Lab5
Fragment

本節目的：

- 了解什麼是 Fragment，以及 Fragment 與 Activity 的關聯。
- 認識 Activity 與 Fragment 的生命週期。
- 認識 Android 的滑頁（ViewPager）。

5.1 片段（Fragment）

片段（Fragment）是活動（Activity）中的一部分使用者介面，一個 Activity 可以擁有數個 Fragment，你可以將 Fragment 視為 Activity 中的子 Activity，可以透過 Activity 去新增或移除它們，每頁 Fragment 皆擁有自己的生命周期與監聽事件，但 Fragment 必須依賴於 Activity，因此 Activity 的生命週期會直接影響到 Fragment 的生命週期。

如下圖的無線通訊頁面，就是基於 Fragment 實現出來的，圖 5-1 通訊、好友與加好友頁面都是一個獨立的 Fragment，擁有自己的布局與監聽事件，但他們共享無線通訊頁面的 Activity。

圖 5-1 通訊 fragment（左）、好友 fragment（中）、加好友 fragment（右）

5.1.1 生命週期

前面我們提到了片段（Fragment）與活動（Activity）的生命週期息息相關，究竟甚麼是生命週期？現今使用者大多習慣一邊聽音樂一邊滑臉書，每多執行一個應用程式，就會多耗費一些記憶體。然而手機裡的記憶體是有限的，當同時執行過多的程式，或是關閉的程式沒有正確釋放資源，系統時就會變得緩慢而不穩定。

為了解決這個問題，Android 在系統設計中引入了**生命週期（Life Cycle）**機制，並提供了幾種對應的 Callback 函式（onCreate、onDestroy…等），這些函式只有在特定情況下被執行（創建、銷毀…等），同時也簡化了 APP 的開發流程，讓使用者能更靈活的應用。

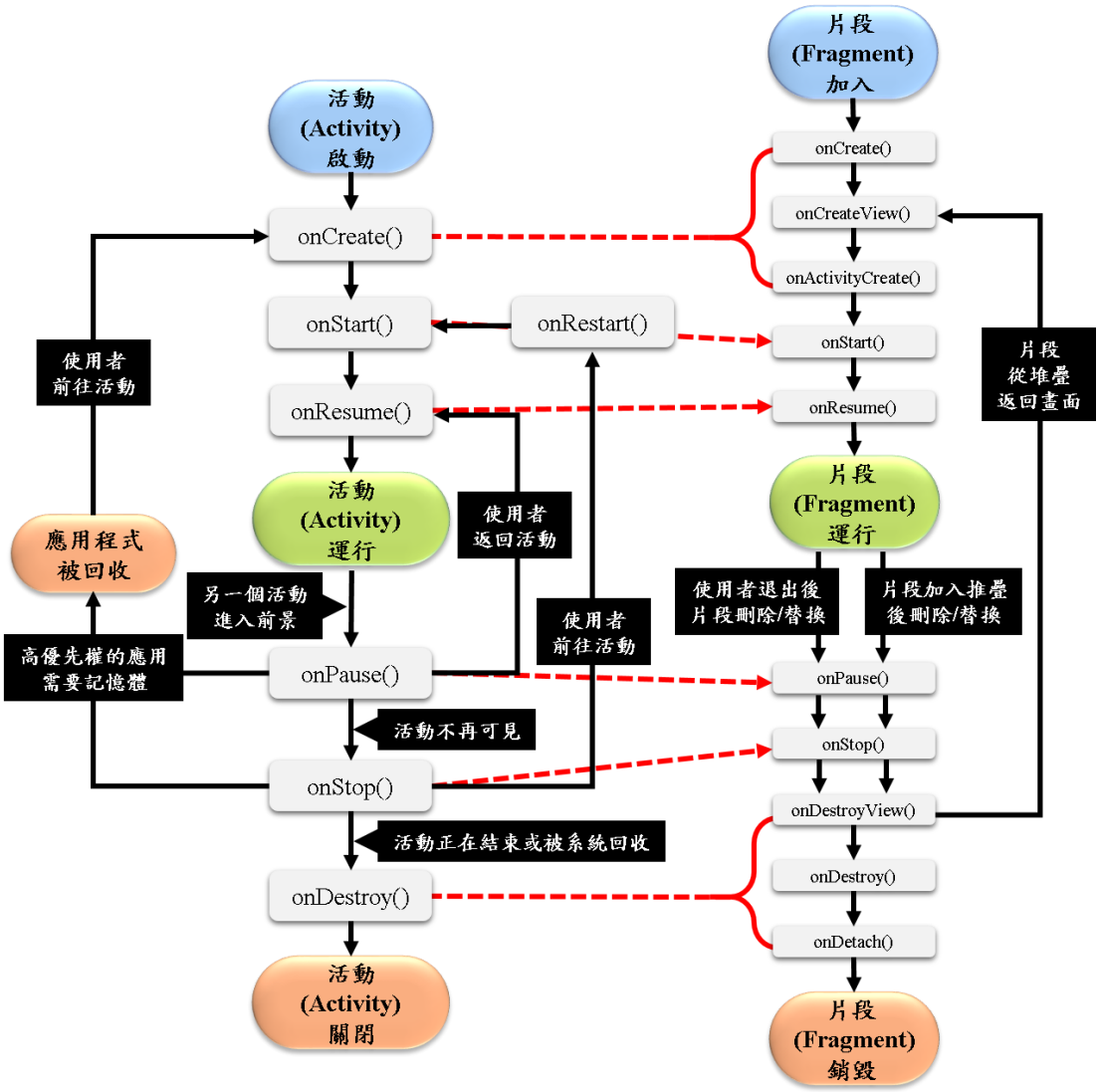

圖 5-2 Activity 與 Fragment 生命週期對照

圖 5-2 左側為 Activity 的生命週期，右側為 Fragment 的生命週期，生命週期定義了 Android 元件（Activity、Fragment、Service…等）在工作階段的任務。例如:當 Activity 切換到另一個 Activity 時，會進入暫停（onPause）並在畫面不可見後進入停止（onStop）；當 Fragment 被移除或被其他 Fragment 取代時，則會依序進入到暫停（onPause）、停止（onStop）與銷毀畫面（onDestroyView）。

