

製作記事APP

劉治廷

下載教材

下載教材

- 說明

- https://github.com/silencecork/shuworkshop_early2018

- 進度

- https://github.com/silencecork/shuworkshop_early2018/branches/all

製作記事APP

建立ACTIVITY、LISTVIEW、ADAPTER及測試資料

目標

- ArrayList的基本使用
- 建立Activity
- 建立ListView與ArrayAdapter

ARRAY LIST基本使用

- 建立一個ArrayList

```
ArrayList<Object> list = new ArrayList<Object>();
```

- 放入資料

```
list.add(object)
```

- 取得資料

```
list.get(index)
```

- 刪除資料

```
list.remove(index)
```

- 取得資料數目

```
list.size()
```

LISTVIEW與ARRAY ADAPTER

- 使用方式重點提示

使用findViewById找到ListView

```
mListView = findViewById(R.id.listview);  
mAdapter = new ArrayAdapter(this, android.R.layout.simple_list_item_1, arraylist);  
mListView.setAdapter(mAdapter);
```

LISTVIEW與ARRAY ADAPTER

- 使用方式重點提示

```
mListView = findViewById(R.id.listview);  
mAdapter = new ArrayAdapter(this, android.R.layout.simple_list_item_1, arraylist);  
mListView.setAdapter(mAdapter);
```

使用android內建的layout

要呈現的列表資料

LISTVIEW與ARRAY ADAPTER

- 使用方式重點提示

```
mListView = findViewById(R.id.listview);  
mAdapter = new ArrayAdapter(this, android.R.layout.simple_list_item_1, arraylist);  
mListView.setAdapter(mAdapter);
```

呼叫ListView setAdapter讓
Adapter啟用

製作記事APP

建立第二個ACTIVITY，並設定ACTIONMENU

目標

- 如何建立第二個Activity
- 增加ActionMenu
- 使用Log

建立ACTION MENU

- 首先建立Action Menu的設定檔，擺放在res/menu下

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:app="http://schemas.android.com/apk/res-auto"
      xmlns:android="http://schemas.android.com/apk/res/android">

  <item
    android:id="@+id/menu_add"
    android:icon="@drawable/ic_action_add"
    android:title="Item"
    app:showAsAction="always" />

</menu>
```

1. android:id表示選單的識別字，方便在程式中找到
2. android:icon 選項圖示

建立ACTION MENU

- 首先建立Action Menu的設定檔，擺放在res/menu下

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:app="http://schemas.android.com/apk/res-auto"
      xmlns:android="http://schemas.android.com/apk/res/android">

  <item
    android:id="@+id/menu_add"
    android:icon="@drawable/ic_action_add"
    android:title="Item"
    app:showAsAction="always" />

</menu>
```

app:showAsAction 表示此選單
要顯示為Action Menu

建立ACTION MENU

- 在Activity中讀取選單資源檔

複寫Activity的
onCreateOptionsMenu

```
@Override  
public boolean onCreateOptionsMenu(Menu menu) {  
    getMenuInflater().inflate(R.menu.main_activity_menu, menu);  
    return true;  
}
```

建立ACTION MENU

- 在Activity中讀取選單資源檔

```
@Override  
public boolean onCreateOptionsMenu(Menu menu) {  
    getMenuInflater().inflate(R.menu.main_activity_menu, menu);  
    return true;  
}
```

getMenuInflater()
可以取得專門用來展開menu的物件

建立ACTION MENU

- 在Activity中讀取選單資源檔

```
@Override  
public boolean onCreateOptionsMenu(Menu menu) {  
    getMenuInflater().inflate(R.menu.main_activity_menu, menu);  
    return true;  
}
```

設定展開後的選單物件要加入
哪個menu

設定要展開的選單資源檔

建立ACTION MENU

- 在Activity中確定哪個Action Menu被按下

複寫Activity的
onOptionsItemSelected

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    if (item.getItemId() == R.id.menu_add) {
        return true;
    }
    return super.onOptionsItemSelected(item);
}
```

建立ACTION MENU

- 在Activity中確定哪個Action Menu被按下

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    if (item.getItemId() == R.id.menu_add) {
        return true;
    }
    return super.onOptionsItemSelected(item);
}
```

使用ItemId來確定點下
的是哪個選項

對應分支

03.change_activity

製作記事APP

切換ACTIVITY

目標

- 切換Activity
- 由第二個Activity帶數值回第一個Activity
- 更新ListView

切換ACTIVITY

- 如何使用Intent啟動一個要求回傳結果的Activity

建立新的Intent
Intent可以指定Android系統啟動
其他Activity

```
Intent intent = new Intent(this, AddItemActivity.class);  
startActivityForResult(intent, ADD_ITEM);
```

