

web聊天室总结

前言: 最近在写一个聊天室的项目，前端写了挺多的JS(function)，导致有点懵比，出了BUG，也迟迟找不到。所以昨天把写过的代码总结了一下，写成博客。

项目背景

参考博客: <http://www.cnblogs.com/alex3714/articles/5337630.html>

项目界面

最开始的界面布局:



加点bootstrap样式:

left	title
好友 群组	window
Alex Li	
Eric Wu	
Li Uber	
	emoj
	发送

实时的聊天效果:

left	正在和 ZhangChengLiang 聊天	好友	ZhangChengLiang2017/5/25
好友 群组	31495682324.140846	群组	有空吗
ZhangChengLiang	有空吗	Alex Li	21495682365.644271
	Eric Wu2017/5/25	Eric Wu	干嘛
	干嘛	Li Uber	ZhangChengLiang2017/5/25
	31495682397.986495		长夜漫漫.....
	长夜漫漫.....		21495682447.211894
	Eric Wu2017/5/25		不好意思,今晚已经约人了,你明晚吧
	不好意思,今晚已经约人了,你明晚吧		

第一步:点击左侧界面的好友,触发事件,打开聊天界面

left	正在和 Alex Li 聊天
好友 群组	
Alex Li	
Eric Wu	
Li Uber	
	emoj
	发送

1.1、给点击好友添加active属性，使其高亮。

Alex Li是一个li标签，属性有联系类型，与Alex Li的用户id。

```
<li contact-type="single" id="1" class="list-group-item active" onclick="OpenChatWindow(this)">
</li>
```

1.2、上面的single与id是怎么来的呢？见如下html代码。

```
<ul class="list-group">
  {% for friend in request.user.userprofile.friends.select_related %}
  <li contact-type="single" id="{{friend.id}}" class="list-group-item"
  onclick="OpenChatWindow(this)">
    <span class="badge hide">14</span> <!-- 新消息提醒数量 -->
    <span class="contact-name">{{friend.name}}</span>
  </li>
  {%endfor%}
</ul>
```

第二步：使界面标题框出现"正在和Alex Li聊天"字样。

并给其div 添加contact-id,与contact-type属性。

```
<div class="chat-box-title" contact-id="1" contact-type="single">
  <span>正在和 Alex Li 聊天</span>
</div>
```

第三步：通过事件委托(可看博客《[聊一聊JQ中delegate事件委托的好处](#)》) 绑定事件

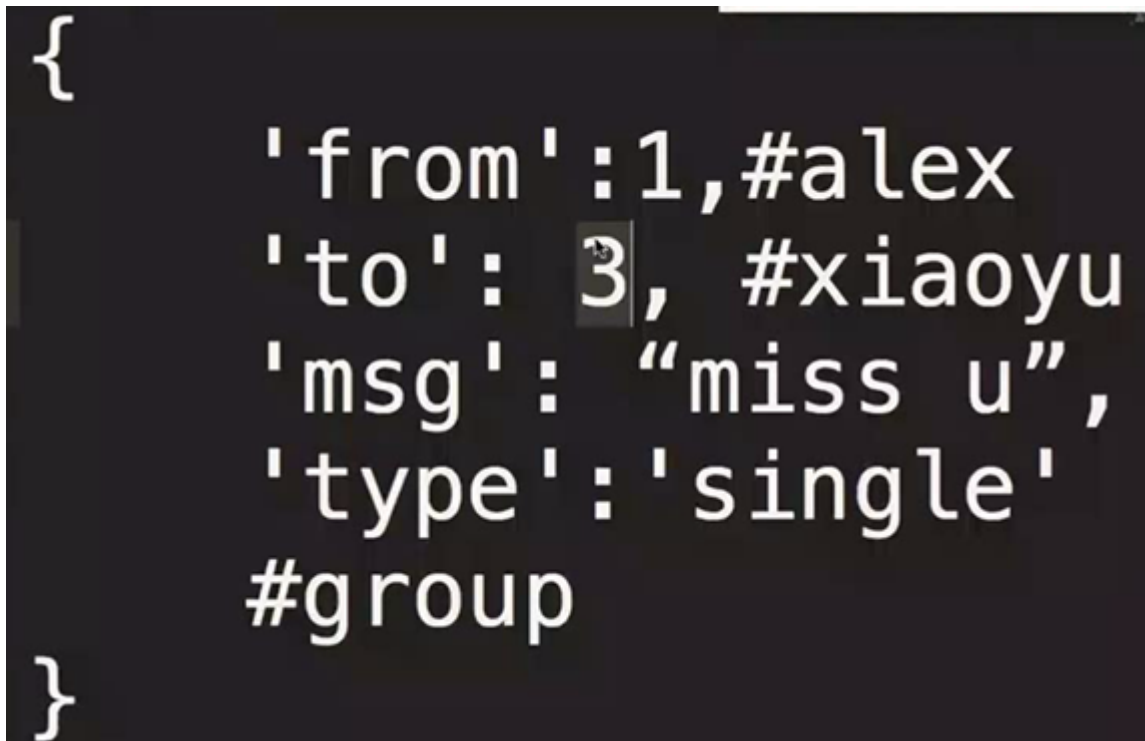
一按回车键就调用SendMsg(msg_text);发送消息

```
// 事件委托
$("body").delegate("textarea", "keydown", function (e) { // e==event
  if(e.which == 13){ // 按下键的数字(e.which);13是enter键的ASCII码
    var msg_text = $("textarea").val();
    if ($.trim(msg_text).length > 0){
      console.log(msg_text); // 发送消息给对方
      SendMsg(msg_text); // 将发送的信息打印到自己的window界面
      AddSendMsgIntoWindow(msg_text);
      $("textarea").val(""); // 将输入框清空
    }else {
      alert("请输入要发送的消息")
    }
  }
});
```

