2020年3月20日 星期五

总结 -hudie

1.web端和Android互联

要求在同一局域网。（采用parallel虚拟机 MAC只能通过网卡，虚拟机设置没有wifi设置）

遇到的问题：我安装了虚拟机，虚拟机没有wifi驱动，通过网卡的设置连接，所以虚拟机和手机并没有在同一个互联网中。

解决：装个小米随身wifi,下个驱动

2.互联类 基础的urlConnection 流传输

```
HttpURLConnection connection = (HttpURLConnection) url.openConnection();
    connection.setRequestMethod("GET");//获取服务器数据
    connection.setReadTimeout(8000);//设置读取超时的毫秒数
    connection.setConnectTimeout(8000);//设置连接超时的毫秒数
   InputStream in = connection.getInputStream();
   BufferedReader reader = new BufferedReader(new InputStreamReader(in));
   String result = reader.readLine();//读取服务器进行逻辑处理后页面显示的数据
```

Gitclone问题：

fatal: unable to access 'https://github.com/17801094126/RSADemo.git/': Failed to connect to github.com port 443: Timed out

解决

1.以管理员身份运行中输入cmd。

2.然后输入命令 netsh winsock reset。

3.重启计算机。

git config --global http.proxy

2020年3月20日 星期五

git config --global --unset http.proxy

这个问题出现主要是因为虚拟机我设置了静态ip却忘记了改回来....

443问题出现主要是网络问题，不要怀疑自己的电脑

3.RSA

文章链接：https://blog.csdn.net/weixin_41315081/article/details/89673923（内附Demo）

RSA加密算法是一种非对称的加密算法，是目前最有影响力的公钥加密算法，该算法基于一个十分简单的数论事实：两个大素数相乘十分容易，但是因式分解却很难，因此乘积公开作为加密秘钥—公钥，两个大素数合称为私钥。

问题一：想传送publickey给服务器，但是由于字数太长 以及有特殊符号出现了问题

然后我使用了URL_SAFE这个属性进行测试，发现\n回车后面的数字会被删除

keySize变小传送成功keysize128

设置成分两次传送的方法要配合文件存储使用

改变keysize之后直接传送

公钥不需要传送，加密意义不存在

Notes:先找到对应的Demo，再过一遍，批量学习。

RSA不需要传送公钥，固定生成之后直接让他利用String类型生成秘钥对。

之前的方法注释掉，采用仅传送密文的方式

传送密文会发生截断

String 变量=android.util.Base64.encodeToString(字符串.getBytes(),Base64.DEFAULT);

加密后的字符串带有"\n"，导致在进行字符串比较的时候出现错误，解决办法是将

Base64.DEFAULT替换成Base64.NO_WRAP

Log.e("hh","私钥加密 : “+Base64.encodeToString(result,Base64.NO_WRAPl Base64.URL_SAFE));

可以同时使用

okthttp版本信息记住！！！！！

javax.crypto.lllegalBlockSizeException: Data must not be longer than 64 bytes

at com.sun.crypto.provider.RSACipher.doFinal(RSACipher.java:337)

at com.sun.crypto.provider.RSACipher.engineDoFinal(RSACipher.java:382)

cipher.dofinal有长度限制，所以要截断处理

//Log.e("hh","私钥加密 : “+Base64.encodeToString(result,Base64.URL_SAFEl Base64.NO_WRAP));,

截断处理之后没有效？？？

放弃了这种方式！

选择使用Byte.toString()的方式

输入V

私钥加密、公钥解密——解密：118

这种方式有问题，（猜测是秘钥对复制的问题）但是我测试了我的秘钥是没有问题的?

