App Development for Smart Devices

CS 495/595 - Fall 2011

Lec #18: Advanced Topics

Tamer Nadeem
Dept. of Computer Science
Objective

- Web Browsing
- Android Animation
- Android Backup

- Presentation
  - Developing Secure Mobile Applications for Android
    - Presenter: Zachary Howland
Web Browsing
Objectives

• Show how to launch the built-in Browser application in three ways.
  ➤ Launch the browser to a specific URL.
  ➤ Create text with links.
  ➤ Launch a Google web search and specify the search criteria.

• You will achieve these goals by creating and configuring the appropriate Intents within your application’s Activity class.
Launch the Browser

- Working with URIs:
  ➤ Use Uri objects to identify the unique location of a piece of data.
  ➤ Create a Uri object from a web URL using the `parse()`

  ```java
  Uri uriUrl = Uri.parse("http://www.google.com/");
  ```

- Creating the Intent:
  ➤ Use `android.content.Intent.ACTION_VIEW` to view HTML
  ➤ Specify the URI in the intent

  ```java
  Intent launchBrowser = new Intent(Intent.ACTION_VIEW, uriUrl);
  ```

- Launching the Intent:
  ➤ Call the `startActivity()` method, passing in your Intent

  ```java
  startActivity(launchBrowser);
  ```
Text with Links

• Another easy is simply by including links within text on the screen.

• The TextView object can be configured to find these and turn them into clickable links
  ➤ No special formatting commands or tags are needed within the string.

• Example:

```xml
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/contains_links"
    android:textSize="14dp"
    android:autoLink="web" />
```
Google web search

• Enabling Web Searches
  ➤ If content with a Google search, use web search Intent

    android.content.Intent.ACTION_WEB_SEARCH.

    Intent search = new Intent(Intent.ACTION_WEB_SEARCH);

• Supplying Search Criteria
  ➤ Uses SearchManager.QUERY intent extra field for search criteria

    Intent search = new Intent(Intent.ACTION_WEB_SEARCH);
    search.putExtra(SearchManager.QUERY, "pygmy goats");
    startActivity(search);
Becoming a Browser

• For more fine control over web content, use the WebView control.
  ➤ This special view allows fine control over rendering of web content.

• In the following, we describe:
  ➤ How you can embed an Android WebView in your application with the WebKit engine.
  ➤ How to both load an external URL and render custom markup that you supply in-app.
Becoming a Browser

• Setting up the WebView Widget
  ➤ Declare it in a layout file
  ➤ Access it from an Activity and tell it what to do.

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <WebView android:id="@+id/web_engine"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
    />
</LinearLayout>
```
Becoming a Browser

- Requesting Internet Permission
  
  `<uses-permission android:name="android.permission.INTERNET" />

- Load a Web Page
  
  ➤ In your main activity, add the following code into the `onCreate()`:

  ```java
  WebView engine = (WebView) findViewById(R.id.web_engine);
  engine.loadUrl("http://mobile.tutsplus.com");
  ```
Becoming a Browser

• Render Custom Markup

➢ Replace the loadUrl() call with loadData(), which takes three arguments: String htmlData, String mimeType, String encoding

```java
String data = "<html>" +
            "<body><h1>Yay, Mobiletuts+! </h1></body>" +
            "</html>";
engine.loadData(data, "text/html", "UTF-8");
```

• Note

➢ JavaScript should be enabled:

```java
engine.getSettings().setJavaScriptEnabled(true);
```
Becoming a Browser

• Extra features
  ➤ **reload()**: Refreshes and re-renders the page.
  ➤ **goForward()**: Goes one step forward in browser history.
  ➤ **goBack()**: Goes one step back in browser history.

• Use WebSettings class to define the features of the browser.

  ```java
  WebSettings webSettings = webView.getSettings();
  ```
  
  ➤ **setBlockNetworkImage()**: Block network images to reduce the data loading using the method.
  ➤ **setDefaultFontSize()**: Set font size of the displayed web content
  ➤ Other methods: setSaveFormData(), setJavaScriptEnabled(), setSavePassword(), setSaveFormData(), setJavaScriptEnabled(), setSupportZoom()
Android Backup
Overview

- Users store a lot of data on different applications like notes, game data, application settings, address book entries, …
  - All these data cannot be recovered after they are gone.
  - Backup service hosted by Google was introduced in Android 2.2.
  - All the application data can use the backup service to store any data to the cloud.
Creating a Backup of Runtime Data

• Use **BackupManager** class to notify the Backup service to do backup and restore operations.
  ➢ After the notification is received, the backup manager requests backup data from the application and delivers it to a cloud storage server during backup.
  ➢ It also retrieves backup data from the backup transport and returns it to applications during a restore process.

• A backup agent is the interface where the BackupManager communicates with the applications.
  ➢ Extend the **BackupAgent** in their class.
  ➢ Two methods need to be overridden:
    ▪ **onBackup()**: triggered whenever there is a dataChanged() method call.
    ▪ **onRestore()**: triggered whenever there is a requestRestore()
Creating a Backup of Runtime Data

• **onBackup()** has three parameters:
  ➤ **oldState**—Return the state from the last backup.
  ➤ **data**—The data that is backed up
  ➤ **newState**—Write the current state of the backup, which becomes the oldState for the next backup

• **onRestore()** has three parameters:
  ➤ **data**—The data from the last backup.
  ➤ **appVersionCode**—The application’s version code during the backup operation. The version code is defined as the attribute android:versionCode in the Android-Manifest XML file.
  ➤ **newState**—Write the current state as the restore point.
Creating a Backup of Runtime Data