從圖 5-2 對照中可以發現，Activity 與 Fragment 擁有類似的生命週期，如創建（onCreate）、開始（onStart）銷毀與（onDestroy）…等，不同點在於 Fragment 是 Activity 中的一部分使用者介面，所以多了創建畫面（onCreateView）、與銷毀畫面（onDestroyView）…等處理畫面的生命週期，以下介紹各生命週期函式的功用：

- onCreate()：初始化頁面並定義 UI（Fragment 則在 onCreateView()中定義 UI）。
- onRestart()：當使用者返回頁面時呼叫。
- onStart()：在 onCreate()後呼叫，或在 onRestart()之後呼叫。
- onResume()：在 onStart()與 onPause()後呼叫，使頁面與使用者開始互動。
- onPause()：使用者離開頁面，通常在此階段將資料保存，以便返回後繼續使用（畫面為可見狀態）。
- onStop()：當頁面切換導致畫面不再為可見狀態時呼叫。
- onDestroy()：在頁面被回收前呼叫，不建議在此做資料保存，因為系統即將釋放被占用的資源。
- onCreateView()：系統會呼叫這個方法來建立與 Fragment 相關聯的 UI。
- onActivityCreated()：在 onCreateView()後被呼叫，Fragment 與 Activity 已建立關聯。
- onDestroyView()：當畫面移除與 Fragment 相關聯的 UI 時呼叫。
- onDetach()：當 Activity 與 Fragment 解除關聯時會呼叫這個方法。

說明

Fragment 依賴於 Activity，因此 Fragment 必須等待 Activity 創建後才能創建，而當 Activity 進入暫停（onPause）、停止（onStop）與銷毀（onDestroy）階段時，Fragment 也會觸發圖 5-2 中對應的生命週期，在開發時要特別注意。