最后的实现方式是借助了Base64Utils

我这里出现的问题是getinstance java和Android端的不一致，而且java的解码和android的加密方式用的工具类之前一直不一致。问题已解决。

实现方式：

最后的版本参考博客:https://www.cnblogs.com/jpfss/p/10037850.html
https://blog.csdn.net/qq_35605213/article/details/80591869

以下知识点来源于AES，但是我认为解析RSA前面的讲解类似。

为指定算法生成一个 KeyGenerator 对象。

*此类提供（对称）密钥生成器的功能。

*密钥生成器是使用此类的某个 getInstance 类方法构造的。

*KeyGenerator 对象可重复使用，也就是说，在生成密钥后，

*可以重复使用同一 KeyGenerator 对象来生成进一步的密钥。

*生成密钥的方式有两种：与算法无关的方式，以及特定于算法的方式。

*两者之间的惟一不同是对象的初始化：

*与算法无关的初始化

*所有密钥生成器都具有密钥长度 和随机源 的概念。

*此 KeyGenerator 类中有一个 init 方法，它可采用这两个通用概念的参数。

*还有一个只带 keysize 参数的 init 方法，

*它使用具有最高优先级的提供程序的 SecureRandom 实现作为随机源

*（如果安装的提供程序都不提供 SecureRandom 实现，则使用系统提供的随机源）。

*此 KeyGenerator 类还提供一个只带随机源参数的 inti 方法。

*因为调用上述与算法无关的 init 方法时未指定其他参数，

*所以由提供程序决定如何处理将与每个密钥相关的特定于算法的参数（如果有）。

*特定于算法的初始化

*在已经存在特定于算法的参数集的情况下，

*有两个具有 AlgorithmParameterSpec 参数的 init 方法。

*其中一个方法还有一个 SecureRandom 参数，

*而另一个方法将已安装的高优先级提供程序的 SecureRandom 实现用作随机源

*（或者作为系统提供的随机源，如果安装的提供程序都不提供 SecureRandom 实现）。

*如果客户端没有显式地初始化 KeyGenerator（通过调用 init 方法），

*每个提供程序必须提供（和记录）默认初始化。

知识点借鉴的文章：https://blog.csdn.net/u011160994/article/details/47399275

MVVM Android了解模式知道一下
理解模式概念 设计的运行原理
MVVM 简化代码是什么样子的
编码习惯是什么
M V VM？ ？ ？

1. **Mvvm定义**MVVM是Model-View-ViewModel的简写。即模型-视图-视图模型。【模型】指的是后端传递的数据。【视图】指的是所看到的页面。【视图模型】mvvm模式的核心，它是连接view和model的桥梁。它有两个方向：一是将【模型】转化成【视图】，即将后端传递的数据转化成所看到的页面。实现的方式是：数据绑定。二是将【视图】转化成【模型】，即将所看到的页面转化成后端的数据。实现的方式是：DOM 事件监听。这两个方向都实现的，我们称之为数据的双向绑定。总结：在MVVM的框架下视图和模型是不能直接通信的。它们通过ViewModel来通信，ViewModel通常要实现一个observer观察者，当数据发生变化，ViewModel能够监听到数据的这种变化，然后通知到对应的视图做自动更新，而当用户操作视图，ViewModel也能监听到视图的变化，然后通知数据做改动，这实际上就实现了数据的双向绑定。并且MVVM中的View 和ViewModel可以互相通信。

Vm是viewModel，替换的是MVC中的Controller

mvp的全称为Model-View-Presenter，Model提供数据，View负责显示，Controller/Presenter负责逻辑的处理。MVP与MVC有着一个重大的区别：在MVP中View并不直接使用Model，它们之间的通信是通过Presenter(MVC中的Controller)来进行的，所有的交互都发生在Presenter内部，而在MVC中View会直接从Model中读取数据而不是通过 Controller。在MVP里，

