មជ្ឈមណ្ឌលកូរ៉ែ សហ្វវែរ អេច អ ឌី

Korea Software HRD Center

Data Storage

ណែនាំដោយ : បណ្ឌិត គីម ថេខ្យុង

Webcash

KOICA
Korea International Cooperation Agency

មាតិកា

១. SharedPreferences

២. File Access

១. SharedPreferences

SharedPreference APIs ត្រូវគេប្រើដើម្បីផ្ទុកទិន្នន័យជាលក្ខណៈ key-values។

- SharedPreferences Object ចង្អុលទៅកាន់ file ដែលមានផ្ទុក key-values pair ហើយវាមាន method មួយចំនួនដើម្បី read និង write។
- SharedPreferences file នីមួយៗត្រូវបានគ្រប់គ្រងដោយ framework និងអាចជា private ឬ shared

១. SharedPreferences (ក)

កំណត់ចំណាំ: The SharedPreferences APIs ត្រូវគេប្រើសំរាប់តែក្នុងគោលបំណងសំរាប់តែ read និង write key-values pairs ប៉ុណ្ណោះ។

សូមអ្នកកុំច្រលំជាមួយ Preference API ដែលត្រូវគេប្រើដើម្បីបង្កើត interface សំរាប់ app setting

១. SharedPreferences (ត)

យើងអាចបង្កើត shared preference file ថ្មីមួយឬក៏ប្រើ shared preference មួយដែលមានស្រាប់ ដោយប្រើប្រាស់ method ណាមួយក្នុងចំនោម method ២ ខាងក្រោម:

- getSharedPreferences() - ប្រើ method មួយនេះប្រសិនបើអ្នកត្រូវការ shared preferences file ដែលកំណត់ដោយឈ្មោះ។អ្នកអាចហៅវាពីគ្រប់ Context នៅក្នុង app របស់អ្នក។
- getPreferences() - ប្រើ method មួយនេះពីក្នុង activity ណាមួយដែលអ្នកចង់ហៅ preference file នោះទៅប្រើ។វាហៅ default shared preference file ដែលជាកម្មសិទ្ធរបស់ activity ដូច្នេះអ្នកមិនចាំបាច់ប្រាប់ឈ្មោះ preference នោះទេ។

១. SharedPreferences (ត)

ឧទាហរណ៍: ប្រើជាមួយ getSharedPreferences(String,Mode) នៅពេលមានច្រើន preference file ក្នុង activity តែមួយ

```
Context context = getActivity();
SharedPreferences sharedPref = context.getSharedPreferences(
        getString(R.string.preference_file_key), Context.MODE_PRIVATE);
```

๑. SharedPreferences (๓)

ឧទាហរណ៍: ប្រើជាមួយ getPreferences(Mode) នៅពេលមានតែមួយ preference file ក្នុង activity មួយ

```
SharedPreferences sharedPref = getActivity().getPreferences(
                                    Context.MODE_PRIVATE);
```

១. SharedPreferences (ត)

MODE

MODE_APPEND: វានឹងបន្ថែម preference ថ្មីទៅលើ preference មួយដែលមានស្រាប់។

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

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

១. SharedPreferences (ត)

MODE_PRIVATE: File Creation Mode: វាគឺជា default mode ដែលបញ្ជាក់ថា file ដែលយើងបង្កើតអាចហៅមកប្រើប្រាស់តែក្នុង app របស់យើងប៉ុន្នោះ (ឬ app ទាំងអស់ដែល share the same user ID)។

MODE_WORD_READABLE: Constant មួយនេះត្រូវបានគេបោះបង់ចោលនៅក្នុង API level 17 ។ ការបង្កើត World-Readable file គឺវាគ្រោះថ្នាក់ខ្លាំងណាស់ដោយវាប្រៀបបានទៅនឹងប្រហោង security មួយនៅក្នុង app របស់យើងអញ្ចឹង។