5.1.2 產生 Fragment

要產生出一個新的 Fragment，首先選擇「File→New→Kotlin File/Class」，來產生出空白的 Class，如圖 5-3 所示。

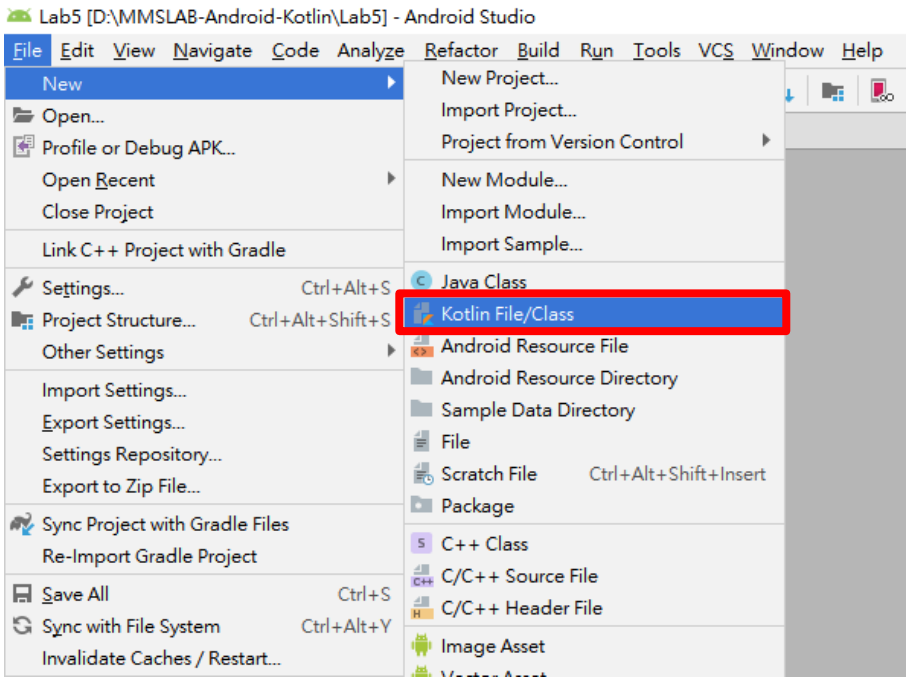

圖 5-3 點擊 File 建立 Kotlin Class

選擇後，在視窗中輸入 File 的名稱與類型，如圖 5-4 所示。在這裡我們要創建第一個 Fragment 類別命名為 FirstFragment。

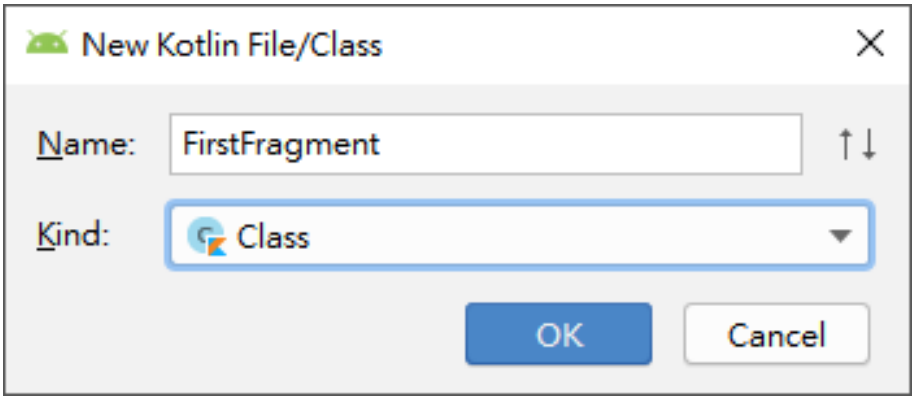

圖 5-4 建立新的 Class 檔

按下「OK」後，可以於左邊目錄中看到系統幫你產生出 FirstFragment，右邊則是一個空白的 FirstFragment 類別，如圖 5-5 所示。

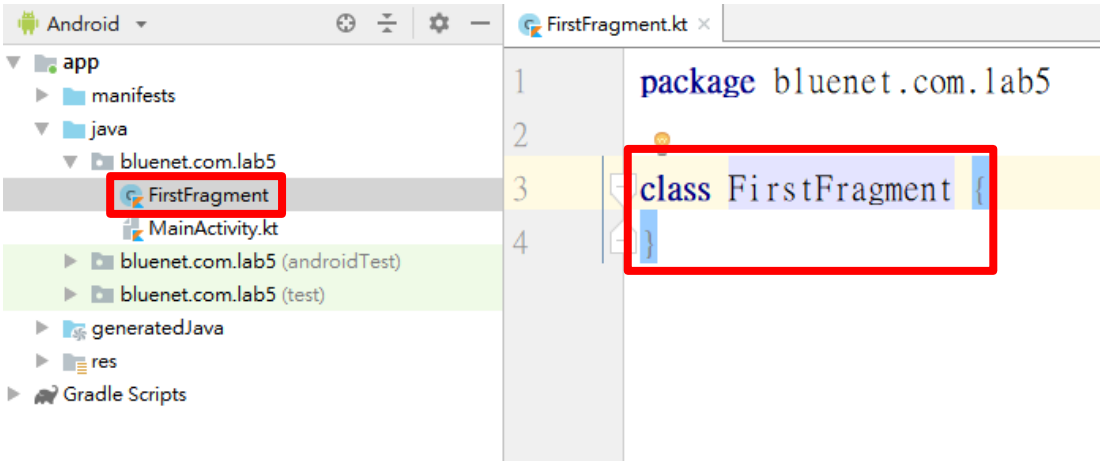

圖 5-5 產生新的類別

說明

Fragment 是依賴於 Activity 中，所以不用在 AndroidManifest.xml 中額外加入 Fragment 的資訊。

接著，我們要開始撰寫 FirstFragment，程式碼如下：

```
class FirstFragment : Fragment() { //繼承 supportv4 的 Fragment 類別
    //定義 Fragment 的畫面
    override fun onCreateView(inflater: LayoutInflater,
container: ViewGroup?, savedInstanceState: Bundle?): View? {
        return inflater.inflate(R.layout.fragment_first,
                                              container, false)
    }

    override fun onActivityCreated(savedInstanceState: Bundle?)
{
        super.onActivityCreated(savedInstanceState)
        ... //主程式
    }
}
```

說明

在 Fragment 中，定義畫面是在 onCreateView()中進行，主程式則建議寫在 onActivityCreated()，確保畫面的元件已經與畫面連接。

5.1.3 滑頁（ViewPager）

ViewPager（滑頁）是 Android 應用程式中的一種布局管理元件，允許使用者透過手勢左右滑動來切換頁面，**錯誤! 找不到參照來源。**5-1 就是一個 ViewPager 的實例，ViewPager 必須搭配對應的 PagerAdapter 類別來實現滑頁功能，本章節使用 FragmentPagerAdapter 實作滑頁，程式碼如下：

```
class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        //建立 FragmentPagerAdapter 物件
        val adapter = ViewPagerAdapter(supportFragmentManager)
        //連接 Adapter，讓畫面(Fragment)與 ViewPager 建立關聯
        viewPager.adapter = adapter
    }
}

class ViewPagerAdapter(fm: FragmentManager) :
FragmentPagerAdapter(fm) { //繼承 FragmentPagerAdapter 類別
    //回傳對應位置的 Fragment，決定頁面的呈現順序
    override fun getItem(position: Int) = when(position){
        0 ->FirstFragment() //第一頁要呈現的 Fragment
        1 ->SecondFragment() //第二頁要呈現的 Fragment
        else ->ThirdFragment() //第三頁要呈現的 Fragment
    }
    //回傳 Fragment 頁數
    override fun getCount() = 3
}
```

如圖 5-6 所示，向左滑動畫面切換到第二頁，ViewPager 提供快速切換頁面的功能，並且預先載入前後的頁面。

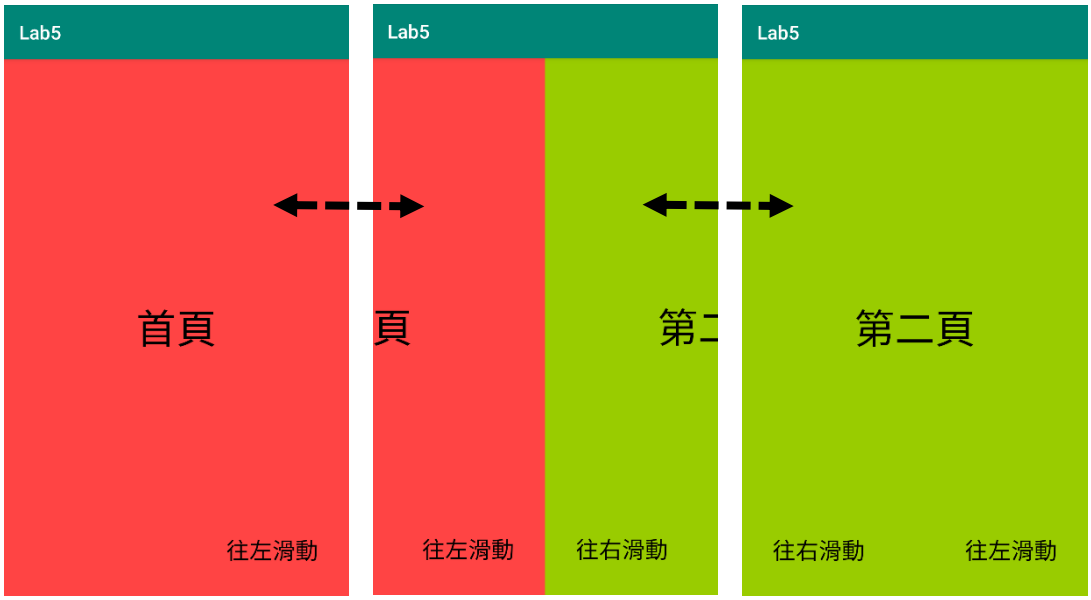

圖 5-6 從首頁（左）向左滑動切換到第二頁（右）

5.2 觀察生命週期

- 本次範例實作一個帶有滑動頁面功能的 APP 如下**錯誤! 找不到參照來源。**所示，擁有 3 個不同布局的 Fragment（首頁 FirstFragment、第二頁 SecondFragment、第三頁 ThirdFragment）。
- 使用 ViewPager 實現左右滑動切換頁面。
- 觀察 Activity 與 3 個 Fragment 的生命週期變動。

