**一、实验题目**  
仿QQWeb即时聊天系统

**二、实验目的**

1. 学习并掌握即时通讯的技术原理和实现方法；

1. 复习和巩固计算机软件设计知识
2. 提高编程能力和开发经验

**三、总体设计**

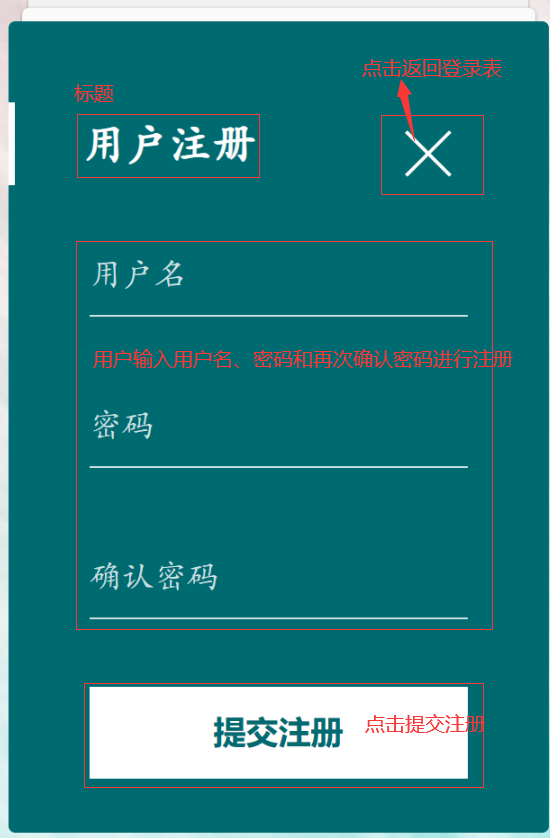
1. **技术栈与开发工具**

前端：html5+jquery+ajax+WebSocket

后端：springboot+WebSocket

开发工具：idea+navicat

1. **模块介绍**
2. 登录注册模块



1. 看板模块



1. 聊天模块



1. **基本原理**
2. WebSocket介绍

①websocket是一个持久化协议，通过一次http request建立连接后，后续不需要重新发送httprequest。

②websocket是双向通信的连接，一个tcp连接上既可以发送，也可以接收。

③具有多路复用功能，几个不同的url用同一个websocket连接，类似/websocket/{userid},给不同的user推送消息。



图1 聊天功能时序图

1. 前后端分离架构

前后端采用 JSON 格式通过ajax 进行数据交互。

图2 前后端交互原理图

**四、详细设计**

1. **需求分析**
2. 实现系统的点对点即时的文本消息聊天功能。
3. 实现系统的表情的发送、接收和显示功能。
4. 实现系统的图片的发送、接收和显示功能。
5. 实现本地消息的存储，在离线的时候也能加载和查看历史消息。
6. **主要功能具体实现**
7. **登录功能**
8. 前台实现

通过form表单获取到用户输入的用户名和密码，携带from表单数据向后台发送post请求。请求成功后进入到聊天界面，同时向后台发送请求获取用户名用于聊天界面信息展示。只有当这一步成功后，才能进行后面的建立websocket连接操作。

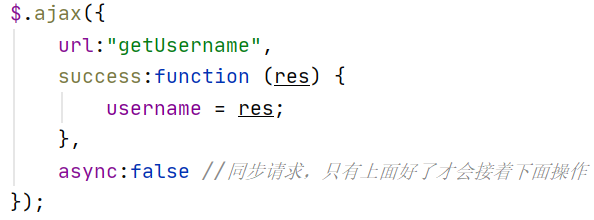


图3 登录功能前端核心代码

1. 后台实现

在进行验证登录之前先做一次拦截，检查session对象里面有没有该用户或者是否获取到了用户名，如果有直接放行进入到聊天界面，如果没有重定向到登录界面，后台根据前台传递过来的from表单数据查询数据库进行验证，避免了用户直接跳过登录界面通过输入聊天界面地址直接进入。每次验证成功后，将该用户信息作为一个session对象存放起来。并通过"getUsername"接口返回用户名。



