Intents, Intent Filters, and Invoking Activities: Part II: Using URI

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 a URI

Summary

- **Idea**
  - Supply a URI that indirectly refers to new Activity. The new Activity registers as target for URIs of a certain form.
    - *The originating Activity and the new Activity need not be in the same project*
    - More than one Activity could match the URI.
      - If so, Android will ask you which one to use.
- **Syntax**
  - Java (original Activity)

```
Uri uri = Uri.parse("foo://bar.example.com/baz");
Intent intent = new Intent(Intent.ACTION_VIEW, uri);
startActivity(activityIntent);
```

  - XML (AndroidManifest.xml)

```
<intent-filter>
    <action android:name="android.intent.action.VIEW" />
    <category android:name="android.intent.category.DEFAULT"/>
    <data android:scheme="foo" android:host="bar.example.com" />
</intent-filter>
```

Registering to Handle URIs

- **Matching the URI itself**
  - Register for a scheme and a host
    - Example URI
      - loan://coreservlets.com/calc
    - intent-filter entry
      - <data android:scheme="loan" android:host="coreservlets.com" />
      - Note that the “calc” part is arbitrary – just to make URL look better.
- **Matching the data type**
  - Register for a MIME type
    - Example URIs
      - content:// (referring to that MIME type)
      - file:// (referring to that MIME type)
      - anything:// (the Intent can call setType to specify MIME type)
    - intent-filter entry
      - <data android:mimeType="some/type" />
      - <data android:mimeType="something/*" />

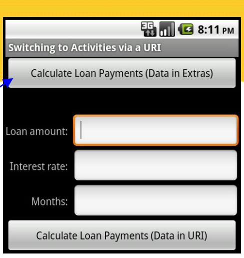
Predefined Action/URI Combinations

Action	URI	Meaning
Intent.ACTION_CALL	tel:*phone_number*	Opens phone application and calls *phone_number*.
Intent.ACTION_DIAL	tel:*phone_number*	Opens phone application and dials (but doesn't call) *phone_number*.
Intent.ACTION_DIAL	voicemail:	Opens phone application and dials (but doesn't call) the voice mail number.
Intent.ACTION_VIEW	geo:*lat*,*long*	Opens the maps application centered on (*lat*, *long*).
Intent.ACTION_VIEW	geo:0,0?q=*address*	Opens the maps application centered on the specified address.
Intent.ACTION_VIEW	http://*url* https://*url*	Opens the browser application to the specified address.
Intent. ACTION_WEB_SEARCH	*plain_text*	Opens the browser application and uses Google search for given string.

Table adapted from Section 4.1.5 of *Android in Action* by Ableson et al.

Example: Invoking Loan Calculator (Data in Extras Bundle)

Example: Overview


8:11 PM
Switching to Activities via a URI
Calculate Loan Payments (Data in Extras)
Loan amount:
Interest rate:
Months:
Calculate Loan Payments (Data in URI)

- **Initial Activity**
  - Has Button that, when pressed, invokes the loan calculator activity
    - Initial Activity uses URI to indirectly invoke loan calculator
    - Initial Activity is in different project than loan calculator
    - Data is sent via extras Bundle as in previous example
- **Approach**
  - Create Intent with Intent.ACTION_VIEW and URI of "loan://coreservlets.com/calc“
    - The “calc” at the end is arbitrary – just for aesthetics
  - Create and attach Bundle as in previous example
  - Call startActivity
  - Put data entry for LoanCalculatorActivity in manifest
    - <data android:scheme="loan" android:host="coreservlets.com" />

XML: Layout File (res/layout/main.xml – 2nd Proj.)

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
             android:layout_width="match_parent"
             android:layout_height="match_parent"
             android:stretchColumns="1">
    <TableRow>
        <Button
            android:text="Calculate Loan Payments (Data in Extras)"
            android:layout_span="2"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:onClick="showLoanPayments1"/>
    </TableRow>
    ...
</TableLayout>
```

Entries for input form and second button shown later.

XML: Manifest File Action Declaration (Loan Proj.)