MODE_WORD_WRITABLE: Constant មួយនេះត្រូវបានគេបោះបង់ចោលនៅក្នុង API level 17 ។ ការបង្កើត World-Writable file គឺវាគ្រោះថ្នាក់ខ្លាំងណាស់ដោយវាប្រៀបបានទៅនឹងប្រហោង security មួយនៅក្នុង app របស់យើងអញ្ចឹង។

๑. SharedPreferences - WRITE

ឧទាហរណ៍:

```
SharedPreferences sharedPref = getActivity().getPreferences(Context.MODE_PRIVATE);
SharedPreferences.Editor editor = sharedPref.edit();
editor.putInt(getString(R.string.saved_high_score), newHighScore);
editor.commit();
```

๑. SharedPreferences - READ

ឧទាហរណ៍:

```
SharedPreferences sharedPref = getActivity().getPreferences(Context.MODE_PRIVATE);
int defaultValue = getResources().getInteger(R.string.saved_high_score_default);
long highScore = sharedPref.getInt(getString(R.string.saved_high_score), defaultValue);
```

២. File Access

File object គឺវាសាកសមសំរាប់ការ read និង write large amount of data in start-to-finish order without skipping around។

គ្រប់ Android devices ទាំងអស់មាន file storage ពីរគឺ

- internal storage: build-in non volatile memory
- external storage: removable storage medium such as micro SD card

๒. File Access (๓)

ភាពខុសគ្នារវាង internal និង external storage

Internal storage

- It's always available.
- File saved here are accessed only by our app
- When user uninstalls app, System remove all app´s file in internal storage

Internal storage

- It is not always available because user can mount or remove it from device
- It´s world read, so file save here can be read outside of your control
- When user uninstalls app, the system removes your app´s files from here only if you save them in the directory getExternalFilesDir().

២. File Access (៤)

ទោះបីជា apps ត្រូវបាន android system install ក្នុង internal storage by default, អ្នកអាចកំណត់ android:installLocation attribute នៅក្នុង manifest ដូច្នេះ app របស់អ្នកអាចត្រូវ android system install នៅក្នុង external storage។

๒. File Access – Permission

ដើម្បី write ទៅកាន់ external storage អ្នកត្រូវតែសំ WRITE_EXTERNAL_STORAGE permission នៅក្នុង manifest file របស់យើង

```
<manifest ...>
    <uses-permission
      android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    ...
</manifest>
```

២. File Access – Permission (៣)

ចំណាំ: បច្ចុប្បន្នគ្រប់ apps ទាំងអស់អាច read external storage ដោយពុំចាំបាច់សុំនូវ permission នោះទេ។ទោះបីជាយ៉ាងណាក៏ដោយ ករណីនេះនឹងផ្លាស់ប្តូរនូវថ្ងៃខាងមុខ។ដូចនេះសូមអ្នកធ្វើការសុំនូវ READ_EXTERNAL_STORAGE permission នៅពេលដែលអ្នកត្រូវការ read data ពី file ក្នុង external storage។

```
<manifest ...>
    <uses-permission
      android:name="android.permission.READ_EXTERNAL_STORAGE" />
    ...
</manifest>
```

២. File Access – Permission (៣)

ចំណាំ: បច្ចុប្បន្នគ្រប់ apps ទាំងអស់អាច read external storage ដោយពុំចាំបាច់សុំនូវ permission នោះទេ។ទោះបីជាយ៉ាងណាក៏ដោយ ករណីនេះនឹងផ្លាស់ប្តូរនូវថ្ងៃខាងមុខ។ដូចនេះសូមអ្នកធ្វើការសុំនូវ READ_EXTERNAL_STORAGE permission នៅពេលដែលអ្នកត្រូវការ read data ពី file ក្នុង external storage។

```
<manifest ...>
    <uses-permission
      android:name="android.permission.READ_EXTERNAL_STORAGE" />
    ...
</manifest>
```

