Session 8: Preferences

Assignment – Preferences basics.

Problem Statement

a. What is Shared Preferences.

b. What values can be saved in SharedPreferences.

Solution:

1. SharedPreferences are key-value pairs of primitive data types that are saved in a file within an apps file structure. You can then access this file from anywhere within the app to either put data into the file or take data out of the file. You can’t access the file from another app so it’s pretty secure from that point of view. For example use SharedPreferences in a game where you would save the user’s name, high score, and state of the game when they logged off. Then the next time they log in, their score and game level would be retrieved from the preference file and they would continue the game from where they ended it when they logged off.

In order to use shared preferences , one have to call a method getSharedPreferences() that returns a SharedPreference instance pointing to the file that contains the values of preferences.

SharedPreferences sharedpreferences = getSharedPreferences(MyPREFERENCES, Context.MODE\_PRIVATE);

The first parameter is the key and the second parameter is the MODE. Apart from private there are other modes available that are listed below:

Mode and description

1 MODE\_APPEND

This will append the new preferences with the already existing preferences

2 MODE\_ENABLE\_WRITE\_AHEAD\_LOGGING

Database open flag. When it is set , it would enable write ahead logging by default

3 MODE\_MULTI\_PROCESS

This method will check for modification of preferences even if the sharedpreference instance has already been loaded

4 MODE\_PRIVATE

By setting this mode , the file can only be accessed using calling application

5 MODE\_WORLD\_READABLE

This mode allow other application to read the preferences

6 MODE\_WORLD\_WRITEABLE

This mode allow other application to write the preferences

You can save something in the sharedpreferences by using SharedPreferences.Editor class. We will call the edit method of SharedPreference instance and will receive it in an editor object. Its syntax is −

Editor editor = sharedpreferences.edit();

editor.putString("key", "value");

editor.commit();

Apart from the putString method , there are methods available in the editor class that allows manipulation of data inside shared preferences.

1. A SharedPreferences object points to a file containing key-value pairs and provides simple methods to read and write them. Each SharedPreferences file is managed by the framework and can be private or shared.

To write to a shared preferences file, create a SharedPreferences.Editor by calling edit() on your SharedPreferences.

Pass the keys and values you want to write with methods such as putInt() and putString(). Then call commit() to save the changes. For example:

SharedPreferences sharedPref = getActivity().getPreferences(Context.MODE\_PRIVATE);

SharedPreferences.Editor editor = sharedPref.edit();

editor.putInt(getString(R.string.saved\_high\_score), newHighScore);

editor.commit();

Android SharedPreferences allows us to store private primitive application data in the form of key-value pair.

Android stores shared preference settings as XML file in shared\_prefs folder under DATA/data/[application package] directory. The DATA folder can be obtained by calling Environment.getDataDirectory() (usually it is /data).

SharedPreferences is application specific, i.e.) the data is lost when you perform one of the options,

* once you uninstall the application
* once you clear application data (through Settings)