Fpt University

FPT POLYTECHNIC

LẬP TRÌNH ANDROID NÂNG CAO

Bài 6: Các thao tác mạng

www.poly.edu.vn

Nội dung bài học

- Kiểm tra kết nối mạng của thiết bị
- Quản lý sử dụng mạng của thiết bị
- Miêu tả Preference Activity
- Đáp ứng thay đổi của Preference
- Nhận biết thay đổi kết nối mạng
- Phân tích XML Data

Kiểm tra kết nối mạng của thiết bị

- Một thiết bị có thể có nhiều kiểu kết nối mạng
- Hai kiểu kết nối mạng chủ yếu là kết nối WiFi hoặc 3G
- WiFi thường có tốc độ nhanh hơn, 3G thì thường chậm hơn
- Trước khi bạn thực hiện các thao tác mạng, bạn nên kiểm tra trạng thái kết nối mạng
- Nếu ứng dụng của bạn không có kết nối mạng, bạn nên thông báo với người dùng

Kiểm tra kết nối mạng của thiết bị

- Hai lớp sau được dùng để kiểm tra kết nối mạng
  - **ConnectionManager**: Truy vấn trạng thái kết nối mạng, đồng thời thông báo người dùng khi có thay đổi về kết nối mạng của thiết bị
  - **NetworkInfo**: miêu tả trạng thiết của mạng ứng với một kiểu cụ thể (WiFi hoặc 3G)

Kiểm tra kết nối mạng của thiết bị

- Sử dụng phương thức **getNetworkInfo** của lớp **NetworkInfo** để kiểm tra kết nối mạng
- Lớp ConnectivityManager có các kiểu **TYPE_WIFI** (mạng WiFi) hoặc **TYPE_MOBILE** (mạng 3G)
- Phương thức isConnected của NetworkInfo để kiểm tra kết nối mạng WIFI, 3G,.. của thiết bị. Phương thức trả lại giá trị true hoặc false
- Khi kiểm tra kết nối mạng trên Emulator, bạn sử dụng phím F8 để tắt và bật 3G trên Emulator
- Phải bổ sung quyền **android.permission.INTERNET** và **android.permission.ACCESS_NETWORK_STATE** trong AndroidManifest.xml để ứng dụng có thể thực hiện các kết nối INTERNET

Kiểm tra kết nối mạng của thiết bị

```
public void checkNetworkStatus(View v)
{

    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();
    if (isWifiConn)
    {
        Toast.makeText(getApplicationContext(), "Thiết bị đã kết nối Wifi", Toast.LENGTH_LONG)
        .show();
    }
    if (isMobileConn)
    {
        Toast.makeText(getApplicationContext(), "Thiết bị đã kết nối 3G", Toast.LENGTH_LONG)
        .show();
    }
}
```

DEMO

Kiểm tra kết nối mạng Wifi, 3G của thiết bị Android

Quản lý sử dụng mạng

- Bạn có thể xây dựng một preference activity cho phép người dùng điều khiển sử dụng tài nguyên cho ứng dụng.
- Ví dụ: Bạn cho phép người dùng upload video chỉ khi thiết bị kết nối mạng Wifi
- Bạn có thể khai báo intent filter cho hành động **ACTION_MANAGE_NETWORK_USAGE** (từ Android 4.0) để miêu tả rằng ứng dụng của bạn sẽ định nghĩa một Activity cho phép điều khiển sử dụng mạng
- **ACTION_MANAGE_NETWORK_USAGE** hiển thị setting để quản lý dung lượng mạng của ứng dụng cụ thể
- Bạn nên khai báo intent filter cho setting activity

