android网络视频播放框架研究

基于MVP架构的单元测试

##### android网络视频播放框架研究

Running之后

VCL-Android 人气低，基本没有issue，估计是兼容性太差。

JieCanvideoPlayer和VideoListDemo都是使用ijkplayer插件做的二次封装

VideoListDemo是一个Demo for ijkplayer

JieCanVideoPlayer是一个可用的lib，而且视图也比较完善。

Step3：研究ijkplayer

很明显这次使用的核心播放功能是ijkplayer

public interface IMediaPlayer {  
 Do not change these values without updating their counterparts in native  
 void setDisplay(SurfaceHolder sh);  
  
 void setDataSource(Context context, Uri uri)  
 throws IOException, IllegalArgumentException, SecurityException, IllegalStateException;  
  
 @TargetApi(Build.VERSION\_CODES.*ICE\_CREAM\_SANDWICH*)  
 void setDataSource(Context context, Uri uri, Map<String, String> headers)  
 throws IOException, IllegalArgumentException, SecurityException, IllegalStateException;  
  
 void setDataSource(FileDescriptor fd)  
 throws IOException, IllegalArgumentException, IllegalStateException;  
  
 void setDataSource(String path)  
 throws IOException, IllegalArgumentException, SecurityException, IllegalStateException;  
  
 String getDataSource();  
  
 void prepareAsync() throws IllegalStateException;  
  
 void start() throws IllegalStateException;  
  
 void stop() throws IllegalStateException;  
  
 void pause() throws IllegalStateException;  
  
 void setScreenOnWhilePlaying(boolean screenOn);  
  
 int getVideoWidth();  
  
 int getVideoHeight();  
  
 boolean isPlaying();  
  
 void seekTo(long msec) throws IllegalStateException;  
  
 long getCurrentPosition();  
  
 long getDuration();  
  
 void release();  
  
 void reset();  
  
 void setVolume(float leftVolume, float rightVolume);  
  
 int getAudioSessionId();  
  
 MediaInfo getMediaInfo();  
  
 @SuppressWarnings("EmptyMethod")  
 @Deprecated  
 void setLogEnabled(boolean enable);  
  
 @Deprecated  
 boolean isPlayable();  
  
 void setOnPreparedListener(OnPreparedListener listener);  
  
 void setOnCompletionListener(OnCompletionListener listener);  
  
 void setOnBufferingUpdateListener(  
 OnBufferingUpdateListener listener);  
  
 void setOnSeekCompleteListener(  
 OnSeekCompleteListener listener);  
  
 void setOnVideoSizeChangedListener(  
 OnVideoSizeChangedListener listener);  
  
 void setOnErrorListener(OnErrorListener listener);  
  
 void setOnInfoListener(OnInfoListener listener);  
  
 /\*--------------------  
 \* Optional  
 \*/  
 void setAudioStreamType(int streamtype);  
  
 @Deprecated  
 void setKeepInBackground(boolean keepInBackground);  
  
 int getVideoSarNum();  
  
 int getVideoSarDen();  
  
 @Deprecated  
 void setWakeMode(Context context, int mode);  
  
 void setLooping(boolean looping);  
  
 boolean isLooping();  
  
 /\*--------------------  
 \* AndroidMediaPlayer: JELLY\_BEAN  
 \*/  
 ITrackInfo[] getTrackInfo();  
  
 /\*--------------------  
 \* AndroidMediaPlayer: ICE\_CREAM\_SANDWICH:  
 \*/  
 void setSurface(Surface surface);  
  
 /\*--------------------  
 \* AndroidMediaPlayer: M:  
 \*/  
 void setDataSource(IMediaDataSource mediaDataSource);  
}

上面这个接口就是ijkplayer提供给开发者的接口，即API

不过基本上看到方法名就基本可猜出方法的作用。

Step 4

jcvideoplayer-lib 结构

库的核心就是全局只能有一个 IjkMediaPlayer 实例。

所以通过JCMediaManager封装了IjkMediaPlayer，然后JCMediaManager仅提供单例的获取方法。

public interface JCMediaPlayerListener {  
 void onPrepared();  
  
 void onCompletion();  
  
 void onAutoCompletion();  
  
 void onBufferingUpdate(int percent);  
  
 void onSeekComplete();  
  
 void onError(int what, int extra);  
  
 void onInfo(int what, int extra);  
  
 void onVideoSizeChanged();  
  
 void goBackThisListener();  
  
 boolean goToOtherListener();  
  
}

JCMediaPlayerListener 接口控制界面控制UI和视频流的“交互”

