Do it. Do it right. Do it right now.

昵称：LeoLiang
园龄：4年6个月
粉丝：39
关注：5
+加关注

< 2016年1月 >

日	一	二	三	四	五	六
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31	1	2	3	4	5	6

搜索
找找看
谷歌搜索

常用链接
我的随笔
我的评论
我的参与
最新评论
我的标签
更多链接

我的标签
Android(8)
java(3)
android studio(2)
zygote(2)
常用命令(1)
架构设计(1)
虚拟机(1)
序列化(1)
android architecture(1)
android binder(1)
更多

随笔档案
2016年1月 (6)
2015年12月 (11)
2015年11月 (3)
2012年3月 (2)

最新评论
1. Re:从.NET的宠物商店到Android MVC MVP
@付腾升不能，留在公司了，no backup...
--LeoLiang
2. Re:从.NET的宠物商店到Android MVC MVP
Training Project其实就是做一个购物网

从零开始搭建架构实施Android项目

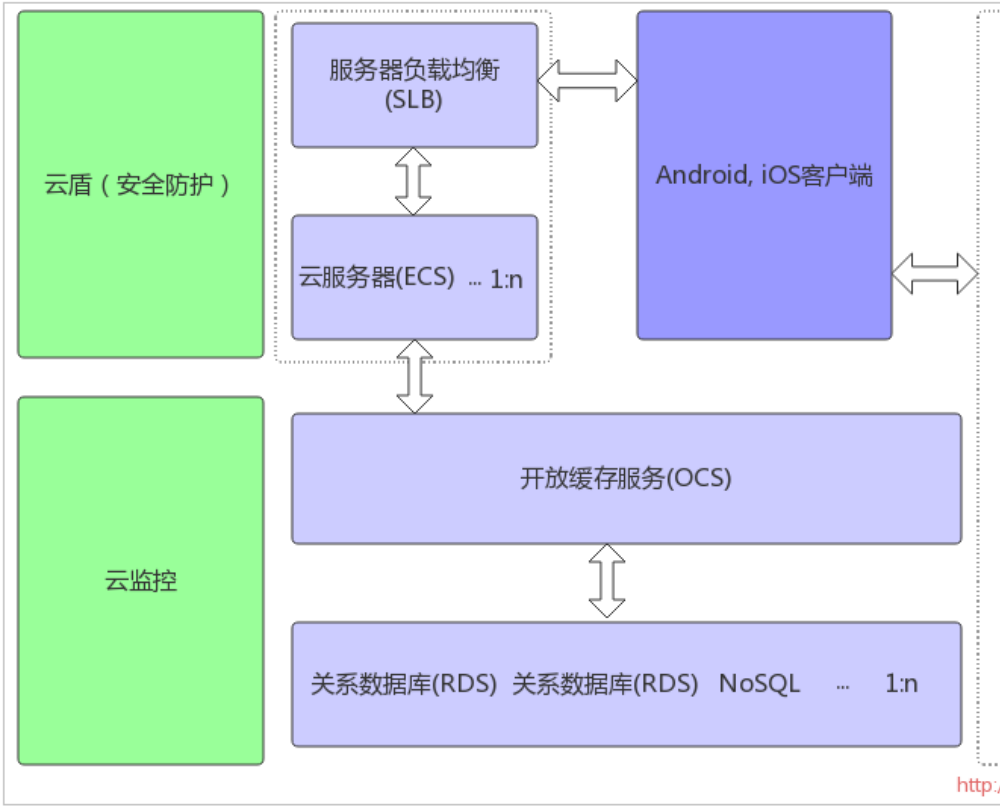
我们先假设一个场景需求：刚有孩子的爸爸妈妈对用照片、视频记录宝宝成长有强烈的意愿，但苦于目前没有一款专门的手机APP做这件事。A公司洞察到市场需求，要求开发团队尽快完成Android客户端的开发。以下模拟团队和工作开展。

