Android 应用软件设计

E 6　XML or JSON

学号：SA17225268
姓名：彭明

报告撰写时间：2017/11/06

1.主题概述

本次的主题是以 JSON 和 XML 的方式来进行客户端和服务器之间的数据传输。

2.假设

主题内容以 tomcat 服务器作为项目的服务端，在本机上访问时，ip 为 localhost，在模拟器上访问时，ip 为 10.0.2.2。

3.实现或证明

1. 新建 Java web 工程 SCOSServer，在该工程下定义源码包“esd.scos.servlet”，在 SCOSServer 包 esd.scos.servlet 下新建类 LoginValidator 继承 HttpServlet

```
@WebServlet(name = "LoginValidator")
public class LoginValidator extends HttpServlet {
    protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
        request.setCharacterEncoding("UTF-8");
        response.setCharacterEncoding("UTF-8");
        response.setContentType("application/json; charset=utf-8");

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

        OutputStream out = response.getOutputStream();
        if("pengming".equals(username) && "123456".equals(password)){
            out.write("{RESULTCODE:1}".getBytes("UTF-8"));
        } else {
            out.write("{RESULTCODE:0}".getBytes("UTF-8"));
        }

        out.flush();
        out.close();
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        doPost(request, response);
    }
}
```

2. 在 SCOSServer 包 esd.scos.servlet 下新建类 FoodUpdateService 继承 HttpServlet

   其中用到的 Food 类同客户端一样

```
@WebServlet(name = "FoodUpdateService")
public class FoodUpdateService extends HttpServlet {
    protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
        doGet(request, response);
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

        response.setCharacterEncoding("UTF-8");
        response.setContentType("application/json; charset=utf-8");

        List<Food> foods = new ArrayList<Food>();

        for(int i = 0; i < 3; i++){
            Food food = new Food();
            food.setFoodName("农家小炒肉" + i);
            food.setFoodPrice(24 + i);
            food.setKind("热菜");
            foods.add(food);
        }

        String result = JSON.toJSONString(foods);

        OutputStream out = response.getOutputStream();
        out.write(result.getBytes("UTF-8"));
        out.flush();
        out.close();
    }
}
```

在 web.xml 中对两个 servlet 进行绑定

```
  <servlet>
     <servlet-name>LoginValidator</servlet-name>
     <servlet-class>esd.scos.servlet.LoginValidator</servlet-class>
  </servlet>

   <servlet-mapping>
      <servlet-name>LoginValidator</servlet-name>
      <url-pattern>/LoginValidator</url-pattern>
   </servlet-mapping>

   <servlet>
      <servlet-name>FoodUpdateService</servlet-name>

<servlet-class>esd.scos.servlet.FoodUpdateService</servlet-class>
   </servlet>

   <servlet-mapping>
      <servlet-name>FoodUpdateService</servlet-name>
      <url-pattern>/FoodUpdateService</url-pattern>
   </servlet-mapping>
```

3. 修改 SCOS 的 LoginOrRegister 代码，当用户点击“登录”或“注册”按钮时，使用 HttpURLConnection 访问 SCOSServer 的 Servlet 类 LoginValidator

