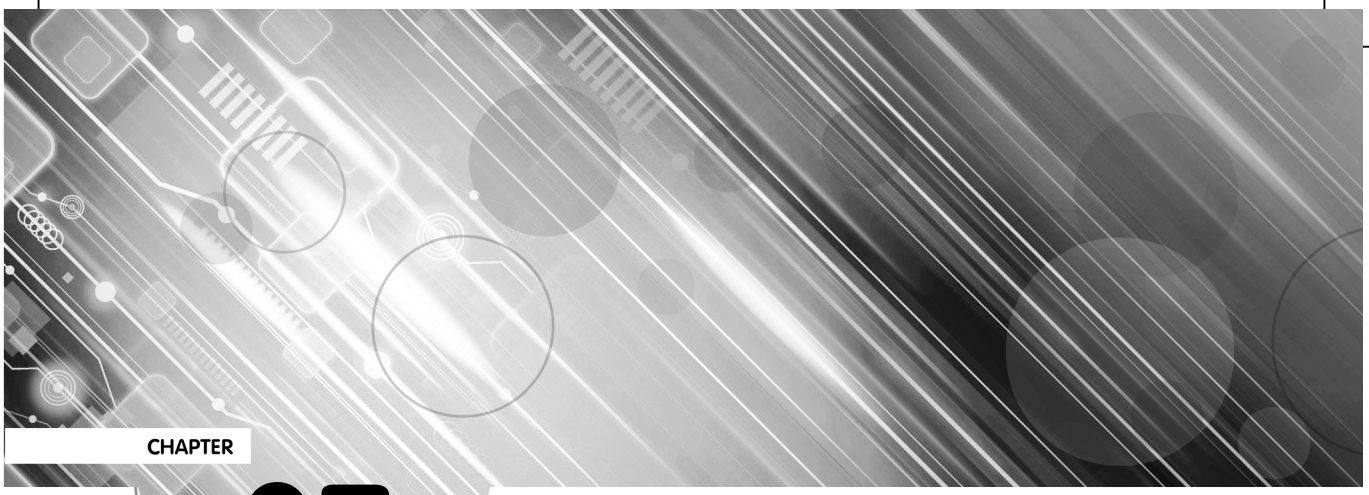


CHAPTER

07 การติดต่อเน็ตเวิร์ก

เนื้อหาในบทนี้

- การดาวน์โหลดไฟล์ข้อความจากเซิร์ฟเวอร์
- การดาวน์โหลดไฟล์รูปภาพจากเซิร์ฟเวอร์
- การดาวน์โหลดรูปภาพโดยสร้างซับคลาสของ ImageView
- การ Parse ข้อมูล JSON
- การเรียกใช้เว็บเซอร์วิส
- การ Parse ข้อมูล XML ด้วย SAX
- การ Parse ข้อมูล XML ด้วย XmlPull
- การอ่าน RSS Feed
- การส่งข้อมูลไปยังเซิร์ฟเวอร์ด้วย HTTP POST
- การอัพโหลดไฟล์ไปยังเซิร์ฟเวอร์

การดาวน์โหลดไฟล์ข้อความจากเซิร์ฟเวอร์

คุณต้องการอ่านข้อมูลจาก URL หนึ่งๆในอินเทอร์เน็ต ซึ่งไฟล์ที่อยู่ ณ URL นั้นอาจเป็นไฟล์ข้อความธรรมดา, ไฟล์เว็บเพจ HTML, ไฟล์ XML หรืออาจเป็นสคริปต์ฝั่งเซิร์ฟเวอร์ (Server-side script) ที่ให้ผลลัพธ์กลับมาเป็นข้อความธรรมดา, HTML, XML หรือ JSON ฯลฯ ก็ได้เช่นกัน

NOTE

การอ่านข้อมูลจาก URL หนึ่งๆนี้ ในทางเทคนิคก็คือการส่ง HTTP Request (ส่งคำขอผ่านโปรโตคอล HTTP) ไปยังเซิร์ฟเวอร์เพื่อขอข้อมูลจากไฟล์ที่ URL นั้น

ตัวอย่าง

หน้าจอของแอพนี้จะมีปุ่ม 1 ปุ่ม ซึ่งเมื่อคลิกจะดาวน์โหลดข้อมูลจากหน้าหลักของเว็บไซต์ www.promlert.com มาแสดงใน TextView

1 กำหนด Layout ของหน้าจอ

โปรเจ็ค HttpDownloadText, ไฟล์ activity__main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp" >

    <Button
        android:id="@+id/download_button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Download Text" />

    <TextView
        android:id="@+id/text"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:text="@string/hello_world" />

</LinearLayout>
```

2 เพิ่มโค้ดในเมธอด onCreate ของแอคทิวิตี

โปรเจ็ค HttpDownloadText, ไฟล์ MainActivity.java

```
Button button = (Button) findViewById(R.id.download_button);
button.setOnClickListener(new View.OnClickListener() {
```

```
    @Override
    public void onClick(View v) {
        DownloadTextTask task = new DownloadTextTask();
        task.execute("http://www.promlert.com");
    }
});
```

3 เพิ่มเมธอด downloadText และเมธอด readStream ในแอคทิวิตี

โปรเจ็ค HttpDownloadText, ไฟล์ MainActivity.java

```
private String downloadText(String strUrl) {
    String strResult = "";
    try {
        URL url = new URL(strUrl);
        HttpURLConnection con = (HttpURLConnection) url.openConnection();
        strResult = readStream(con.getInputStream());
    } catch (Exception e) {
        e.printStackTrace();
    }
    return strResult;
}

private String readStream(InputStream in) {
    BufferedReader reader = null;
    StringBuilder sb = new StringBuilder();
    try {
        reader = new BufferedReader(new InputStreamReader(in));
        String line;
        while ((line = reader.readLine()) != null) {
            sb.append(line + "\n");
        }
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (reader != null) {
            try {
                reader.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
    return sb.toString();
}
```

4 เพิ่มคลาส DownloadTextTask ในแอคทิวิตี (คลาส DownloadTextTask จะเป็น Inner Class ของแอคทิวิตี)

โปรเจ็ค HttpDownloadText, ไฟล์ MainActivity.java

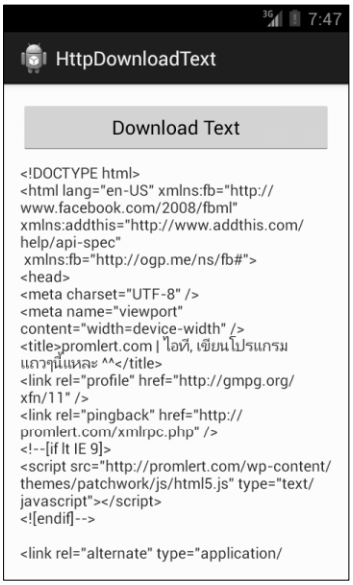
```
private class DownloadTextTask extends AsyncTask<String, Void, String> {

    @Override
    protected String doInBackground(String... urls) {
        return downloadText(urls[0]);
    }

    @Override
    protected void onPostExecute(String result) {
        TextView textview = (TextView) findViewById(R.id.text);
        textview.setText(result);
    }
}
```

5 เพิ่มบรรทัดต่อไปนี้ในไฟล์ AndroidManifest.xml โดยใส่ไว้ก่อนแท็ก <application> เพื่อขอสิทธิ์ในการเข้าถึงอินเทอร์เน็ต

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

ผลการรัน

เมื่อรันแล้วคลิกปุ่ม Download Text จะปรากฏข้อมูลใน TextView ดังรูป ข้อมูลนี้คือแท็ก HTML ของเพจหลักของเว็บไซต์ www.promlert.com ซึ่งเป็นข้อมูลเดียวกับที่เราเห็นตอนใช้คำสั่ง View Source ของโปรแกรมเว็บบราวเซอร์



คำอธิบาย

แอนดรอยด์มีไลบรารีสำหรับเชื่อมต่อเน็ตเวิร์กผ่านโปรโตคอล HTTP อยู่ 2 ไลบรารีคือ `HttpClient` ของ Apache กับ `HttpURLConnection` ของจาวา ในที่นี้เลือกใช้ `HttpURLConnection` เนื่องจากเป็นไลบรารีที่ทีมผู้พัฒนาแอนดรอยด์ปรับปรุงให้มีประสิทธิภาพและความสามารถเพิ่มขึ้นอยู่อย่างต่อเนื่อง และเอกสารของแอนดรอยด์ก็แนะนำให้ใช้ไลบรารีตัวนี้ ส่วนไลบรารี `HttpClient` ที่แอนดรอยด์หยิบยืมมาจาก Apache นั้นข้อดีหลักๆคือมี abstraction มากกว่า ทำให้เราเขียนโค้ดน้อยกว่า

การเชื่อมต่อและอ่านข้อมูลจากเซิร์ฟเวอร์

ในตัวอย่างนี้ โค้ดที่ใช้เชื่อมต่อและอ่านข้อมูลจากเซิร์ฟเวอร์จะอยู่ในเมธอด `downloadText` และเมธอด `readStream` โดยการทำงานเริ่มจากเมธอด `downloadText`

```
private String downloadText(String strUrl) {
    String strResult = "";
    try {
        URL url = new URL(strUrl); ❶
        HttpURLConnection con = (HttpURLConnection) url.openConnection(); ❷
        strResult = readStream(con.getInputStream()); ❸
    } catch (Exception e) {
        e.printStackTrace();
    }
    return strResult; ❹
}
```

อันดับแรกเราสร้างออบเจ็ค `URL` โดยระบุ URL ที่ต้องการข้อมูล ❶ แล้วทำการเชื่อมต่อเซิร์ฟเวอร์โดยใช้เมธอด `openConnection` ❷ การเชื่อมต่อจะถูกเก็บในตัวแปร `con` จากนั้นจึงขอข้อมูลจาก URL ที่ระบุโดยใช้เมธอด `getInputStream` ❸ ซึ่งจะได้ผลลัพธ์เป็นออบเจ็ค `InputStream` กลับมา และเราจะอ่านข้อมูลจาก `InputStream` นี้โดยใช้เมธอด `readStream` ที่เขียนขึ้นเอง ก่อนจะ return ข้อมูลที่อ่านได้ทั้งหมดออกไป ❹

สำหรับเมธอด `readStream` มีรายละเอียดดังนี้

```
private String readStream(InputStream in) {
    BufferedReader reader = null;
    StringBuilder sb = new StringBuilder();
    try {
        reader = new BufferedReader(new InputStreamReader(in)); ❶
        String line;
        while ((line = reader.readLine()) != null) { ❷
            sb.append(line + "\n"); ❸
        }
```

```
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (reader != null) {
            try {
                reader.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
    return sb.toString(); ❹
}
```

โค้ดส่วนใหญ่เป็นการดักจับข้อผิดพลาดที่อาจเกิดขึ้นในช่วง runtime ซึ่งถ้าตัดโค้ดพวกนี้ออกไปก็จะเหลือโค้ดที่ใช้อ่านข้อมูลจาก `InputStream` จริงๆแค่ไม่กี่บรรทัด

การอ่านข้อมูลจาก `InputStream` เริ่มจากให้สร้าง `InputStreamReader` ขึ้นมาครอบ `InputStream` และสร้าง `BufferedReader` ขึ้นมาครอบ `InputStreamReader` อีกที ❶ แล้วเรียกใช้เมธอด `readLine` เพื่ออ่านข้อมูลแต่ละบรรทัดเข้ามาเรื่อยๆ ถ้าข้อมูลที่อ่านได้เป็น `null` เมื่อไหร่ก็แสดงว่าหมดข้อมูลแล้ว ❷ เราจะรวบรวมข้อมูลที่อ่านได้ทั้งหมดเก็บไว้ใน `StringBuilder` (ตัวแปร `sb`) ❸ แล้ว return ค่าสตริงของ `StringBuilder` นั้นออกไปในตอนท้าย ❹

รันโค้ดที่เชื่อมต่อเน็ตเวิร์กด้วย AsyncTask

ถ้าคุณพยายามอ่านข้อมูลจากเซิร์ฟเวอร์โดยใช้แค่สองเมธอดข้างต้น จะพบว่ามันไม่ work หรือเกิดข้อผิดพลาดในตอนรัน เนื่องจากแอนดรอยด์ตั้งแต่เวอร์ชั่น 3.0 (API Level 11) ขึ้นไปจะไม่อนุญาตให้รันโค้ดที่อาจใช้เวลานาน เช่น การเชื่อมต่อเน็ตเวิร์กจากภายในเธรดหลัก (Main Thread หรือ UI Thread) ของแอพได้ เพราะเธรดหลักมีหน้าที่จัดการ UI ดังนั้นหากมีการทำงานที่ใช้เวลานานจะทำให้ UI ค้าง ซึ่งหากการทำงานนั้นนานเกิน 5 วินาที แอนดรอยด์จะแสดงไดอะล็อก Application Not Responding และให้ผู้ใช้เลือกว่าจะปิดแอพหรือรอต่อไป

NOTE

เธรดหลัก (Main Thread) มีชื่อเรียกอีกอย่างว่า UI Thread เนื่องจากมันมีหน้าที่จัดการกับอีเวนต์ของ UI และอัพเดท UI ซึ่ง UI Thread จะเป็นเพียงเธรดเดียวที่เข้าถึง UI ของแอพได้ ถ้าหากคุณสร้างเธรดใหม่เพื่อรันโค้ดบางอย่างแล้วเข้าถึง UI จากโค้ดส่วนนั้น จะทำให้เกิดข้อผิดพลาดขึ้น

จากเหตุผลข้างต้น ทำให้เราจำเป็นต้องรันโค้ดที่เชื่อมต่อเน็ตเวิร์กในเธรดใหม่ (อาจเรียกว่า Worker Thread หรือ Background Thread) ซึ่งการสร้างเธรดใหม่ไม่ใช่เรื่องยาก เพราะเป็นความสามารถที่มีในจาวาอยู่แล้ว แต่อย่างไรก็ตาม ในตัวอย่างนี้เมื่ออ่านข้อมูลจาก URL มาแล้วจะนำมาแสดงที่ TextView ซึ่งทำให้เกิดปัญหาตามมาอีก เพราะอย่างที่บอกใน Note คือ แอนดรอยด์ไม่อนุญาตให้เราเข้าถึง UI จากเธรดอื่นนอกเหนือจาก UI Thread ได้

สรุปปัญหาคือ

- แอนดรอยด์ตั้งแต่เวอร์ชั่น 3.0 ขึ้นไปไม่อนุญาตให้รันโค้ดที่เข้าถึงเน็ตเวิร์กจาก UI Thread เราต้องสร้างเธรดใหม่เพื่อรันโค้ดนี้
- เมื่อรันโค้ดในเธรดอื่นที่ไม่ใช่ UI Thread เราจะไม่สามารถเข้าถึง UI ได้ เช่น ไม่สามารถใส่ข้อมูลลงใน TextView ได้

แน่นอนว่าแอนดรอยด์มีวิธีให้เรารันโค้ดในเธรดใหม่ แต่ขณะเดียวกันก็สามารถเข้าถึง UI ได้ ซึ่งวิธีหนึ่งที่นักพัฒนานิยมใช้กันมากที่สุด ก็คือการใช้ AsyncTask

AsyncTask คือคลาสในแอนดรอยด์ที่ช่วยให้เรารันโค้ดในเธรดใหม่ และเมื่อโค้ดนั้นรันจบแล้วก็จะเรียกมายังโค้ดอีกส่วนหนึ่งซึ่งรันในเธรดหลักจึงสามารถเข้าถึง UI ได้ การใช้งาน AsyncTask นั้นเราจะต้องสร้างซับคลาสของมัน แล้ว Override เมธอดต่างๆที่ต้องการ ในตัวอย่างนี้คือคลาส DownloadTextTask ที่เราสร้างเป็น Inner Class ของแอคทิวิตี

```
private class DownloadTextTask extends AsyncTask<String, Void, String> {

    @Override
    protected String doInBackground(String... urls) {
        return downloadText(urls[0]); ❶
    }

