课程编号：B080109003

网络应用程序设计
实践报告

姓　　名	康 愈 圆	学　　号	20154862
班　　级	软 英 1502	指 导 教 师	刘 益 先
实践课程名称	网 络 应 用 程 序 设 计 实 践		
开 设 学 期	2016-2017 第 二 学 期		
开 设 时 间	第 1 周 — — 第 3 周		
报 告 日 期	2017.03.17		
评 定 成 绩		评定人签字	
		评 定 日 期	

东北大学软件学院

1. 实践目的

1.1 编写简单的 FTP 客户端程序和 FTP 服务器

（1）理解 FTP 通信原理；

（2）学习使用 Socket 和多线程编写简单的 FTP 服务器；

（3）学习使用 Sokcet 编写简单的 FTP 客户端；

1.2 编写简单的 HTTP 1.1 客户端和服务器程序

（1）理解 HTTP 通信原理；

（2）学习使用 Socket 编写 HTTP 服务器；

（3）学习使用 Socket 编写 HTTP 客户端

1.3 编写 WebSocket 聊天室功能

（1）理解 WebSocket 的通信原理；

（2）使用 tomcat 服务器搭建多人聊天室；

2. 预习内容

1.1 编写简单的 FTP 客户端程序和 FTP 服务器

（1）FTP 通信工作原理

FTP 服务器与客户端通过两个端口使彼此相连通，分别为数据端口(data port)和命令端口(command port)。命令端口用来传输指令，数据端口用来传输数据。如此传输不会造成信息混乱。一般情况下，服务器上的数据端口为 21 号端口在通信还没开始前，服务器须打开命令端口（#21 端口）等待客户端的连接。当客户端向服务器发送请求时，服务器会在剩余的端口给出一个临时端口用于与和客户端的实际指令连接。当客户端发送一个请求时，服务器都会给出响应，或传回指令，或开始传输数据。

数据的传输有两种模式：主动模式和被动模式。在主动模式的情况下，客户端先打开一个大于 1024 的端口向服务器的命令端口（#21 端口）发送请求，同时打开另一个端口（N+1 端口）。请求的内容时（PORT N+1）。由此服务器将从数据端口（#20 端口）向客户端的 N+1 端口发送数据。这里主动的含义是服务器主动向客户端发送数据。但往往由于客户端防火墙拦截等问题，数据正常传输收到影响。被动模式可以解决这个问题。在被动模式下，客户端先向服务器命令端口发送连接请求(PASV)，服务器开启一个任意数据端口，并向客户端发送响应“227 entering passive mode (h1,h2,h3,h4,p1,p2)”，其中 h1，h2，h3，h4 为服务器的 ip 地址，p1*256+p2 为服务器开启的数据端口号，客户端将与服务器的这个端口接收数据。

（2）客户端向服务器发送请求的重要指令及其含义

USER	传输账户
PASS	传输密码
SIZE	数据大小请求

CWD	当前目录请求
PASV	被动模式连接
PORT	主动模式连接
RETR	下载文件
STOR	上传文件
QUIT	退出

（3）服务器向客户端发送的响应码

220	Service ready for new user.
221	Service closing control connection.
225	Data connection open; no transfer in progress.
226	Closing data connection. Requested file action successful (for example, file transfer or file abort).
227	Entering Passive Mode (h1,h2,h3,h4,p1,p2).
230	User logged in, proceed. Logged out if appropriate.
231	User logged out; service terminated.
331	User name okay, need password.
332	Need account for login.
421	Service not available, closing control connection. This may be a reply to any command if the service knows it must shut down.
425	Can't open data connection.
426	Connection closed; transfer aborted.
430	Invalid username or password
501	Syntax error in parameters or arguments.
502	Command not implemented.
553	Requested action not taken. File name not allowed.

1.2 编写简单的 HTTP 1.1 客户端和服务器程序

（1）HTTP 工作原理

HTTP 是 Web 的应用层协议。是 Web 的核心。HTTP 由两个程序实现：一个客户端程序，一个服务器程序。HTTP 定义了 Web 客户向 Web 服务器请求 Web 页面的方式，以及服务器向客户传送 Web 页面的方式。

