7.2: Hubungkan ke Internet

Materi:

- Pengantar
- Keamanan jaringan
- Menyertakan izin dalam manifes
- Menjalankan operasi jaringan di thread pekerja
- Membuat koneksi HTTP
- Mem-parse hasil
- Mengelola keadaan jaringan
- Praktik terkait
- Ketahui selengkapnya

Sebagian besar aplikasi Android memiliki beberapa data yang berinteraksi dengan pengguna; seperti artikel berita, informasi cuaca, kontak, data game, informasi pengguna, dan lainnya. Seringkali, data ini disediakan melalui jaringan oleh API web.

Dalam pelajaran ini, Anda akan mempelajari tentang keamanan jaringan dan cara membuat panggilan jaringan, yang melibatkan tugas-tugas ini:

1. Sertakan izin dalam file AndroidManifest.xml.
2. Pada thread pekerja, buat koneksi klien HTTP yang menghubungkan ke jaringan dan mengunduh (atau mengunggah) data.
3. Parse hasil, yang biasanya dalam format JSON.
4. Periksa keadaan jaringan dan tanggapi sebagaimana semestinya.

Keamanan jaringan

Transaksi jaringan berisiko inheren, karena mereka melibatkan transmisi data yang boleh jadi bersifat privat untuk pengguna. Orang semakin sadar dengan risiko ini, terutama ketika perangkat mereka melakukan transaksi jaringan, jadi sangat penting jika aplikasi Anda mengimplementasikan praktik terbaik untuk menjaga data pengguna tetap aman setiap saat.

Praktik terbaik keamanan untuk operasi jaringan:

- Gunakan protokol yang sesuai untuk data sensitif. Misalnya untuk lalu lintas web aman, gunakan subkelas `HttpsURLConnection` dari `HttpURLConnection`.
- Gunakan HTTPS sebagai ganti HTTP di mana saja HTTPS didukung pada server, karena perangkat seluler sering menghubungkan pada jaringan tidak aman seperti hotspot Wi-Fi publik. Pertimbangkan penggunaan `SSLSocketClass` untuk mengimplementasikan komunikasi level soket yang diautentikasi dan dienkripsi.
- Jangan gunakan porta jaringan host lokal untuk menangani komunikasi interproses sensitif (IPC), karena aplikasi lain di perangkat bisa mengakses porta lokal ini. Sebagai gantinya, gunakan mekanisme yang memungkinkan Anda menggunakan autentikasi, misalnya `Service`.
- Jangan mempercayai data yang telah diunduh dari HTTP atau protokol tidak aman lainnya. Validasi masukan yang dimasukkan ke dalam `WebView` dan respons ke maksud yang Anda keluarkan terhadap HTTP.

Untuk praktik terbaik dan tips keamanan, lihat artikel Tips Keamanan.

Menyertakan izin dalam manifes

Sebelum aplikasi bisa melakukan panggilan jaringan, Anda perlu menyertakan izin dalam file AndroidManifest.xml. Tambahkan tag berikut di dalam tag `<manifest>` :

```
<uses-permission android:name="android.permission.INTERNET" />
```

