使用單位：技術處　　姓名：

技術：Sqlite 與 Xml 下載技術查核表　　開始時間：

Sqlite 與 Xml 教學查核表

說明：此文件主要是用來學習如何從網站上下載 Xml 文件，並且將資料存入 Sqlite 資料庫中，並且可以顯示在列表中。

- 有關於此教學查核表學習前的必要條件：

  A. 對於軟體程式寫作已經有初步的了解。
  B. 已經對於 Android JAVA 撰寫有一定的能力。
  C. 已經完成進階技術_Amdroid_自訂類別畫面物件設計的查核表查核。

- 有關於本查核表的最終有價值產品：

  有信心從 Server 上面下載 xml 檔案並且可以將 xml 的資料存入 Sqlite 中，隨時可以取出使用。。

- 查核表完成的時間：4 個小時以內。

- 注意事項：請依照步驟進行，不可以任意跳過未完成的步驟。若該步驟是關於查核或是時做練習，必須要將作業(成品)，交給主管進行查核，該項目請由主管簽名。

執行步驟

第一部分：基本概念

1. 字的定義：甚麼是 Sqlite

   以下定義出自維基百科：

   SQLite 是遵守 ACID 的關聯式資料庫管理系統，它包含在一個相對小的 C 庫中。它是 D.RichardHipp 建立的公有領域項目。

   不像常見的客戶端/伺服器結構範例，SQLite 引擎不是個程式與之通訊的獨立行程，而是連線到程式中成為它的一個主要部分。所以主要的通訊協議是在程式語言內的直接 API 呼叫。這在消耗總量、延遲時間和整體簡單性上有積極的作用。整個資料庫（定義、表、索引和資料本身）都在宿主主機上儲存在一個單一的檔案中。它的簡單的設計是透過在開始一個事務的時候鎖定整個資料檔案而完成的。

2. 字的定義：甚麼是「XML」

   可延伸標記式語言（英語：eXtensible Markup Language，簡稱:XML），是一種標記式語言。標記指電腦所能理解的訊息符號，透過此種標記，電腦之間可以處理包含各種訊息的文章等。如何定義這些標記，既可以選擇國際通用的標記式語言，比如 HTML，也可以

使用像 XML 這樣由相關人士自由決定的標記式語言，這就是語言的可延伸性。

3. 說明：XML 的使用用途

XML 設計用來傳送及攜帶資料訊息，不用來表現或展示資料，HTML 語言則用來表現資料，所以 XML 用途的焦點是它說明資料是什麼，以及攜帶資料訊息。

- 豐富檔案（Rich Documents）- 自定檔案描述並使其更豐富
  - 屬於檔案為主的 XML 技術應用
  - 標記是用來定義一份資料應該如何呈現
- 後設資料（Metadata）- 描述其它檔案或網路資訊
  - 屬於資料為主的 XML 技術應用
  - 標記是用來說明一份資料的意義
- 配置文件（Configuration Files）- 描述軟體設定的參數

4 XML 的範例：

XML 定義結構、儲存訊息、傳送訊息。下例為比爾發送給賈伯斯的便條，儲存為 XML。

```
<小纸条>
    <收件人>賈伯斯</收件人>
    <发件人>比爾</发件人>
    <主题>問候</主题>
    <具体内容>嗨，過些年去找你。</具体内容>
</小纸条>
```

這 XML 文件僅是純粹的訊息標籤，這些標籤意義的展開依賴於應用它的程式。

5 甚麼是 Android App 的 Application Class:

Base class for those who need to maintain global application state. You can provide your own implementation by specifying its name in your AndroidManifest.xml's <application> tag, which will cause that class to be instantiated for you when the process for your application/package is created.

