

# Programação Para Dispositivos Móveis I

## FICHEIROS

2024/\_25 CTeSP – Desenvolvimento para a Web e Dispositivos Móveis

Ricardo Barbosa , [rmb@estg.ipp.pt](mailto:rmb@estg.ipp.pt)

Carlos Aldeias, [cfpa@estg.ipp.pt](mailto:cfpa@estg.ipp.pt)

# Índice

- Ficheiros;
- Armazenamento Interno;
- Armazenamento Externo;
- Notas Finais;
- Leitura Adicional.

# Ficheiros

Em conjunto com SharedPreferences e Databases, Ficheiros são um de três métodos de **persistência de dados** em Android.

- A sua utilização **não é recomendada** (geralmente);
- O Android utiliza os mesmos mecanismos de manipulação de ficheiros do Java.

Os ficheiros podem ser armazenados em:

- Armazenamento interno (Ficam na mesma localização que os outros recursos como os ícones, imagens, música,...);
- Armazenamento externo (ex. cartão SD).

# Ficheiros

## Armazenamento Interno

Os ficheiros gravados neste tipo de armazenamento são privados à aplicação (tal como os resources), e as outras aplicações (e o utilizador) não conseguem obter acesso aos mesmos.

- Quando o utilizador desinstala a aplicação estes ficheiros são também eliminados;

**ATENÇÃO:** O espaço Interno é muito limitado comparativamente com o externo.

# Ficheiros

## Armazenamento Interno –> Criação e Escrita

1. Invocar o método `openFileOutput()`
  - Recebe como parâmetros o nome do ficheiro e o modo de operação;
    - `MODE_PRIVATE` – cria o ficheiro e torna-o privado à aplicação;
    - `MODE_APPEND` – se o ficheiro já existir então adiciona os dados no final deste em vez de o apagar;
  - Retorna um objeto do tipo `FileOutputStream`;
2. Escrever o ficheiro utilizando o método `write()`;
3. Fechar o stream de dados utilizando o método `close()`.

# Ficheiros

## Armazenamento Interno -> Criação e Escrita

```
String file_name = "test_file.txt";
String textContent = "Hello World!";

FileOutputStream fos = null;

try {
    fos = openFileOutput(file_name, this.MODE_PRIVATE);
    fos.write(textContent.getBytes());

} catch (FileNotFoundException fileNotFoundException) {
    Log.e("FILE_NOT_FOUND", fileNotFoundException.getMessage());
} catch (IOException ioException) {
    Log.e("INPUT", ioException.getMessage());
} finally {
    if (fos != null)
        fos.close();
}
```

# Ficheiros

## Armazenamento Interno -> Leitura

1. Invocar o método `openFileInput( )`
  - Recebe como parâmetro o nome do ficheiro a ler;
  - Retorna um objeto do tipo `FileInputStream`;
2. Ler os bytes do ficheiro utilizando o método `read( )`;
3. Fechar o stream de dados utilizando o método `close( )`;

# Ficheiros

## Armazenamento Interno -> Leitura

```
String file_name = "test_file.txt";
FileInputStream fileInputStream = null;
try {
    fileInputStream = openFileInput(file_name );
    int i;
    while((i=fileInputStream.read())≠-1) {
        ...
    }

} catch (FileNotFoundException fileNotFoundException) {
    Log.e("FILE_NOT_FOUND", fileNotFoundException.getMessage());
} catch (IOException ioException) {
    Log.e("INPUT", ioException.getMessage());
} finally {
    if (fileInputStream ≠ null)
        fileInputStream.close();
}
```



# Ficheiros

## Armazenamento Interno –> Outros métodos

`getFilesDir()`

- Obtém o caminho absoluto para o diretório onde os ficheiros são gravados;

`getDir()`

- Cria (ou abre caso exista) o diretório especificado;

`deleteFile()`

- Apaga o ficheiro;

`fileList()`

- Retorna um array com todos os ficheiros existentes no diretório.

# Ficheiros

## Armazenamento Externo

Este tipo de armazenamento pode ser do tipo **removível** (ex. SD Card) ou **embebido** no dispositivo. Neste tipo de armazenamento os ficheiros podem ser adicionados, editados, removidos e copiados pelo utilizador **não havendo controlo de segurança** sobre os mesmos.

- O armazenamento externo pode ficar indisponível caso o utilizador faça *mount* num computador ao armazenamento externo (ligação por USB)

# Ficheiros

## Armazenamento Externo –> Disponibilidade

Antes de fazer qualquer operação sobre o armazenamento externo deve-se verificar qual o estado do mesmo, utilizando o método `Environment.getExternalStorageState()`, o qual pode retornar:

- `Environment.MEDIA_MOUNTED` – o armazenamento está mounted a um computador e não pode ser utilizado;
- `Environment.MEDIA_MOUNTED_READ_ONLY` – só temos acesso de leitura ao armazenamento;
- `Environment.MEDIA_REMOVED` – armazenamento externo não está disponível;
- Entre outros...

# Ficheiros

## Armazenamento Externo -> Permissões [AndroidManifest.xml]

Permissões de leitura e escrita



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

Permissões só de leitura



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

# Ficheiros

## Armazenamento Externo -> Acesso

A path para este armazenamento pode ser obtida utilizando os métodos:

`getExternalFilesDir()`

- Recebe como parâmetro o tipo de diretório:
  - `DIRECTORY_MUSIC`;
  - `DIRECTORY_RINGTONES`;
  - `DIRECTORY_PICTURES`;
  - entre outros...
- Ou `null` caso seja de outro tipo;
- O ficheiro **é apagado** quando a aplicação é desinstalada.

`getExternalStoragePublicDirectory()`

- Recebe como parâmetro o tipo de diretório:
  - `DIRECTORY_MUSIC`;
  - `DIRECTORY_RINGTONES`;
  - `DIRECTORY_PICTURES`;
  - entre outros...
- Ou `null` caso seja de outro tipo;
- O ficheiro **não é apagado** quando a aplicação é desinstalada.

# Ficheiros

## Armazenamento Externo -> Exemplo

```
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("DIRECTORY", "Directory not created");  
    }  
  
    return file;  
}
```

# Notas

## Armazenamento Interno

Está sempre disponível

Ficheiros apenas acessíveis pela aplicação

Quando a aplicação é removida os ficheiros também são removidos

## Armazenamento Externo

Não está sempre disponível. (Ex: Ligado ao PC, retirado o cartão memória, ...)

É visível a todas as aplicações android e pode ser acedido por fontes externas

Quando a aplicação é desinstalada, o sistema android só elimina o ficheiro se a diretoria onde foi gravado foi obtida através de `getExternalFilesDir()`

# Leitura Adicional

## Data and File Storage:

<https://developer.android.com/training/data-storage>



# Programação Para Dispositivos Móveis I

## FICHEIROS

2024/\_25 CTeSP – Desenvolvimento para a Web e Dispositivos Móveis

Ricardo Barbosa , [rmb@estg.ipp.pt](mailto:rmb@estg.ipp.pt)

Carlos Aldeias, [cfpa@estg.ipp.pt](mailto:cfpa@estg.ipp.pt)

Adaptação do conteúdo dos slides de João Ramos [jrmr@estg.ipp.pt](mailto:jrmr@estg.ipp.pt) e Fábio Silva [fas@estg.ipp.pt](mailto:fas@estg.ipp.pt)