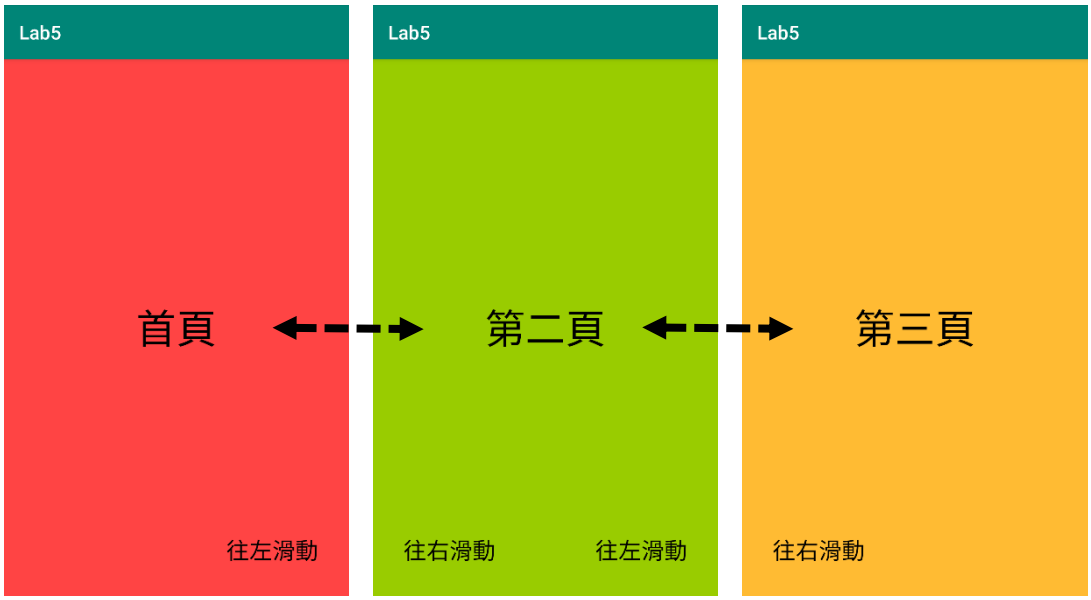

圖 5-7 FirstFragment（左）、SecondFragment（中）、ThirdFragment（右）

5.2.1 滑頁布局設計

Step1 建立新專案，以及圖 5-8 對應的 class 與 xml 檔。

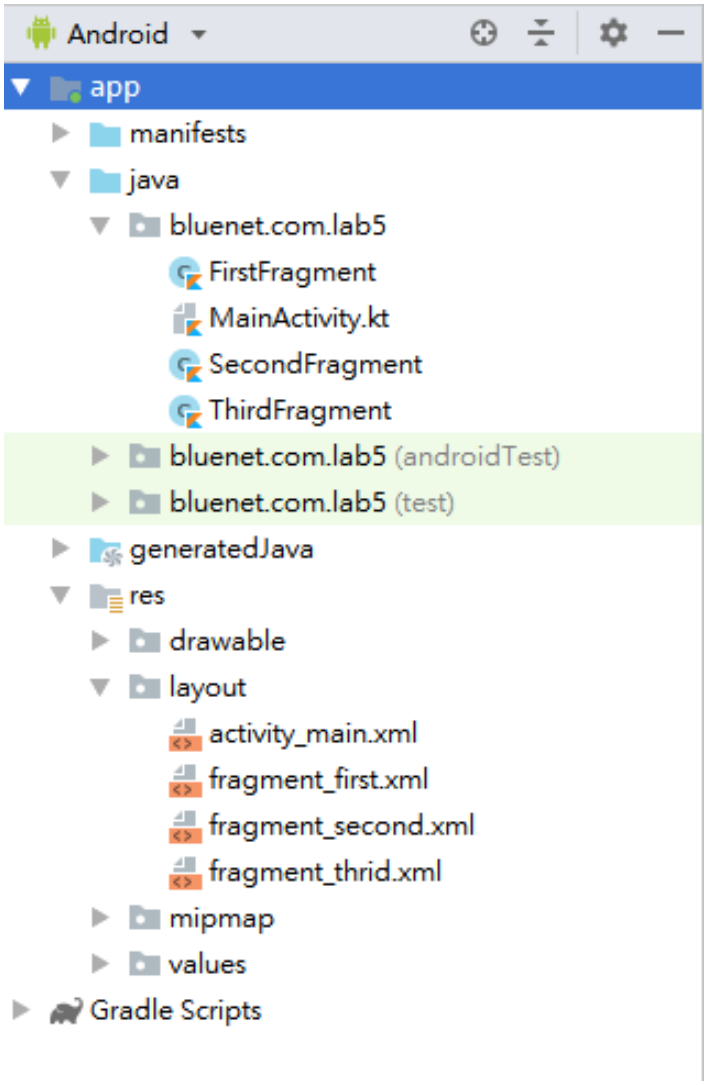

圖 5-8 Lab5 專案架構

Step2 繪製 activity_main.xml 檔，如圖 5-9 所示。

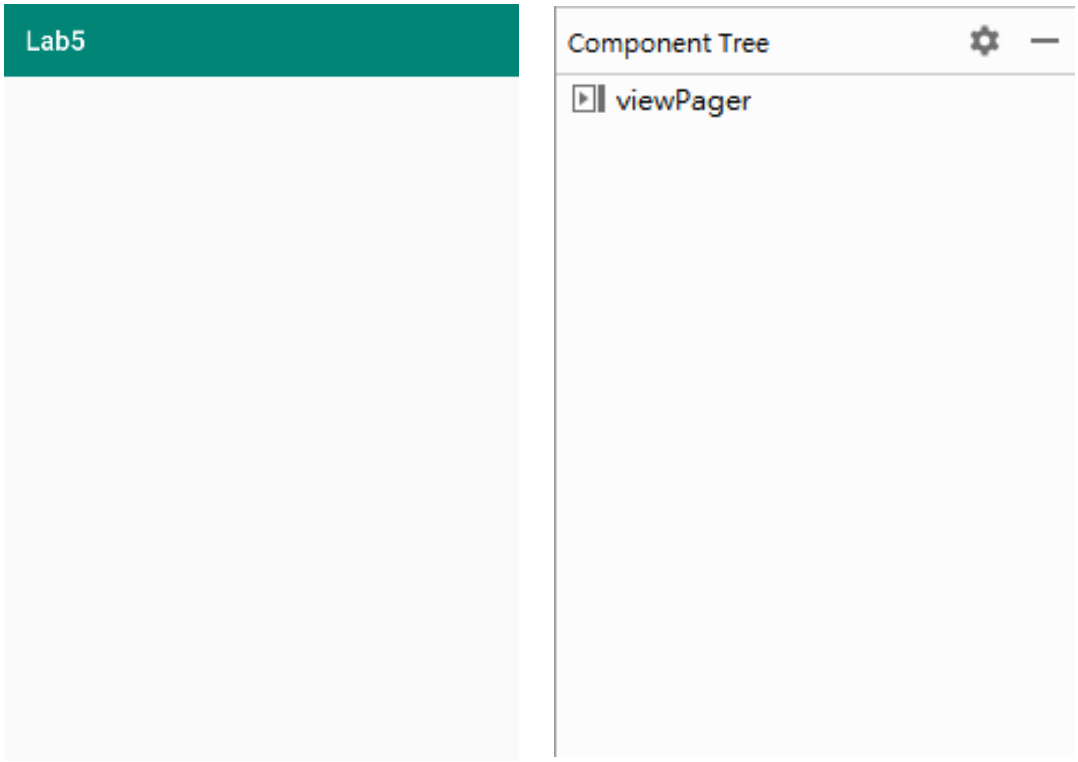

圖 5-9 MainActivity 預覽畫面（左）與布局元件樹（右）

對應的 xml 如下：

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v4.view.ViewPager
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/viewPager"
        android:layout_width="match_parent"
        android:layout_height="match_parent">
</android.support.v4.view.ViewPager>
```

Step3 繪製 fragment_first.xml 檔如圖 5-10 所示。

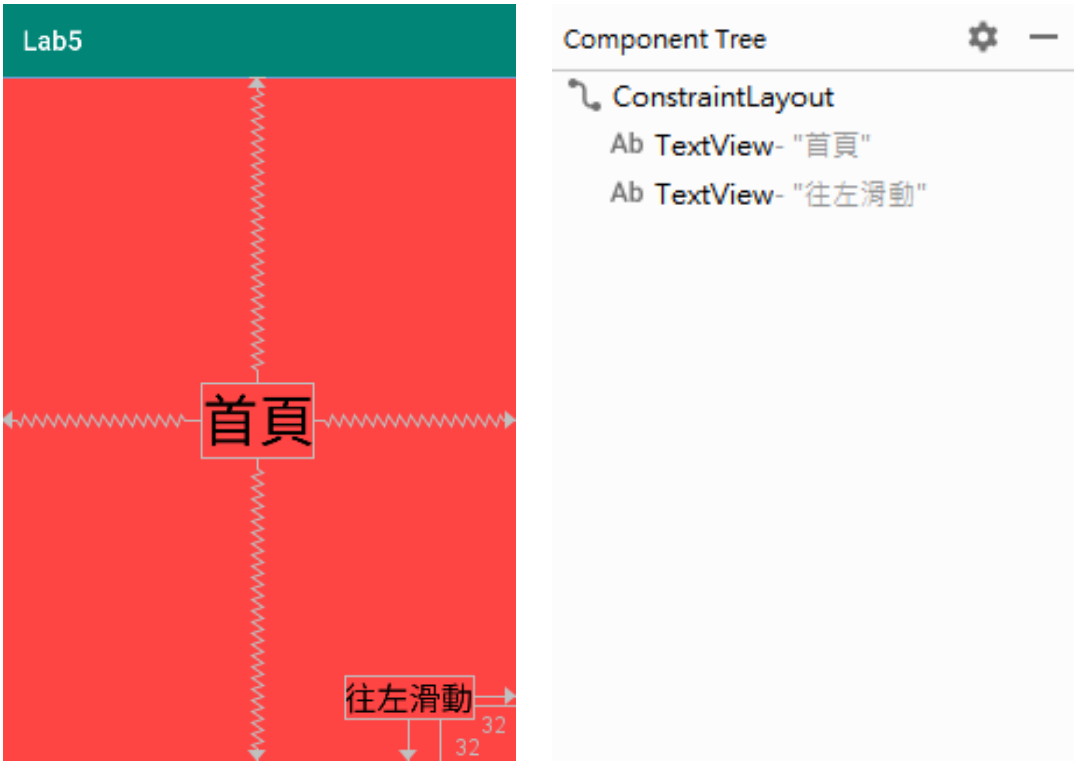

圖 5-10 首頁預覽畫面（左）與布局元件樹（右）

對應的 xml 如下：

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@android:color/holo_red_light">

    <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="首頁"
            android:textSize="42dp"
            android:textColor="@android:color/black"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
```

```
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintStart_toStartOf="parent"/>

    <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="往左滑動"
            android:textSize="24dp"
            android:textColor="@android:color/black"
            android:layout_marginEnd="32dp"
            android:layout_marginBottom="32dp"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintBottom_toBottomOf="parent"/>
</android.support.constraint.ConstraintLayout>
```

Step4 繪製 fragment_second.xml 檔，如圖 5-11 所示。

圖 5-11 第二頁預覽畫面（左）與布局元件樹（右）

對應的 xml 如下：

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@android:color/holo_green_light">

    <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="第二頁"
            android:textSize="42dp"
            android:textColor="@android:color/black"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintStart_toStartOf="parent"/>

    <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="往右滑動"
            android:textSize="24dp"
            android:textColor="@android:color/black"
            android:layout_marginBottom="32dp"
            android:layout_marginStart="32dp"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintStart_toStartOf="parent"/>

    <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="往左滑動"
            android:textSize="24dp"
            android:textColor="@android:color/black"
            android:layout_marginEnd="32dp"
```