Presenter完全把Model和View进行了分离，主要的程序逻辑在Presenter里实现。而且，Presenter与具体的View是没有直接关联的，而是通过定义好的接口进行交互，从而使得在变更View时候可以保持Presenter的不变，即重用！不仅如此，我们还可以编写测试用的View，模拟用户的各种操作，从而实现对Presenter的测试--而不需要使用自动化的测试工具。 我们甚至可以在Model和View都没有完成时候，就可以通过编写Mock Object（即实现了Model和View的接口，但没有具体的内容的）来测试Presenter的逻辑。 在MVP里，应用程序的逻辑主要在Presenter来实现，其中的View是很薄的一层。因此就有人提出了Presenter First的设计模式，就是根据User Story来首先设计和开发Presenter。在这个过程中，View是很简单的，能够把信息显示清楚就可以了。在后面，根据需要再随便更改View，而对Presenter没有任何的影响了。 如果要实现的UI比较复杂，而且相关的显示逻辑还跟Model有关系，就可以在View和Presenter之间放置一个Adapter。由这个 Adapter来访问Model和View，避免两者之间的关联。而同时，因为Adapter实现了View的接口，从而可以保证与Presenter之间接口的不变。这样就可以保证View和Presenter之间接口的简洁，又不失去UI的灵活性。 在MVP模式里，View只应该有简单的Set/Get的方法，用户输入和设置界面显示的内容，除此就不应该有更多的内容，绝不容许直接访问Model—这就是与MVC很大的不同之处。

WPF（Windows Presentation Foundation）是微软推出的基于Windows的用户界面框架，属于.NET Framework 3.0的一部分。它提供了统一的编程模型、语言和框架，真正做到了分离界面设计人员与开发人员的工作；同时它提供了全新的多媒体交互用户图形界面。程序员在WPF的帮助下，要开发出媲美Mac程序的酷炫界面已不再是遥不可及的奢望。 WPF相对于Windows客户端的开发来说，向前跨出了巨大的一步，它提供了超丰富的.NET UI 框架，集成了矢量图形，丰富的流动文字支持(flow text support)，3D视觉效果和强大无比的控件模型框架。

界面风格和windows一样

MVVM： ViewModel包含所有由UI特定的接口和属性，并由一个 ViewModel 的视图的绑定属性，并可获得二者之间的松散耦合，所以需要在ViewModel 直接更新视图中编写相应代码。数据绑定系统还支持提供了标准化的方式传输到视图的验证错误的输入的验证。优点：低耦合、可重用、独立开发、可测试

组成部分：模型，视图，视图模型，绑定器。

MVVM教程链接:
https://juejin.im/post/5d89d9f8f265da03f2340e2b

在DataBinding里，已经处理了null。所以这个时候你在Activity里给user设置为null。也不会崩溃
https://juejin.im/post/5d9c333cf265da5b8a515abb
https://juejin.im/post/5d9d8f756fb9a04dd8591b8e

理解：通过建立ViewModel 绑定在activity上，那么ViewModel的生命周期也会跟随Activity中的oncreateondestroy（这里销毁之后的clear操作会使得ViewModelclear），data—ViewModel —view 绑定有单项和双向绑定，xml绑定
1.转换成databinding的布局规则<data>标签放内容
binding = DataBindingUtil.setContentView(this,R.layout.Xx)
2.variable标签生成会自动生成set，get方法（舍弃findViewById 以及butterknife 框架只会越来越成熟以及特殊的情景和实现越来越专项专能）

3.xml中可以使用类里面的方法 viewModel—view
Android:onclick="@{onclickUtil (类) :: onclickWithMe（方法）}"

4.同名起别称
5.ViewHolder 全局用一个也可以，解放双手
adapter绑定ViewHolder即可
6.单项绑定双向绑定 区别@={} @{}
双向绑定容易有bug 这里可以结合ViewModel
8.include同样可以注入类变量
<include
Layout="@layout/xxx"
app:user="@{user}"

