SCIENCES SORBONNE UNIVERSITÉ

Persistent Data

F. Kordon - Sorbonne Université - CC2018

Renault@lrde.epita.fr

Persistent Data

The bundle used during rotation cannot be used!

- Only focuses on currently active datas
- It does not survive to application closing/reopening

You have to chose between on of these

- SharedPreferences
  - **might be slow!**
- FileSystem
  - **Internal: the memory of the device**
  - **External: the SD card (if available)**
  - **Cloud: on the internet**
- Database
  - **Sqlite3**
  - **...**

SharedPreferences

Key-value pair of primitive data types

- String / some_basic_type

One XML file per activity

- Name can be fixed
- Options for this file
  - **MODE_PRIVATE: access only for the activity**
  - **MODE_WORLD_READABLE: read access for everyone**
  - **MODE_WORLD_WRITABLE: write access for everyone**

Acts like Windows registry

Name may be confusing!

- Not for saving ringtone selected by the user but to save any kind of simple data

Manipulating PreferencesFile (1/2)

Naming your Preferences file

- File will be saved at com.example.myapp.MyPreferenceFile

```
public static final String PREFS_NAME = "MyPreferenceFile";

Context context = getActivity();
SharedPreferences sharedPref =
    context.getSharedPreferences(PREFS_NAME),
    Context.MODE_PRIVATE);
```

Reading in the file

```
SharedPreferences settings =
    getSharedPreferences(PREFS_NAME, Context.MODE_PRIVATE);
advertising = settings.getBoolean("withAdvertising", false);
```

Manipulating PreferencesFile (2/2)

Writing in the file

```
SharedPreferences.Editor editor = settings.edit();
editor.putBoolean("withAdvertising", advertising);
// Commit the edits!
editor.commit();
```

- Use apply to propagate your changes atomically
- Use commit to propagate your changes asynchronously

Use the internal Memory

Default behavior: files are private to the application

- Not accessible from the user
- Not accessible from other applications
- When an application is removed, all the associated files are also removed

A dedicated directory per application

- Created when the application is installed
- All files created by the application are in this directory
- We can have access to this directory programmatically
  - **getFilesDir(): the directory**
  - **getFileList(): the list of all files created by the application**
  - **…**

Writing in a file (internal memory)

Fix read / write mode

```
String FILENAME = "hello_file";
String string_hello = "hello world!";
FileOutputStream fos = null;
try {
    fos = openFileOutput(FILENAME, Context.MODE_PRIVATE);
    fos.write(string_hello.getBytes());
    fos.close();
} catch (Exception e) {
      e.printStackTrace();
}
```

Context.MODE_APPEND for appending a current file

getCacheDir: helps to manipulate temporary files

Reading from a file (internal memory)


```
private String readFromFile() {
    String ret = "";
    try {
     InputStream input = openFileInput(FILENAME);
     if ( input != null ) {
        InputStreamReader inputSR = new InputStreamReader(input);
         BufferedReader bufferedReader = new BufferedReader(inputSR);
         String receiveString = "";
        StringBuilder stringBuilder = new StringBuilder();
         while ( (receiveString = bufferedReader.readLine()) != null )
{
             stringBuilder.append(receiveString);
         }
         input.close();
         ret = stringBuilder.toString();
    }
   }
   catch (Exception e) {
   }
 return ret;
}
```

Memory that can be addressed by the User

- Application's directories are private
- Some directories are shared
  - DIRECTORY_PICTURES
  - DIRECTORY_RINGTONES
  - DIRECTORY_MUSIC
  - ...

```
public File getAlbumStorageDir(String albumName) {
    // Get the dir. 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;
}
```

External Memory

- Can be removed anytime!
- Specify AndroidManifest.xml

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

- Test if media is mounted or not

```
String state = Environment.getExternalStorageState();
if (Environment.MEDIA_MOUNTED.equals(state)){
    // some stuff...
}
else if (Environment.MEDIA_MOUNTED_READ_ONLY.equals(state)){
    // some other stuff
}
else{
    // What to do here?
}
```

Data Base & Cloud

Data can be stored in a database

- Full support for SQLite
- All data base are private
- Have a look to
  - **SQLDataBase**
  - **SQLiteQueryBuilder**
- Android suggests to add unique ID for each record

Data can be stored in the cloud

- Warning! Internet is not always available
- Use java.net and android.net
- Think to require permissions in AndroidManifest.xml

Summary

Use SharedPreferences for saving small datas

- Limited to basic types

Use Filesystem otherwise

- Temporary files
  - Internal memory
    - **Always available**
  - External memory
    - **Not always available**

Use database

- When you have a lot of structured data to store

Use Cloud and related API if your app consume a lot of memory