ОСОБЕННОСТИ ОБРАБОТКИ, ХРАНЕНИЯ И ОТОБРАЖЕНИЯ ВЕБ-СОДЕРЖИМОГО В МОБИЛЬНЫХ ПРИЛОЖЕНИЯХ НА ПЛАТФОРМЕ ANDROID

Дектярев К. В.
Кафедра информационно-вычислительных систем, Поволжский государственный технологический университет
Йошкар-Ола, Российская Федерация
E-mail: dektyarev.k@gmail.com

Довольно часто данные, отдаваемые популярными онлайн-сервисами не являются оптимизированными для мобильных устройств. Хорошо, если есть возможность получать адаптированные данные от веб-сервиса, однако такая возможность присутствует далеко не всегда. В таком случае приходится решать данную проблему на стороне клиента, который может быть существенно ограничен в ресурсах системы.

Введение

С каждым днем все больше людей становится активными пользователями различных смартфонов, планшетных компьютеров и других мобильных вычислительных устройств. Наиболее популярными из них являются устройства на базе ОС Google Android, Apple iOS и RIM BlackBerry OS. По данным агентства Millenial Media[1] доля ОС Android в данном сегменте по итогам 2011 года составляет около 47% и прогнозируется дальнейший рост в будущем. Из этого определенно следует, что данная платформа является одной из наиболее перспективных для разработки мобильных приложений, существенную долю из которых составляют клиентские приложения для различных онлайн-сервисов. В работе рассмотрено клиентское приложения для онлайн-сервиса Google Reader.

I. Клиентское приложения для онлайн-сервиса

В качестве примера будет рассмотрено клиентское приложения для онлайн-сервиса Google Reader. Данный сервис является популярным RSS-агрегатором. Помимо этого он предоставляет дополнительные возможности, такие как просмотр наиболее популярных новостей по версии Google в интернете. К данному приложению были предъявлены следующие основные требования:

В результате было спроектировано и разработано приложение под названием Webby.

- Удобный механизм навигации между основными категориями, такими как: закладки, популярные новости, популярные источники;
- Быстрый доступ к наиболее часто используемым функциям;
- Получение наиболее популярных новостей и рекомендаций для каждого пользователя в отдельности;
- Возможность чтения содержимого лент новостей оффлайн;
- Постраничная загрузка записей новостных лент;
- Дополнительная возможность просматривать изображения новости в виде галереи и на полный экран с поддержкой функции multi-touch;
- Совместимость с версиями ОС Android 2.1 и выше;

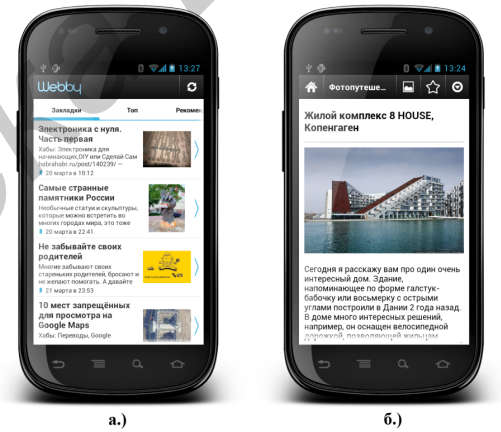

Рис. 1 – Клиентское приложение Webby: а.) главный экран, категория "закладки"; б.) экран чтения новостей.

II. Разработка приложения

Для визуализации списка новостей используются стандартные компоненты фреймворка Android, кроме компонента отображения миниатюр изображений (Рис. 1, а), функционал которого был расширен для возможности загрузки и отображения изображений из интернета.

Для отображения веб-содержимого новости (Рис. 1, б) используется стандартный компонент библиотеки android.webkit.WebView[2], который базируется на фреймворке WebKit, активно развиваемом в том числе и корпорациями Apple и Google[3]. В стандартной реализации он позволяет отображать html содержимое, а также содержит в себе компоненты WebCore и JavaScriptCore для доступа к DOM-модели

содержимого и интерпретации JavaScript кода. Данный компонент также содержит встроенный механизм кэширования изображений.

