基于 AJAX 的在线客服系统的设计与实现①

Design and Implementation of the Online Customer Service System Based on AJAX

王 璐 朱晓民 (北京邮电大学 网络与交换技术国家重点实验室 北京 100876)

摘 要: 在线客服系统是一种网页形式的即时通讯软件平台的统称，为企业提高形象，加强企业与企业网站访客的互动，为企业进行网络营销提供了很好的平台。AJAX（Asynchronous JavaScript and Xml）作为 Web2.0 的代表技术之一，是在线客服系统得以实现，推广的关键技术之一。介绍了 AJAX 的技术和工作原理，并给出了基于 AJAX 技术的在线客服系统的设计与实现。

关键词: AJAX Web2.0 在线客服系统 服务器 客户端

1 引言

在线客服系统[1]是 Web2.0 技术的代表产物之一，是一种基于网页的即时通讯工具，它不需要安装任何软件，就可以进行实时交谈。访客浏览网站时，可以看到在线客服交流图标，或是客服发出的邀请对话框，访客可以通过主动点击图标，或被动接受邀请动作即可与网站客服人员进行实时在线交流。在线客服系统为中小型企业进行市场营销，推广产品提供了很好的平台。鉴于在线客服系统需具备很强的互动性，AJAX 技术为系统的实现提供了很好的技术保障。AJAX 是一种创建交互式网页应用的网页开发技术，它提供了与服务器异步通信的能力，使用户从传统 Web 应用中的请求—响应的循环模式中解脱出来；同时，AJAX 技术也有效节省了客户端与服务器通信的带宽，可以在不刷新页面的情况下动态更新页面数据，使 Web 应用程序更加自然，响应更加灵敏，从而提升用户的浏览体验。

2 AJAX概述

AJAX[2]是多种技术的综合，它使用 XHTML (Extensible HyperText Markup Language) 和 CSS(Cascading Style Sheet)标准化呈现；使用 DOM(Document Object Model)实现动态显示和交互；使用 XML(eXtensible Markup Language)或 JSON(JavaScript Object Notation)进行数据交换与处理；使用 XMLHttpRequest 对象进行异步数据读取；使用 JavaScript 绑定和处理所有数据。

传统的 Web 应用采用同步交互过程，用户首先向 HTTP 服务器发送一个 HTTP 请求，服务器执行某些任务再向用户返回一个 HTTP 响应。这是一种不连贯的用户体验，服务器在处理请求 (特别是负载比较大)的时候，用户大多数时间处于等待状态，屏幕内容也是一片空白，极大影响用户的使用体验。

与传统的 Web 应用不同，AJAX 采用异步交互过程。AJAX 在用户与服务器之间引入一个 AJAX 引擎，从而消除了网络交互过程中的处理—等待的缺点。在会话的开始，浏览器加载一个 AJAX 引擎，这个引擎负责绘制用户界面以及与服务器端进行通讯。AJAX 的工作原理如图 1 所示，相当于在用户和服务器之间加了一个中间层(AJAX 引擎)，AJAX 引擎实际上是一个比较复杂的 JavaScript 应用程序，用来处理用户请求，读写服务器和更改 DOM 内容，它使用户操作与服务器响应异步化。当然，并不是所有的用户请求都提交给服务器，像一些数据验证和数据处理等是交给 AJAX 引擎自己来做，只有确定需要从服务器读取新数据时再由 AJAX 引擎代为向服务器提交请求[3]。

① 基金项目:国家杰出青年科学基金(60525110);国家重点基础研究发展计划(973)(2007CB307100,2007CB307103);新世纪优秀人才支持计划(NCET-04-0111);电子信息产业发展基金
收稿时间:2009-03-06

