添加权限：

```
    <!-- 访问网络，网络定位需要上网 -->
    <uses-permission android:name="android.permission.INTERNET" />

    <!-- 如果将下载的文件保存到外部存储器，还需要配置外部存储器的读写权限 -->
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

```

    public void download(String downloadUrl) {
        // 创建下载请求
        DownloadManager.Request request = new DownloadManager.Request(Uri.parse(dow
       /*
        * 设置在通知栏是否显示下载通知(下载进度)，有 3 个值可选:
        *    VISIBILITY_VISIBLE:                   下载过程中可见，下载完后自动消失 (
        *    VISIBILITY_VISIBLE_NOTIFY_COMPLETED:  下载过程中和下载完成后均可见
        *    VISIBILITY_HIDDEN:                    始终不显示通知
        */
        request.setNotificationVisibility(DownloadManager.Request.VISIBILITY_VISIBI
        // 设置通知的标题和描述
        request.setTitle("通知标题XXX");
        request.setDescription("对于该请求文件的描述");
        /*
        * 设置允许使用的网络类型，可选值:
        *     NETWORK_MOBILE:      移动网络
        *     NETWORK_WIFI:        WIFI网络
        *     NETWORK_BLUETOOTH:   蓝牙网络
        * 默认为所有网络都允许
        */
        request.setAllowedNetworkTypes(DownloadManager.Request.NETWORK_WIFI);

        // 添加请求头
        // request.addRequestHeader("User-Agent", "Chrome Mozilla/5.0");

        // 设置下载文件的保存位置
        File saveFile = new File(Environment.getExternalStorageDirectory(), item.ge
        request.setDestinationUri(Uri.fromFile(saveFile));

        /*
        * 2. 获取下载管理器服务的实例，添加下载任务
        */
        DownloadManager manager = (DownloadManager) getApplicationContext().getSyst

        // 将下载请求加入下载队列，返回一个下载ID
        long downloadId = manager.enqueue(request);
        // 如果中途想取消下载，可以调用remove方法，根据返回的下载ID取消下载，取消下载后
        // manager.remove(downloadId);
    }

```

```java
    private void queryDownload(long downloadId) {
        // 获取下载管理器服务的实例
        DownloadManager manager = (DownloadManager) getApplicationContext().getSyst

        // 创建一个查询对象
        DownloadManager.Query query = new DownloadManager.Query();

        // 根据 下载ID 过滤结果
        query.setFilterById(downloadId);

        // 还可以根据状态过滤结果
        // query.setFilterByStatus(DownloadManager.STATUS_SUCCESSFUL);

        // 执行查询，返回一个 Cursor (相当于查询数据库)
        Cursor cursor = manager.query(query);

        if (!cursor.moveToFirst()) {
            cursor.close();
            return;
        }

        // 下载ID
        long id = cursor.getLong(cursor.getColumnIndex(DownloadManager.COLUMN_ID));
        // 下载请求的状态
        int status = cursor.getInt(cursor.getColumnIndex(DownloadManager.COLUMN_STA
        // 下载文件在本地保存的路径
        String localFilename = cursor.getString(cursor.getColumnIndex(DownloadManag
        // 已下载的字节大小
        long downloadedSoFar = cursor.getLong(cursor.getColumnIndex(DownloadManager
        // 下载文件的总字节大小
        long totalSize = cursor.getLong(cursor.getColumnIndex(DownloadManager.COLUM

        cursor.close();

        System.out.println("下载进度: " + downloadedSoFar + "/" + totalSize);

        /*
                 * 判断是否下载成功，其中状态 status 的值有 5 种:
                 *     DownloadManager.STATUS_SUCCESSFUL:    下载成功
                    *     DownloadManager.STATUS_FAILED:        下载失败
                 *     DownloadManager.STATUS_PENDING:       等待下载
                    *     DownloadManager.STATUS_RUNNING:       正在下载
                *     DownloadManager.STATUS_PAUSED:        下载暂停
                */
        if (status == DownloadManager.STATUS_SUCCESSFUL) {
            /*
            * 特别注意: 查询获取到的 localFilename 才是下载文件真正的保存路径，在创建
            * 请求时设置的保存路径不一定是最终的保存路径，因为当设置的路径已是存在的文件
            * 下载器会自动重命名保存路径，例如: .../demo-1.apk, .../demo-2.apk
            */
            System.out.println("下载成功，打开文件，文件路径: " + localFilename);
        }
    }
```