**455-555 Week 6 Session A Notes**

**Chapter Readings:**

Chapter 8 – Data persistence (Shared Preferences review)

Chapter 9 – Intro to Sqlite3 part 1 (see in Powerpoints content area)

Other:

Lab 3 overview

Week 4 Review on Intents

­­­­

**[ Shared Preferences review ]**

Using the SharedPreferences class, you can create named maps consisting of name/value pairs that can be persisted across sessions and *shared* among application components running within the same application sandbox, but that aren’t accessible to other apps.

**Shared Preferences** are suitable in different situations. Example, when the user’s settings need to be ***saved*** or to ***store*** data that can be used in different activities within the app.

As you know, **onPause()** will always be called before your activity is placed in the background or destroyed, so for the data to be saved persistently, it’s preferred to save it in onPause(), which could be *restored* in onCreate() of the activity.

The data stored using shared preferences are kept private within the scope of the application. However, shared preferences are different from that activity’s instance state.

**How is Shared Preferences different from Saved Instance State?**

|  |  |
| --- | --- |
| **Shared Prefs.** | **Instance State** |
| Persist Data across user sessions, even if the app is killed and restarted, or the device is rebooted | Preserves state data across activity instances in the same user session. |
| Data that *should* be remembered across sessions, such as user’s preferred settings or their game score. | Data that *should not* be remembered across sessions, such as the currently selected tab or current state of activity. |
| A common use is to store user preferences | A common use is to recreate the state after the device has been rotated |

**How to Create Shared Preferences?**

The first thing we need to do is to create one shared preferences file per app. So name it with the package name of your app- unique and easy to associate with the app. When you want to get the values, call the **getSharedPreferences()** method. Shared Preferences provide modes of storing the data (private mode and public mode). It is for backward compatibility- use only **MODE\_PRIVATE**to be secure.

**Following are some of the common methods of Shared Preferences**

* **contains(String key**): This method is used to check whether the preferences contain a preference.
* **edit()**: This method is used to create a new Editor for these preferences, through which you can make modifications to the data in the preferences and atomically commit those changes back to the SharedPreferences object.
* **getAll()**: This method is used to retrieve all values from the preferences.
* **getString(String key, String defValue)**: This method is used to retrieve a String value from the preferences.

**Run demo (SharedPrefs) for Week 6.** Notice the log statement in the onPause() override method that fires when you pause the app. Try it. By doing so, you should see a xml created that represents stored data! Look for this file named MySharedPref.xml by opening up your Device File Explorer by going to View in the menu bar, selecting Tool Windows, and then clicking on Device File Explorer. Then, navigate to the data > data folder inside the Device File Explorer. You should see a list of folders with the package names of the apps installed on your device. Find the folder with the package name of your app and expand it. You should see a list of files, including the XML file that contains your app's shared preferences.

Creating multiple shared preferences per an app

Note you can create more than one shared preferences file per app in Android. The SharedPreferences API allows activities and applications to keep preferences in the form of key-value pairs, similar to a Map, that will persist even when the user closes the application. You can use the getSharedPreferences() method to create multiple shared preference files identified by name, or the getPreferences() method to create different preference files for different activities. Additionally, you can use the getDefaultSharedPreferences() method to get the default shared preference file for your entire app.

[ Sqlite tools & CLI ]

Intro to Sqlite

Refs:

* [Downloads - DB Browser for SQLite (sqlitebrowser.org)](https://sqlitebrowser.org/dl/) – all OSes
* [SQLite Download Page](https://www.sqlite.org/download.html) (Windows)
* Android Studio -> Built in [Database Inspector](https://developer.android.com/studio/inspect/database)!

Lab 3 Review!

Use of Custom Adapter.

Consider layouts

Initial List View layout on start up

Text

Description automatically generated

via Adapter – the *binder*! layout on start up

List View Containment

Consider java source files (making use of a Data Source)

Graphical user interface, text, application

Description automatically generated

For *starter* class

Working a timer for the splash screen

**1. Add in a new activity and make it a launcher based one.**

**2. Include a timer in it for automatic loading to another activity file when timer task ends.**

**Sample Code follows…**

**new** Timer().schedule(**new** TimerTask() {  
 @Override  
 **public void** run() {  
 startActivity(**new** Intent(getApplicationContext(), MainActivity.**class**));  
 }  
}, 5000);