    @Override
    protected void onPostExecute(String result) {
        TextView textview = (TextView) findViewById(R.id.text);
        textview.setText(result); ❷
    }
}
```

เมธอด doInBackground เอาไว้ระบุโค้ดที่ต้องการให้รันในเธรดใหม่ ในที่นี้คือการเรียกเมธอด downloadText เพื่ออ่านข้อมูลจาก URL แล้ว return ข้อมูลที่อ่านได้ออกไป ❶ ข้อมูลนี้จะถูกส่งเป็นพารามิเตอร์ให้แก่เมธอด onPostExecute ที่จะถูกเรียกให้ทำงานหลังจากเมธอด doInBackground ทำงานจบแล้ว ซึ่งภายในเมธอด onPostExecute เราจะนำข้อมูลนี้มาแสดงที่ TextView ❷ (เมธอด onPostExecute จะถูกรันใน UI Thread ดังนั้นจึงเข้าถึง UI ได้)

หลังจากสร้างคลาส DownloadTextTask ที่เป็นซับคลาสของ AsyncTask โดยระบุโค้ดที่ต้องการเรียบร้อยแล้ว เมื่อต้องการรันโค้ดเหล่านั้นก็ให้สร้างออบเจ็คจากคลาส DownloadTextTask แล้วเรียกเมธอด execute

```
DownloadTextTask task = new DownloadTextTask();
task.execute("http://www.promlert.com");
```

URL ที่เราระบุให้กับเมธอด execute จะถูกส่งเป็นพารามิเตอร์ให้แก่เมธอด doInBackground ใน DownloadTextTask

NOTE

สังเกตว่าเราประกาศพารามิเตอร์ของเมธอด doInBackground เป็นรูปแบบที่เรียกว่า Parameter Array (มีเครื่องหมาย ... ต่อท้ายชื่อชนิดข้อมูล)

```
protected String doInBackground(String... urls) {
```

นั่นหมายความว่า เราสามารถส่งค่าพารามิเตอร์ให้ doInBackground ได้หลายค่าในคราวเดียว เช่น

```
task.execute("http://www.promlert.com", "http://www.provision.co.th")
```

ซึ่งภายใน doInBackground เราจะอ่านค่าพารามิเตอร์มาใช้งานโดยคิดเสมือนว่าตัวแปร urls นั้นเป็นตัวแปรอาร์เรย์ เช่น urls[0] จะหมายถึง "http://www.promlert.com" และ urls[1] จะหมายถึง "http://www.provision.co.th" เป็นต้น

อย่างไรก็ตาม ในตัวอย่างนี้ส่ง URL ไปเพียงค่าเดียว เนื่องจากเราออกแบบให้ DownloadTextTask ดาวน์โหลดข้อมูลจาก URL เดียวเท่านั้น แต่ถึงอย่างนั้นก็จำเป็นต้องประกาศพารามิเตอร์ของ doInBackground เป็น Parameter Array ตามข้อกำหนดของ AsyncTask

รู้จัก AsyncTask ให้มากยิ่งขึ้น

คลาส AsyncTask มีเมธอดหลักอยู่ 4 เมธอดไว้ให้เรา Override เพื่อระบุการทำงานในแต่ละช่วงเวลา ได้แก่ onPreExecute, doInBackground, onProgressUpdate และ onPostExecute นอกจากนี้การสร้างซับคลาสของ AsyncTask จะต้องระบุชนิดข้อมูล (Generic Type) 3 ชุด รูปต่อไปนี้อธิบายความหมายของเมธอดทั้งสี่และชนิดข้อมูลทั้งสาม (ชนิดข้อมูลในรูปเป็นเพียงตัวอย่างเท่านั้น จริงๆสามารถระบุชนิดอะไรก็ได้)

```
class SomeTask extends AsyncTask<String, Integer, Long> {
    @Override
    protected void onPreExecute() {
        รันใน UI Thread และรันก่อน doInBackground
        โดยทั่วไปจะใช้จัดเตรียมสิ่งต่างๆก่อนที่ doInBackground จะทำงาน
        เช่น การแสดง Progress Bar เป็นต้น
    }

    @Override
    protected Long doInBackground(String... urls) {
        รันในเธรดใหม่และรันหลังจาก onPreExecute ทำงานจบแล้ว
        สำหรับระบุการทำงานที่ใช้เวลานาน เมื่อทำงานจบแล้วต้อง return ค่า
        ออกไป และค่านั้นจะถูกส่งไปให้ onPostExecute
        ภายในเมธอดนี้สามารถเรียกเมธอด publishProgress เพื่อแจ้งความ
        คืบหน้าในการทำงานได้ ซึ่งแอนดรอยด์จะเรียก onProgressUpdate
        ให้อีกที
    }

    @Override
    protected void onProgressUpdate(Integer... progress) {
        รันใน UI Thread เมื่อมีการเรียกเมธอด publishProgress
        เมธอดนี้จะใช้แสดงความคืบหน้าในการทำงานของโค้ดใน
        doInBackground
    }

    @Override
    protected void onPostExecute(Long result) {
        รันใน UI Thread หลังจาก doInBackground ทำงานจบแล้ว
        โดยผลลัพธ์ของ doInBackground จะถูกส่งผ่านมาเป็นพารามิเตอร์
        ของเมธอดนี้
    }
}
```

การดาวน์โหลดไฟล์รูปภาพจากเซิร์ฟเวอร์

นอกจากไฟล์ข้อความแล้ว บางครั้งคุณอาจต้องการดาวน์โหลดไฟล์รูปภาพที่ URL หนึ่งๆมาแสดงผลหรือใช้งานภายในแอพของคุณ ซึ่งทำได้ไม่ยากเลย เราจะมาดูรายละเอียดกันในหัวข้อนี้

ตัวอย่างและคำอธิบาย

ตัวอย่างนี้จะอ่านไฟล์รูปภาพจาก `http://promlert.com/wp-content/uploads/2013/04/android_complete.jpg` มาแสดงใน `ImageView` ภายในแอพของเรา ซึ่งหลักการทำงานและโค้ดส่วนใหญ่จะเหมือนตัวอย่างที่แล้ว

1 กำหนด Layout ของหน้าจอ

โปรเจ็ค HttpDownloadImage, ไฟล์ activity__main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp" >

    <Button
        android:id="@+id/download_button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Download Image" />

    <ImageView
        android:id="@+id/image"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginTop="8dp" />

</LinearLayout>
```

2 เพิ่มโค้ดในเมธอด onCreate ของแอคทิวิตี

โปรเจ็ค HttpDownloadImage, ไฟล์ MainActivity.java

```
Button button = (Button) findViewById(R.id.download_button);
button.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {
        DownloadImageTask task = new DownloadImageTask();
        task.execute("http://promlert.com/wp-content/uploads/2013/04/
                    android_complete.jpg");
    }
});
```

3 เพิ่มเมธอด downloadImage ในแอคทิวิตี

โปรเจ็ค HttpDownloadImage, ไฟล์ MainActivity.java

```
private Bitmap downloadImage(String strUrl) {
    Bitmap bmpResult = null;
    try {
        URL url = new URL(strUrl);
        HttpURLConnection con = (HttpURLConnection) url.openConnection();
```

```
            bmpResult = BitmapFactory.decodeStream(con.getInputStream()); ❶
        } catch (Exception e) {
            e.printStackTrace();
        }
        return bmpResult; ❷
    }
```

เราใช้เมธอด `decodeStream` ของคลาส `BitmapFactory` แปลง `InputStream` ที่ได้จากเมธอด `getInputStream` ไปเป็น `Bitmap` ❶ แล้ว return `Bitmap` นี้ออกไป ❷

4 เพิ่มคลาส `DownloadImageTask` ในแอคทิวิตี (คลาส `DownloadImageTask` จะเป็น Inner Class ของแอคทิวิตี)

โปรเจ็ค HttpDownloadImage, ไฟล์ MainActivity.java

```
private class DownloadImageTask extends AsyncTask<String, Void, Bitmap> {

    @Override
    protected Bitmap doInBackground(String... urls) {
        return downloadImage(urls[0]); ❶
    }

    @Override
    protected void onPostExecute(Bitmap result) {
        ImageView imageview = (ImageView) findViewById(R.id.image);
        imageview.setImageBitmap(result); ❷
    }
}
```

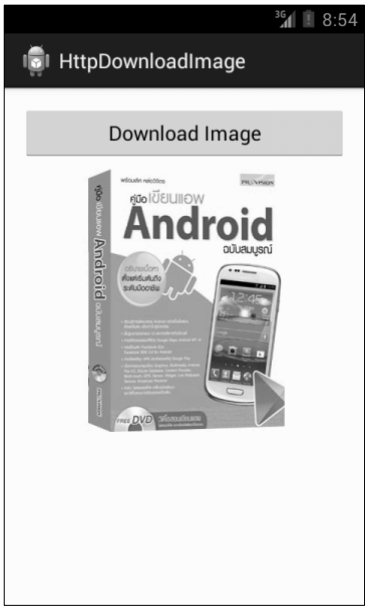
ในเมธอด `onPostExecute` เรานำ `Bitmap` ที่เมธอด `doInBackground` return ออกมา ❶ (ซึ่งเมธอด `downloadImage` return มาให้อีกทีหนึ่ง) มาแสดงผลที่ `ImageView` โดยใช้เมธอด `setImageBitmap` ❷

5 เพิ่มบรรทัดต่อไปนี้ในไฟล์ AndroidManifest.xml โดยใส่ไว้ก่อนแท็ก `<application>` เพื่อขอสิทธิ์ในการเข้าถึงอินเทอร์เน็ต

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

ผลการรัน

เมื่อรันแล้วคลิกปุ่ม Download Image จะปรากฏภาพปกหนังสืออีกเล่มหนึ่งของผู้เขียน ซึ่งไฟล์ภาพนี้เก็บอยู่ในเว็บไซท์ของผู้เขียนเอง (ตาม URL ในโค้ด)



การดาวน์โหลดรูปภาพโดยสร้างซับคลาสของ ImageView

เราสามารถปรับปรุงการดาวน์โหลดรูปภาพในหัวข้อที่ผ่านมาให้ดีขึ้น โดยรวมโค้ดที่ใช้ดาวน์โหลดเข้าไปเป็นส่วนหนึ่งของ `ImageView` เลย หลังจากนั้นเมื่อต้องการดาวน์โหลดรูปภาพใดก็เพียงแต่กำหนด URL ของภาพโดยใช้เมธอดที่เราเตรียมไว้ใน `ImageView` ซึ่งการจะทำเช่นนี้ได้เราต้องสร้างซับคลาสของ `ImageView` ขึ้นมา

ตัวอย่างและคำอธิบาย

1 สร้างคลาสใหม่ชื่อ `WebImageView` แล้วพิมพ์โค้ดดังนี้

โปรเจ็ค WebImageView, ไฟล์ WebImageView.java

```
package com.example.webimageview;

import java.net.HttpURLConnection;
import java.net.URL;

import android.content.Context;
import android.graphics.Bitmap;
```

```
import android.graphics.BitmapFactory;
import android.graphics.drawable.BitmapDrawable;
import android.graphics.drawable.Drawable;
import android.os.AsyncTask;
import android.util.AttributeSet;
import android.widget.ImageView;

public class WebImageView extends ImageView { ❶

    private Drawable mPlaceholder;  // รูปภาพชั่วคราว ก่อนโหลดรูปภาพจาก URL
    private Drawable mImage;        // รูปภาพที่โหลดจาก URL

    public WebImageView(Context context) {
        this(context, null);
    }

    public WebImageView(Context context, AttributeSet attrs) {
        this(context, attrs, 0);
    }

    public WebImageView(Context context, AttributeSet attrs,
                        int defaultStyle) {
        super(context, attrs, defaultStyle);
    }

    public void setPlaceholderImage(Drawable drawable) {
        mPlaceholder = drawable;
        if (mImage == null) {
            setImageDrawable(mPlaceholder);
        }
    }

    public void setPlaceholderImage(int resid) {
        mPlaceholder = getResources().getDrawable(resid);
        if (mImage == null) {
            setImageDrawable(mPlaceholder);
        }
    }

    public void setImageUrl(String url) { ❹
        DownloadTask task = new DownloadTask();
        task.execute(url);
    }
```

❷

❸

```
    private class DownloadTask extends AsyncTask<String, Void, Bitmap> {
        @Override
        protected Bitmap doInBackground(String... urls) {
            Bitmap bmpResult = null;
            String strUrl = urls[0];
            try {
                URL url = new URL(strUrl);
                HttpURLConnection con = (HttpURLConnection)
                                        url.openConnection();
                bmpResult = BitmapFactory.decodeStream(
                                con.getInputStream());
            } catch (Exception e) {
                e.printStackTrace();
            }
            return bmpResult;
        }

        @Override
        protected void onPostExecute(Bitmap result) {
            mImage = new BitmapDrawable(getResources(), result); ❺
            if (mImage != null) {
                setImageDrawable(mImage); ❻
            }
        }
    }
}
```

เรากำหนดคลาส WebImageView เป็นซับคลาสของ ImageView ❶ เพื่อสร้างวิวชนิดใหม่ที่เป็นการนำ ImageView มาเพิ่มเติมความสามารถบางอย่างเข้าไป

WebImageView มีคอนสตรัคเตอร์ 3 แบบเช่นเดียวกับ ImageView และการทำงานของคอนสตรัคเตอร์ทั้งสามจะเรียกไปยังคอนสตรัคเตอร์ของ ImageView อีกทีหนึ่ง ❷

เมธอด setPlaceholderImage ใช้กำหนดรูปภาพชั่วคราวที่ต้องการให้แสดงออกมา ก่อนจะแสดงภาพที่ดาวน์โหลดจาก URL เมธอดนี้สามารถเรียกใช้งานได้ 2 แบบ โดยระบุพารามิเตอร์เป็นออบเจ็ค Drawable หรือ Resource ID ของไฟล์ภาพ ❸

เมธอด setImageUrl ใช้กำหนด URL ของไฟล์ภาพที่จะดาวน์โหลดมาแสดง ❹ เมธอดนี้จะเรียกไปยัง AsyncTask (คลาส DownloadTask) เพื่อดาวน์โหลดไฟล์ภาพในเธรดใหม่ (การทำงานของ doInBackground) เมื่อได้ข้อมูลภาพมาแล้วก็จะเก็บลงฟิลด์ mImage ❺ และแสดงรูปภาพออกมาโดยใช้เมธอด setImageDrawable ของ ImageView ❻

2 กำหนด Layout ของหน้าจอ

โปรเจ็ค WebImageView, ไฟล์ activity__main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp" >

    <com.example.webimageview.WebImageView
        android:id="@+id/image1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <com.example.webimageview.WebImageView
        android:id="@+id/image2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <com.example.webimageview.WebImageView
        android:id="@+id/image3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <com.example.webimageview.WebImageView
        android:id="@+id/image4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</LinearLayout>
```

เนื่องจากคลาส WebImageView ของเราคือวิวชนิดหนึ่ง (เพราะเป็นซับคลาสของ ImageView ซึ่งเป็นซับคลาสของ View อีกที) เราจึงสามารถสร้างมันเป็นอิลิเมนต์ในหน้าจอได้ โดยให้นำชื่อคลาสแบบเต็มมาระบุเป็นชื่อแท็ก (คลาส WebImageView อยู่ในแพ็คเกจ com.example.webimageview ดังนั้นชื่อเต็มของมันคือ com.example.webimageview.WebImageView)

ในที่นี้สร้าง WebImageView ขึ้นมาในหน้าจอจำนวน 4 ออบเจ็ค

3 เพิ่มโค้ดในแอคทิวิตีจนเป็นดังนี้

```
package com.example.webimageview;

import android.app.Activity;
import android.os.Bundle;

public class MainActivity extends Activity {
```

```
    // ID ของ WebImageView ทั้งสี่ใน Layout File
    private int imageId[] = new int[] {
            R.id.image1, R.id.image2, R.id.image3, R.id.image4 };

    // URL ของรูปภาพที่จะดาวน์โหลดมาแสดงใน WebImageView
    private String urls[] = new String[] {
            "http://www.provision.co.th/components/com_virtuemart/
             shop_image/product/_____________517a435824117.jpg",
            "http://www.provision.co.th/components/com_virtuemart/
             shop_image/product/_____________515d4161ac324.jpg",
            "http://www.provision.co.th/components/com_virtuemart/
             shop_image/product/_____________5135fb333a2f9.jpg",
            "http://www.provision.co.th/components/com_virtuemart/
             shop_image/product/_____________5135f653054dd.jpg"
    };

