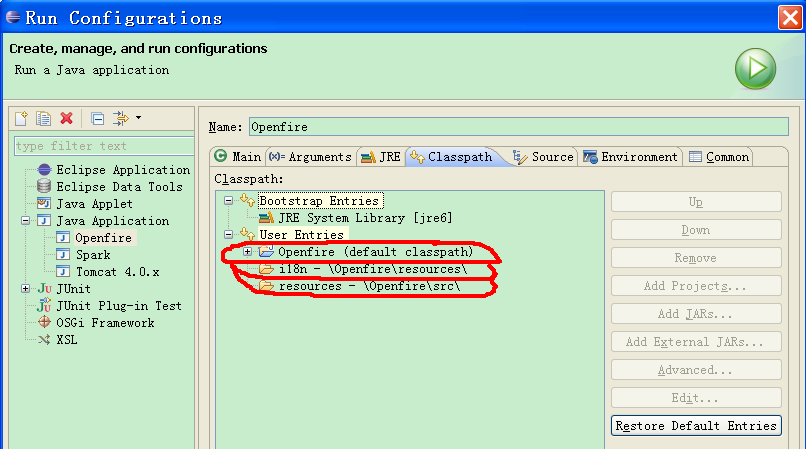
2.3 Arguments页面，在VM arguments中加下内容，如下图：

-DopenfireHome="${workspace\_loc:Openfire}/target/openfire"



2.4点击Classpath标签页

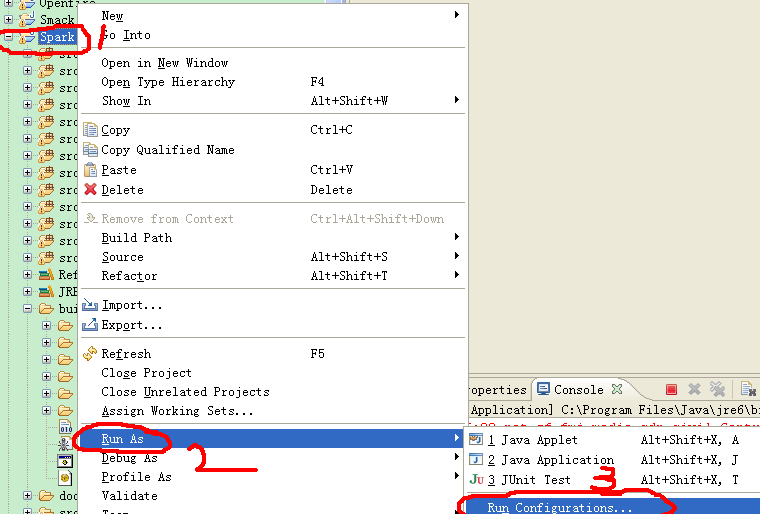
选择User Entries ,使得Advanced..按钮变的可用.点击Advanced按钮.在弹出来的Advanced Options窗口,选择Add Folders,再点OK, 在Folder Selection窗口选择Openfire/src/i18n文件夹，同样的方式选择Openfire/src/resources文件夹，点击OK。



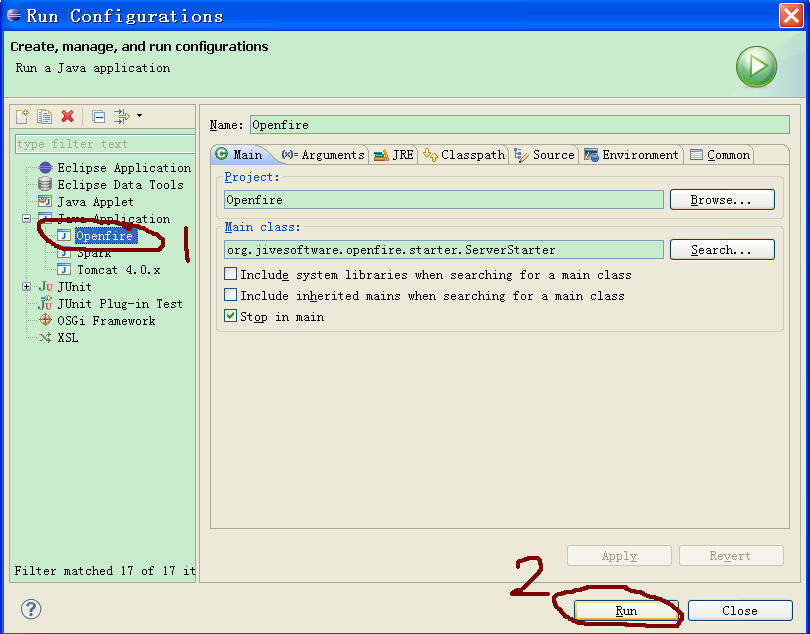
2.5选择Common标签页,勾选Debug,Run前面的框。点击Apply,再点击Close。

## 3、运行Openfire

3.1如下图



3.2 如下图，点击“run”运行Openfire



运行后，在控制台显示如下：

Openfire 3.6.4 [Aug 17, 2010 8:59:32 PM]