- 团队情况：产品经理1人，Android开发2人，服务端开发2人，UI设计1人。
- 开发周期：两个月。
- 工作量：大约50个界面。
- 隐含需求：考虑到用户群体有可能激增的情况，服务端需要有一定的并发能力。
- 前提：原型已设计完成。

1 服务端概要设计

1.1 系统架构

先给出服务端的架构图。



由于服务端开发有Java、PHP背景，为了快速完成开发任务，我们选择PHP作为服务端开发语言，顺便也把数据库定为MySQL。考虑后期扩展和数据库访问性能，拟引入Redis非关系型数据库。同时为了提高数据读的性能，在云服务器和数据库之间用上缓存，并为数据库主从备份、读写分离。服务器就不搭建在本地了，管理是一大问题。现在云服务器一大把，七牛、阿里云、腾讯云、百度云、金山云等等，技术成熟，而且价格还算公道。在此我们选择阿里云。为了应对可能面临的并发问题，云服务器要考虑负载均衡。项目中可能存在大量的需要上传和下载照片和视频，我们选择阿里云的开放存储服务，同时为了提升各个地区的下载体验，我们引入CDN。客户端通过API Service和服务端交换数据，图片和视频的下载直接通过CDN。

1.2 模块划分

根据需求和原型设计，可能的模块划分如下：

- 注册登录模块
- 用户模块
- 小孩模块
- 媒体（图片+视频）模块
- 相册模块
- ……

1.3 数据交换和API接口

服务端与客户端使用JSON交换数据，使用自定义JSON格式，约定返回code、message，实体封装在result中，支持单个实体、实体列表、多个实体列表，定义如下：

```
{
    "code":500,
    "message":"系统异常，请稍后重试",
    "result":""
}
```

```
{
    "code":200,
    "message":"登录成功",
    "result":{
        "user":{
            "userId":1,
            "nickName":"Leo",
            "email":"Leo@xxx.com",
            "gender":0
        }
    }
}
```

```
{
    "code":200,
    "message":"SUCCESS",
    "result":{
        "album":{
            "kid":{
                 "kidId":1,
                 "nickName":"LEE",
                 "gender":1,
                 "birthday":"2014-3-6",
                 ......
             },
            "media":{
                 "mediaId":123,
                 "mediaType":1,
                 "createdTime":193743546746,
                 "mediaDescription":"这是小孩第一次出去春游",
                 ......
            }
        }
    }
}
```

主要API接口设计如下：

```
http://api.xxx.com/service/v1.0/user/login
http://api.xxx.com/service/v1.0/user/third-login
http://api.xxx.com/service/v1.0/user/register
http://api.xxx.com/service/v1.0/user/logout
```

站，做一个WinForm，最后用上WCF。 这个项目的源码可以发给我吗？ 谢谢

--付腾升

3. Re:从零开始搭建架构实施Android项目

对我这个菜鸟有用，赞一个

--玩吧华华

4. Re:从零开始搭建架构实施Android项目

好，不错！

--Xanaduo

5. Re:从零开始搭建架构实施Android项目

@tc310@马三小伙儿@gongyuan073@风来风往风伤@AK47@天翔e派@其实我不笨@幻天芒@小小小的夏@DataCool@MikeXu多谢支持...

--LeoLiang

阅读排行榜

1. 从零开始搭建架构实施Android项目(3691)
2. 从.NET的宠物商店到Android MVC MVP(633)
3. 一个新手写给自己的几句话(522)
4. Android子线程真的不能更新UI么(433)
5. Android开发的那些坑和小技巧(426)

评论排行榜

1. 从零开始搭建架构实施Android项目(26)
2. 一个新手写给自己的几句话(17)
3. Android Studio导入Eclipse项目和一些常见的问题(4)
4. Android子线程真的不能更新UI么(3)
5. Android开发的那些坑和小技巧(2)