```
<?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>
```

Java (IntentFilter2Activity.java)

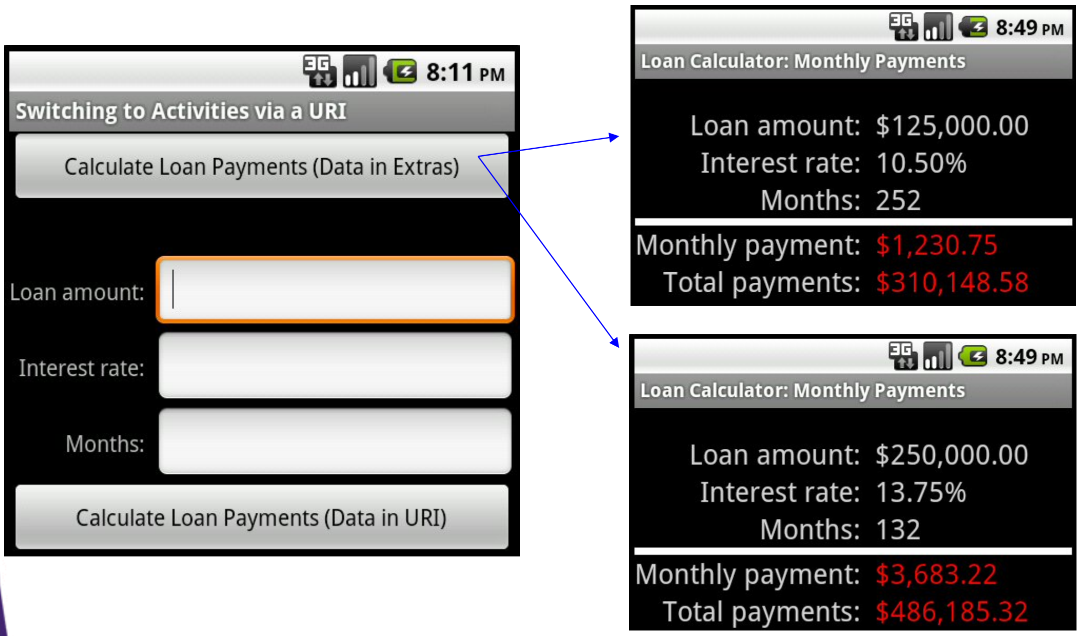
```
public class IntentFilter2Activity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }

    public void showLoanPayments1(View clickedButton) {
        Uri uri = Uri.parse("loan://coreservlets.com/calc");
        Intent intent = new Intent(Intent.ACTION_VIEW, uri);
        intent.putExtras
                (LoanBundler.makeRandomizedLoanInfoBundle());
        startActivity(intent);
    }
    ...
}
```

Code for second button (that embeds data in the URI) shown later.

Java Code Shown Earlier

- **LoanBundler**
  - Makes a Bundle that stores the loan amount, interest rate, and loan period
- **LoanCalculatorActivity**
  - Calls getIntent().getExtras() and reads the data out of the resultant Bundle. Uses that for the initial values for the loan amount, interest rate, and loan period
  - Passes the values to PaymentInfo, which in turn uses LoanUtils to calculate monthly payment and total payments
  - Puts all five values (loan amount, interest rate, loan period, monthly payment, total payments) into TextViews

Example: Results


8:11 PM
Switching to Activities via a URI
Calculate Loan Payments (Data in Extras)
Loan amount:
Interest rate:
Months:
Calculate Loan Payments (Data in URI)
8:49 PM
Loan Calculator: Monthly Payments
8:49 PM
Loan Calculator: Monthly Payments

Loan amount: $125,000.00
Interest rate: 10.50%
Months: 252
Monthly payment: $1,230.75
Total payments: $310,148.58

Loan amount: $250,000.00
Interest rate: 13.75%
Months: 132
Monthly payment: $3,683.22
Total payments: $486,185.32

Sending Data via Parameters in the URI

Summary

- **Idea**
  - Embed query parameters in the URI. These parameters will represent data to be used by the new Activity.
- **Syntax**
  - Java (original Activity)

