Mobile Application (Design and) Development

6th class

Prof. Stephen Intille
s.intille@neu.edu

Q&A

- Anything new?

- Workspace setup in lab
  - Will send an email from systems
  - Please try in Windows lab

Today

- Overview of saving data (part 1)
- Looking at Sudoku code (if time)

- Two design papers
  - Ginsburg, Exploring App Concepts
    Presenter: Jiamin Hu
  - Ginsburg, Prototyping App Concepts
    Presenter: Matt Garnes

Tomorrow

- Start of 2d graphics

- Design:
  One reading one video linked from syllabus

http://www.ccs.neu.edu/home/intille/teaching/MobileApplicationDevelopment2011Syllabus.htm

Saving data

- Your options (most complex apps use all)
  - Shared Preferences
  - Bundles
  - Local files
  - Local database (SQLite)
  - Remote database
- Saving data obviously essential
- New: save data to thwart app assassin

Saving UI state

- Activity should save its user interface state **each time it moves to the background**
  - Required due to OS killing processes
  - Even if you think your app will be in the foreground all the time, you need to do this, because of....
    - Phone calls
    - Other apps
    - The nature of mobile use

Saving other info

- Application preferences
- UI selections
- Data entry

Shared Preferences

- Simple, lightweight key/value pair (or name/value pair NVP) mechanism
- Shared among application components running in the same application context
- Support primitive types: Boolean, string, float, long and integer

Bundles

- Activities offer onSaveInstanceState handler
  - Works like SharedPreferences
  - Bundle parameter represents key/value map of primitive types that can be used save the Activity's instance values
  - Bundle made available as a parameter passed in to the onCreate and onRestoreInstance method handlers
  - Bundle stores info needed to recreate UI state

Using Shared Preferences

- Shared across an application's components, but are not available to other applications
  - Caveat: they can be made available to multiple processes within same application, but be careful
  - Stored as XML in the protected application directory on main memory
    (usually /data/data/[package name]/shared_prefs/[SP Name].xml)

Creating and sharing

```
public static String MY_PREFS = "MY_PREFS";

int mode = Activity.MODE_PRIVATE;
SharedPreferences mySharedPreferences =
      getSharedPreferences(MY_PREFS, mode);

SharedPreferences.Editor editor = mySharedPreferences.edit();

 // Store new primitive types in the shared preferences object.
 editor.putBoolean("isTrue", true);
 editor.putFloat("lastFloat", 1f);
 editor.putInt("wholeNumber", 2);
 editor.putLong("aNumber", 3l);
 editor.putString("textEntryValue", "Not Empty");

 // Commit the changes.
 editor.commit();
```

Retrieving

```
 public static String MY_PREFS = "MY_PREFS";

 public void loadPreferences() {
  // Get the stored preferences
  int mode = Activity.MODE_PRIVATE;
  SharedPreferences mySharedPreferences =
       getSharedPreferences(MY_PREFS, mode);

  // Retrieve the saved values.
  boolean isTrue = mySharedPreferences.getBoolean("isTrue", false);
  float lastFloat = mySharedPreferences.getFloat("lastFloat", 0f);
  int wholeNumber = mySharedPreferences.getInt("wholeNumber", 1);
  long aNumber = mySharedPreferences.getLong("aNumber", 0);
  String stringPreference =
       mySharedPreferences.getString("textEntryValue", "");
}
```

Constants are your friend

```
public static String MY_PREFS = "MY_PREFS";
public static String PREFS_KEY_IS_TRUE = "KEY_IS_TRUE";
public static String PREFS_KEY_WHOLE_NUMBER = "KEY_WHOLE_NUMBER";

int mode = Activity.MODE_PRIVATE;
SharedPreferences mySharedPreferences =
      getSharedPreferences(MY_PREFS, mode);

SharedPreferences.Editor editor = mySharedPreferences.edit();

 // Store new primitive types in the shared preferences object.
 editor.putBoolean(PREFS_KEY_IS_TRUE, true);
 editor.putInt(PREFS_KEY_WHOLE_NUMBER, 2);

 // Commit the changes.
 editor.commit();
```

Can be helpful: put all setter/getters for SPs in one class.

Constants are your friend

```
public static String MY_PREFS = "MY_PREFS";
public static String PREFS_KEY_IS_TRUE = "KEY_IS_TRUE";
public static String PREFS_KEY_WHOLE_NUMBER = "KEY_WHOLE_NUMBER";

int mode = Activity.MODE_PRIVATE;
SharedPreferences mySharedPreferences =
      getSharedPreferences(MY_PREFS, mode);

SharedPreferences.Editor editor = mySharedPreferences.edit();

 // Store new primitive types in the shared preferences object.
 editor.putBoolean(PREFS_KEY_IS_TRUE, true);
 editor.putInt(PREFS_KEY_WHOLE_NUMBER, 2);

 // Commit the changes.
 editor.commit();
```

Can be helpful: put all setter/getters for SPs in one class.