JCVideoPlayer 和 JCMediaManager 被联系起来

同时 JCVideoPlayer持有 JCMediaManager 类的实例（single）

即JCVideoPlayer做控制UI，持有容器，在适当的时候放入Player

在addTextureView()

textureViewContainer.addView(JCMediaManager.*textureView*, layoutParams);

另外使用TextureView包裹视频流

public class JCResizeTextureView extends TextureView {  
 public JCResizeTextureView(Context context) {  
 super(context);  
 }  
  
 public JCResizeTextureView(Context context, AttributeSet attrs) {  
 super(context, attrs);  
 }  
  
 @Override  
 protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {  
 int videoWidth = JCMediaManager.*instance*().currentVideoWidth;  
 int videoHeight = JCMediaManager.*instance*().currentVideoHeight;  
  
 int width = *getDefaultSize*(videoWidth, widthMeasureSpec);  
 int height = *getDefaultSize*(videoHeight, heightMeasureSpec);  
 if (videoWidth > 0 && videoHeight > 0) {  
  
 int widthSpecMode = MeasureSpec.*getMode*(widthMeasureSpec);  
 int widthSpecSize = MeasureSpec.*getSize*(widthMeasureSpec);  
 int heightSpecMode = MeasureSpec.*getMode*(heightMeasureSpec);  
 int heightSpecSize = MeasureSpec.*getSize*(heightMeasureSpec);  
  
 if (widthSpecMode == MeasureSpec.*EXACTLY* && heightSpecMode == MeasureSpec.*EXACTLY*) {  
 // the size is fixed  
 width = widthSpecSize;  
 height = heightSpecSize;  
  
 // for compatibility, we adjust size based on aspect ratio  
 if (videoWidth \* height < width \* videoHeight) {  
 width = height \* videoWidth / videoHeight;  
 } else if (videoWidth \* height > width \* videoHeight) {  
 height = width \* videoHeight / videoWidth;  
 }  
 } else if (widthSpecMode == MeasureSpec.*EXACTLY*) {  
 // only the width is fixed, adjust the height to match aspect ratio if possible  
 width = widthSpecSize;  
 height = width \* videoHeight / videoWidth;  
 if (heightSpecMode == MeasureSpec.*AT\_MOST* && height > heightSpecSize) {  
 // couldn't match aspect ratio within the constraints  
 height = heightSpecSize;  
 }  
 } else if (heightSpecMode == MeasureSpec.*EXACTLY*) {  
 // only the height is fixed, adjust the width to match aspect ratio if possible  
 height = heightSpecSize;  
 width = height \* videoWidth / videoHeight;  
 if (widthSpecMode == MeasureSpec.*AT\_MOST* && width > widthSpecSize) {  
 // couldn't match aspect ratio within the constraints  
 width = widthSpecSize;  
 }  
 } else {  
 // neither the width nor the height are fixed, try to use actual video size  
 width = videoWidth;  
 height = videoHeight;  
 if (heightSpecMode == MeasureSpec.*AT\_MOST* && height > heightSpecSize) {  
 // too tall, decrease both width and height  
 height = heightSpecSize;  
 width = height \* videoWidth / videoHeight;  
 }  
 if (widthSpecMode == MeasureSpec.*AT\_MOST* && width > widthSpecSize) {  
 // too wide, decrease both width and height  
 width = widthSpecSize;  
 height = width \* videoHeight / videoWidth;  
 }  
 }  
 } else {  
 // no size yet, just adopt the given spec sizes  
 }  
 setMeasuredDimension(width, height);  
 }  
}