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        for (int i = 0; i < imageId.length; i++) {
            WebImageView image = (WebImageView) findViewById(imageId[i]);
            // กำหนดรูปภาพชั่วคราวโดยใช้ไฟล์รูปภาพ placeholder.png ที่เตรียมไว้ในโปรเจ็ค
            image.setPlaceholderImage(R.drawable.placeholder);
            // ดาวน์โหลดรูปภาพจาก URL
            image.setImageUrl(urls[i]);
        }
    }
}
```

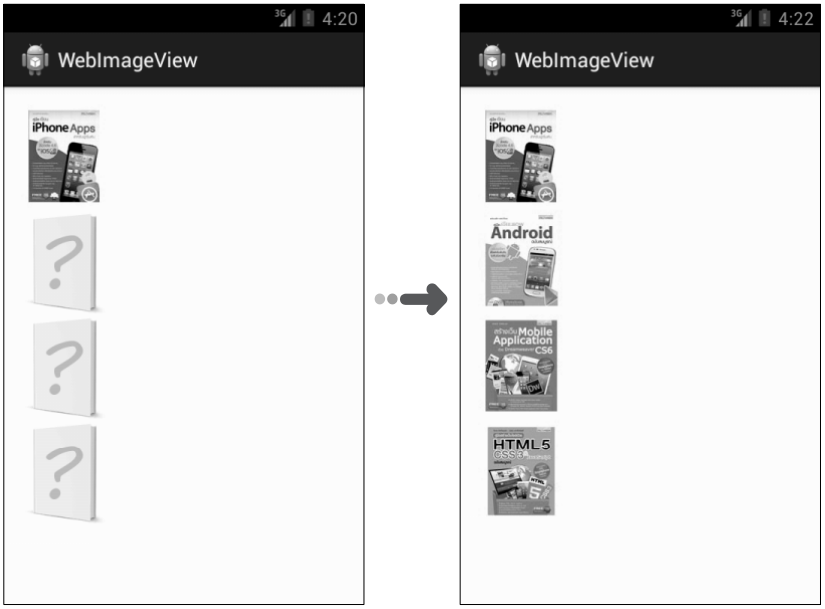
placeholder.png

4 เพิ่มบรรทัดต่อไปนี้ในไฟล์ AndroidManifest.xml โดยใส่ไว้ก่อนแท็ก `<application>` เพื่อขอสิทธิ์ในการเข้าถึงอินเทอร์เน็ต

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

ผลการรัน

เมื่อรันแอพ ตอนแรก `WebImageView` ทั้งสี่จะแสดงภาพชั่วคราว แต่สักครู่ก็จะแสดงภาพปกหนังสือที่ดาวน์โหลดจาก URL ดังรูป



การ Parse ข้อมูล JSON

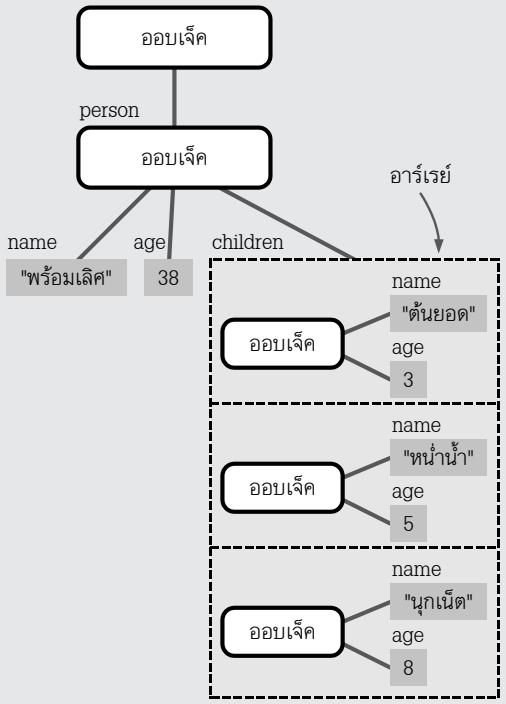
ปัจจุบันมีเว็บเซอร์วิส (Web Services) จำนวนมากที่ให้เราเรียกใช้งานโดยส่ง HTTP Request ไปยัง URL หนึ่ง หลังจากนั้นเว็บเซอร์วิสจะส่งข้อมูลผลลัพธ์ในรูปแบบ JSON (JavaScript Object Notation) กลับมา และเราจะต้องดึงเอาข้อมูลที่ต้องการภายในข้อมูล JSON นั้นมาใช้งานอีกที ซึ่งกระบวนการนี้เรียกว่า Parsing

การ parse ข้อมูลแบบ JSON บนแอนดรอยด์สามารถทำได้ไม่ยาก เนื่องจากแอนดรอยด์จัดเตรียมไลบรารีที่เป็น JSON Parser มาให้แล้ว โดยอยู่ในแพ็คเกจ `org.json`

ตัวอย่างและคำอธิบาย

ใน JSON นั้น เครื่องหมาย `{ }` จะแทนออบเจ็ค และ `[ ]` จะแทนอาร์เรย์ของออบเจ็ค ดังรูป

```
{
   "person" : {
      "name" : "พร้อมเลิศ"
      "age" : 38,
      "children" : [
         {
            "name" : "ต้นยอด",
            "age" : 3
         },
         {
            "name" : "หน่ำน้ำ",
            "age" : 5
         },
         {
            "name" : "นุกเน็ต",
            "age" : 8
         }
      ]
   }
}
```



ตัวอย่างนี้จะแสดงโค้ดที่ใช้ parse ข้อมูล JSON ในรูปข้างต้น

1 ที่แอคทิวิตี ให้เพิ่มการประกาศตัวแปรแบบค่าคงที่ชื่อ `JSON_STRING` ที่ส่วนประกาศของคลาส

โปรเจ็ค ParseJSON, ไฟล์ MainActivity.java

```
private static final String JSON_STRING =
        "{\"person\":{\"name\":\"พร้อมเลิศ\",\"age\":30,\"children\":["
        + "{\"name\":\"ต้นยอด\",\"age\":3},"
        + "{\"name\":\"หน่ำน้ำ\",\"age\":5},"
        + "{\"name\":\"นุกเน็ต\",\"age\":8}"
        + "]}}";
```

เนื่องจากประเด็นในหัวข้อนี้คือการทำ JSON parsing ผู้เขียนจึงนำข้อมูล JSON ที่จะแสดงการ parse ให้ดูเป็นตัวอย่างมากำหนดเป็นค่าคงที่ในคลาสเลย เพื่อเน้นโค้ดที่ใช้ทำ parsing แต่แน่นอนว่าในการพัฒนาแอพจริงๆ ข้อมูลนี้ย่อมมาจากแหล่งข้อมูลภายนอกแอพ เช่น เป็นผลลัพธ์จากการเรียกใช้เว็บเซอร์วิส เป็นต้น (ดูหัวข้อถัดไป)

2 เพิ่มโค้ดในเมธอด onCreate ของแอคทิวิตี (การทำความเข้าใจโค้ดส่วนนี้ ให้ดูรูปแผนผังออบเจ็คในหน้าที่แล้วประกอบด้วย)

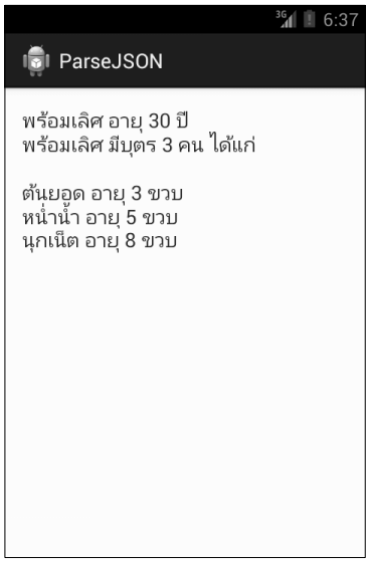
โปรเจ็ค ParseJSON, ไฟล์ MainActivity.java

```
TextView textview = (TextView) findViewById(R.id.text);
String msg = "";

try {
    /* ทำการ parse ข้อมูล JSON ที่เก็บอยู่ในค่าคงที่ JSON_STRING (หลังจาก parse เรียบร้อยแล้ว
       ตัวแปร json จะชี้ไปยังรูทออบเจ็ค หรือออบเจ็คอันบนสุดในรูปแผนผังก่อนหน้านี้) */
    JSONObject json = new JSONObject(JSON_STRING);
    // เข้าถึงออบเจ็ค person
    JSONObject person = json.getJSONObject("person");

    // อ่านพร็อพเพอร์ตี้ name และ age ของออบเจ็ค person
    String personName = person.getString("name");
    int personAge = person.getInt("age");
    msg = String.format("%s อายุ %d ปี\n", personName, personAge);

    // เข้าถึงอาร์เรย์ children
    JSONArray children = person.getJSONArray("children");
    // นับจำนวนออบเจ็คในอาร์เรย์ children
    int childrenCount = children.length();
    msg += String.format("%s มีบุตร %d คน ได้แก่\n\n", personName,
                         childrenCount);

    // วนลูปเพื่อเข้าถึงแต่ละออบเจ็คในอาร์เรย์ children
    for (int i = 0; i < childrenCount; i++) {
        // เข้าถึงออบเจ็คหนึ่งๆในอาร์เรย์ children
        JSONObject child = children.getJSONObject(i);

        // อ่านพร็อพเพอร์ตี้ name และ age ของออบเจ็คหนึ่งๆในอาร์เรย์ children
        String childName = child.getString("name");
        int childAge = child.getInt("age");
        msg += String.format("%s อายุ %d ขวบ\n", childName, childAge);
    }
} catch (JSONException e) {
    e.printStackTrace();
}

textview.setText(msg);
```

จากโค้ด ให้สังเกตว่าออบเจ็คที่ได้จากการ parse ข้อมูล JSON นั้นมีอยู่เพียง 2 ชนิด (คลาส) คือ `JSONObject` และ `JSONArray` โดยข้อมูลต่างๆที่เราดึงออกมาใช้งาน (ชื่อ, อายุ) ล้วนแต่เป็นพร็อพเพอร์ตี้ของออบเจ็คชนิด `JSONObject` ทั้งสิ้น (ลองดูรูปก่อนหน้านี้จะเข้าใจยิ่งขึ้น)

ออบเจ็ค `JSONObject` นอกจากจะมีพร็อพเพอร์ตี้ที่ให้ค่าเป็นชนิดข้อมูลพื้นฐานอย่าง `int` หรือ `String` แล้ว ยังอาจมีพร็อพเพอร์ตี้ที่ให้ค่าเป็นออบเจ็ค `JSONObject` หรือ `JSONArray` ได้ด้วย หรือมองง่ายๆคือมีออบเจ็คย่อยหรืออาร์เรย์ของออบเจ็คซ้อนอยู่ข้างในอีกที ลักษณะนี้ทำให้เกิดโครงสร้างแบบต้นไม้ โดยเริ่มจากออบเจ็คหลักอันหนึ่ง แล้วแตกแขนงไปเป็นออบเจ็คย่อย ซึ่งแต่ละออบเจ็คย่อยก็สามารถมีออบเจ็คย่อยลงไปได้อีก

ผลการรัน



การเรียกใช้เว็บเซอร์วิส

เพื่อให้เห็นตัวอย่างการใช้งานจริง จะขอแสดงการเรียกใช้เว็บเวอร์วิสที่ส่งคืนข้อมูลแบบ JSON กลับมา และการ parse ข้อมูลนั้นมาใช้ในแอพของเรา

เว็บเซอร์วิสที่จะเรียกใช้ในตัวอย่างนี้คือ v2 Forecast API ของเว็บไซท์ `forecast.io` ซึ่งให้ข้อมูลเกี่ยวกับสภาพอากาศ รูปแบบการเรียกใช้คือ

```
https://api.forecast.io/forecast/apikey/latitude,longitude
```

โดย apikey คือคีย์หรือรหัสที่นักพัฒนาแต่ละคนต้องขอจาก forecast.io แล้วนำมาระบุในโค้ด จึงจะสามารถเรียกใช้ v2 Forecast API ได้, latitude และ longitude คือพิกัดบนโลกที่เราต้องการรู้สภาพอากาศ

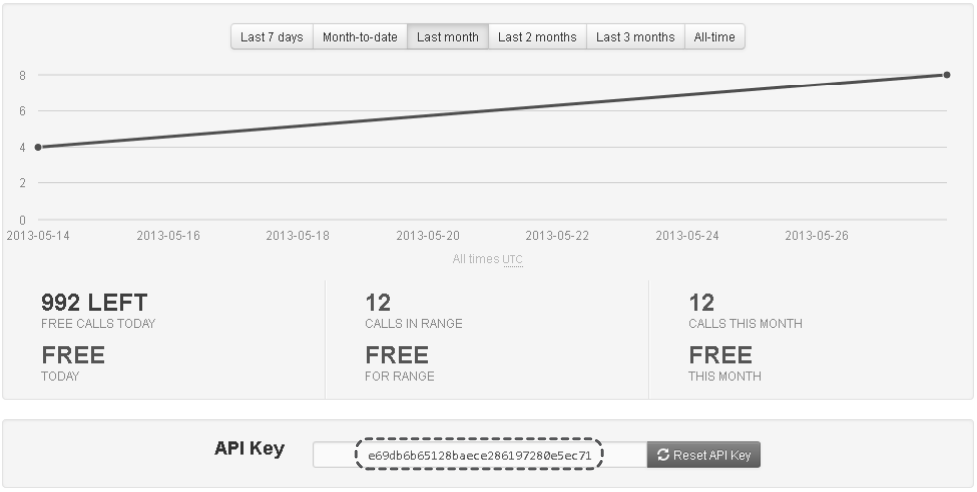
นอกจากนี้หากต้องการข้อมูลในหน่วย SI ให้ระบุ ?units=si ต่อท้าย URL ข้างต้น เช่น ปกติอุณหภูมิจะเป็นหน่วยองศาฟาเรนไฮต์ แต่ถ้าระบุ ?units=si จะได้อุณหภูมิเป็นหน่วยองศาเซลเซียส

ตัวอย่างข้อมูลที่ได้จาก v2 Forecast API (ตัวหนาคือข้อมูลที่เราจะอ่านมาแสดงในแอพ) โดย currently จะให้ข้อมูลสภาพอากาศขณะนั้น, daily จะให้ข้อมูลสภาพอากาศใน 7 วันข้างหน้า


```
{
    "latitude" : 13.75,
    "longitude" : 100.4833,
    "offset" : 7,
    "timezone" : "Asia/Bangkok",

    "currently" : {
        "cloudCover" : 0.87,
        "dewPoint" : 22.960000000000001,
        "humidity" : 0.46999999999999997,
        "icon" : "cloudy",
        "ozone" : 270.51999999999998,
        "precipIntensity" : 0.053999999999999999,
        "precipProbability" : 0.10000000000000001,
        "precipType" : "rain",
        "pressure" : 1006.63,
        "summary" : "Mostly Cloudy",
        "temperature" : 35.960000000000001,
        "time" : 1369724954,
        "visibility" : 9.9900000000000002,
        "windBearing" : 207,
        "windSpeed" : 4.1200000000000001
    },

    "daily" : {
        "data" : [
            {
                "cloudCover" : 0.77000000000000002,
                "dewPoint" : 23.77,
                "humidity" : 0.65000000000000002,
                "icon" : "rain",
                "ozone" : 271.81,
                "precipIntensity" : 0.125,
                "precipIntensityMax" : 0.029000000000000001,
```

