7.1. Введение

Приложение `WeatherViewer` (рис. 7.1) использует бесплатные REST-совместимые веб-сервисы *OpenWeatherMap.org* для получения 16-дневного прогноза погоды для заданного города. Приложение получает данные в формате *JSON* (JavaScript Object Notation). Результаты отображаются в `ListView` — компоненте для вывода списка с поддержкой прокрутки. В этом приложении будет использоваться пользовательский формат элементов списка:

- значок погодных условий,
- день недели с текстовым описанием погоды в этот день,
- самая высокая и самая низкая температура за день (по шкале Фаренгейта) и
- влажность в процентах.

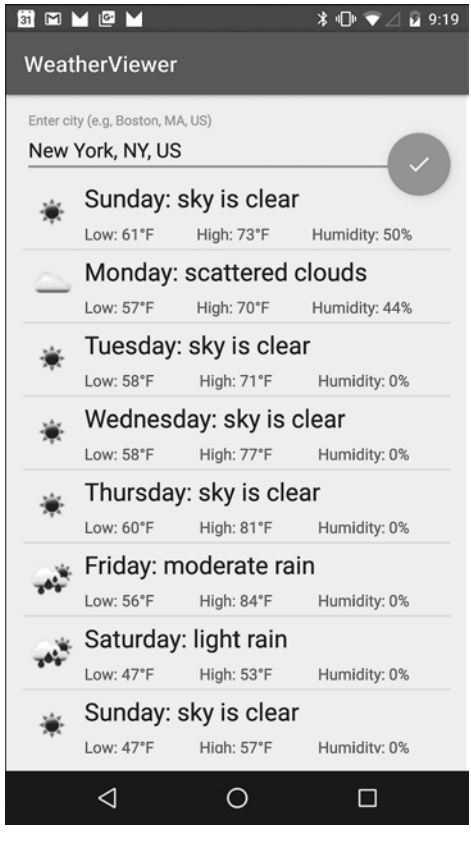

Рис. 7.1. Приложение WeatherViewer с прогнозом погоды для Нью-Йорка (США)

Эти элементы составляют небольшое подмножество возвращаемых данных прогноза. За подробной информацией о данных, возвращаемых API 16-дневного прогноза погоды, обращайтесь по адресу

http://openweathermap.org/forecast16

Полный список погодных API, предоставляемых *OpenWeatherMap.org*, доступен по адресу

http://openweathermap.org/api

7.2. Тестирование приложения WeatherViewer

Запустите Android Studio и откройте приложение `WeatherViewer` из папки `WeatherViewer` в примерах книги. Прежде чем запускать приложение, необходимо добавить собственный ключ API для *OpenWeatherMap.org*. О том, как получить ключ и где он должен храниться в проекте, рассказано в разделе 7.3.1. Этот шаг обязательно должен быть выполнен до запуска приложения. После того как ключ API будет добавлен в проект, запустите приложение в эмуляторе AVD или на устройстве.

Просмотр 16-дневного прогноза погоды

При запуске приложения фокус передается полю `EditText` в верхней части интерфейса пользователя, и на экране появляется виртуальная клавиатура для ввода названия города (рис. 7.2). Например, для получения прогноза для Нью-Йорка была введена строка `New York, NY, US`. После того как название города будет введено, коснитесь круглой кнопки `FloatingActionButton` со значком ✓ для передачи данных приложению. Приложение запрашивает 16-дневный прогноз погоды для выбранного вами города (рис. 7.2).

7.3. Обзор применяемых технологий

В этом разделе представлены новые технологии, которые будут использоваться для создания приложения `WeatherViewer`.

7.3.1. Веб-сервисы

В этой главе мы начнем использовать веб-сервисы, обеспечивающие портируемость и расширяющие возможности повторного использования кода в приложениях, работающих по Интернету. *Веб-сервис* представляет собой программный компонент, к которому можно обращаться по сети.