```
String address =
    "loan://coreservlets.com/calc?loanAmount=xxx&...";
Uri uri = Uri.parse(address);
Intent intent = new Intent(Intent.ACTION_VIEW, uri);
startActivity(activityIntent);
```

  - Java (new Activity)

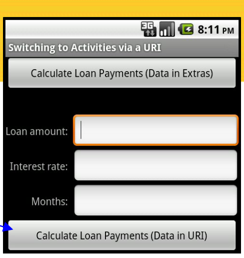
```
Uri uri = getIntent().getData();
String loanAmountString = uri.getQueryParameter("loanAmount");
// Convert String to double
...
```

Sending Data: Extras vs. URI Parameters

- **Extras Bundle**
  - Pros
    - Can send data of different types.
    - No parsing required in Activity that receives the data.
  - Cons
    - More complex for originating Activity
      - Requires parsing in originating Activity if values come from EditText
- **URI parameters**
  - Pros
    - Simpler for originating Activity, especially if EditText used
    - More consistent with URI usage
  - Cons
    - Can send Strings only
    - Requires parsing in receiving Activity

Example: Invoking Loan Calculator (Data in URI Parameters)

Example: Overview


8:11 PM
Switching to Activities via a URI
Calculate Loan Payments (Data in Extras)
Loan amount:
Interest rate:
Months:
Calculate Loan Payments (Data in URI)

- **Initial Activity**
  - Has Button that, when pressed, invokes the loan calculator activity
    - Data is extracted from textfields (EditTexts) and embedded in the URI that is used to invoke loan calculator
- **Approach**
  - Create Intent with Intent.ACTION_VIEW and URI of "loan://coreservlets.com/calc?*data*"
    - Data is
      "loanAmount=…&annualInterestRateInPercent=…&…"
  - Call startActivity
  - Put data entry for LoanCalculatorActivity in manifest
    - <data android:scheme="loan" android:host="coreservlets.com" />

XML: Layout File (res/layout/main.xml – 2nd Proj.)

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
             android:layout_width="match_parent"
             android:layout_height="match_parent"
             android:stretchColumns="1">
    ...
    <TableRow android:layout_marginTop="30dp">
        <TextView android:layout_width="wrap_content"
                  android:layout_height="wrap_content"
                  android:text="@string/loan_amount_prompt"
                  android:gravity="right"/>
        <EditText android:id="@+id/loan_amount"
                  android:inputType="numberDecimal"
                  android:layout_height="wrap_content">
            <requestFocus></requestFocus>
        </EditText>
    </TableRow>
    ...
</TableLayout>
```

Entry for first button shown earlier. Entries for other textfields (EditTexts) similar to the one shown. Entry for button at the bottom just has android:onClick="showLoanPayments2".

XML: Strings File (res/values/strings.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Intent Filters and Activity Switching</string>
    <string name="loan_calculator_app_name">
        Loan Calculator: Monthly Payments
    </string>
    <string name="tabs_app_name">Tabbed Windows</string>
    <string name="loan_amount_prompt">Loan amount:&#160;&#160;</string>
    <string name="interest_rate_prompt">Interest rate:&#160;&#160;</string>
    <string name="loan_period_prompt">Months:&#160;&#160;</string>
    <string name="monthly_payment_prompt">Monthly payment:&#160;&#160;</string>
    <string name="total_payments_prompt">Total payments:&#160;&#160;</string>
</resources>
```

The same prompts are also used in the output display.

Note that &#160; represents a non-breaking space. Regular spaces are not preserved at the beginning and end of strings in Android resource files. Note also that &nbsp; is *not* legal here, since that is a character entity specific to HTML, not general in XML.

XML: Manifest File Action Declaration (Loan Proj.)

```
<?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 example.

Java (IntentFilter2Activity.java)

```
public class IntentFilter2Activity extends Activity {
    ...

    public void showLoanPayments2(View clickedButton) {
        String address = makeLoanAddressFromEditTextInputs();
        Uri uri = Uri.parse(address);
        Intent intent = new Intent(Intent.ACTION_VIEW, uri);
        startActivity(intent);
    }
```

Code for onCreate and first button shown earlier.

Java (IntentFilter2Activity, Continued)

```
private String makeLoanAddressFromEditTextInputs() {
    EditText loanAmountInput = (EditText)findViewById(R.id.loan_amount);
    Editable loanAmount = loanAmountInput.getText();
    String loanAmountParam =
            String.format("loanAmount=%s", loanAmount);
    EditText interestRateInput = (EditText)findViewById(R.id.interest_rate);
    Editable interestRate = interestRateInput.getText();
    String interestRateParam =
            String.format("annualInterestRateInPercent=%s", interestRate);
    EditText loanPeriodInput = (EditText)findViewById(R.id.loan_period);
    Editable loanPeriod = loanPeriodInput.getText();
    String loanPeriodParam =
            String.format("loanPeriodInMonths=%s", loanPeriod);
    String baseAddress = "loan://coreservlets.com/calc";
    String address =
            String.format("%s?%s&%s&%s", baseAddress, loanAmountParam,
                          interestRateParam, loanPeriodParam);
    return(address);
}
```

Java (LoanCalculatorActivity.java)

```
public class LoanCalculatorActivity extends Activity {
    private double mLoanAmount=100000,
                   mAnnualInterestRateInPercent=5.0;
    private long mLoanPeriodInMonths=360; // 30 years
    private String mCurrencySymbol = "$";

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.loan_payments);
        setInputsFromExtras();
        setInputsFromUri();
        calculateAndSetOutputValues();
    }
```

