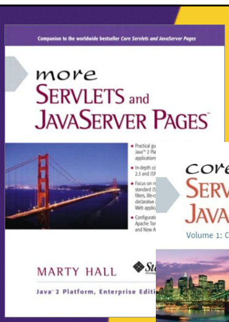
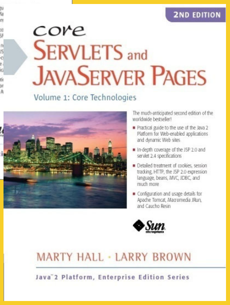
Intents, Intent Filters, and Invoking Activities: Part III: Using Tabs

Originals of Slides and Source Code for Examples:
http://www.coreservlets.com/android-tutorial/





Topics in This Section

- **Part I**
  - Invoking Activities by class name
  - Defining dimensions in res/values
  - Sending data via the “extras” Bundle
- **Part II**
  - Invoking Activities with a URI
  - Sending data via parameters in the URI
- **Part III**
  - Invoking Activities with tabbed windows
  - Defining two-image icons in res/drawable

Overview

Summary of Options

- **Invoke Activity by class name (Part I)**
  - Exactly one Activity can match
  - New Activity must be in same project as original
  - Can send data via an "extras" Bundle
- **Invoke Activity by URI (Part II)**
  - More than one Activity could match
  - New Activity need not be in the same project as original
  - Can send data via URI parameters or "extras" Bundle
- **Switch Activities via tabs (Part III)**
  - Can use class name or URI to specify Activity
  - New Activity must be in same project as original
  - Can send data via URI parameters or "extras" Bundle

Invoking Activities with Tabbed Windows

Summary

- **Idea**
  - Make tabbed windows. Each tab invokes a different Activity, or an Activity with different data.
    - Can use either specific-class approach or URI approach
    - Can send data either with an extras Bundle or in URI
    - Tab window Activity and new Activities must be in same project
      - Due to security reasons
- **Syntax**
  - Java
    - Extends TabActivity. Uses TabHost and TabSpec
      - Details on next slide
  - XML (AndroidManifest.xml)
    - Same as shown earlier

Using TabActivity: Outline

```
public class SomeActivity extends TabActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Resources resources = getResources();
        TabHost host = getTabHost();
        Intent intent1= ...;
        Drawable tabIcon =
            resources.getDrawable(R.drawable.icon_name);
        TabSpec tab1Spec =
            host.newTabSpec("Tab One")
                .setIndicator("Some Text", tabIcon)
                .setContent(intent1);
        host.addTab(tab1Spec);
        // Repeat for other tabs
    }
}
```

Note that the setter methods for TabSpec return the TabSpec so that you can do chained assignments.
Note also that there is *no* layout file when using this approach.

Defining Tab Icons

- **Idea**
  - Although it is legal to call setIndicator(someString), the resultant tab looks bad because of blank space at top. So, more common to do setIndicator(someString, someIcon).
    - You can also do setIndicator(someView) for fancy tabs
- **Icon option 1**
  - Use a single image for the icon
    - Same image used when the tab is or is not selected
- **Icon option 2**
  - Use 2 similar but differently colored images for the icon
    - One for when selected, one for when not

Option 1: A Single Image

- **Pros**
  - Simpler
  - Text color and background color of the tab already change on selection, so not confusing if icon stays same.
- **Cons**
  - Doesn't look quite as good as with two images
- **Approach**
  - Put image file in res/drawable/some_icon.png
  - Refer to image with
    - Drawable tabIcon =
      resources.getDrawable(R.drawable.some_icon);
  - Put icon in tab label with
    - tabSpec.setIndicator("Some Text", tabIcon);

Option 2: Two Images (Normal and Selected)

- **Pros**
  - Looks better
- **Cons**
  - More work
- **Approach**
  - Put image files in
    - res/drawable/some_icon_normal.png and
    - res/drawable/some_icon_selected.png
  - Make XML file (next page)
    - res/drawable/some_icon.xml
  - Refer to XML file with
    - Drawable tabIcon = resources.getDrawable(R.drawable.some_icon);
  - Put icon in tab label with
    - tabSpec.setIndicator("Some Text", tabIcon);

