Android 平台下快速加载图文信息的研究与实现①

王 结, 魏振钢

(中国海洋大学 信息科学与工程学院学院, 青岛 266100)

摘 要: 为了加快 Android 移动端图文信息的加载速度, 方便用户快速浏览界面获取相关信息, 在汲取前人的相关技术基础上, 结合了 Android 图像开源视图 smart-Image-View、网络请求框架 android-async-http 以及分页显示技术. 采用大量图文数据做实验, 结果表明三种方法的结合使用明显提高了图文加载速度.

关键词: 图文信息; 加载速度; 图像开源视图; 网络请求框架; 分页显示

Research and Implementation of Rapid Loading Picture and Text Information Based on Android

WANG Jie, WEI Zhen-Gang

(College of Information Science and Engineering, Ocean University of China, Qingdao 266100, China)

Abstract: This paper proposes the issue that we should speed up the loading rate of image and text information on the Android mobile terminal, and meanwhile make users convenient to fast access to relevant information by browsing interface. On the basis of the relevant technology, this paper combines the Android image open source view with smart-Image-View, Network request framework, android-async-http and paging display technology. After using a large number of textual and graphic data to do the experiment, the result shows that the combination use of the three methods made it success to improve the loading rate of image and text information.

Key words: image and text information; loading rate; image open source view; network request framework; paging display

1 引言

随着Android平台的应用不断涌现, 为了提高App内容的简洁直观性, 开发者偏向以图文信息的形式来布局界面内容. 例如目前广泛受到用户关注的购物App, QQ、微信等聊天软件以及各种题库软件等都少不了图文内容的展示. 然而由于数据库内容的不断增多, 在 Android 端前台界面要显示的图文信息也愈发增多. 因此, 寻求新的技术方法来加快图文信息的加载速度, 增强用户体验性是非常有必要的.

2 Android图像开源视图

Android 平台的应用开发主要采用 MVC 框架[1], 通过控制层、模型层以及视图层之间的通信, 完成对图文信息的加载, 具体过程是 Android 端前台的 Activity 将用户的请求信息交于模型业务逻辑层(即后台相关的Servlet)处理, 之后Servlet从服务器中获取相关的图文信息以 Json 数据包[2]形式返回给前台, 再由前台视图层(即相应的 XML)处理显示. 具体的通信过程如图 1 所示.

而图片在服务器中的存储格式一般是字符串形式的 URL, 当前台从 Json 数据包中解析出各种数据后, 文字可以立即显示出来, 但由于获取到的图片是其URL形式存储的, 需通过URL再次访问后台来获取图片资源. 这样不仅使得代码量繁杂同时也大大降低了图片的加载速度.

为了解决图片和文字加载速度的不同步问题, 查阅了相关的文献期刊, 先后有 AsyncHttpClient 框架[3]、用于获取网络图片的Universal-Image-Loader框架以及 Volley 网络通信框架被推出应用到网络图片加载中, 表 1 所示为这三种网络通信框架[4]的特点比较.

① 收稿时间:2016-03-18;收到修改稿时间:2016-04-24 [doi: 10.15888/j.cnki.csa.005466]