(翻譯：這是一個基礎的類別提供給需要維護整體程式狀態的設計師來使用。你可以藉由 AndroidManifest.xml 中的<Application>標籤中定義 Application 類別來提供你自己的 Application 實作部分，當這個 Application/package 被開啟時，將為被參照使用。

第二部分：設計概念說明與前置準備動作

1. 設計概念：本查核表的設計概念主要是下載網站上的 xml 檔案，並且存入 sqlite 資料庫中。最後將資料庫中的資料使用 Sql command 去進行查詢的動作。

2. 程式設計步驟：這個實作範例是以一個卡路里程式與列表，作為本查核表預定要完成的程

式項目。以下是說明本專案的製作流程：

1. 確認設計使用資源。
2. 設定與實作專案的 Application Class.
3. 實作 xml 檔案下載所需要的方法。
4. 在 Application 中實作 sqlite 的起始物件與相關方法。.
5. 將 xml 解譯並且存入 sqlite 中。
6. 將 sqlite 資料讀取出來，並且製作成列表。

3. 實作：設定一個新的專案，請將專案名稱設定為 SqliteTest
這個專案是用來製作與設計 Sqlite 與 XML 整合部分。

4. 準備：檢查網路上的 xml 檔案是否存在。
http://www.v7idea.com.tw/xml/NutritionType.xml
http://www.v7idea.com.tw/xml/NutritionXMLSmall.xml
(如果無法找到這兩個檔案，請告知輔導員)

5 實作：設定 AndroidManifest.xml 設定兩個權限：*ACCESS_NETWORK_STATE 與 INTERNET*

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

6 實作：定義 Application。請在 Application Tag 增加一個屬性：`android:name=".SqliteTestApp"` (請留意有一個.符號)

7 實作：請增加一個類別檔案SqliteTestApp，並且繼承Application。(這個類別檔案請與這個專案所設定的套件放在一起，請檢查/src/中的套件目錄) 以下是程式碼設定範例：

```
public class SqliteTestApp extends Application  {
}
```

8 增加實作的方法 OnCreate()。按下滑鼠右鍵，選擇「程式碼」，選擇「置換/實作方法」，這時會出現列表，勾選 OnCreate 方法，Eclipse 將會自動產生以下程式碼：

```
@Override
public void onCreate() {
        // TODO Auto-generated method stub
        super.onCreate();
}
```

9 實作：在 SqliteTestApp 中增加三個方法：getStringFromURL(下載網頁字串)與 checkNetworkStatus(檢查網路狀態)、String2InputStream
(將字串轉換成 ImputStream)

```
// 取得thisURL的傳回值
public String getStringFromURL(String thisURLString) {
          Log.d("httpRequest:Start", "進行Request:" +  thisURLString);
          int ifError = 0;
          String resultString = "";
          URL submitURL = null;
          ifError = 0;
          if (thisURLString == "" || thisURLString == null) ifError = 1; //
檢查目前的傳入值是否非空字串。
          if (ifError == 0) {
               try {
                    submitURL = new URL(thisURLString);
          } catch (MalformedURLException e) {
               e.printStackTrace();
               Log.d("URL_IOERROR", e.toString());
               // resultString = "URL 錯誤!!" + e.toString();
               resultString = "IOERROR";
               //alertbox("網路狀態不正常","請檢查您的裝置是否已經連上網路!!");
          }
         if (submitURL != null) {
               try {
                    // 使用HttpURLConnection 進行連結;
                    HttpURLConnection urlConn = (HttpURLConnection)
submitURL.openConnection();
                    // 得到讀取的內容;
                    InputStreamReader httpStream = new
InputStreamReader(urlConn.getInputStream());
                    // 為輸出建立Buffer;
                    BufferedReader httpBuffer = new
BufferedReader(httpStream);
                    String inputLine = null;
                    while ((inputLine = httpBuffer.readLine()) != null ) {
                    // 結果只有一行，所以讀取一行即可。
                    // 未來如果要變成一個XML，再做另外的處理。
                    resultString += inputLine + "\n";
                    }

                    httpStream.close();
                    urlConn.disconnect();
} catch (IOException e) {
```

```
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                Log.d("WaWaHealthApp:WEBERROR", e.toString());
                    resultString = "WEBERROR";
                }
            }
        }
        Log.d("APP:HttpWebRequest", resultString);
        return resultString;
    }

        // 檢查目前是否連線狀態

        public boolean checkNetworkStatus() {

            boolean ifConnected = false;
            ConnectivityManager CM = (ConnectivityManager)
    getSystemService(Context.CONNECTIVITY_SERVICE);
           NetworkInfo info = CM.getActiveNetworkInfo();

           if (info.isConnected() == true) ifConnected = true;
           return ifConnected;
       }

    public static InputStream String2InputStream(String str) {
              ByteArrayInputStream stream = null;
             try {
                 stream = new
    ByteArrayInputStream(str.getBytes("UTF-8"));

             } catch (UnsupportedEncodingException e) {
                 // TODO Auto-generated catch block
                 e.printStackTrace();
             }
              return stream;
            }
```