切換ACTIVITY

- 如何使用Intent啟動一個要求回傳結果的Activity

第一個參數表示由目前的Activity
切換到參數二的Activity

```
Intent intent = new Intent(this, AddItemActivity.class);  
startActivityForResult(intent, ADD_ITEM);
```

切換ACTIVITY

- 如何使用Intent啟動一個要求回傳結果的Activity

```
Intent intent = new Intent(this, AddItemActivity.class);  
startActivityForResult(intent, ADD_ITEM);
```

startActivityForResult()表示切換Activity，並要求另一個Activity要回傳資料

切換ACTIVITY

- 另一個Activity帶回資料到原有Activity

```
Intent data = new Intent();  
data.putExtra("update", true);  
setResult(RESULT_OK, data);
```

建立一個Intent
負責帶資料回原有Activity

切換ACTIVITY

- 另一個Activity帶回資料到原有Activity

看需要帶回甚麼資料
使用Intent.putExtra()擺到
Intent中

```
Intent data = new Intent();  
data.putExtra("update", true);  
setResult(RESULT_OK, data);
```

切換ACTIVITY

- 另一個Activity帶回資料到原有Activity

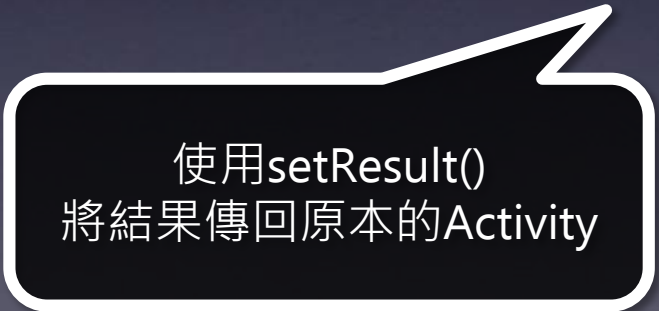
```
Intent data = new Intent();  
data.putExtra("update", true);  
setResult(RESULT_OK, data);
```

語法：
Intent.putExtra(索引值, 數值)

切換ACTIVITY

- 另一個Activity帶回資料到原有Activity

```
Intent data = new Intent();  
data.putExtra("update", true);  
setResult(RESULT_OK, data);
```



使用setResult()
將結果傳回原本的Activity

切換ACTIVITY

- 原有Activity接收回傳資料

複寫Activity的
onActivityResult

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == ADD_ITEM && resultCode == RESULT_OK) {
        boolean needUpdate = data.getBooleanExtra("update", false);
        .....
    }
}
```

切換ACTIVITY

- 原有Activity接收回傳資料

第一個參數是呼叫
startActivityForResult時帶的第二個參數

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == ADD_ITEM && resultCode == RESULT_OK) {
        boolean needUpdate = data.getBooleanExtra("update", false);
        .....
    }
}
```

切換ACTIVITY

- 原有Activity接收回傳資料

第二個參數是另一個Activity呼叫
setResult()時的第一個參數

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == ADD_ITEM && resultCode == RESULT_OK) {
        boolean needUpdate = data.getBooleanExtra("update", false);
        .....
    }
}
```

切換ACTIVITY

- 原有Activity接收回傳資料

第三個參數是另一個Activity呼叫
setResult()時的第二個參數

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == ADD_ITEM && resultCode == RESULT_OK) {
        boolean needUpdate = data.getBooleanExtra("update", false);
        .....
    }
}
```