Quản lý sử dụng mạng

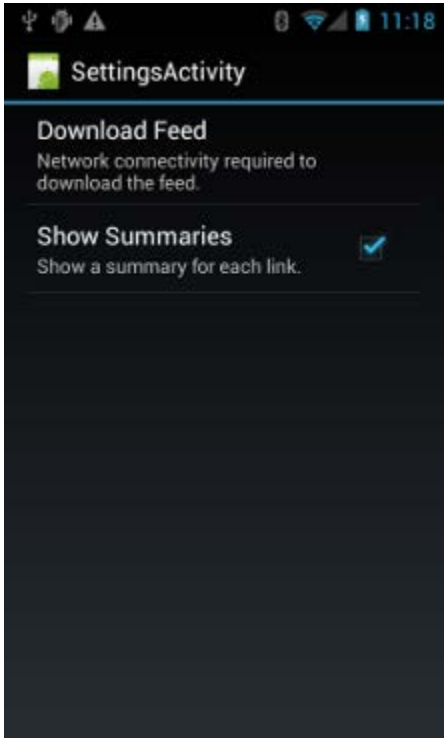
- Trong ví dụ sau, lớp SettingActivity hiển thị giao diện preference cho phép người dùng chọn khi nào download feed từ StackOverflow

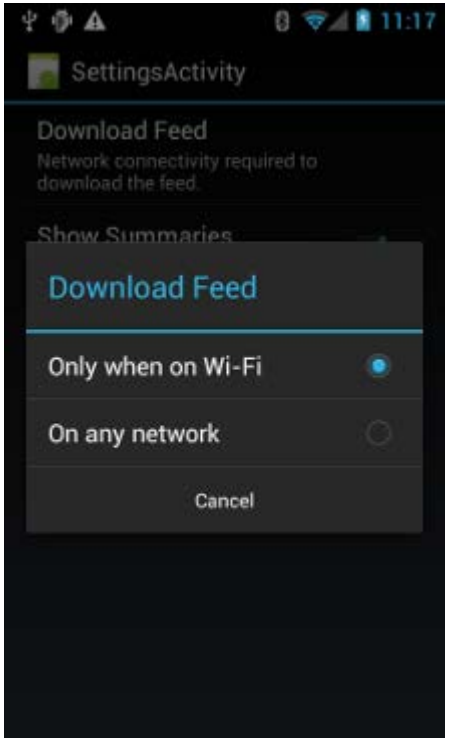
```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.android.networkusage"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk android:minSdkVersion="4"
        android:targetSdkVersion="16" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:name="com.example.android.networkusage.NetworkActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:label="SettingsActivity" android:name=".SettingsActivity">
            <intent-filter>
                <action android:name="android.intent.action.MANAGE_NETWORK_USAGE" />
                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Xây dựng Preference Activity

- Ví dụ, chúng ta sẽ hiển thị màn hình preference cho phép người dùng thiết lập các thông tin sau:
  - Hiển thị thông tin tóm tắt của mỗi XML feed entry hoặc chỉ hiển thị link của mỗi entry
  - Cho phép tải XML feed từ bất kỳ mạng nào hoặc chỉ cho phép tải khi có WiFi

Quản lý sử dụng mạng






DEMO
Ví dụ NetworkActivity

Đáp ứng thay đổi Preference

- Khi người dùng thay đổi preference trong màn hình setting, ta phải thay đổi hành vi của ứng dụng
- Trong ví dụ code sau, ứng dụng sẽ kiểm tra preference setting trong onStart(), ví dụ kiểm tra nếu setting là WiFi và thiết bị có kết nối Wifi, ứng dụng sẽ tải feed và refresh kết quả

Đáp ứng thay đổi Preference

```
public class NetworkActivity extends Activity {
    public static final String WIFI = "Wi-Fi";
    public static final String ANY = "Any";
    private static final String URL =
            "http://stackoverflow.com/feeds/tag?tagnames=android&sort=newest";

    // Whether there is a Wi-Fi connection.
    private static boolean wifiConnected = false;
    // Whether there is a mobile connection.
    private static boolean mobileConnected = false;
    // Whether the display should be refreshed.
    public static boolean refreshDisplay = true;