Java (LoanCalculatorActivity, Continued)

```
private void setInputsFromUri() {
    Uri uri = getIntent().getData();
    if (uri != null) {
        double loanAmount = getDoubleParam(uri, "loanAmount");
        double annualInterestRateInPercent =
                getDoubleParam(uri, "annualInterestRateInPercent");
        long loanPeriodInMonths =
                getLongParam(uri, "loanPeriodInMonths");
        String currencySymbol =
                uri.getQueryParameter("currencySymbol");
        setInputs(loanAmount, annualInterestRateInPercent,
                  loanPeriodInMonths, currencySymbol);
    }
}
```

getQueryParameter is the builtin method of Uri. getDoubleParam and getLongParam (next slides) are methods of LoanCalculatorActivity that call getQueryParameter and then parse the resultant String.

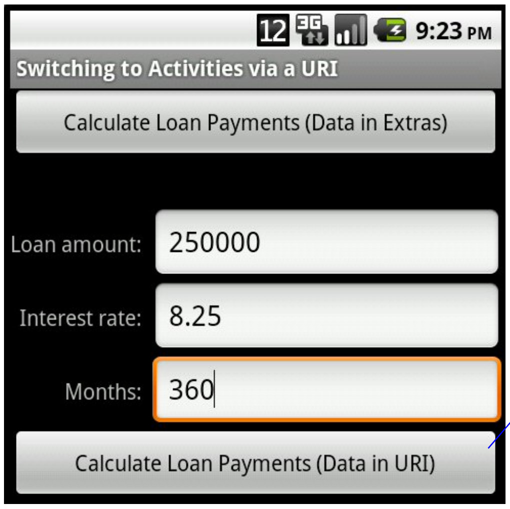
Java (LoanCalculatorActivity, Continued)

```
private void setInputsFromUri() {
    Uri uri = getIntent().getData();
    if (uri != null) {
        double loanAmount = getDoubleParam(uri, "loanAmount");
        double annualInterestRateInPercent =
                getDoubleParam(uri, "annualInterestRateInPercent");
        long loanPeriodInMonths =
                getLongParam(uri, "loanPeriodInMonths");
        String currencySymbol =
                uri.getQueryParameter("currencySymbol");
        setInputs(loanAmount, annualInterestRateInPercent,
                  loanPeriodInMonths, currencySymbol);
    }
}
```

Java (LoanCalculatorActivity, Continued)

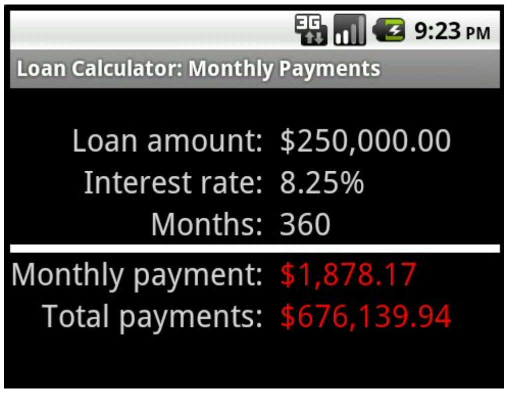
```
private double getDoubleParam(Uri uri, String queryParamName) {
    String rawValue = uri.getQueryParameter(queryParamName);
    double value = 0.0;
    try {
        value = Double.parseDouble(rawValue);
    } catch(Exception e) { } // NumberFormatEx or NullPointerEx
    return(value);
}

private long getLongParam(Uri uri, String queryParamName) {
    String rawValue = uri.getQueryParameter(queryParamName);
    long value = 0;
    try {
        value = Long.parseLong(rawValue);
    } catch(Exception e) { } // NFE or NPE
    return(value);
}
```

Example: Results


Switching to Activities via a URI

12
9:23 PM
Calculate Loan Payments (Data in Extras)
Calculate Loan Payments (Data in URI)

Loan amount:	250000
Interest rate:	8.25
Months:	360


Loan Calculator: Monthly Payments

Loan amount:	$250,000.00
Interest rate:	8.25%
Months:	360
Monthly payment:	$1,878.17
Total payments:	$676,139.94

Wrap-Up

Summary

- **Java (original Activity)**

```
String address =
    "loan://coreservlets.com/calc?loanAmount=xxx&...";
Uri uri = Uri.parse(address);
Intent intent = new Intent(Intent.ACTION_VIEW, uri);
startActivity(activityIntent);
```

- **Java (new Activity – can be different project)**

```
Uri uri = getIntent().getData();
String loanAmountString = uri.getQueryParameter("loanAmount");
// Convert String to double, handle bad data
...
```

- **XML (AndroidManifest.xml)**

```
<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>
```

Questions?