第四步：通过ajax将消息发送到后台

4.1、消息的格式：

```
var msg_item={
    "from": "{{request.user.userprofile.id}}",
    "to": contact_id, "type": contact_type,
    "msg": msg_text    //要发送的消息
};
```



```
{
    'from': 1, #alex
    'to': 3, #xiaoyu
    'msg': "miss u",
    'type': 'single'
    #group
}
```

type为发送的格式：

- 为single表示一对一发送；
- 为group表示群发，此时的"to",后面的3表示群组id

前面都没什么难度的，到这里通过ajax将消息字典(json格式)发到后台，后台怎么处理么？

简单阿，将消息发给用户阿，那要是用户没登陆呢??那只好先将数据存起来，存在哪呢?要满足先进先出，可以用队列queue，当然，生产环境最好用rabbitmq(《[python之rabbitMQ](#)》)；

4.2、后台每个用户都有一条队列queue. 后台的全局队列如下: 以用户的id作为key, 队列作为value

```
GLOBAL_MSG_QUEUES={
    "id": queue.Queue(),
}#队列：全局变量
```

4.3、将消息字典存到待接收用户的队列中去，如果用户此时队列不存在，则先给待接收消息的用户生成对应的一条队列，再将消息字典存到待接收用户的队列中去。

```
# 如果用户队列不存在,注意, 如果id(key)对应的队列不存在, 则输出None,不会报错
if not GLOBAL_MSG_QUEUES.get(queue_id):
    GLOBAL_MSG_QUEUES[queue_id]=queue.Queue()    #创建队列
GLOBAL_MSG_QUEUES[queue_id].put(msg_dic)        #将消息字典(带时间戳)放进队列
```

4.4、将消息字典(含时间戳)成功存到待接收用户的队列后, ajax请求完毕, 返回"---receive msg---", 表示后台已成功接收到要发送的消息, 表示消息字典成功存到待接收用户的队列中。

```
return HttpResponse("---receive msg---")
```

第五步：将发送的信息打印到自己的window界面，调用 AddSendMsgIntoWindow(msg_text);

```
function AddSendMsgIntoWindow(msg_text){
    var new_msg_ele="<divclass='msg-item'>" +
        "<span>" + "{request.user.userprofile.name}" + "</span>" +
        "<span>" + newDate().toLocaleDateString() + "</span>" +
        "<divclass='msg-text'>" + msg_text + "</div>" +
        "</div>";
    $(".chat-box-window").append(new_msg_ele);

    $(".chat-box-window").animate({
        scrollTop: $(".chat-box-window")[0].scrollHeight//每隔0.5s自动向下滚动
    },500); //console.log($(".chat-box-window")[0]);
}
```

将消息打印到自己的界面一开始会出现问题, 比如你发了很多数据, 界面整个div已经装不下了, 就会“溢出”div。简单阿, 给聊天界面加个"overflow"样式就行了

```
overflow: auto; /* 给div 内容多了自动加滚动条 */
```

太棒了, 现在聊天内容一多, 就自动出现滚动条, 牛!! 但问题又来了, 虽然有滚动条, 但每次你一发消息, 都得自己去拉滚动条到最底部才能看到刚刚发的消息。这.....