    // The user's current network preference setting.
    public static String sPref = null;

    // The BroadcastReceiver that tracks network connectivity changes.
    private NetworkReceiver receiver = new NetworkReceiver();

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Register BroadcastReceiver to track connection changes.
        IntentFilter filter = new IntentFilter(ConnectivityManager.CONNECTIVITY_ACTION);
        receiver = new NetworkReceiver();
        this.registerReceiver(receiver, filter);
    }
```

Đáp ứng thay đổi Preference

```
// Refreshes the display if the network connection and the
// pref settings allow it.
@Override
public void onStart() {
    super.onStart();

    // Gets the user's network preference settings
    SharedPreferences sharedPrefs = PreferenceManager.getDefaultSharedPreferences(this);

    // Retrieves a string value for the preferences. The second parameter
    // is the default value to use if a preference value is not found.
    sPref = sharedPrefs.getString("listPref", "Wi-Fi");

    updateConnectedFlags();

    // Only loads the page if refreshDisplay is true. Otherwise, keeps previous
    // display. For example, if the user has set "Wi-Fi only" in prefs and the
    // device loses its Wi-Fi connection midway through the user using the app,
    // you don't want to refresh the display--this would force the display of
    // an error page instead of stackoverflow.com content.
    if (refreshDisplay) {
        loadPage();
    }
}

@Override
public void onDestroy() {
    super.onDestroy();
    if (receiver != null) {
        this.unregisterReceiver(receiver);
    }
}
```

Đáp ứng thay đổi Preference

```
// Checks the network connection and sets the wifiConnected and mobileConnected
// variables accordingly.
private void updateConnectedFlags() {
    ConnectivityManager connMgr =
            (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);

    NetworkInfo activeInfo = connMgr.getActiveNetworkInfo();
    if (activeInfo != null && activeInfo.isConnected()) {
        wifiConnected = activeInfo.getType() == ConnectivityManager.TYPE_WIFI;
        mobileConnected = activeInfo.getType() == ConnectivityManager.TYPE_MOBILE;
    } else {
        wifiConnected = false;
        mobileConnected = false;
    }
}

// Uses AsyncTask subclass to download the XML feed from stackoverflow.com.
// This avoids UI lock up. To prevent network operations from
// causing a delay that results in a poor user experience, always perform
// network operations on a separate thread from the UI.
private void loadPage() {
    if (((sPref.equals(ANY)) && (wifiConnected || mobileConnected))
            || ((sPref.equals(WIFI)) && (wifiConnected))) {
        // AsyncTask subclass
        new DownloadXmlTask().execute(URL);
    } else {
        showErrorPage();
    }
}
```

Nhận biết sự thay đổi kết nối mạng

- NetworkReceiver là lớp con của BroadcastReceiver. Khi thay đổi kết nối mạng của thiết bị, NetworkReceiver ngăn cản action CONNECTIVITY_ACTION, xác định trạng thái của mạng và thiết lập cờ wifiConnected và mobileConnected là true hoặc false
- Thiết lập Broadcast Receiver có thể gây tốn tài nguyên hệ thống nếu Broadcast Receiver thực hiện một số lời gọi không cần thiết
- Trong ví dụ NetworkExample, NetworkReceiver được đăng ký trong onCreate() và hủy trong onDestroy().
- Khi bạn đăng ký <receiver> trong Manifest, Broadcast Receiver có thể được gọi bất kỳ khi nào, kể cả khi nó không chạy trong một thời gian dài

Nhận biết sự thay đổi kết nối mạng

- Bằng cách đăng ký Broadcast Receiver trong main activity, bạn sẽ đảm bảo Broadcast Receiver sẽ không được gọi khi người dùng rời ứng dụng
- Nếu bạn khai báo <receiver> trong AndroidManifest và biết chắc khi nào dùng nó, bạn có thể sử dụng setComponentEnabledSetting() để enable và disable khi cần thiết

Lớp NetworkReceiver trong ví dụ