```
        "precipIntensityMaxTime" : 1369747029,
        "precipType" : "rain",
        "pressure" : 1007.59,
        "summary" : "Light rain in the evening.",
        "sunriseTime" : 1369695061,
        "sunsetTime" : 1369741353,
        "temperatureMax" : 36,
        "temperatureMaxTime" : 1369720800,
        "temperatureMin" : 28.989999999999998,
        "temperatureMinTime" : 1369695600,
        "time" : 1369674000,
        "visibility" : 9.9900000000000002,
        "windBearing" : 221,
        "windSpeed" : 2.48
    },
    {
        "cloudCover" : 0.68000000000000005,
        "dewPoint" : 23.34,
        "humidity" : 0.66000000000000003,
        "icon" : "rain",
        "ozone" : 270.69999999999999,
        "precipIntensity" : 0.189,
        "precipIntensityMax" : 0.034000000000000002,
        "precipIntensityMaxTime" : 1369786613,
        "precipType" : "rain",
        "pressure" : 1007.76,
        "summary" : "Sprinkling starting in the afternoon.",
        "sunriseTime" : 1369781457,
        "sunsetTime" : 1369827772,
        "temperatureMax" : 37.5,
        "temperatureMaxTime" : 1369810800,
        "temperatureMin" : 25.870000000000001,
        "temperatureMinTime" : 1369782000,
        "time" : 1369760400,
        "windBearing" : 227,
        "windSpeed" : 2.7200000000000002
    },
    {
        "cloudCover" : 0.95999999999999996,
        "dewPoint" : 24.039999999999999,
        "humidity" : 0.68999999999999995,
        ...
```

สามารถลงทะเบียนใช้งาน v2 Forecast API ได้ที่ `https://developer.forecast.io/register` โดยเพียงแค่กรอกอีเมล์และรหัสผ่านก็จะได้ API Key สำหรับนำมาระบุในโค้ดทันที



NOTE

เงื่อนไขการใช้งาน v2 Forecast API

- เรียกใช้ API ได้ฟรีวันละ 1,000 ครั้ง ถ้าเกินจากนั้นคิด 1 เหรียญ ($1) ต่อการเรียก API 10,000 ครั้ง (คุณต้องเข้าไปกรอกข้อมูลการชำระเงินด้วย ถ้าไม่กรอกก็จะไม่เสียเงิน แต่จะเรียกใช้ API ได้ไม่เกินวันละ 1,000 ครั้ง อย่าลืมว่าจำนวนครั้งนี้นับจากเครื่องแอนดรอยด์ทุกเครื่องที่ติดตั้งใช้งานแอพของคุณ)
- ใส่ข้อความเครดิตว่า “Powered by Forecast.io” ในหน้าจอที่นำข้อมูลจาก API มาแสดง และให้ลิงค์ไปยังเว็บไซต์ `forecast.io`

ตัวอย่างและคำอธิบาย

1 เพิ่มเติมโค้ดในแอคทิวิตีจนเป็นดังนี้

โปรเจ็ค WeatherForecast, ไฟล์ MainActivity.java

```
package com.example.weatherforecast;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
```

```
import java.net.URL;
import java.util.Date;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import android.app.Activity;
import android.os.AsyncTask;
import android.os.Bundle;
import android.text.format.DateFormat;
import android.widget.TextView;

public class MainActivity extends Activity {

   static final String API_KEY = "e69db6b65128baece286197280e5ec71";
   // พิกัดของกรุงเทพฯ
   static final String LATITUDE = "13.7500";
   static final String LONGITUDE = "100.4833";

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

      // กำหนด API Key และพิกัดลงใน URL
      String url = String.format(
                     "https://api.forecast.io/forecast/%s/%s,%s?units=si",
                     API_KEY, LATITUDE, LONGITUDE);
      GetWeatherTask task = new GetWeatherTask();
      task.execute(url);
   }

   // ขอข้อมูลสภาพอากาศจาก v2 Forecast API
   private String getWeatherData(String strUrl) {
      String strResult = "";
      try {
         URL url = new URL(strUrl);
         HttpURLConnection con = (HttpURLConnection) url.openConnection();
         strResult = readStream(con.getInputStream());
      } catch (Exception e) {
         e.printStackTrace();
      }
      return strResult;
```

```
}

// อ่านข้อมูลจาก InputStream
private String readStream(InputStream in) {
   BufferedReader reader = null;
   StringBuilder sb = new StringBuilder();
   try {
      reader = new BufferedReader(new InputStreamReader(in));
      String line;
      while ((line = reader.readLine()) != null) {
         sb.append(line + "\n");
      }
   } catch (IOException e) {
      e.printStackTrace();
   } finally {
      if (reader != null) {
         try {
            reader.close();
         } catch (IOException e) {
            e.printStackTrace();
         }
      }
   }
   return sb.toString();
}

private class GetWeatherTask extends AsyncTask<String, Void, String> {

   @Override
   protected String doInBackground(String... urls) {
      return getWeatherData(urls[0]);
   }

   @Override
   protected void onPostExecute(String jsonString) {
      String msg = "";

      try {
         JSONObject json = new JSONObject(jsonString);

         msg += String.format("Latitude: %s, Longitude: %s\n",
                              json.getString("latitude"),
                              json.getString("longitude"));
         msg += String.format("Timezone: %s\n\n",
```

```
                                json.getString("timezone"));

            JSONObject currentWeather = json.getJSONObject("currently");
            msg += "***** Current Weather *****\n";
            msg += String.format("Temperature: %.1f\n",
                                currentWeather.getDouble("temperature"));
            msg += String.format("Icon: %s\n",
                                currentWeather.getString("icon"));
            msg += String.format("Summary: %s\n\n",
                                currentWeather.getString("summary"));

            JSONArray dailyWeather = json.getJSONObject("daily")
                                    .getJSONArray("data");
            msg += "***** Next 7 Days *****\n";
            for (int i = 0; i < dailyWeather.length(); i++) {
               JSONObject item = dailyWeather.getJSONObject(i);

               /* แปลงข้อมูล Timestamp ที่ได้จาก v2 Forecast API ไปเป็นออบเจ็ค Date
                  (สาเหตุที่ต้องคูณ 1000 เพราะ Timestamp ของ API เป็นหน่วยวินาที แต่การ
                  สร้างออบเจ็ค Date ต้องระบุค่า Timestamp เป็นหน่วยมิลลิวินาที */
               Long timestamp = item.getLong("time") * 1000;
               Date date = new Date(timestamp);
               String dateString = DateFormat.format("yyyy-MM-dd hh:mm:ss",
                                                date).toString();

               msg += String.format("Date: %s\n", dateString);
               msg += String.format("Minimum Temperature: %.1f\n",
                                    item.getDouble("temperatureMin"));
               msg += String.format("Maximum Temperature: %.1f\n",
                                    item.getDouble("temperatureMax"));
               msg += String.format("Icon: %s\n", item.getString("icon"));
               msg += String.format("Summary: %s\n\n",
                                    item.getString("summary"));
            }
         } catch (JSONException e) {
            e.printStackTrace();
         }

         TextView textview = (TextView) findViewById(R.id.text);
         textview.setText(msg);
      }
   }
}
```

2 เพิ่มบรรทัดต่อไปนี้ในไฟล์ AndroidManifest.xml โดยใส่ไว้ก่อนแท็ก `<application>` เพื่อขอสิทธิ์ในการเข้าถึงอินเทอร์เน็ต

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

ผลการรัน

การ Parse ข้อมูล XML ด้วย SAX

นอกจาก JSON แล้ว ข้อมูลที่ส่งผ่านระหว่างเว็บเซอร์วิสกับแอพยังมีอีกรูปแบบหนึ่งที่ใช้กันบ่อยๆ นั่นก็คือ XML (Extensible Markup Language)

การ parse ข้อมูล XML ในแอนดรอยด์สามารถทำได้ 3 วิธี คือการใช้ DOM (Document Object Model), SAX (Simple API for XML) และ XmlPull แต่วิธี DOM ไม่เป็นที่นิยม เนื่องจาก Parser ต้องอ่านข้อมูล XML มาทั้งหมดก่อนจึงจะเริ่ม parse ได้ ดังนั้นหากข้อมูลมีขนาดใหญ่จะใช้หน่วยความจำ (RAM) มาก ไม่เหมาะกับอุปกรณ์พกพาที่มักมีหน่วยความจำจำกัด

วิธีที่ดีกว่าคือการทยอย parse ข้อมูลไปเรื่อยๆระหว่างที่อ่านข้อมูลเข้ามา ซึ่งนอกจากไม่ต้องจองหน่วยความจำสำหรับเก็บข้อมูลทั้งหมดแล้วก็ยังไม่ต้องเสียเวลารอข้อมูลมาครบก่อน จึงมีประสิทธิภาพดีกว่า DOM มาก การ parse วิธีนี้ยังแบ่งย่อยได้เป็น 2 รูปแบบ คือ Push Parsing กับ Pull Parsing

- **Push Parsing** รูปแบบนี้ Parser จะเรียก (callback) มายังเมธอดที่เราเตรียมไว้ เมื่อมันพบแท็กต่างๆในข้อมูล XML หรือกล่าวได้ว่า Parser ผลัก (push) ส่วนต่างๆในข้อมูลมาให้เรา

- **Pull Parsing** รูปแบบนี้เราจะอ่านหรือดึง (pull) ส่วนต่างๆในข้อมูล XML เข้ามาเองเมื่อต้องการ

การทำ Push Parsing ในแอนดรอยด์จะใช้ไลบรารี SAX (แพคเกจ `org.xml.sax`) ส่วน Pull Parsing จะใช้ไลบรารี XmlPull (แพคเกจ `org.xmlpull.v1`) ในหัวข้อนี้จะแสดงการใช้ SAX

ตัวอย่างและคำอธิบาย

ภาษา XML จะใช้แท็ก (Tag) ในการกำหนดความหมายของข้อมูลแบบเดียวกับ HTML เช่น

```
<?xml version="1.0" encoding="utf-8" ?>
<person>
    <name>พร้อมเลิศ</name>
    <age>38</age>
    <child>
        <name>ต้นยอด</name>
        <age>3</age>
    </child>
    <child>
        <name>หน่ำน้ำ</name>
        <age>5</age>
    </child>
    <child>
        <name>นุกเน็ต</name>
        <age>8</age>
    </child>
</person>
```

ตัวอย่างนี้จะแสดงการ parse ข้อมูล XML ข้างต้น

1 สร้างคลาสใหม่ชื่อ `MyHandler` แล้วพิมพ์โค้ด ดังนี้

โปรเจ็ค ParseXML_SAX, ไฟล์ MyHandler.java

```
package com.example.parsexml_sax;

import java.util.ArrayList;

import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;

public class MyHandler extends DefaultHandler {

    public class Person { ❶
        public String name;
        public int age;
```

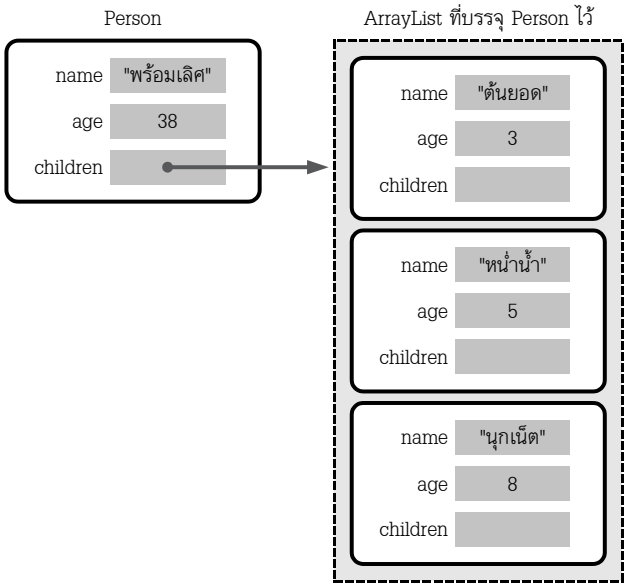
```
    public ArrayList<Person> children; ❷
}

private StringBuffer buf; // ข้อมูลในอิลิเมนต์
private Person father;    // แทนอิลิเมนต์ person
private Person child;     /* แทนอิลิเมนต์ child หนึ่งๆ (เป็นตัวแปรชั่วคราว โดยหลังจาก
                             กำหนดข้อมูลต่างๆในออบเจ็คเรียบร้อยแล้ว เราจะใส่ออบเจ็ค
                             ลงใน ArrayList แล้วนำตัวแปร child นี้กลับมาใช้ซ้ำ
                             สำหรับอิลิเมนต์ child ถัดไป) */
private boolean inPerson = false; // กำลัง parse อยู่ภายในอิลิเมนต์ person หรือไม่
private boolean inChild = false;  // กำลัง parse อยู่ภายในอิลิเมนต์ child หรือไม่

// เมธอดนี้จะถูกเรียกเมื่อพบแท็กเปิดของอิลิเมนต์
@Override
public void startElement(String uri, String localName, String qName,
                         Attributes attributes) throws SAXException {
    if (localName.equals("person")) {                          ❸
        father = new Person();
        father.children = new ArrayList<Person>();
        inPerson = true;
    } else if (localName.equals("child")) {                    ❹
        child = new Person();
        inChild = true;
    } else if (localName.equals("name") ||                     ❺
               localName.equals("age")) {
        buf = new StringBuffer();
    }
}

// เมธอดนี้จะถูกเรียกเมื่อพบข้อมูลในอิลิเมนต์
@Override
public void characters(char[] ch, int start, int length)      ❻
                       throws SAXException {
    if (buf != null) {
        for (int i = start; i < start + length; i++) {
            buf.append(ch[i]);
        }
    }
}

// เมธอดนี้จะถูกเรียกเมื่อพบแท็กปิดของอิลิเมนต์
@Override
```

```
    public void endElement(String uri, String localName, String qName)
                        throws SAXException {
        if (localName.equals("person")) {
            inPerson = false;
        } else if (localName.equals("child")) {
            father.children.add(child);
            inChild = false;
        } else if (localName.equals("name") &&
                (inPerson == true) && (inChild == false)) {
            father.name = buf.toString();
        } else if (localName.equals("age") &&
                (inPerson == true) && (inChild == false)) {
            father.age = Integer.valueOf(buf.toString());
        } else if (localName.equals("name") &&
                (inPerson == true) && (inChild == true)) {
            child.name = buf.toString();
        } else if (localName.equals("age") &&
                (inPerson == true) && (inChild == true)) {
            child.age = Integer.valueOf(buf.toString());
        }

        buf = null;
    }

    // เมธอดนี้จะ return ออบเจ็ค Person ที่เป็นออบเจ็คหลักออกไป
    public Person getParsedPerson() {
        return father;
    }
}
```

❾ ❽ ❼

เมื่อ SAX อ่านสตรีมข้อมูล XML เข้ามาแล้วพบจุดเริ่มต้น, จุดสิ้นสุด หรือข้อมูลในอิลิเมนต์ต่างๆ มันจะเรียกมายังเมธอดใน Handler ที่เราเตรียมไว้ (คลาส `MyHandler` ข้างต้น) ซึ่งเราต้องวางแผนเองทั้งหมดว่าจะทำอะไร

ในที่นี้เราจะสร้างออบเจ็คโมเดล (Object Model) ที่สอดคล้องกับโครงสร้างอิลิเมนต์ในข้อมูล XML นั้น ดังรูปถัดไป ออบเจ็คโมเดลของเราจะมี `Person` เป็นออบเจ็คหลัก (เป็นตัวแทนของพ่อ) ออบเจ็คนี้สร้างจากคลาส `Person` ที่เรากำหนดไว้ภายในคลาส `MyHandler` อีกที ❶ ซึ่งจะเก็บชื่อและอายุของพ่อ และข้อมูลเกี่ยวกับบุตรทั้งหมด โดยบุตรแต่ละคนจะถูกแทนด้วยออบเจ็ค `Person` เช่นเดียวกัน และจะถูกเก็บลงใน `ArrayList` ซึ่งเป็นฟิลด์หนึ่งของคลาส `Person` ❷ (ออบเจ็ค `Person` ที่แทนบุตรแต่ละคนก็มีฟิลด์นี้เช่นกัน แต่เราไม่ได้สนใจในตัวอย่างนี้)



หลักการทำงานของคลาส MyHandler

- เมื่อพบจุดเริ่มต้น (แท็กเปิด) ของอิลิเมนต์ person เราจะสร้างออบเจ็ค Person และสร้าง ArrayList สำหรับเก็บข้อมูลเกี่ยวกับบุตรขึ้นภายในออบเจ็ค Person นั้น (เริ่มต้นจะเป็น ArrayList ว่างๆ) ❸
- เมื่อพบจุดเริ่มต้นของอิลิเมนต์ child เราจะสร้างออบเจ็ค Person เพื่อเตรียมเก็บข้อมูลของอิลิเมนต์ child นั้น ❹
- เมื่อพบจุดเริ่มต้นของอิลิเมนต์ name หรือ age เราจะสร้าง StringBuffer เพื่อเตรียมเก็บข้อมูลในอิลิเมนต์ ❺
- เมื่อพบข้อมูลในอิลิเมนต์ เราจะเก็บข้อมูลลงใน StringBuffer ที่สร้างเตรียมไว้ ❻
- เมื่อพบจุดสิ้นสุด (แท็กปิด) ของอิลิเมนต์ name หรือ age ซึ่งอยู่ภายในอิลิเมนต์ child เราจะนำข้อมูลจาก StringBuffer ไปเก็บลงออบเจ็ค Person ที่แทนบุตรคนหนึ่งๆ ❼
- เมื่อพบจุดสิ้นสุดของอิลิเมนต์ name หรือ age ซึ่งอยู่ภายในอิลิเมนต์ person เราจะนำข้อมูลจาก StringBuffer ไปเก็บลงออบเจ็ค Person ที่เป็นออบเจ็คหลัก ❽
- เมื่อพบจุดสิ้นสุดของอิลิเมนต์ child เราจะเพิ่มออบเจ็ค Person ที่แทนบุตรคนหนึ่งลงใน ArrayList ของออบเจ็ค Person ที่เป็นออบเจ็คหลัก ❾

2 ที่แอคทิวิตี ให้เพิ่มการประกาศตัวแปรแบบค่าคงที่ชื่อ XML_STRING ที่ส่วนประกาศของคลาส

โปรเจ็ค ParseXML__SAX, ไฟล์ MainActivity.java

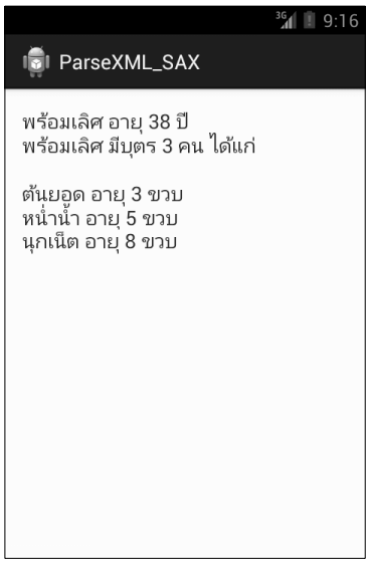
```
private static final String XML_STRING =
        "<?xml version=\"1.0\" encoding=\"utf-8\" ?>"
        + "<person><name>พร้อมเลิศ</name><age>38</age>"
        + "<child><name>ต้นยอด</name><age>3</age></child>"
        + "<child><name>หน่ำน้ำ</name><age>5</age></child>"
        + "<child><name>นุกเน็ต</name><age>8</age></child>"
        + "</person>";
```

3 เพิ่มโค้ดในเมธอด onCreate ของแอคทิวิตี

โปรเจ็ค ParseXML__SAX, ไฟล์ MainActivity.java