Admin console listening at <http://127.0.0.1:9090>

## 4、运行Openfire出现的错误，及解决方法

4.1 错误信息



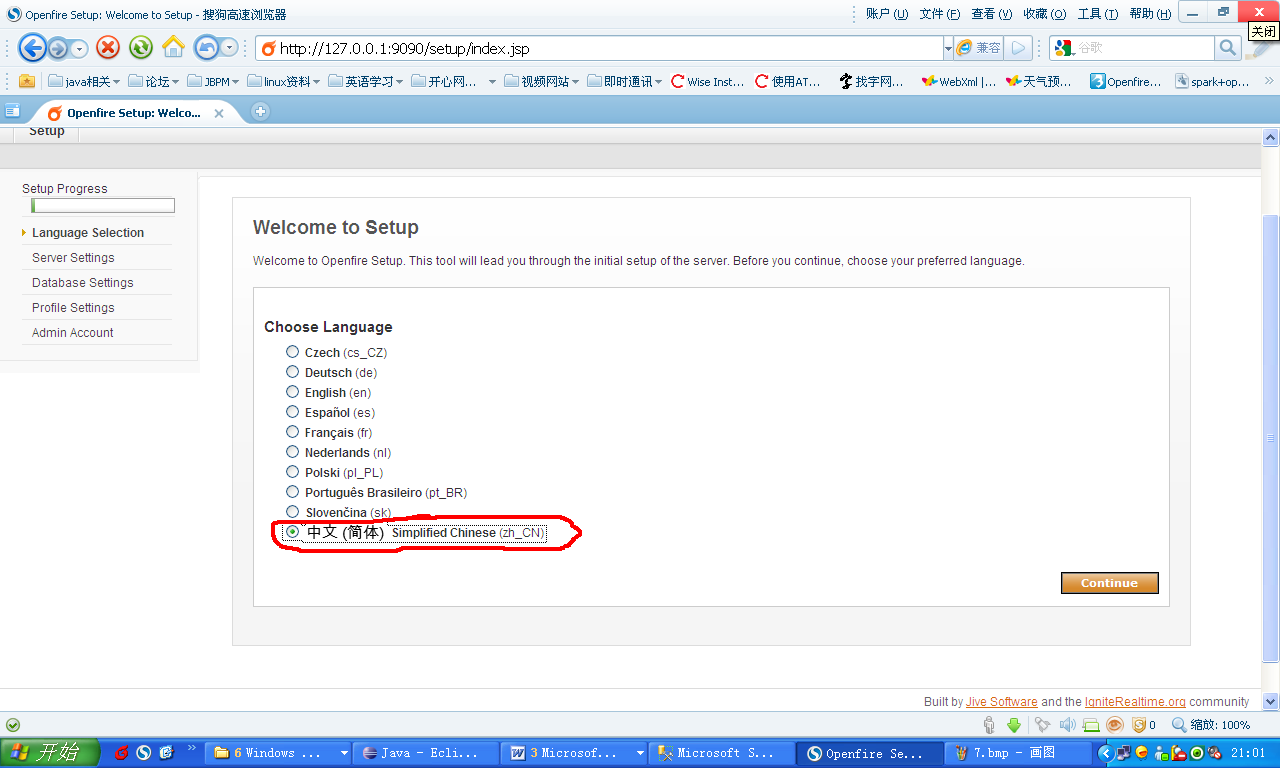
4.2 解决方法

拷贝src\i18n\openfire\_i18n\_en.properties和src\resources\jar\admin-sidebar.xml到openfire\bin目录下。其中openfire\bin目录在Eclipse中无法看到，到源目录下操做。