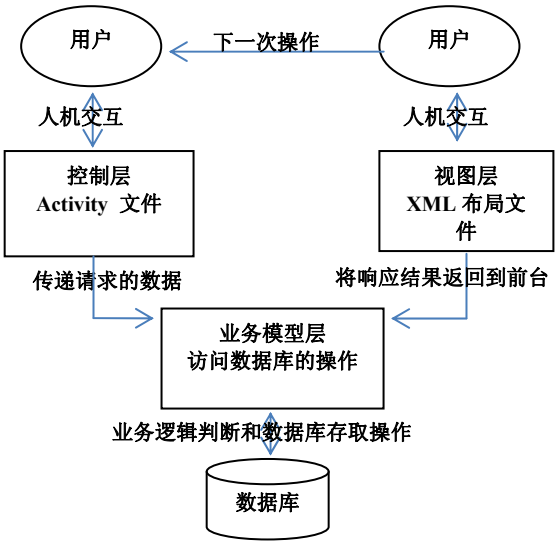

图 1　Android 端 MVC 通信过程

表 1　三种网络通信框架的特点比较

	AsyncHttpClient	Universal-Image-Loader	Volley
基本功能	适用于一般性的网络请求	可用于频繁的网络图片请求	可用于频繁的网络请求
是否提供缓存功能	无	提供图片请求的缓存功能	提供网络请求的缓存功能
是否能够多线程	不提供多线程功能	提供多线程功能且可自行扩展管理接口	提供多线程功能且可自行扩展管理接口
运行稳定性	网络一旦错误需开发者自行处理，稳定性低	图片请求一旦失败无法重试或者放弃，稳定性高	网络请求有重试和放弃策略，稳定性极高

通过分析比较可以看出这三种网络框架都有各自的优缺点，应用的比较好的是 Volley 框架，可以实现多线程请求，稳定性及高，同时还提供缓存功能. 但在只加载图文信息的具体项目中，若采用这种框架实现，空间占用太大，且代码存在冗余，反而使得程序更加复杂化. 而本文采用的框架是集多线程、缓存机制、代码量少于一体的 Android 图像视图控件 smart-Image-View[5].

smart-Image-View 可以实现各种来源的图片资源获取. 根据来源的不同，smart-Image-View 定义了一个公共接口 SmartImage，用不同的类分别来实现该接口下的一个 getBitmap 函数. 如图 2 所示是 smart-Image-View 的基本框架.

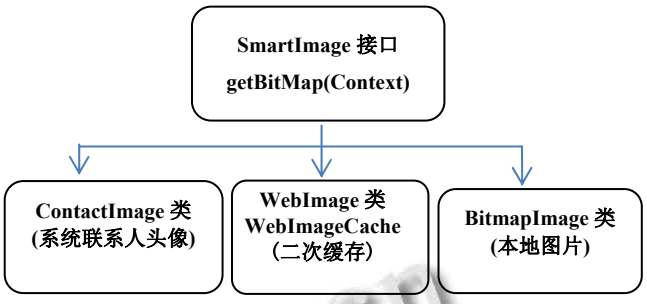

图 2　smart-Image-View 基本框架

将 smart-Image-View 控件取代 ImageView 应用到图片加载中，只需调用自定义控件 SmartImageView 的 setImageUrl 方法，即可从指定的 URL 路径中读取图片资源并部署在该控件上.

同时，smart-Image-View 通过 WebImage 类下的 WebImage Cache 实现了图片的缓存. WebImage 类根据 URL 获取资源时，并不是每次都要从网络上加载，而是实现了二级缓存，即内存缓存和磁盘缓存. 每次加载时，先要判断该 URL 对应的图片是否已存在于缓存中，若有，直接从缓存中获取，反之，再从 URL 上加载. smart-Image-View 在图文内容加载上的应用使得图片资源和文字实现了几乎同步显示的效果，并且节约了用户的下载流量和载入时间.

3　Android网络请求框架

Http 网络数据交互请求是 Android 端应用程序开发关键部分之一，网络请求方式的效率和性能直接影响到 App 的整体用户体验流畅性. 以往 Android 端的网络请求处理一般使用 Apache HTTP Client 或者 HttpURLConnect 这两个类库，同时需要编写大量的 get 和 post 方法完成请求，并且网络请求都会占用 UI 线程，使得其他操作无法进行，遇到网络异常情况还会出现程序无响应现象. 对于这些问题，开发者们提出过很多的优化方案，谭东等人在《Android 网络负载请求优化方案设计》[6]一文中提出了几个网络优化方法，创建网络请求线程管理池，使用优先级请求排序策略以及网络请求及时回收等，都能很好地达到优化效果. 而本文提出的 android-async-http[7]开源框架则是集成了这些优化思想，实现了一个异步网络请求处理库，独立在 UI 主线程之外，通过线程池的网络请求

回调方法处理请求结果.