推荐排行榜

1. 从零开始搭建架构实施Android项目(22)
2. 从.NET的宠物商店到Android MVC MVP(6)
3. 理解Java虚拟机体系结构(3)
4. 理解Android虚拟机体系结构(2)
5. Android Studio配置Git及Git文件状态说明(2)

```
http://api.xxx.com/service/v1.0/user/info/update
http://api.xxx.com/service/v1.0/album/upload
http://api.xxx.com/service/v1.0/album/update
http://api.xxx.com/service/v1.0/album/delete
......
```

也许你看到了，API做了二级域名映射，同时为了服务端后期API版本的升级管理，在URL中加上了版本标识V1.0。命名方面我尽量做到restful的风格。对了，此处没有使用Https。为了解决数据传输的安全，我做了点特别的处理：对请求体和响应结果进行RSA加密（如果服务端返回的数据稍稍过大，这个RSA严重影响客户端解密，后来我换成了AES），所有请求为POST请求，所以API URL后面没有带参数，你也看不到任何请求相关的信息。

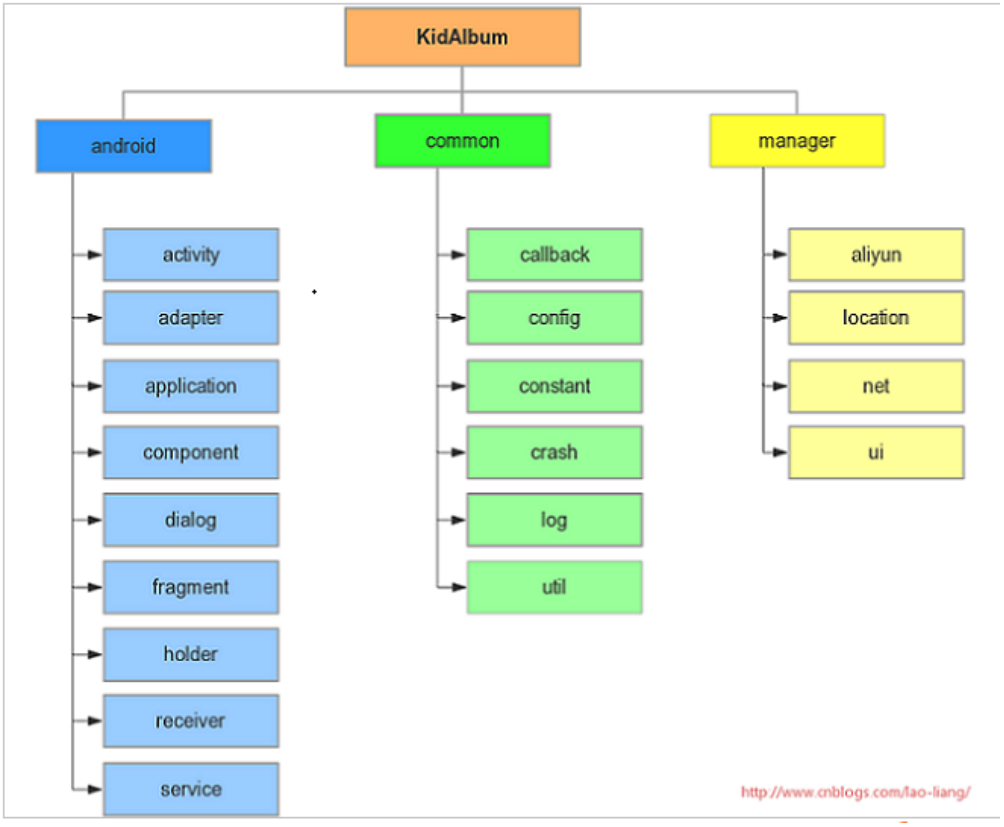
1.4 数据库设计

根据需求和原型设计，数据库的设计大概需要两周时间。其实一周基本搞定了，但为了考虑充分，留出一周时间来检验和调整。数据库E-R图略。

2 Android客户端

2.1 基本结构

Android本身就是MVC，所以我不打算引入MVP或MVVM。我的理念是职责分层，快速推出Android 1.0。主要的包结构如下：



工程的搭建和包的划分有各种各样的，适合自己的就行了。想讨论或想看别人怎么做的，点击这里：App工程结构搭建：几种常见Android代码架构分析

2.2 功能划分

注册登录，个人信息，我的小孩，相册管理，消息通知，系统设置等等。

2.3 引入的第三方技术

重复发明轮子是不可取的。有些模块根本没必要自己写。以下是引入的第三方库，以及优势说明。

2.3.1 网络请求库android-async-http

- 在匿名回调中处理请求结果
- 在UI线程外进行http请求
- 文件断点上传
- 智能重试

- 默认gzip压缩
- 支持解析成Json格式
- 可将Cookies持久化到SharedPreferences

2.3.2 云巴推送