切換ACTIVITY

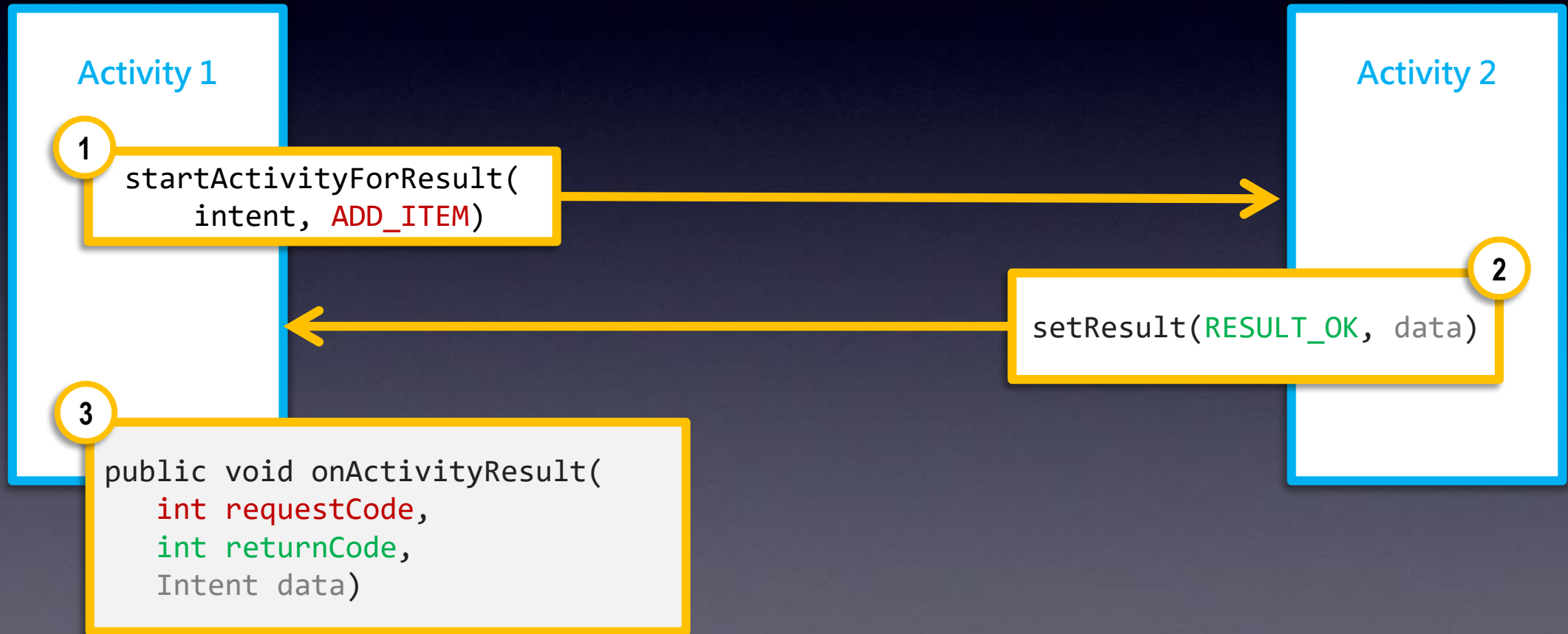
- 原有Activity接收回傳資料

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == ADD_ITEM && resultCode == RESULT_OK) {
        boolean needUpdate = data.getBooleanExtra("update", false);
        .....
    }
}
```

Intent.getStringExtra()
帶入索引值
可以由回傳的Intent中取得另一個
Activity回傳的資料

切換ACTIVITY

- 流程對應表 (注意顏色的對應)



更新LISTVIEW

- 使用Adapter來更新ListView

呼叫clear清空目前的內容

```
mAdapter.clear();  
mAdapter.addAll(ItemManager.getAllItem());  
mAdapter.notifyDataSetChanged();
```

更新LISTVIEW

- 使用Adapter來更新ListView

呼叫ArrayAdapter的
addAll()，放入新資料

```
mAdapter.clear();  
mAdapter.addAll(ItemManager.getAllItem());  
mAdapter.notifyDataSetChanged();
```

更新LISTVIEW

- 使用Adapter來更新ListView

```
mAdapter.clear();  
mAdapter.addAll(ItemManager.getAllItem());  
mAdapter.notifyDataSetChanged();
```

呼叫Adapter的
notifyDataSetChange()
更新資料

製作記事APP

LISTVIEW點擊事件及更多ACTIVITY帶數值

目標

- 使用ListView OnItemClickListener
- Activity間互帶數值

ONITEMCLICKLISTENER

ListView.setOnItemClickListener
設定ListView的點擊事件接收器

```
mListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {  
    @Override  
    public void onItemClick(AdapterView<?> parent, View v, int pos, long id) {  
    }  
});
```

ONITEMCLICKLISTENER

```
mListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {  
    @Override  
    public void onItemClick(AdapterView<?> parent, View v, int pos, long id) {  
    }  
});
```

用戶點到的索引值

當用戶點下ListView的項目時
註冊的事件接收器中的
onItemClick就會被呼叫到

對應分支

05.listview_click_delete_item

製作記事APP

LISTVIEW長按事件

目標

- 使用ListView OnItemLongClickListener
- 顯示AlertDialog

ONITEMLONGCLICKLISTENER

ListView.setOnItemClickListener
設定ListView的長按事件接收器

```
mListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {  
    @Override  
    public boolean onItemClick(AdapterView<?> parent, View v, int pos, long id) {  
        return true;  
    }  
});
```

ONITEMLONGCLICKLISTENER

