УДК 004.9

ИСПОЛЬЗОВАНИЕ СЕТЕВЫХ ВОЗМОЖНОСТЕЙ ANDROID ДЛЯ РАЗВЛЕЧЕНИЯ И ПОЛЕЗНЫХ ЦЕЛЕЙ. СЕТЕВЫЕ ВОЗМОЖНОСТИ ANDROID

Д. Н. СЫЦЕВИЧ
(Представлено: канд. техн. наук, доц. О. Н. ПЕТРОВИЧ)

В этой статье, основанной на принципах, описанных в статье «Разработка приложений для Android с помощью Android Studio», рассматриваются сетевые возможности Android. Узнайте, как использовать сетевые возможности Android для развлечения и полезных целей. Платформа Android идеальна для разработчиков Java™: они могут использовать имеющиеся навыки для подключения к сети мобильных или «встроенных» платформ.

Android основан на ядре Linux® и содержит множество полезных сетевых возможностей. Если вы не установили Android Studio и не настроили ее для создания приложений, возможно, вы захотите сделать это сейчас, чтобы лучше следовать примеру.

В таблице 1 показаны некоторые сетевые пакеты в Android SDK.

Таблица 1

Сетевые пакеты	Описание
java.net	Предоставляет классы, связанные с сетью, включая сокеты потоков и дейтаграмм, интернет-протокол и общую обработку HTTP. Это универсальный сетевой ресурс. Опытные разработчики Java могут сразу же создавать приложения с помощью этого знакомого пакета
Java.io	Хотя это явно не сеть, это очень важно. Классы в этом пакете используются сокетами и соединениями, предоставляемыми в других пакетах Java. Они также используются для взаимодействия с локальными файлами (частое явление при взаимодействии с сетью)
Java.nio	Содержит классы, представляющие буферы определенных типов данных. Удобно для сетевого взаимодействия между двумя конечными точками на основе языка Java
Org.apache.*	Представляет ряд пакетов, обеспечивающих точное управление и функции для HTTP-коммуникаций. Вы можете узнать Apache как популярный веб-сервер с открытым исходным кодом
Android.net	Содержит дополнительные сокеты доступа к сети помимо основных классов java.net.*. Этот пакет включает класс URI, который часто используется при разработке приложений для Android помимо традиционных сетей
Android.net.http	Содержит классы для управления SSL-сертификатами
Android.net.wifi	Содержит классы для управления всеми аспектами WiFi (беспроводной Ethernet 802.11) на платформе Android. Не все устройства оснащены функцией WiFi, особенно в связи с тем, что Android делает успехи в сегменте «раскладушек» сотовых телефонов от таких производителей, как Motorola и LG
android.telephony.gsm	Содержит классы, необходимые для управления и отправки SMS (текстовых) сообщений. Со временем, скорее всего, будет представлен дополнительный пакет для обеспечения аналогичных функций в сетях, отличных от GSM, таких как CDMA или что-то вроде android.telephony.cdma

Приведенный выше список не является исчерпывающим, но он предназначен для того, чтобы дать вам общее представление о том, на что способна платформа.

Чтобы продемонстрировать, насколько просто подключить Android к сети, в примере показано, как извлечь текст с веб-страницы. Загрузите исходный код примера. Рисунок 1 демонстрирует приложение в действии.

Есть три элемента пользовательского интерфейса:

– EditText позволяет пользователю войти на веб-страницу (https://google.com показан на рисунке 1 и в листинге 2).

– Кнопка используется для указания программе получить текст веб-страницы.

– Как только данные получены, они отображаются в TextView.

Прежде чем углубляться в детали самого кода, важно понять два основных правила, которые должен соблюдать каждый разработчик Android:

1. Никогда не пытайтесь выполнять «длительную» задачу непосредственно в потоке пользовательского интерфейса. Почему бы и нет? Мы не хотим, чтобы медленная сетевая активность, такая как загрузка

большого файла с веб-сервера, ухудшала работу пользователя. Никому не нравится не отвечающее приложение. Эта идея адаптивного пользовательского интерфейса приводит нас ко второму правилу.

2. Только основной поток/поток пользовательского интерфейса приложения может взаимодействовать с элементами пользовательского интерфейса (другими словами, изменять содержимое представления). Это означает, что все модификации пользовательского интерфейса должны выполняться из основного потока приложения.