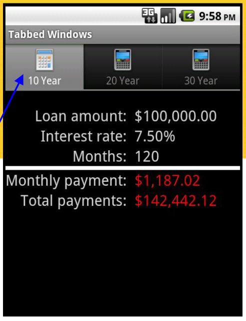
XML Code for Dual-Image Icon

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://...">
    <!-- When tab selected, use some_icon_selected.png -->
    <item android:drawable="@drawable/some_icon_selected"
          android:state_selected="true" />
    <!-- When tab not selected, use some_icon_normal.png -->
    <item android:drawable="@drawable/some_icon_normal" />
</selector>
```

The file names of the two images are arbitrary. They need not end in _selected and _normal, although this can be a useful convention so that you know what the images are for.

Example: Invoking Loan Calculator (Each Tab Sends Different Data)

Example: Overview



Loan amount: $100,000.00
Interest rate: 7.50%
Months: 120
Monthly payment: $1,187.02
Total payments: $142,442.12

- **Initial Activity**
  - Has tabs that, when pressed, invoke the loan calculator with different data
    - Activity specified either with class name or URI
      - But either way, initial Activity must be in same project as new one
    - Data sent via either in extras Bundle or in URI
- **Approach**
  - Intents and data created in same way as before
  - Intent associated with tab via tabHost.setContent
  - Put entry for LoanCalculatorActivity in manifest
    - Same as shown previously

XML: Icon File (res/drawable/calculator.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <!-- Calculator images from http://www.fatcow.com/free-icons -->
    <!-- When selected, use white -->
    <item android:drawable="@drawable/calculator_white"
          android:state_selected="true" />
    <!-- When not selected, use black-->
    <item android:drawable="@drawable/calculator_black" />
</selector>
```

XML: Manifest File Action Declaration

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.coreservlets.intentfilter1"
      android:versionCode="1"
      android:versionName="1.0">
    <uses-sdk android:minSdkVersion="8" />

    <application android:icon="@drawable/icon"
                 android:label="@string/app_name">
        ... <!-- Declaration for IntentFilter1Activity shown earlier -->
        <activity android:name=".LoanCalculatorActivity"
                  android:label="@string/loan_calculator_app_name">
            <intent-filter>
                <action android:name="android.intent.action.VIEW" />
                <category android:name="android.intent.category.DEFAULT" />
                <data android:scheme="loan" android:host="coreservlets.com" />
            </intent-filter>
        </activity>
        ...
     </application>
</manifest>
```

Unchanged from previous examples.

Java (TabbedActivity: Tab 1)

```
public class TabbedActivity extends TabActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Resources resources = getResources();
        TabHost host = getTabHost();
        Intent intent1 =
                new Intent(this, LoanCalculatorActivity.class);
        Bundle loanBundle1 =
               LoanBundler.makeLoanInfoBundle(100000, 7.5, 120);
        intent1.putExtras(loanBundle1);
        Drawable tabIcon =
                resources.getDrawable(R.drawable.calculator);
        TabSpec tab1Spec = host.newTabSpec("Tab One")
                               .setIndicator("10 Year", tabIcon)
                               .setContent(intent1);
        host.addTab(tab1Spec);
```

This first tab uses an Intent that specifies the Activity by class name.
It sends data via an extras Bundle. Reminder: there is *no* layout file when using TabActivity.

Java (TabbedActivity: Tab 2)