```
TextView textview = (TextView) findViewById(R.id.text);
Person person = null;
String msg = "";

try {
    SAXParserFactory factory = SAXParserFactory.newInstance();
    SAXParser parser = factory.newSAXParser();
    MyHandler handler = new MyHandler();

    // parse ข้อมูล XML ในสตริง XML_STRING โดยใช้ Handler ที่เราเขียนขึ้น
    parser.parse(new InputSource(new StringReader(XML_STRING)), handler);
    // เข้าถึงออบเจ็ค Person ใน Handler ซึ่งออบเจ็คนี้จะเก็บข้อมูลทั้งหมดที่ parse แล้วเอาไว้
    person = handler.getParsedPerson();
} catch (Exception e) {
    e.printStackTrace();
}

// อ่านข้อมูลจากออบเจ็ค Person มาแสดงผลใน TextView
if (person != null) {
    msg = String.format("%s อายุ %d ปี\n", person.name, person.age);
    int childrenCount = person.children.size();
    msg += String.format("%s มีบุตร %d คน ได้แก่\n\n", person.name,
                                                    childrenCount);

    for (int i = 0; i < childrenCount; i++) {
        Person child = person.children.get(i);
        msg += String.format("%s อายุ %d ขวบ\n", child.name, child.age);
    }

    textview.setText(msg);
}
```

ผลการรัน



การ Parse ข้อมูล XML ด้วย XmlPull

ในหัวข้อที่แล้วได้อธิบายการ parse ข้อมูล XML แบบ Push Parsing ไปแล้ว สำหรับหัวข้อนี้จะเป็นการ parse แบบ Pull Parsing โดยใช้ XmlPull ที่รวมอยู่ในเฟรมเวิร์คของแอนดรอยด์

ตัวอย่าง

ตัวอย่างนี้จะแสดงการ parse ข้อมูล XML ชุดเดียวกับตัวอย่างที่แล้ว

1 ที่แอคทิวิตี ให้เพิ่มการประกาศตัวแปรแบบค่าคงที่ชื่อ XML_STRING ที่ส่วนประกาศของคลาส

โปรเจ็ค ParseXML__Pull, ไฟล์ MainActivity.java

```
private static final String XML_STRING =
        "<?xml version=\"1.0\" encoding=\"utf-8\" ?>"
        + "<person><name>พร้อมเลิศ</name><age>38</age>"
        + "<child><name>ต้นยอด</name><age>3</age></child>"
        + "<child><name>หน่ำน้ำ</name><age>5</age></child>"
        + "<child><name>นุกเน็ต</name><age>8</age></child>"
        + "</person>";
```

2 เพิ่มโค้ดในเมธอด onCreate ของแอคทิวิตี

โปรเจ็ค ParseXML__Pull, ไฟล์ MainActivity.java

```
TextView textview = (TextView) findViewById(R.id.text);
Person person = null;
String msg = "";

try {
    XmlPullParserFactory factory = XmlPullParserFactory.newInstance(); ❶
    XmlPullParser parser = factory.newPullParser(); ❷
    parser.setInput(new StringReader(XML_STRING)); ❸

    /* เริ่ม parse โดยเรียกเมธอด parsePersonElement ที่เราเตรียมไว้ในคลาส MyPaserEngine
       พร้อมทั้งส่งออบเจ็ค XmlPullParser ไปเป็นพารามิเตอร์ */
    person = MyParserEngine.parsePersonElement(parser);
} catch (XmlPullParserException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
}

// อ่านข้อมูลจากออบเจ็ค Person มาแสดงผลใน TextView
if (person != null) {
    msg = String.format("%s อายุ %d ปี\n", person.name, person.age);
    int childrenCount = person.children.size();
    msg += String.format("%s มีบุตร %d คน ได้แก่\n\n", person.name,
                                                    childrenCount);

    for (int i = 0; i < childrenCount; i++) {
        Person child = person.children.get(i);
        msg += String.format("%s อายุ %d ขวบ\n", child.name, child.age);
    }

    textview.setText(msg);
}
```

การ parse ข้อมูล XML ด้วย XmlPull จะเริ่มจากการสร้าง Factory Object ❶ จากนั้นสร้างตัว Parser (ออบเจ็ค `XmlPullParser`) ❷ และกำหนดข้อมูลที่จะ parse ❸ แล้วจึงทำการ parse โดยเรียกเมธอด `next` ของ Parser ไปเรื่อยๆ แต่เพื่อไม่ให้โค้ดในเมธอด `onCreate` ยาวเกินไป ผู้เขียนจึงแยกโค้ดส่วนนี้ไปสร้างเป็นเมธอด `parsePersonElement` ในคลาส `MyParserEngine` ซึ่งเมธอดนี้จะรับ Parser เข้าไปเป็นพารามิเตอร์

3 สร้างคลาสใหม่ชื่อ MyParseEngine แล้วพิมพ์โค้ดดังนี้

โปรเจ็ค ParseXML__Pull, ไฟล์ MyParserEngine.java

```
package com.example.parsexml_pull;

import java.io.IOException;
import java.util.ArrayList;

import org.xmlpull.v1.XmlPullParser;
import org.xmlpull.v1.XmlPullParserException;

public class MyParserEngine {

    public static class Person {
        public String name;
        public int age;
        public ArrayList<Person> children;
    }

    // เมธอดสำหรับ parse อิลิเมนต์ person
    public static Person parsePersonElement(XmlPullParser parser)
                         throws XmlPullParserException, IOException {
        Person father = null;

        // วนลูปจนกว่าจะพบแท็กปิด </person>
        while (parser.next() != XmlPullParser.END_TAG) { ❶
            String tagName = parser.getName();

            if (tagName.equals("person")) {
                father = new Person();
                father.children = new ArrayList<Person>();
            } else if (tagName.equals("name")) {
                father.name = readContent(parser);
            } else if (tagName.equals("age")) {
                father.age = Integer.valueOf(readContent(parser));
            } else if (tagName.equals("child")) {
                Person child = parseChildElement(parser);
                father.children.add(child);
            }
        }

        return father;
    }
```

❷ ❸

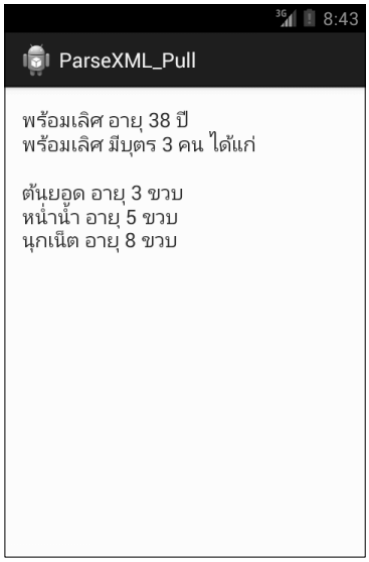
```
    // เมธอดสำหรับ parse อิลิเมนต์ child
    private static Person parseChildElement(XmlPullParser parser)
                          throws XmlPullParserException, IOException {
        Person child = new Person();

        // วนลูปจนกว่าจะพบแท็กปิด </child>
        while (parser.next() != XmlPullParser.END_TAG) {
            String tagName = parser.getName();

            if (tagName.equals("name")) {
                child.name = readContent(parser);
            } else if (tagName.equals("age")) {
                child.age = Integer.valueOf(readContent(parser));
            }
        }

        return child;
    }

    // เมธอดสำหรับอ่านข้อมูลในอิลิเมนต์
    private static String readContent(XmlPullParser parser)
                          throws XmlPullParserException, IOException {
        String result = "";
        if (parser.next() == XmlPullParser.TEXT) {
            result = parser.getText();
            parser.next();  // ข้ามแท็กปิดของอิลิเมนต์นั้นๆไป
        }
        return result;
    }
}
```

คลาส `MyParserEngine` ข้างต้นมีบทบาทแตกต่างจากคลาส `MyHandler` ในตัวอย่างที่แล้ว เราไม่ได้เตรียมคลาสนี้ไว้สร้างออบเจ็ค แต่เอาไว้รวบรวมเมธอดที่ใช้ parse อิลิเมนต์ต่างๆในข้อมูล XML (สังเกตว่าเมธอดในคลาสนี้ถูกประกาศเป็น `static` ทั้งหมด ทำให้เรียกใช้ได้โดยไม่ต้องสร้างออบเจ็คขึ้นมาก่อน)

จากที่อธิบายก่อนหน้านี้ เราจะเรียกมายังเมธอด `parsePersonElement` ในคลาสนี้จากเมธอด `onCreate` ในแอคทิวิตี โดยส่งผ่านตัว Parser มาให้ด้วย สิ่งที่เราทำในเมธอด `parsePersonElement` ก็คือการเรียกเมธอด `next` ของ Parser ไปเรื่อยๆจนกว่าจะพบแท็กปิด `</person>` จึงจะออกจากลูป ❶

ภายในลูปจะตรวจสอบว่าพบแท็กเปิดของอิลิเมนต์ใด แล้วจัดเตรียมหรือใส่ข้อมูลลงในออบเจ็คโมเดลอย่างเหมาะสม โดยหากพบแท็กเปิด <name> หรือ <age> จะเรียกเมธอด readContent เพื่ออ่านข้อมูลของอิลิเมนต์มาเก็บในออบเจ็คโมเดล ❷ แต่หากพบแท็กเปิด <child> จะเรียกเมธอด parseChildElement พร้อมทั้งส่ง Parser ไปให้เพื่อ parse อิลิเมนต์ child ต่อไป ❸ ซึ่งโค้ดในเมธอด parseChildElement จะเป็นรูปแบบ (pattern) เดียวกันกับเมธอด parsePersonElement

ผลการรัน



อธิบายเพิ่มเติม: การทำงานของ XmlPull

XmlPull เป็นวิธีที่เอกสารของแอนดรอยด์แนะนำให้ใช้ในการ parse ข้อมูล XML ด้วยเหตุผลหนึ่งก็คือ รูปแบบ Pull Parsing นั้นเขียนโค้ดง่ายกว่า Push Parsing ถ้าคุณยังจำได้ กรณีของ Push Parsing เราต้องมีตัวแปรชนิด boolean ที่คอยจำว่าขณะนั้น parse อยู่ที่ส่วนใดของข้อมูล XML เนื่องจากมีอิลิเมนต์ที่มีชื่อเดียวกันแต่อยู่คนละระดับกัน (อิลิเมนต์ <name> และ <age>) ในขณะที่ Pull Parsing เราไม่ต้องมีตัวแปรที่ว่านี้ เพราะเราเป็นคนควบคุมการ parse เอง

NOTE

การทำงานของตัวแอนดรอยด์เองก็ใช้ XmlPull ในการ inflate Layout File ไปเป็นโค้ดจาวา ดังนั้นหากทีมผู้พัฒนาแอนดรอยด์คิดจะปรับปรุง XML Parser ให้มีประสิทธิภาพสูงขึ้นหรือมีฟีเจอร์มากขึ้นในอนาคต แน่นอนว่า XmlPull คือตัวเลือกอันดับแรก (และอาจเป็นตัวเลือกเดียว) ที่จะได้รับการปรับปรุง

การ parse ด้วย XmlPull ให้คิดง่ายๆว่าเป็นการขยับตัวชี้ไปยังส่วนต่างๆในข้อมูล XML ซึ่ง XmlPull เรียกข้อมูลแต่ละส่วนนี้ว่า อีเวนต์ (Event) เราสามารถตรวจสอบประเภทของอีเวนต์ได้โดยใช้เมธอด `getEventType` ซึ่งจะให้ผลลัพธ์เป็นค่าใดค่าหนึ่งในบรรดาค่าคงที่ต่อไปนี้ (ค่าคงที่เหล่านี้ถูกกำหนดไว้ในคลาส `XmlPullParser`)

START_DOCUMENT	ขณะนั้นตัวชี้อยู่ที่จุดเริ่มต้นของข้อมูล XML
START_TAG	ขณะนั้นตัวชี้อยู่ที่แท็กเปิดของอิลิเมนต์หนึ่งๆ
TEXT	ขณะนั้นตัวชี้อยู่ที่ข้อมูลในอิลิเมนต์หนึ่งๆ (ข้อความระหว่างแท็กเปิดกับแท็กปิด)
END_TAG	ขณะนั้นตัวชี้อยู่ที่แท็กปิดของอิลิเมนต์หนึ่งๆ
END_DOCUMENT	ขณะนั้นตัวชี้อยู่จุดสิ้นสุดของข้อมูล XML

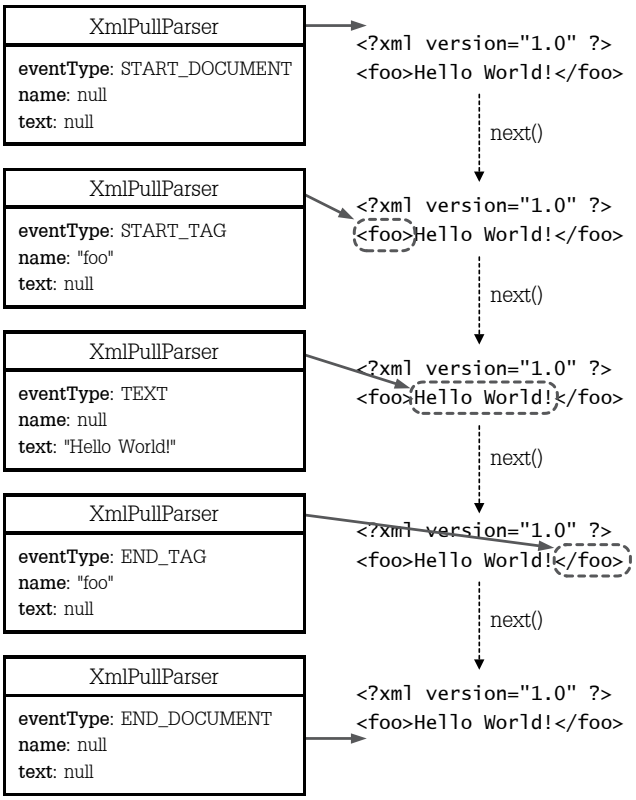
เมธอดอื่นๆที่สำคัญของคลาส `XmlPullParser`

- เมื่อตัวชี้อยู่ที่แท็กเปิดหรือแท็กปิด เราสามารถอ่านชื่อแท็กได้โดยใช้เมธอด **`getName`**
- เมื่อตัวชี้อยู่ที่แท็กเปิด เราสามารถอ่านค่าแอตทริบิวต์ภายในแท็กได้โดยใช้เมธอด **`getAttributeValue`** (ไม่มีในตัวอย่างนี้)
- เมื่อตัวชี้อยู่ที่ข้อมูลในอิลิเมนต์ เราสามารถอ่านข้อมูลนี้ได้โดยใช้เมธอด **`getText`**
- เมธอด **`next`** จะขยับตัวชี้ไปยังอีเวนต์ถัดไป และ return ประเภทของอีเวนต์นั้นกลับมาให้ด้วย (ทำให้ไม่ต้องเรียกเมธอด `getEventType` อีก กรณีที่คุณอยากรู้ว่าอีเวนต์ถัดไปคือประเภทใด)

เพื่อให้เห็นภาพชัดเจนยิ่งขึ้น สมมติว่าเรามีข้อมูล XML ดังนี้