将 android-async-http 应用到图文信息加载上的优势有几点: 第一, 移动端应用要考虑到网络移动连接的不稳定性, App 在加载图文信息的过程中网络一旦断开, 信息将无法显示出来, 而 android-async-http 框架实现了网络的自动智能请求重试, 在请求失败的情况下可以设置多次请求; 第二, App 端向服务器请求获取相关图文信息时, 利用后台 Servlet 将数据打包成 Json 格式返回给前台后, 一般都需要进行相当费时的一系列格式转换处理才能显示给用户浏览. 而 android-async-http 框架内置了一个 JsonHttpResponseHandler 解析类, 可以实现自动化高效地 Json 格式数据解析, 大大缩短了图文数据从返回前台到显示于界面这一时间过程. 第三, android-async-http 框架具有持久化 cookie 存储的特点, 在请求服务器时能够将访问到的 cookie 信息方便的存储起来.

4 分页显示

随着数据库技术在网络领域的广泛应用, Android 端与数据库交互信息的显示方式也在不断地创新和改进. 传统的方法是 Android 端一次性向后台 Servlet 请求所有相关图文信息, Servlet 得到命令后从数据库中获取所有符合条件的图文信息并打包成 Json 数据发送给前台. 这种方法显然在数据信息较大的情况下使用会明显降低加载速度; 之后有了分页处理[8]的思想, 即利用结果集处理. 通过首次执行 SQL 语句, 将查询结果存于结果集对象中, 之后前台的每次请求数据, 实际上是在结果集中获取指定行位置的数据, 这一过程虽然减少了对数据库 SQL 语句的查询, 但数据库仍要一直保持连接状态, 使得占用的数据库访问资源比较多, 而利用率不高. 而针对 Android 端请求的图文信息量大的情况, 本文提出了分页显示技术, 即利用 SQL 语句处理, 在前台界面加载时, 只向后台数据库请求固定行数的图文信息, 存于结果集, 并显示在列表视图中, 等到需要查看更多时, 再向数据库请求加载. 这样虽然对数据库进行 SQL 语句查询操作变得频繁, 但数据库的访问资源使用完毕之后就立即被释放.

首先在 Android 端和后台模型层分别创建一个用于传递分页信息的数据包 PageBean 类, 该类包含了分页的具体信息(如数据总行数, Android 端当前显示页号, 总显示页数, 每页显示行数)和分页数据(每页中显示的 List 数据集), 前台向后台模型层传递每页显示的行数和当前显示页号, 后台相应 Servlet 通过 doPost 方法接收前台传递来的数据参数后, 调用实现 DAO 接口的具体类的方法访问数据库, 获取相关信息打包存于 PageBean 类的对象中, 以 Json 包的形式返回给前台. 该过程的具体实现如图 3 所示.