```
        Uri uriTwentyYear =
                Uri.parse("loan://coreservlets.com/calc");
        Intent intent2 =
                new Intent(Intent.ACTION_VIEW, uriTwentyYear);
        Bundle loanBundle2 =
               LoanBundler.makeLoanInfoBundle(100000, 7.5, 240);
        intent2.putExtras(loanBundle2);
        tabIcon = resources.getDrawable(R.drawable.calculator);
        TabSpec tab2Spec = host.newTabSpec("Tab Two")
                               .setIndicator("20 Year", tabIcon)
                               .setContent(intent2);
        host.addTab(tab2Spec);
```

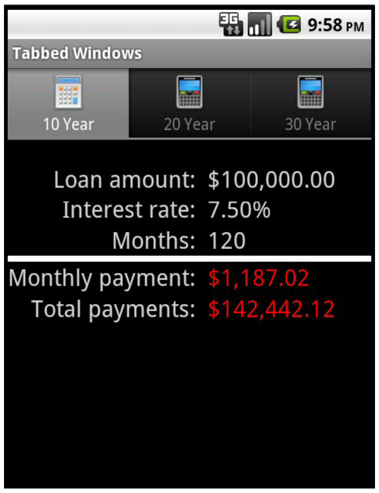
This second tab uses an Intent that specifies the Activity with a URI.
It sends data via an extras Bundle.

Java (TabbedActivity: Tab 3)

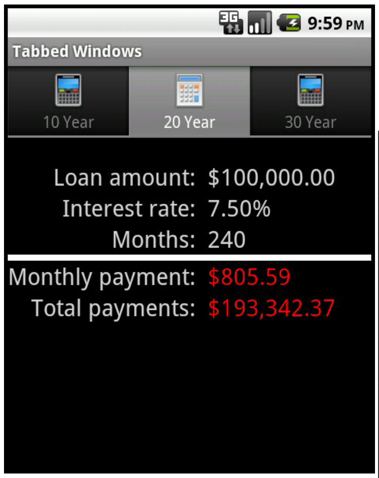
```
String baseAddress = "loan://coreservlets.com/calc";
String address =
        String.format("%s?%s&%s&%s",
                      baseAddress,
                      "loanAmount=100000",
                      "annualInterestRateInPercent=7.5",
                      "loanPeriodInMonths=360");
Uri uriThirtyYear = Uri.parse(address);
Intent intent3 =
        new Intent(Intent.ACTION_VIEW, uriThirtyYear);
tabIcon = resources.getDrawable(R.drawable.calculator);
TabSpec tab3Spec = host.newTabSpec("Tab Three")
                       .setIndicator("30 Year", tabIcon)
                       .setContent(intent3);
host.addTab(tab3Spec);
```

This second tab uses an Intent that specifies the Activity with a URI.
It sends data via parameters embedded in the URI.

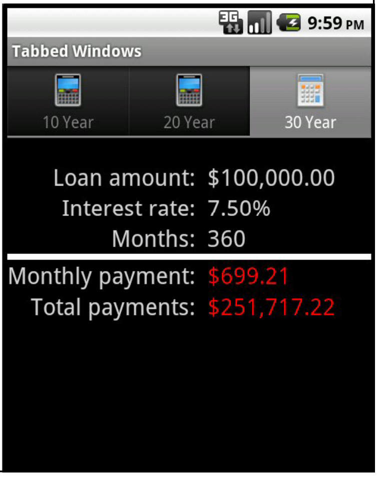
Example: Results



```
    Loan amount:  $100,000.00
  Interest rate:  7.50%
         Months:  120
Monthly payment:  $1,187.02
 Total payments:  $142,442.12
```



```
    Loan amount:  $100,000.00
  Interest rate:  7.50%
         Months:  240
Monthly payment:  $805.59
 Total payments:  $193,342.37
```



```
    Loan amount:  $100,000.00
  Interest rate:  7.50%
         Months:  360
Monthly payment:  $699.21
 Total payments:  $251,717.22
```

Wrap-Up

Summary

- **Java (extends TabActivity)**

```
TabHost host = getTabHost();
Intent intent1= ...;   // Refers to Activity in same project
Drawable tabIcon = resources.getDrawable(R.drawable.some_icon);
TabSpec tab1Spec = host.newTabSpec("Tab One")
                       .setIndicator("Some Text", tabIcon)
                       .setContent(intent1);
host.addTab(tab1Spec);
// Repeat for other tabs
```

- **Icon (res/drawable/some_icon.xml)**

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://...">
  <item android:drawable="@drawable/some_icon_selected"
        android:state_selected="true" />
  <item android:drawable="@drawable/some_icon_normal" />
</selector>
```

Questions?