HTTP 请求报文的第一行叫做请求行，其后继的行叫做首部行。

HTTP 客户端使用 get 方法向服务器发送请求。如果成功，服务器将返回响应的值。响应报文的第一行将为“HTTP//1.1 200 OK”。

1.3 编写 WebSocket 聊天室功能

现很多网站为了实现即时通讯，所用的技术都是轮询(polling)。轮询是在特定的的时间间隔（如每 1 秒），由浏览器对服务器发出 HTTP request，然后由服务器返回最新的数据给客户端的浏览器。这种传统的 HTTP request 的模式带来很明显的缺点 – 浏览器需要不断的向服务器发出请求，然而 HTTP request 的 header 是非常

长的，里面包含的有用数据可能只是一个很小的值，这样会占用很多的带宽。

在 WebSocket API，浏览器和服务器只需要做一个握手的动作，然后，浏览器和服务器之间就形成了一条快速通道。两者之间就直接可以数据互相传送。在此 WebSocket 协议中，为我们实现即时服务带来了两大好处：

1. Header

互相沟通的 Header 是很小的-大概只有 2 Bytes

2. Server Push

服务器的推送，服务器不再被动的接收到浏览器的 request 之后才返回数据，而是在有新数据时就主动推送给浏览器。

3. 实践内容和实践过程

1. 编写简单的 FTP 客户端程序；编写简单的 FTP 服务器程序

（1） 实践内容

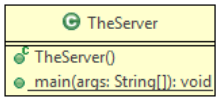
使用 Socket 和多线程技术编写简单的 FTP 服务器和客户端程序。FTP 服务器需要满足用户验证、PASV 连接服务、给出当前目录、上传文件到用户指定工作目录中、上传文件夹到用户指定工作目录中支持断点续传、给定用户工作目录下载文件，给定用户工作目录下载文件夹。支持断点续传。

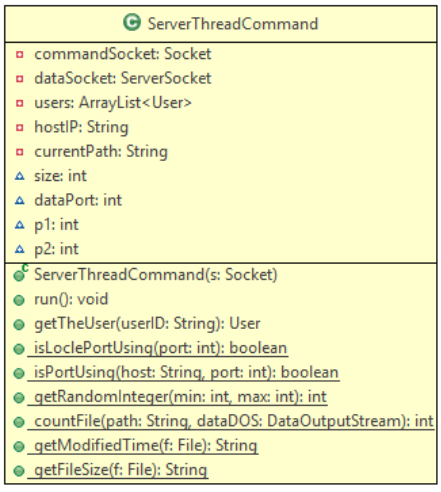
（2） 实现过程

i)类图设计

- 服务器

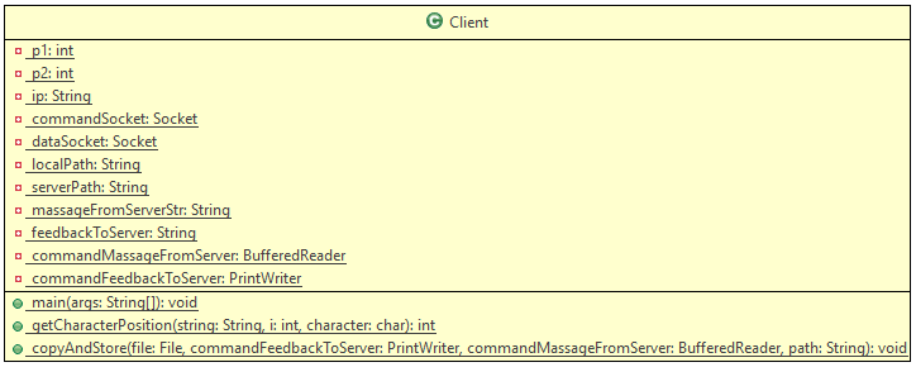
在服务器工程下，主要建立 TheServer 类。该类实现多线程功能，实际与客户端通信往来由线程 ServerThreadCommand 实现。





- 客户端

  客户端由一个 Client 对象构成。客户端的主要方法和行为都在 main 方法里实现。



ii)时序图设计

（3）关键代码说明

服务器端：

- 服务器端开启多线程

  首先构造一个 ServerSocket 对象。多线程在死循环里实现。ServerSocket 的 accept()方法会返回一个 Socket 类型对象，该对象相当于服务器命令端口的临时端口。然后将这个 Socket 对象传入新的线程。当有客户端连接时，服务器的控制台将打出一条语句 Linked successfully!

```
try{
     ServerSocket commandSocket = new ServerSocket(21);
          while(true){

               Socket tempCommandSocket = commandSocket.accept();

               System.out.println("Linked successfully!");
               ServerThreadCommand commandServerThread =
                    new ServerThreadCommand(tempCommandSocket);
               commandServerThread.start();
          }

} catch(IOException e){
          e.printStackTrace();
}
```

