



BINAR
ACADEMY

File Processing

Gold - Chapter 6 - Topic 4

Selamat datang di **Chapter 6 Topik 4** *online course* **Android** Binar Academy!



Masih di Chapter 6 ✨

Pada Chapter sebelumnya kita melewati beberapa tools yang membantu dalam pengembangan Android.

Pada topik kali ini, kita akan mempelajari tools yang membantu dalam menangani berbagai jenis file. Tidak lain dan tidak bukan adalah... ✨File Processing✨

Wohoo!! Siapa yang udah nggak sabar? sini~sini~sini~



Detailnya, kita bakal bahas hal-hal berikut ini:

-  Pengenalan File
-  Cara Handle File PDF
-  Cara Handle File Image
-  Cara Handle File Video





Selama menggunakan PC, pasti kamu cukup paham kalau File punya jenis dan formatnya masing-masing.

Nah, File Processing yang dibahas hari ini nggak akan jauh-jauh dari situ.

Apa sih **maksudnya File Processing?**

Jadi apa sih File Processing itu?

File Processing atau **Pemrosesan File** menggunakan prinsip dimana setiap aplikasi memiliki kemampuan untuk mengolah/membaca sebuah data tersendiri.

Dengan hal tersebut, **Pemrosesan File** akan mempengaruhi efisiensi dan efektifitas user ketika menggunakan aplikasi kita.





Penerapan File Processing ini udah sering kamu temui dalam keseharian, lho~

Beberapa contoh penerapan **File Processing** dalam sebuah aplikasi yaitu seperti **pembaca PDF** atau **Microsoft Office**.

Nah, sebagaimana fungsi File Processing berjalan, aplikasi yang disebutkan itu dapat mengelola file tertentu yang nantinya digunakan oleh user untuk perihal lain.



Konsep File Processing pada Android berarti memahami jika aplikasi dapat membaca variasi format File.

Eits.. nggak sekedar itu. Format File ini juga perlu ditinjau lebih detail lagi.

Untuk itu, kita bedah dulu... **Apa sih yang dimaksud File?**

Apa itu file?

File atau dalam bahasa Indonesiannya disebut **berkas** merupakan **kumpulan data dan informasi yang saling berhubungan** dan juga tersimpan di dalam sebuah ruang penyimpanan sekunder.

Biasanya file juga diartikan sebagai **arsip** atau **data yang tersimpan dalam bentuk digital**.

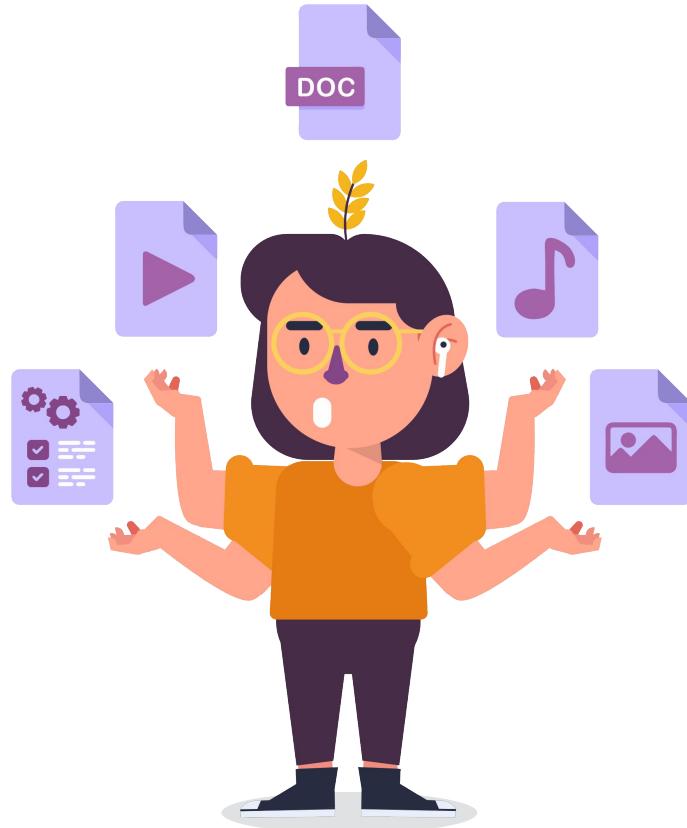


Jenis file dan fungsinya?