# 四、配置Openfire

## 1、在浏览器中打开：<http://127.0.0.1:9090>

## 2、选择“中文”，如下图



## 3、设置域名及端口号



## 4、数据库设置



## 5、数据库设置



## 6、特性设置



## 7、管理员账户设置

## 13

## 8、完成安装



## 9、重启Openfire，控制台显示：

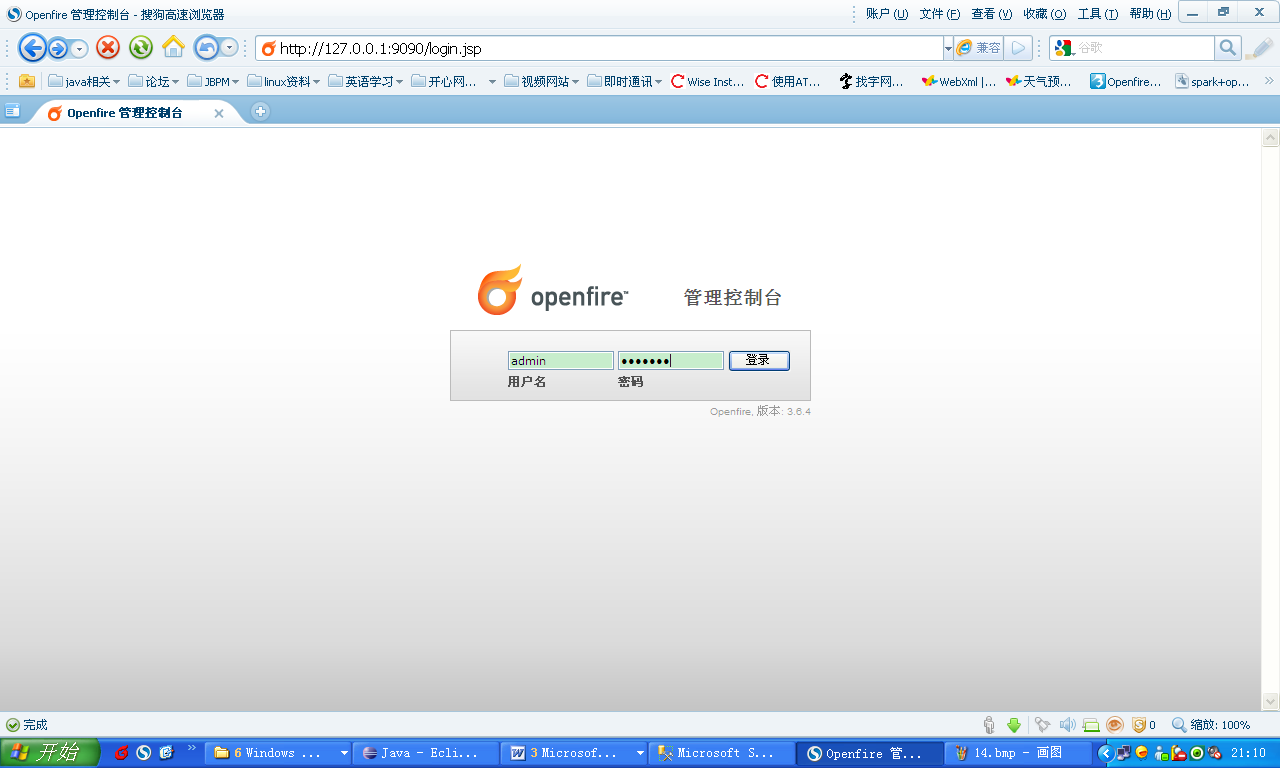
Openfire 3.6.4 [2010-8-17 21:09:43]

管理平台开始监听:

http://dibin:9090

<https://dibin:9091>

## 10、打开：<http://dibin:9090>，输入用户名：admin，和密码：刚才设置



## 11、进入管理平台界面



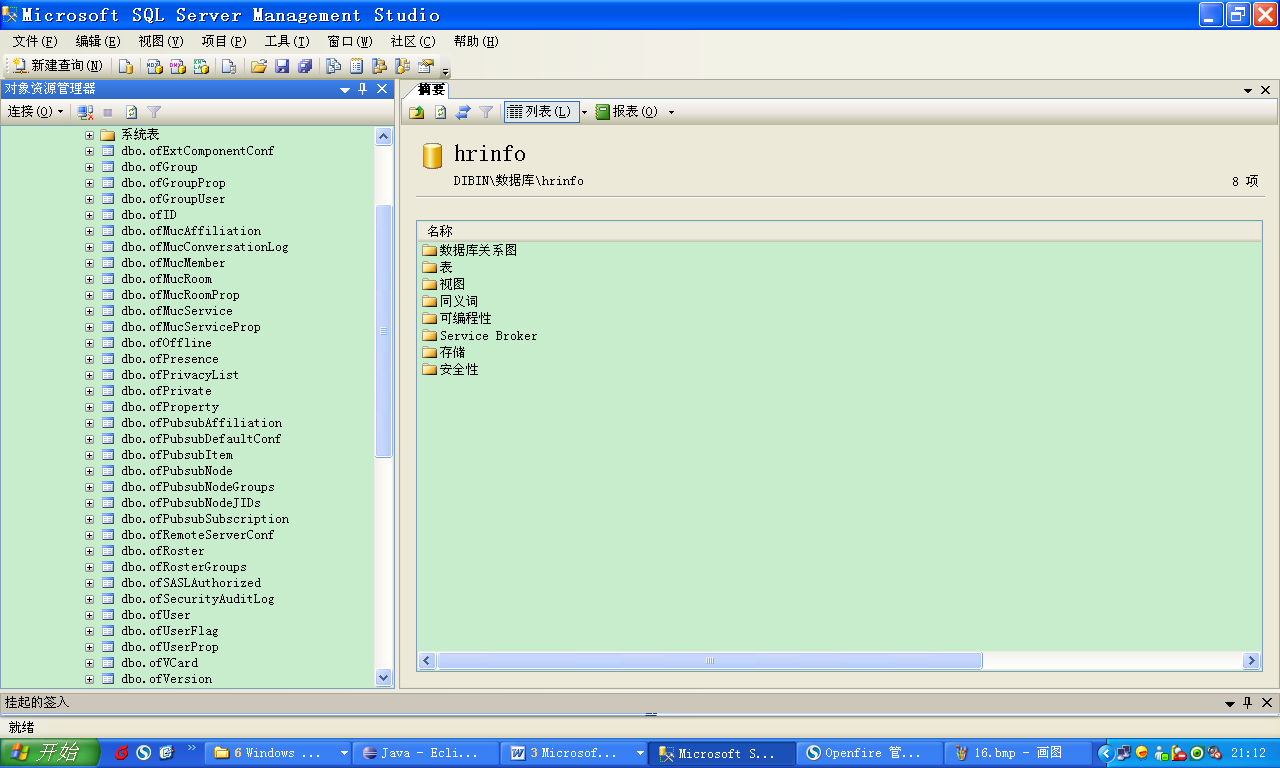
# 五、删除或卸载Openfire

## 1、删除生成的target、work文件夹；

## 2、删除数据库中生成的表，见[附录1](#_附录1：Openfire安装过程中创建的数据库表结构，如下图)；

## 3、重复[三](#_三、生成、编译、运行Openfire)、[四](#_四、配置Openfire)的步骤，进行安装。

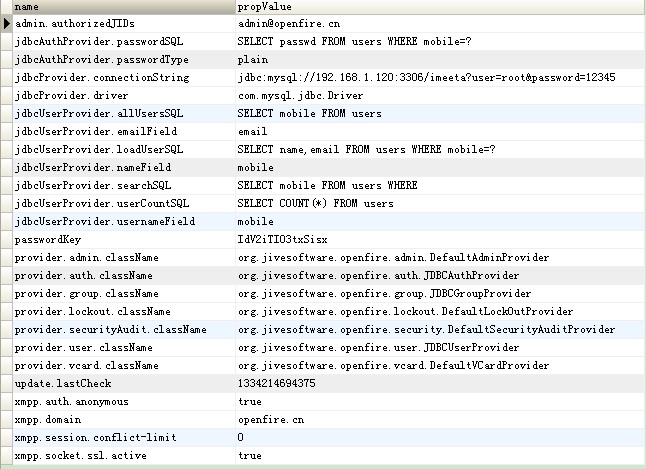
# 附录1：Openfire安装过程中创建的数据库表结构，如下图



# 六、配置Openfire服务器使用自己的用户表（<http://www.4ucode.com/Study/Topic/333201>）

openfire扩展小试 整合现有系统用户   
  
如果我想使用现有系统的用户/组(部门)，而不想使用openfire再去管理一套用户/组，用openfire可以非常方便的整合现有系统用户。   
  
进入openfire管理控制台-服务器－服务管理器－系统属性   
可以发现如下配置   
provider.auth.className   
org.jivesoftware.openfire.auth.DefaultAuthProvider   
\*用户验证   
provider.group.className   
org.jivesoftware.openfire.group.DefaultGroupProvider   
\*获取组相关数据   
provider.user.className   
org.jivesoftware.openfire.user.DefaultUserProvider   
\*获取用户相关数据   
这些Provider是openfire默认自己管理用户组   
  
但同时openfire还提供了支持JDBC相关的Provider，可以从其它的数据源获取用户/组数据   
将上面三个属性分别修改为   
org.jivesoftware.openfire.auth.JDBCAuthProvider   
org.jivesoftware.openfire.group.JDBCGroupProvider   
org.jivesoftware.openfire.user.JDBCUserProvider   
  
然后，需要配置一下数据源，添加如下属性   
jdbcProvider.driver   
\*数据源驱动   
jdbcProvider.connectionString   
\*连接字符串   
  
对每一个JDBC Provider需要配置相关的SQL语句和属性（在系统属性里添加项目）   
  
JDBCAuthProvider   
jdbcAuthProvider.passwordSQL   
\*获取用户密码的SQL   
\*输入参数：登录名   
\*输入列：密码   
\*例：SELECT pwd FROM user WHERE name=?   
jdbcAuthProvider.passwordType   
\*密码类型可以是:plain(文本),md5,sha1   
\*如果你的密码加密不为以上三种 就需要自己提供一个AuthProvider，在下一章会专门介绍   
  
  
JDBCGroupProvider   
jdbcGroupProvider.allGroupsSQL   
\*获取所有组的SQL   
\*输入参数：无   
\*输出列:组的KEY   
\*例：SELECT sn FROM department   
  
jdbcGroupProvider.descriptionSQL   
\*获取组的名称（描述）   
\*输入参数：组记录的KEY   
\*输出列:组的名称（描述）   
\*例：SELECT name FROM department where sn=?   
  
jdbcGroupProvider.groupCountSQL   
\*获取组的数量   
\*输入参数：组的KEY   
\*输出列:组的数量   
\*例：SELECT count(sn) FROM department   
  
jdbcGroupProvider.loadAdminsSQL   
\*获取组的管理员   
\*输入参数：组记录的KEY   
\*输出列:组管理员的KEY   
\*例：SELECT admin FROM department where sn=?   
  
jdbcGroupProvider.loadMembersSQL   
\*获取组的成员   
\*输入参数：组的KEY   
\*输出列:组成员的KEY（集合）   
\*例：SELECT usersn FROM department\_user where departmentsn=?   
  
jdbcGroupProvider.userGroupsSQL   
\*获取成员的组   
\*输入参数：成员的KEY   
\*输出列:成员所性组的KEY   
\*例：SELECT departmentsn FROM department\_user where usersn=?   
  