- 获取文件大小

该方法用以得到文件的大小。参数只有一个，为 File 类的对象。然后调用 File 类的 lenth 方法得到文件的比特数，再对此值进行转换，用 if…else…语句判断并，赋予合适的单位。

```
public static String getFileSize(File f){
          long s = f.length();
          double size = Double.valueOf(s);

          if(size>=0&&size<1024){
               return String.format("%10.4f B", size);
          }else if(size>=1024&&(size/1024.0)<1024){
               return String.format("%10.4f KB", size/1024.0);
          }else if((size/1024.0)>=1024&&(size/1024.0/1024.0)<1024){
               return String.format("%10.4f MB", size/1024.0/1024.0);
          }else if
              ......
              ......
          }
     }
```

- 文件复制功能的实现

  此方法为本项目的一个亮点。该方法需要两个参数，文件复制的源地址和目标地址。首先该方法用 File 类的 isDirectory 方法判断文件是否为文件夹，如何过文件。如果不是文件夹就进行数据传输。用 BufferedInpuyStream 读取源数据，用 BufferedOutputStream 封装输出流。如果是文件夹则先在目标路径上用相同的文件名创建文件夹，让后用 For-each 循环遍历源目录下的所有文件，修改原路径地址和目标路径地址。这里采用递归的思想传输文件夹。

```
public static void copyFile(String src, String des) throws IOException{
     File f = new File(src);
     if(f.isDirectory()){
          String folderName = f.getName();
          File folder = new File(des);
          folder.mkdir();

          File[] files = f.listFiles();
          for( File ft : files ){
               String newSrc = src+"\\"+ft.getName();
               String newDes = des+"\\"+ft.getName();
               copyFile(newSrc,newDes);
          }

     }else{
          BufferedInputStream bis = new BufferedInputStream(
                    new FileInputStream(f));
          BufferedOutputStream bos = new BufferedOutputStream(
                    new FileOutputStream(des));

          byte[] bys = new byte[1024];
          int len = 0;
          while((len = bis.read(bys))!=-1){
               bos.write(bys,0,len);
               bos.flush();
          }
          bis.close();
          bos.close();
     }
}
```

- 判断一个端口是否被占用

  在直行 PASV 指令是，服务器给出一个可用的端口用于数据传输。本项目将用以下方法判断该端口是否可用。isPortUsing()方法接收两个参数，一个

为应以表示 ip 地址的字符串，另一个为端口号。用 Socket 的构造方法构造一个 Socket 对象，如果可以通过则返回布尔值 true，表示可以使用；如果不能则返回布尔值 false。

```
public static boolean isPortUsing(String host,int port)
    throws UnknownHostException{
            boolean flag = false;
            InetAddress theAddress = InetAddress.getByName(host);
            try {
                Socket socket = new Socket(theAddress,port);
                flag = true;
            } catch (IOException e) {

            }
            return flag;
        }
```

ii) 客户端：

- 读取服务器给出的响应码

  用 String 类的 subString 方法读取响应语句的前三位。用 switch 语句根据这三位数字产生分支，给出不通的反馈。如 221 表示程序结束，则关闭所有的 Socket 对象。

```
switch(Integer.valueOf(massageFromServerStr.toString().substring(0,3))){

                        case 221:
                            commandMassageFromServer.close();
                            commandFeedbackToServer.close();
                            dataSocket.close();
                            System.exit(0);
                        break;

                        case 212:
                            System.out.println("Directory status");
                            break;
                        ......
                        ......
```

- 断点续传的实现

  在写入数据的时候首先在本地寻找该文件，如果有相同的文件名且文件大小不为 0 则说明该文件下载过，可能完整，也可能因为网络原因中断过。这里，调用 BufferedInputStream 方法的 skip()方法跳过已存的数据往后加新的内容。最后传输完毕，关闭所有的流对象。

```
else{
    int bits = Integer.parseInt(b);

    commandFeedbackToServer.println("STOR "+file.getName());
    commandFeedbackToServer.flush();
    dataSocket = new Socket(ip,p1*256+p2);

    BufferedInputStream dataFromFile = new BufferedInputStream(new
                        FileInputStream(file));
    BufferedOutputStream dataToServer = new
                        BufferedOutputStream(dataSocket.getOutputStream());

    dataFromFile.skip(bits);

    byte[] bys = new byte[1024];
    int len = 0;
    while((len = dataFromFile.read(bys))!=-1){
        dataToServer.write(bys, 0, len);
        dataToServer.flush();
    }

    dataFromFile.close();
    dataToServer.close();
    dataSocket.close();
}
```