最终所有的自定义实现都继承JCVideoPlayer，但是要注意xml中控件的名称

大致结构这样完。

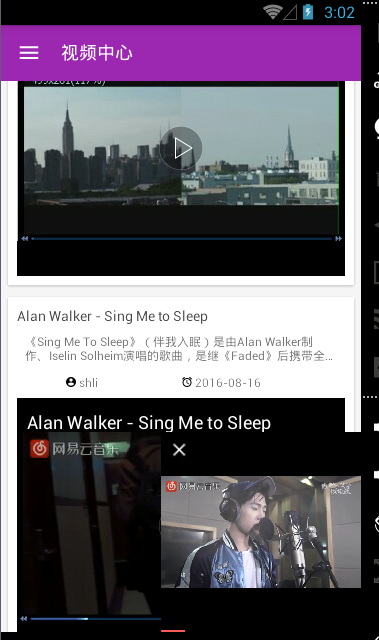
Step5 改善使用

通过视频接口

<http://172.19.24.23/fr-here/rest/app/movie/datalist?pageNum=1&pageSize=15>

提供视频服务，做出的效果。





经过不断的调整

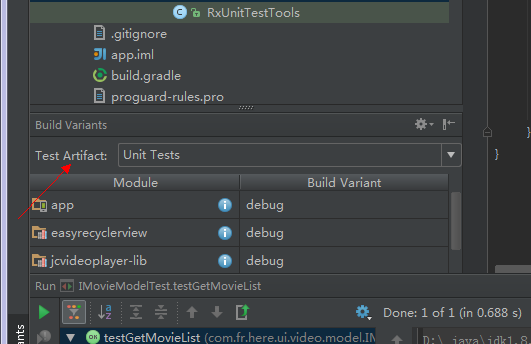
目前还存在的问题：小窗口视频有些问题，不过解决方法写在app的注释中

会出现内存泄露，但是没有发现原因（不严重）

##### 基于MVP架构的Java单元测试

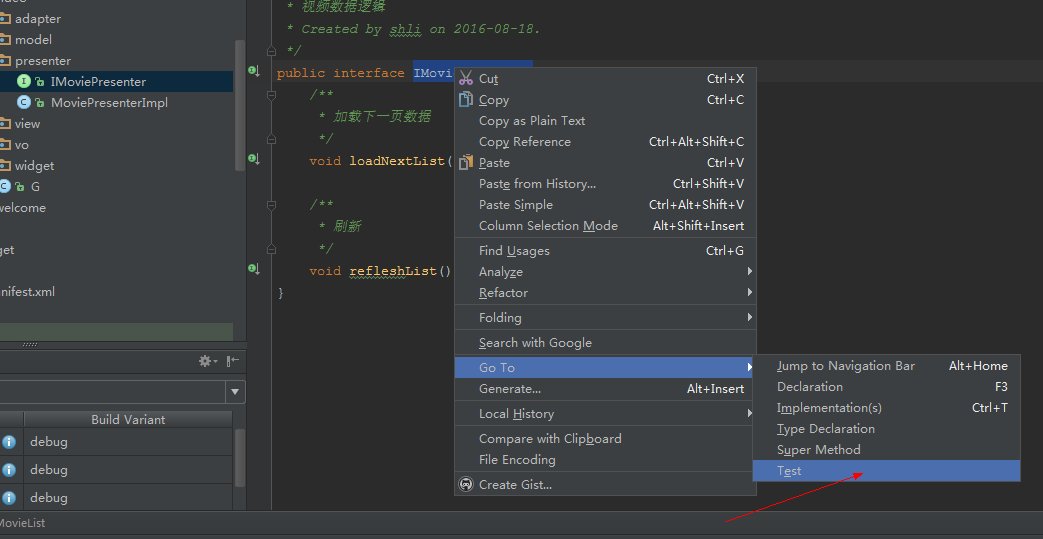
**MVP模式**通过接口编程，把依赖Android SDK的View层(Activity)与Presenter、Model隔离。

当P层、网络M层不依赖Android SDK，我们就可以用JUnit写单元测试，并直接运行在JVM上了。



选择UnitTests，就是进行Java单元测试了。

###### 对Prestenter层进行单元测试



按照这种写法：

public class IMoviePresenterTest {  
  
 IMovieView movieView;  
 IMoviePresenter moviePresenter;  
 IMovieModel movieModel;  
  
 @Before  
 public void setUp() throws Exception {  
 RxUnitTestTools.*openRxTools*();  
 movieModel = *mock*(IMovieModel.class);  
 movieView = *mock*(IMovieView.class);  
  
 moviePresenter = new MoviePresenterImpl(movieModel,movieView);  
 }  
  
 @Test  
 public void testRefleshList() throws Exception {  
 Movie movie = new Movie(1,new Date(),"aclululu","aclululululu","ab","ac","shli");  
 BaseResponse<Movie> baseResponse = new BaseResponse<Movie>();  
 List<Movie> list = new ArrayList<>();  
 list.add(movie);  
 baseResponse.list =list;  
 *when*(movieModel.getMovieList(*anyInt*())).thenReturn(Observable.*just*(baseResponse));  
 moviePresenter.refleshList();  
 ArgumentCaptor<List> captor = ArgumentCaptor.*forClass*(List.class);  
  
 *verify*(movieModel).getMovieList(1);  
 *verify*(movieView).setMovieListRefresh(captor.capture());  
  
 List<Movie> result = captor.getValue(); // 捕获的User  
  
 Assert.*assertEquals*(result.get(0).id, 1);  
 }  
}

when...thenReturn...