  
  
JDBCUserProvider   
jdbcUserProvider.allUsersSQL   
\*获取所有用户   
\*输入参数：无   
\*输出列:用户的KEY   
\*例：SELECT sn from user   
  
jdbcUserProvider.userCountSQL   
\*获取所有用户数量   
\*输入参数：无   
\*输出列:用户数量   
\*例：SELECT count(sn) from user   
  
jdbcUserProvider.loadUserSQL   
\*获取用户信息   
\*输入参数：用户的KEY   
\*输出列:登录名，名称,email(至少应该这三列，下面要用到)   
\*例：SELECT loginname,name,email from user where sn =?   
  
jdbcUserProvider.emailField   
\*指定用户email的列名如：email   
jdbcUserProvider.nameField   
\*指定用户名称的列名如：name   
jdbcUserProvider.usernameField   
\*指定用户登录名的列名如：loginname   
  
  
最后 还需要配置新的管理员用户   
admin.authorizedJIDs   
\*指定新数据源中的管理员用户注意是是完整JID(user@域名)   
\*例:admin@server.cn   
  
配置好如上属性 重启openfire   
使用admin.authorizedJIDs中的用户名登录openfire管理控制台   
如果配置成功，进入openfire管理控制台-用户/组   
就可以看到你数据源中的用户/组信息了   
同时可以使用spark登录进行测试

也可以找到Openfire的系统配置表 ofproperty 更改以上配置信息：



# 七、更换Openfire服务器数据库的IP地址（<http://blog.csdn.net/happysheepherder/article/details/4707124>）

如果你的服务器名称和mysql的地址都是使用的静态ip地址配置的，更改ip后，openfire就会开启失败，这种情况下请看下面的解决方法。

    比如你的ip地址由 192.168.0.111 改为192.168.0.222后，openfire开启失败，控制台会出现一些红字，openfire database error …. ，解决方法：

打开E:/Program Files/Openfire\target\openfire\conf\openfire.xml, 修改serverURL>jdbc:mysql://192.168.0.111:3306/openfire 字段中的ip更改为新的ip，其实这里最好写成 127.0.0.1 比较好，这样就始终指向本机。

    这样mysql数据库连接成功，openfire管理界面可以正常登陆，但是客户端还是无法连接openfire，这是由于服务器名称还是以前的ip造成的，解决方法：

1.       登陆openfire管理页面，在主页面下方选择编辑服务器属性，修改新的服务器名称为新的ip地址，也就是192.168.0.222，点击保存属性，页面提示从启服务器。

2.       重启后服务器名称出现一个叹号，鼠标放上去显示Found RSA certificate that is not valid for the server domain， 这样由于RSA认证无效造成的，需要对新的ip地址进行RSA证书的配置。

3.       选择【服务器配置】菜单，选择左下方的【服务器证书】，会看到两个证书，点击后面的删除按钮，全部删除，删除后系统提示重启服务器，点击重启

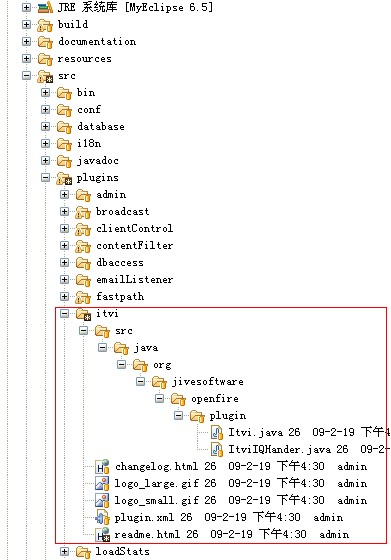
4.       重启后，系统提示“一个或更多的证书丢失。单击[这里](http://127.0.0.1:9090/ssl-certificates.jsp?generate=true)产生自定义签名证书”，点击这里，自动生成和新的ip匹配的RSA证书，生成后，系统提示重启。

5.       再次登录后，会看到主界面的服务器名称的叹号消失了，openfire正常，客户端可以正常登陆了

# 八、Openfire的插件的添加和编译（<http://jimmy-duan.iteye.com/blog/921117>）

一，src/plugins目录下新建文件夹itvi。并按下图的目录结构新建对应的文件。

如图：



上图中plugin.xml文件代码如下：

<?xml version="1.0" encoding="UTF-8"?>

<plugin>

<class>org.jivesoftware.openfire.plugin.Itvi</class>

<name>Itvi</name>

<description>Broadcasts messages to users.</description>

<author>openfireteam</author>

<version>1.7.0</version>

<date>2/19/2009</date>

<url>http://openfireteam.blogjava.net</url>

<minServerVersion>3.6.3</minServerVersion>

</plugin>

二，如果是开发好的插件包直接把插件添加到上图的plugins文件下

三，打包插件

a.第一次打包需配置ant工具。在环境变量中，

添加 path=C:\Program Files\Java\jdk1.5.0\_09\;D:\Program Files\MyEclipse6.5\eclipse\plugins\org.apache.ant\_1.7.0.v200706080842\bin (如果原来有path,在变量值的尾部添加半角分号 ; 然后再加入上面的jdk和ant路径 )

b.编译

进入cmd，进入 openfire／build所在目录：输入命令：ant plugins 等待执行完，而且没有报错，说明打包成功。在openfire3.6.3\target\openfire\plugins目录下就生成了itvi.jar