图4 登录功能后台核心代码

1. 注册功能

①前台实现

通过form表单获取到用户输入的用户名和密码，监听用户点击“提交注册”按钮事件，该事件触发时将携带from表单数据向后台发送post请求。

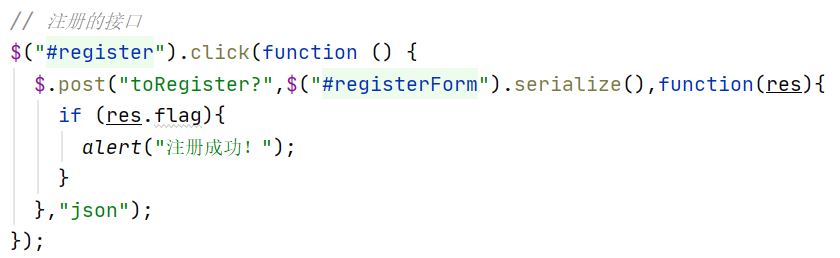


图5 注册功能前台核心代码

②后台实现

通过SQL语句操作数据库，将新增的用户添加到用户表中进行数据的持久化保存。

1. 点对点即时文本聊天功能

①实现思路

不同用户登录到聊天界面，将文本消息发送到服务器，服务器作为消息的中转站，再将消息发送到与之聊天的用户。

②前台实现

1）前端设置一个容器用来承载用户输入的聊天内容，需要注意的是，因为聊天内容包括图片，所以不能使用textarea标签，而应该给div盒子添加contenteditable属性使其成为富文本编辑器才能够输入图片。

2）涉及到大量js的dom操作，在开发中使用了jquery对dom操作进行简单的封装，使用起来更加方便。

3）当用户点击“发送”按钮时，**首先**通过$("#input\_text").html()获取到用户在文本框输入的内容，并将其赋值给一个变量data，**然后** $("#input\_text").html("")清空文本输入框，这样下次输入内容时无需手动清除上一次的文本信息。**接下来**将需要发送的聊天信息和用户名数据封装成一个json对象 let json = {"toName": toName, "message": data}。**然后**将这条聊天信息追加到聊天区右侧自己的聊天记录栏，并通过localStorage.setItem(toName, str)对这条聊天记录进行本地保存。**最后**调用已经创建好的websocket对象的send()方法，将消息发送到服务器。



图6 点对点即时文本聊天功能前台核心代码

③后台实现

**首先**后台拿到前台发送的json字符串，将其解析为message对象。**然后**从message对象找到服务器接收到的消息的客户端接收方以及获取到消息数据。**接下来**获取到当前登录用户也就是消息发送方。**最后**格式化需要发送的消息和接收人。



图7 点对点即时文本聊天功能后台核心代码

（4）表情发送、接收和显示功能

①前台实现

1. 前台采用引入一个叫做jQuery-emoji的表情插件，**首先**在插件官网将需要的代码文件下载下来。**然后**将文件夹放置在静态资源目录下以便使用时引入。**接下来**，在需要使用表情插件的页面引入css和js的压缩文件。

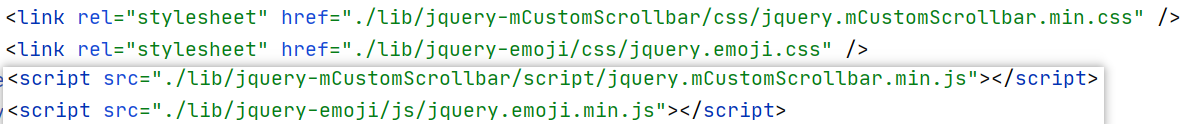


图8 表情插件引入文件

1. 自定义一个表情图标，添加点击事件，当触发时，文本输入框盒子与表情tab进行绑定，点击选中的表情自动添加到文本输入框。



图9 表情发送、接收和显示功能前台核心代码

②后台实现

表情发送、接收和显示功能的后台实现与点对点即时文本聊天功能的后台实现相同，这里就不做赘述。

（5）图片发送、接收和显示功能

①前台实现

1）上传本地图片资源需要用到html5的input标签，给标签添加type = “file”属性就可以从本地拿到图片资源，由于input标签默认的样式不太好看，我们需要使用display:none对其进行隐藏，并添加上自己的图标。

