



SHARED PREFERENCES E ARQUIVOS

Professor Emerson Alencar
emerson@imd.ufrn.br

SHARED PREFERENCES

- ▷ É muito comum armazenar configurações simples nos aplicativos, para que o usuário não tenha que defini-las toda vez que executar;
- ▷ No Android pode ser feito com a classe **SharedPreferences**;
- ▷ Informações são salvas no formato "chave/valor".

SHARED PREFERENCES

```
12
13      val pref = getSharedPreferences( name: "configurações", mode: 0)
14
15      val edit = pref.edit()
16      edit.putString("musica","mpb")
17      edit.putBoolean("som", true)
18      edit.commit()
19
20      val musica = pref.getString("musica",null)
21      val som = pref.getBoolean("som", false)
22
23
```

SHARED PREFERENCES

- ▷ Será criado um arquivo chamado *configuracoes.xml* no diretório */data/data/pacotedaaplicacao/shared_prefs*
- ▷ Esses dados podem ser apagados pelo usuário limpe os dados da aplicação nas configurações

Lendo e Escrevendo em arquivos

- ▷ Memória Interna
- ▷ Memória Externa (privada)
- ▷ Memória Externa (pública)

Escrevendo o conteúdo em um arquivo

```
200
201 private fun save(fos: FileOutputStream){
202
203     val lines = TextUtils.split(edtText.text.toString(), expression: "\n")
204     val writer = PrintWriter(fos)
205
206     for (line in lines){
207         writer.println(line)
208     }
209
210     writer.flush()
211     writer.close()
212     fos.close()
213 }
214
```

Lendo o conteúdo em um arquivo

```
215 private fun load(fis: FileInputStream){
216
217     val reader = BufferedReader(InputStreamReader(fis))
218     val sb = StringBuilder()
219
220     do{
221         val line = reader.readLine() ?: break
222         if (sb.isNotEmpty()) sb.append('\n')
223         sb.append(line)
224     }while (true)
225
226     reader.close()
227     fis.close()
228
229     txtText.text = sb.toString()
230 }
231
```

Arquivos na memória interna

- ▶ É possível ler e gravar arquivos na memória interna do aparelho que ficam armazenados no diretório:

/data/data/pacotedaaplicacao/files

```
89
90     private fun saveToInternal() {
91         try {
92             val fos = openFileOutput( name: "arquivo.txt",Context.MODE_PRIVATE)
93             save(fos)
94         }catch (e: Exception){
95             Log.e( tag: "ERROR_FILE", msg: "Erro ao salvar arquivo",e)
96         }
97     }
98
```

```
101     private fun loadFromInternal() {
102         try {
103             val fis = openFileInput( name: "arquivo.txt")
104             load(fis)
105         }catch (e: Exception){
106             Log.e( tag: "ERROR_FILE", msg: "Erro ao salvar arquivo",e)
107         }
108     }
109
```


Arquivos na memória interna

- ▷ O método `openFileOutput(String, Int)` retorna um objeto `FileOutputStream` que permite salvar o arquivo
- ▷ O método `openFileInput(String)` retorna um objeto `FileInputStream` que permite ler o arquivo
- ▷ Assim como `Shared Preferences` o usuário pode apagar esse arquivo indo nas configurações do aparelho
- ▷

Arquivos no cartão de memória

- ▶ Os arquivos salvos no cartão de memória podem ser acessados por qualquer aplicativo ou pelo próprio usuário.
- ▶ Para salvar no cartão de memória deve adicionar uma permissão no *AndroidManifest.xml*

```
6 <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
7 <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
8
9 <application
10     android:allowBackup="true"
11     android:icon="@mipmap/ic_launcher"
```

Arquivos no cartão de memória

- ▷ É preciso checar se o cartão está disponível:

```
126 if(Environment.MEDIA_MOUNTED == state){  
127     //Pode ler e escrever  
128 }else if(Environment.MEDIA_MOUNTED_READ_ONLY == state) {  
129     //Só pode ler  
130 }  
131
```

Arquivos no cartão de memória

- ▶ Ao salvar um arquivo pode-se compartilhar com outra aplicação ou não

```
196
197     private fun getExternalDir(privateDir: Boolean) =
198         //SDCard/Android/data/com.example.projetoarquivos/files
199         if(privateDir) getExternalFilesDir( type: null)
200         //SDcard/DCIM
201         else Environment.getExternalStorageDirectory()
202
203
```


