**即时消息设计**

1. 主要内容

即时消息设计主要用于人员管理，方便团队合作、交流、质询和任务分配。

1. 需求设计
   1. 概要内容通过即时消息软件可以，在局域网内管理团队成员，用于传递消息和安排任务。
   2. 主要技术window From socket 技术
   3. 设备需求操作系统：xp、window7 、window8  
      运行环境：net.frameword 2.0
   4. 软件结构 用户通过软件可以收发消息，用户可以根据权限想自己所在组员发送消息，组长可以给自己的员工群发消息，在线用户可以同步收到消息，不在线用户登录后查看消息。针对于管理员可以查看所有用户的聊天记录。管理员可以配置组成员，分配组长，分配权限给用户。

人员管理

用户

数

消息管理

人员权限管理

* 1. 功能说明

1. 登陆系统

|  |
| --- |
|  |

根据后台的管理员配置用户登录到系统。  
功能：

* 记住密码：方便快捷登录。
* 自动登录：开机自动登录。
* 登 录 ：登录系统页面
* 忘记密码：跳转到忘记密码窗口

1. 用户权限管理

|  |
| --- |
|  |

管理员分配权限给组长或者指定用户。

功能：

角色配置：配置角色权限  
 应用模块：配置应用权限。

1. 用户管理和组员配置

|  |
| --- |
|  |

功能：

|  |  |  |
| --- | --- | --- |
| 图标 | 功能名称 | 说明 |
|  | 组管理 | 增删改 组信息 |
|  | 队友管理 | 增删改 组信息 |

1. 消息收发

|  |
| --- |
|  |

消息收发界面和队友管理查看界面。

功能：

|  |  |  |
| --- | --- | --- |
| 图标 | 功能名称 | 说明 |
|  | 队友查看 |  |
|  | 小组查看 | 针对组长和管理员或者有权限的用户。 |
|  | 文件共享 | 文件共享到队友成员里 |
|  | 刷新成员列表 | 刷新队友在线状态 |
|  | 管理权限和配置用户管理入口 |  |
|  | 帮助手册 |  |
|  | 搜索 | 搜索队友或者小组 |
|  | 用户头像 | 设置自己头像 |
|  | 设置用户状态 | 设定当前用户状态 |
|  | 用户名 |  |
|  | 天气 |  |
|  | 当前在线用户数 |  |
|  | 最近联系人 |  |

**即使消息设计（细节分析）**

jabber客户端：

配置openFire服务器

到openfire官网下载源码或exe安装程序，此处选择不带jre安装版本。官网下载地址

解压缩找到openfire.exe运行

出现，此处乱码为设置字符简体中文后原因，点击launch Admin

或http://127.0.0.1:9090进行初始设置. 第一步选择语言

第二步选择域、服务器管理控制端口

第三步数据库设置，此处选择标准连接--外部数据库（导入所需要的连接外部数据库lib包，如本次连接oracle导入ojdbc14.jar到openfire）

第四步配置数据库驱动、url、用户名密码等

第五步填写管理员密码（账号默认为admin）

配置完成重启服务器登陆，首次进入页面显示为

此处登陆我们使用的是openfire自建的表ofuser中数据，但是我们需要关联整合本地的用户，需要在表ofproperty中做配置或者直接使用页面服务器管理器→系统属性中添加

属性修改

新增下列几项属性NAME,propValue

'admin.authorizedJIDs','admin@liuxin'

'jdbcAuthProvider.passwordSQL','select password from tu\_userinfo where account=?'

'jdbcProvider.connectionString','jdbc:oracle:thin:fnchl/fnchl@172.16.16.164:1521:orcl'

'jdbcProvider.driver','oracle.jdbc.driver.OracleDriver'

.authorizedJIDs：”admin@安装时候你填写的域”中的 admin是必须要在你需要整合的数据库用户表中存在http://www.wendangxiazai.com/b-8b22b8345a8102d276a22f4e.html的用户。这是用来登陆你的openfire后台管理的