Jenis-jenis file yang ada di dalam komputer memiliki format yang sangat banyak. Jika dikelompokkan berdasarkan format filenya, maka akan seperti berikut:

- **File System**
- **File Video**
- **File Dokumen**
- **File Suara**
- **File Gambar**

Kalo dikelompokan seperti di atas, pasti udah tahu dong abis ini kita akan ngapain? Yepp, sesuai ritual, tentunya kita bahas satu persatu~





1. File System

Kamu pernah install aplikasi di komputer kan? Pasti familiar dengan istilah **.exe**. Nah, ini lah yang disebut dengan jenis file system.

File sistem **berfungsi untuk menjalankan program pada komputer** sesuai dengan fungsinya, serta menjalankan berbagai aplikasi yang diinstal ke dalam komputer.

Beberapa **ekstensi** di dalam **file sistem** diantaranya **.sys, .com, .exe, dsb.**



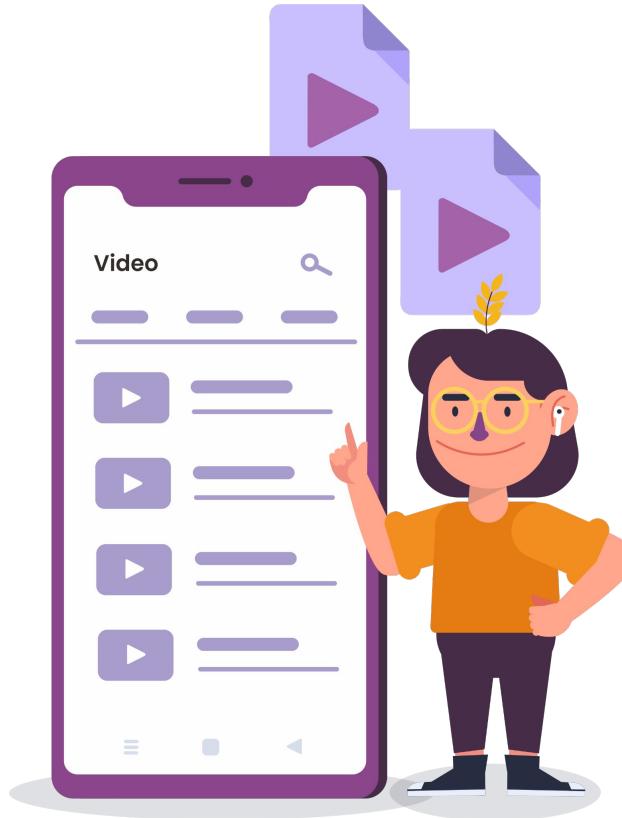


2. File Video

Pernah nonton video? Pernah download MP4 di hape kamu?

Beberapa **ekstensi** pada **file video** diantaranya ialah **.mp4, .3gp, .mkv, dsb.** Setiap ekstensi ini **menunjukkan bahwa masing-masing** mempunyai jenis pemutar yang berbeda.

Tidak seluruh jenis video itu dapat diputar dengan software yang biasanya terinstal di dalam komputer. Terdapat beberapa jenis video yang hanya bisa diputar dengan **software tertentu**.





3. File Dokumen

Jenis file ini pasti familiar dengan kamu yang lagi ngerjain skripsi □

Masing-masing dari ekstensi ini menunjukkan **jenis file dokumennya**, serta cuma bisa dibuka kalau komputer kamu udah install software atau juga aplikasi yang sesuai.

Beberapa ekstensi pada **file dokumen** diantaranya ialah seperti **.doc**, **.xls**, **.pdf**, **.ppt**, **.txt**, dsb.

Skripsi teruuus



4. File Gambar

Yang namanya aplikasi pasti nggak lepas dari menampilkan gambar untuk mempercantik tampilannya.

Beberapa ekstensi file gambar diantaranya seperti **.jpg**, **.png**, **.gif**, dsb. Pada umumnya gambar yang dihasilkan oleh sebuah kamera itu akan berekstensi **.jpg** atau **.jpeg**.

Gambar berekstensi **.tif**, **.png**, serta lainnya biasanya hasil penyimpanan dari software tertentu, misalnya seperti Photoshop, CorelDraw, AutoCad, serta lain-lain.



