断点续传下载实现 问题记录

1. OkHttp：请求步骤：
   1. 创建OkHttpClient 对象
   2. 创建Request对象

new Request.Builder().url(mDownloadInfo.getUrl()).header("Range", "bytes=" + mTempFile.length() + "-").build()

* 1. 得到Call 对象，call = client.newCall(request);
  2. 得到Response对象：response = call.execute();
  3. 判断影响是否成功：isSucessful = response.isSuccessful();
  4. 判断服务端是否支持断点续传功能，并获取下载文件总长度
* 拿到”Content-Range” header域的内容：stringContentRange=response.headers().get(“Content-Range”);
* 判断响应中是否包含stringContentRange且包含请求时添加的“ Range:bytes=length”中的length内容。

if (!TextUtils.*isEmpty*(responseRange) && responseRange.contains(String.*valueOf*(mTempFile.length()))) {  
 outputStream = new FileOutputStream(mTempFile, true); //为true 则追加数据，false 则从头写入  
} else {  
 outputStream = new FileOutputStream(mTempFile, false);  
}

* 获取当前响应response中的实体长度（即为下载文件剩余总长度）contentLength = response.body().contentLength;
* 设置文件总长度：如果原来已下载过则总长度为原长度+contentLength，如果没下载过则总长度就等于contentLength
  1. 获取响应流 inputStream = response.body().byteStream();
  2. 写入文件并通知进度更新：

//将文件写入本地  
byte[] buffer = new byte[1024 \* 8];  
int length;  
boolean isInterrupt = false;  
while ((length = inputStream.read(buffer)) != -1) {  
 if (Thread.*currentThread*().isInterrupted()) {  
 isInterrupt = true;  
 break;  
 }  
 outputStream.write(buffer, 0, length);  
 //更新DownloadInfo 的已下载长度值  
 if (mListener != null) {  
 mDownloadInfo.addDownloadedLength(length);  
 mListener.onProgress(mDownloadInfo);  
 }  
}  
if (outputStream != null) {  
 outputStream.flush();  
 outputStream.close();  
}  
if (inputStream != null) {  
 inputStream.close();  
}  
//暂停下载，结束执行  
if (isInterrupt) {  
 if (mListener != null) {  
 mListener.onPause(mDownloadInfo);  
 }  
 return;  
}  
//下载完成  
mListener.onFinish(mDownloadInfo);  
mTempFile.renameTo(new File(mDownloadInfo.getSavePath(), mDownloadInfo.getName()));

这里实现暂停功能是通过调用future.cancel(true);方法来实现的，它是通过向线程发送Interrupt来实现的。因此在下载文件过程中要对线程的中断标志位进行实时监测（isInterrupt = Thread.currentThread().isInterruputed()），一旦发现isInterrupt为true，即发生了中断（暂停下载），则要跳出写文件循环，对相应流进行刷新、关闭操作（很重要，如果不能保证每次中断都时行刷新、关闭流操作的话，下次继续下载时可能发生异常，及内存对象中下载长度和本地文件中的下载长度不一致的情况）

1. 配置了线程池的 最大核心数为3 （Imageloader的默认线程池大小也是3）、配置了线程工厂（可为新线程设置线程名字、配置unCaughtExceptionHandler及是否为守护线程等，可配置自定义线程，在自定义线程的run方法中设置其线程优先级）

Process.*setThreadPriority*(Process.*THREAD\_PRIORITY\_BACKGROUND*);

1. 如果在子线程中调用adpter.notifyDataChanged()方法无效。
2. 绘制文字时的居中问题，水平居中可通过mPaint.setTextAlign(Paint.Align.*CENTER*)来实现 垂直居中稍有麻烦

要求得文字绘制位置的baseLine位置才行，一般情况下，我们已知的是绘制区域的中心线位置(center)。



我们可以结合FontMetrics 对象来完成 mPaint.setTextSize(mFondSize); mFontMetrics = mPaint.getFontMetrics();

Baseline = center +(fontMetrics.bottom-fontMetrics.top)/2-fontMetrics.bottom

1. 更新RecycleView中单个item项可通过以下方法实现

@Override  
public void onProgress(DownloadInfo info) {  
 int position = mInfoList.indexOf(info);  
 if (position >= 0) {

View view = mListView.getChildAt(position);  
 DownloadViewHolder viewHolder = (DownloadViewHolder) mListView.getChildViewHolder(view);  
 float downloadSize = info.getDownloadedLength();  
 float totalSize = info.getTotalLength() ;  
 viewHolder.pb\_progress.setProgress((int) (downloadSize \* 100 / totalSize));  
 viewHolder.tv\_size.setText(mDecimalFormat.format(downloadSize / 1024 / 1024) + "M /" + mDecimalFormat.format(totalSize / 1024 / 1024) + "M");  
 Log.*e*(*TAG*, "name: " + viewHolder.tv\_name.getText() + "position: " + position);  
 }  
}

1. 对浮点数进行格式化可通过DecimalFormat mDecimalFormat = new DecimalFormat("0.00");来完成