/>
9ViewStub 原理解释理解为慢一步加载（页面被动加载【人主动调用】）
10 imageView 这个比较特殊，加载方法使用百度即可
11databinding不支持 this super new 显示泛型调用？？？
理解:这是一种绑定关系 A&B 所以和泛型就无关了
不然绑定A&B的parent &B & B的son 乱套
11多个fragment公用一个ViewModel
这不是son 这是B的附件

RxJAVA Retrofit

SpringBoot

4.....Room讲解：

https://juejin.im/post/5d9fdacaf265da5bb86ac12c

实现形式：通过注解形式实现SQlite的增删改查

类似于Hibernate Bean Dao(数据库访问) Database（数据库持有者？？？版本管理者） Room（数据库的创建者&管数据库更新）

理解：Database就是抽象类，连接抽象的操作dao
Room就是真实去处理数据库的

```
//添加Room的依赖
implementation 'android.arch.persistence.room:runtime:2.1.4'
annotationProcessor 'android.arch.persistence.room:compiler:2.1.4'
```

（以下实现基本注解！！！！！）
Bean {@Entity： 数据表的实体类。
@PrimaryKey： 每一个实体类都需要一个唯一的标识。
@ColumnInfo： 数据表中字段名称。
@Ignore： 标注不需要添加到数据表中的属性。 li
@Embedded： 实体类中引用其他实体类。(所有属性) 引用两个实体 用@Embeded(prefix="XXXXx")

@ForeignKey： 外键约束。（no action restrict cascade set_null set_default）
}

@Entity(primaryKeys = {"uid","name"})
public class Person {
  //name字段要用@NonNull标注
  @NonNull
  private String name;
}
插入数据的时候加上动@Insert(onConflict=OnConflictStrategy.REPLACE)加上动作，他的意思是主键相同的话，旧数据会替换新数据。但如果我们主键不同，但加了索引唯一性的话，索引相同的话，这次插入则失败。

DAO

- @Dao： 标注数据库操作的类。
- @Query： 包含所有Sqlite语句操作。
- @Insert： 标注数据库的插入操作。
- @Delete： 标注数据库的删除操作。
- @Update： 标注数据库的更新操作。

```
//查询所有数据
@Query("Select * from person")
List<Person> getAll();
//删除全部数据
@Query("DELETE FROM person")
void deleteAll();
```

//根据字段去查找数据
@Query("SELECT * FROM person WHERE uid= :uid")
Person getPersonByUid(int uid);
//一次查找多个数据
@Query("SELECT * FROM person WHERE uid IN (:userIds)")
List<Person> loadAllByIds(List<Integer> userIds);
//多个条件查找
@Query("SELECT * FROM person WHERE name = :name AND age = :age")
Person getPersonByNameage(String name, int age);

```
@Delete
    void delete(Person... persons);

    //一次更新单条数据 或 多条
    @Update
    void update(Person... persons);//
 @Insert(onConflict = OnConflictStrategy.REPLACE)
    @Insert
    void insert(Person... persons);
```

- OnConflictStrategy.REPLACE：冲突策略是取代旧数据同时继续事务
- OnConflictStrategy.ROLLBACK：冲突策略是回滚事务
- OnConflictStrategy.ABORT：冲突策略是终止事务
- OnConflictStrategy.FAIL：冲突策略是事务失败
- OnConflictStrategy.IGNORE：冲突策略是忽略冲突

```
//注解指定了database的表映射实体数据以及版本等信息
@Database(entities = {Person.class, Clothes.class}, version = 1)
public abstract class AppDataBase extends RoomDatabase {

public abstract PersonDao getPersonDao();

    public abstract ClothesDao
getClothesDao();
}
```

这个是在Android中添加插件既可以查看SQlite里面的表
https://blog.csdn.net/qq_34783437/article/details/80770663