PNG
JPG
GIF

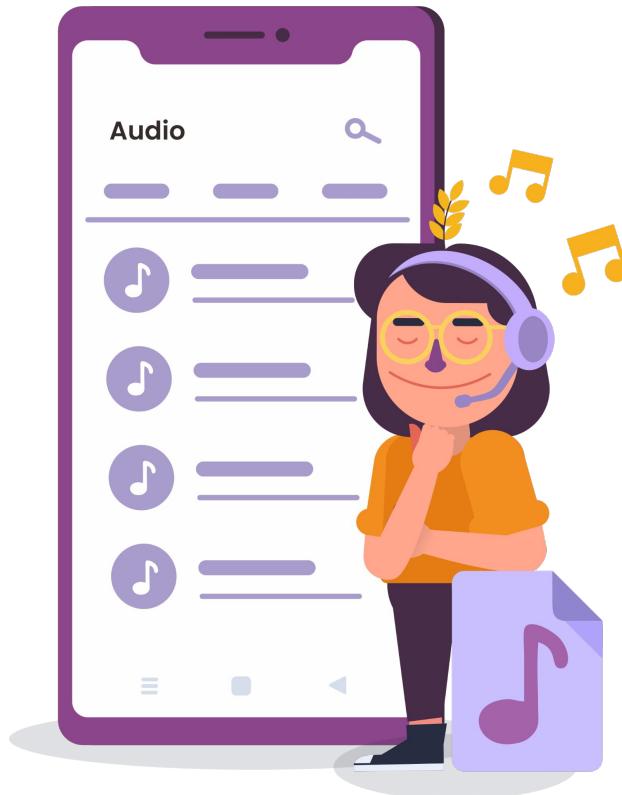


5. File Suara

Beberapa ekstensi file suara ini diantaranya seperti **.wav**, **.mp3**, dan masih buanyaak lagi.

Sama halnya dengan file komputer lainnya, **tidak seluruh file suara** itu bisa dibuka dengan satu aplikasi.

Pada umumnya jenis file ini dikelompokkan dalam **folder khusus** sesuai dengan jenis atau kebutuhannya masing-masing. Maksudnya, file tersebut dapat mudah ditemukan pada saat dibutuhkan.



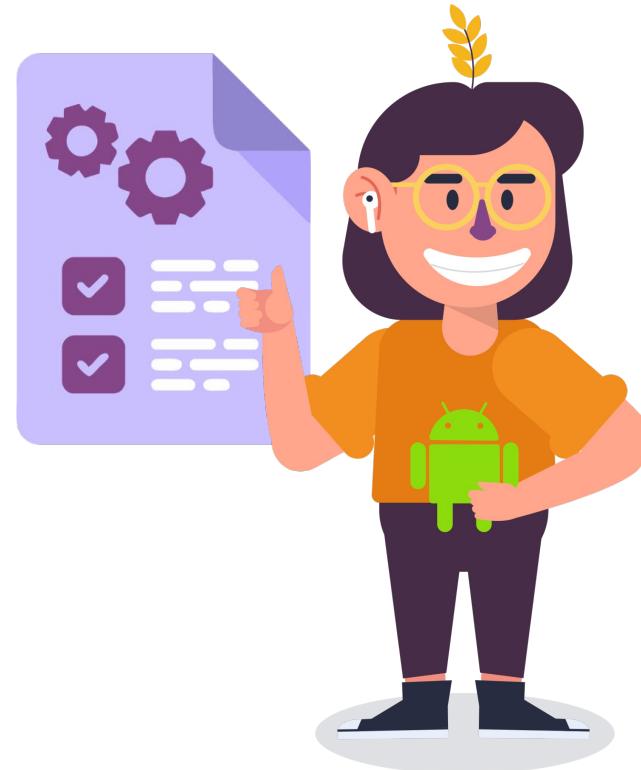


Kesimpulannya

Dari konsep file yang dibahas tadi, ditarik kesimpulan nih kalau **File Processing** merupakan sebuah fungsi dari aplikasi yang dapat membuat, menyimpan atau mengakses konten file.

File Processing juga dapat digunakan untuk menyimpan file baru atau menggeser yang sudah ada.

Semakin memahami konsep File Processing, sambil recall sambil kita intip yuk penerapan **File Processing** di Android ☐





Udah mulai kebayang, kan? bahwa File ini hidup di sekeliling kita.