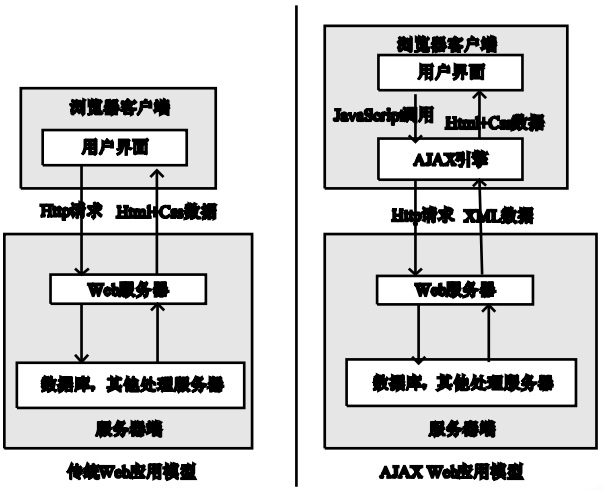

图 1　传统 Web 应用模型与 AJAX Web 应用模型比较

AJAX 处理过程中的第一步是创建一个 XMLHttp Request 实例。使用 HTTP 方法(GET 或 POST)来处理请求，并将目标 URL 设置到 XMLHttpRequest 对象上。当发送 HTTP 请求时，只需为 XMLHttp Request 注册一个回调函数，当服务器响应返回时，回调函数将会被调用，完成相关动作，而在 HTTP 请求发出到服务器响应返回这段时间内，用户可以继续执行页面上的操作，不会出现用户体验中断的状态。

3　系统设计与实现

3.1 概述

在线客服系统可作为一个独立的服务器为任何第三方网站提供在线客服交流功能。第三方网站只需要在其相关页面上注入一段在线客服系统的 JavaScript 脚本即可。当网站访客浏览第三方页面时，浏览器会自动加载运行这段被注入的脚本，从而使用在线客服系统。系统部署图如图 2 所示。

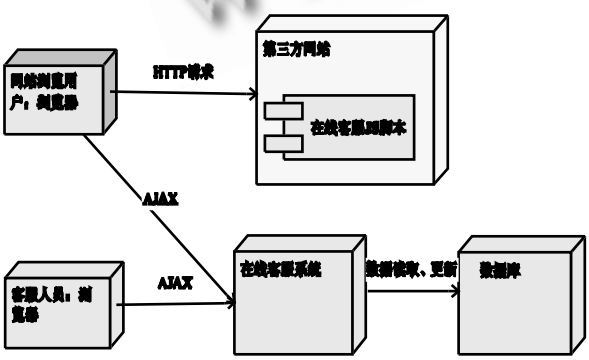

图 2　在线客服系统的部署图

在线客服系统采用 B/S 架构，客服与网站访客的状态及交谈信息都经服务器进行中转，从而使服务器可以对客户端对话过程进行控制。

在线客服系统是即时消息的一种特定的展现形式，目前主要的通用即时消息协议有：XMPP(Extensible Messaging and Presence Protocol)协议簇，SIMPLE (SIP for Instant Messaging and Presence Leverage Extension)协议簇及 IMPS(Instant Messaging and Presence Services)协议簇[4]。鉴于本系统不涉及到与其他即时消息系统互通问题，故采用自定义的消息格式，较以上三种通用协议就有简单性，自定义性等优点。

3.2 服务器

3.2.1 功能结构设计

提供核心控制和管理功能，接收用户的请求，并完成用户登入登出，文字聊天，文件传输，访客邀请，客服转接，信息拉取等子功能模块任务。各模块功能见下述。服务器功能结构图如图 3 。

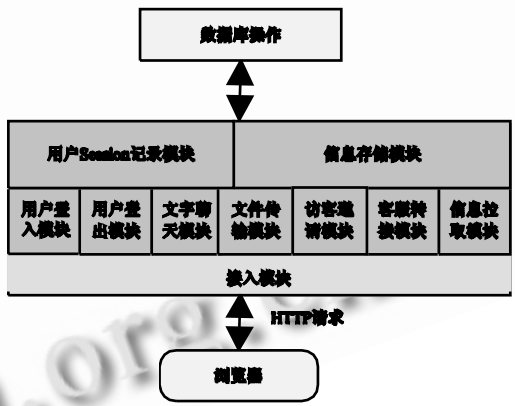

图 3　服务器功能结构图

(1) 接入模块：是系统对外的接口处理模块，接收用户发送的请求，并根据请求类型，调用相应的具体子功能模块，如下所述。

①登入：在线客服系统涉及三类人群：网站浏览访客(浏览网站，但未与企业客服进行即时通信的用户群)，网站即时通信客户(与企业客服进行即时通信的用户群)和企业客服。系统根据不同的用户群，构造相应的初始信息数据，创建用户 session，并将该用户的登入信息储存到信息存储模块中。