Saat menggunakan jaringan, praktik terbaiknya adalah memantau keadaan jaringan perangkat sehingga Anda tidak berupaya membuat panggilan jaringan ketika jaringan tidak tersedia. Untuk mengakses keadaan jaringan perangkat, aplikasi memerlukan izin tambahan:

```
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

Menjalankan operasi jaringan di thread pekerja

Selalu jalankan operasi jaringan di thread pekerja, terpisah dari UI. Misalnya, di kode Java, Anda bisa membuat implementasi `AsyncTask` (atau `AsyncTaskLoader`) yang membuka koneksi jaringan dan mengkueri API. Kode utama Anda akan memeriksa apakah koneksi jaringan aktif. Jika aktif, maka akan menjalankan `AsyncTask` di thread terpisah, kemudian menampilkan hasil di UI.

Catatan: Jika menjalankan operasi jaringan di thread utama sebagai ganti di thread pekerja, Anda akan menerima kesalahan.

Membuat koneksi HTTP

Sebagian besar aplikasi Android yang terhubung ke jaringan menggunakan HTTP dan HTTPS untuk mengirim dan menerima data melalui jaringan. Untuk penyegaran di HTTP, kunjungi Pelajari tutorial HTTP.

Catatan: Jika server web menawarkan HTTPS, Anda harus menggunakannya sebagai ganti HTTP untuk peningkatan keamanan.
Klien Android `HttpURLConnection` mendukung HTTPS, unggahan dan unduhan streaming, waktu tunggu yang dapat dikonfigurasi, IPv6, dan penjajakan koneksi. Untuk menggunakan klien `HttpURLConnection`, bangun URI (tujuan permintaan). Selanjutnya dapatkan koneksi, kirim permintaan dan header permintaan apa pun, unduh dan baca respons dan header respons apa pun, dan putuskan koneksi.

Membangun URI Anda

Untuk membuka koneksi HTTP, Anda perlu membangun URI permintaan. URI biasanya dibuat dari URL dasar dan sekumpulan parameter kueri yang menetapkan sumber daya yang dimaksud. Misalnya untuk menelusuri lima hasil buku pertama untuk "Pride and Prejudice" di Google Books API, gunakan URI berikut:

```
https://www.googleapis.com/books/v1/volumes?q=pride+prejudice&maxResults=5&printType=books
```

Untuk menyusun URI permintaan lewat program, gunakan metode `URI.parse()` dengan metode `buildUpon()` dan `appendQueryParameter()`. Kode berikut membangun URI lengkap yang ditampilkan di atas:

```
// Base URL for the Books API.
final String BOOK_BASE_URL =  "https://www.googleapis.com/books/v1/volumes?";

final String QUERY_PARAM = "q"; // Parameter for the search string
final String MAX_RESULTS = "maxResults"; // Parameter to limit search results.
final String PRINT_TYPE = "printType"; // Parameter to filter by print type

// Build up the query URI, limiting results to 5 items and printed books.
Uri builtURI = Uri.parse(BOOK_BASE_URL).buildUpon()
       .appendQueryParameter(QUERY_PARAM, "pride+prejudice")
       .appendQueryParameter(MAX_RESULTS, "5")
       .appendQueryParameter(PRINT_TYPE, "books")
       .build();
```

Untuk mengubah URI menjadi string, gunakan metode `toString()` :