Salvar arquivos no cartão de memória

```
110 // Salvar arquivo na Memória Externa (Cartão de Memória)
111 private fun saveToExternal(privateDir: Boolean) {
112     val state = Environment.getExternalStorageState()
113     |
114     if(Environment.MEDIA_MOUNTED == state){
115         val myDir = getExternalDir(privateDir)
116         try {
117             if (myDir?.exists() == false){
118                 myDir.mkdir()
119             }
120
121             val txtFile = File(myDir, child: "arquivo.txt")
122
123             if(!txtFile.exists()){
124                 txtFile.createNewFile()
125             }
126
127             val fos = FileOutputStream(txtFile)
128             save(fos)
129         }catch (e: Exception){
130             Log.e( tag: "ERROR_FILE", msg: "Erro ao salvar arquivo",e)
131         }
132     }else{
133         Log.e( tag: "ERROR_FILE", msg: "Não é possível escrever no SD Card")
134     }
135 }
```

Ler arquivos no cartão de memória

```
148 // Ler arquivo na Memória Externa (Cartão de Memória)
149 private fun loadFromExternal(privateDir: Boolean) {
150
151     val state = Environment.getExternalStorageState()
152
153     if(Environment.MEDIA_MOUNTED == state
154         || Environment.MEDIA_MOUNTED_READ_ONLY == state){
155
156         val myDir = getExternalDir(privateDir)
157         if(myDir?.exists() == true){
158             val txtFile = File(myDir, child: "arquivo.txt")
159
160             if(txtFile.exists()){
161                 try {
162                     txtFile.createNewFile()
163                     val fis = FileInputStream(txtFile)
164                     load(fis)
165                 }catch (e: Exception){
166                     Log.e( tag: "ERROR_FILE", msg: "Não é possível escrever no SD Card")
167                 }
168             }
169         }
170     }else{
171         Log.e( tag: "ERROR_FILE", msg: "SD Card indisponível")
172     }
173 }
174
175
176
```

Permissões (pasta pública)

```
34
35 
36
37
38
39 ) {
40     when(requestCode){
41         RC_STORAGE_PERMISSION -> {
42             if (grantResults[0] == PackageManager.PERMISSION_GRANTED){
43                 Toast.makeText( context: this,
44                     text: "Permissão Concedida",Toast.LENGTH_LONG).show()
45             }else{
46                 Toast.makeText( context: this,
47                     text: "Permissão Negada",Toast.LENGTH_LONG).show()
48             }
49         }
50     }
51 }
52
53 companion object{
54     val RC_STORAGE_PERMISSION = 0
55 }
56
```

Permissões (pasta pública)

```
57
58 private fun checkStoragePermission(permission: String, requestCode: Int): Boolean {
59     if (ActivityCompat.checkSelfPermission( context: this, permission)
60         != PackageManager.PERMISSION_GRANTED) {
61         if (ActivityCompat.shouldShowRequestPermissionRationale( activity: this, permission)) {
62             Toast.makeText( context: this,
63                 text: "Você tem q habilitar essa permissão para ler ou salvar o arquivo",
64                 Toast.LENGTH_SHORT).show()
65         }
66         ActivityCompat.requestPermissions( activity: this, arrayOf(permission), requestCode)
67         return false
68     }
69     return true
70 }
```


Permissões (pasta pública)

```
// Salvar arquivo na Memória Externa (Cartão de Memória)  
private fun saveToExternal(privateDir: Boolean) {  
  
    val hasPermission = checkStoragePermission(  
        android.Manifest.permission.WRITE_EXTERNAL_STORAGE,  
        RC_STORAGE_PERMISSION  
    )  
  
    if (!hasPermission){  
        return  
    }  
}
```

Permissões (pasta pública)

```
155 // Ler arquivo na Memória Externa (Cartão de Memória)
156 private fun loadFromExternal(privateDir: Boolean) {
157
158     val hasPermission = checkStoragePermission(
159         android.Manifest.permission.READ_EXTERNAL_STORAGE,
160         RC_STORAGE_PERMISSION
161     )
162
163     if (!hasPermission){
164         return
165     }
166
167
168
169     val state = Environment.getExternalStorageState()
170     //...
171
```

Obrigado!
Dúvidas?

Professor Emerson Alencar
emerson@imd.ufrn.br