```
<?xml version="1.0" ?>
<foo>Hello World!</foo>
```

รูปต่อไปนี้แสดงตำแหน่งตัวชี้และข้อมูลต่างๆในตัว Parser เมื่อเราเรียกเมธอด `next` เพื่อขยับตัวชี้ไปยังอีเวนต์ต่างๆในข้อมูล XML ข้างต้น (ตอนเริ่มต้นตัวชี้อยู่ที่ตำแหน่ง `START_DOCUMENT`)



การอ่าน RSS Feed

เราได้เรียนรู้วิธีการส่ง HTTP Request เพื่อขอข้อมูลจาก URL และวิธีการ parse ข้อมูล XML ไปแล้ว ดังนั้นคงจะน่าสนใจไม่น้อยถ้าเรานำสองอย่างนี้มารวมกัน

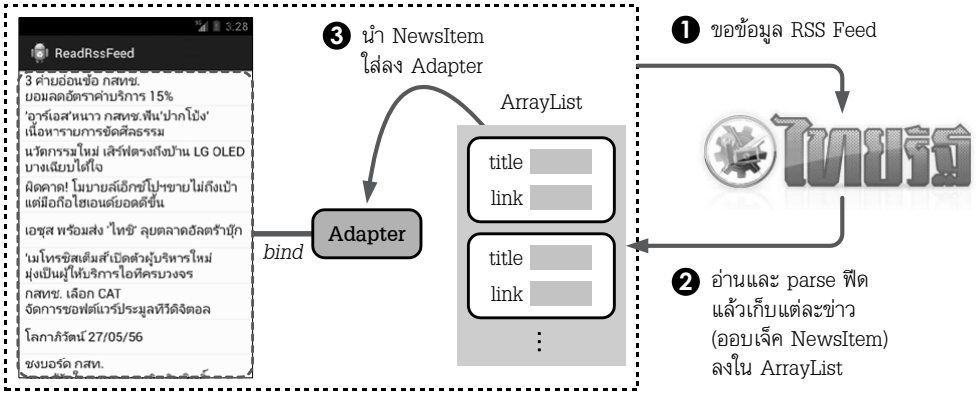
เว็บไซท์ที่เสนอข่าว บทความ หรือบล็อก มักจะจัดเตรียมข้อมูลในรูปแบบ RSS Feed ไว้ให้ผู้ใช้ติดตามข่าวสารล่าสุดผ่านทางโปรแกรมหรือแอพที่เป็น RSS Reader ได้ด้วย ซึ่ง RSS Feed นั้นก็คือข้อมูล XML ในหัวข้อนี้จึงขอแสดงตัวอย่างการอ่าน RSS Feed จากอินเทอร์เน็ตมาแสดงในแอพของเรา

ตัวอย่างและคำอธิบาย

ตัวอย่างนี้จะอ่าน RSS Feed จาก http://www.thairath.co.th/rss/tech.xml ซึ่งเป็นฟีดข่าววิทยาการจากเว็บไซท์ไทยรัฐออนไลน์ โดยรูปแบบข้อมูลเป็นดังนี้

ส่วนที่เราสนใจใน RSS Feed นี้ก็คืออิลิเมนต์ `item` ที่ให้ข้อมูลเกี่ยวกับข่าวหนึ่งๆ ซึ่งเราจะ parse เอาเฉพาะหัวข้อข่าว (อิลิเมนต์ `title`) และลิงค์ที่เชื่อมโยงไปยังเนื้อหาข่าวเต็มๆ (อิลิเมนต์ `link`) มาเก็บลงในออบเจ็คโมเดลของเรา แล้วแสดงหัวข้อข่าวออกมาใน `ListView`

ภาพรวมการทำงานของตัวอย่างนี้จะเป็นดังรูปหน้าถัดไป



1 แก้ไข/เพิ่มเติมโค้ดในแอคทิวิตีจนเป็นดังนี้

โปรเจ็ค ReadRssFeed, ไฟล์ MainActivity.java

```
package com.example.readrssfeed;

import java.io.IOException;
import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.ArrayList;

import org.xmlpull.v1.XmlPullParserException;

import android.app.ListActivity;
import android.os.AsyncTask;
import android.os.Bundle;
import android.widget.ArrayAdapter;

import com.example.readrssfeed.MyParserEngine.NewsItem;

public class MainActivity extends ListActivity { ❶

    private ArrayAdapter<NewsItem> adapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //setContentView(R.layout.activity_main); ❷

        adapter = new ArrayAdapter<NewsItem>(this,
                        android.R.layout.simple_list_item_1,
```

```
                        android.R.id.text1);
    setListAdapter(adapter);

    DownloadFeedTask task = new DownloadFeedTask();
    task.execute("http://www.thairath.co.th/rss/tech.xml");
}

private InputStream downloadFeed(String strUrl) { ❸
    try {
        URL url = new URL(strUrl);
        HttpURLConnection con = (HttpURLConnection)
                                url.openConnection();
        return con.getInputStream();
    } catch (Exception e) {
        e.printStackTrace();
        return null;
    }
}

private class DownloadFeedTask extends AsyncTask<String, Void,
                                                  ArrayList<NewsItem>> {

    @Override
    protected ArrayList<NewsItem> doInBackground(String... urls) {
        InputStream stream = downloadFeed(urls[0]); ❹

        try {
            return MyParserEngine.parse(stream); ❺
        } catch (XmlPullParserException e) {
            e.printStackTrace();
            return null;
        } catch (IOException e) {
            e.printStackTrace();
            return null;
        }
    }

    @Override
    protected void onPostExecute(ArrayList<NewsItem> items) { ❻
        adapter.clear();
        for (NewsItem item : items) {
            adapter.add(item);                                  ❼
        }
        adapter.notifyDataSetChanged();
```

```
        }
    }
}
```

เรากำหนดให้ MainActivity เป็นซับคลาสของ ListActivity ❶ เนื่องจากต้องการให้มี ListView เพียงอย่างเดียวในหน้าจอ และทำให้ไม่ต้องเรียก setContentView เพื่อสร้างหน้าจอจาก Layout File เอง ❷ (ดูหัวข้อ "การแสดงรายการข้อมูลด้วย ListActivity" ในบทที่ 4)

เมธอด downloadFeed ❸ จะดาวน์โหลดข้อมูลฟีดจาก URL ที่เรากำหนด แล้วส่งคืน InputStream กลับมาให้ เราเรียกใช้ downloadFeed จากภายในเมธอด doInBackground ของ AsyncTask เพื่อรันเมธอดนี้ในเธรดใหม่ ❹ ซึ่งเมื่อได้ InputStream มาแล้วจะส่งให้เมธอด parse ของ MyParseEngine ทำการ parse ฟีดต่อไป ❺

เมธอด parse จะให้ผลลัพธ์เป็น ArrayList ที่บรรจุออบเจ็ค NewsItem ที่เก็บข้อมูลของข่าวหนึ่งๆ (จำนวน NewsItem ใน ArrayList จึงเท่ากับจำนวนข่าวในฟีด) ArrayList จะถูก return ออกจากเมธอด doInBackground และถูกส่งไปยังเมธอด onPostExecute ❻ ของ AsyncTask เพื่อใส่ NewsItem ลงใน Adapter ❼

2 สร้างคลาสใหม่ชื่อ MyParseEngine แล้วพิมพ์โค้ดดังนี้

โปรเจ็ค ReadRssFeed, ไฟล์ MyParserEngine.java

```
package com.example.readrssfeed;

import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;

import org.xmlpull.v1.XmlPullParser;
import org.xmlpull.v1.XmlPullParserException;
import org.xmlpull.v1.XmlPullParserFactory;

public class MyParserEngine {

    // แต่ละ NewsItem จะเก็บหัวข้อข่าวและลิงค์ที่เชื่อมโยงไปยังเนื้อหาข่าวเต็ม
    public static class NewsItem {
        public String title;
        public String link;

        /* Override เมธอด toString เพื่อให้แสดงหัวข้อข่าวสำหรับแต่ละไอเท็มใน ListView
           (อย่าลืมว่าข้อมูลที่เราใส่ลง Adapter คือออบเจ็ค NewsItem ไม่ใช่ค่าสตริงธรรมดา
           ดังนั้นข้อมูลที่แสดงใน ListView จึงขึ้นอยู่กับเมธอด toString นี้ */
        @Override
        public String toString() {
```

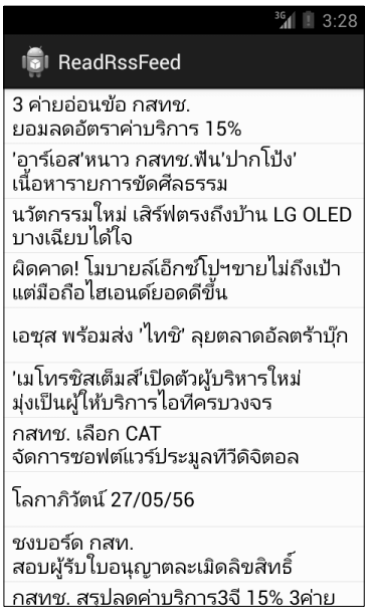
```
            return title;
        }
    }

    // เมธอด parse จะรับ InputStream เข้ามา แล้วเริ่มการ parse
    public static ArrayList<NewsItem> parse(InputStream stream)
                  throws XmlPullParserException, IOException {

        ArrayList<NewsItem> items = new ArrayList<NewsItem>();

        XmlPullParserFactory factory = XmlPullParserFactory.newInstance();
        XmlPullParser parser = factory.newPullParser();
        parser.setInput(stream, "utf-8");

        // วนลูปจนกว่าจะสิ้นสุดข้อมูล XML
        while (parser.next() != XmlPullParser.END_DOCUMENT) {
            // ถ้าไม่ใช่แท็กเปิด ให้ข้ามไปยังรอบถัดไปเลย
            if (parser.getEventType() != XmlPullParser.START_TAG) {
                continue;
            }
            /* ถ้าพบแท็กเปิดของอิลิเมนต์ item ให้ทำการ parse อิลิเมนต์ item นั้น แล้วเก็บ
               ออบเจ็ค NewsItem ซึ่งเป็นผลลัพธ์ของการ parse ลงใน ArrayList */
            if (parser.getName().equals("item")) {
                NewsItem item = parseItem(parser);
                items.add(item);
            }
        }

        return items;  // ส่งคืน ArrayList ออกไปจากเมธอด parse
    }

    // เมธอดสำหรับ parse อิลิเมนต์ item
    private static NewsItem parseItem(XmlPullParser parser)
                   throws XmlPullParserException, IOException {
        NewsItem item = new NewsItem();

        // วนลูปจนกว่าจะพบแท็กปิดของอิลิเมนต์ item
        while (parser.next() != XmlPullParser.END_TAG) {
            // ถ้าไม่ใช่แท็กเปิด ให้ข้ามไปยังรอบถัดไปเลย
            if (parser.getEventType() != XmlPullParser.START_TAG) {
                continue;
            }

            String tagName = parser.getName();
```

```
            /* ถ้าพบแท็กเปิดของอิลิเมนต์ title หรือ link จะอ่านข้อมูลในอิลิเมนต์มาเก็บ
               ลงในออบเจ็ค NewsItem แต่ถ้าเป็นแท็กเปิดอื่นๆนอกเหนือจากสองแท็กนี้จะข้ามไป
               จนถึงแท็กปิดของมัน */
            if (tagName.equals("title")) {
                item.title = readContent(parser);
            } else if (tagName.equals("link")) {
                item.link = readContent(parser);
            } else {
                while (parser.next() != XmlPullParser.END_TAG) {}
            }
        }

        return item;
    }

    // เมธอดสำหรับอ่านข้อมูลในอิลิเมนต์
    private static String readContent(XmlPullParser parser)
                    throws XmlPullParserException, IOException {
        String result = "";
        if (parser.next() == XmlPullParser.TEXT) {
            result = parser.getText();
            parser.next();
        }
        return result;
    }
}
```

3 เพิ่มบรรทัดต่อไปนี้ในไฟล์ AndroidManifest.xml โดยใส่ไว้ก่อนแท็ก `<application>` เพื่อขอสิทธิ์ในการเข้าถึงอินเทอร์เน็ต

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

ผลการรัน



แสดงเนื้อหาข่าวเต็มเมื่อคลิกหัวข้อข่าว

ออบเจ็ค `NewsItem` ไม่เพียงแต่เก็บหัวข้อข่าว แต่ยังเก็บลิงค์ที่เชื่อมโยงไปยังเนื้อหาข่าวเต็มในเว็บไซต์ไทยรัฐออนไลน์ด้วย ดังนั้นเราจะเพิ่มการทำงานของแอพโดยเมื่อผู้ใช้คลิก (แตะ) หัวข้อข่าวในลิสต์ก็ให้แสดงเนื้อหาข่าวเต็มในบราวเซอร์ โดยเพิ่มเมธอด `onListItemClick` ในแอคทิวิตี ดังนี้

โปรเจ็ค ReadRssFeed, ไฟล์ MainActivity.java

```
@Override
protected void onListItemClick(ListView l, View v, int position, long id) {
    // เข้าถึงออบเจ็ค NewsItem ที่สัมพันธ์กับไอเท็มที่ถูกคลิก
    NewsItem item = (NewsItem) l.getItemAtPosition(position);
    String url = item.link;
    // สร้างอินเทนต์เพื่อรันแอคทิวิตีที่มีความสามารถในการแสดงผลเว็บเพจตาม URL ที่ระบุ
    Intent i = new Intent(Intent.ACTION_VIEW, Uri.parse(url));
    startActivity(i);
}
```

การส่งข้อมูลไปยังเซิร์ฟเวอร์ด้วย HTTP POST

โปรโตคอล HTTP มีเมธอด (Request Method) ที่ใช้บ่อยอยู่ 2 เมธอดคือ GET กับ POST โดยหน้าที่หลักของเมธอดทั้งสองคือ GET ใช้ขอข้อมูลจากเซิร์ฟเวอร์ ในขณะที่ POST ใช้ส่งข้อมูลไปยังเซิร์ฟเวอร์ (รวมถึงการอัพโหลดไฟล์ที่จะกล่าวในหัวข้อถัดไป)

ที่ผ่านมาเราอ่านข้อมูลจากเซิร์ฟเวอร์โดยใช้เมธอด GET ของ HTTP ทั้งหมด ดังนั้นในหัวข้อนี้จะแสดงวิธีการส่งข้อมูลไปยังเซิร์ฟเวอร์โดยใช้เมธอด POST

NOTE

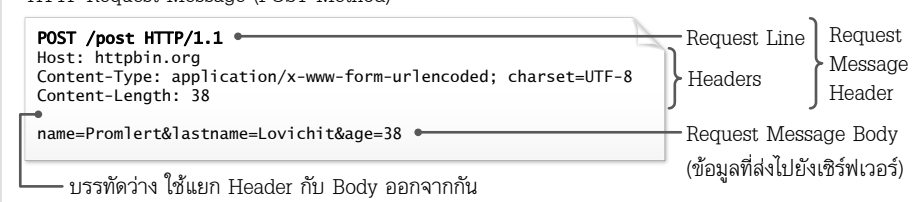
ความจริงเมื่อใช้ HTTP GET เราสามารถส่งข้อมูลไปให้เซิร์ฟเวอร์ก่อนที่จะรับข้อมูลกลับมาได้ โดยระบุ Query String ที่มีรูปแบบดังนี้ต่อท้าย URL