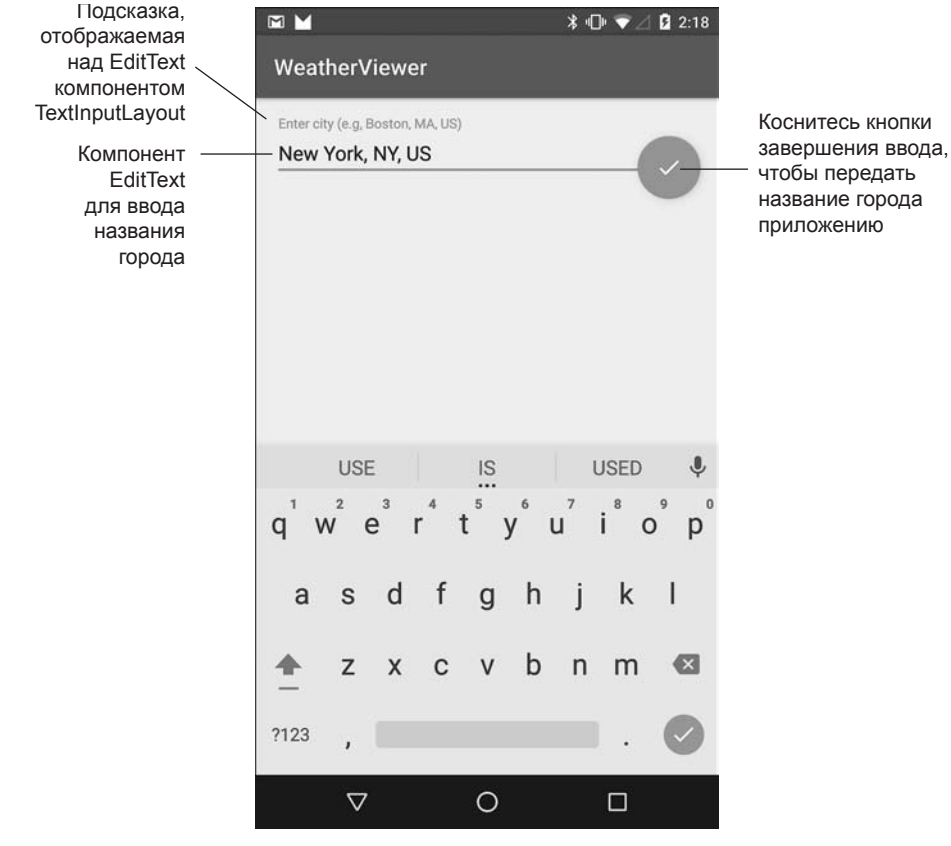

Рис. 7.2. Ввод названия города

Машина, на которой работает веб-сервис, называется *хостом*. *Клиент* — в данном случае приложение `WeatherViewer` — отправляет запрос хосту по сети. Хост обрабатывает запрос и возвращает ответ клиенту по сети. Распределенные вычисления обладают рядом преимуществ. Например, приложение может запрашивать данные по мере необходимости через веб-сервис, вместо того чтобы хранить их прямо на устройстве. Или, скажем, приложение в системе, не обеспечивающей необходимых вычислительных мощностей, может воспользоваться превосходящими ресурсами другой системы.

REST-совместимые веб-сервисы

Сокращением REST (Representational State Transfer) обозначается архитектурный стиль реализации веб-сервисов — такие сервисы часто называются «REST-совместимыми» (RESTful). Многие из популярных веб-сервисов — как бесплатных, так и коммерческих — являются REST-совместимыми. Хотя стиль

REST сам по себе не является стандартом, REST-совместимые веб-сервисы используют веб-стандарты — такие как протокол HTTP, используемый браузерами для взаимодействия с веб-серверами. Каждый метод REST-совместимого веб-сервиса представляется уникальным URL-адресом. Таким образом, когда сервер получает запрос, он немедленно узнает, какую операцию следует выполнить. Такие веб-сервисы могут использоваться в приложениях, но иногда их адреса можно вводить прямо в адресной строке браузера.

Веб-сервисам часто требуется ключ API

Для использования веб-сервиса часто необходимо получить уникальный ключ API у поставщика веб-сервиса. Когда ваше приложение обращается с запросом к веб-сервису, ключ API позволяет поставщику:

- убедиться в том, что вам разрешено использовать данный веб-сервис, и
- следить за интенсивностью использования — многие веб-сервисы ограничивают количество запросов за заданный промежуток времени (в секунду, в минуту, в час и т. д.).

Некоторые веб-сервисы требуют выполнения аутентификации перед получением ключа API — фактически вы должны подключиться к веб-сервису на программном уровне, прежде чем вам будет разрешено ее использовать.