```
mListView.setOnItemLongClickListener(new AdapterView.OnItemLongClickListener() {  
    @Override  
    public boolean onItemClick(AdapterView<?> parent, View v, int pos, long id)  
    {  
        return true;  
    }  
});
```

用戶長按的索引值

當用戶長按ListView的項目時
註冊的事件接收器中的
onItemClick就會被呼叫到

製作記事APP

使用第三方函式庫

目標

- 在Gradle中使用第三方函式庫Sugar ORM
 - <http://satyan.github.io/sugar/>
- 讓資料存入SQLiteDatabase

新增函式庫

- 在app的build.gradle中加入以下

```
dependencies {  
    implementation fileTree(dir: 'libs', include: ['*.jar'])  
    implementation 'com.android.support:appcompat-v7:27.1.1'  
    implementation 'com.android.support.constraint:constraint-layout:1.1.0'  
    testImplementation 'junit:junit:4.12'  
    androidTestImplementation 'com.android.support.test:runner:1.0.2'  
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'  
    compile 'com.github.satyan:sugar:1.5'  
}
```

加入函式庫的方法

利用設定建立資料庫

- 在AndroidManifest.xml加入下述

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/AppTheme"
    android:name="com.orm.SugarApp">
    <meta-data android:name="DATABASE" android:value="note.db" />
    <meta-data android:name="VERSION" android:value="2" />
    <meta-data android:name="QUERY_LOG" android:value="true" />
    <meta-data android:name="DOMAIN_PACKAGE_NAME" android:value="tw.edu.shu.im.noteapp"/>
    .....
</application>
```

將Application改用SugarApp
設定後函式庫會在App啟動時自動配置

利用設定建立資料庫

- 在AndroidManifest.xml加入下述

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/AppTheme"
    android:name="com.orm.SugarApp">
    <meta-data android:name="DATABASE" android:value="note.db" />
    <meta-data android:name="VERSION" android:value="2" />
    <meta-data android:name="QUERY_LOG" android:value="true" />
    <meta-data android:name="DOMAIN_PACKAGE_NAME" android:value="tw.edu.shu.im.noteapp"/>
    .....
</application>
```

設定建立資料庫的檔案名稱

利用設定建立資料庫

- 在AndroidManifest.xml加入下述

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/AppTheme"
    android:name="com.orm.SugarApp">
    <meta-data android:name="DATABASE" android:value="note.db" />
    <meta-data android:name="VERSION" android:value="2" />
    <meta-data android:name="QUERY_LOG" android:value="true" />
    <meta-data android:name="DOMAIN_PACKAGE_NAME" android:value="tw.edu.shu.im.noteapp"/>
    .....
</application>
```

建立資料庫的版本
只要存的資料有更動
版本就生一個號碼

利用設定建立資料庫

- 在AndroidManifest.xml加入下述

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/AppTheme"
    android:name="com.orm.SugarApp">
    <meta-data android:name="DATABASE" android:value="note.db" />
    <meta-data android:name="VERSION" android:value="2" />
    <meta-data android:name="QUERY_LOG" android:value="true" />
    <meta-data android:name="DOMAIN_PACKAGE_NAME" android:value="tw.edu.shu.im.noteapp"/>
    .....
</application>
```

本項目就填App的package

物件改變為可做資料庫操作

- 要儲存的物件，改繼承SugarRecord

要繼承SugarRecord

```
import com.orm.SugarRecord;
public class Item extends SugarRecord {
    public String title;
    public String note;

    @Override
    public String toString() {
        return title;
    }
}
```

物件改變為可做資料庫操作

- 繼承SugarRecord後的物件，就可以進行操作

- 新增

```
Item item = new Item();  
item.title = title;  
item.note = note;  
item.save();
```

繼承SugarRecord的物件
呼叫save()即可儲存

- 取得全部資料

```
ArrayList<Item> itemList = new ArrayList<Item>();  
Iterator<Item> allItemsItr = Item.findAll(Item.class);  
while (allItemsItr.hasNext()) {  
    Item item = allItemsItr.next();  
    itemList.add(item);  
}
```

繼承SugarRecord的物件
使用findAll可拿到全部

物件改變為可做資料庫操作

- 繼承SugarRecord後的物件，就可以進行操作

- 刪除

```
Item item = itemList.get(index);  
item.delete();
```

繼承SugarRecord的物件
使用delete便可從資料庫刪除

- 修改

```
Item item = itemList.get(index);  
item.title = newTitleValue;  
item.note = newNoteValue;  
item.save();
```

繼承SugarRecord的物件
取得後更新完數值，再次呼叫
save即可更新回資料庫

Q & A