- 专注于为需要实时数据交换的产品提供完美解决方案
- 基于发布者/订阅者(publisher/subscriber)模式，集成简单
- 对比了百度云推送、腾讯信鸽推送，云巴效果更好
- 原极光推送CTO创办的

2.3.3 xUtils（只使用其中的DbUtils和ViewUtils）

- Android中的ORM框架，一行代码就可以进行增删改查
- 支持事务，默认关闭
- 可通过注解自定义表名、列名、外键、唯一性约束、NOT NULL约束、CHECK约束等（需要混淆的时候请注解表名和列名）
- Android中的IOC框架，完全注解方式就可以进行UI，资源和事件绑定
- 新的事件绑定方式，使用混淆工具混淆后仍可正常工作

2.3.4 友盟统计

- 国内专业的移动应用统计分析平台
- 统计和分析流量来源、内容使用、用户属性和行为数据
- Crash log跟踪

2.3.5 云通讯验证码

- 解决方案成熟，众多公司使用

2.3.6 高德地图定位

- GPS+基站+wifi的混合定位方式
- 接入简单

2.3.7 异步图片加载库Android-Universal-Image-Loader

- 多线程下载图片，图片可以来源于网络，文件系统，项目文件夹assets中以及drawable中等
- 支持随意的配置ImageLoader，例如线程池，图片下载器，内存缓存策略，硬盘缓存策略，图片显示选项以及其他的一些配置
- 支持图片的内存缓存，文件系统缓存或者SD卡缓存
- 支持图片下载过程的监听
- 根据控件(ImageView)的大小对Bitmap进行裁剪，减少Bitmap占用过多的内存
- 较好的控制图片的加载过程，例如暂停图片加载，重新开始加载图片，一般使用在ListView,GridView中，滑动过程中暂停加载图片，停止滑动的时候去加载图片
- 提供在较慢的网络下对图片进行加载

2.3.8 阿里云OSS Android客户端SDK

- 提供文件（图片、视频等等）上传
- 大文件分块上传
- 删除操作（不推荐在客户端使用）

2.3.9 组件内通讯EventBus

- 基于发布者/订阅者(publisher/subscriber)模式
- 简化了应用程序内各组件间、组件与后台线程间的通信

2.3.10 Android本地数据库加密库SQLCipher

- 基于SQLite扩展的开源数据库，在SQLite的基础之上增加了数据加密功能
- SQLCipher对Android SDK中所有与数据库相关的API都制作了一份镜像，使得开发者可以像操作普遍的数据库文件一样来操作SQLCipher