when(movieModel.getMovieList(anyInt())).thenReturn(Observable.just(baseResponse));

当调用movieModel.getMovieList(...)，参数为任意int，返回Observable.just(baseResponse)对象。

verify

verify(movieModel). getMovieList (1);，验证 movieModel.getMovieList(...)是否被调用，并校验传入参数pagenum==1。

这一步很重要，这个getMovieList (pagenum)参数比较少，当方法参数多时（例如loadXXX(int,int,int,int...String,String....)），特别容易搞错。当后端接口修改了，service相应也要修改，这时多参数的方法很容易出问题。

verify(movieView).onUserLoaded(captor.capture());，验证movieView. (...)是否被调用，并捕获传入的List<Movie>参数

ArgumentCaptor

顾名思义**参数捕获器**，就是捕获传入参数。当movieModel.getMovieList(...)执行完并返回Observable<BaseReponse<Movie>>，在onNext(user)回调User传给movieView. setMovieListRefresh (...)，但我们不确定回调的list是否正确。因此我们需要捕获list参数，并校验其正确性。

如果参数是List<T>类型，ArgumentCaptor<List> captor = ArgumentCaptor.forClass(List.class)即可，不需要写List泛型参数。

assertEquals

这个不用说了吧.....

类中使用到工具类： 将线程同步

public class RxUnitTestTools {  
 private static boolean *isInitRxTools* = false;  
  
 */\*\*  
 \* 把异步变成同步，方便测试  
 \*/* public static void openRxTools() {  
 if (*isInitRxTools*) {  
 return;  
 }  
 *isInitRxTools* = true;  
  
 RxAndroidSchedulersHook rxAndroidSchedulersHook = new RxAndroidSchedulersHook() {  
 @Override  
 public Scheduler getMainThreadScheduler() {  
 return Schedulers.*immediate*();  
 }  
 };  
  
 RxJavaSchedulersHook rxJavaSchedulersHook = new RxJavaSchedulersHook() {  
 @Override  
 public Scheduler getIOScheduler() {  
 return Schedulers.*immediate*();  
 }  
 };  
  
 RxAndroidPlugins.*getInstance*().registerSchedulersHook(rxAndroidSchedulersHook);  
 RxJavaPlugins.*getInstance*().registerSchedulersHook(rxJavaSchedulersHook);  
 }  
}

###### 对model层进行单元测试

MockInterceptor的职责，读取本地数据，并直接返回。因此，OkHttpClient并没有真正请求网络数据，而是用了本地数据。

public class MockRetrofitHelper {  
  
 public <T> T create(Class<T> clazz) {  
 OkHttpClient client = new OkHttpClient.Builder()  
 .addInterceptor(new MockInterceptor())  
 .build();  
  
 GsonBuilder builder = new GsonBuilder();  
 // Register an adapter to manage the date types as long values  
 builder.registerTypeAdapter(Date.class, new JsonDeserializer<Date>() {  
 public Date deserialize(JsonElement json, Type typeOfT, JsonDeserializationContext context) throws JsonParseException {  
 return new Date(json.getAsJsonPrimitive().getAsLong());  
 }  
 });  
 Gson gson = builder.create();  
  
 Retrofit retrofit = new Retrofit.Builder().baseUrl("http://api.\*\*\*.com")  
 .client(client)  
 .addCallAdapterFactory(RxJavaCallAdapterFactory.*create*())  
 .addConverterFactory(GsonConverterFactory.*create*(gson))  
 .build();  
  
 return retrofit.create(clazz);  
 }  
  
 private String path;  
  
 */\*\*  
 \* 直接放内容  
 \** ***@param*** *path  
 \*/* public void setPath(String path) {  
 this.path = path;  
 }  
  
 private class MockInterceptor implements Interceptor {  
  
 @Override  
 public Response intercept(Chain chain) throws IOException {  
 // 模拟网络数据  
 String content = path;  
  
 ResponseBody body = ResponseBody.*create*(MediaType.*parse*("application/x-www-form-urlencoded"), content);  
  
 Response response = new Response.Builder().request(chain.request())  
 .protocol(Protocol.*HTTP\_1\_1*)  
 .code(200)  
 .body(body)  
 .build();  
 return response;  
 }  
 }  
}

测试类

public class IMovieModelTest {  
 MockRetrofitHelper retrofit;  
 FrHereApi movieModel;  
  