២. File Access – Write Internal Storage

នៅពេលដែលយើង save file ទៅក្នុង internal storage, យើងអាចជ្រើសរើស directory ដែលសមនឹង file នោះដោយ ហៅមួយក្នុងចំណោម method ទាំងពីរខាងក្រោមនេះ:

- getFilesDir(): return a File ដែលតំណាងអោយ internal directory សំរាប់ app របស់អ្នក
- getCacheDir(): return a File ដែលតំណាងអោយ internal directory សំរាប់ app's files បណ្តោះអាសន្ន។អ្នកត្រូវលុបគ្រប់ file នៅក្នុង tmp directory នោះនៅពេលដែលយើងលែងត្រូវការវា។

๒. File Access – Write Internal Storage (๓)

ពួកយើងអាចបង្កើត File បានតាម 3 របៀបគឺ:

- File() (constructor)
- openFileOutput() (method)
- createTempFile() (method)

๒. File Access – Write Internal Storage (๓)

ឧទាហរណ៍1:

```
File file = new File(Context.getFilesDir() , filename)
```

ឧទាហរណ៍2:

```
FileOutputStream outStream = openFileOutput(filename , Context.MODE_PRIVATE);
```

ឧទាហរណ៍3:

```
File file = File.createTempFile(filename, null, Context.getCacheDir());
```

๒. File Access – Write Internal Storage (๓)

ឧទាហរណ៍:

```
String filename = "myfile";
String string = "Hello world!";
FileOutputStream outputStream;

try {
  outputStream = openFileOutput(filename, Context.MODE_PRIVATE);
  outputStream.write(string.getBytes());
  outputStream.close();
} catch (Exception e) {
  e.printStackTrace();
}
```

๒. File Access – Read Internal Storage

ឧទាហរណ៍:

```
String filename = "myfile";
String tmp= "";
try {
  FileInputStream inputStream = openFileInputStream (filename);
   InputStreamReader inputStreamReader = new InputStreamReader(inputStream);
   BufferedReader bufferedReader = new BufferedReader(inputStreamReader);
  String line;
  while( (line = bufferedReader.readLine()) != null){
          tmp += line + "\n";
 }
} catch (Exception e) {
  e.printStackTrace();
}
```

២. File Access – External Storage Verification

ដោយសារតែ external storage is not always available ដូចជានៅពេលដែល user mounted the storage ទៅ PC ឬក៏បានដក SD card ចេញពីទូរស័ព្ទ។

ដូចនេះអ្នកគួរតែ verify the volume is available មុននឹងអ្នកចូលទៅប្រើប្រាស់វា។អ្នកអាច query the external storage state ដោយប្រើប្រាស់ method getExternalStorageState()។

ប្រសិនបើ return state ស្មើនឹង MEDIA_MOUNTED មានន័យថាអ្នកអាច read និង write file របស់អ្នកបាន។

๒. File Access – External Storage Verification (๓)

ឧទាហរណ៍:

```
/* Checks if external storage is available for read and write */
public boolean isExternalStorageWritable() {
    String state = Environment.getExternalStorageState();
    if (Environment.MEDIA_MOUNTED.equals(state)) {
        return true;
    }
    return false;
}
```

๒. File Access – External Storage Verification (๓)

ឧទាហរណ៍:

```
/* Checks if external storage is available to at least read */
public boolean isExternalStorageReadable() {
    String state = Environment.getExternalStorageState();
    if (Environment.MEDIA_MOUNTED.equals(state) ||
        Environment.MEDIA_MOUNTED_READ_ONLY.equals(state)) {
        return true;
    }
    return false;
}
```

២. File Access – External Storage File Type

ទោះបីជា external storage អាចនឹងកែប្រែបានដោយ user និង apps ដ៏ទៃទៀត យើងក៏អាចកំណត់ប្រភេទ file នោះបានដែរ។

