项目原型搭建的必要过程

1. 了解需求相关
2. 搭建项目原型： svn版本控制
   - 素材　公共的工具类
   - 分层分包 ： mvc、mvp等
3. 工具类创建，资源拷贝 ： Application 、CommonUtil
4. 主界面设计，封装搭建。

1、分层分包：

分包：

按业务来分 ： applay、order、logist

按层来分：

- mvc ： Android 原生分层 ： m ：model v：view c:controller（Activity）

Model/DAL层（数据访问层）　　View层/UI层（界面层）　　Controller/BLL层（业务逻辑层）

- 对于复杂的业务 ：MVP 分层
- mvvm分层

2、资源拷贝

基类抽取、搭框架 : 列表界面框架、网络框架封装、基类抽取（BaseActivity、BaseFragment、GMBaseListAdpater、BaseHolder、BaseProtocol）

工具类 : 经验库 ： GMApplication、ToastUtil、IOUtil、LogUtil、CommonUtil(UIUtils)

项目所需的资源：　String Drawable Color

运行WebServer

jar\war\aar 包

jar :java archive

war :web archive　服务器的打包资源文件包

aar : android archive (库项目) 、 v7库 ： java ui视图资源

使用方法：

将　.war 文件 放置在tomcat/webapp目录下的web项目

配置资源文件目录：

运行tomcat会在webapp目录中产生一个与 .war 文件名相同的文件夹，在该文件夹中找到\WEB-INF\classes\ system.properties 修改文件指向的资源路径（该路径为 服务端提供的 配置信息为WebInfos的所在位置。 ）

SVN实现在Android Studio中版本控制的项目原型搭建

1、服务端创建仓库

2、创建项目关联

3、设置建议过滤的文件

4、导入常用的工具类，资源，搭建项目原型及分包

5、提交

创建与导入常用工具类

1、开发过程中用到的公共全局对象及初始化

GMApplication 继承 Application

- 作用： 维护全局的变量、初始化项目的操作 ,百度地图： 开发者id的验证
- 列表清单注册
- 维护全局静态的变量：

Context： 全局的上下文

Hanlder

MainThread

MainThreadId

2. 开发过程中涉及UI常见操作

CommonUtil： 常用操作的工具类

ToastUtil

LogUtil

- 把xml布局实例化view对象
- 取得资源 String Drawable Color
- 取得字符串资源
- 取得字符串数组资源
- 取图片资源
- 取得颜色资源
- dp转px ： 参照当前屏幕密度与标准密度的比例来转化
- px转dp
- 判断当前是否为主线程
- 确保任务在主线程运行（假如任务涉及到UI的操作）

Application类

用于维护项目全局的变量，初始化项目的相关操作 如：上下文Context、Handler、主线程、主线程ID等

```
 * Created by Liu-JinRong on 2017/2/4.
 * 拓展一个Application，用于维护全局的变量，初始化项目的相关操作
 */
public class GMApplication extends Application {
    private static Context mContext;    //应用全局上下文环境
    private static Handler mHandler;    //
    private static Thread mainThread;   //主线程
    private static int mainThreadId;    //主线程ID
    private String nick;
    @Override
    public void onCreate() {
        mContext = getApplicationContext();   //获取全局上下文
        mHandler = new Handler();
        mainThread = Thread.currentThread();  //获取主线程
        mainThreadId = android.os.Process.myTid();//获取主线程ID
    }
```

简单工厂模式根据不同的条件生产同类型的不同对象　如：

```
 * Fragment简单工厂   通过简单的工厂设计模式创建片段
 */
public class FragmentFactory {
    //创建集合对象来保存已有fragment对象页面
    // 好处是：下载再调用可以不用在创建可以对象直接从集合获取
    public static HashMap<Integer,BaseFragment> fragments = new HashMap<>();
    public static Fragment ca(int position){
        //通过集合获取相应位置的fragment对象
        BaseFragment fragment = fragments.get(position);
        //如果fragment对象为空则通过对应位置来创建fragment
        if(fragment==null) {
            switch (position) {
                case 0://首页片段
                    fragment = new HomeFragment();
                    break;
                case 1://应用片段
```

```
                fragment = new AppFragment();
                break;
            case 2://游戏片段
                fragment = new GameFragment();
                break;
            case 3://专题片段
                fragment = new SubjectFragment();
                break;
            case 4://分类片段
                fragment = new CategoryFragment();
                break;
            case 5://推荐片段
                fragment = new RecommendFragment();
                break;
            case 6://排行片段
                fragment = new HotFragment();
                break;
        }
        //将创建好的fragment对象添加到集合中保存
        fragments.put(position,fragment);
    }
    //返回fragment对象
    return fragment;
```