# 九、Openfire 删除已经编译的插件。

一，在target目录的openfire下找到对应的插件文件删除。

二，在work目录下lib中找到对应的插件jar包删除。

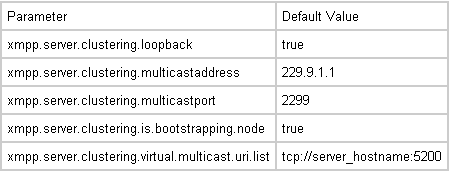
# 十、Openfire 集群配置。

一，集群插件下载网站（<http://code.google.com/p/open-clustering/>）

把下载的clustering.war包放到src\plugins下，如果有target也加一份到plugins下，work也一样。

二，配置系统属性（ofProperty）：

共有一个数据库时：配置如下，添加下面属性：

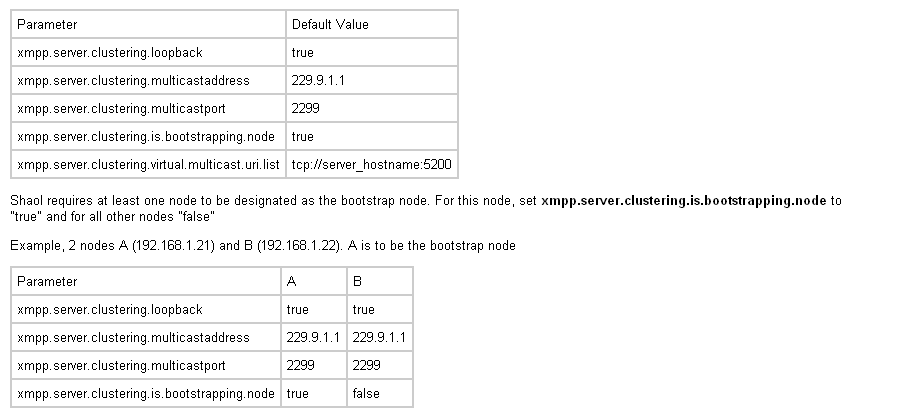


其中xmpp.server.clustering.is.bootstrapping.node 的值为 false。

Xmpp.server.clustering.virtual.multicast.uri.list 的值为：tcp://192.168.50.54:5222.

(tcp://openfire的主机ip地址：客户端连接端口号)

三，不同的数据库：



变化就是xmpp.server.clustering.is.bootstrapping.node，一个为true，一个为false。

# 十一、eclipse编译Openfire源码命令行部署运行。

一，先要为eclipse配置log4j。

在build\lib\dist\log4j.xml文件修改成：

<?xml version="1.0" encoding="UTF-8" ?>

<!DOCTYPE log4j:configuration SYSTEM "log4j.dtd">

<log4j:configuration xmlns:log4j="http://jakarta.apache.org/log4j/">

<!-- add -->

<appender name="STDOUT" class="org.apache.log4j.ConsoleAppender">

<layout class="org.apache.log4j.PatternLayout">

<param name="ConversionPattern"