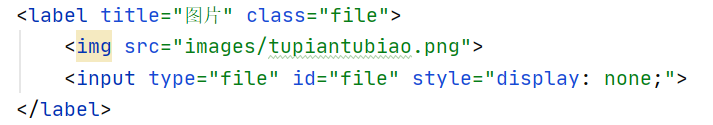


图10 上传图片标签代码

2）发送图片时，首先需要将图片转为base64的格式，再通过ajax发送给后台。其它操作与文本、表情相同，不做赘述。

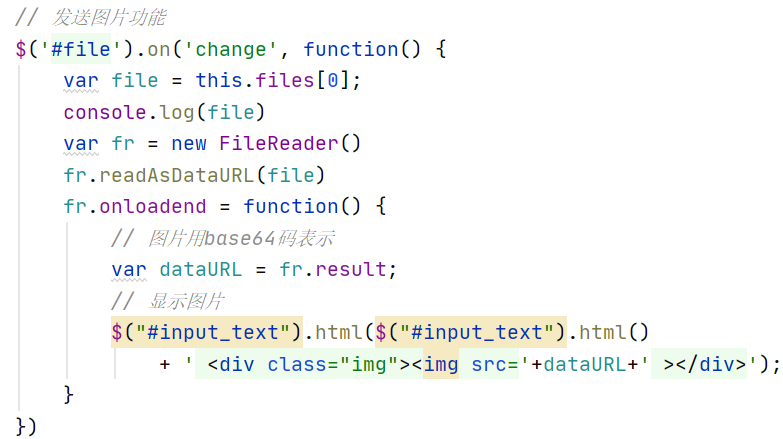


图11 发送图片前台核心代码

②后台实现

WebSocket接收前端图片自动断开连接异常，是因为Websocket中maxMessageSize默认是8K，接收的图片大于8K就导致接收失败，需要适当加大maxMessageSize，其它操作与点对点即时文本聊天功能的后台实现相同，这里就不做赘述。

1. 查看历史消息功能

①主要使用浏览器的本地缓存 localStorage，当用户打开聊天区域时，从localStorage中获取历史信息，加载到聊天区域。实现了离线时也能加载和查看历史消息。

②与sessionStorage的区别在于localStorage可以做到永久存储，而sessionStorage只在单个会话中存储，当会话结束后，存储的键值就被清除了。

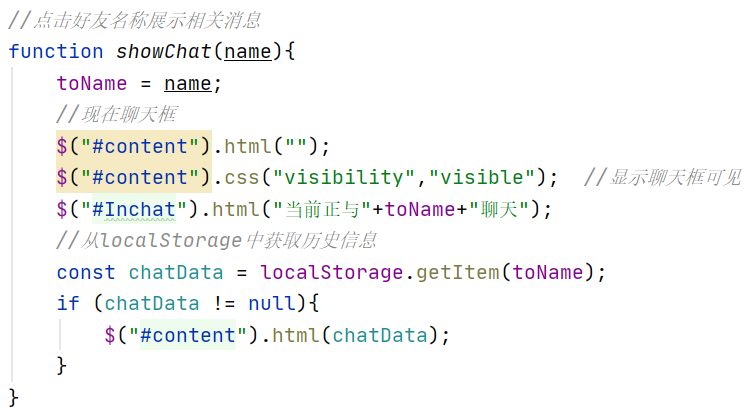


图12 查看历史消息功能核心代码

1. 其它功能具体实现

（1）在线用户列表和系统广播功能

①前台实现

1）客户端与服务器建立WebSocket连接，通过onmessage从后台获取消息，并将消息格式化为json对象。

2）判断是否为系统消息，如果是，把从后台拿到的message列表过滤掉自己的名字。然后通过userlistStr存储上线用户，broadcastListStr存储用户上线广播通知。

3）最后，通过dom操作，用户列表和系统消息渲染到页面上。



图13 在线用户列表和系统广播功能前台核心代码

②后台实现

1）后台使用线程安全的map来保存当前用户，当有用户上线或者用户下线都会进行系统广播，将最新的用户列表返回给前台。

2）后台封装发送的消息内容里添加标志位isSystemMessage，当为true时，证明是系统的消息。



图14 在线用户列表和系统广播功能后台核心代码

（2）看板功能

①功能介绍

聊天界面引入live2D看板元素，增加用户聊天趣味性。

②具体实现

**首先，**该插件需要 Font Awesome (v4 或 v5) 图标支持，先下载好图标库并放置在静态资源目录下。**然后，**在需要的页面引入确保相关样式表已在页面中加载。**接下来，**通过外部连接方式导入模型的脚本文件，即可在浏览器生效。**最后，**初始化看板模型数据，调整到自己喜欢的位置。