 @Before  
 public void setUp() throws Exception {  
 retrofit = new MockRetrofitHelper();  
  
 movieModel = retrofit.create(FrHereApi.class);  
 }  
  
 @Test  
 public void testGetMovieList() throws Exception {  
 retrofit.setPath("{\"result\":{\"requltType\":null,\"errorMessage\":null},\"list\":[{\"id\":6,\"cdate\":1471337988000,\"creator\":\"shli\",\"picurl\":\"/fr-here/appmovie\_image/movie\_20160816165948.png\",\"movieurl\":\"/fr-here/appmovie\_movie/movie\_20160816165948.mp4\",\"title\":\"中国乒乓球队 - 乒乒乓乓天下无双 (第九季 ： 乒乓海洋)\",\"abstract\_\":\"中国乒乓球队\\r\\n简介：中国乒乓球队成立于1952年，拼搏不息，攀登不止，经历了由弱到强、持久昌盛的发展历程。中国乒乓球队包括中国女子\"},{\"id\":5,\"cdate\":1471337870000,\"creator\":\"shli\",\"picurl\":\"/fr-here/appmovie\_image/movie\_20160816165750.png\",\"movieurl\":\"/fr-here/appmovie\_movie/movie\_20160816165750.mp4\",\"title\":\"Skrillex - Purple Lamborghini\",\"abstract\_\":\"《Purple Lamborghini》由美国DJSkrillex、美国说唱歌手Rick Ross共同制作完成。该单曲收录在电影《X特遣队》原声曲目中。歌曲信息美国DJSkriilex、说唱歌手Rick Ross为电影《X特遣队》献曲。\"},{\"id\":4,\"cdate\":1471337845000,\"creator\":\"shli\",\"picurl\":\"/fr-here/appmovie\_image/movie\_20160816165725.png\",\"movieurl\":\"/fr-here/appmovie\_movie/movie\_20160816165725.mp4\",\"title\":\"Charlie Puth - We Don't Talk Anymore\",\"abstract\_\":\"《We Don't Talk Anymore》由Charlie Puth等人创作，邀请到美国流行歌手Selena Gomez助阵献声，将于2016年作为本张专辑的第三首单曲发行。Charlie Puth在日本的旅行中创作了这首歌的旋律，回到LA之后完成录制工作。\"},{\"id\":3,\"cdate\":1471337919000,\"creator\":\"shli\",\"picurl\":\"/fr-here/appmovie\_image/movie\_20160816165839.png\",\"movieurl\":\"/fr-here/appmovie\_movie/movie\_20160816165839.mp4\",\"title\":\"Alan Walker - Sing Me to Sleep\",\"abstract\_\":\"《Sing Me To Sleep》（伴我入眠）是由Alan Walker制作、Iselin Solheim演唱的歌曲，是继《Faded》后携带全新舞曲风格的作品，收录于2016年6月3日发行的同名专辑中。《Sing Me To Sleep》的音乐标题听似在描述爱情生活的甜蜜无比实则与其有天壤之别。\"},{\"id\":2,\"cdate\":1471337717000,\"creator\":\"shli\",\"picurl\":\"/fr-here/appmovie\_image/movie\_20160816165517.png\",\"movieurl\":\"/fr-here/appmovie\_movie/movie\_20160816165517.mp4\",\"title\":\"微微一笑很倾城\",\"abstract\_\":\"根据顾漫同名小说改编，都市青春偶像剧《微微一笑很倾城》，8月22日起优酷独播，微微一笑在优酷。\"},{\"id\":1,\"cdate\":1471337111000,\"creator\":\"shli\",\"picurl\":\"/fr-here/appmovie\_image/movie\_20160816164511.png\",\"movieurl\":\"/fr-here/appmovie\_movie/movie\_20160816164511.mp4\",\"title\":\"谭维维 - 无问\",\"abstract\_\":\"《无问》是电影《盗墓笔记》主题曲。作词人：尹约作曲人：Julian Emery、谭维维、刘迦宁由谭维维演唱的一首歌曲。基本信息电影《盗墓笔记》主题曲 作词：尹约 作曲：Julian Emery、谭维维、刘迦宁 演唱：谭维维 发行时间：2016-07-29\"}],\"model\":null}");  
 TestSubscriber<BaseResponse<Movie>> testSubscriber = new TestSubscriber<>();  
  
 BaseResponse<Movie> baseResponse = movieModel.getMovieList(1, 10)  
 .toBlocking()  
 .first();  
  
  
 List<Movie> movies = baseResponse.list;  
 Assert.*assertEquals*(movies.get(0).id, 6);  
 Assert.*assertEquals*(movies.get(1).id, 5);  
  
 }  
}

最后获取list便可以验证请求是否正确。

#### 总结

单元测试在有复杂的逻辑时还是必须要写的。