```
public class NetworkReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        ConnectivityManager connMgr =
                (ConnectivityManager) context.getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo networkInfo = connMgr.getActiveNetworkInfo();
        // Checks the user prefs and the network connection. Based on the result, decides
        // whether
        // to refresh the display or keep the current display.
        // If the userpref is Wi-Fi only, checks to see if the device has a Wi-Fi connection.
        if (WIFI.equals(sPref) && networkInfo != null
                && networkInfo.getType() == ConnectivityManager.TYPE_WIFI) {
            // If device has its Wi-Fi connection, sets refreshDisplay
            // to true. This causes the display to be refreshed when the user
            // returns to the app.
            refreshDisplay = true;
            Toast.makeText(context, R.string.wifi_connected, Toast.LENGTH_SHORT).show();
            // If the setting is ANY network and there is a network connection
            // (which by process of elimination would be mobile), sets refreshDisplay to true.
        } else if (ANY.equals(sPref) && networkInfo != null) {
            refreshDisplay = true;

            // Otherwise, the app can't download content--either because there is no network
            // connection (mobile or Wi-Fi), or because the pref setting is WIFI, and there
            // is no Wi-Fi connection.
            // Sets refreshDisplay to false.
        } else {
            refreshDisplay = false;
            Toast.makeText(context, R.string.lost_connection, Toast.LENGTH_SHORT).show();
        }
    }
}
```

Phân tích dữ liệu XML

- Chọn Parser
- Phân tích Feed
- Khởi tạo Parser
- Đọc Feed
- Phân tích XML
- Bỏ qua các tag không cần thiết
- Sử dụng XML Data

Chọn Parser

- Google khuyến cáo bạn nên sử dụng XmlPullParser
- XmlPullParser cung cấp cơ chế hiệu quả và dễ bảo trì khi phân tích XML trên Android

Phân tích Feed

- Bước đầu tiên là phân tích Feed để quyết định trường (field) nào mà bạn quan tâm
- Parser sẽ trích rút dữ liệu các trường cần thiết và bỏ qua cac trường khác
- Chúng ta sẽ tìm hiểu về ví dụ NetworkActivity
- Trong ví dụ NetworkActivity, mỗi post của StackOverflow.com sẽ xuất hiện trong một feed như là một entry tag có chứa một số tag con

Cấu trúc feed của StackOverflow

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom"
xmlns:creativeCommons="http://backend.userland.com/creativeCommonsRssModule" ...">
<title type="text">newest questions tagged android - Stack Overflow</title>
...
   <entry>
   ...
   </entry
   <entry>
      <id>http://stackoverflow.com/q/9439999</id>
      <re:rank scheme="http://stackoverflow.com">0</re:rank>
      <title type="text">Where is my data file?</title>
      <category scheme="http://stackoverflow.com/feeds/tag?tagnames=android&sort=newest/tags"
term="android"/>
      <category scheme="http://stackoverflow.com/feeds/tag?tagnames=android&sort=newest/tags"
term="file"/>
      <author>
         <name>cliff2310</name>
         <uri>http://stackoverflow.com/users/1128925</uri>
      </author>
      <link rel="alternate" href="http://stackoverflow.com/questions/9439999/where-is-my-data-file" />
      <published>2012-02-25T00:30:54Z</published>
      <updated>2012-02-25T00:30:54Z</updated>
      <summary type="html">
         <p>I have an Application that requires a data file...</p>

      </summary>
   </entry>
   <entry>
   ...
   </entry>
...
</feed>
```

Khởi tạo Parser

- Bước tiếp theo là khởi tạo Parser và bắt đầu quá trình phân tích
- Trong đoạn code ví dụ sau, parser được khởi tạo mà không xử lý namespace, và sử dụng InputStream là đầu vào.
- Parser bắt đầu tiến trình phân tích bằng lời gọi nextTag() và gọi phương thức readFeed (dùng để trích xuất và xử lý dữ liệu mà ứng dụng quan tâm)