更改数据库中的表一定要更新，不然暴崩

```
//修改版本信息为2
@Database(entities = {Person.class, Clothes.class}, version = 2)
public abstract class AppDataBase extends RoomDatabase {
    public abstract PersonDao getPersonDao();

    public abstract ClothesDao getClothesDao();

    //数据库变动添加Migration，简白的而说就是版本1到版本2改了什么东西
    public static final Migration MIGRATION_1_2 = new
Migration(1, 2) {
        @Override
```

```
        public void migrate(SupportSQLiteDatabase database) {
            //告诉person表，增添一个String类型的字段 son
            database.execSQL("ALTER TABLE person ADD
COLUMN son TEXT");
        }
    };
}
public class DBInstance {
//    private static final String DB_NAME = "/sdcard/LianSou/
room_test.db";
    private static final String DB_NAME = "room_test";
    public static AppDataBase appDataBase;
    public static AppDataBase getInstance(){
        if(appDataBase==null){
            synchronized (DBInstance.class){
                if(appDataBase==null){
                    return
Room.databaseBuilder(MyApplication.getInstance(),AppDataBa
se.class, DB_NAME)
                            .allowMainThreadQueries()
                            //加上版本升级信息
                            .addMigrations(AppDataBase.MIGRATION_1
_2)
                            .build();
                }
            }
        }
```

```
        return appDataBase;
    }
}
```

更改的过程也是现象拥有者告知，然后再向操作者操作

数据更改借助RxJava操作，Android主线程一般跟新UI 子线程去处理数据

记录（https://juejin.im/post/5d5ce44d5188252231108e68）这个还没看太多了！！！！
之后可以看一下
这个直接用大佬的demo 非常棒！

5
HTTPD SDK给第三方API
后端的接口
Android轻量级的后端接口 APache 不难

app开启自动启动服务
提供接口 手机ip、helloworld

文档实现一下

NanoHTTPD

```
public AndroidWebServer(int port) {
        super(port);
    }
```

构造器

```
   public Response serve(IHTTPSession session) {
        String msg = "<html><body><h1>Hello server</
h1>\n";
        Map<String, String> parms = session.getParms();
        if (parms.get("username") == null) {
            msg += "<form action='?' method='get'>\n
<p>Your name: <input type='text' name='username'></
p>\n" + "</form>\n";
        } else {
            msg += "<p>Hello, " + parms.get("username") + "!
</p>";
        }
        return newFixedLengthResponse( msg + "</body></
html>\n" );
    }
```

相应之后产生的内容newFixedLengthResponse 主要利用这个类进行传送

```
    private void
initBroadcastReceiverNetworkStateChanged() {
        final IntentFilter filters = new IntentFilter();
```

```
filters.addAction("android.net.wifi.WIFI_STATE_CHA
NGED");

filters.addAction("android.net.wifi.STATE_CHANGE");
        broadcastReceiverNetworkState = new
BroadcastReceiver() {
            @Override
            public void onReceive(Context context, Intent
intent) {
                setIpAccess();
            }
        };

super.registerReceiver(broadcastReceiverNetworkStat
e, filters);
 public boolean isConnectedInWifi() {
        WifiManager wifiManager = (WifiManager)
getSystemService(Context.WIFI_SERVICE);
        NetworkInfo networkInfo =
((ConnectivityManager)
getSystemService(Context.CONNECTIVITY_SERVICE
)).getActiveNetworkInfo();
        if (networkInfo != null &&
networkInfo.isAvailable() &&
networkInfo.isConnected()
```

```
            && wifiManager.isWifiEnabled() &&
networkInfo.getTypeName().equals("WIFI")) {
            return true;
        }
        return false;
    }
```

判断网络状态的权限监听和方法

httpd

6.