10 實作：新增一個新的自訂類別屬性 MyDatabaseAdapter (沒有繼承任何屬性)

11 實作：設定 private 變數

private static final String *DB_NAME* = "SqliteTestDb.db";// 資料庫命名:預設

```
private static final int DB_VERSION = 1; // 資料庫版本
private Context mContext = null; // 儲存Context物件
private SQLiteDatabase mSQLiteDatabase = null; // 儲存傳回的資料庫
private DatabaseHelper mDatabaseHelper = null;
```

12 實作：在 MyDatabaseAdapter 中增加一個子類別：DatabaseHelper 繼承 SQLiteOpenHelper

以下是 DatabaseHelper 的內容範例：

```
private   class DatabaseHelper extends SQLiteOpenHelper {
            DatabaseHelper (Context context) {
                // 當呼叫getWriteableDatabase() 或是
               // getReadableDatabase() 方法時，建立一個資料庫
                super(context, DB_NAME, null, DB_VERSION);
            }

            @Override
            public void onCreate(SQLiteDatabase db) {
                // TODO Auto-generated method stub
                // TODO Auto-generated method stub
                // 當資料庫建立時的相關程序
                Log.d(this.getClass().getSimpleName(), "當資料庫建立時的相關程序");

                String sql = "";

                // 建立 department group

                sql = "CREATE TABLE IF NOT EXISTS [nutritiontype] (" + "[_id] AUTOINC, "
                        + "[typeid] VARCHAR(50) NOT NULL, "
                        + "[name] TEXT, "
                        + "[code] TEXT, "
                        + "[serialno] TEXT, "
                        + "CONSTRAINT [sqlite_nutritiontype_1] PRIMARY KEY ([_id]))";

                db.execSQL(sql);

                Log.d("CreateiveStreetApp", "建立第一個資料表");

                sql = "CREATE TABLE IF NOT EXISTS [nutritionfood] (" + "[_id] AUTOINC, "
                    + "[nutritionid] VARCHAR(50) NOT NULL, "
                    + "[code] TEXT, "
```

```
                + "[name] TEXT, "
                + "[typeid] TEXT, "
                + "[unittype] TEXT, "
                + "[calorie] TEXT, "
                + "CONSTRAINT [sqlite_nutritionfood_1] PRIMARY KEY ([_id]))";

        db.execSQL(sql);

        Log.d("CreateiveStreetApp", "建立第二個資料表");

        }

        @Override
        public void onUpgrade(SQLiteDatabase db, int oldVersion,
                int newVersion) {
                // TODO Auto-generated method stub
                Log.d("資料庫相關作業","資料庫升級動作;");
        }
}
```

13 實作：增加 MyDatabaseAdapter 的建構子。

```
// 建構子
    public MyDatabaseAdapter(Context context) {
        mContext = context;
    }
```

14 實作：增加 MyDatabaseAdapter 的方法。

```
// 開啟資料庫的相關處理
    public void open() throws SQLException {
        mDatabaseHelper = new DatabaseHelper(mContext);
        mSQLiteDatabase = mDatabaseHelper.getWritableDatabase();
    }

    // 關閉資料庫
    public void close() {
        mDatabaseHelper.close();
    }

    public Cursor query(String sql, String[] args) {   // 執行Select
```

```
        SQLiteDatabase db = mDatabaseHelper.getWritableDatabase();
        Cursor cursor = db.rawQuery(sql, args);
        return cursor;
    }

    public SQLiteDatabase getWritableDatabase() {

        if (mDatabaseHelper != null) {
            return mDatabaseHelper.getWritableDatabase();
        } else return null;

    }

    public SQLiteDatabase getReadableDatabase() {
        if (mDatabaseHelper != null) {
            return mDatabaseHelper.getReadableDatabase();
        } else return null;
    }
```

