UNIVERSITY *of* WASHINGTON

# Android 210 - Lecture 5

## Files, SharedPreferences & Settings

Margaret Maynard-Reid

February 9, 2015

## Agenda

Topics

- Homework 2 solution
- ActionBar, Toolbar & ViewPager
- Files, SharedPreferences, Settings
- Homework 3 requirements

Demo Code

- SampleToolbar, SampleViewPager*
- SampleStorage
- SampleSettings

## Android Stories

- Google Glass' new boss wants to redesign the headset 'from scratch'
- YouTube Video: Dive into the Gradle-based Android Build System (from AnDevCon 2014)

## Homework 2 Solution

- Walk through Homework 2 solution

## Review from last week

- What is a ListView?
- What is a Fragment?
- How do you add a fragment?
- What is an ActionBar?

## Toolbar

- Introduced in Android 5.0 Lollipop
- Two ways to use a ToolBar:
    1. Use it as an ActionBar
    2. Standalone: can be placed anywhere in the layout
- Greater flexibility in embedding custom views

## Hands-on

- Walk through **SampleToolbar** (as ActionBar)
  - create a layout for the toolbar
  - reference the toolbar layout
  - set it as the actionbar
  - remove the original actionbar in theme (styles.xml)

## Iconography

Guideline for launcher, ActionBar menu icons and generic icons...

http://developer.android.com/design/style/iconography.html

## Break

## Storage Options

- Location: Internal vs External Storage
- Types
    - Files
    - Shared Preferences
    - Databases (next lecture)
    - Internet (3rd Course)

http://developer.android.com/guide/topics/data/data-storage.html

## Storage Options

| Option | Persistent | Data |
|---|---|---|
| Bundle | No | Key/value pair |
| Preferences | Yes | Key/value pair |
| Databases | Yes | Structured |
| Files | Yes | Unstructured |

## Storage

Internal vs External Storage

## Storage

- Internal Storage
  - Privately accessible data in the device memory
  - SQLite Databases
  - Shared Preferences
- External Storage
  - Publicly accessible data in the device storage, i.e. a SD card

## Internal vs External Storage

- Typical Locations
  - Internal - /data/data/{your package}
  - External - /mnt/sdcard/Android/data/{your package}
  - External Package Location - /mnt/asec/{your package}
- User and Profile Locations

  Users and Profiles changes things a bit

  - /mnt/sdcard/Android/data/{your package}
  - /mnt/asec/{your package}
  - /mnt/shell/emulated/#

## Tip: Storage Device Variations

- Manufacturers sometimes do their own thing
  - Don't expose an SDCARD
  - Have more than one SDCARD
  - Make the SDCARD internal only
- Be safe when accessing storage!
  - External storage in particular
  - Check Environment.getExternalStorageState()

## Internal Storage

Get your application storage via the Context

- Context.getFilesDir()
- Context.getCacheDir()

## External Storage

- Get via the Context

  Context.getExternalFilesDir()

- Get public storage via the Environment

  Environment.getExternalStoragePublicDirectory()

- SD Card is available at /sdcard
  - Officially it is at /mnt/sdcard (as of 2.3 and greater)
  - Symbolic link maintains backward compatibility
  - You can open Files via the path names

## Internal vs External Storage

Application Package Location is controlled via manifest entry ***android:installLocation***

- **auto** - install the app wherever there is space. The user can move the app between internal and external storage through the system settings.
- **internalOnly** (Default) - only allow app to be installed in internal storage. User cannot move the app.
- **preferExternal** - prefers that the app be installed in external storage. The user can move app between internal and external storage through the system settings.

## Hands-on

Take a look in Android Studio:

- Android Device Monitor
- DDMS
- File Explorer

## Storage

Files

## Files

Android uses the java.io namespace for File IO

- File
- Reader / Writer based classes
- FileInputStream / FileOutputStream
- BufferedInputStream / BufferedOutputStream
- Etc.

## Files - write to storage

```
File file = new File(context.getFilesDir(), filename); ← internal storage
File file = new File(context.getExternalFilesDir(null), filename);← external storage
String string = “some string here”;

try {
        FileOutputStream fileOutputStream = new FileOutputStream(file);
        fileOutputStream.write(string.getBytes());
        fileOutputStream.close();
  } catch (IOException e) {
      e.printStackTrace();
  }
```

## Hands-on

- Walk through SampleStorage

## Break

## Storage

SharedPreferences

## Shared Preferences

Persistent application Key/Value pair storage

Really intended for "Preferences"

- Supports primitive data types
- Stored in an XML file in internal storage
- "Shared" means shared across same application components (within the process)
- Shared between other applications is more work and involves getting that app Context

## Shared Preferences

- Available via android.content
- Also related to the Preference Storage
- android.preference
  - PreferenceFragment
  - PreferenceManager
  - PreferenceScreen
  - PreferenceGroup
  - ...

## Accessing SharedPreferences

**Get Application SharedPreferences:**

- *Context.getSharedPreferences(String name, int mode)*
  - The name is the preference file created
- Activity.getPreferences(int mode)
  - Calls getSharedPreferences() with Activity's class name as the preference file name
- PreferenceManager.getDefaultSharedPreferences(context)
  - Gets a default named shared preference
  - Defaults to {YOUR_APP_ID}_preferences (i.e. "com.example.app_preferences")
  - Used to access preferences stored by default with your PreferenceScreens, etc.
- PreferenceManager.getSharedPreferences()

## Modes

- MODE_PRIVATE (Default) - allow access only to the calling application
- MODE_WORLD_READABLE - allow all other applications to have read access to the created file
- MODE_WORLD_WRITEABLE - allow all other applications to have write access to the created file
- MODE_MULTI_PROCESS - special loading mode to check for preference modification in applications with multiple processes

## Saving data to a Shared Preference

```
SharedPreferences prefs = getSharedPreferences("my_prefs", 0);
SharedPreferences.Editor editor = prefs.edit();
editor.putString("key1", "value");
editor.commit();
```

***Note: editor.commit() vs. editor apply()***

## Retrieving data from prefs

```
SharedPreferences prefs = getSharedPreferences("my_prefs", 0);

String key1 = prefs.getString("key1", "default");
```

## SharedPreferences Full Example

// Access

*SharedPreferences sharedPreferences = getSharedPreferences();*

// Writing

*SharedPreferences.Editor editor = sharedPreferences.edit();*

*editor.putBoolean("is_on", true); ← saving a key/value pair*

*editor.commit();*

// Reading

*boolean isOn = sharedPreferences.getBoolean("is_on", false); ← retrieve the data, with a default as false.*

## Hands-on

- Walk through SampleStorage - added SharedPreferences for persisting data

## Settings

User Preference

## Settings

- Allow user to set their preferences for app
- Subclasses of Preference:
  - CheckBoxPreference
  - EditTextPreference
  - ListPreference
- Design guideline:

  http://developer.android.com/design/patterns/settings.html

## Settings - how to create it

1. Add **res/xml** directory
2. Create a **preferences.xml** with root <PreferenceScreen> ← defines Settings UI
3. Create a Fragment (extends from PreferenceFragment)
4. Add PreferenceFragment to Activity

## Hands-on

- Walk through SampleSettings

## Homework 3

- Persisting data using SharedPreferences
- Go over requirements
- Due 2/23/2015 6PM