FTP

Android 在android 搭建ftpserver实现

目的：做导游的app 点名 创建形成，导入旅客

Ftp传入的应用的，快速创建人员信息

手机ftp文件 照片

http://www.sauronsoftware.it/projects/ftp4j/manual.php

ftp4j文档

所有东西

```
//1、登录至FTPp服务器
        mFTPClient.connect(mFTPHost, mFTPPort);
        //2、请求授权
```

mFTPClient.login(mFTPUser, mFTPPassword);
//3、各种FTP操作
mFTPClient.upload();
mFTPClient.download();
//4、断开FTP连接
mFTPClient.disconnect();

public java.lang.String [] connect(java.lang.String host, int port) throws java.lang.IllegalStateException,java.io.IOException , FTPIllegalReplyException , FTPException
连接（hostname port）

public void login(java.lang.String username , java.lang.String password) throws java.lang.IllegalStateException,java.io.IOException, FTPIllegalReplyException, FTPException
Login（name,password）

public java.lang.String currentDirectory() throws java.lang.IllegalStateException, java.io.IOException,FTPIllegalReplyException, FTPException
当前目录

public long fileSize(java.lang.String path)

指定文件大小

public void deleteFile(java.lang.String path) throws java.lang.IllegalStateException, java.io.IOException, FTPIllegalReplyException, FTPException
删除指定文件
deleteDirectory 删除指定文件夹

public java.lang.String[] listNames()
获取当前工作目录的所有文件名

public void upload (java.io.File file , FTPDataTransferListener listener) throws java.lang.IllegalStateException,java.io.FileNotFoundException, java.io.IOException, FTPIllegalReplyException, FTPException,FTPDataTransferException, FTPAbortedException
上传文件到服务器，该操作会阻塞线程，直至完成

public void download(java.lang.String remoteFileName, java.io.File localFile,FTPDataTransferListener listener)

下载狗哥文件到指定路径

public
void abortCurrentDataTransfer(boolean sendAborCommand) throws java.io.IOException,

中断当前操作sendAborCommand true,falseFTPDataTransferListener　Interface

　　该接口约定了文件传输过程中的回调方法类型。我们可以实现该类去监听每个具体的文件操作过程。

　回调接口有：

void started()　　回调方法，表示已开始文件传输。

void completed()　　回调方法，表示文件传输已完成。

void aborted()　　回调方法，表示文件传输已被中断，可由abortCurrentDataTransfer()触发。

void failed()　　回调方法，表示文件传输由于某种错误而失败。

void transferred(int length)　回调方法，表示本次回调过程中已下载/上传的文件字节大小。　参数：　length --本次回调过程中下载/上传的文件字节大小。　PS：我们可以累积length大小，以便获得当前文件已传输的字节数。

这篇文章是ftp4j的客户端解释工具

https://blog.51cto.com/lavasoft/236934

手机端服务器是SwiFTP(Notes这里具体Demo里面的函数没有看)

okhttp：用的版本 compile ‘com.squareup.okhttp3:okhttp:3.8.1'

7

Websocket 设备心跳，包活人脸机公交车 设备的运行的状态
双工通讯WebSocket
设备主动注册 主动交互

后端像webSocket

WebSocket Demo

一、什么是websocket

WebSocket协议是基于TCP的一种新的网络协议。它实现了客户端与服务器全双工通信，学过计算机网络都知道，既然是全双工，就说明了服务器可以主动发送信息给客户端。这与我们的推送技术或者是多人在线聊天的功能不谋而合。
为什么不使用HTTP 协议呢？这是因为HTTP是单工通信，通信只能由客户端发起，客户端请求一下，服务器处理一下，这就太麻烦了。于是websocket应运而生。搭配Tomcat8 JRE1.5??