```
?field1=value1&field2=value2&field3=value3...
```

ยกตัวอย่างเช่น ตอนที่เรียกใช้เว็บเซอร์วิส v2 Forecast API นั้นเราระบุ Query String เป็น `?units=si` เพื่อบอกเว็บเวอร์วิสว่าเราต้องการข้อมูลในหน่วย SI เป็นต้น

การส่งข้อมูลผ่านทาง Query String ของ HTTP GET จะใช้กับข้อมูลขนาดเล็ก ซึ่งมักเป็นตัวเลือกในการทำงานของเว็บเซอร์วิสหรือ Server-side script ที่อยู่ฝั่งเซิร์ฟเวอร์ ดังเช่นกรณีของ v2 Forecast API ที่กล่าวข้างต้น ส่วนการส่งข้อมูลจำนวนมากจะต้องใช้ HTTP POST ซึ่งข้อมูลจะถูกใส่ลงในส่วน Body ของ

Request Message ดังรูปหน้าถัดไป



ตัวอย่างและคำอธิบาย

ตัวอย่างนี้จะใช้ HTTP POST ส่งข้อมูลไปยัง `httpbin.org/post` ซึ่งเป็นเว็บที่เอาไว้ทดสอบ/ตรวจสอบ HTTP Request ที่ส่งไปจาก Client โดย `httpbin.org/post` จะส่งข้อมูลแบบ JSON ที่บอกรายละเอียดเกี่ยวกับ HTTP Request ที่มันได้รับกลับมาให้

1 กำหนด Layout ของหน้าจอ

โปรเจ็ค HttpPost, ไฟล์ activity__main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp" >

    <Button
        android:id="@+id/post_button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Post Data" />

    <TextView
        android:id="@+id/text"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:text="@string/hello_world" />

</LinearLayout>
```

2 สร้างคลาสใหม่ชื่อ PostTask แล้วพิมพ์โค้ดดังนี้

โปรเจ็ค HttpPost, ไฟล์ PostTask.java

```
package com.example.httppost;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.UnsupportedEncodingException;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLEncoder;
import java.nio.charset.Charset;
import java.util.ArrayList;

import org.apache.http.NameValuePair;
import org.apache.http.message.BasicNameValuePair;

import android.app.Activity;
import android.content.Context;
import android.os.AsyncTask;
import android.widget.TextView;

public class PostTask extends AsyncTask<String, Void, String> {

    private Context mContext;
    private ArrayList<BasicNameValuePair> mData;

    public PostTask(Context context, ArrayList<BasicNameValuePair> data) { ❶
        mContext = context;
        mData = data;
    }

    @Override
    protected String doInBackground(String... urls) {
        return postData(urls[0]);
    }

    @Override
    protected void onPostExecute(String result) {
        TextView textview = (TextView) ((Activity) mContext)
                                        .findViewById(R.id.text);
```

```
        textview.setText(result);
    }

    private String postData(String strUrl) {
        // หาชื่อ Character Set ที่เป็นดีฟอลต์ของระบบ
        String charset = Charset.defaultCharset().displayName();
        String strResult = "";

        try {
            // นำข้อมูลจาก ArrayList มาจัดรูปแบบให้เหมาะสมตามข้อกำหนดของโปรโตคอล HTTP
            String requestBody = setRequestBody(charset);

            URL url = new URL(strUrl);
            HttpURLConnection con = (HttpURLConnection)
                                    url.openConnection();

            // เมธอดนี้จะทำให้เรียกไปยังเซิร์ฟเวอร์ด้วย HTTP POST แทนที่จะเป็น HTTP GET
            con.setDoOutput(true);
            // กำหนด Content Type
            con.setRequestProperty("Content-Type",
                "application/x-www-form-urlencoded; charset=" + charset);
            // เปิดการใช้งาน Streaming ในแบบที่รู้ขนาดข้อมูลล่วงหน้า
            con.setFixedLengthStreamingMode(requestBody.length());

            /* เขียนข้อมูลที่จัดรูปแบบแล้วลงใน Output Stream ซึ่งจะทำให้ข้อมูลถูกใส่ลงใน
               ส่วน Body ของ HTTP Request Message ที่เรียกไปยัง URL ปลายทาง */
            OutputStream out = con.getOutputStream();
            out.write(requestBody.getBytes(charset));
            out.flush();

            /* อ่านข้อมูล (ผลลัพธ์) จาก URL ซึ่งกรณีของ httpbin.org/post ข้อมูลนี้จะเป็น
               รายละเอียดเกี่ยวกับ HTTP Request ที่เราส่งไป */
            strResult = readStream(con.getInputStream());
        } catch (Exception e) {
            e.printStackTrace();
        }
        return strResult;
    }

    /* เมธอดที่จะนำข้อมูลจาก ArrayList<BasicNameValuePair> มาจัดรูปแบบให้เหมาะสมสำหรับ
       การใส่ลงในส่วน Body ของ HTTP Request Message ตามข้อกำหนดของโปรโตคอล HTTP
       (ดูเพิ่มเติมในกรอบ Note ถัดไป) */
    private String setRequestBody(String charset) throws
                    UnsupportedEncodingException {
```

```
        StringBuilder sb = new StringBuilder();

        for (int i = 0; i < mData.size(); i++) {
            NameValuePair item = mData.get(i);
            sb.append(URLEncoder.encode(item.getName(), charset)); // ชื่อ
            sb.append("=");
            sb.append(URLEncoder.encode(item.getValue(), charset)); // ค่า
            if (i != (mData.size() - 1)) {
                sb.append("&"); // ใช้ & แยกระหว่างข้อมูลแต่ละชุด
            }
        }
        return sb.toString();
    }

    /* เมธอด readStream ใช้อ่านข้อมูลผลลัพธ์จาก URL ซึ่งโค้ดเหมือนในตัวอย่าง
       HttpDownloadText ก่อนหน้านี้ทุกประการ */
    private String readStream(InputStream in) {
        BufferedReader reader = null;
        StringBuilder sb = new StringBuilder();
        try {
            reader = new BufferedReader(new InputStreamReader(in));

            String line;
            while ((line = reader.readLine()) != null) {
                sb.append(line + "\n");
            }

        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if (reader != null) {
                try {
                    reader.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
        return sb.toString();
    }
}
```

โค้ดจะคล้ายกับตัวอย่าง HttpDownloadText ก่อนหน้านี้ แต่ในตัวอย่างนี้แยกคลาส PostTask ซึ่งเป็นซับคลาสของ AsyncTask ออกมาเป็นไฟล์ใหม่ต่างหาก (เดิมเป็น Inner Class ของแอคทิวิตี) เพื่อให้โค้ดดูง่ายขึ้น

เราออกแบบให้คอนสตรัคเตอร์ของ PostTask ❶ มีพารามิเตอร์ 2 ตัว คือ คอนเท็กซ์ กับ ArrayList ของ BasicNameValuePair ซึ่งเก็บข้อมูลเป็นชุดๆในรูปแบบชื่อ/ค่า เราจะสร้าง ArrayList ขึ้นในเมธอด onCreate ของแอคทิวิตี (อีกสักครู่จะได้เห็น) แล้วส่งผ่านมายังคอนสตรัคเตอร์ของ PostTask

การทำงานของ PostTask จะเริ่มที่เมธอด doInBackground ดังที่เคยอธิบายแล้ว รายละเอียดขอให้ดูจากคอมเมนต์ในโค้ด

3 เพิ่มโค้ดในเมธอด onCreate ของแอคทิวิตี

โปรเจ็ค HttpPost, ไฟล์ MainActivity.java

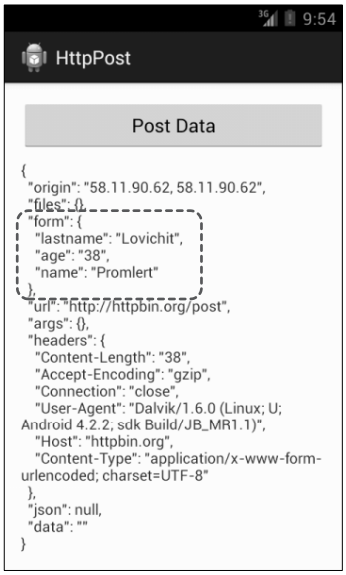
```
Button button = (Button) findViewById(R.id.post_button);
button.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {
        // สร้าง ArrayList แล้วเพิ่มข้อมูลลงไป 3 ชุด
        ArrayList<BasicNameValuePair> data =
                new ArrayList<BasicNameValuePair>();

        data.add(new BasicNameValuePair("name", "Promlert"));
        data.add(new BasicNameValuePair("lastname", "Lovichit"));
        data.add(new BasicNameValuePair("age", "38"));

        /* สร้างอินสแทนซ์ของ PostTask แล้วสั่งให้ทำงาน (execute) ซึ่งการทำงานจะเริ่มที่
           เมธอด doInBackground ของ PostTask */
        PostTask task = new PostTask(MainActivity.this, data);
        task.execute("http://httpbin.org/post");
    }
});
```

4 เพิ่มบรรทัดต่อไปนี้ในไฟล์ AndroidManifest.xml โดยใส่ไว้ก่อนแท็ก <application> เพื่อขอสิทธิ์ในการเข้าถึงอินเทอร์เน็ต

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

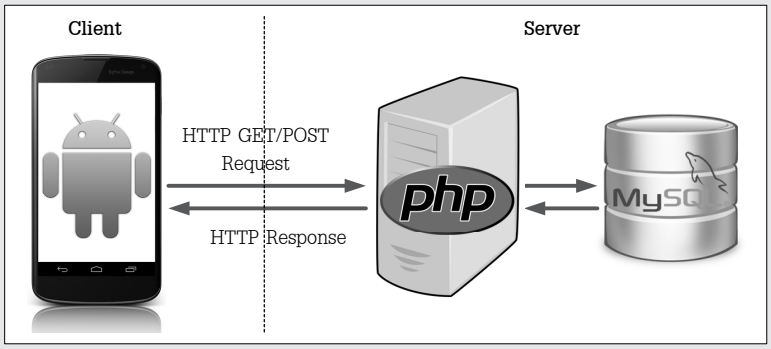
ผลการรัน

เมื่อคลิก Post Data จะปรากฏข้อมูลดังรูป ซึ่งเป็นข้อมูล JSON ที่ `httpbin.org/post` ส่งกลับมาหลังจากได้รับ HTTP POST แล้ว สังเกตว่ามีข้อมูลที่เราส่งไปให้รวมอยู่ด้วย



TIP

ในทางปฏิบัติจริงๆ เมื่อ URL ปลายทางได้รับข้อมูลแล้วอาจนำข้อมูลไปประมวลผลหรือบันทึกลงฐานข้อมูลที่ฝั่งเซิร์ฟเวอร์ หลักการนี้ทำให้เราสามารถสร้างแอพแอนดรอยด์ติดต่อกับฐานข้อมูลที่ฝั่งเซิร์ฟเวอร์ (เช่น MySQL) ได้ โดยเราจะเตรียม Server-side script (เขียนด้วยภาษาเช่น PHP) ไว้ที่ URL หนึ่ง ซึ่ง Server-side script จะเป็นตัวกลางในการรับข้อมูลที่ส่งมาจากแอพแอนดรอยด์ทาง HTTP POST แล้วนำข้อมูลไปบันทึกหรือดำเนินการใดๆกับฐานข้อมูลที่ฝั่งเซิร์ฟเวอร์ หรืออาจเป็นการคิวรีฐานข้อมูลตาม Query String ที่ระบุมาใน HTTP GET แล้วส่งข้อมูลกลับไปให้แอพแอนดรอยด์ ดังรูป



การอัพโหลดไฟล์ไปยังเซิร์ฟเวอร์

การอัพโหลดไฟล์ไปยังเซิร์ฟเวอร์จะใช้ HTTP POST เช่นเดียวกับตัวอย่างที่แล้ว แต่วิธีการเขียนโค้ดยุ่งยากกว่าเล็กน้อย เราจะมาดูกันในหัวข้อนี้

ตัวอย่างและคำอธิบาย

ตัวอย่างนี้จะอัพโหลดไฟล์รูปภาพ ic_launcher.png ที่ ADT เตรียมมาให้ในโฟลเดอร์ res\drawable-*xxx* ของโปรเจ็ค ไปยัง `httpbin.org/post` แล้วรับข้อมูลผลลัพธ์มาแสดงผล ซึ่งจะเป็นรายละเอียดเกี่ยวกับ HTTP Request ที่เราส่งไป หลังจากนั้นจะแสดงการเขียนโค้ด PHP ที่ฝั่งเซิร์ฟเวอร์เพื่อรับไฟล์รูปภาพดังกล่าวจากแอพแอนดรอยด์ แล้วส่งข้อมูลในรูปแบบ HTML กลับไป

1 กำหนด Layout ของหน้าจอ

โปรเจ็ค HttpUpload, ไฟล์ activity__main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp" >

    <Button
        android:id="@+id/upload_button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Upload File" />

    <TextView
        android:id="@+id/text"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:text="@string/hello_world" />

</LinearLayout>
```

2 สร้างคลาสใหม่ชื่อ `UploadTask` แล้วพิมพ์โค้ดดังนี้

โปรเจ็ค HttpUpload, ไฟล์ UploadTask.java

```
package com.example.httpupload;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.FileInputStream;
```

```
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLConnection;
import java.nio.charset.Charset;

import android.app.Activity;
import android.content.Context;
import android.os.AsyncTask;
import android.widget.TextView;

public class UploadTask extends AsyncTask<String, Void, String> {

    private Context mContext;
    private File mFile;

    public UploadTask(Context context, File file) {
        mContext = context;
        mFile = file;
    }

    @Override
    protected String doInBackground(String... urls) {
        return uploadFile(urls[0]);
    }

    @Override
    protected void onPostExecute(String result) {
        TextView textview = (TextView) ((Activity) mContext)
                                        .findViewById(R.id.text);
        textview.setText(result);
    }

    private String uploadFile(String strUrl) {
        String charset = Charset.defaultCharset().displayName();
        /* สตริงที่ใช้กำหนดขอบเขตข้อมูลในส่วน Body ของ Request Message (จะใช้สตริงอะไรก็ได้
           แต่ต้องใช้ให้เหมือนกันทุกจุด และต้องแน่ใจว่าสตริงนี้จะไม่เป็นส่วนหนึ่งในเนื้อหาไฟล์ที่อัพโหลด
           ในที่นี้จึงใช้วิธีสร้าง boundary ขึ้นจากค่าเวลาในขณะนั้นๆ) */
        String boundary = Long.toHexString(System.currentTimeMillis());
        String strResult = "";
```

```
        try {
            URL url = new URL(strUrl);
            HttpURLConnection con = (HttpURLConnection)
                                    url.openConnection();

            con.setDoOutput(true);
            /* การอัพโหลดไฟล์ต้องกำหนด Content-Type เป็น multipart/form-data เสมอ
               และบอกด้วยว่า boundary ที่ใช้คืออะไร */
            con.setRequestProperty("Content-Type",
                    "multipart/form-data; boundary=" + boundary);

            /* หาขนาดของ metadata ที่จะเขียนลงในส่วน Body ของ Request Message
               โดยเขียนข้อมูลลง ByteArrayOutputStream แทน และไม่เขียนเนื้อหาไฟล์
               (พารามิเตอร์ false ที่ส่งให้เมธอด writeMultipart) จากนั้นจึงหาขนาดข้อมูลใน
               ByteArrayOutputStream นั้น แล้วรวมกับขนาดไฟล์ ก็จะเป็นขนาดของข้อมูลใน
               ส่วน Body ทั้งหมด */
            ByteArrayOutputStream bos = new ByteArrayOutputStream();
            writeMultipart(boundary, charset, bos, false);
            byte[] extra = bos.toByteArray();
            int contentLength = extra.length;
            contentLength += mFile.length();