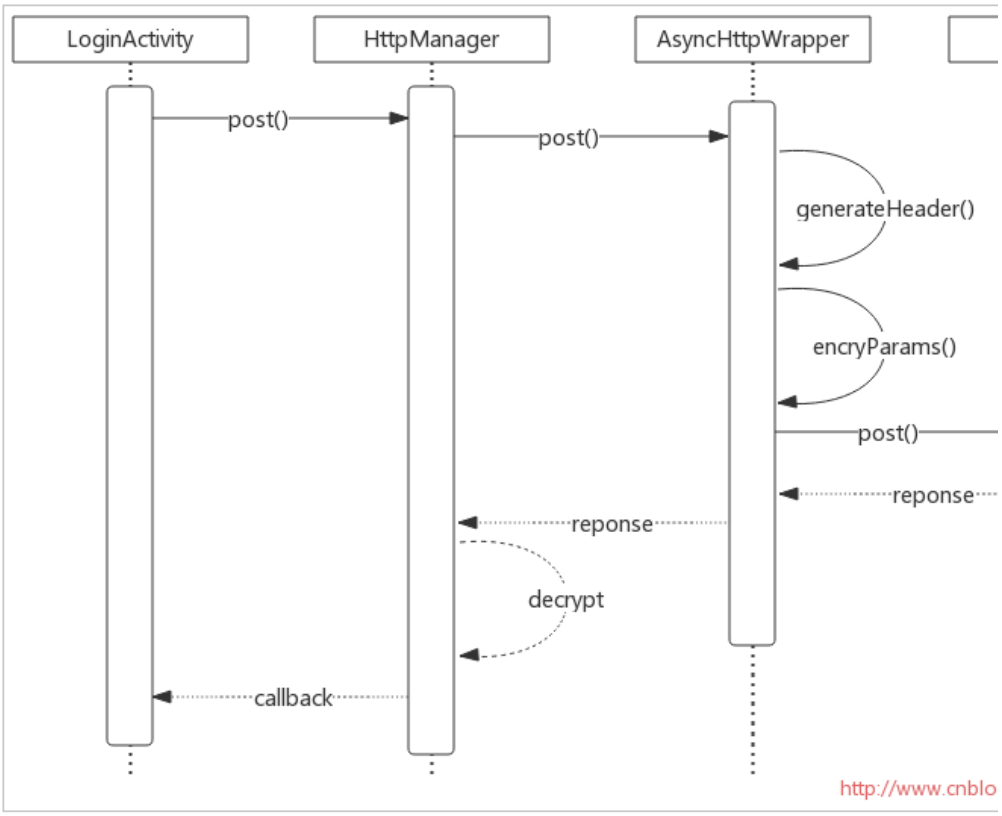
2.4 基础组件封装

2.4.1 基础回调接口

```
public interface DataCallback {

    void onSuccess(Object result);

    void onFailure(Object result);

}
```

2.4.2 网络访问

先看一下登录的序列图：



HttpManager类负责调用AsyncHttpWrapper中的post方法，和对服务端返回的数据解密、JSON转对象、回调上层；AsyncHttpWrapper则负责请求体的封装加密和其它的校验参数封装。看一下HttpManager类的post方法：

```
public void post(Context context, String url, RequestParams params, final String
modelName,
                     final DataCallback callback) {
        AsyncHttpWrapper.post(context, url, params, new AsyncHttpResponseHandler() {
            @Override
            public void onSuccess(int statusCode, Header[] headers, byte[] responseBody)
{
                try {
                    if (modelName != null) {
                        handleResponse(responseBody, callback, modelName);
                    } else {
                        String response = new String(responseBody);
                        // 解密
                        response =
AES128.getInstance().decrypt(AppUtil.decodeReplace(response));

                        // JSON转对象
                        BaseMessage message = AppUtil.getMessage(response);
                        if (callback != null) {
                            // 如果自定义code是200
                            if (Coder.CODE_200.equals(message.getCode())) {
                                callback.onSuccess(message.getMessage());
```

```
                        } else {
                            callback.onFailure(new ServerError(message.getCode()));
                        }
                    }
                }
            } catch (JSONException e) {
                LogUtil.e(e);
                callback.onFailure("服务端返回的数据不能解析成JSON");
            } catch (Exception e) {
                LogUtil.e(e);
                callback.onFailure(e);
            }
        }

        @Override
        public void onFailure(int statusCode, Header[] headers, byte[] responseBody,
Throwable error) {
            if (callback != null) {
                callback.onFailure(error);
                if (responseBody != null) {
                    String s = new String(responseBody);
                    LogUtil.e(s);
                }
            }
        }
    });
}
```