Khởi tạo Parser

```
/**
 * This class parses XML feeds from stackoverflow.com.
 * Given an InputStream representation of a feed, it returns a List of entries,
 * where each list element represents a single entry (post) in the XML feed.
 */
public class StackOverflowXmlParser {
    private static final String ns = null;

    // We don't use namespaces

    public List<Entry> parse(InputStream in) throws XmlPullParserException, IOException {
        try {
            XmlPullParser parser = Xml.newPullParser();
            parser.setFeature(XmlPullParser.FEATURE_PROCESS_NAMESPACES, false);
            parser.setInput(in, null);
            parser.nextTag();
            return readFeed(parser);
        } finally {
            in.close();
        }
    }
```

Đọc Feed

- Phương thức readFeed thực hiện công việc xử lý feed
- Tìm kiếm các element có tag là “entry” như là đầu vào của quá trình xử lý đệ quy feed
- Nếu tag không có **entry** tag, nó sẽ bỏ qua
- Khi toàn bộ feed đã được xử lý đệ quy, trả lại List chứa các entry (bao gồm cả các thành phần dữ liệu) được trích xuất từ feed
- Cuối cùng List được trả lại cho parser

Đọc Feed

```
private List<Entry> readFeed(XmlPullParser parser) throws XmlPullParserException, IOException {
    List<Entry> entries = new ArrayList<Entry>();

    parser.require(XmlPullParser.START_TAG, ns, "feed");
    while (parser.next() != XmlPullParser.END_TAG) {
        if (parser.getEventType() != XmlPullParser.START_TAG) {
            continue;
        }
        String name = parser.getName();
        // Starts by looking for the entry tag
        if (name.equals("entry")) {
            entries.add(readEntry(parser));
        } else {
            skip(parser);
        }
    }
    return entries;
}
```

Phân tích XML

- Như miêu tả trong phần phân tích Feed, bạn phải xác định các tag bạn muốn phân tích. Trong ví dụ của chúng ta, bạn phải trích xuất dữ liệu cho các tag entry và các tag con như title, link, và summary
- Tạo các phương thức sau:
  - Phương thức read cho các tag mà bạn quan tâm. Ví dụ readEntry(), readTitle()
  - Parser đọc tag từ input stream. Khi parser gặp tag tên là entry, titlte, summary, nó sẽ gọi các phương thức tương ứng. Nếu không sẽ bỏ qua tag này

Phân tích XML

- Đối với tag tittle và summary, parser gọi readText().
- Phương thức này trích xuất dữ liệu của các tag này bằng lời gọi parser.getText()

```
// Processes title tags in the feed.
private String readTitle(XmlPullParser parser) throws IOException, XmlPullParserException {
    parser.require(XmlPullParser.START_TAG, ns, "title");
    String title = readText(parser);
    parser.require(XmlPullParser.END_TAG, ns, "title");
    return title;
}
```

Phân tích XML

- Đối với tag link, parser trích xuất dữ liệu của link bằng cách kiểm tra xem link có phải link cần quan tâm không
- Sử dụng parser.getAttributeValue() để trích xuất dữ liệu của link

```
// Processes link tags in the feed.
private String readLink(XmlPullParser parser) throws IOException, XmlPullParserException {
    String link = "";
    parser.require(XmlPullParser.START_TAG, ns, "link");
    String tag = parser.getName();
    String relType = parser.getAttributeValue(null, "rel");
    if (tag.equals("link")) {
        if (relType.equals("alternate")) {
            link = parser.getAttributeValue(null, "href");
            parser.nextTag();
        }
    }
    parser.require(XmlPullParser.END_TAG, ns, "link");
    return link;
}
```

