链滴

Android OkHttp + Retrofit 断点续传

作者：RustFisher

原文链接：https://ld246.com/article/1571463215650

来源网站：链滴

本文链接

前面我们已经知道如何使用OkHttp+Retrofit下载文件。
下载文件时，可能会遇到一些意外情况，比如网络错误或是用户暂停了下载。
再次启动下载，如果又要从头开始，会白白浪费前面下载好的内容。
断点续传功能可以从上次停止的地方继续下载文件。

http范围请求

Range 是一个请求首部，告知服务器返回文件的哪一部分。
在一个 Range 首部中，可以一次性请求多个部分，服务器会以 multipart 文件的形式将其返回。
如果服务器返回的是范围响应，需要使用 206 Partial Content 状态码。
假如所请求的范围不合法，那么服务器会返回 416 Range Not Satisfiable 状态码，表示客户端错误。
服务器允许忽略Range首部，从而返回整个文件，状态码用200。

示例

```
Range: <unit>=<range-start>-
Range: <unit>=<range-start>-<range-end>
Range: <unit>=<range-start>-<range-end>, <range-start>-<range-end>
Range: <unit>=<range-start>-<range-end>, <range-start>-<range-end>, <range-start>-<r
nge-end>
```

发起请求时，一般Range的内容写成 bytes=0-100 这样的形式。
或者请求多个部分时，指定多个范围。

```
Range: bytes=200-1000, 2000-6576, 19000-
```

Content-Range 表示主体长度或者尺寸。

参考：

https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/Range

使用示例

代码可以参考

https://github.com/RustFisher/android-Basic4/tree/master/appdowloadsample

使用OkHttp添加Range头部，告知服务器我们需要的文件数据范围。

定义的方法中要求传入 @Header("Range")

```
    private interface ApiService {
        @Streaming
        @GET
        Observable<ResponseBody> downloadPartial(@Url String url, @Header("Range") String
ange);
```

```
    }
```

需要传入的Range字符串形如 bytes=200-1000

```
retrofit.create(ApiService.class)
    .downloadPartial(callBack.getUrl(), "bytes=" + startByte + "-")
    .subscribeOn(Schedulers.newThread())
    .observeOn(Schedulers.io())
    .doOnNext(new Consumer<ResponseBody>() {
        @Override
        public void accept(ResponseBody responseBody) throws Exception {
            callBack.saveFile(responseBody);
        }
    })
    .doOnError(new Consumer<Throwable>() {
        @Override
        public void accept(Throwable throwable) throws Exception {
            tellDownloadError(callBack.getUrl(), throwable);
        }
    })
    .observeOn(AndroidSchedulers.mainThread())
    .subscribe(new Observer<ResponseBody>() {
        @Override
        public void onSubscribe(Disposable d) {

        }

        @Override
        public void onNext(ResponseBody responseBody) {

        }

        @Override
        public void onError(Throwable e) {
            callBack.setState(DownloadTaskState.ERROR);
            tellDownloadError(callBack.getUrl(), e);
        }

        @Override
        public void onComplete() {

        }
    });
```

我们也可以在下载前，先去检查文件已下载的部分的大小，再决定Range范围。

续传时，写入本地文件注意选择流的append模式。

```
fos = new FileOutputStream(file, true);
```

更多请参考：

Android OkHttp + Retrofit 使用示例

Android OkHttp + Retrofit 取消请求的方法

Android OkHttp + Retrofit 下载文件与进度监听

Android OkHttp + Retrofit 断点续传