```
String myurl = builtURI.toString();
```

Hubungkan dan unduh data

Di thread pekerja yang menjalankan transaksi jaringan Anda, misalnya dalam penggantian metode `doInBackground()` di `AsyncTask` , gunakan kelas `HttpURLConnection` untuk menjalankan permintaan `GET` HTTP dan mengunduh data yang diperlukan aplikasi. Begini caranya:

1. Untuk memperoleh `HttpURLConnection` baru, panggil `URL.openConnection()` menggunakan URI yang Anda bangun. Transmisikan hasilnya ke `HttpURLConnection` .

   URI adalah properti utama permintaan, namun header permintaan juga bisa menyertakan metadata seperti kredensial, tipe materi pilihan, dan cookie sesi.

2. Setel parameter opsional:
   - Untuk koneksi lambat, Anda mungkin menginginkan waktu tunggu koneksi yang lama (waktu untuk membuat koneksi awal ke sumber daya) atau waktu tunggu baca (waktu untuk benar-benar membaca data).
   - Untuk mengubah metode permintaan ke selain `GET` , gunakan `setRequestMethod()` .
   - Jika Anda tidak ingin menggunakan jaringan untuk masukan, setel `setDoInput` ke `false` . (Defaultnya adalah `true` .)
   - Untuk metode lain yang bisa Anda setel, lihat dokumentasi referensi `HttpURLConnection` dan `URLConnection.`
3. Buka aliran masukan menggunakan `getInputStream()` , kemudian baca respons dan ubah menjadi string. Header respons biasanya menyertakan metadata seperti tipe materi dan panjang isi respons, tanggal modifikasi, dan cookie sesi. Jika respons tidak memiliki isi, `getInputStream()` akan mengembalikan aliran kosong.
4. Panggil `disconnect()` untuk menutup koneksi. Memutus koneksi akan melepas sumber daya yang ditahan oleh koneksi sehingga sumber daya bisa ditutup atau digunakan kembali.

Langkah-langkah ini ditampilkan dalam Contoh permintaan, di bawah.

Jika mengeposkan data melalui jaringan dan tidak hanya menerima data, Anda perlu mengunggah *isi permintaan*, yang menampung data yang akan diposkan. Caranya:

1. Konfigurasikan koneksi sehingga keluaran dimungkinkan dengan memanggil `setDoOutput(true)` . (Secara default, `HttpURLConnection` menggunakan permintaan `GET` HTTP. Bila `setDoOutput` adalah `true` , `HttpURLConnection` menggunakan permintaan `POST` HTTP secara default.)
2. Buka aliran keluaran dengan memanggil `getOutputStream()` .

Untuk informasi selengkapnya tentang mengeposkan data ke jaringan, lihat "Mengeposkan Materi" dalam `dokumentasi HttpURLConnection` .

Catatan: Semua panggilan jaringan harus dijalankan di thread pekerja dan bukan di thread UI.

Contoh permintaan

Contoh berikut mengirimkan permintaan ke URL yang dibangun di bagian Membangun URI Anda, di atas. Permintaan memperoleh `HttpURLConnection` baru, membuka aliran masukan, membaca respons, mengubah respons menjadi string, dan menutup koneksi.

```
private String downloadUrl(String myurl) throws IOException {
    InputStream inputStream = null;
    // Only display the first 500 characters of the retrieved
    // web page content.
    int len = 500;

    try {
        URL url = new URL(myurl);
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();
        conn.setReadTimeout(10000 /* milliseconds */);
        conn.setConnectTimeout(15000 /* milliseconds */);
        // Start the query
        conn.connect();
        int response = conn.getResponseCode();
        Log.d(DEBUG_TAG, "The response is: " + response);
        inputStream = conn.getInputStream();

        // Convert the InputStream into a string
        String contentAsString = convertInputToString(inputStream, len);
        return contentAsString;

    // Close the InputStream and connection
    } finally {
      conn.disconnect();
        if (inputStream != null) {
            inputStream.close();
        }
    }
}
```

Mengubah InputStream menjadi string

`InputStream` merupakan sumber byte yang dapat dibaca. Setelah Anda mendapatkan `InputStream`, biasanya kemudian mendekode atau mengubahnya menjadi tipe data yang diperlukan. Dalam contoh di atas, `InputStream` menyatakan teks biasa dari laman web yang berada di https://www.googleapis.com/books/v1/volumes?q=pride+prejudice&maxResults=5&printType=books.

Metode `convertInputToString` yang didefinisikan di bawah ini mengubah `InputStream` menjadi string sehingga aktivitas bisa menampilkanya di UI. Metode ini menggunakan instance `InputStreamReader` untuk membaca byte dan mendekode byte menjadi karakter:

```
// Reads an InputStream and converts it to a String.
public String convertInputToString(InputStream stream, int len)
           throws IOException, UnsupportedEncodingException {
    Reader reader = null;
    reader = new InputStreamReader(stream, "UTF-8");
    char[] buffer = new char[len];
    reader.read(buffer);
    return new String(buffer);
}
```

Catatan: Jika Anda mengharapkan respons yang panjang, bungkus `InputStreamReader` di dalam `BufferedReader` agar pembacaan karakter, larik, dan baris menjadi lebih efisien. Misalnya:
`reader = new BufferedReader(new InputStreamReader(stream, "UTF-8"));`

Mem-parse hasil

Bila Anda membuat kueri API web, hasilnya seringkali dalam format JSON.

Di bawah ini adalah contoh respons JSON dari permintaan HTTP. Respons ini menampilkan nama tiga item menu di menu munculan dan metode yang dipicu bila item menu diklik:

```
{"menu": {
  "id": "file",
  "value": "File",
  "popup": {
    "menuitem": [
      {"value": "New", "onclick": "CreateNewDoc()"},
      {"value": "Open", "onclick": "OpenDoc()"},
      {"value": "Close", "onclick": "CloseDoc()"}
    ]
  }
}
```

Untuk menemukan nilai item dalam respons, gunakan metode dari kelas `JSONObject` dan `JSONArray`. Misalnya, inilah cara untuk menemukan nilai `"onclick"` dari item ketiga di larik `"menuitem"` :

```
JSONObject data = new JSONObject(responseString);
JSONArray menuItemArray = data.getJSONArray("menuitem");
JSONObject thirdItem = menuItemArray.getJSONObject(2);
String onClick = thirdItem.getString("onclick");
```

Mengelola keadaan jaringan

Membuat panggilan jaringan bisa mahal dan lambat, terutama jika perangkat memiliki konektivitas yang kecil. Mewaspadai keadaan koneksi jaringan bisa mencegah aplikasi Anda dari berupaya membuat panggilan jaringan saat jaringan tidak tersedia.

Kadang-kadang juga penting bagi aplikasi untuk mengetahui jenis konektivitas yang dimiliki perangkat: Jaringan Wi-Fi biasanya lebih cepat daripada jaringan data, dan jaringan data biasanya seringkali berkuota dan mahal. Untuk mengontrol kapan tugas tertentu dijalankan, pantau keadaan jaringan dan tanggapi sebagaimana mestinya. Misalnya, Anda mungkin ingin menunggu hingga perangkat terhubung ke Wi-Fi untuk melakukan pengunduhan file besar.

Untuk memeriksa koneksi jaringan, gunakan kelas berikut:

- `ConnectivityManager` akan menjawab kueri tentang keadaan konektivitas jaringan. Juga memberi tahu aplikasi bila konektivitas jaringan berubah.
- `NetworkInfo` akan menjelaskan status antarmuka jaringan dari tipe yang diberikan (saat ini seluler atau Wi-Fi).

Cuplikan kode berikut menguji apakah Wi-Fi dan seluler terhubung. Dalam kode:

- Metode `getSystemService` mendapatkan instance `ConnectivityManager` .
- Metode `getNetworkInfo` mendapatkan status koneksi Wi-Fi perangkat, kemudian koneksi selulernya. Metode `getNetworkInfo` mengembalikan objek `NetworkInfo` , yang berisi informasi tentang status koneksi jaringan yang diberikan (apakah menganggur, terhubung, dan seterusnya).
- Metode `networkInfo.isConnected()` akan mengembalikan `true` jika jaringan yang diberikan terhubung. Jika jaringan terhubung, maka bisa digunakan untuk membangun soket dan meneruskan data.

```
private static final String DEBUG_TAG = "NetworkStatusExample";
ConnectivityManager connMgr = (ConnectivityManager)
    getSystemService(Context.CONNECTIVITY_SERVICE);
NetworkInfo networkInfo = connMgr.getNetworkInfo(ConnectivityManager.TYPE_WIFI);
boolean isWifiConn = networkInfo.isConnected();
networkInfo = connMgr.getNetworkInfo(ConnectivityManager.TYPE_MOBILE);
boolean isMobileConn = networkInfo.isConnected();
Log.d(DEBUG_TAG, "Wifi connected: " + isWifiConn);
Log.d(DEBUG_TAG, "Mobile connected: " + isMobileConn);
```

Praktik terkait

Latihan terkait dan dokumentasi praktik ada di Dasar-Dasar Developer Android: Praktik.

- Menghubungkan ke Internet dengan AsyncTask dan AsyncTaskLoader

Ketahui selengkapnya

- Menghubungkan ke Jaringan
- Mengelola Penggunaan Jaringan
- Referensi HttpURLConnection
- Referensi ConnectivityManager
- Referensi InputStream