至少JRE变成1.8变成尽量新的
WebSocket 做心跳 更新

发送一个照片base64 看看安卓端能不能显示，发特征码

IE低版本的浏览器不支持WebSocket 我的是win7的IE

(下载了个谷歌 ok)

java端 javaee-api-8.0jar

```
/**
 * @ServerEndpoint 注解是一个类层次的注解，它的功能主要是将目前的类定义成一个websocket服务器端,
 * 注解的值将被用于监听用户连接的终端访问URL地址,客户端可以通过这个URL来连接到WebSocket服务器端
 */
@ServerEndpoint("/websocket")

    //concurrent包的线程安全Set，用来存放每个客户端对应的MyWebSocket对象。若要实现服务端与单一客户端通信的话，可以使用Map来存放，其中Key可以为用户标识
        private static CopyOnWriteArraySet<WebSocketTest> webSocketSet = new CopyOnWriteArraySet<WebSocketTest>();


    //与某个客户端的连接会话，需要通过它来给客户端发送数据
    private Session session;


    /**
   * 连接建立成功调用的方法
    * @param session  可选的参数。session为与某个客户端的连接会话，需要通过它来给客户端发送数据
   */
     @OnOpen
     public void onOpen(Session session){
```

```
        this.session = session;
        webSocketSet.add(this);        //加入set中
       addOnlineCount();   //在线数加1
         System.out.println("有新连接加入！ 当前在线人数为" +
getOnlineCount());
 }


       /**
    * 连接关闭调用的方法
    */
       @OnClose
       public void onClose(){
       webSocketSet.remove(this);   //从set中删除
       subOnlineCount();   //在线数减1
         System.out.println("有一连接关闭！ 当前在线人数为" +
getOnlineCount());
       }


       /**
    * 收到客户端消息后调用的方法
    * @param message 客户端发送过来的消息
    * @param session 可选的参数
    */
       @OnMessage
        public void onMessage(String message, Session session) {
       System.out.println("来自客户端的消息:" + message);
```

```
        //群发消息
         for(WebSocketTest item: webSocketSet){
        try {
         item.sendMessage(message);
        } catch (IOException e) {
        e.printStackTrace();
        continue;
        }
        }
        }
```

客户端使用okhttp 3.8

Demo wsManager

```
WsStatusListener wsStatusListener = new WsStatusListener() {
//监听WebSocket
 public void onOpen(Response response) {
      Log.d(TAG, "WsManager-----onOpen");
           tv_content.append(Spanny.spanText("服务器连接成功\n\n",
new ForegroundColorSpan(
                                ContextCompat.getColor(getBaseContext(),
R.color.colorPrimary))));
    }

    @Override
    public void onMessage(String text) {
      Log.d(TAG, "WsManager-----onMessage");
```

```
                    tv_content.append(Spanny.spanText("服务器  "  +
DateUtils.formatDateTime(getBaseContext(),
System.currentTimeMillis(),
            DateUtils.FORMAT_SHOW_TIME) + "\n",
            new ForegroundColorSpan(
                        ContextCompat.getColor(getBaseContext(),
R.color.colorPrimary))));
     tv_content.append(fromHtmlText(text) + "\n\n");
   }

   @Override
   public void onMessage(ByteString bytes) {
     Log.d(TAG, "WsManager-----onMessage");
   }

   @Override
   public void onReconnect() {
     Log.d(TAG, "WsManager-----onReconnect");
         tv_content.append(Spanny.spanText("服务器重连接中...\n",
new ForegroundColorSpan(
                        ContextCompat.getColor(getBaseContext(),
android.R.color.holo_red_light))));
   }

   @Override
   public void onClosing(int code, String reason) {
     Log.d(TAG, "WsManager-----onClosing");
       tv_content.append(Spanny.spanText("服务器连接关闭中...\n",
new ForegroundColorSpan(
```

```
                                ContextCompat.getColor(getBaseContext(),
android.R.color.holo_red_light))));
    }

    @Override
    public void onClosed(int code, String reason) {
      Log.d(TAG, "WsManager-----onClosed");
          tv_content.append(Spanny.spanText("服务器连接已关闭\n",
new ForegroundColorSpan(
                                ContextCompat.getColor(getBaseContext(),
android.R.color.holo_red_light))));
    }

    @Override
    public void onFailure(Throwable t, Response response) {
      Log.d(TAG, "WsManager-----onFailure");
       tv_content.append(Spanny.spanText("服务器连接失败\n", new
ForegroundColorSpan(
                                ContextCompat.getColor(getBaseContext(),
android.R.color.holo_red_light))));
    }
}


  classpath   'com.github.dcendents:android-maven-gradle-plugin:
1.5'
       classpath 'com.jfrog.bintray.gradle:gradle-bintray-plugin:1.7.3'
```

