高效率的Android开发

--新兴框架，模式和工具

张西涛

架构师@千米网

博客http://offbye.com

微信号offbye

Agenda

- Android开发面临的问题
- 什么是高效率的Android开发
- 如何实现高效率开发？
- 框架、模式和工具
- Android开发常见问题
- 总结

如何快速开发高质量的Android App？

Android开发面临的问题

- Android碎片化
- UI设计缺乏规范，按iOS设计做
- 工程师水平参差不齐，人力成本高
- App发布渠道复杂
- 网络安全问题严峻

做高质量的Android App很难

什么不是Android高效率开发

典型场景：我们想做个App，招2个人一个月开发个App

高效率 != 粗制滥造 重复堆代码 加班熬夜 一堆Bug

如何快速开发高质量的Android App？

什么是高效率的Android开发

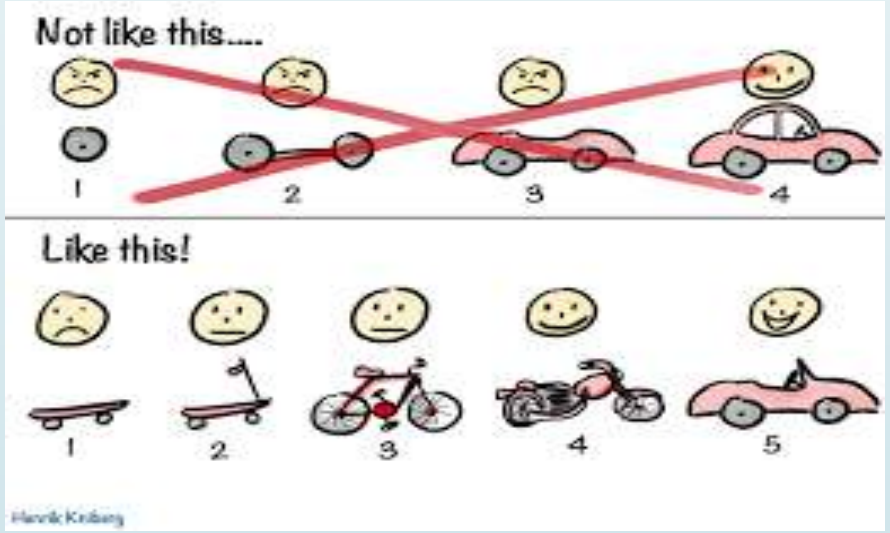
- 项目管理铁三角：范围，成本，时间
- 产品能够可持续的快速迭代
- 良好的用户体验
- 良好的可维护性
- 代码质量高

如何实现高效率开发？

- 深刻理解需求
- 先进的方法论
- 站在巨人的肩膀上 ，善用开源
- 合适的架构设计
- 用好第三方开发服务
- 重视团队技术积累

MVP方法论

- MVP —— Mininum Viable Product，最小可用产品，针对最小用户级的最小产品级
- 最小成本快速验证需求
- 精益开发
- 小心求证、科学试验和深入思考



如何选择开源项目

- Stars, Issues, Pull Requests
- 文档和Demo，典型用户
- 弄清楚原理
- 结合实际业务场景

Github搜索Android
312,304个开源项目
如何选择？

Android开源框架分类

- UI框架和自定义控件
- 网络请求框架
- 图片缓存框架
- 数据存储框架
- 事件总线框架
- 插件化和热部署

UI框架

- View注入框架
  - Butter Knife专注于Android 系统View 的注入框架，结合Android Studio插件使用
- 下拉刷新和加载更多
  - Android-PullToRefresh,android-Ultra-Pull-to-Refresh, Android-PullToRefreshRecyclerView
- 侧边栏菜单
  - SlidingMenu，通过拖动屏幕边缘滑出菜单，支持屏幕左右划出，支持菜单 Zoom、Scale、Slide Up 三种动画样式出现
- base-adapter-helper
  - 对传统的 BaseAdapter ViewHolder 模式的一个封装，主要功能就是简化书写 AbsListView 的 Adapter 的代码，如 ListView，GridView
- MPAndroidChart图表库

Material Design

- Google为了适应并统一手机、车载、电视等设备，设计出一种全新的界面设计准则即Material Design
- 和传统的拟物化界面设计和iOS扁平化界面设计不同，Material Design采取的是一种大量使用阴影和层次化的设计
- 目前新手机基本上是Android5.1
- 使用Material Design支持库


```
compile "com.android.support:appcompat-v7:${supportLibVersion}"
compile "com.android.support:support-v4:${supportLibVersion}"
compile "com.android.support:cardview-v7:${supportLibVersion}"
compile "com.android.support:design:${supportLibVersion}"
compile 'com.android.support:recyclerview-v7:${supportLibVersion}'
compile "com.android.support:support-annotations:${supportLibVersion}"
```

