Temari

1) Introducció
2) Framework Android
3) Projectes Android i Android SDK
4) Activity
5) Fragments, Views i ListViews
6) Intents
7) Layouts i Custom Views
8) Resources i Themes
9) Dialogs, Menus i WebView
10) Persistència de dades
11) Tasques en Background i internet
12) SQLite i content providers
13) Notificacions

10 – Persistència de dades

- Mecanismes d'emmagatzemament
- Shared preferences
- Fitxers interns
- Fitxers com a recursos
- Fitxers externs (SD)

Mecanismes d'emmagatzemament

Persistir: mantenir dades entre diferents execucions i reinicis de l'aplicació o del sistema

Hi ha vàries alternatives:

- **SharedPreferences**: dades pròpies de l'aplicació (no es pot accedir des de altres Apps) de tipus **primitiu**, en format *key-value*
- **Internal Storage:** emmagatzemar dades en fitxers en la memòria del dispositiu
- **External Storage**:emmagatzemar dades en la SD o similar, accessibles des d'altres apps i en connectar el dispositiu per USB
- **SQLite**: dades en una base de dades SQL privada (no es pot accedir des d'altres apps
- **Network storage**: emmagatzemar dades a la xarxa

SharedPreferences (I)

- El mètode mes simple
- Per a dades de tipus primitiu (String, int, float, boolean...) → poc complexes
- En format *key-value* (clau-valor) → semblant als extras

Dos maneres d'obtenir-les:

→ **getPreferences(int mode)** → preferences especifiques per a l'activity actual

→ **getSharedPreferences(String preferencesName, int mode)** → preferences compartides entre diferents components (amb el mateix name)

SharedPreferences (II)

Es pot llegir directament:

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

@Override
protected void onCreate(Bundle state){
   super.onCreate(state);
   . . .

   // Restore preferences
   SharedPreferences settings = getSharedPreferences(PREFS_NAME, 0);
   boolean silent = settings.getBoolean("silentMode", false);
   setSilent(silent);
}
```


SharedPreferences (III)

Per escriure cal un **Editor** i gestionar una transacció:

```
@Override
protected void onStop(){
   super.onStop();

  // We need an Editor object to make preference changes.
  // All objects are from android.context.Context
  SharedPreferences settings = getSharedPreferences(PREFS_NAME, 0);
  SharedPreferences.Editor editor = settings.edit();
  editor.putBoolean("silentMode", mSilentMode);

  // Commit the edits!
  editor.commit();
}
```

Internal Storage (I)

Fitxers en un emmagatzemament dedicat a cada app

Es programa igual que I/O en Java. Escriure a fitxer:

```
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();
}
```


Internal Storage (II)

Es guarden al **/data/data/<appname>/files**

Els diferents modes per escriure son:

- **MODE_PRIVATE** : crea un fitxer nou (o n'elimina el contingut previ), que només la app pot llegir i escriure. **Per defecte**
- **MODE_APPEND**: el contingut s'afegeix al final del fitxer.
- **MODE_WORLD_READABLE:** crea de nou el fitxer, i qualsevol altra app en pot llegir el contingut (però no escriure'l). **Deprecated**
- **MODE_WORLD_WRITABLE:** crea de nou el fitxer, i qualsevol altra app el pot llegir i escriure. **Deprecated**

Internal Storage (III)

Hi ha altres metodes interessants:

getFilesDir()
Gets the absolute path to the filesystem directory where your internal files are saved.

getDir()
Creates (or opens an existing) directory within your internal storage space.

deleteFile()
Deletes a file saved on the internal storage.

fileList()
Returns an array of files currently saved by your application.

getCacheDir()
Gets the cache directory for temporary files (saved on cache folder, may be removed by the system when low on space)

```
private String readFromFile() {

 String ret = "";

 try {
  InputStream inputStream = openFileInput("config.txt");

  if (inputStream != null) {
   InputStreamReader inputStreamReader =
         new InputStreamReader(inputStream);
   BufferedReader bufferedReader =
         new BufferedReader(inputStreamReader);
   String receiveString = "";
   StringBuilder stringBuilder = new StringBuilder();

   while ((receiveString = bufferedReader.readLine()) != null) {
    stringBuilder.append(receiveString);
   }

   inputStream.close();
   ret = stringBuilder.toString();
  }
 } catch (FileNotFoundException e) {
  Log.e("login activity", "File not found: " + e.toString());
 } catch (IOException e) {
  Log.e("login activity", "Can not read file: " + e.toString());
 }

 return ret;
}
```


Fitxers com a recursos

Al **res/raw** hi ha fitxers que no es modifiquen ni processen en generar l'apk

- Per a contingut multimèdia o fitxers estàtics (que **no canvien**)
- No es pot escriure, **només llegir**

Per llegir-los es fa amb **getResources().openRawResource(int id)**

```
InputStream inputStream = ctx.getResources().openRawResource(resId);
```


External Storage (I)

Cal un permis especial al AndroidManifest.xml (per llegir o per escriure-llegir)

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

Cal detectar si esta disponible o no amb **getExternalStorageState()**:

```
public boolean isExternalStorageWritable() {
    String state = Environment.getExternalStorageState();
    if (Environment.MEDIA_MOUNTED.equals(state)) {
        return true;
    }
    return false;
}
```

Molts possibles estats!

External Storage (II)

Es poden desar fitxers **publics** o **privats**

- **Publics** amb **getExternalStoragePublicDirectory()**:

```
// 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");
}
```

- **Privats** amb **getExternalFilesDir()** (només necessita permission amb API<19)

```
File file = new File(context.getExternalFilesDir(
        Environment.DIRECTORY_PICTURES), albumName);
if (!file.mkdirs()) {
    Log.e(LOG_TAG, "Directory not created");
}
```