（4）遇到的问题及解决方案

i) 问题描述：在实际编写服务器与客户端信息通讯过程中，不清楚服务器和客户端会发送怎样的信息。

解决方案：利用 FileZilla 软件对东北大学的 ftp 服务器做测试，观察交互信息的代码。在编写程序过程中模拟该过程。

2. 编写简单的 HTTP 1.1 客户端程序；编写简单的 HTTP 1.1 服务器程序；

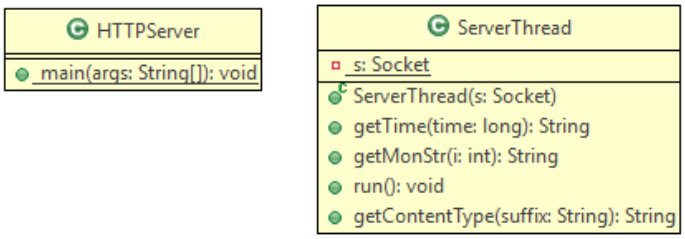
（1）实践内容

使用 Socket 编写 HTTP1.1 服务器和客户端。

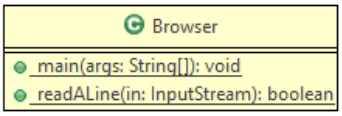
（2）实现过程

i) 类图设计

服务器：一共构造两个类，分别为 HTTPServer，ServerThread。其中 HTTPServer 用以开启新线程，使在同一时间可以有多个客户端进行访问。ServerSocket 中 getTime() 函数用于获取当前时间，返回一个字符串显示当前时间并具有一定格式。getMonStr() 将整型类型的月份转化成文字类型，并返回 String。getContentType()函数用以确定文件的类型，String 类型的参数为文件的后缀名。该函数将返回一个标准的字符串用以表示文件类型。如输入的参数为“txt”，将返回“text/plain”。



客户端：由一个类构成，Brower。其中含有两个方法。客户端的人机交互在 main 方法中实现。readALine()方法将 InputStream 作为参数，用于读入文件。一次读入一行，读到“\r\n”结束。读取的方式为字节读取。



（3）关键代码展示

i) 服务器：

- 多线程的实现

  首先构造一个 ServerSocket 对象。多线程在死循环里实现。ServerSocket 的 accept()方法会返回一个 Socket 类型对象，该对象相当于服务器命令端口的临时端口。然后将这个 Socket 对象传入新的线程。

```
ServerSocket ss = null;
        try{
            ss = new ServerSocket(80);
        }catch(IOException e){
            e.printStackTrace();
        };
        while(true){
            try{
                Socket s = ss.accept();
                System.out.println(">>"+
                                s.getRemoteSocketAddress()+"is linked");
                ServerThread t = new ServerThread(s);
                t.start();
            }catch(IOException e){
                e.printStackTrace();
            }
        }
```

- 向客户端发送文件数据

  本服务器读写数据使用 DataOutputStream 对象和 DataInputStream 对象。HTTP 服务器对客户端传输的文件不仅仅有文字信息，更有图片、视频、音频等非文本类型文件。如果以字符形式传输（如 PrintWriter 和 BufferedReader）会产生严重错误，导致文件损坏。为了加快传输速度，本次实践使用数组形式输出数据，每一次输出 1024 位，及 1KB。

```
DataOutputStream dout = new DataOutputStream(s.getOutputStream());
     ......
     ......
DataInputStream fin = new DataInputStream(new FileInputStream(f));
                    byte[] bysData = new byte[1024];
                    int len = 0;
                    while((len = fin.read(bysData))!=-1){
                         dout.write(bysData,0,len);
                         dout.flush();
                    }
                    dout.close();
```

客户端：

- 重写的读取字节流函数

  首先，客户端需要读取服务器发送的 HTTP 响应。响应的格式为字符格式。并且每一条指令都以“\r\n”结束。所以我们需要一种方法读取一行数据，当读到“\r\n”时停止读取。BufferedReader 类的 readline()方法功能于此类似。但 BufferedReader 继承于超类 Reader，字符类型的输入流。而在后续我们需要读取数据，数据类型不仅仅是字符类型（如视频、音频等文件）。而读取字节的方法，如 DataInputStream 无法识别一行数据的结束，即无法识别“\r\n”，因此我们需要重写一种方法来同时实现这两个功能。这里我们定义一种方法 readALine()，将 InputStream 类的输入流作为参数。该方法是本次项目的一个亮点。下面对此函数做出解释：