Веб-сервисы OpenWeatherMap.org

Веб-сервисы *OpenWeatherMap.org*, которые будут использоваться в приложении `WeatherViewer`, бесплатны, но *OpenWeatherMap.org* ограничивает количество запросов к веб-сервису — в настоящее время эти ограничения составляют 1200 запросов в минуту и 1,7 миллиона запросов в день. Сервис *OpenWeatherMap.org* является условно-бесплатным — наряду с бесплатным уровнем, который будет использоваться в нашем приложении, существуют платные услуги с более высокими порогами запросов, более частыми обновлениями и другими возможностями. За дополнительной информацией о веб-сервисах *OpenWeatherMap.org* обращайтесь по адресу:

http://openweathermap.org/api

Лицензия на использование веб-сервиса OpenWeatherMap.org

Доступ к веб-сервисам *OpenWeatherMap.org* предоставляется на условиях открытой лицензии Creative Commons. За условиями лицензии обращайтесь по адресу

http://creativecommons.org/licenses/by-sa/2.0/

Дополнительная информация об условиях лицензии доступна по адресу

http://openweathermap.org/terms

Получение ключа API OpenWeatherMap.org

Прежде чем запускать это приложение, необходимо получить собственный ключ API OpenWeatherMap.org по адресу

http://openweathermap.org/register

После регистрации скопируйте шестнадцатеричный ключ API со страницы подтверждения, а затем замените строку `YOUR_API_KEY` в файле `strings.xml` скопированным ключом.

7.3.2. Формат JSON и пакет org.json

Формат JSON (JavaScript Object Notation) может использоваться для представления данных вместо XML. JSON — текстовый формат обмена данных, используемый для представления объектов в JavaScript в виде коллекций пар «имя—значение», представленных в строковом виде. Это простой формат, в котором легко создавать, читать и разбирать объекты; к тому же он существенно компактнее XML, что повышает эффективность передачи данных по Интернету. Каждый объект JSON представляется в виде списка имен и значений свойств, заключенных в фигурные скобки, в следующем формате:

{свойствоИмя1: значение1, свойствоИмя2: значение2}

Каждое имя свойства представляет собой `String`. Массивы в JSON заключаются в квадратные скобки в следующем формате:

[значение1, значение2, значение3]

Каждый элемент массива может быть строкой, числом, объектом JSON, `true`, `false` или `null`.

В листинге 7.1 приведен пример данных в формате JSON, возвращаемых сервисом прогноза погоды *OpenWeatherMap.org*. Этот конкретный пример содержит погодные данные за два дня (строки 15–57).

Листинг 7.1. Пример JSON из OpenWeatherMap.org — веб-сервиса прогноза погоды

```
 1 {
 2    "city": {
 3       "id": 5128581,
 4       "name": "New York",
 5       "coord": {
 6          "lon": -74.005966,
 7          "lat": 40.714272
 8       },
 9       "country": "US",
10       "population": 0
11    },
12    "cod": "200",
13    "message": 0.0102,
14    "cnt": 2,
```

```
{
   "list": [{ // Мы будем использовать этот массив объектов
       "dt": 1442419200,
       "temp": {
         "day": 79.9,
          "min": 71.74,
          "max": 82.53,
         "night": 71.85,
         "eve": 82.53,
         "morn": 71.74
      },
      "pressure": 1037.39,
      "humidity": 64,
      "weather": [{
         "id": 800,
         "main": "Clear",
         "description": "sky is clear",
         "icon": "01d"
      }],
      "speed": 0.92,
      "deg": 250,
      "clouds": 0
   }, { // Конец первого элемента массива, начало второго
      "dt": 1442505600,
      "temp": {
         "day": 79.92,
         "min": 66.72,
         "max": 83.1,
         "night": 70.79,
         "eve": 81.99,
         "morn": 66.72
      },
      "pressure": 1032.46,
      "humidity": 62,
      "weather": [{
         "id": 800,
         "main": "Clear",
         "description": "sky is clear",
         "icon": "01d"
      }],
      "speed": 1.99,
      "deg": 224,
      "clouds": 0
   }] // Конец второго элемента и конец массива
}
```

Объект JSON, возвращаемый в прогнозе погоды, содержит много свойств. Мы будем использовать только свойство `"list"` — массив объектов JSON, представляющих прогнозы на время до 16 дней (7 по умолчанию, если в запросе не указано обратное). Каждый элемент массива `"list"` также содержит много свойств, из которых мы будем использовать:

- `"dt"` — целое значение типа `long` с временной меткой, представленной в виде количества секунд, прошедших с 1 января 1970 года (GMT). Мы преобразуем это значение в название дня;

- `"temp"` — объект JSON со свойствами `double`, представляющими дневные температуры. В приложении используется только минимальная (`"min"`) и максимальная (`"max"`) температуры, но веб-сервис также возвращает среднюю дневную (`"day"`), ночную (`"night"`), вечернюю (`"eve"`) и утреннюю (`"morn"`) температуру;
- `"humidity"` — значение `int`, представляющее влажность в процентах;
- `"weather"` — объект JSON с несколькими свойствами, включая описание погодных условий (`"description"`) и имя значка, представляющего условия (`"icon"`).