.passwordSQL:查询用户登陆密码sql

.connectionString：使用jdbc的Url

.driver：整合数据库驱动

还有若干属性参考官方文档

作为jabber服务器支持大多数的jabber客户端，建立基于支持XMPP协议的即时通讯系统，当然openfire自己网站上有客户端spark，但是需要安装才能实现聊天，大多数的资料显示其对jwchat支持好，同样也支持ijab，我个人感觉这两款即时聊天符合当前的无需安装客户端即可通讯要求，支持私聊、群聊、留言、在线离线状态等，但是网上的资料都非常的少。但是如果是单纯的聊天功能，jwchat和openfire就能很好的完成，但是我们需要的是将IM的提醒功能当作一个接口提供给应用系统。同时用资料配置后发现对jwchat客户端配置一直未成功。如果需要使用此类框架需要做大量改造，可能是有点得不偿失。

技术：

基于http长连接的服务器推送技术称之为comet，将数据直接从服务器推送到浏览器，而不必等到浏览器请求数据。网络上的大多数也是基于此类似功能实现：webqq，开心网。

用户请求会在服务器产生大量并发的长连接，这样会对服务器性能产生影响，需要对长连接做优化处理，具体影响性能的程度还没能测试。

对comet有较好的支持，可实现在线聊天.其是使用长轮询(long-polling)来实现的在线聊天。使用jetty的特性和dwr使用来开发comet程序会比tomcat优秀。(具体参考资料)

所做的例子已经能完成聊天功能，对于离线发送留言还未实现，个人认为还是可以实现的。

基于我们的用户门户和专家门户是两个系统，dwr的方案中有支持跨域访http://www.wendangxiazai.com/b-8b22b8345a8102d276a22f4e.html问方法，我们是否可以通过分别部署两个dwr然后通过跨域远程调用不同系统的方法来实现发送消息等功能。

来做基于长连接的聊天原理为：

首先我们通过请求request获取dwr的上下文ServerContext

.get(request.getSession().getServletContext())；

通过servercontext. getScriptSessionsByPage(&quot;/chat/index.jsp&quot;)获取所有连接

.jsp网页的用户的scriptsession，然后再通过此scriptsession连接对所有的用户发出通知。

页面需要导入相应的/dwr/engine.js（必须），/dwr/util.js（可选），

/dwr/interface/ChatManager.js（后台java类名为ChatManager，前台需写入相应js），还有些其他需要注意的可以参考文档。

注意：dwr3.0的js引入engine.js位置要在ChatManager.js前，不然会报错

一对一聊天：

首先将httpsessionid作为聊天用户的id，同时也使用一个map将sessionid与

绑定，这样方便我们在用户退出的时候根据sessionid刷新map中的有效scriptsession。同时使用DWR中的Util类把在线列表的人员的sessionid以名字显示，如下：util.addOptions(&quot;users&quot;, users, &quot;userid&quot;,&quot;username&quot;);

参考流程：

发送消息人（sender）发送消息，js调用后台java程序send发送消息方法,传入参数sender, receiverid(接收者的id,即sessionid), msg(消息), time(发送时间), text(先前聊http://www.wendangxiazai.com/b-8b22b8345a8102d276a22f4e.html天记录),通过我维护的在线有效scriptsession绑定的map，用map来认证receiverid是否有效，如果无效我们就将消息存入数据库方便用户下次上线提取。

如果有效我们就通过此receiverid拿到的scriptsession来获取跟接收者的通话句柄，将消息发送给接收者，其中同样可以将每次发送的消息保存数据库方便用户查询聊天记录。

一对多：

一对一聊天中已经可以实现一对多的聊天，但是对于一是将消息全部显示在一个文本域（textarea）中，这肯定是不行的，我的想法是每次对于不同的发送者就创建一个单独div聊天窗口，可以实现。

多对多的聊天室功能暂不予考虑实现。

下线：

.0版本中客户端js中可加入

.engine.setActiveReverseAjax(true)【使用反转ajax】，

.engine.setNotifyServerOnPageUnload(true)【3.0版本特有，页面刷新或退出等动作注销当前scriptsession。dwr2.0版本我使用servercontext.