  首先创建 ArrayList<Character>用以保存某一行的字节数据。当读到“\n”时停止存储。然后将 ArrayList 中的数据重组成字符串形式输出。如果该字符串是空字符串，即“\r\n”，返回布尔值 false，反之返回布尔值 true。返回布尔值的目的是为了标记服务器 HTML 响应头部的结束。用 while(readALine(InputStream))就能完整地读出响应头部的所有信息。

```
public static boolean readALine(InputStream in) throws IOException{
          boolean tag  =true;
          ArrayList<Character> ary = new ArrayList<>();
          int b = 0;
          while(true){
               b = in.read();
               ary.add((char)b);
               if (b=='\n')
                    break;
          }
          String line = "";
          for(char c : ary){
               line += String.valueOf(c);
          }
          System.out.print(line);
          if(line.equals("\r\n"))
               tag = false;
          return tag;
     }
```

- 发送 HTTP 请求的头部

  由于向服务器发送请求为按行发送，本次项目用 PrintWriter 的 Println()方法实现。

```
PrintWriter pw = new PrintWriter(socket.getOutputStream());
            pw.println("GET "+fileLongName+" HTTP/1.1");
            pw.println("Host: "+addr[0]);
            pw.println("Connection: closed");
            pw.println("Accept-Encoding: deflate, sdch");
            pw.println();
            pw.flush();
```

- 下载文件

  文件的下载由 FileOutputStream 输出流实现。用 DataInputStreamd 从服务器端读取文件数据，得到输入字节流。然后将数据存储到文件中。默认地址为 F 盘下 HTTPClient 文件夹。

```
String fileShortName = addr[addr.length-1];
                FileOutputStream fout = new FileOutputStream(
                        "F:\\HTTPClient\\"+fileShortName);
                byte[] bys = new byte[1024];
                int len = 0;
                while((len = dis.read(bys))!=-1){
                    fout.write(bys,0,len);
                    fout.flush();
                }
                fout.close();
```

（4）遇到的问题及解决方案

i) 问题描述：服务器从客户端传来的数据中包含 HTTP 响应头部，需要显示在客户端的控制台中，如果客户端使用纯碎的字节读取就回造成控制台显示乱码。但如果客户端不使用字节流读取，将不能正确保存文件数据，会造成文件格式错误或文件损坏。

解决方案：在客户端自定义一种函数用以读取一行字节流并转化为字符形式。本项目中 HTTPClient 类下 readALine(InputStream)就能完成此任务。

ii) 问题描述：在用 HTTPClient 类与外界网站做测试时，当数据数据传输完毕后出现程序停运现象，即程序没有关闭，但确实没有数据传输。

解决方案：将 HTTP 请求头部的 Connection 值从 keep-alive 改为 closed。通过尝试，发如果 Connection 的值为 closed，当数据传输完毕后服务器会与本机客户端断开连接。客户端程序立即即关闭，与实际相符。

iii)问题描述：在下载图片时，出现数据丢失情况。图片应为彩色，下载后图片大面积失真。

解决方案：经过 Debug，发现 dout.write(bysData,0,len) 方法错写成了 dout.write(bysData)，即最后 1KB 数据传输失败。

3. 编写 Websocket 聊天室功能，包括客户端 JavaScript 和服务器端 Servlet

（1） 实践内容

利用 WebSocket 编写聊天室程序，程序需满足以下功能：群聊，私聊。当用户进入系统使，系统会广播欢迎信息。当用户退出系统时，系统会广播用户退出信息。在用户发送信息时能够显示发送的时间。当有用户登陆系统时，在线用户列表将添加对应的用户名。同样，当用户退出系统时，在线用户列表将删除该用户的用户名。用户可以进行私聊。