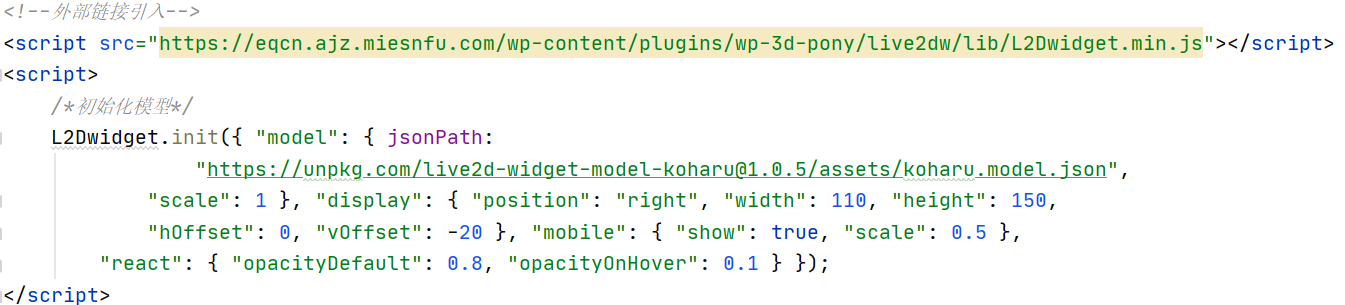


图15 看板功能核心代码

五、实验结果与分析

1.新用户打开网站时首先进行注册操作，输入用户名和密码点击提交注册后打开浏览器调试功能可以看到发送了携带用户注册数据的网络请求。

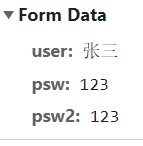


图16 注册网络请求负载

2.注册成功后，用户回退到登录窗口输入提交的注册信息，可以看到该用户成功进入到聊天界面，并且显示在线状态。



图17 用户在线状态展示

1. 当有其他新的用户重复执行以上两个步骤，可以看到用户列表更新以及系统消息重新广播。



图18 用户列表及系统消息

1. 任意点击用户列表名字，即可打开聊天对话框并与其在线交流。可以看到两个用户之间可以进行文字、图片和表情的发送接收。



图19 在线用户聊天图片

1. 当用户离线的时候，也可以加载和查看历史聊天记录。



图20 离线查看历史聊天记录

以上便是仿QQWeb即时聊天系统的实验结果展示。  
**六、小结与心得体会**

通过本次课设，我收获了很多东西。将书本上的理论知识运用到实际开发当中，才真正理解里面的原理。此外，还在网上查询了很多博客，解决开发当中遇到的问题。

1. 当在看到题目中提到要用WebSocket技术实现时，就很想尝试一下。在这之前有在开发中使用到**轮询**技术，但是请求中有大半是无用的，这样很**浪费带宽和服务器资源**。经过快速学习了解WebSocket技术的基本原理以及如何使用。

2. 前台开发使用的是**html5+jquery**比较原生的开发方式，里面涉及到大量js的**dom操作**，开发起来比较麻烦。每次有新的聊天内容发送时都会携带html**标签与数据**一起发送给服务器，这样增加了服务器的压力，也不利于代码后续的维护工作。

3. 目前前台比较主流的**vue开发框架**可以很好的解决第2点提到的问题。vue是一个js库，并且无依赖别的js库，直接引入一个js文件就可以使用，与传统JS和JQuery框架不同，Vue的渐进式框架让开发者可以由简单组件写起，渐渐搭建出一个复杂的前端平台。形成Vue渐进式框架的核心概念为：组件化，**MVVM数据双向绑定**，响应式，和生命周期。Vue一切是以数据为核心，使用**数据来驱动视图刷新**。

4. 在实现登录注册功能时发现如果用户知道聊天界面的地址，则可以跳过登录操作直接进入到聊天界面，我们不希望这样。这里采取在后台使用**拦截器技术和重定向技术**。在放行到聊天界面之前先进行拦截检查**session对象**里面有没有该用户或者用户名是否为空，如果有直接放行进入到聊天界面，如果没有重定向到登录界面。

5. 前台在进行聊天内容布局时，花费了很多时间。我们希望发送者和接收者的聊天内容分别在左右进行展示，这需要给消息盒子**添加左右浮动**。但是添加完之后发现由于加了浮动的盒子是浮起来的，漂浮在其他标准流盒子的上面，而且**不再占据位置，**这样下面的消息盒子会挤上来和上一个浮动的消息盒子并排显示。这里我们通过**额外标签法清除浮动**解决这个问题。

6. 刚开始做聊天记录这块并不清楚cookie、sessionStorage，localStorage区别，就用了sessionStorage进行存储。在测试的时候发现，当关闭浏览器之后再次登录的时候，之前的聊天记录并没有被保存下来。仔细翻阅文档发现存储在 sessionStorage 里面的**数据在页面会话结束时会被清除**。这里改为使用localStorage进行存储，就成功解决这个问题。**localStorage生命周期是永久的**，除非主动清除localStorage信息，否则这些信息将永远存在。

7. 在做图片发送功能时，并不知道前台该以怎样的方式传输到后台。查询资料发现可以将图片**转码为base64格式**。前台将转码后的base64编码添加到img标签的src属性上即可重新复原图片。解决完图片传输格式问题后，又发现每次用户发送图片时会**自动断开连接异常**，是因为Websocket中**maxMessageSize默认是8K**，接收的图片大于8K就导致接收失败，需要适当加大maxMessageSize。

8. 在开发中学习到了如何使用表情插件和看板插件，**插件的使用**很方便，只需要下载好依赖的文件并引入到页面中来，就可以快速完成一个模块的开发，大大节约了开发成本，提高开发效率。