```
            android:layout_marginBottom="32dp"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintBottom_toBottomOf="parent"
            android:id="@+id/textView"/>
</android.support.constraint.ConstraintLayout>
```

Step5 繪製 fragment_third.xml 檔，如圖 5-12 所示。

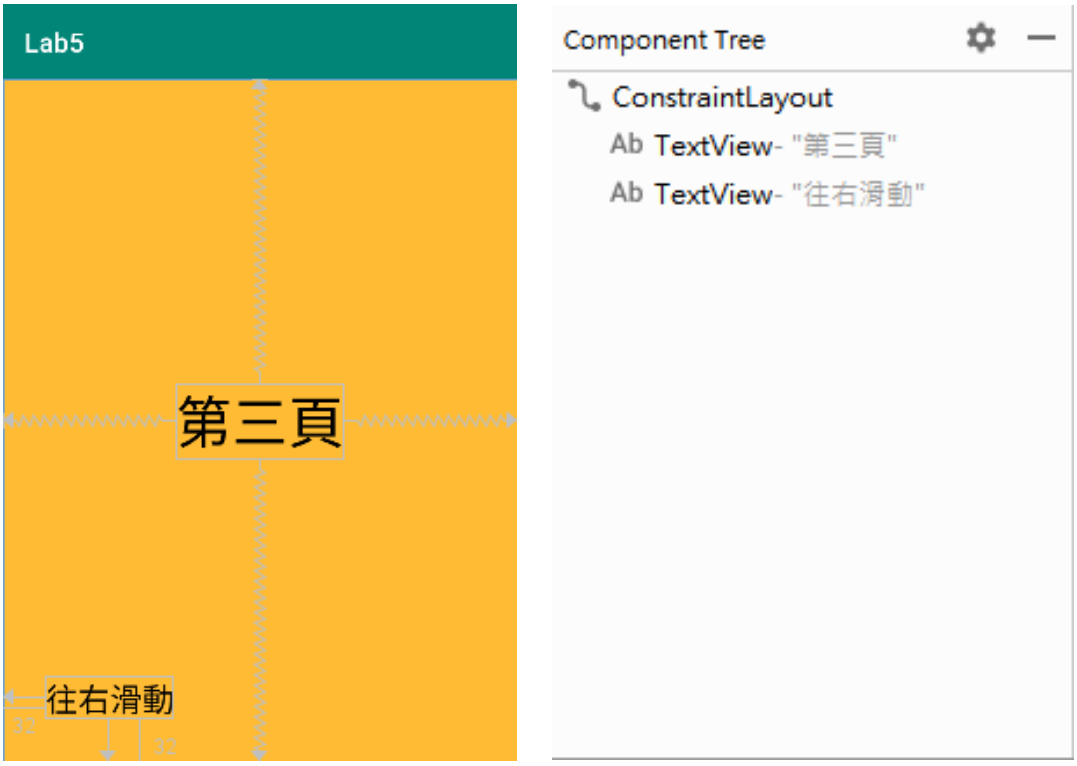

圖 5-12 第三頁預覽畫面（左）與布局元件樹（右）

對應的 xml 如下：

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@android:color/holo_orange_light">

    <TextView
            android:layout_width="wrap_content"
```

```
            android:layout_height="wrap_content"
            android:text="第三頁"
            android:textSize="42dp"
            android:textColor="@android:color/black"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintStart_toStartOf="parent"/>

    <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="往右滑動"
            android:textSize="24dp"
            android:textColor="@android:color/black"
            android:layout_marginBottom="32dp"
            android:layout_marginStart="32dp"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintStart_toStartOf="parent"/>
</android.support.constraint.ConstraintLayout>
```

5.2.2 使用 Log 觀察生命週期

Step1 撰寫 MainActivity 程式，並且加入 Log 以便觀察生命週期變化。

```
import android.support.v7.app.AppCompatActivity
import android.os.Bundle
import android.support.v4.app.FragmentManager
import android.support.v4.app.FragmentPagerAdapter
import android.util.Log
import kotlinx.android.synthetic.main.activity_main.*

class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
```

```
        //使用 Log 追蹤 MainActivity 生命週期
        Log.e("MainActivity","onCreate")
        //建立 FragmentPagerAdapter 物件
        val adapter = ViewPagerAdapter(supportFragmentManager)
        //連接 Adapter，讓畫面(Fragment)與 ViewPager 建立關聯
        viewPager.adapter = adapter
    }

    override fun onRestart() {
        super.onRestart()
        //使用 Log 追蹤 MainActivity 生命週期
        Log.e("MainActivity","onRestart")
    }

    override fun onStart() {
        super.onStart()
        //使用 Log 追蹤 MainActivity 生命週期
        Log.e("MainActivity","onStart")
    }

    override fun onResume() {
        super.onResume()
        //使用 Log 追蹤 MainActivity 生命週期
        Log.e("MainActivity","onResume")
    }

    override fun onPause() {
        super.onPause()
        //使用 Log 追蹤 MainActivity 生命週期
        Log.e("MainActivity","onPause")
    }

    override fun onStop() {
        super.onStop()
        //使用 Log 追蹤 MainActivity 生命週期
        Log.e("MainActivity","onStop")
    }
```

```
    override fun onDestroy() {
        super.onDestroy()
        //使用 Log 追蹤 MainActivity 生命週期
        Log.e("MainActivity","onDestroy")
    }
}

class ViewPagerAdapter(fm: FragmentManager) :
FragmentPagerAdapter(fm) {
    //回傳對應位置的 Fragment，決定頁面的呈現順序
    override fun getItem(position: Int) = when(position){
        0 ->FirstFragment() //第一頁要呈現的 Fragment
        1 ->SecondFragment() //第二頁要呈現的 Fragment
        else ->ThirdFragment() //第三頁要呈現的 Fragment
    }
    //回傳 Fragment 頁數
    override fun getCount() = 3
}
```