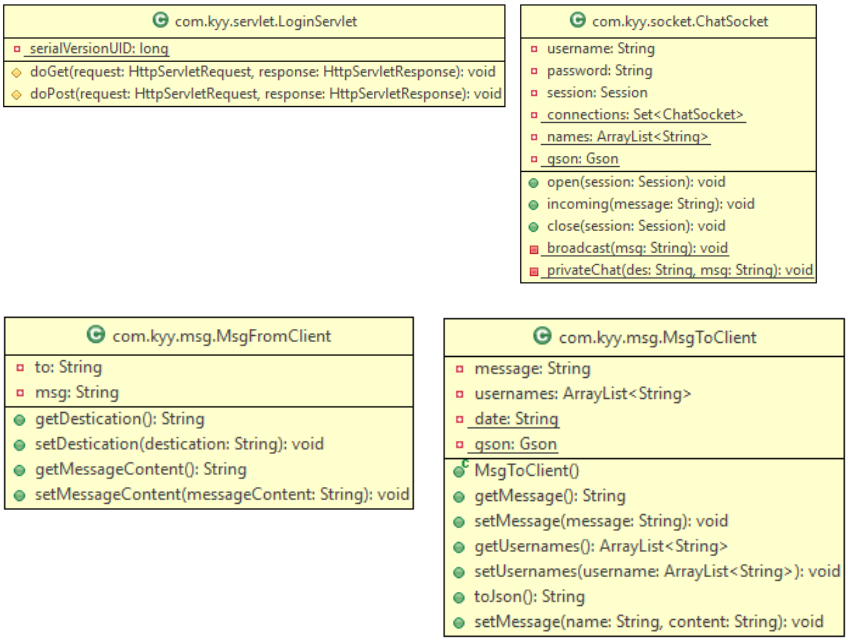
利用 WebSocket 编写客服系统。顾客向客服发送请求，系统自动分配在线客服给顾客。顾客作为系统游客不能访问在线用户列表。客服可以看到对接服务的顾客列表和在线的客服账户列表。客服和顾客之间交流以私聊的形式进行。

（2） 实现过程

i) 类图设计

- 聊天室程序

LoginServlet 实现了 HttpServlet 的两个接口——doGet()和 doPost()，用以处理用户信息。如账户和密码的验证，输入信息的验证等。ChatSocket 类用于存储当前客户端（浏览器）和服务器的信息。包含当前用户的用户名，密码。同时有一个集合包含所有正在运行的 ChatSocket 类和一个集合包含所有的在线用户名。



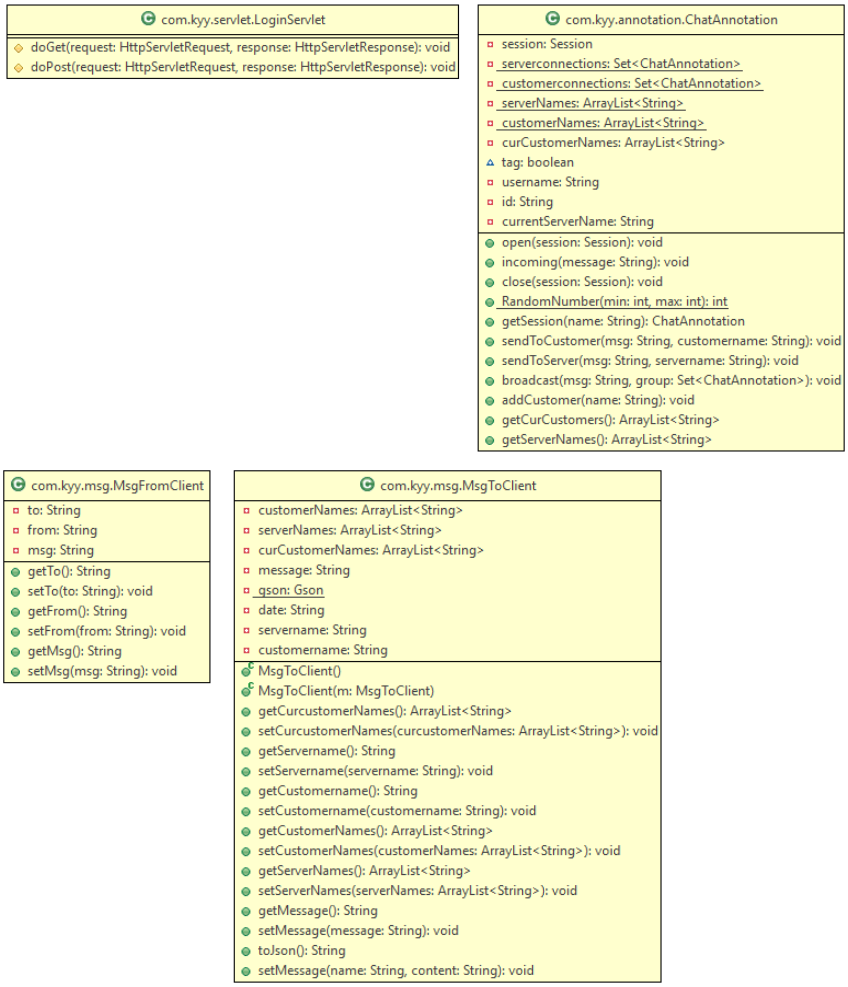
- 客服程序

LoginServlet 继承 HttpServlet，用于获取用户的信息，包括用户的用户名和身份（客户或客服）。

ChatAnnotation 用于存储当前客户端（浏览器）和服务器的信息。包含当前用户的用户名，身份。同时有一个集合包含所有正在运行的 ChatAnnotation 类和一个集合包含所有的在线用户名。