②登出：根据接收到的登出数据信息，清空用户 session 容器中该用户所对应的 session，并将该用户的登出信息存储到信息模块中。

③文字聊天：接收客户端输入的文本内容，并构造成文字信息数据，储存到信息存储模块中。

④文件传输：接收一个客户端的文件传输请求，并完成文件上传到服务器的功能，同时构造文件传输信息数据，并保存到信息存储模块中。

⑤访客邀请：客服可以获取当前网站浏览访客的信息，并及时向访客发出即时通信的邀请。此模块接收客服对访客的邀请请求，并构造邀请信息数据，并保存到信息存储模块中。

⑥客服转接：客服可以将与其进行交流的客户转接至其他客服。系统收到此请求，构造转接信息数据，并保存到信息存储模块中。

⑦信息拉取：是系统中使用最频繁的模块，是客户端获取最新数据的入口。鉴于本系统采用 HTTP 短连接，客户端需要通过定时向服务器进行拉取，以获得与其相关的数据信息，以便动态完成浏览器客户端内容的更新。

(2) 信息存储模块：是保存终端用户相关信息的容器，它接收子功能模块所构造的信息数据，当信息拉取模块被调用时，从容器中提取出自上次拉取之后，容器所收到的与此用户相关的信息数据，并以 JSON 数据格式传给浏览器客户端进行处理。

(3) 用户 Session 记录模块：是系统中记录用户 session 的容器，它为每个用户创建独立的 session，同时记录每个在线用户的标识信息。

3.2.2 核心类实现

MessageService 和 MessageRepository 是本系统与信息存储相关的核心类，类图如图 4 所示。

(1) MessageService：采用单实例模式，主要负责信息的接收和发送功能，是信息存储模块的核心对外接口类。

其中 sendContentMsg 方法用于接收用户发送的文字聊天内容，然后通过调用 createIMMessage 方法创建 IMMessage 对象，并使用 Message Repository 的 cacheMsg 方法将其保存到内存和数据库中。

sendFileMsg 方法用于接收用户传送文件的信息，然后通过调用 createIMMessage 方法创建 IMMessage 对象，并使用 MessageRepository 的 cacheMsg 方法将其保存到内存中。

EventMsg 方法用于接收用户状态改变信息，企业客服邀请网站浏览用户进行即时通信信息，及企业客服

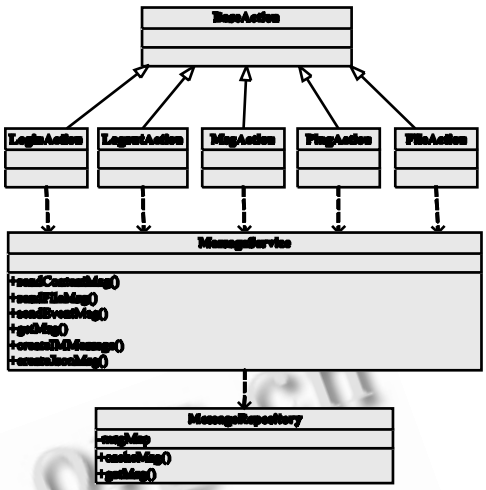

图 4 信息存储模块类图

对 IM 用户进行客服转接操作的信息，通过调用 createIMMessage 方法创建 IMMessage 对象，并使用 MessageRepository 的 cacheMsg 方法保存到内存中。

createIMMessage 方法创建 IMMessage 对象。

getMsg 方法用于提取该用户自上次调用此方法后至这次期间内存所保存的与其相关的数据信息，然后调用 createJSONMsg 方法将信息转为 JSON 数据格式，传给前端浏览器。

(2) MessageRepository：采用单实例模式，是真正的信息存储库。内部采用 ConcurrentHashMap 数据结构，将每个用户与此用户的相关信息作绑定。ConcurrentHashMap 的 value 值采用 LinkList 数据结构，将 MessageService 中通过调用 sendContentMsg，sendFileMsg，EventMsg 方法中 createIMMessage 所创建的 IMMessage 对象连接起来。

(3) UserRepository：采用单实例模式存储用户信息，是用户 session 记录模块。图 5 为 UserRepository 的类图。由于本系统采用自定义 session，系统需要自己解决 session 的创建，保存及消亡。该类负责保存当前在线的各类用户的登陆数据及相应的 session 标识，控制并消除同一时间段同一账号多地登陆的问题。用户第一次登陆后，系统会返回一个 userStatus 值，此后用户所发的所有消息都需带上此值，以保证路由到正确 session 中。对于同一帐户在两地登陆情景，系统会针对两个登陆产生两个不同的 userStatus 值，并且按照系统产生 userStatus 值的先后，后者自动替换掉前者，以解决同一帐号多地登