            // เปิดการใช้งาน Streaming ในแบบที่รู้ขนาดข้อมูลล่วงหน้า (ซึ่งหาขนาดไว้แล้วข้างต้น)
            con.setFixedLengthStreamingMode(contentLength);

            /* เขียนข้อมูลลง OutputStream ซึ่งข้อมูลจะถูกใส่ลงในส่วน Body ของ HTTP
               Request Message */
            OutputStream out = con.getOutputStream();
            writeMultipart(boundary, charset, out, true);

            /* อ่านข้อมูล (ผลลัพธ์) จาก URL ซึ่งกรณีของ httpbin.org/post ข้อมูลนี้จะเป็น
               รายละเอียดเกี่ยวกับ HTTP Request ที่เราส่งไป */
            strResult = readStream(con.getInputStream());
        } catch (Exception e) {
            e.printStackTrace();
        }
        return strResult;
    }

    /* เมธอดสำหรับเขียนข้อมูลลงใน OutputStream ที่กำหนด และเลือกได้ว่าจะเขียนเนื้อหาไฟล์ลงไป
       ด้วยหรือไม่ (พารามิเตอร์ writeContent) (ทางเลือกในการเขียนหรือไม่เขียนเนื้อหาไฟล์นี้ ทำไว้เพื่อ
       หาขนาด metadata โดยหากต้องการหาขนาด metadata ก็ให้ส่งค่า false มาที่พารามิเตอร์นี้
       แต่ถ้าจะเขียนข้อมูลเพื่อส่งไปยังเซิร์ฟเวอร์จริงๆก็ให้ส่งค่า true มา) */
```

```
private void writeMultipart(String boundary, String charset,
        OutputStream output, boolean writeContent) throws IOException {

    BufferedWriter writer = null;

    try {
        /* สร้าง OutputStreamWriter ครอบ OutputStream ที่ส่งเป็นพารามิเตอร์เข้ามา
           แล้วสร้าง BufferedWriter ครอบ OutputStreamWriter อีกที เพื่อเขียน
           ข้อความธรรมดาลงใน OutputStream */
        writer = new BufferedWriter(new OutputStreamWriter(output,
                    Charset.forName(charset)), 8192);

        // จุดเริ่มต้นของ multipart/form-data
        writer.write("--" + boundary);
        writer.write("\r\n");
        /* กำหนดข้อมูล metadata
           Content-Disposition: form-data; name="myfile";
           filename="ชื่อไฟล์"
           โดย name="myfile" คือชื่อที่เราจะใช้อ้างถึงไฟล์จากโค้ดที่ฝั่งเซิร์ฟเวอร์
           (เดี๋ยวตอนเขียน PHP จะเห็น) */
        writer.write("Content-Disposition: form-data; "
                    + "name=\"myfile\"; "
                    + "filename=\"" + mFile.getName() + "\"");
        writer.write("\r\n");
        // กำหนด Content Type โดยพิจารณาจากนามสกุลไฟล์
        writer.write("Content-Type: "
                    + URLConnection.guessContentTypeFromName(
                      mFile.getName()));
        writer.write("\r\n");
        writer.write("Content-Transfer-Encoding: binary");
        writer.write("\r\n");
        writer.write("\r\n");
        writer.flush();

        // ภายใน if นี้เป็นการเขียนเนื้อหาไฟล์ลง OutputStream
        if (writeContent) {
            FileInputStream fis = null;
            // วนลูปอ่านเนื้อหาไฟล์มาทีละ 1,024 ไบต์ แล้วเขียนลง OutputStream
            try {
                fis = new FileInputStream(mFile);
                byte[] buffer = new byte[1024];
                for (int len = 0; (len = fis.read(buffer)) > 0;) {
                    output.write(buffer, 0, len);
                }
```

```
                    output.flush();
                } catch (IOException e) {
                    e.printStackTrace();
                } finally {
                    if (fis != null) {
                        try {
                            fis.close();
                        } catch (IOException e) {
                        }
                    }
                }
            }
            writer.write("\r\n");  // จุดสิ้นสุดของข้อมูล Binary
            writer.flush();

            // จุดสิ้นสุดของ multipart/form-data
            writer.write("--" + boundary + "--");
            writer.write("\r\n");
            writer.flush();
        } finally {
            if (writer != null) {
                writer.close();
            }
        }
    }

    private String readStream(InputStream in) {
        // โค้ดเหมือนตัวอย่างที่แล้วทุกประการ
    }
}
```

โครงสร้างการทำงานของคลาส UploadTask จะเหมือนกับคลาส PostTask ในตัวอย่างที่แล้ว โดยเป็นซับคลาสของ AsyncTask ซึ่งจะอัพโหลดไฟล์ไปยังเซิร์ฟเวอร์ (เมธอด doInBackground ซึ่งจะเรียกไปยังเมธอด uploadFile และเมธอดอื่นๆอีกที) แล้วนำข้อมูลผลลัพธ์ที่ได้จากเซิร์ฟเวอร์มาแสดงที่ TextView (เมธอด onPostExecute)

คอนสตรัคเตอร์ของ UploadTask มีพารามิเตอร์ 2 ตัว คือ คอนเท็กซ์ และออบเจ็ค File ที่เป็นตัวแทนของไฟล์ที่จะอัพโหลด

3 เพิ่มโค้ดในเมธอด onCreate ของแอคทิวิตี

โปรเจ็ค HttpUpload, ไฟล์ MainActivity.java

```
Button button = (Button) findViewById(R.id.upload_button);
button.setOnClickListener(new View.OnClickListener() {
```

```
    @Override
    public void onClick(View v) {
        try {
            // อ่านข้อมูลจากรูปภาพ ic_launcher มาเก็บในออบเจ็ค Bitmap
            Bitmap image = BitmapFactory.decodeResource(getResources(),
                                R.drawable.ic_launcher);
            // สร้างไฟล์ android.png ในไดเร็คทอรีชั่วคราว
            File imageFile = new File(getCacheDir(), "android.png");
            // สร้าง FileOutputStream สำหรับเขียนข้อมูลลงไฟล์ android.png
            FileOutputStream fos = new FileOutputStream(imageFile);
            // บีบอัดข้อมูลจากออบเจ็ค Bitmap เป็นฟอร์แมต PNG แล้วเขียนลงไฟล์ android.png
            image.compress(CompressFormat.PNG, 0, fos);
            fos.flush();
            fos.close();

            /* สร้างอินสแทนซ์ของ UploadTask แล้วสั่งให้ทำงาน (execute) ซึ่งการทำงานจะ
               เริ่มที่เมธอด doInBackground ของ UploadTask */
            UploadTask task = new UploadTask(MainActivity.this, imageFile);
            task.execute("http://httpbin.org/post");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
});
```

แอนดรอยด์ไม่มีวิธีให้เราสร้างออบเจ็ค `File` จากไฟล์รูปภาพที่เป็นรีซอร์สได้โดยตรง เนื่องจากรีซอร์สต่างๆจะถูกบีบอัดก่อนรวมเข้าไปใน APK ดังนั้นเราจึงต้องใช้วิธีอ้อมๆโดยอ่านข้อมูลรูปภาพมาเก็บในออบเจ็ค `Bitmap` แล้วนำไปสร้างเป็นไฟล์ใหม่ในไดเร็คทอรีชั่วคราว หลังจากนั้นจึงส่งออบเจ็ค `File` ของไฟล์นี้ไปให้คอนสตรัคเตอร์ของ `UploadTask`

4 เพิ่มบรรทัดต่อไปนี้ในไฟล์ AndroidManifest.xml โดยใส่ไว้ก่อนแท็ก `<application>` เพื่อขอสิทธิ์ในการเข้าถึงอินเทอร์เน็ต

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

ผลการรัน

เมื่อคลิก Upload File จะได้ผลลัพธ์จาก `httpbin.org/post` ดังรูป จะเห็นว่ามีข้อมูลเกี่ยวกับไฟล์ที่เราอัพโหลดรวมอยู่ เช่น ชื่อ myfile ที่เราจะใช้ในการอ้างอิงไฟล์จากโค้ดที่ฝั่งเซิร์ฟเวอร์ รวมถึงเนื้อหาไฟล์ด้วย แต่เนื่องจากไฟล์รูปภาพเป็นข้อมูลแบบ Binary เราจึงอ่านข้อมูลไม่รู้เรื่อง

สร้างไฟล์ PHP รับไฟล์ที่ถูกอัพโหลดมา

ผลลัพธ์ที่ `httpbin.org/post` ส่งกลับมานั้น ถึงแม้จะมีประโยชน์มาก เพราะช่วยให้เราตรวจสอบ Request Message ที่ส่งไปจากแอพของเราได้ แต่มันดูไม่น่าสนใจสำหรับตัวอย่างนี้ ผู้เขียนจึงคิดว่าเราน่าจะสร้างไฟล์ PHP ที่ฝั่งเซิร์ฟเวอร์ขึ้นมาเอง แล้วแก้โค้ดในแอพแอนดรอยด์ให้เรียก (อัพโหลดไฟล์) ไปยังไฟล์ PHP นั้นแทนที่จะเรียกไปยัง `httpbin.org/post`

โค้ดในไฟล์ PHP ของเราจะส่งข้อมูล HTML กลับไปให้แอพแอนดรอยด์ ซึ่งจะเป็นรายละเอียดเกี่ยวกับไฟล์ภาพ รวมถึงแสดงรูปภาพออกมาด้วย (โดยสร้างแท็ก `<img>` และระบุ URL ของไฟล์ภาพที่แอตทริบิวต์ `src`) แต่ทั้งนี้เราต้องใช้คอนโทรล `WebView` ในการแสดงผล HTML เหล่านั้น

NOTE

เราจะเตรียมไฟล์ PHP ดังกล่าวไว้ที่ฝั่งเซิร์ฟเวอร์ ดังนั้นผู้อ่านต้องมีพื้นที่ในเซิร์ฟเวอร์สำหรับสร้างโฮมเพจ/เว็บเพจของตัวเอง และเซิร์ฟเวอร์ต้องอนุญาตให้รันไฟล์ PHP ได้ หรืออีกวิธีก็คือให้จำลองเครื่องพีซีของตัวเองเป็นเซิร์ฟเวอร์ โดยติดตั้งชุดโปรแกรม AppServ (`http://www.appservnetwork.com`) ซึ่งจะทำให้เครื่องพีซีของคุณทำงานเป็นเซิร์ฟเวอร์และรัน PHP ได้ สำหรับรายละเอียดการติดตั้งจะไม่อธิบายในที่นี้

1 สร้างไฟล์ PHP ชื่อ get_file.php ขึ้นที่เซิร์ฟเวอร์

ไฟล์ get__file.php

```
<?php
if ($_FILES["myfile"]["error"] > 0) {
    echo "Error: " . $_FILES["myfile"]["error"];
}
else {
    echo "File uploaded successfully ^_^\n\n";

    $fileName = $_FILES["myfile"]["name"];
    $fileType = $_FILES["myfile"]["type"];
    $fileSize = $_FILES["myfile"]["size"];

    echo "File name: $fileName\n";
    echo "Type: $fileType\n";
    echo "Size: $fileSize Bytes\n";
}
?>
```

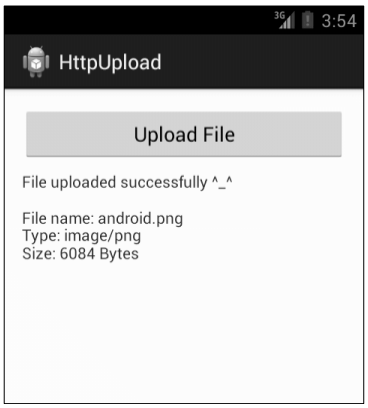
ใน PHP รายละเอียดต่างๆของไฟล์ที่ถูกอัพโหลดมาจะเก็บอยู่ในอาร์เรย์ $_FILES โดยคีย์อันแรก ("myfile") จะระบุถึงไฟล์ที่อัพโหลดมาจากแอพแอนดรอยด์ของเรา (ตามข้อมูล metadata ที่กำหนดในส่วน Body ของ Request Message)

ค่าของ $_FILES["myfile"]["name"] จะเป็นชื่อไฟล์ที่เรากำหนดไว้ใน metadata ดังกล่าวเช่นกัน ในที่นี้คือชื่อ android.png

2 ที่โปรเจ็คแอนดรอยด์ ให้แก้ไขบรรทัด `task.execute` ในเมธอด `onCreate` ของแอคทิวิตี โดยระบุ URL ของไฟล์ PHP ข้างต้นลงไปแทน `http://httpbin.org/post`

```
task.execute("http://www.promlert.com/temp/get_file.php");
```

3 รันแอพ แล้วคลิก Upload File จะได้ดังรูป



ผลลัพธ์ที่ได้จาก PHP ตอนนี้ยังเป็นแค่ข้อความธรรมดา ถัดไปเราจะแก้ไขโค้ด PHP ให้ส่งข้อมูล HTML กลับไป

4 แก้ไขไฟล์ get_file.php ดังนี้ (ในซีดีจะแยกออกมาเป็นไฟล์ใหม่ชื่อ get_file2.php)

ไฟล์ get__file2.php

```
<?php
if ($_FILES["myfile"]["error"] > 0) {
    echo "Error: " . $_FILES["myfile"]["error"];
}
else {
    echo "File uploaded successfully ^_^<br><br>";

    $fileName = $_FILES["myfile"]["name"];
    $fileType = $_FILES["myfile"]["type"];
    $fileSize = $_FILES["myfile"]["size"];
?>

    <table width="100%" border="0" cellpadding="5" cellspacing="3">
        <tr bgcolor="#FFFF99">
            <td>File name:</td><td><?php echo $fileName; ?></td>
        </tr>
        <tr bgcolor="#FFFF99">
            <td>Type:</td><td><?php echo $fileType; ?></td>
        </tr>
        <tr bgcolor="#FFFF99">
            <td>Size:</td><td><?php echo $fileSize; ?></td>
        </tr>
    </table><br>

<?php
    /* เมื่อไฟล์ถูกอัพโหลดมาแล้วจะถูกเก็บไว้ในไดเร็คทอรีชั่วคราว เราต้องย้ายไปยังไดเร็คทอรีที่ต้องการเอง
       โดยใช้ฟังก์ชั่น move_uploaded_file ในที่นี้จะย้ายไปยังไดเร็คทอรีเดียวกันกับไฟล์ PHP นี้ */
    $destination = './' . $_FILES["myfile"]["name"];

    if (move_uploaded_file($_FILES["myfile"]["tmp_name"], $destination)) {
        /* หลังจากย้ายไฟล์รูปภาพสำเร็จ จะหา URL ของไฟล์เพื่อนำไปกำหนดในแท็ก <img> เพื่อแสดง
           รูปภาพออกมา */
        $imageFullPath = realpath($destination);
        $imagePath = str_replace($_SERVER['DOCUMENT_ROOT'], '',
                                 $imageFullPath);
        $imageUrl = 'http://' . $_SERVER['HTTP_HOST'] . $imagePath;

        echo "This is an image you've just uploaded<br><br>";
        // สร้างแท็ก <img> เพื่อแสดงรูปภาพ
```

```
        echo "<div align=\"center\"><img src=\"$imageUrl\" border=\"2\" />
              </div>";
    } else {
        echo "Error copying file to the destination directory";
    }
}
?>
```

5 ที่โปรเจ็คแอนดรอยด์ ให้แก้ไข Layout File โดยลบ `TextView` แล้วเพิ่ม `WebView` เข้ามาแทน

โปรเจ็ค HttpUpload, ไฟล์ activity__main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp" >

    <Button
        android:id="@+id/upload_button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Upload File" />

    <WebView
        android:id="@+id/web"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginTop="8dp" />

</LinearLayout>
```

6 ที่คลาส `UploadTask` ให้แก้โค้ดในเมธอด `onPostExecute` เพื่อนำข้อมูลผลลัพธ์มาแสดงที่ `WebView`

โปรเจ็ค HttpUpload, ไฟล์ UploadTask.java

```
@Override
protected void onPostExecute(String result) {
    WebView webview = (WebView) ((Activity) mContext)
                                .findViewById(R.id.web);
    webview.loadData(result, "text/html", "utf-8");
}
```

7 รันแอพใหม่ เมื่อคลิก Upload File จะได้ดังรูป