其中wsManger模块里面的Build.gradle 中install{pom.xml,bintray这里面的配置没看明白}

Idea 项目管理工具 Android Studio创建工具 maven仓库

搭建小服务器 下载到本地 通过名称自动更新Intel Idea

8.MQTT
后端有一个MQTTServer 成产这消费者
订阅者被订阅者
同步人员信息（demo）

apache后端有Demo，手机跟他通讯就可以了

MQTT（Message Queuing Telemetry Transport，消息队列遥测传输协议），是一种基于发布/订阅（publish/subscribe）模式的"轻量级"通讯协议，该协议构建于TCP/IP协议上，由IBM在1999年发布。MQTT最大优点在于，可以以极少的代码和有限的带宽，为连接远程设备提供实时可靠的消息服务。作为一种低开销、低带宽占用的即时通讯协议，使其在物联网、小型设备、移动应用等方面有较广泛的应用。
MQTT是一个基于客户端-服务器的消息发布/订阅传输协议。MQTT协议是轻量、简单、开放和易于实现的，这些特点使它适用范围非常广泛。在很多情况下，包括受限的环境中，如：机器与机器（M2M）通信和物联网（IoT）。其在，通过卫星链路通信传感器、偶尔拨号的医疗设备、智能家居、及一些小型化设备中已广泛使用。

精简、发布订阅模式，零运维、传输率低、考虑低带宽，高延迟，不稳定

客户计算能力低，连续会话，不强求传输数据的类型和格式！！！

非常适用于工作场景 基于TCP/IP

```
 // classpath = files(variant.javaCompile.classpath.files) +
files(ext.androidJar)
        doFirst {
            classpath = files(variant.javaCompile.classpath.files,
project.android.getBootClasspath())
        }
```

https://upload-images.jianshu.io/upload_images/4037823-3c30de0d64f0205f.png?imageMogr2/auto-orient/striplimageView2/2/w/1240

下载的jar包
https://repo.eclipse.org/content/repositories/paho-releases/org/eclipse/paho/org.eclipse.paho.client.mqttv3/1.2.2/

运行但是没有服务器....

服务器端搭建：

MQTT官网 http://mqtt.org/

MQTT http://mqtt.org/software

博客实现：

https://blog.csdn.net/yannanxiu/article/details/52703946

配置maven

http://wiki.jikexueyuan.com/project/maven/environment-setup.html

问题（未解决）

安装了服务器demo，按照activemq start之后不能使用tcp://ip:61614访问 ws没问题，Android可以跑 看了js代码，但是没找到链接但是Android这这边用的链接访问，服务器这边index.html中使用mqttjs中Client.onConnect(没找到链接)

远程在线升级 现有代码
App弹出升级，升级重新启动
升级的apk

upgrade实际上是利用webservice访问链接，从网址上复制一个文件。

maven版本记录

1。命名规范 协同开发 注释规范（复盘）

2。异常处理 return new Exception?

3。尽量弹出窗口，给人明示尽量用try{}catch（）{} 记录日志怎么样不打断用户操作

4。日志

https://github.com/facebook/stetho
https://blog.csdn.net/pzm1993/article/details/78264994
Logger工具类
控制台日志

文件日志（Android 日志处理）

网络远程传输用户使用日志

接入友盟SDK 搜集远程

崩溃的日志crash 网络/本地

非功能性的指标

机顶盒+显示器 HDMI

开机以后做应用显示在显示器上 更成熟更稳定

WebSocket httpd 远程升级 提供SDK接口 综合应用