value="%d [%24F:%-4L:%-5p][%x] -%m%n"/>

</layout>

</appender>

<!-- -->

<appender name="debug-out" class="org.apache.log4j.RollingFileAppender">

<param name="File" value="${openfireHome}/logs/debug.log" />

<param name="MaxFileSize" value="1024KB"/>

<param name="MaxBackupIndex" value="5"/>

<layout class="org.apache.log4j.PatternLayout">

<param name="ConversionPattern" value="%d{yyyy.MM.dd HH:mm:ss} %c - %m%n" />

</layout>

<filter class="org.apache.log4j.varia.LevelRangeFilter">

<param name="LevelMax" value="debug" />

<param name="AcceptOnMatch" value="true" />

</filter>

</appender>

<appender name="info-out" class="org.apache.log4j.RollingFileAppender">

<param name="File" value="${openfireHome}/logs/info.log" />

<param name="MaxFileSize" value="1024KB"/>

<param name="MaxBackupIndex" value="5"/>

<layout class="org.apache.log4j.PatternLayout">

<param name="ConversionPattern" value="%d{yyyy.MM.dd HH:mm:ss} %c - %m%n" />

</layout>

<filter class="org.apache.log4j.varia.LevelRangeFilter">

<param name="LevelMax" value="info" />

<param name="LevelMin" value="info" />

<param name="AcceptOnMatch" value="true" />

</filter>

</appender>

<appender name="warn-out" class="org.apache.log4j.RollingFileAppender">

<param name="File" value="${openfireHome}/logs/warn.log" />

<param name="MaxFileSize" value="1024KB"/>

<param name="MaxBackupIndex" value="5"/>

<layout class="org.apache.log4j.PatternLayout">

<param name="ConversionPattern" value="%d{yyyy.MM.dd HH:mm:ss} %c - %m%n" />

</layout>

<filter class="org.apache.log4j.varia.LevelRangeFilter">

<param name="LevelMax" value="warn" />

<param name="LevelMin" value="warn" />

<param name="AcceptOnMatch" value="true" />

</filter>

</appender>

<appender name="error-out" class="org.apache.log4j.RollingFileAppender">

<param name="File" value="${openfireHome}/logs/error.log" />

<param name="MaxFileSize" value="1024KB"/>

<param name="MaxBackupIndex" value="5"/>

<layout class="org.apache.log4j.PatternLayout">

<param name="ConversionPattern" value="%d{yyyy.MM.dd HH:mm:ss} %c - %m%n" />

</layout>

<filter class="org.apache.log4j.varia.LevelRangeFilter">

<param name="LevelMin" value="error" />

<param name="AcceptOnMatch" value="true" />

</filter>

</appender>

<!-- add -->

<logger name="java.sql.Connection">

<level value="DEBUG" />

</logger>

<!-- -->

<root>

<level value="info" />

<!-- add -->

<appender-ref ref="STDOUT" />

<!-- -->

<appender-ref ref="debug-out" />

<appender-ref ref="info-out" />

<appender-ref ref="warn-out" />

<appender-ref ref="error-out" />

</root>

</log4j:configuration>

二，制作运行脚本：

添加openfire依赖的jar包和文件：

d:

cd /openfire/bin

set classpath=.;%CLASSPATH%;D:\Openfire\build\lib\dist\activation.jar;D:\Openfire\build\lib\dist\bouncycastle.jar；D:\Openfire\build\lib\dist\log4j.xml;D:\Openfire\src\i18n;D:\Openfire\build\recouce;

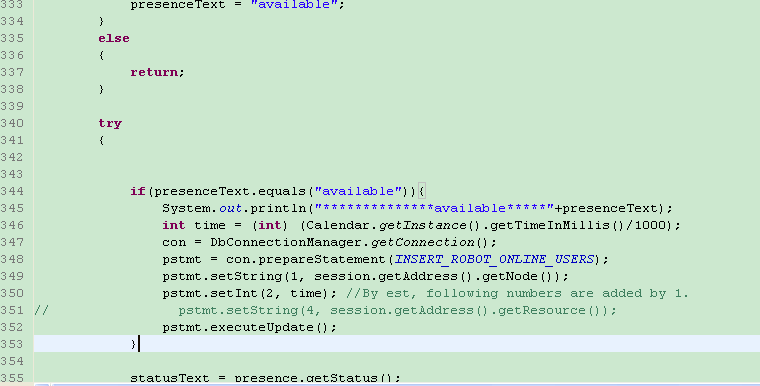
(要把openfire所有引用jar包和文件全部引用，如果安装过程中有错，可以根据日志导入未加入的包)

添加启动类的路径并且启动openfire

java -DopenfireHome=/openfire/target/openfire org.jivesoftware.openfire.starter.ServerStarter.

# 十二、Openfrie记录用户登录状态，userstatus插件。

1. openfire运行userstatus插件。先把userstatus插件报到工程src目录下的plugins目录中，在命令行下找到工程目录（Openfire）下，找到build目录下，输入命令ant plugins编译插件。编译成功后，刷新工程，如果在工程target和work目录下找到插件名的jar包，就证明编译成功。然后运行工程，如果数据库中自动生成userstatus表证明插件运行成功。
2. 修改userstatus插件代码，实现往robot\_online\_user表里插入数据。首先自己要在数据库中创建robot\_online\_user表，字段uid，active\_time。



在340行try – catch下加入if（presenceText.equals(“available”)）{}，然后重新ant plugins 编译运行。

# 十三、Openfrie记录用户登录用户聊天记录。

一，在数据库中加入chathistory表：字段：user\_id, target\_id, message, status, dateline

主要数据库操作类：ChatHistory.java:

package org.jivesoftware.openfire;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.Calendar;

import java.util.List;

import org.jivesoftware.database.DbConnectionManager;

import org.jivesoftware.openfire.user.User;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

@SuppressWarnings("unused")

public class ChatHistory {

private static final Logger Log = LoggerFactory.getLogger(User.class);

private static final String INSERT\_PROPERTY = "INSERT INTO chatHistory (user\_id, target\_id, message, status, dateline) VALUES (?, ?, ?, ?, ?)";

private static final String LOAD\_PROPERTY = "SELECT target\_id FROM chatHistory WHERE user\_id=?";

private static final String LOAD = "SELECT user\_id FROM chatHistory"+ " WHERE target\_id=?";

private String user\_id;

private String target\_id;

private String message;

private String status;

@SuppressWarnings("unchecked")

public List<String> getPropertyValue(String user, String domain) {

List<String> list = new ArrayList();