說明

在使用 Log 時要注意，部分的機型使用 Log.d()時，會發生無法顯示在 LogCat 上的情況，因此建議您使用 Log.e()來進行程式偵錯與追蹤。

Step2 撰寫 FirstFragment 程式，並且加入 Log 以便觀察生命週期變化。

```
package bluenet.com.lab5

import android.os.Bundle
import android.support.v4.app.Fragment
import android.util.Log
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup

class FirstFragment : Fragment() {
```

```
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        //使用 Log 追蹤 FirstFragment 生命週期
        Log.e("FirstFragment","onCreate")
    }
    //在 onCreateView 中定義 FirstFragment 的畫面為 fragment_first
    override fun onCreateView(inflater: LayoutInflater,
container: ViewGroup?, savedInstanceState: Bundle?): View? {
        //使用 Log 追蹤 FirstFragment 生命週期
        Log.e("FirstFragment","onCreate")
        return inflater.inflate(R.layout.fragment_first,
                                              container, false)
    }

    override fun onActivityCreated(savedInstanceState: Bundle?)
    {
        super.onActivityCreated(savedInstanceState)
        //使用 Log 追蹤 FirstFragment 生命週期
        Log.e("FirstFragment","onActivityCreated")
    }

    override fun onStart() {
        super.onStart()
        //使用 Log 追蹤 FirstFragment 生命週期
        Log.e("FirstFragment","onStart")
    }

    override fun onResume() {
        super.onResume()
        //使用 Log 追蹤 FirstFragment 生命週期
        Log.e("FirstFragment","onResume")
    }

    override fun onPause() {
        super.onPause()
        //使用 Log 追蹤 FirstFragment 生命週期
        Log.e("FirstFragment","onPause")
```

```
    }

    override fun onStop() {
        super.onStop()
        //使用 Log 追蹤 FirstFragment 生命週期
        Log.e("FirstFragment","onStop")
    }

    override fun onDestroyView() {
        super.onDestroyView()
        //使用 Log 追蹤 FirstFragment 生命週期
        Log.e("FirstFragment","onDestroyView")
    }

    override fun onDestroy() {
        super.onDestroy()
        //使用 Log 追蹤 FirstFragment 生命週期
        Log.e("FirstFragment","onDestroy")
    }

    override fun onDetach() {
        super.onDetach()
        //使用 Log 追蹤 FirstFragment 生命週期
        Log.e("FirstFragment","onDetach")
    }
}
```

Step3 撰寫 SecondFragment 程式，參考 FirstFragment 加入 Log。

```
package bluenet.com.lab5

import android.os.Bundle
import android.support.v4.app.Fragment
import android.util.Log
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
```

```
class SecondFragment : Fragment() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        //使用 Log 追蹤 SecondFragment 生命週期
        Log.e("SecondFragment","onCreate")
    }
    //在 onCreateView 中定義 SecondFragment 的畫面為 fragment_second
    override fun onCreateView(inflater: LayoutInflater,
container: ViewGroup?, savedInstanceState: Bundle?): View? {
        //使用 Log 追蹤 SecondFragment 生命週期
        Log.e("SecondFragment","onCreate")
        return inflater.inflate(R.layout.fragment_second,
                                                container, false)
    }

    override fun onActivityCreated(savedInstanceState: Bundle?)
    {
        super.onActivityCreated(savedInstanceState)
        //使用 Log 追蹤 SecondFragment 生命週期
        Log.e("SecondFragment","onActivityCreated")
    }

    override fun onStart() {
        super.onStart()
        //使用 Log 追蹤 SecondFragment 生命週期
        Log.e("SecondFragment","onStart")
    }

    override fun onResume() {
        super.onResume()
        //使用 Log 追蹤 SecondFragment 生命週期
        Log.e("SecondFragment","onResume")
    }

    override fun onPause() {
        super.onPause()
        //使用 Log 追蹤 SecondFragment 生命週期
        Log.e("SecondFragment","onPause")
```

```
    }

    override fun onStop() {
        super.onStop()
        //使用 Log 追蹤 SecondFragment 生命週期
        Log.e("SecondFragment","onStop")
    }

    override fun onDestroyView() {
        super.onDestroyView()
        //使用 Log 追蹤 SecondFragment 生命週期
        Log.e("SecondFragment","onDestroyView")
    }

    override fun onDestroy() {
        super.onDestroy()
        //使用 Log 追蹤 SecondFragment 生命週期
        Log.e("SecondFragment","onDestroy")
    }

    override fun onDetach() {
        super.onDetach()
        //使用 Log 追蹤 SecondFragment 生命週期
        Log.e("SecondFragment","onDetach")
    }
}
```

Step4 撰寫 ThirdFragment 程式，參考 FirstFragment 加入 Log。

```
package bluenet.com.lab5

import android.os.Bundle
import android.support.v4.app.Fragment
import android.util.Log
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
```

```
class ThirdFragment : Fragment() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        //使用 Log 追蹤 ThirdFragment 生命週期
        Log.e("ThirdFragment","onCreate")
    }
    //在 onCreateView 中定義 ThirdFragment 的畫面為 fragment_third
    override fun onCreateView(inflater: LayoutInflater,
container: ViewGroup?, savedInstanceState: Bundle?): View? {
        //使用 Log 追蹤 ThirdFragment 生命週期
        Log.e("ThirdFragment","onCreate")
        return inflater.inflate(R.layout.fragment_third,
                                                container, false)
    }

    override fun onActivityCreated(savedInstanceState: Bundle?)
    {
        super.onActivityCreated(savedInstanceState)
        //使用 Log 追蹤 ThirdFragment 生命週期
        Log.e("ThirdFragment","onActivityCreated")
    }

    override fun onStart() {
        super.onStart()
        //使用 Log 追蹤 ThirdFragment 生命週期
        Log.e("ThirdFragment","onStart")
    }

    override fun onResume() {
        super.onResume()
        //使用 Log 追蹤 ThirdFragment 生命週期
        Log.e("ThirdFragment","onResume")
    }

    override fun onPause() {
        super.onPause()
        //使用 Log 追蹤 ThirdFragment 生命週期
        Log.e("ThirdFragment","onPause")
```

```
    }

    override fun onStop() {
        super.onStop()
        //使用 Log 追蹤 ThirdFragment 生命週期
        Log.e("ThirdFragment","onStop")
    }

    override fun onDestroyView() {
        super.onDestroyView()
        //使用 Log 追蹤 ThirdFragment 生命週期
        Log.e("ThirdFragment","onDestroyView")
    }

    override fun onDestroy() {
        super.onDestroy()
        //使用 Log 追蹤 ThirdFragment 生命週期
        Log.e("ThirdFragment","onDestroy")
    }

    override fun onDetach() {
        super.onDetach()
        //使用 Log 追蹤 ThirdFragment 生命週期
        Log.e("ThirdFragment","onDetach")
    }
}
```

Step5 開啟位於 Android Studio 左下方的 Debug 工具「**Logcat**」，如圖 5-13 所示。我們需要設定追蹤的裝置與應用程式，並且可以過濾標籤類別以及 Log 中的字串。