```java
public class MyBackupAgent extends BackupAgent {
    @Override
    public void onCreate() {
        ...
    }

    @Override
    public void onBackup(ParcelFileDescriptor oldState, BackupDataOutput data,
                       ParcelFileDescriptor newState) {
        ...
    }

    @Override
    public void onRestore(BackupDataInput data, int appVersionCode,
                          ParcelFileDescriptor newState) {
        ...
    }
}
```
Backing Up Files to the Cloud

- **BackupAgent** is intended to save application run-time data.
- **BackupAgentHelper** is intended to save files
  - Wrapper class for the backup agent class.
  - Supports two different kinds of backup helpers
    - **SharedPreferencesBackupHelper** to backup SharedPreferences files
    - **FileBackupHelper** to backup file
Triggering Backup and Restore

- To define Backup agent, add **android:backupAgent** attribute inside the application manifest file

```
<application android:label="Backup/Restore" android:backupAgent="myBackupAgent"/>
```

- Anytime the application triggers a backup or restore to the BackupManager, it initiates with the identified backup agent.

```java
public class MyBandRActivity extends Activity {
    BackupManager mBackupManager;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        mBackupManager = new BackupManager(this);
    }
    void dataUpdate() {
        ... // We also need to perform an initial backup; ask for one
        mBackupManager.dataChanged(); //use BackupAgent defined in manifest file
    }
}
```
Triggering Backup and Restore

• Use `requestRestore()` of the BackupManager to trigger restore.
  ➤ triggers a call to the backup agent’s `onRestore()` method

• Also, a factory data reset or when the application is reinstalled trigger restore for the application.

• Android provides a command-line script `bmgr` that can trigger backup/restore
  ➤ Back trigger: “> `adb shell bmgr backup <package>`”
  ➤ Restore trigger: “> `adb shell bmgr restore <package>`”
  ➤ To force the BackupManager to do the backup right away: “> `adb shell bmgr run`”
Android Animation
Android Animation

- Two types of animation: frame-by-frame and Tween animation.
- Frame-by-frame animation needs an animation-list element in the layout file
  ➢ Containing a list of item elements specifying an ordered list of the different pictures to display.
  ➢ `oneshot` attribute specifies whether the animation is played only once or repeatedly
  ➢ Example: res/anim/animated.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<animation-list xmlns:android="http://schemas.android.com/apk/res/android"
    android:oneshot="false">
  <item android:drawable="@drawable/anddev1" android:duration="200" />
  <item android:drawable="@drawable/anddev2" android:duration="200" />
  <item android:drawable="@drawable/anddev3" android:duration="200" />
</animation-list>
```
Android Animation

- To display the frame-by-frame animation,
  ➤ set the animation to a view’s background
  ➤ a drawable can be retrieved by calling `getBackground()` and casting it to `AnimationDrawable`.
  ➤ calling the `start()` method starts the animation.

```java
ImageView im = (ImageView) this.findViewById(R.id.myanimated);
im.setBackgroundResource(R.anim.animated);
AnimationDrawable ad = (AnimationDrawable)im.getBackground();
ad.start();
```
Android Animation

• Tween animation uses a different approach that creates an animation by performing a series of transformations on a single image.

• Android provides access to the following classes that are the basis for all the animations
  ➤ AlphaAnimation—Controls transparency changes
  ➤ RotateAnimation—Controls rotations
  ➤ ScaleAnimation—Controls growing or shrinking
  ➤ TranslateAnimation—Controls position changes

• Used for transitions between activities, layouts, views, etc.

• Defined in the layout XML file as <alpha>, <rotate>, <scale>, and <translate>. 
Example

• Creates a new mail animation that can be used when mail is received.

• Layout file: res/layout/main.xml

```xml
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:gravity="center"
>
    <ImageView
        android:id="@+id/myanimated"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/mail"
    />

    <Button
        android:id="@+id/startAnimated"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="you’ve got mail"
    />
</LinearLayout>
```
Example

- Animation file: res/anim/animated.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
     android:interpolator="@android:anim/accelerate_interpolator">
    <translate android:fromXDelta="100%p" android:toXDelta="0" android:duration="5000" />
    <alpha android:fromAlpha="0.0" android:toAlpha="1.0" android:duration="3000" />
    <rotate android:fromDegrees="0" android:toDegrees="-45"
            android:toYScale="0.0" android:pivotX="50%"
            android:pivotY="50%" android:startOffset="700"
            android:duration="3000" />
    <scale android:fromXScale="0.0" android:toXScale="1.4"
           android:fromYScale="0.0" android:toYScale="1.0"
           android:pivotX="50%" android:pivotY="50%"
           android:startOffset="700" android:duration="3000"
           android:fillBefore="false" />
</set>
```
Example

- Main Activity

```java
package edu.odu.cs.cs495.animation;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.Button;
import android.widget.ImageView;

public class myanimation extends Activity {

  /** Called when the activity is first created. */
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    final ImageView im = (ImageView) this.findViewById(R.id.myanimated);
    final Animation an = AnimationUtils.loadAnimation(this, R.anim.animated);
    im.setVisibility(View.INVISIBLE);
    Button bt = (Button) this.findViewById(R.id.startAnimated);
    bt.setOnClickListener(new OnClickListener(){
      public void onClick(View view){
        im.setVisibility(View.VISIBLE);
        im.startAnimation(an);
      }
    });
  }
}
```
Questions?
To DO

• Examples –

  • Social Networking - Tweeter:
    http://www.cs.odu.edu/~cs495/materials/Lec-18_Tweeter_Example.pdf