Connection con = null;

PreparedStatement pstmt = null;

ResultSet rs = null;

try {

con = DbConnectionManager.getConnection();

pstmt = con.prepareStatement(LOAD\_PROPERTY);

pstmt.setString(1, user);

rs = pstmt.executeQuery();

while (rs.next()) {

list.add(rs.getString(1));

}

pstmt = con.prepareStatement(LOAD);

pstmt.setString(1, user);

rs = pstmt.executeQuery();

while (rs.next()) {

list.add(rs.getString(1));

}

} catch (SQLException sqle) {

Log.error(sqle.getMessage(), sqle);

sqle.printStackTrace();

} finally {

DbConnectionManager.closeConnection(rs, pstmt, con);

}

return list;

}

public void createChatHistory(String user\_id, String target\_id, String message, String status) {

Connection conn = null;

PreparedStatement pstmt = null;

try {

conn = DbConnectionManager.getConnection();

pstmt = conn.prepareStatement(INSERT\_PROPERTY);

int dateline = (int) (Calendar.getInstance().getTimeInMillis()/1000);

pstmt.setString(1, user\_id);

pstmt.setString(2, target\_id);

pstmt.setString(3, message);

pstmt.setString(4, status);

pstmt.setInt(5, dateline);

pstmt.executeUpdate();

} catch (SQLException sql) {

sql.printStackTrace();

Log.error(sql.getMessage(), sql);

System.out.println("error");

} finally {

DbConnectionManager.closeConnection(pstmt, conn);

}

}

public ChatHistory() {

}

public ChatHistory(String user\_id, String target\_id, String message, String status) {

if (user\_id == null) {

throw new NullPointerException("user\_id cannot be null");

}

this.user\_id = user\_id;

if (target\_id == null) {

throw new NullPointerException("target\_id cannot be null");

}

this.target\_id = target\_id;

this.message = message;

this.status = status;

}

public boolean isOnlineFromTo(String target\_id){

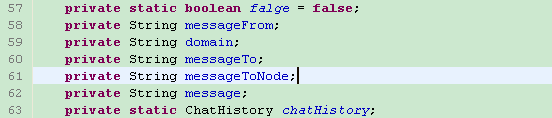
return false;

}

}

源码MessageRouter.java修改，添加对消息记录的入库：

主要修改地方：



添加以上变量，falge 为标识变量，区分消息“未送达”和“已送达”状态。

messageFrom，谁发的消息

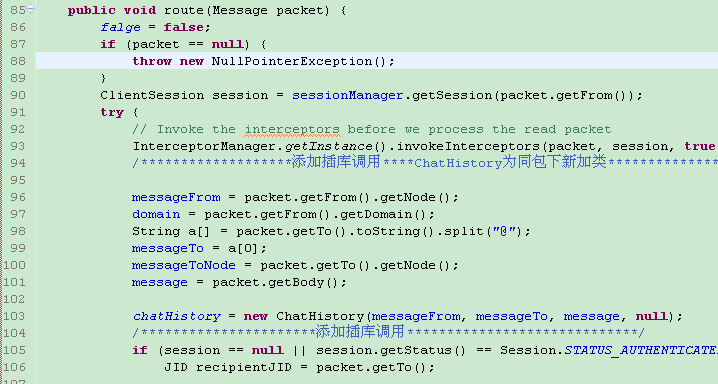
domain，为域名。

Messageto，消息发给谁

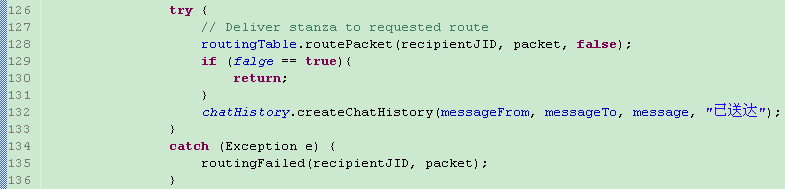
Message，消息内容，

messageToNode，发送消息的服务器节点，这里没有用到。

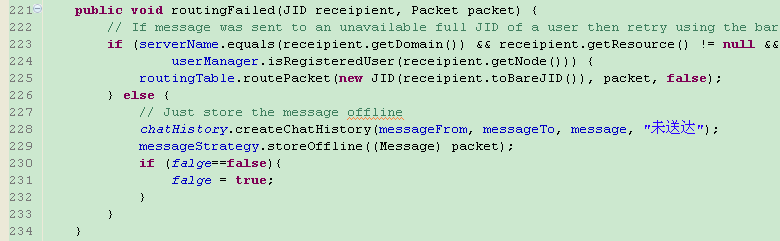
chatHistroy，数据可操作类。



这段代码添加是初始化数据库操作类，获取数据信息。先把falge=false，为默认消息已送达。



如果falge=true证明消息为送达，不再执行保存“已送达”的操作。



如果上一部有异常就会执行routingFailed(JID receipient, Packet packet)

在else中加入“未送达”的保存，然后把falge=true表明消息未送达。