AsyncHttpWrapper中的post方法

```
public static void post(Context context, String url, RequestParams params,
AsyncHttpResponseHandler responseHandler) {
        // 设置请求头部信息
        generateHeader(context);

        // 加密请求参数
        String encryParams = AES128.getInstance().encrypt(params.toString());

        RequestParams requestParams = new RequestParams();
        requestParams.put("param", AppUtil.encodeReplace(encryParams));

        client.post(context, url, requestParams, responseHandler);
    }
```

```
private static AsyncHttpClient client = new AsyncHttpClient();
```

2.4.3 Adapter封装

为了加快开发速度，重用代码，Adapter的使用有技巧。每次在getView中查找控件id、利用ViewHolder、赋值，最后返回convertView，看着都是差不多的代码。是时候脱离这个苦海了。先看怎么解决共用的ViewHolder问题。

```
public static <T extends View> T get(View view, int id) {
        SparseArrayCompat<View> viewHolder = (SparseArrayCompat<View>) view.getTag();
        if (viewHolder == null) {
            viewHolder = new SparseArrayCompat<>();
            view.setTag(viewHolder);
        }
        View childView = viewHolder.get(id);
        if (childView == null) {
            childView = view.findViewById(id);
            viewHolder.put(id, childView);
        }
        return (T) childView;
    }
```

ViewHolder的作用，就是通过convertView.setTag与convertView进行绑定。当convertView复用时，直接从与之对应的ViewHolder(getTag)中拿到convertView布局中的控件，省去了findViewById的时间。上面的代码就是这样

的原理。

然后就是CommonAdapter了。

```
public abstract class CommonAdapter<T> extends BaseAdapter {

    protected LayoutInflater inflater;
    protected Context context;
    protected List<T> datas;
    protected final int itemLayoutId;

    public CommonAdapter(Context context, List<T> datas, int itemLayoutId) {
        this.context = context;
        this.inflater = LayoutInflater.from(context);
        this.datas = datas;
        this.itemLayoutId = itemLayoutId;
    }

    @Override
    public int getCount() {
        return datas != null ? datas.size() : 0;
    }

    @Override
    public T getItem(int position) {
        return datas.get(position);
    }

    @Override
    public long getItemId(int position) {
        return position;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        final CommonViewHolder viewHolder = getViewHolder(position, convertView, parent);
        convert(viewHolder, getItem(position), position);
        return viewHolder.getConvertView();

    }

    public abstract void convert(CommonViewHolder viewHolder, T item, int position);
}
```

好了，不贴代码了。看不明白了请点击这里：Android 快速开发系列 打造万能的ListView GridView 适配器

2.4.5 其它Utils封装

如AES128加密类、BitmapUtils、SecurePreferences、StringUtil、ToastUtil、IOUtil等等。

2.5 接口测试

为了保证数据交换、加解密正常，首先对某一个接口进行测试，以验证API Service能正常跑通。比如可以先对登录进行模拟测试，看是否成功，同时包括异常的测试，服务端是不是处理了边界异常，返回给客户端的都是封装过的异常信息，而不是抛一个敏感信息给客户端。提前进行接口测试有助于我们的基础组件运行没问题，方便后期其它模块的快速集成。

2.6 快速开发

基础组件封装好后，除了少量的从网络获取数据逻辑和本地数据库的增删改查，客户端基本上就是界面的布局工作了。界面开发基本看熟练程度和自定义View的重用。

好了，基本就这些。两个Android开发人员两个月内完成肯定是可以的，前提是至少有一个熟手。后面再谈谈MVP，毕竟这个客户端设计没法进行单元测试，如果业务逻辑越来越复杂，Activity的职责会越来越重，问题多多，不利于后期维护。

标签: Android, 架构设计, android architecture