Example

Can be helpful: put all setter/getters for SPs in one class.

Preference Activity

- System-style preference screens
  - Familiar
  - Can merge settings from other apps (e.g., system settings such as location)
- 3 parts
  - Preference Screen Layout (XML)
  - Extension of PreferenceActivity
  - onSharedPreferenceChangeListener

Preference Activity

- XML

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen
  xmlns:android="http://schemas.android.com/apk/res/android">
  <PreferenceCategory
    android:title="My Preference Category"/>
    <CheckBoxPreference
      android:key="PREF_CHECK_BOX"
      android:title="Check Box Preference"
      android:summary="Check Box Preference Description"
      android:defaultValue="true"
    />
  </PreferenceCategory>
</PreferenceScreen>
```

Preference Activity

- Options
  - CheckBoxPreference
  - EditTextPreference
  - ListPreference
  - RingtonePreference

Preference Activity

```
Public class MyPreferenceActivity extends PreferenceActivity
{

@Override
Public void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       addPreferencesFromResource(R.xml.preferences);
}

In manifest...
<activity android:name=“.MyPreferenceActivity”
       android:label=“My Preferences”>

To start...
Intent I = new Intent(this, MyPreferenceActivity.class);
startActivityForResult(i, SHOW_PREFERENCES);
```

Using prefs from Pref. Activity

- Recorded shared prefs are stored in Application Context. Therefore, available to any component:
  - Activities
  - Services
  - BroadcastReceivers

```
Context context = getApplicationContext();
SharedPreferences prefs =
PreferenceManager.getDefaultSharedPreferences(context);
// then Retrieve values using get<type> methods
```

SP change listeners

- Run some code whenever a Shared Preference value is added, removed, modified
- Useful for Activities or Services that use SP framework to set application preferences

SP change listeners

```
public class MyActivity extends Activity implements
        OnSharedPreferenceChangeListener {
  @Override
  public void onCreate(Bundle SavedInstanceState) {
    // Register this OnSharedPreferenceChangeListener
    Context context = getApplicationContext();
    SharedPreferences prefs =
        PreferenceManager.getDefaultSharedPreferences(context);
    prefs.registerOnSharedPreferenceChangeListener(this);
  }

  public void onSharedPreferenceChanged(SharedPreferences prefs,
                                        String key) {
    // TODO Check the shared preference and key parameters
    // and change UI or behavior as appropriate.
  }
}
```

Saving/loading files

- Biggest decision: location
  - Local pros/cons
  - External pros/cons
- Code is standard Java

```
String FILE_NAME = "tempfile.tmp";

// Create a new output file stream that's private to this
// application.
FileOutputStream fos = openFileOutput(FILE_NAME,
                            Context.MODE_PRIVATE);
// Create a new file input stream.
FileInputStream fis = openFileInput(FILE_NAME);
```

Saving/loading files

- Can only write in current application folder or external storage
- Exception thrown otherwise

- For external, must be careful about availability of storage card
  - Behavior when docked
  - Cards get wiped

Check SD card!

```
private static final String ERR_SD_MISSING_MSG = "CITY cannot see your SD card.
Please reinstall it and do not remove it.";

private static final String ERR_SD_UNREADABLE_MSG = "CITY cannot read your SD
(memory) card. This is probably because your phone is plugged into your
computer. Please unplug it and try again.";

public static String getSDCard() throws CITYException {
        if (Environment.getExternalStorageState().equals(
                        Environment.MEDIA_REMOVED))
                throw new CITYException(ERR_SD_MISSING_MSG);
        else if (!Environment.getExternalStorageState().equals(
                        Environment.MEDIA_MOUNTED))
                throw new CITYException(ERR_SD_UNREADABLE_MSG);
        File sdCard = Environment.getExternalStorageDirectory();
        if (!sdCard.exists())
                throw new CITYException(ERR_SD_MISSING_MSG);
        if (!sdCard.canRead())
                throw new CITYException(ERR_SD_UNREADABLE_MSG);
        return sdCard.toString();
}
```

Static files as resources

- Only good option for small files that NEVER change
  - Otherwise, grab on the fly from net
- Put in res/raw folder

```
Resources myResources = getResources();
InputStream myFile =
       myResources.openRawResource(R.raw.myfilename);
```

Other data storage options

- Later in the course:
  - Local database (SQLite)
    - If you need to perform queries on data
    - Content Providers to share data
  - External database
    - Grab info on demand (or overnight)
    - If you can rely on sporadic Internet connectivity

Sudoku

- Looking at/questions about code...