于是我上网找到了这篇博客: [scrollTop 和 scrollHeight的意思](#)

今天要用到实时显示最近更新内容, 也就是要让对话框随时都在最底部。查了一下, 用

```
div.scrollTop=div.scrollHeight; 就可以了。
```

又查了查这两个参数什么意思。stackoverflow上面有人是这样解答的。

If I scroll down 5px in this window, the window's scrollTop value is 5. If I scroll right 10px in a scrollable div, the div's scrollLeft value is 10. When I scroll to the top left corner of this window, both its scrollTop and scrollLeft values are 0. 还有一个人作了补充: scrollTop and scrollHeight. In summary, scrollTop is how much it's currently scrolled, and scrollHeight is the total height, including content scrolled out of view.

总的来说, scrollTop就是卷起来的部分, 也就是我们随着下拉, 看不见的部分。scrollHeight就是整个窗口可以滑动的高度。

so, 我用下面的方法就可以解决了, 每隔0.5s 自动向下滚动至底部, [animate\(\) 方法](#)

```
$(".chat-box-window").animate({
    scrollTop: $(".chat-box-window")[0].scrollHeight
},5000);
```

第六步：界面一加载完毕就开始调用GetNewMsgs();开始取消息，向后台发起ajax起求

如何取消息?? 即是说:A向B发数据, 后台收到A的数据后, 如何返回给B

1. 后台设置一个队列, 将A发送的消息存到专属于B的队列中。
2. 前端写一个定时器, 每隔3秒就通过ajax去后台查询用户的队列是否为空, 不为空的话, 则取出数据。

```
{#          用定时器浏览器会崩(卡)#}
{#          setInterval(function () {#}
{#              GetNewMsgs();#}
{#          }, 3000)#}
```

3、用定时器的话, 会出现消息不实时的情况。比如, A用户啪啪啪很快发了很多数据, 但B用户每隔3秒才去后台取数据, 中间有最多3秒的时延。这就出现消息不实时的问题。

- 4、用户查询自己的队列中是否有无数据, 无数据的话则后台挂起。

```
# 超时挂起
# 若队列为空, 则60秒后会曝queue.Empty异常(相当在这60秒内卡住挂起)
try:
    msg_list.append(q_obj.get(timeout=60))
except queue.Empty: # 若队列为空, 则60秒后会曝 queue.Empty异常(相当在这60秒内卡住挂起)
    print("\033[41;1mno msg for [%s][%s],timeout\033[0m" %
        (request.user.userprofile.id,request.user))
```

虽然队列无数据时, 后台可以挂起, 但是前端用定时器每3秒发起一次ajax起求, 问 "我的队列中有没有来新数据阿??", 每次起求相当于浏览器起一个线程, 时间一长, 浏览器会卡, 撑不住。这怎么办呢?? 难道不能用定时器?? 那还能用什么牛逼屌炸天的方法?

按我最好情况的理解是, 前端每发起一次请问, 若队列有数据, 则立马取数据, 若无数据, 则挂起60秒(时间可在后台设置), 这60秒内若队列中有数据, 同样从队列取数据, 若60秒内队列都无数据, 则出queue.Empty异常, 此时前端再发起ajax请求。

在解决上面的浏览器线程过多, 太卡前, 我们先来看看后台是如何处理队列为空时挂起 这一功能的。看下面这张图加上上面第4点的代码, 你应该懂的。

```

>>> import queue
>>> q=queue.Queue()
>>> q.put("11")
>>> q.get()
'11'
>>> q.put("22")
>>> q.put(timeout=4)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: put() missing 1 required positional argument: 'item'
>>> q.get(timeout=4)
'22'
>>> q.get(timeout=4)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "C:\Python34\lib\queue.py", line 175, in get
    raise Empty
queue.Empty
>>>

```

虽然设置了timeout=4,但队列原本有数据,就立马取出来

队列无数据,4秒后出异常

前程明亮: www.cnblogs.com/0xcl

好,回到浏览器太卡这个BUG上来,我用了递归这个方法,代码如下:

```

// 用户获取消息
function GetNewMsgs() {
    console.log("----getting new msg----");
    $.getJSON("{% url 'get_new_msgs' %}",
        function (callback) {
            // callback是列表对象, object, 列表每个元素都是一个消息字典
            console.log(callback, typeof callback); // Array [Object] object
            // 解析消息, 用户可能收到与当前正在聊天用户的消息, 也可能是其他用户发的消息
            ParseNewMsgs(callback);

            return GetNewMsgs(); // 递归
        })
}

```

不用定时器。前端发起一个ajax请求, 后端队列无数据则挂起。当后台向前端返回数据时, 有两种情况, 一种是超时, 如60秒内, 队列都无新消息, 出queue.Empty异常后返回数据; 第二种是用户接收到别人发给他的数据, 队列一有数据, 则返回加给前端。前端收到数据后, 回调函数再发起一个ajax请求。