Пакет org.json

Для обработки данных JSON, получаемых приложением, будут использоваться следующие классы из пакета `org.json` (раздел 7.7.6):

- `JSONObject` — один из конструкторов класса преобразует строку данных JSON в объект `JSONObject`, который содержит коллекцию `Map<String, Object>`, связывающую ключи JSON со значениями. Для обращения к свойствам JSON в коде используются методы `get` класса `JSONObject`, что позволяет получить значение, ассоциированное с ключом, в одном из типов `JSONObject`, `JSONArray`, `Object`, `boolean`, `double`, `int`, `long` или `String`;
- `JSONArray` — класс представляет массив данных JSON и предоставляет методы для обращения к его элементам. В приложении `WeatherViewer` свойство `"list"` ответа *OpenWeatherMap.org* будет обрабатываться как `JSONArray`.

7.3.3. HttpUrlConnection и обращение к REST-совместимым веб-сервисам

Чтобы передать вызов веб-сервису прогноза погоды *OpenWeatherMap.org*, мы преобразуем строку URL-адреса веб-сервиса в объект `URL`, а затем используем `URL` для открытия `HttpUrlConnection` (раздел 7.7.5). При этом выдается запрос HTTP к веб-сервису. Чтобы получить ответ JSON, мы прочитаем все данные из потока `InputStream` объекта `HttpUrlConnection` и поместим их в `String`. Мы покажем, как преобразовать их в `JSONObject` для обработки.

7.3.4. Использование AsyncTask для обработки сетевых запросов вне потока GUI

Продолжительные операции, а также операции, которые блокируют выполнение приложения до их завершения (сетевые операции, обращения к файлам

и базам данных), должны выполняться вне потока графического интерфейса. Тем самым обеспечивается быстрый отклик приложения и предотвращается появление диалоговых окон ANR (Activity Not Responding), которые выводятся системой Android, если она считает, что графический интерфейс перестал реагировать на действия пользователя. Однако вспомните, о чем говорилось в главе 6: обновления интерфейса пользователя должны происходить в потоке GUI, потому что компоненты GUI не обладают потоковой безопасностью.

Для выполнения долгих операций, результатом которых является обновление графического интерфейса, Android предоставляет класс `AsyncTask` (пакет `android.os`). Этот класс выполняет продолжительную операцию в одном потоке и передает результаты потоку GUI. Класс `AsyncTask` берет на себя все подробности создания и управления потоками, как и передачу результатов от `AsyncTask` потоку GUI. В приложении будут использоваться два субкласса `AsyncTask` — один будет обращаться с вызовом к веб-сервису *OpenWeatherMap.org* (раздел 7.7.5), а другой загружает изображение, представляющее погодные условия (раздел 7.6.5).

7.3.5. ListView, ArrayAdapter и паттерн View-Holder

Приложение выводит погодные данные в компоненте `ListView` (пакет `android.widget`) — списке с поддержкой прокрутки. Класс `ListView` является субклассом класса `AdapterView` (пакет `android.widget`), представляющего получение данных от источника через объект `Adapter` (пакет `android.widget`). В этом приложении субкласс `ArrayAdapter` (пакет `android.widget`) используется для создания объекта, заполняющего `ListView` данными из объекта коллекции `ArrayList` (раздел 7.6). Когда приложение обновляет `ArrayList` погодными данными, мы вызываем метод `notifyDataSetChanged` класса `ArrayAdapter`, чтобы сообщить об изменении данных в `ArrayList`. Адаптер сообщает `ListView` о необходимости изменить список отображаемых данных. Этот механизм называется *связыванием данных* (data binding). Существует несколько типов `AdapterView`, которые могут связываться с данными с использованием адаптера. В главе 9 вы узнаете, как связать компонент `ListView` с информацией из базы данных. Дополнительную информацию о связывании данных в Android, а также некоторые обучающие примеры можно найти по адресу

http://developer.android.com/guide/topics/ui/binding.html

Паттерн View-Holder

По умолчанию компонент `ListView` может отображать один или два компонента `TextView`. В этом приложении мы настроим элементы `ListView` так, чтобы в них отображался компонент `ImageView` и несколько компонентов `TextView`

в пользовательском (нестандартном) макете. Создание пользовательских элементов `ListView` сопряжено с высокими затратами ресурсов на динамическое создание новых объектов. В больших списках со сложными макетами элементов, в которых пользователь быстро прокручивает содержимое, это может привести к нарушению плавности прокрутки. Для сокращения затрат ресурсов при выходе элементов за границы экрана Android повторно использует элементы списка для новых объектов, появляющихся на экране.