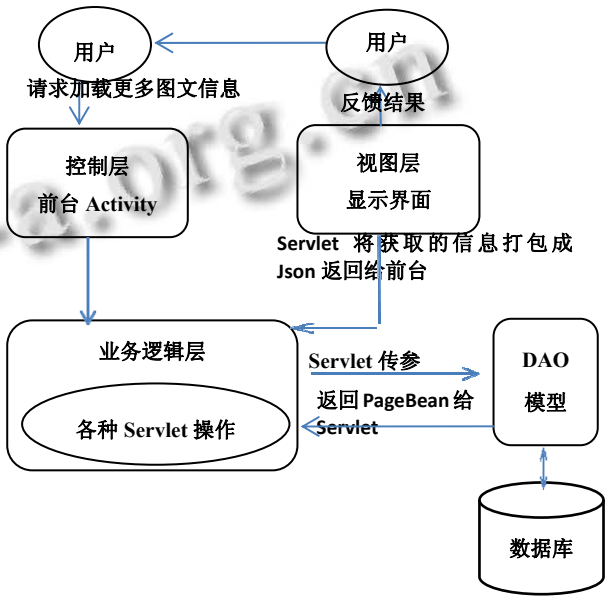

图 3 Android 端分页显示具体实现过程

该过程中使用了数据访问模型[9,10](Data Access Object), 它是处于业务逻辑层与数据源之间的一种抽象数据源, 通过 DAO 层的抽象, 将具体的业务逻辑层和数据源区分开, 其实质是向外部提供了一个访问数据源的统一接口, 对外隐藏操作数据源的实现细节, 从而提高软件的可维护性.

5 具体实现

在前台界面 XML 文件中引用了 smart-Image-View 图像开源视图作为图片的显示容器. 实验结果显示前台读取商品信息时, 文字和图片几乎是同时加载出来的, 说明图片的加载速度有了明显的提高. 图 4 和图 5 所示为用 ImageView 视图容器和 smart-Image-View 视图容器加载图片的效果显示. 从中可以明显看出使用 smart-Image-View 的优势.

而在 App 端应用 android-async-http 网络请求框架也是比较简便的, 首先需要将该框架的 jar 包添加到 Android 应用程序 libs 文件中, 再创建一个

AsyncHttpClient 对象，并通过 RequestParams 对象设置请求参数，然后调用 Async -HttpClient 的某个 get 或 post 方法，传递你需要的 callback 接口实现 AsyncHttpResponseHandler. 实现代码如图 6 所示.

图 4　ImageView 视图容器加载的效果

图 5　smart-Image-View 视图容器加载的效果

```
RequestParams params = new RequestParams(); // 绑定参数
params.put("page", page);
params.put("pagesize", pagesize);
params.put("listway", listway);
HttpUtil.get(ConnNet.URLVAR + url, params,new AsyncHttpResponseHandler() {
        pageBean bean = new pageBean();
        @Override
        public void onSuccess(int arg0, Header[] arg1, byte[] arg2) {// 数据库连接成功
            String result = new String(arg2);
            Gson gson = new Gson();
            bean = gson.fromJson(result,new TypeToken<pageBean>() {}.getType());
            Toast.makeText(getApplicationContext(),"服务器连接成功", Toast.LENGTH_LONG).show();
        }
        @Override
        public void onFailure(int arg0, Header[] arg1, byte[] arg2,
                Throwable arg3) {// 数据库连接失败
            Toast.makeText(getApplicationContext(),"服务器连接超时" , Toast.LENGTH_LONG).show();
        }
    });
}
```

图 6　android-async-http 框架的具体代码实现

代码中 HttpUtil 是一个自定义的网络请求工具类，它将 AsyncHttpClient 对象和各种含不同参数的 get 和 post 方法封装在一起. 这样实现的优点是可以避免网络请求处理时 AsyncHttpClient 对象的反复创建.

Web 端的分页处理技术应用到 Android 移动端上，首先要对前台用于显示图文信息的 ListView 或 GridView 设置滚动监听事件，当用户滑动到已加载的最后一条信息时，将当前显示页号和每页显示总行数传递到后台 Servlet，通过 DAO 方法访问数据库加载指定数量的图文内容. 监听滚动事件的核心代码如图 7 所示.

```
// 当list开始滚动时
public void onScroll(AbsListView view, int firstVisibleItem,
        int visibleItemCount, int totalItemCount) {
    final int totalCount = firstVisibleItem + visibleItemCount;// 最下面的条目数
    int currentPage;
    if ((totalCount % pageSize) != 0)
        currentPage = (totalCount / pageSize) + 1;// 当前页
    else  currentPage = totalCount / pageSize;
    final int nextPage = currentPage + 1;// 下一页
    if (totalCount == totalItemCount && nextPage <= countPage && finish) {
        finish = false;// 已经移动到了listview的最后
        listview.addFooterView(footer); // 添加页脚
        new Thread() {
            public void run() {
                Operaton operaton = new Operaton();
                pageBean result = operaton.getJson(MainActivity.userid+ "",
                        MainActivity.flag + ""+"/ShowOrderServlet", nextPage + ""
                , pageSize+ "", tag, start, end, supselect.getText().toString());
                Message msg = new Message();
                msg.what = SUCCESS_GET_DATA;
                msg.obj = result;
                mHandler.sendMessage(msg);  };
        }.start();
    }
}
```