录问题，从而进一步保证信息的正确路由。

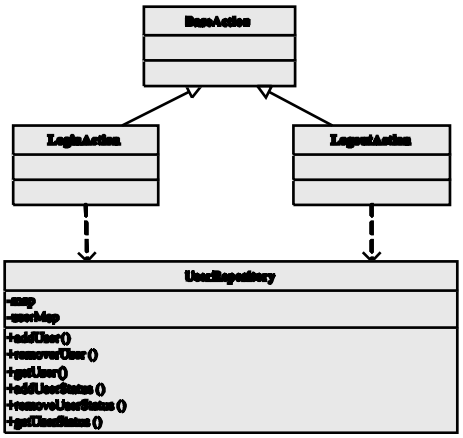

图 5　UserRepository 类图

3.3 客户端

3.3.1 设计

接收客户的请求，完成与服务器端的通信，并根据服务器返回的内容，动态更改界面内容，给用户提供桌面工具的使用体验。

在线客服系统包含三种类型的客户群体：网站浏览用户，网站 IM 用户，网站客服人员。三类客户群体共同的核心交互流程如图 6 所示。

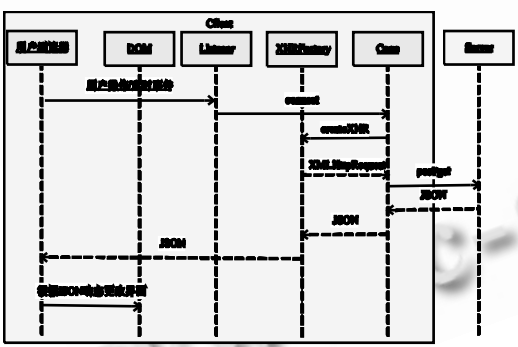

图 6　客户端核心交互图

其中 DOM：用于前台根据服务器返回的信息，动态更改界面呈现内容。

Listener：是事件监听器，用于监听用户操作(例如用户状态更改事件，用户发送即时信息，发送文件等操作)及定时 ping 操作。ping 操作是浏览器客户端的核心方法，是客户端从服务器获取最新数据信息的接口。在客户端页面被浏览器完全加载一刻起就产生一个 Timer，每隔一定时间，浏览器自动触发 ping 操作。触发 ping 操作的频率直接影响着客户端获取信息的即时性。

XHRFactory：是产生 XMLHttpRequest 的工厂类，预先产生一定数量的 XMLHttpRequest 对象，以便程序运行中快速构建与后台交互的 XMLHttpRequest 对象。

Conn：是对 XMLHttpRequest 的二次封装，简化了前台使用 XMLHttpRequest 对象与后台交互的过程。

3.3.2 核心方法实现

JavaScript 是本系统客户端的核心控制语言，它不单单只控制页面呈现效果，同时还控制着客户端的业务逻辑及客户端与服务器端的交互流程。采用设计模式思想对客户端进行设计与开发，可以使前台的逻辑更加清晰化，功能模块化，便于日后重用和维护。