OKHttp

1、 **OkHttp** 是一个高效的 **HTTP** 库（地址：**https://github.com/square/okhttp**）

- 支持**SPDY**，共享同一个**Socket**来处理同一个服务器的所有请求
- 如果**SPDY**不可用，则通过连接池来减少请求延时
- 无缝的支持**GZIP**来减少数据流量
- 缓存响应数据来减少重复的网络请求

2、 SPDY:

SPDY（读作**“SPeeDY”**）是Google开发的基于TCP的应用层协议，用以最小化网络延迟，提升网络速度，优化用户的网络使用体验。**SPDY**并不是一种用于替代**HTTP**的协议，而是对**HTTP**协议的增强。新协议的功能包括数据流的多路复用、请求优先级以及**HTTP**报头压缩。谷歌表示，引入**SPDY**协议后，在实验室测试中页面加载速度比原先快**64%**。

存放位置

1.

外部存储sd卡的文件路径手写：

"mnt/sdcard/文件名.xxx"；　　模拟器默认的路径

通过外界环境去获取路径自动获取sd卡的文件路径：

File f = Environment.getExternalStorageDirectory();　　返回一个文件夹路径

String str = Environment.getExternalStorageDirectory().getPath()+"/文件名.xxx";

外部存储的私有路径中/mnt/sdcard/android/data/包名/files cache 等

Context.getExternalFilesDir(Environment.DIRECTORY_DOCUMENTS);　外部存储的私有路径

getType()+index+getParams();　文件名

2.

手机内部存储

/data/data/应用包名/files/文件名.xxx;

getApplication().getFileStreamPath("文件名.xxx"); 加getPath()方法将路径文件转字符串

自动获取手机内部存储位置

FileInputStream fis = openFileInput("文件名.xxx")　　读出文件

FileOutputStream fos = openFileOutput("文件名.xxx", 文件的权限参数);

Environment 类

Environment 是一个提供访问环境变量的类。

Environment 包含常量：

MEDIA_BAD_REMOVAL

解释：返回getExternalStorageState()，表明SDCard 被卸载前已被移除

MEDIA_CHECKING

解释：返回getExternalStorageState()，表明对象正在磁盘检查。

MEDIA_MOUNTED

解释：返回getExternalStorageState()，表明对象是否存在并具有读/写权限

MEDIA_MOUNTED_READ_ONLY

解释：返回getExternalStorageState()，表明对象权限为只读

MEDIA_NOFS

解释：返回getExternalStorageState()，表明对象为空白或正在使用不受支持的文件系统。

MEDIA_REMOVED

解释：返回getExternalStorageState()，如果不存在 SDCard 返回

MEDIA_SHARED

解释：返回getExternalStorageState()，如果 SDCard 未安装，并通过 USB 大容量存储共享 返回

MEDIA_UNMOUNTABLE

解释：返回getExternalStorageState()，返回 SDCard 不可被安装 如果 SDCard 是存在但不可以被安装

MEDIA_UNMOUNTED

解释：返回getExternalStorageState()，返回 SDCard 已卸掉如果 SDCard 是存在但是没有被安装

Environment 常用方法：

方法：getDataDirectory()

解释：返回 File，获取 Android 数据目录。

方法：getDownloadCacheDirectory()

解释：返回 File，获取 Android 下载/缓存内容目录。

方法：getExternalStorageDirectory()

解释：返回 File，获取外部存储目录即 SDCard

方法：getExternalStoragePublicDirectory(String type)

解释：返回 File，取一个高端的公用的外部存储器目录来摆放某些类型的文件

方法：getExternalStorageState()

解释：返回 File，获取外部存储设备的当前状态

方法：getRootDirectory()

解释：返回 File，获取 Android 的根目录

StatFs 类

StatFs 一个模拟linux的df命令的一个类,获得SD卡和手机内存的使用情况

StatFs 常用方法:

getAvailableBlocks()

解释：返回 Int，获取当前可用的存储空间

getBlockCount()

解释：返回 Int，获取该区域可用的文件系统数

getBlockSize()

解释：返回 Int，大小，以字节为单位，一个文件系统

getFreeBlocks()

解释：返回 Int，该块区域剩余的空间

restat(String path)

解释：执行一个由该对象所引用的文件系统