   这里在对用户名和密码进行之前的检查符合之后，调用如下代码：

```
       mAuthTask = new UserLoginTask(username, password);
       mAuthTask.execute((Void) null);
```

```
public class UserLoginTask extends AsyncTask<Void, Void, Boolean> {
      private final String mUsername;
      private final String mPassword;
```

```
        UserLoginTask(String username, String password) {
            mUsername = username;
            mPassword = password;
        }

        @Override
        protected void onPreExecute() {
            mProgressView.setVisibility(View.VISIBLE);
        }

        @Override
        protected Boolean doInBackground(Void... params) {
            HttpURLConnection connection = null;
            BufferedReader reader = null;

            try{
                URL url = new
URL("http://10.0.2.2:8080/SCOSServer/LoginValidator");
                connection = (HttpURLConnection) url.openConnection();
                connection.setRequestMethod("POST");
                connection.setConnectTimeout(8000);
                connection.setReadTimeout(8000);
                connection.setDoInput(true);
                connection.setDoOutput(true);
                connection.setUseCaches(false);

connection.setRequestProperty("Content-Type","application/x-www-form-
urlencoded");
                connection.setRequestProperty("Charset", "UTF-8");
                connection.connect();

                DataOutputStream out = new
DataOutputStream(connection.getOutputStream());
```

```
                String content = "username="+ URLEncoder.encode(mUsername,
"UTF-8");
                content += "&password=" + URLEncoder.encode(mPassword,
"UTF-8");

                out.writeBytes(content);
                out.flush();
                out.close();

                InputStream in = connection.getInputStream();
                reader = new BufferedReader(new InputStreamReader(in));
                StringBuilder response = new StringBuilder();
                String line;
                while ((line = reader.readLine()) != null){
                    response.append(line);
                }

                return parseJSON(response.toString());

            } catch (Exception e) {
                e.printStackTrace();
            } finally {
                if (reader != null){
                    try {
                        reader.close();
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                }
                if(connection != null){
                    connection.disconnect();
                }
            }
            return false;
        }
        @Override
        protected void onPostExecute(final Boolean success) {
            mAuthTask = null;
            mProgressView.setVisibility(View.GONE);

            if (success) {
                User loginUser = new User();
                loginUser.setUserName(mUsername);
                loginUser.setPassword(mPassword);
                if(oldUser){
                    loginUser.setOldUser(oldUser);
                } else {
                    loginUser.setOldUser(oldUser);
                }

                Intent returnIntent = new Intent();
                returnIntent.putExtra("user", loginUser);
                setResult(RESULT_OK, returnIntent);

                SharedPreferences.Editor editor =
getSharedPreferences("userdata", MODE_PRIVATE).edit();
                editor.putString("username", mUsername);
                editor.putInt("loginState", 1);
                editor.apply();

                finish();
            } else {

mPasswordView.setError(getString(R.string.error_incorrect_password));
                mPasswordView.requestFocus();
            }
        }

        @Override
        protected void onCancelled() {
            mAuthTask = null;
            mProgressView.setVisibility(View.GONE);
        }
    }

    private Boolean parseJSON(String data){
        int resultcode = 0;

        try {
            JSONObject jsonObject = new JSONObject(data);
            resultcode = jsonObject.getInt("RESULTCODE");
        } catch (JSONException e) {
            e.printStackTrace();
        }

        if(resultcode == 1){
            return true;
        }

        return false;
    }
```

4. 修改 SCOS 的 UpdateService 代码，在该类中使用 HttpURLConnection 访问 SCOSServer 中 FoodUpdateService 的 doGet()方法，解析返回结果 JSON 信息

```
private List<Food> foods;
```

```
        HttpURLConnection connection = null;
        BufferedReader reader = null;

        try{
            URL url = new
URL("http://10.0.2.2:8080/SCOSServer/FoodUpdateService");
```

```
            connection = (HttpURLConnection) url.openConnection();
            connection.setRequestMethod("GET");
            connection.setConnectTimeout(8000);
            connection.setReadTimeout(8000);

            InputStream in = connection.getInputStream();
            reader = new BufferedReader(new InputStreamReader(in));
            StringBuilder response = new StringBuilder();
            String line;
            while ((line = reader.readLine()) != null){
                response.append(line);
            }

            foods = parseJSON(response.toString());

        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            if (reader != null){
                try {
                    reader.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
            if(connection != null){
                connection.disconnect();
            }
        }
```

这里用到了 Google 的 Gson 开源库

```
private List<Food> parseJSON(String data){
        Gson gson = new Gson();
        List<Food> foods = gson.fromJson(data, new
TypeToken<List<Food>>(){}.getType());
        return foods;
    }
```