MsgFromClinet 用以解析从客户端发向服务器的消息内容，其中包含消息发行的对象，消息发送来源，消息的具体内容。

MsgToClient 用以封装发向客户端的消息。其包含的信息有当前在线的客户用户名集合，当前的客服名集合，使用此通道客户的用户名和客服名，用以表示日期的字符串等。toJason 方法用以将该消息类封装成字符串形式，用以消息的发送。



（3） 关键技术

i) 聊天室程序

- 读取用户信息

  HttpServletRequest 类的 getPramater()方法能获取从 login.html 页面的表单里获取用户名和密码。

```
String username = request.getParameter("username");
String password = request.getParameter("password");
```

- 保存用户信息并跳转到聊天界面

  利用 Session 的 setAttribute()方法将获得的用户名信息存到名为“username”的属性中，将保存下来的密码复制到名为“password”的属性中。用 HttpResponse 类的 senRedirect()方法将页面跳转到聊天界面，并将用户信息传输到 chat.jsp 中。

```
request.getSession().setAttribute("username", username);
request.getSession().setAttribute("password", password);
response.sendRedirect("chat.jsp");
```

- 私聊方法

  该方法以两个字符串作为参数，分别为信息发送对象的名字和信息内容。该方法中，用 for each 循环遍历 connections（该集合保存所有正在使用的

ChatSocket 对象）。用 if 语句进行判断，如果找到名字对应的 ChatSocket，则调用 session.getBasicRemote().sendText(String)方法，将文字发送到对象的页面中，并退出循环，完成私聊功能。

```
private static void privateChat(String des, String msg){
       for (ChatSocket client : connections) {

         if(client.username.equals(des)){
              try {
                  synchronized (client) {
                   client.session.getBasicRemote().sendText(msg);
                  }
               }
     ......
     ......
                  break;
          }
       }
    }
```

- 广播函数
  该函数只接收一个 String 类型的参数，为广播的内容。当调用方法时，用 for-each 循环遍历 connections 中所有的正处于连接状态的 ChatSocket，并调用 session.getBasicRemote().sendText(String)方法将消息内容发送到每个客户端 jsp 中。

```
private static void broadcast(String msg) {
       for (ChatSocket client : connections) {
          try {
              synchronized (client) {
                 client.session.getBasicRemote().sendText(msg);
              }
           }
     ......
     ......
       }
    }
```

- 将消息处理后发送给客户端浏览器
  为了使每条消息包含更多信息，用一个类 MsgToClient 来封装消息内容，形成消息对象。该对象包含消息的内容，当前在线的所有用户的用户名集合，用于显示日期的字符串。在发送消息时，首先判断发送对象。如果为 GroupChat，就调用广播方法；如果不是，则调用私聊方法。发送消息前用 MsgToClient 类中的 toJason 方法把该消息对象包装成字符串发送。

```
if(des.equals("GroupChat")){
            broadcast(msgToCleint.toJson());
         }else{
            privateChat(des,msgToCleint.toJson());
            privateChat(username,msgToCleint.toJson());
         }
```

- 解析 json 数据（javascript 代码）
  在 chat.jsp 文件中，用 eval 方法解析从服务器传来的数据 message。得到 msg 这个对象。在实际操作中，用 console.info()方法，使用浏览器中的 web 检查器可以很清楚的看到 msg 对象中的内容。

```
eval("var msg = "+ message.data+";");
```