15 實作：在 SqliteTestApp 類別中設定 Private 變數：

```
private MyDatabaseAdapter DBAdapter;
```

16 實作：在 SqliteTestApp 類別中增加兩個使用 MyDatabaseAdapter 的方法。

```
// 起始資料庫
public int initDatabase() {
        int ifSuccess = 1;
        if (DBAdapter == null) {
            // 需要重新創造一個新的DBService
            DBAdapter = new MyDatabaseAdapter(this);
            Log.d("DBAdapter:Success", "產生MyDatabaseAdapter");

        }
        return ifSuccess;
    }

// 取得自訂的DB類別物件
public MyDatabaseAdapter getDBAdapter() {
        initDatabase();
        return DBAdapter;
}
```

17 實作：取得 XML 的 Element 元件的屬性值方法，請在 Sqlite 下新增一個新的方法。

```
public String getAttributeValueByXMLTagName(Element thisElement, String attributeName) {
        String returnValue = "";
        if(thisElement != null) {
            if(thisElement.hasAttribute(attributeName)) {
              returnValue = thisElement.getAttribute(attributeName);
             }
        }
        return returnValue;
        }
```

第三部分：程式碼的實作撰寫(有關於主要的 Activity 程式部分

1. 實作：設定 Application 物件

   請在你自訂的主要 Activity 中的 OnCreate 方法，增加這行指令：

   SqliteTestApp thisApp = (SqliteTestApp) getApplicationContext();

   你將可以使用 Application 的物件。

2. 實作：為了要下載資料時可以顯示下載資料中，因此在此 Activity 中定義一個新的 Private 變數。。

   ```
   private ProgressDialog loadingProgress;
   ```

3. 實作：增加一個下載檔案的背景 Task，並且實作以下部分：

   Step 1: 背景下載 XML 文件。

   Step 2: 檢查是否下載成功。

   Step 3: 解譯 XML 文件。

   Step 4: 檢查每筆資料是否存在資料庫中，如果沒有那就開始加入資料庫中。

   範例：

```
private class downloadBackgroundTask extends AsyncTask<String, Integer, String> {
    private SqliteTestApp thisApp;

    @Override
    protected String doInBackground( String... urls) {
                // TODO Auto-generated method stub
            int count = urls.length;
              String getResult = "";
              if (count > 0) {

              Log.d("FORM_POST:","參數：" + urls[0]);
              thisApp = (SqliteTestApp) getApplicationContext();
              getResult = thisApp.getStringFromURL(urls[0]);
```

```
            }
        Log.d("LOGIN_GET_RESULT", "取得資料" + getResult);
        return getResult;
      }
    protected void onProgressUpdate(Integer... progress) {
     }
    protected void onPostExecute(String result) {     // 回傳之後後續動作;
        SqliteTestApp thisApp = (SqliteTestApp) getApplicationContext();
        if(loadingProgress.isShowing() == true) {
            loadingProgress.hide();
            loadingProgress.dismiss();
         }
         if(result.contentEquals("WEBERROR")) {
          boolean ifSuccess = false;
         } else {
          // 準備要放有關於產生資料的相關程序。
         DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
          DocumentBuilder db;
          try {
             db = dbf.newDocumentBuilder();
             // Document doc = db.parse(mainMapXMLString);
             InputStream thisStream = thisApp.String2InputStream(result);
             Document doc = db.parse(thisStream);
             Log.d("產生主街區物件","已經整理出一個XML Document");
             NodeList thisNode = doc.getElementsByTagName("Food_NutritionType");
            if(thisNode.getLength() > 0) {
               for (int i = 0; i < thisNode.getLength(); i ++ ) {
                    Node singleNode = thisNode.item(i);
                    if (singleNode.getNodeType() == Node.ELEMENT_NODE) {
                        Element thisElement = (Element) singleNode;
                        String typeid = thisApp.getAttributeValueByXMLTagName(thisElement,
"NutritionTypeID");
                        String name = thisApp.getAttributeValueByXMLTagName(thisElement,
"NutritionTypeName");
                         String code = thisApp.getAttributeValueByXMLTagName(thisElement,
"NutritionCode");
                         String serialno = thisApp.getAttributeValueByXMLTagName(thisElement,
"SerialNo");
                        if(typeid != null && typeid.length() > 0) {
                        // 表示有取得值
```

```
                        MyDatabaseAdapter DBAdapter = null;
                        DBAdapter = thisApp.getDBAdapter();
                        String sql = "";
                        boolean ifExist = false;
                        if(DBAdapter != null) {
                            DBAdapter.open();
                            SQLiteDatabase thisDatabase = DBAdapter.getWritableDatabase();
                            sql = "select * from nutritiontype where typeid = ?";
                            Cursor resultCur = DBAdapter.query(sql, new String[]{typeid});
                            if (resultCur != null) {
                                if(resultCur.moveToFirst()) {
                                    ifExist = true;
                                    Log.d("尋找是否已經下載資料","曾經下載過資料了");
                                }
                            }
                            resultCur.close();
                            if(ifExist == false) {
                                sql = "INSERT INTO nutritiontype(typeid, name, code, "
                                        + "serialno) values(?,?,?,?)";
                                thisDatabase.execSQL(sql,
                                        new String[]{typeid, name, code, serialno});
                            }
                            thisDatabase.close();
                            DBAdapter.close();
                        }
                    }
                }
            }
        }

    }catch (ParserConfigurationException e1) {
                // TODO Auto-generated catch block
                e1.printStackTrace();
     } catch (SAXException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
     } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
      }
```

```
}
}
}
```