5. 改 SCOS 的 UpdateService 代码，当有菜品更新时，使用 MediaPlayer 播放更新提示音，并使用 NotificationManager 在状态栏提示用户“新品上架：菜品数量”，通知中含有“清除”按钮，当点击清除按钮时，通知消除；当点击通知其他区域时，页面跳转至 SCOS 的 MainScreen 屏幕

```
    private MediaPlayer mediaPlayer;
    private Handler mHandler = new Handler();
```

```
        if(foods != null){
            Uri uri = RingtoneManager.getActualDefaultRingtoneUri(this,
RingtoneManager.TYPE_RINGTONE);
            mediaPlayer = MediaPlayer.create(this, uri);

            try {
                mediaPlayer.prepare();
            } catch (IllegalStateException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }
            mediaPlayer.start();
            mHandler.postDelayed(new Runnable() {
                @Override
                public void run() {
                    mediaPlayer.stop();
                }
            }, 3000);
```

```
          Intent detailIntent = new Intent(UpdateService.this,
MainScreen.class);
          PendingIntent pi =
PendingIntent.getActivity(UpdateService.this, 0, detailIntent, 0);


          RemoteViews views = new RemoteViews(getPackageName(),
R.layout.notification);
          PendingIntent btn = PendingIntent.getBroadcast(this, 1, new
Intent("es.source.code.service.notification"), 0);
          views.setOnClickPendingIntent(R.id.btn_update, btn);

          views.setTextViewText(R.id.tv_update, "新品上架：菜品数量" +
foods.size());


          NotificationManager manager = (NotificationManager)
getSystemService(NOTIFICATION_SERVICE);

          Notification notification = new
NotificationCompat.Builder(UpdateService.this)
                 .setContent(views)
                 .setWhen(System.currentTimeMillis())
                 .setSmallIcon(R.mipmap.ic_launcher)
                 .setContentIntent(pi)
                 .build();

          manager.notify(1, notification);
       }
```

接收清除按钮的 BroadcastReceiver

```
public class CleanNotification extends BroadcastReceiver {

   @Override
```

```
    public void onReceive(Context context, Intent intent) {
        NotificationManager manager = (NotificationManager)
context.getSystemService(NOTIFICATION_SERVICE);
        manager.cancel(1);
    }
}
```

在 AndroidManifest.xml 中添加 Action

```
        <receiver
            android:name="es.source.code.br.CleanNotification"
            android:enabled="true"
            android:exported="true">
            <intent-filter>
                <action
android:name="es.source.code.service.notification" />
            </intent-filter>
        </receiver>
```

6. 修改 SCOSServer 工程 FoodUpdateService 代码，使用 XML 封装菜品更新信息，内容要求不变

   用 JAXB 实现 javabean 到 xml 的转换，先在 Food 类属性的 getter 方法上加上注解 @XmlElement(name=" ")对应的属性名，再新建 FoodList 类

```
@XmlRootElement(name="list")
public class FoodList {
    private List<Food> foods;

    @XmlElement(name = "food")
    public List<Food> getFoods() {
        return foods;
    }
    public void setFoods(List<Food> foods) {
        this.foods = foods;
    }
}
```

```
protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

        response.setCharacterEncoding("UTF-8");
        response.setContentType("application/xml; charset=utf-8");

        List<Food> foods = new ArrayList<Food>();

        for(int i = 0; i < 3; i++){
            Food food = new Food();
            food.setFoodName("农家小炒肉" + i);
            food.setFoodPrice(24 + i);
            food.setKind("热菜");
            foods.add(food);
        }

        FoodList foodList = new FoodList();
        foodList.setFoods(foods);

        String result = convertToXml(foodList);

        OutputStream out = response.getOutputStream();
        out.write(result.getBytes("UTF-8"));
        out.flush();
        out.close();
    }
```

```
public static String convertToXml(Object obj){
        String result = null;
        try {
            JAXBContext context =
JAXBContext.newInstance(FoodList.class);
            Marshaller marshaller = context.createMarshaller();
            marshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT,
true);
            marshaller.setProperty(Marshaller.JAXB_ENCODING, "UTF-8");

            StringWriter writer = new StringWriter();
            marshaller.marshal(obj, writer);
            result = writer.toString();
        } catch (JAXBException e) {
            e.printStackTrace();
        }

        return result;
    }
```

7. 修改 SCOS 的 UpdateService 代码，使用 HttpURLConnection 访问 SCOSServer 中 FoodUpdateService 的 doGet()方法，并解析返回结果，保持提示功能不变

   HttpURLConnection 部分和 JSON 一样，只需要修改解析得到的字符串部分，将以下代码

```
foods = parseJSON(response.toString());
```