图 7 为浏览器客户端所涉及的主要类图，在此，笔者主要以客户端与服务器端核心交互类的实现进行说明。

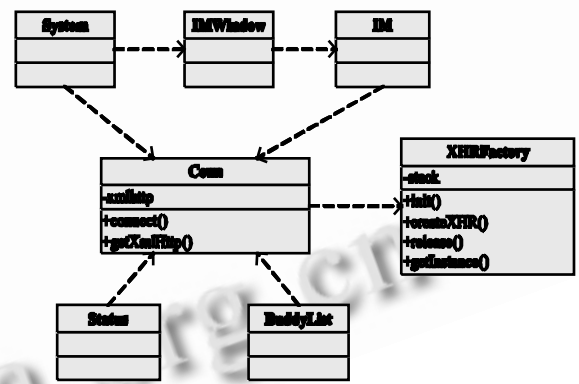

图 7　浏览器客户端主要类图

XHRFactory：是产生 XMLHttpRequest 对象的工厂类，维护 XMLHttpRequest 对象的产生，使用及销毁，故采用单实例模式。

其中的 init 方法：用于创建一定数量的 XMLHttpRequest 对象，并保存于 stack 中。

createXHR 方法：调用浏览器提供的创建 XMLHttpRequest 对象的方法。如：

Release 方法：用于销毁释放 XMLHttpRequest 占用的资源，以便节省系统资源开销。

getInstance 方法：用于从 stack 中提取空闲的 XMLHttpRequest 或自行调用 createXHR 产生一个 XMLHttpRequest 对象。

```
if (window.XMLHttpRequest) { //非IE浏 览 器
    return new XMLHttpRequest();
} else if (window.ActiveXObject) { //IE浏 览 器
    var versions = ['Microsoft.XMLHTTP', 'MSXML.XMLHTTP',
                    'Microsoft.XMLHTTP', 'Msxml2.XMLHTTP.7.0',
                    'Msxml2.XMLHTTP.6.0', 'Msxml2.XMLHTTP.5.0',
                    'Msxml2.XMLHTTP.4.0', 'MSXML2.XMLHTTP.3.0',
                    'MSXML2.XMLHTTP'];
    for(var i=0; i<versions.length; i++) {
        try {
            request = new ActiveXObject(versions[i]);
            if(request) {
                return request;
            }
        } catch(e) {}
    }
}
```

Conn：对应一次前后台的交互，采用多实例模式。

其中的 getXmlHttp 方法：通过调用 XHRFactory 的 getInstance 方法获取 XMLHttpRequest 实例。

Connect 方法：封装一次与后台 post/get 方法的调用及自定义回调函数的注册。

```
if ("GET方法")
{
    xmlhttp.open("GET", 服务器地址+"?"+参数值, true);
    sVars = "";
}else if ("POST方法"){
    xmlhttp.open("POST",服务器地址, true);
    xmlhttp.setRequestHeader("Method",
                "POST "+服务器地址+" HTTP/1.1");
    xmlhttp.setRequestHeader("Content-Type",
                "application/x-www-form-urlencoded");
    sVars = 需向server传递的参数;
}
xmlhttp.onreadystatechange = function(){
    if (xmlhttp.readyState == 4 && !bComplete){
        bComplete = true;
        if(fnDone != null) 自定义回调函数（xmlhttp）;
        XHRFactory.release(xmlhttp);
    }
};
xmlhttp.send(sVars);
```

3.3.3 即时消息发送流程的具体实现

下面，笔者将以本系统的核心功能之一——发送消息为例，结合伪代码对客户端发送即时消息的具体实现过程进行阐述。

(1) 需要为用户的消息发送动作进行事件监听。

```
1）监听发送按钮的onclick事件。
    发送按钮.onclick =function(){
            IMWindow.send();
            return false;
    };
2）监听输入文本框的回车键。
    输入文本框.onkeyPress =function(){
            IMWindow.keyHandler();
            return false;
    };
    IMWindow.prototype.keyHandler = function (ev) {
            //判断回车键
            if(!ev.shiftKey && ev.keyCode==13 )
                this.send();
    } ;
```

(2) 在捕获到上述所监听的事件后，提取有关信息，调用聊天对话框控制器 IM 的发送 即时消息接口。

```
IMWindow.prototype.send = function () {
    var msg = 输入文本框的内容;
    IM.sendMessage (msg,其他参数信息);
}
```

(3) 创建一个客户端与服务器端通信实例，注册回调函数，并发送即时消息；根据回调函数的返回值，通过 DOM 操作向用户报告消息发送结果。

```
var IM = function (){
 sendMessage : function(参数){
        var xhConn = new Conn ();
        xhConn.connect(服务器入口地址, "POST",
            传递的具体参数,
            function(xh) {
                检测xh.responseText值，
                使用DOM呈现发送结果
            }) ;
    }
};
```

4 结语

本文采用 AJAX 技术构建了在线客服系统，提供 Web 版即时通信功能。浏览器客户端采用软件设计模式的思想进行设计与开发，并在浏览器客户端中增加了业务逻辑的处理，分担了服务器的功能。本系统中用户通过定时拉取 pull 方法从服务器上获取数据，pull 触发的频率越快，越能近实时保证客户端获取数据的实时性，但这同时也会加大对服务器的压力。对于大用户量，如何能在保证数据实时性的基础上，缓解 pull 操作对服务器的压力，增大服务器的负载能力，需要日后进一步的研究。

参考文献

1 http://www.linkstalk.com/linkstalk_im_3.html

2 Crane D, Pascarello E, James D. Ajax 实战.北京:人民邮电出版社, 2007.25－27.

3 齐文新,谢军,熊涛.基于 Ajax 技术即时通讯系统的设计与实现.计算机与数字工程, 2007,35(7):148－150.

4 张雨廷,廖建新,戴忠,朱晓民,武威.即时消息业务中 SIMPLE 和 IMPS 的 Petri 网互通模型.电子与信息学报, 2008,30(10):2481－2485.