网络请求框架

- ~~自己封装的网络类~~
- Volley
- OKHTTP
- Volley + OKHTTP
- Retrofit
- Retrofit + RxAndroid
- 考虑支持SPDY和HTTP2

Volley + Gson

- Volley 是 Google 推出的 Android 异步网络请求框架和图片加载框架。在 Google I/O 2013 大会上发布。
- 特别适合**数据量小，通信频繁**的网络操作
- 结合Activity生命周期
- 设计优秀，扩展性好

扩展GsonRequest实例

- 统一的Http Header安全验证，集成JWT权限验证，数据签名防篡改
- 统一的返回报文格式BaseResponse<T>，Gson自动化解析

```
public class ItemListResponse extends BaseResponse<Page<Goods>> {
}
private void getGoodsRequest() {
    String url = AppConfig.SERVER_URL + "api/item/list";
    ItemListRequest itemListRequest = new ItemListRequest();
    GsonRequest request = new GsonRequest<>(url, itemListRequest,
            ItemListResponse.class, new Response.Listener<ItemListResponse>() {
        @Override
        public void onResponse(ItemListResponse response) {
            if (response.getStatus() > 0) {
                //TODO
            }
        }
    }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            showSnackMsg("网络不给力");
        }
    });
    startRequest(request);
    showLoadingDialog();
}
```

Retrofit

- Retrofit 是一套 RESTful 架构的 Android（Java）客户端实现，基于注解，提供 JSON to POJO封装
- 管理 REST API 调用最优雅、最方便的解决方案
- Retrofit 默认使用 Gson自动解析JSON

Retrofit turns your HTTP API into a Java interface.

```
public interface GitHubService {
  @GET("users/{user}/repos")
  Call<List<Repo>> listRepos(@Path("user") String user);
}
```

The `Retrofit` class generates an implementation of the `GitHubService` interface.

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("https://api.github.com")
    .build();

GitHubService service = retrofit.create(GitHubService.class);
```

Each `Call` from the created `GitHubService` can make a synchronous or asynchronous HTTP

```
Call<List<Repo>> repos = service.listRepos("octocat");
```

数据存储ORM框架

- DBFlow
- Sprinkles
- SugarORM
- ActiveAndroid
- GreenDAO
- OrmLite
- Realm DB

```
@Table(databaseName = AppDatabase.NAME)
public class TestModel1 extends BaseModel {
    @Column(columnType = Column.PRIMARY_KEY)
    public
    String name;
}
```

```
@ModelView(query = "SELECT * FROM TestModel2 WHERE model_order > 5",
public class TestModelView extends BaseModelView {
    @Column
    long model_order;
}
```

```
@Migration(version = {versionOfMigration}, databaseName = {DatabaseName})
public class Migration1 extends BaseMigration {

    @Override
    public void migrate(SQLiteDatabase database) {

    }
}
```

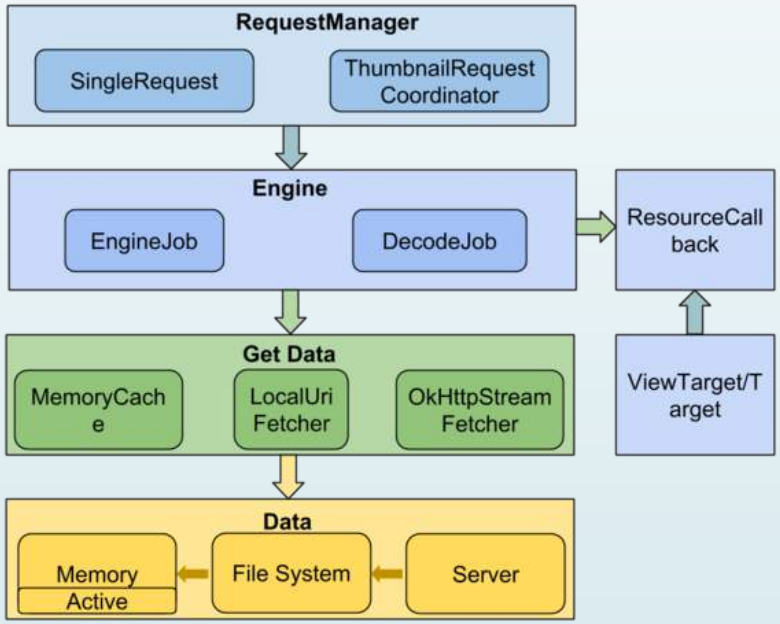
图片缓存框架

- Glide，Google I/O 大会推荐框架，Google的一些官方App使用
- Fresco，FaceBook的开源库，功能超级强大，支持WebP、Gif、JPEG渐进显示，关键是其对图片内存的设计思想，使得图片内存开销大大减少。React Native默认使用
- Picasso，Square的开源库，主导者是 Jake Wharton大神
- Android-Universal-Image-Loader，在出现上述图片库之前的图片库，早期被很多应用使用