修改为

```
foods = parseXML(response.toString());
```

增加 parseXML 方法

```
private List<Food> parseXML(String data){

        List<Food> foods = new ArrayList<Food>();
        XmlPullParserFactory factory = null;
```

```java
        try {
            factory = XmlPullParserFactory.newInstance();
            XmlPullParser xmlPullParser = factory.newPullParser();
            xmlPullParser.setInput(new StringReader(data));
            int eventType = xmlPullParser.getEventType();

            String foodName = "";
            int foodPrice = 0;
            String kind = "";
            while (eventType != XmlPullParser.END_DOCUMENT) {
                String nodeName = xmlPullParser.getName();
                switch (eventType) {
                    // 开始解析某个节点
                    case XmlPullParser.START_TAG: {
                        if ("foodName".equals(nodeName)) {
                            foodName = xmlPullParser.nextText();
                        } else if ("foodPrice".equals(nodeName)) {
                            foodPrice =
Integer.parseInt(xmlPullParser.nextText());
                        } else if ("kind".equals(nodeName)) {
                            kind = xmlPullParser.nextText();
                        }
                        break;
                    }
                    // 完成解析某个节点
                    case XmlPullParser.END_TAG: {
                        if ("food".equals(nodeName)){
                            Food food = new Food();
                            food.setFoodName(foodName);
                            food.setFoodPrice(foodPrice);
                            food.setKind(kind);
                            foods.add(food);
                        }
                        break;
                    }
                default:
            }
            eventType = xmlPullParser.next();
        }
    } catch (Exception e) {
        e.printStackTrace();
    }

    return foods;
}
```

4.结论

1. XML 解析方式主要有三类：DOM、SAX、PULL

   DOM 方式：

   优点：整个文档树存在内存中，可对 XML 文档进行操作：删除、修改等等；可多次访问已解析的文档；由于在内存中以树形结构存放，因此检索和更新效率会更高

   缺点：解析 XML 文件时会将整个 XML 文件的内容解析成树型结构存放在内存中并创建新对象，比较消耗时间和内存

   SAX 方式：

   优点：解析效率高、占存少、灵活性高

   缺点：解析方法复杂，代码量大；可拓展性差：无法对 XML 树内容结构进行任何修改

   PULL 方式：

   优点：SAX 的优点 PULL 都有，而且解析方法比 SAX 更加简单

   缺点：可拓展性差：无法对 XML 树内容结构进行任何修改

2. XML：

   优点：
   1. 格式统一，符合标准
   2. 容易与其他系统进行远程交互，数据共享比较方便

   缺点：
   1. XML 文件格式文件庞大，格式复杂，传输占用带宽
   2. 服务器端和客户端都需要花费大量代码来解析 XML，不论服务器端和客户端代码变的异常复杂和不容易维护
   3. 客户端不同浏览器之间解析XML的方式不一致，需要重复编写很多代码
   4. 服务器端和客户端解析 XML 花费资源和时间

   JSON：

   优点：
   1. 数据格式比较简单，易于读写，格式都是压缩的，占用带宽小
   2. 易于解析这种语言，客户端 JavaScript 可以简单的通过 eval_r()进行 JSON 数据的读取
   3. 因为 JSON 格式能够直接为服务器端代码使用，大大简化了服务器端和客户端的代码开发量，但是完成的任务不变，且易于维护

   缺点：
   1. 没有 XML 格式推广的深入人心和使用广泛，没有 XML 那么通用性

5.参考文献

1. mac 上使用 IntelliJ IDEA 创建第一个 javaWeb 项目
   http://www.jianshu.com/p/53afed9941e9

2. httpURLConnection-网络请求的两种方式-get 请求和 post 请求
   http://www.cnblogs.com/fangg/p/5886233.html

3. Android 官方文档中关于 HttpURLConnection 的介绍
   https://developer.android.google.cn/reference/java/net/HttpURLConnection.html

4. Android 官方文档中关于 PendingIntent 的介绍
   https://developer.android.google.cn/reference/android/app/PendingIntent.html

5. JAXB 实现 java 对象与 xml 之间互相转换
   http://www.cnblogs.com/liuk/p/5829389.html