4 在這個 Activity 中 OnCreate 繼續以下的程序：

- 檢查是否已經下載過資料
- 如果已經下載過，顯示一個 Toast，說明已經下載過資料了
- 如果還沒有下載過，那就開始下載的背景程序。

5 步驟 4 的執行範例：

```
MyDatabaseAdapter DBAdapter = null;
DBAdapter = thisApp.getDBAdapter();
boolean ifExist = false;
String sql = "";
if(DBAdapter != null) {
    Log.d("DBAdapter","DBAdapter存在並且檢查");
    DBAdapter.open();   // 開啟資料庫
    sql = "select * from nutritiontype";
    Cursor resultCur = DBAdapter.query(sql, null);
    if (resultCur != null) {
        if(resultCur.moveToFirst()) {
            ifExist = true;
            Log.d("尋找是否已經下載資料","曾經已經下載過資料了");
            Toast.makeText(this, "之前已經下載過資料庫的資料了",
                    Toast.LENGTH_LONG).show();
        }
    } else {
        Log.d("resultCur","無法產生cursor");
    }
    resultCur.close();
    DBAdapter.close();
}
if(ifExist == false) {   // 表示資料還沒有下載
    // 要先下載資料
    loadingProgress = new ProgressDialog(this);
    loadingProgress.setProgressStyle(ProgressDialog.STYLE_SPINNER);
    loadingProgress.setMessage("連結主機，下載營養表類別資料中.....");
    loadingProgress.setIndeterminate(false);
    loadingProgress.setCancelable(false);
    loadingProgress.show();
```

```
        String dataUrl = "http://www.v7idea.com.tw/xml/NutritionType.xml";
        Log.d("MainActivity","重新下載資料!");
        downloadBackgroundTask thisTask = new downloadBackgroundTask();
        thisTask.execute(dataUrl);
    }
```

第四部分：顯示下載的資料 (運用自訂物件來製作列表)

1 使用自訂物件的查核表去製作一個新的類別物件 NutritionType，並且建立一個建構子，傳入 id 與 name 去產生畫面物件。

2. 實作：在主Activity中設計一個新的方法去顯示列表，這個方法定名為：generateNutritionTypeList(); _____

3 實作：檢查該列表是否有產生。 _____

4. 實作：請講師檢查。 _____

5. 此階段為基礎階段的完成。 _____

第五部分：延伸練習 (進階實作)

1. 實作：下載 http://www.v7idea.com.tw/xml/NutritionXMLSmall.xml 資料並且存入資料表中。 _____

2 實作：NutritionType 列表畫面中點選其中一個物件，將會自動連到第二個畫面去產生出這個 Type 的食物列表。 _____

3 實作：點選某個食物，將會顯示那個食物的相關資訊。 _____

4 實作：請講師檢查這個部分 _____