第五章(网络)作业Q&A

大作业的API?

是不是只有网络课上这两个?

是的，POST和GET已经足够完成大作业的发布和展示需求。

想做特别的东西，例如评论/点赞/转发/个人主页?

可以Mock随机数据，因为本课程主要是Android开发，把你们设计的UI和交互的效果实现出来即可。

网络请求和更新UI的线程?

建议看看Android官方文档线程，注意下面的两点：

> 此外，Android UI 工具包*并非*线程安全工具包。因此，您不得通过工作线程操纵 UI，而只能通过 UI 线程操纵用户界面。因此，Android 的单线程模式必须遵守两条规则：
>
> 1. 不要阻塞 UI 线程
> 2. 不要在 UI 线程之外访问 Android UI 工具包

1. 网络请求不能在主线程上执行

因为网络请求耗时较长，在主线程执行的话，会导致app的界面卡死无法响应，系统为了避免这个问题，禁止在主线程发网络请求。

2. 更新UI只能在主线程

不要在主线程之外更新UI，这可能导致出现不明确、不可预见的行为，但要跟踪此行为困难而又费时。

3. 建议的做法

结合以上两点，推荐的做法是，在其他线程执行网络请求，回到主线程更新UI。
方法一：使用新的线程

```
new Thread() {
    @Override public void run() {
        // 在后台线程执行请求
        final String s = NetworkUtils.getResponseWithHttpURLConnection(ICatService.HOST + ICatService.PATH);
```

```
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                // 回到主线程更新UI
                tvOut.setText(s);
            }
        });
    }
}.start();
```

方法二：使用AsyncTask

```
new AsyncTask<Object, Integer, String>() {
    @Override
    protected String doInBackground(Object... objects) {
        // 在后台线程执行请求
        return NetworkUtils.getResponseWithHttpURLConnection(ICatService.HOST + ICat
Service.PATH);
    }

    @Override
    protected void onPostExecute(String s) {
        // 回到主线程更新UI
        tvOut.setText(s);
    }
}.execute();
```

方法三：使用Retrofit

```
Call<List<Cat>> call = getCatService().randomCat(1);
// enqueue进到后台线程队列处理
call.enqueue(new Callback<List<Cat>>() {
    @Override
    public void onResponse(Call<List<Cat>> call, Response<List<Cat>> response) {
        // callback会回到主线程
        if (response.isSuccessful() && response.body() != null) {
            List<Cat> cats = response.body();
            Cat cat = cats.get(0);
```

```
            tvOut.setText(cat.getUrl());
        }
    }

    @Override
    public void onFailure(Call<List<Cat>> call, Throwable t) {
        // callback会回到主线程
        Toast.makeText(MainActivity.this, "retrofit: " + t.getMessage(), Toast.LENGTH_SHORT).show();
    }
});
```

AsyncTask?

1. 如何使用AsyncTask

先看看使用AsyncTask

2. AsyncTask原理

看看AsyncTask文档，阅读官方文档是最好的学习方法（英语很重要）。

Gson原理?

先看一下gson的使用

1. 定义Model类

```
public class Cat {
    @SerializedName("id") public String id;
    @SerializedName("url") public String url;
}
```

2. 使用Gson解析json字符串

```
String CAT_JSON = "{\"id\":\"293\",\"url\":\"https://cdn2.thecatapi.com/images/293.jpg\"}";
Cat cat = getGson().fromJson(CAT_JSON, Cat.class);
```

Gson做了什么?

1. 在使用gson的fromJson方法时，我们会传入一个Cat.class，这个class对象包含Cat这个类的描述信息，比如Cat的成员变量id，它有个注解SerializedName，值是"id"。
2. gson会在内部new一个cat对象，然后根据class里的描述信息，找到SerializedName定义的字段名"id"，解析json里的"id"字段，赋值给cat对象。
3. 等到gson返回cat对象的时候，已经从json里把能找到的字段都赋值上去了，没有找到的字段会初始化为默认值（例如int型默认0，String类型默认null）。

Retrofit原理?