В паттерне `View-Holder` создается класс (обычно с именем `ViewHolder`), который содержит переменные экземпляров для представлений, отображающих данные элементов `ListView`. При создании элемента `ListView` также создается объект `ViewHolder`, переменные которого инициализируются ссылками на вложенные представления элемента. Затем объект `ViewHolder` сохраняется вместе с элементом `ListView`, который относится к классу `View`. Метод `setTag` класса `View` позволяет добавить к `View` произвольный объект. В дальнейшем этот объект может быть получен методом `getTag` класса `View`. С этими вызовами используется объект `ViewHolder`, содержащий ссылки на вложенные представления элемента `ListView`.

Когда новый элемент готовится к появлению на экране, компонент `ListView` проверяет наличие представления, доступного для повторного использования. Если такого представления нет, мы заполняем представление нового элемента на основании XML-файла макета, а затем сохраняем ссылки на компоненты GUI в объекте `ViewHolder`. Затем этот объект `ViewHolder` связывается с элементом `ListView` вызовом `setTag`. Если элемент, пригодный для повторного использования, существует, мы получаем ассоциированный с ним объект `ViewHolder` вызовом `getTag`; вызов возвращает существующий объект `ViewHolder`, созданный ранее для элемента `ListView`. Независимо от того, как был получен объект `ViewHolder`, данные выводятся в представлениях, ссылки на которые хранятся в `ViewHolder`.

7.3.6. FloatingActionButton

Пользователи нажимают кнопки для выполнения операций. С появлением концепции материального дизайна в Android 5.0 компания Google представила кнопку `FloatingActionButton`, «плавающую» над интерфейсом пользователя (иначе говоря, эта кнопка находится на большей высоте, чем остальные элементы интерфейса) и обозначающую некоторое важное действие. Например, в адресной книге плавающая кнопка со значком + может выделять действие добавления нового контакта. В нашем приложении плавающая кнопка со значком ✓ будет использоваться для передачи приложению названия города и получения прогноза для этого города. С выходом Android 6.0 и новой библиотеки Android Design Support Library компания Google формализовала плавающую

кнопку в классе FloatingActionButton (пакет android.support.design.widget). В Android Studio 1.4 были введены новые реализации шаблонов приложений, использующие материальный дизайн, и во многие новые шаблоны кнопка FloatingActionButton включалась по умолчанию.

Класс FloatingActionButton является субклассом ImageView, что позволяет выводить изображение на FloatingActionButton. Рекомендации материального дизайна советуют размещать кнопки FloatingActionButton по крайней мере в 16dp от границ экрана на телефонах и по крайней мере в 24dp от границ экрана на планшетах — шаблоны по умолчанию настраивают эти интервалы за вас. За дополнительной информацией о том, когда и как следует использовать кнопку FloatingActionButton, обращайтесь по адресу

https://www.google.com/design/spec/components/buttons-floating-action-button.html

7.3.7. Компонент TextInputLayout

В этом приложении компонент EditText будет использоваться для ввода названия города, для которого запрашивается прогноз погоды. Чтобы пользователю было проще понять назначение EditText, можно создать подсказку, которая будет отображаться, пока в EditText нет символов. Когда пользователь начинает вводить текст, подсказка исчезает — и пользователь может забыть, для чего нужно данное поле.

Компонент TextInputLayout из библиотеки поддержки Android (пакет android.support.design.widget) решает эту проблему. При получении фокуса EditText компонент TextInputLayout уменьшает текст подсказки и размещает его над EditText, чтобы подсказка была видна и в процессе ввода данных (см. рис. 7.2). В нашем приложении EditText получает фокус в начале выполнения приложения, так что TextInputLayout немедленно перемещает подсказку над EditText.

7.3.8. Компонент Snackbar

Компонент материального дизайна Snackbar (пакет android.support.design.widget) по своему назначению напоминает временные оповещения Toast. Он не только появляется на экране на заданный промежуток времени, но и поддерживает интерактивные взаимодействия. Пользователь может провести пальцем по экрану, чтобы закрыть оповещение. С компонентом Snackbar также может быть связано действие, выполняемое при касании Snackbar. В приложении WeatherViewer компонент Snackbar будет использоваться для вывода информационных сообщений.