```
ImageView imageView = (ImageView) findViewById(R.id.my_image_view);
Glide.with(this).load("http://goo.gl/gEgYUd").into(imageView);
```

图片缓存框架优点

- 使用简单
  - 都可以通过一句代码可实现图片获取和显示。
- 可配置度高，自适应程度高
- 多级缓存
  - 都至少有两级缓存、提高图片加载速度
- 支持多种数据源
  - 支持网络、本地、资源、Assets 等
- 支持多种Displayer
  - 不仅仅支持 ImageView，同时支持其他View 以及虚拟的 Displayer 概念。



事件总线框架

- EventBus
  - 一个发布 / 订阅的事件总线
- Otto
  - Otto 是Square公司出的一个事件库（pub/sub模式），用来简化应用程序组件之间的通讯。
  - Otto 修改自Google的Guava库，专门为Android平台进行了优化
- RxBus
  - RxBus 并不是一个库，而是一种设计思维，它可以巧妙利用 RxJava 的特性，完美替换掉了原事件总线类库

RxAndroid函数响应式编程

- RxJava
- RxAndroid是RxJava的一个针对Android平台的扩展
- RxBinding
- RxBus
- Retrolambda

```
public interface GithubApi {
    @GET("/users/{user}")
    Observable<User> user(@Path("user") String user);

    @GET("/users/{user}")
    User getUser(@Path("user") String user);
}
```

```
_api.user(_username.getText().toString())
        .subscribeOn(Schedulers.io())
        .observeOn(AndroidSchedulers.mainThread())
        .subscribe(new Observer<User>() {
            @Override
            public void onCompleted() {
            }

            @Override
            public void onError(Throwable e) {
            }

            @Override
            public void onNext(User user) {
                //update UI
            }
        });
```

RxAndroid适合场景

优点

复杂异步编程

代码逻辑清晰

线程之间的切换更加方便

代码调试困难

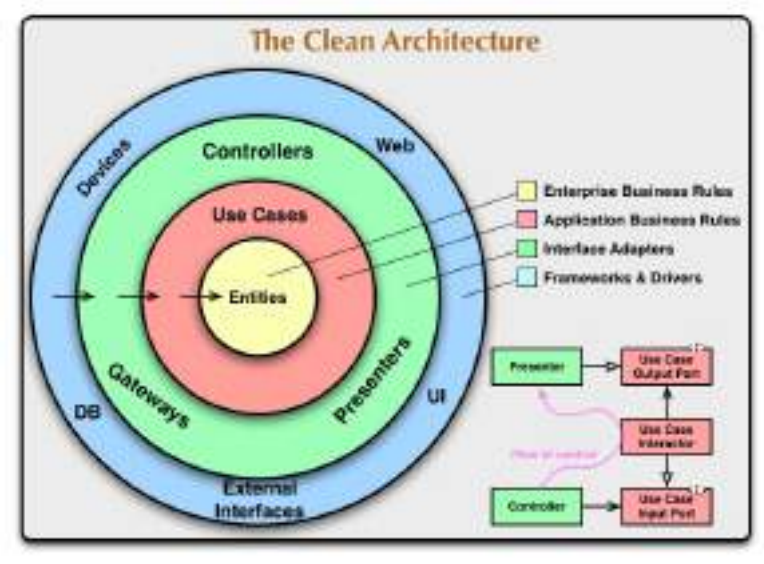
学习曲线比较陡峭

坑点

适合场景：复杂异步场景，团队整体技术水平高

App架构：MVP、MVVM、Clean

- App架构 ？ MVC
- 2016.2-2016.4 谷歌Android Architecture Blueprints的6个Demo
- MVVM（Model View ViewModel）
- MVP（Model View Presenter）
- Clean清晰架构



- 开发者自主选择组织和架构app的方式
- 关注代码结构、整体架构、可测试性、可维护性

Proguard混淆

- ApkTool，dex2jar等工具可以对apk安装包进行反编译
- ProGrard可以对代码进行混淆，并去除没有使用到的代码，对程序进行优化和压缩，减小Apk Size
- ProGrard配置比较复杂
- 常用开源库的ProGuard配置
  - https://github.com/offbye/Android-ProGuardRules

Multidex问题解决

- DEX文件总方法数超过65K，引入开源库太多了？
- Review代码，减少依赖库，启用Proguard优化
- Google官方的multiDex解决方案不完善
  - 修改gradle脚本来产生多dex
  - 修改manifest 使用MulitDexApplication
- FaceBook加载方案
- 微信/手Q加载方案
- 插件化方案

希望Android N jack工具链能彻底解决吧

常用第三方工具

友盟 UMENG

Bugly
最专业的质量跟踪平台

优测
让测试更简单

360加固保
jiagu.360.cn

个推
GeTui.com

环信
easemob.com

总结


深刻理解需求
用好第三方开发服务
高效率Android开发
MVP，快速迭代
选择合适的架构
善用开源框架和组件

Thanks