先看一下Retrofit的使用

1. 创建retrofit

```
String BASE_URL = "http://test.androidcamp.bytedance.com/mini_douyin/invoke/";
Retrofit retrofit = new Retrofit.Builder()
            .baseUrl(BASE_URL)
            .addConverterFactory(GsonConverterFactory.create())
            .build();
```

2. 定义IMiniDouyinService接口

```
public interface IMiniDouyinService {
    @Multipart
    @POST("video")
    Call<PostVideoResponse> postVideo(
            @Query("student_id") String studentId,
            @Query("user_name") String userName,
            @Part MultipartBody.Part image, @Part MultipartBody.Part video);

    @GET("video")
    Call<GetVideosResponse> getVideos();
}
```

3. 使用retrofit创建IMiniDouyinService的实例

```
IMiniDouyinService miniDouyinService = retrofit.create(IMiniDouyinService.class);
```

4. 调用miniDouyinService的方法，获取call对象，使用call对象执行网络请求（同步用execute，异步用enqueue）

```
Call<GetVideosResponse> call = miniDouyinService.getVideos();
```

```
call.enqueue(new Callback<GetVideosResponse>() {
    @Override
    public void onResponse(Call<GetVideosResponse> call, Response<GetVideosResponse> response) {
        if (response.body() != null && response.body().getVideos() != null) {
            mVideos = response.body().getVideos();
            mRv.getAdapter().notifyDataSetChanged();
        }
        mBtnRefresh.setText(R.string.refresh_feed);
        mBtnRefresh.setEnabled(true);
    }

    @Override
    public void onFailure(Call<GetVideosResponse> call, Throwable throwable) {
        mBtnRefresh.setText(R.string.refresh_feed);
        mBtnRefresh.setEnabled(true);
        Toast.makeText(MainActivity.this, throwable.getMessage(), Toast.LENGTH_SHORT).show();
    }
});
```

Retrofit做了什么?

1. 定义接口IMiniDouyinService的时候，我们声明了api的方法，这个方法满足一定的规律：
   - 在方法上加了注解@GET或者@POST表示http请求的method
   - @GET或者@POST注解里的值"video"表示请求的时候接在BASE_URL后面的path
   - 方法的返回值是一个Call<T>，为了在调用以后能获取到一个call对象
   - Call<T>里的泛型T定义了这个请求的response类型
   - 方法的参数也加了各种注解如@Query，@Part等，表示这个参数的值在请求时属于哪一部分
2. 调用retrofit.create方法并传入IMiniDouyinService.class时，retrofit会解析我们定义在IMiniDouyinService这个类里的方法和方法的参数，以及各个地方的注解（类似Gson里解析Cat.class），通过这些信息，retrofit能知道我们每个请求是什么http method，完整url是什么，需要哪些参数表，response的类型是什么。
3. 当我们调用miniDouyinService.getVideos()方法时，实际上retrofit内部帮我们完成了一些操作：
   - url和query参数的拼接，例如CatApiService里提到的@Query("limit") int limit参数，当我们传入1作为limit的值时，retrofit会帮我们拼接出类似这样的url => "http://cat.com/image/search?limit=1"

    - post body的组装，例如postVideo方法里定义的@Part MultipartBody.Part image，当我们把本地的图片文件转成MultipartBody.Part类型传入时，retrofit帮我们把封面图这个part装进了post body。（这里涉及到一种POST请求的类型，@Multipart，大家可以自行Google了解一下）
    - retrofit完成各种请求url/参数/body的组装后，返回给我们一个call对象，让我们可以自行决定调用的时机和调用的方式（execute或者enqueue）
4. 当我们调用call的execute或者enqueue方法时，才真正把请求发出去，接收到服务端的返回值以后，自动帮我们转成GetVideosResponse类型。这部分是Gson做的，如何关联retrofit和gson呢？答案是构造retrofit时的addConverterFactory()方法，我们传入了一个Gson的转换器，GsonConverterFactory.create()

总结一下Retrofit的优点

1. 自动拼接url和query
2. 自动组装请求的body
3. 自动解析response的数据
4. 灵活切换线程

发布视频崩溃？

因为发布视频会读取相册里的视频和图片，需要授权读写手机存储。
设置=>应用权限管理=>Dou=>读写手机存储权限，开启

Android Studio环境问题？

问Google，问Baidu，问Stackoverflow，老师真的帮不上忙[捂脸]。

模拟器/手机联网问题？

1. 看一下是否有代理，vpn之类的，需要关闭代理才能访问https的资源。
2. 把资源链接的https前缀改成http试试