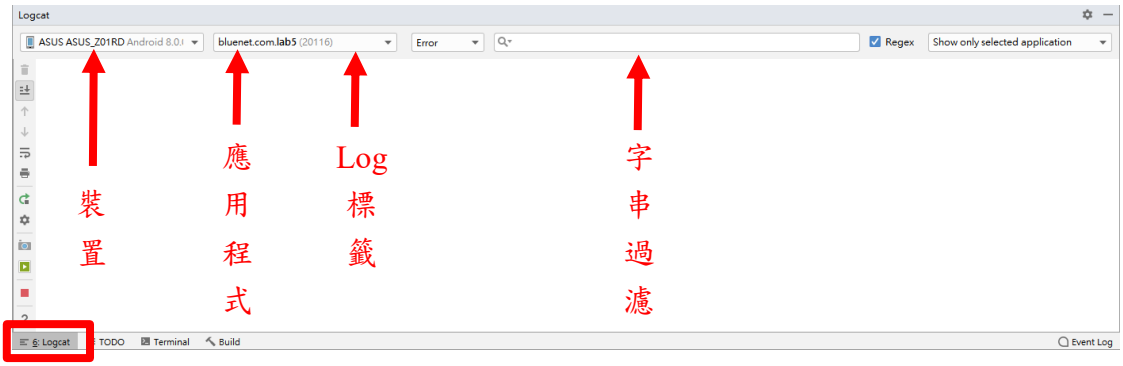

圖 5-13 Debug 工具 Logcat 位於編譯器下方

Step6 啟動 Lab5 如圖 5-14 所示，觀察 MainActivity、FirstFragment、SecondFragment 的創建過程。

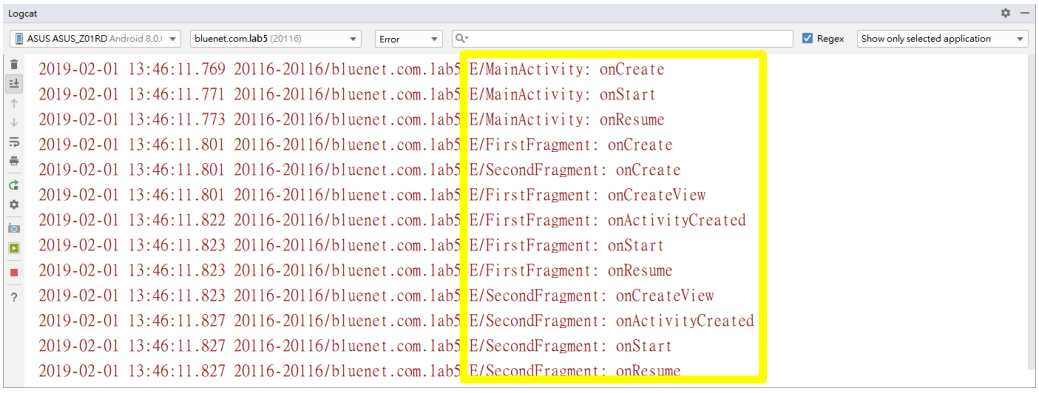

圖 5-14 啟動 Lab5 觸發的生命週期

Step7 圖 5-15 滑動頁面至第二頁，ThirdFragment 在此時才開始創建實體。

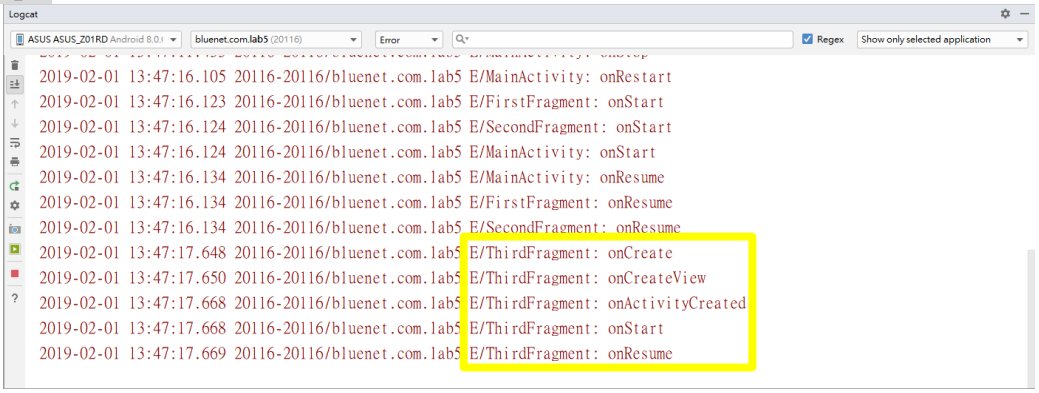

圖 5-15 滑動至第二頁後的生命週期變化

Step8 圖 5-16 滑動至第三頁，FirstFragment 的畫面被回收，資源進入背景待命。

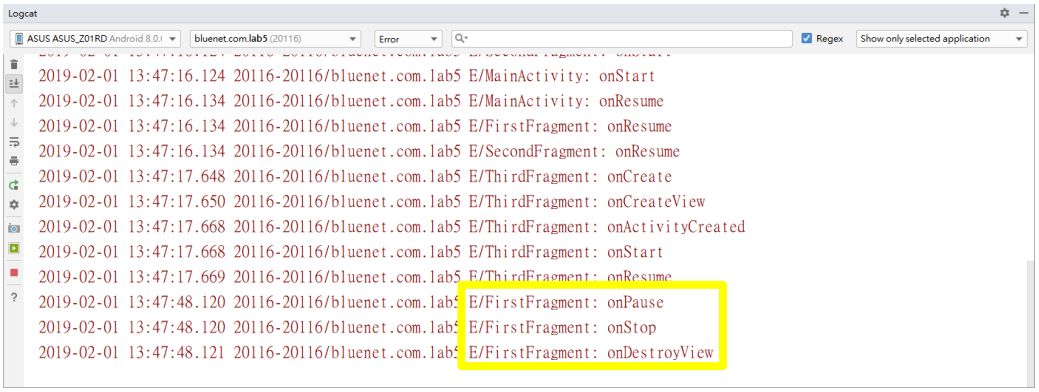

圖 5-16 滑動至第三頁後的生命週期變化

說明

試試看退出 Lab5 的應用程式，Activity 與 Fragment 的生命週期會發生什麼變化?創建另一個 Activity 並使用 startActivity()切換，又會生什麼變化?