若前端没收到数据(在后台挂起的时间), 则不会发起ajax请求, 此时相当于实现前端挂起。不会起太多的线程。

(注: python专门设置的一种机制用来防止无限递归造成Python溢出崩溃。在python的递归是有层数限制的, 999层。超过就抛出“RuntimeError: maximum recursion depth exceeded”)

第七步

后台查询用户队列中是否有消息(数据字典), 无的话就挂起, 一分钟无消息, 则前端再发起ajax请求。若队列中有消息的话, 则返回给用户, 返回形式是列表形式, 列表每个元素都是数据字典形式。

```

eg:{"from":"3","to":"2","type":"single","msg":"1111"} <class 'str'>
return HttpResponse(json.dumps(msg_list))#序列化, 转化为json格式

```

第八步：

问题: A用户与B女神聊的正嗨, 点击左侧好友切换到C女神, 与C聊天, 但聊天窗口, 依旧是与B女神聊天的内容。

接下来要完成视图切换功能, 在切换之前将原本的视图的html元素存到全局字典

```
// 全局字典, 用于存切换视图前的html元素
GLOBAL_CHAT_RECORD_DIC = {
  "single": {},
  "group": {}
};
```

全局字典的作用可将C女神发来的消息存入, 格式应在"single"的value(字典), 再设置一个用户id与用户聊天数据的字典(C用户的id为3, xxx为将经过处理的html元素)。

如: "single":{"3": "xxx"},

切换视图前将与B女神聊天窗口的html元素(数据)添加到用户的全局字典中去:

```
// 视图切换, 在切换之前将原本的视图的html元素存到全局字典
// 原本的框题框contact-id属性不为空, 即切换视图前左侧已有点击对象
if ($("#chat-box-title").attr("contact-id")){ // 从标题框取出切换前用户的id, 与联系方式contact-type
  var session_id = $("#chat-box-title").attr("contact-id");
  var session_type = $("#chat-box-title").attr("contact-type");
  // 将切换产前的视图存入全局字典
  GLOBAL_CHAT_RECORD_DIC[session_type][session_id] = $("#chat-box-window").html();
}
```

框题框显示"正在与C女神聊天"后, 从全局字典取出与C的聊天数据, 并显示在聊天窗口。

```
// 把chat-window的html元素从全局字典中存出来 // 第一次点击该联系人时, chat_record为undefined, 因为字典
// 的single下还没有生成id(key)对应的value
var chat_record = GLOBAL_CHAT_RECORD_DIC[contact_type][contact_id];
console.log(chat_record, typeof chat_record);
if (typeof chat_record == "undefined"){
  $("#chat-box-window").html("");
}else { // 如果chat_record为undefined, 则下面代码无法将对话界面清空(重要)
  $("#chat-box-window").html(chat_record);
  console.log("haha>>", chat_record)
}
```

视图切换功能基本完成。

第九步：前端通过ajax接收到后台返回的数据后, 将数据渲染成html样式后显示在聊天窗口。

这里就要分情况了。

如果用户收到的是与当前正在聊天用户的消息，直接将html元素添加到聊天窗口\$(".chat-box-window").append(new_msg_ele); 否则，如：用户A正在与B聊天，此时收到来自C发来的消息。C发来的消息要发在哪里呢？当然是前面设置的全局变量啊!!

```
// 用户收到的是与当前正在聊天用户的消息
if (current_session_id==callback[msg_item]["from"] &&
current_session_type==callback[msg_item]["type"]){ // 将消息的html元素添加到聊天窗口
    $(".chat-box-window").append(new_msg_ele);
}else { // 用户没打开消息发送者的对话框，消息暂存到内存中(全局变量中)
    if (typeof GLOBAL_CHAT_RECORD_DIC[callback[msg_item]["type"]][callback[msg_item]
["from"]]=="undefined"){
        GLOBAL_CHAT_RECORD_DIC[callback[msg_item].type]
[callback[msg_item].from]=new_msg_ele;
    }else { // 如果GLOBAL_CHAT_RECORD_DIC[current_session_type][current_session_id]不为
undefined
        GLOBAL_CHAT_RECORD_DIC[callback[msg_item].type]
[callback[msg_item].from]+=new_msg_ele;
    }
}
```

先写这么多吧。转发注明出版: <http://www.cnblogs.com/0zcl/p/6903017.html>

前几天学了git，有想一起做这个小项目的么??一个人写代码总感觉太慢了，有想一起完成的可以看看我的git项目 https://github.com/0zcl/bbs_project，可以pull给我。欢迎交流。