Задача обеспечения корректного и схожего отображения неадаптированного вебсодержимого на различных устройствах может быть решена путем инъекции в html разметку мета тега viewport с описанием некоторых необходимых свойств отображения:

- target-densitydpi - данное свойство используется для указания специфичной плотности пикселей для отображения страницы. В случае установки значения данного свойства в device-dpi страница не будет отмасштабирована компонентом WebView;
- width - ширина вьюпорта (в пикселях). В случае, если WebView занимает всю ширину экрана, то можно указать значение device-width;
- initial-scale - начальный масштаб страницы, значение устанавливается в 1, чтобы отобразить содержимое в исходном размере;
- maximum-scale - максимальное увеличение содержимого, так же установлено значение 1, чтобы предотвратить его масштабирование;
- user-scalable - свойство, позволяющее пользователю масштабировать содержимое посредством встроенных элементов управления компонента WebView, значение установлено в false, чтобы предотвратить изменение масштаба пользователем;

Поскольку была поставлена задача реализации чтения новостей в режиме оффлайн, то необходимо было иметь возможность управлять политикой кэширования изображений. Данный компонент не позволяет управлять его политикой, и предоставляет лишь возможность управления порядком обращения к кэшу. В итоге было применено следующее решение:

1. Заменить значения атрибутов “src” всех тегов “img” содержимого новости ссылкой на изображение размером 1x1 пиксель, со значением альфа-канала 255 (для 32-битного изображения в формате PNG);
2. Поместить старые значения атрибутов “src” в атрибут “id”, предварительно применив URL-кодирование (либо BASE64);
3. Запустить на выполнение загрузку всех необходимых изображений, посредством собственной реализации менеджера загрузок с последующим кэшированием;
4. Отобразить страницу;
5. По окончанию загрузки каждого изображения посредством выполнения JavaScript кода производить замену аттрибута “src” изображений на локальный путь к кэшированному файлу изображения.

В итоге данный способ загрузки и управления кэшем оказался довольно эффективным и позволяет использовать собственную реализацию кэша изображений.

Еще одной особенностью обработки вебсодержимого является отображение большого количества изображений, поскольку зачастую они не оптимизированы для мобильных устройств. Данная проблема решается в случае использования специализированных компонентов, таких как WebView, однако в остальных случаях визуализация необработанных изображений повлечет большие затраты памяти, что может привести к нестабильной работе приложения, и, вероятнее всего приведет к переполнению "кучи"виртуальной машины Dalvik[4] (VM Heap). Данная проблема должна решаться непосредственно разработчиком приложения. Самым распространенным подходом является применение операции даунсэмплинга (принудительного понижения качества) к изображению. В первую очередь необходимо определить максимальный требуемый размер изображения, который фактически будет отображаться на дисплее.

III. Разработка приложения

Существует большое многообразие устройств на данной платформе, каждое из которых отличается своим техническим исполнением, в том числе разрешением экрана, количеством точек на дюйм (dpi) и размером дисплея. Именно поэтому необходимо рассчитывать максимальный требуемый размер для каждого устройства. Разработчику приложений для ОС Android предлагается абстрагироваться от абсолютных величин измерения отображаемого содержимого (пикселей) и использовать независимые от плотности дисплея пиксели (density-independent pixel, dpi). Для расчета максимального требуемого размера для даунсэмплинга необходимо перевести величину независимых от плотности пикселей в абсолютные пиксели. После чего при помощи стандартного класса android.graphics.BitmapFactory[5] можно получить уменьшенное до необходимых размеров изображение. Для более быстрой работы алгоритма рекомендуется, чтобы максимальный размер изображения являлся равным 2n, где n - положительное целое число.

IV. Список литературы

1. Millenial Media: 2011 Year in Review Mobile Mix™ Report. http://www.millennialmedia.com/downloads/mobile-mix/mobilemix-2011-yir.pdf
2. Android Developers - WebView Class Overview http://developer.android.com/reference/android/webkit/WebView.html
3. Companies and Organizations that have contributed to WebKit http://trac.webkit.org/wiki
4. Виртуальная машина Dalvik / Р. Майер // Android 2. Программирование приложений для планшетных компьютеров и смартфонов. – Москва: ЭКСМО, 2011. - С. 40 – 42.
5. Android Developers - BitmapFactory Class Overview http://developer.android.com