External file អាចមានពីរប្រភេទគឺ:

Public file: គឺជា File ដែល freely available សំរាប់ user និង apps ដ៏ទៃ។នៅពេលដែល user uninstall app របស់អ្នក files ទាំងនេះគួរតែ remain available សំរាប់ user។

Private file: គឺជា File ដែលមិនអាច access បានដោយ user និង apps ដ៏ទៃ។នៅពេលដែល user uninstall app របស់អ្នក files ទាំងនេះនឹងត្រូវ Android System លុបចោល។

៦. File Access – External Storage

ប្រើ getExternalStoragePublicDirectory() method ប្រសិនបើអ្នកចង់ save public file

```
public File getAlbumStorageDir(String albumName) {
    // Get the directory for the user's public pictures directory.
    File file = new File(Environment.getExternalStoragePublicDirectory(
            Environment.DIRECTORY_PICTURES), albumName);
    if (!file.mkdirs()) {
        Log.e(LOG_TAG, "Directory not created");
    }
    return file;
}
```

๒. File Access – External Storage (๓)

ប្រើ getExternalFilesDir() method ប្រសិនបើអ្នកចង់ save private file

```
public File getAlbumStorageDir(Context context, String albumName) {
    // Get the directory for the app's private pictures directory.
    File file = new File(context.getExternalFilesDir(
            Environment.DIRECTORY_PICTURES), albumName);
    if (!file.mkdirs()) {
        Log.e(LOG_TAG, "Directory not created");
    }
    return file;
}
```

២. File Access – External Storage Query Space

ប្រសិនបើយើងដឹងពីទំហំ data ដែលយើងនឹង save ពួកយើងអាចពិនិត្យមើលថាតើ external storage របស់យើងមាន space គ្រប់គ្រាន់ដើម្បីផ្ទុកដែរឬទេដោយប្រើ methods getFreeSpace() ឬ getTotalSpace() វាអាចការពារមិនអោយមាន IOException កើតឡើង។

getFreeSpace(): provide current available space

getTotalSpace(): provide total storage volume

២. File Access – Delete File

- អ្នកគួរតែលុប file ដែលអ្នកលែងត្រូវការប្រើ។យើងអាចលុប file បានដោយ reference ដែលកំពុងចង្អុលទៅកាន់ file នោះរួចហើយហៅ method delete()។

ឧទាះហរណ៍: myFile.delete();

- ប្រសិនបើត្រូវបាន save នៅក្នុង internal storage, អ្នកអាចប្រើ deleteFile() method ដើម្បីលុប file នោះ។

ឧទាហរណ៍: mContext.delete(filename);

២. File Access – Delete File (៣)

ចំណាំ: នៅពេលដែល user uninstall your app, the Android System លុបនូវ file ដូចខាងក្រោម

- Files ទាំងអស់ដែលត្រូវបានគេ save ក្នុង internal storage
- Files ទាំងអស់ដែលត្រូវបានគេ save ក្នុង external storage ដោយប្រើប្រាស់ getExternalFilesDir()។

ទោះពីជាយ៉ាងណាក៏ដោយអ្នកគួរតែលុប all catch file ដោយផ្ទាល់(files ដែលបានបង្កើតក្នុង getCacheDir())។

២. File Access – Delete File (៣)

ចំណាំ: នៅពេលដែល user uninstall your app, the Android System លុបនូវ file ដូចខាងក្រោម

- Files ទាំងអស់ដែលត្រូវបានគេ save ក្នុង internal storage
- Files ទាំងអស់ដែលត្រូវបានគេ save ក្នុង external storage ដោយប្រើប្រាស់ getExternalFilesDir()។

ទោះពីជាយ៉ាងណាក៏ដោយអ្នកគួរតែលុប all catch file ដោយផ្ទាល់(files ដែលបានបង្កើតក្នុង getCacheDir())។