(&quot;/chat/index.jsp&quot;)方法会将我已经退出的页面的无效scriptsession取得，可能这个获取就会过滤】

参考流程：

因为本系统应该是不允许刷新的，可以不需要考虑刷新的维护map的scriptsession，我们只需要考虑退出后删除退出用户。前台用onbeforeunload（）函数，在刷新或者退出时会调用，同时将userid（sessionid）传入后台。根据userid我们能维护在线列表list，最新scriptsession的map，然后getScriptSessionsByPage获取获取所有会话句柄将当前在线人数传递给各个用户。

离线消息：

同样是将消息、接收者传递到后台再存入数据库，实现起http://www.wendangxiazai.com/b-8b22b8345a8102d276a22f4e.html来过程是比较简单。就是还未想通如何区分名字相同的两个人，或者我们可以直接用sessionid跟用户实体对象绑定区分各个聊天对象之间的不同

在线人数：

在用户首次登录系统获取在线人数、每次关闭网页更新在线人数来维护在线人数模块，但是还有疑问：用户门户的用户可以显示所有的在线专家和离线专家人数。但是专家门户是否全部显示用户门户的所有人，那样到后期聊天列表会相当的长，这样设计是

不可取的。或者可以直接在专家门户保留最后多少个联系人，原型上好像是如此做的。

聊天表情：

聊天表情方面我想参考qq表情的实现

参考流程：

首先我们将每个表情用特殊的标识来标记，然后放入一个隐藏&lt;div&gt;中，当用户点击聊天表情的时候显示此div，由于使用的是文本域，每当用户选择表情后分析聊天框输入消息中的特殊标识直接追加&lt;img&gt;标签，此标签对应表情，这样就能随时添加表情聊天。因为现阶段实现都是使用的两个文本域&lt;textarea&gt;，可以在添加表情的时候一个使用&lt;div&gt;嵌套在文本域中。

聊天字体、大小、粗细：

参考流程：

同理跟表情的实现是相差无几的，这里可以使用div css来改变文本的颜色、大小、粗细等。

做聊天系统还有很多细节方面需要去完成，这里只是简单的介绍了一下根据原型来实现的一些主要聊天功能。比如一些具体业务，如聊天次数控制、声音等等具体事项都需要进一步去研究。

作为一款comet开源框架，个人感觉与dwr类似，同样都是基于长连接推送消息到服务器端。

由于未找到dwr、pushlet两者做的聊天系统性能方面测试，所以不知道具体谁优谁劣，但是dwr官方给出的示例中有一对一的在线聊天，个人倾向于用dwr来实现。

规范的目标是在浏览器中实现和服务器端双向通信.双向通http://www.wendangxiazai.com/b-8b22b8345a8102d276a22f4e.html信可以拓展浏览器上的应用类型,例如实时的数据推送，聊天/im 等。

目前在浏览器中通过http仅能实现单向的通信,comet可以一定程度上模拟双向通信,但效率较低,并需要服务器有较好的支持; flash中的socket和xmlsocket可以实现真正的双向通信,通过 flex ajax bridge,可以在javascript中使用这两项功能.

但是客户端方面目前的浏览器较高版本的chrome、firefox支持websocket，而ie不支持，服务端方面最新的tomcat7.0.27、jetty 7.0.1等支持，所以基于html5的

技术应该是不予考虑的。

用来做聊天室的性能应该会是非常的高，因为他旨在解决服务器高并发的瓶颈。 有使用nodejs与node-websocket-server做聊天室，但是上面已经分析websocket并不适合此系统实现聊天功能，所以就不去研究.

写的聊天系统通常也是要基于websocket的，有可能有偏颇，但是我还是觉得不应该使用此来实现。

Socket：

前面也有提到过使用flex和java的socket的通讯也能很好的达到实时的聊天效果。但是现在最主要的瓶颈应该是我们都没有人开发过flex。如果能找到相关人员来开发也是一条很好的解决方案,现今有很多网络聊天室用的就是此技术.

刚刚了解到的一种聊天实现，为sina开发。去sina申请网站地址后直接在网页中添加几行代码即可使用，访问此页面的所有人员都能进行访问。感觉挺新颖的所以拿来介绍一下

结语：

经过这些天的调查实现聊天方面资料，个人感觉使用flex socket来实现最好，有许多成熟的案例。如果不行综合考虑还是可以使用dwr来做，就是不知道大多数用户长连接对服务器的影响了。

参考资料：

基于http://www.wendangxiazai.com/b-8b22b8345a8102d276a22f4e.htmlhtpp长连接的“服务器推”技术

基于iframe的服务器推送(Server Push)

：