Phân tích XML

- Đối với tag entry, parser gọi readEntry(). Phương thức này sẽ phân tích các tag con của entry và trả lại đối tượng Entry với các dữ liệu thành viên là tittle, link, summary

```
// Parses the contents of an entry. If it encounters a title, summary, or link tag, hands them
// off
// to their respective &quot;read&quot; methods for processing. Otherwise, skips the tag.
private Entry readEntry(XmlPullParser parser) throws XmlPullParserException, IOException {
    parser.require(XmlPullParser.START_TAG, ns, "entry");
    String title = null;
    String summary = null;
    String link = null;
    while (parser.next() != XmlPullParser.END_TAG) {
        if (parser.getEventType() != XmlPullParser.START_TAG) {
            continue;
        }
        String name = parser.getName();
        if (name.equals("title")) {
            title = readTitle(parser);
        } else if (name.equals("summary")) {
            summary = readSummary(parser);
        } else if (name.equals("link")) {
            link = readLink(parser);
        } else {
            skip(parser);
        }
    }
    return new Entry(title, summary, link);
}
```

Phân tích dữ liệu XML

- Ví dụ phân tích dữ liệu XML sử dụng AsyncTask
- AsyncTask không xử lý dữ liệu ở main thread
- Khi kết thúc xử lý, ứng dụng cập nhật UI của main activity

Sử dụng dữ liệu XML

```
public class NetworkActivity extends Activity {
    public static final String WIFI = "Wi-Fi";
    public static final String ANY = "Any";
    private static final String URL =
            "http://stackoverflow.com/feeds/tag?tagnames=android&sort=newest";
    // Whether there is a Wi-Fi connection.
    private static boolean wifiConnected = false;
    // Whether there is a mobile connection.
    private static boolean mobileConnected = false;
    // Whether the display should be refreshed.
    public static boolean refreshDisplay = true;
    // The user's current network preference setting.
    public static String sPref = null;
    // The BroadcastReceiver that tracks network connectivity changes.
    private NetworkReceiver receiver = new NetworkReceiver();
    // Uses AsyncTask subclass to download the XML feed from stackoverflow.com.
    // This avoids UI lock up. To prevent network operations from
    // causing a delay that results in a poor user experience, always perform
    // network operations on a separate thread from the UI.
    private void loadPage() {
        if (((sPref.equals(ANY)) && (wifiConnected || mobileConnected))
                || ((sPref.equals(WIFI)) && (wifiConnected))) {
            // AsyncTask subclass
            new DownloadXmlTask().execute(URL);
        } else {
            showErrorPage();
        }
    }
```

Lớp DownloadXmlTask

- Lớp DownloadXmlTask là lớp con của Asynctask
- Lớp này gồm có hai phương thức sau:
  - doInBackground():thi hành phương thức loadXmlFromNetwork(). Nó truyền tham số feed URL. Phương thức loadXmlFromNetwork() xử lý dữ liệu feed. Khi kết thúc trả lại xâu kết quả
  - onPostExecute() trả lại xâu kết quả và hiển thị trên UI

Lớp DownloadXmlTask

```
// Implementation of AsyncTask used to download XML feed from stackoverflow.com.
private class DownloadXmlTask extends AsyncTask<String, Void, String> {

    @Override
    protected String doInBackground(String... urls) {
        try {
            return loadXmlFromNetwork(urls[0]);
        } catch (IOException e) {
            return getResources().getString(R.string.connection_error);
        } catch (XmlPullParserException e) {
            return getResources().getString(R.string.xml_error);
        }
    }

    @Override
    protected void onPostExecute(String result) {
        setContentView(R.layout.main);
        // Displays the HTML string in the UI via a WebView
        WebView myWebView = (WebView) findViewById(R.id.webview);
        myWebView.loadData(result, "text/html", null);
    }
}
```

Tổng kết nội dung bài học

- Kiểm tra kết nối mạng của thiết bị
- Quản lý sử dụng mạng của thiết bị
- Miêu tả Preference Activity
- Đáp ứng thay đổi của Preference
- Nhận biết thay đổi kết nối mạng
- Phân tích XML Data