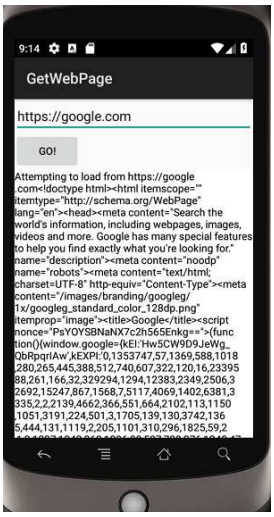

Рисунок 1. – Получение текста с веб-страницы

Если вы можете получать данные только из «фонового» потока и вам не разрешено обновлять пользовательский интерфейс из чего-либо, кроме основного потока, вы попали в затруднительное положение. Как удовлетворить оба требования?

Существует несколько доступных шаблонов проектирования, соответствующих этим двум ограничениям. Некоторые из этих подходов более прямолинейны, чем другие. Легко попасть в ловушку сорняков и идиосинкразических последствий потоков и обработчиков, статических классов, внутренних классов и использования подклассов и WeakReferences для обхода утечек памяти. В то время как некоторым специалистам в области компьютерных наук и пуристам Java могут понравиться подробности этих подходов, мы собираемся использовать класс, предлагаемый Android, чтобы упаковать всю эту драму в простой шаблон проектирования — мы используем AsyncTask.

Класс AsyncTask оборачивает все потоки и взаимодействия с пользовательским интерфейсом таким образом, что предлагает как место для реализации нашего сетевого кода, так и место для взаимодействия с пользовательским интерфейсом. Файл MainActivity.java, показанный на рисунке 1, содержит два класса: Activity нашего приложения и нашу реализацию AsyncTask.

Код Activity довольно скучный — мы расширяем пользовательский интерфейс из activity_main.xml, получаем ссылки на EditText, который разрешает ввод URL-адреса, TextView для отображения извлеченного содержимого страницы и Button для запуска действия.

Единственным интересным элементом в этом классе является вызов setMovementMethod для TextView. Этот метод заставляет TextView прокручиваться по вертикали, чтобы увидеть все извлеченные данные.

В обработчике нажатия кнопки мы создаем экземпляр класса GetData, который расширяет AsyncTask.

В конструкторе GetData мы передаем элемент пользовательского интерфейса, который мы хотим впоследствии обновить, — в данном случае наш TextView.

Тяжелая работа с AsyncTask выполняется в методе doInBackground. Именно здесь классы URL и HttpURLConnection объединяются для обеспечения фактического подключения к веб-сайту по выбору пользователя. Экземпляр BufferedReader заботится о чтении данных, поступающих из соединения с веб-сайтом. По мере чтения каждой строки текст добавляется к StringBuilder. После завершения doInBackground метод onPostExecute получает результаты doInBackground и обновляет пользовательский интерфейс. Данные о фоновом потоке? Проверять. Обновить пользовательский интерфейс в основном потоке? Проверять. Спасибо, AsyncTask!

В этом примере приложение Android обменивается данными с веб-сервером HTTP, таким как Apache или Internet Information Server (IIS на сервере Microsoft®). Хотя HTTP является чрезвычайно

популярным протоколом, в Интернете используются и другие протоколы. Если бы приложение общалось напрямую с сокетом TCP, а не с HTTP, вы бы реализовали сетевой код несколько иначе. В листинге 3 показан фрагмент кода, показывающий другой способ взаимодействия с удаленным сервером. Листинг реализован как отдельный поток, а не в методе doInBackground реализации AsyncTask. В этом примере код считывает один байт (или символ) за раз. Как только полный ответ прочитан, данные упаковываются в экземпляр класса Message и отправляются экземпляру Handler. Хотя обработчик не показан в этом списке, это показывает альтернативный способ получения данных и отправки их в другой поток. Рекомендуется придерживаться подхода AsyncTask, если только не представится более неотложная причина.

ЛИТЕРАТУРА

1. Википедия. Информационная система. [Электронный ресурс] Режим доступа: https://ru.wikipedia.org/wiki/Информационная_система - Дата доступа: 22.04.2022. 3.
2. MySQL Workbench. Documentation [Электронный ресурс] Режим доступа: https://dev.mysql.com/doc/ – Дата доступа: 22.05.2022. 8.