图 7　前台 ListView 或 GridView 的滚动事件的监听

而访问数据库加载更多图文信息是通过 SQL 语句查询来实现的，由于实现的效果是分页显示，SQL 语句在书写时，要传入当前行号和指定获取的行数等参数，具体代码如图 8 所示.

```
// 根据所在页数，获取该页数据
@SuppressWarnings({ "unchecked", "rawtypes" })
public pageBean getGoods(int page,int pagesize) {
    @SuppressWarnings("rawtypes")
    pageBean bean = new pageBean();
    bean.setTotalrows(getGoodRows());// 设置数据总行数
    bean.setPagesizes(pagesize);// 设置每页显示15行
    bean.setCurpage(page);// 设置当前页数
    int offset = (bean.getCurpage() - 1) * bean.getPagesizes();// 起始数据的行数
    int rows = bean.getPagesizes();// 要显示的记录数
    SessionFactory sessionFactory = HibUtil.getSessionFactory();
    Session session = sessionFactory.openSession();
    String hql ="from Goods";
    Query query = session.createQuery(hql);
    query.setFirstResult(offset);// 设置起始行
    query.setMaxResults(rows);// 每页条数
    List<Goods> list = query.list();
    List aryList = new ArrayList();
    bean.setData(list);
    session.close();
    return bean;
  }
}
```

图 8　后台 SQL 语句查询实现的 PageBean 数据获取代码

6　结语

为了提高 Android 端图文信息的加载速度，本文是从三个方面考虑的：一是从信息本身特点出发；二是从网络请求处理方式上考虑；三是从数据库访问方式上出发. Android 开源图像视图 smart-Image-View 和

开源框架 android-async-http 的设计是广大 Android 编程爱好者不断修改创新的成果, 而分页显示技术则是在借鉴 Web 端分页处理数据的基础上应用到 Android 移动端的. 这三种方法结合使用到加载图文信息上可以为其他 App 端项目开发提供参考价值.

参考文献

1 任中方,张华,闫明松,陈世福.MVC 模式研究的综述.计算机应用研究,2004,21(10):1–4,8

2 高静,段会川.JSON 数据传输效率研究.计算机工程与设计,2011,32(7):2267–2270.

3 Zhao Z, Song JD, Haihong E. Research and design for mobile application development on Android platform. The Journal of New Industrialization, 2013, 3(6): 78–89.

4 孟远.Android 网络通信框架 Volley 的解析和比较.软件,2014,(12):66–68.

5 Smith J. Android Smart Image View. http://loopj.com/android- smart-image-view/.

6 谭东,杨德刚.Android 网络负载请求优化方案设计.中国新通信,2015,(2):107–108.

7 Smith J. Android Asynchronous HttpClient. http://loopj.com/android-async-http/.

8 卢成均. ASP．NET 中数据列表分页方法研究.计算机系统应用,2006,15(11):83–86.

9 王正玉,李斌.基于 DAO 模式的 Hibernate 框架在 Java Web 开发中的应用.微型机与应用,2015,(11):14–17.

10 欧阳宏基,解争龙,黄素萍,丁要军.一种基于 DAO 设计模式与 Hibernate 框架的数据持久化层模型.微计算机应用,2009,30(3):36–40.