- 更新在线用户列表（javascript 代码，引入 jQuery）
  在解析数据后，判断 msg 对象中 usernames 这一属性是否无定义。如有定义，则先用$("#userlist").html("")方法清空页面中的用户名列表，然后调用 append 方法将 msg 对象中的用户列表重新加载到页面相应位置。即每一次接收消息都会实时刷新用户列表。

```
if(undefined!=msg.usernames){
            $("#userlist").html("");
            $("#userlist").append("<li class = 'namelist'
                    onclick = 'privateMsg(this)'
                    onmousemove = 'changecolor(this)'
                    onmouseout = 'changecolorback(this)'>GroupChat</li>");
            $(msg.usernames).each(function(){
                $("#userlist").append("<li class = 'namelist'
                        onclick = 'privateMsg(this)'
                        onmousemove = 'changecolor(this)'
                        onmouseout = 'changecolorback(this)'>"
                        +this+"</li>");
```

- 在浏览器中消息发送函数（javascript 代码，引入 jQuery）
  当用户点击按钮“发送”时，将会触发以下方法。首先构造一个对象 obj，含有两个属性，to 和 msg。to 为发送的目的地，即消息的发送对象；msg 为待发生送消息的内容，用 val()方法从页面中获取。然后调用 JSON.stringify()方法将此消息对象封装成 String 类，通过 WebSocket 的 send 方法发送给服务器。最后清空文字输入框。

```
function sendmsg()
    {
          var obj = {
               to:des,
               msg:$("#writting").val();
          }
          var str = JSON.stringify(obj);
          ws.send(str);
          $("#writting").val("");
    }
```

ii) 客服程序

客服程序与聊天室相比，主要体现在项目的功能上。如用户无法接受到所有信息的列表，这可以通过对话界面的布局实现。客服只能获得对应服务的客户列表，这可以通过在接收消息时做出判断决定。客服程序这些功能的实现都与聊天室类似，不再于此赘述。

- 服务器广播函数
  该方法接收两个参数，String 类的消息内容和 ChatAnnotation 集合。通过 for-each 循环遍历 group，调用 session.getBasicRemote().sendText()方法将消息发送给集合中的每一个对象。

```
public void broadcast(String msg, Set<ChatAnnotation> group){
    for (ChatAnnotation client : group) {
              try {
                       synchronized (client) {
                           client.session.getBasicRemote().sendText(msg);
                       }
                   }
              ......
              ......
     }
}
```

（4） 遇到的问题及解决方案

i) 问题描述：第一次完成更新用户列表代码时，出现重复显示现象。即第一次出现的名词在第二次更新时没有小时，而更新仅仅在已有的列表下增添上了数据。

解决方案：在添加数据前清空当前列表中的数据。

ii) 问题描述：完成用户列表更新时，如果服务器一条一条地把名字发送给客户端，在循环中添加一个用户名后又会被删除，导致最后列表中只有一个用户名。

解决方案：将用户名列表封装在一个 MsgToClient 内部，再将消息类通过 toJson() 方法封装成字符串，完成一次性传输，这样可以避免循环读信息。

实践总结

1. 本次实践加深了我对 HTTP 协议的理解。比如 HTTP 响应报文和请求报文的构成，每行消息的意义等。在上学期我们已经学习了计算机网络相关技术，这次项目，使我从理论走向实践。用自己写的客户端总 Web 服务器下载自己喜欢的音乐由很大的成就感。
2. 本次实践加深了我对 FTP 协议的理解。熟悉了 FTP 请求和响应的过程。熟悉了 Java 文件传输功能。学习了 Java 多线程的操作。
3. 本次实践我学习了 html, javascript, css 等网络知识，用以实现页面的展示等。

实践成绩评定表

考核内容	考 核 标 准	分值	得分
平时表现	实践过程中，无缺勤、迟到、早退现象，学习态度积极。实践过程未做与本实践无关的事情。	10	
实验（一）内容	能够按实验要求合理设计并开发出FTP客户端与服务器程序，认真记录实验数据，原理及实验结果分析准确，归纳总结充分。程序代码书写规范，简洁，有重要位置的注释内容。	20	
实验（二）内容	能够按实验要求合理设计并开发出HTTP客户端与服务器程序，认真记录实验数据，原理及实验结果分析准确，归纳总结充分。程序代码书写规范，简洁，有重要位置的注释内容。	20	
实验（三）内容	能够按实验要求合理设计并开发出WebSocket聊天室程序，认真记录实验数据，原理及实验结果分析准确，归纳总结充分。程序代码书写规范，简洁，有重要位置的注释内容。	20	
创新性	在实践内容实现上，有自己独立的想法，并在功能实现上有一定的创新性；完成选做实验项目。	10	
报告质量	实验报告格式规范，体例符合要求；报告内容充实、正确，实验目的归纳合理到位，思考题回答准确。	20	
总 分（百分制）			