Biar makin sakti, kayanya belum afdol nih kalau kita nggak praktik~

**Yuk, kita coba implementasi Basic FILE
PROCESSING di Android!**



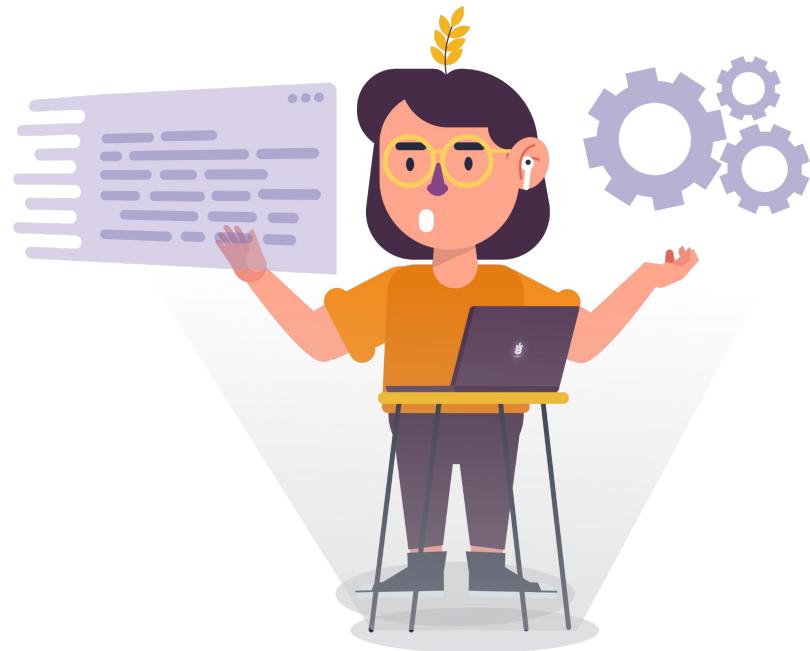


Mau ngapain nih sekarang?

Pada praktek kali ini kita akan coba menerapkan penggunaan **File Processing** dalam sebuah aplikasi. Kita akan membuat sebuah aplikasi dimana aplikasi tersebut :

1. Dapat membaca file **PDF** melalui **WebView**, **assets**, **storage** dan **internet**.
2. Dapat menampilkan gambar dari kamera maupun gallery kita.
3. Dapat memutar video.

Meluncur ke langkah praktiknya, kita geser dulu skuuy~





Step 1: Buat Project Baru

Pertama-tama tentu saja kita perlu membuat sebuah project baru dengan nama **MyFileProcessingPractice**.

Pada fitur pertama ini, kita akan fokus untuk membuat sebuah fitur yang menghandle file PDF.

Setelah projectnya jadi, kita tambahkan **viewBinding** dan juga beberapa **dependency** berikut di **build.gradle**.

```
// ViewModel + LiveData  
implementation "androidx.activity:activity-ktx:1.4.0"  
implementation "androidx.lifecycle:lifecycle-viewmodel-ktx:2.4.1"  
implementation "androidx.lifecycle:lifecycle-livedata-ktx:2.4.1"  
  
// PDFViewers  
implementation 'com.github.barteksc:android-pdf-viewer:2.8.2'  
implementation 'com.mindorks.android:prdownloader:0.6.0'  
  
//ExoPlayer dependency  
implementation 'com.google.android.exoplayer:exoplayer:2.17.1'
```



Selanjutnya untuk kalian yang menggunakan Android Studio versi Artic Fox ke bawah tambahkan kode `jcenter()` pada **build.gradle(Project)**

```
buildscript {  
    ....  
    repositories {  
        ....  
        jcenter()  
    }  
}
```

Sedangkan untuk kalian yang menggunakan Android Studio versi Bumblebee ke atas, tambahkan kode `jcenter()` pada **settings.gradle**

```
pluginManagement {  
    repositories {  
        ....  
        jcenter()  
    }  
}  
dependencyResolutionManagement {  
    repositoriesMode.set(RepositoriesMode.FAIL_ON_PROJECT_REPOS)  
    repositories {  
        ....  
        jcenter()  
    }  
}
```



Step 2: Buat Layout

Lanjut~ Abis itu kita tambahkan syntax di bawah ini pada **activity_layout.xml**

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="56dp"
        android:layout_margin="24dp"
        android:gravity="center"
        android:text="@string/title"
        android:textColor="@color/color_black"
        android:textSize="32sp"
        android:textStyle="bold" />
```

```
<Button
    android:id="@+id/btnPdfHandle"
    android:layout_width="match_parent"
    android:layout_height="56dp"
    android:layout_margin="8dp"
    android:background="@color/colorPrimary"
    android:text="Handle PDF"
    android:textAllCaps="false"
    android:textColor="@android:color/white"
    android:textSize="20sp"
    android:textStyle="bold" />

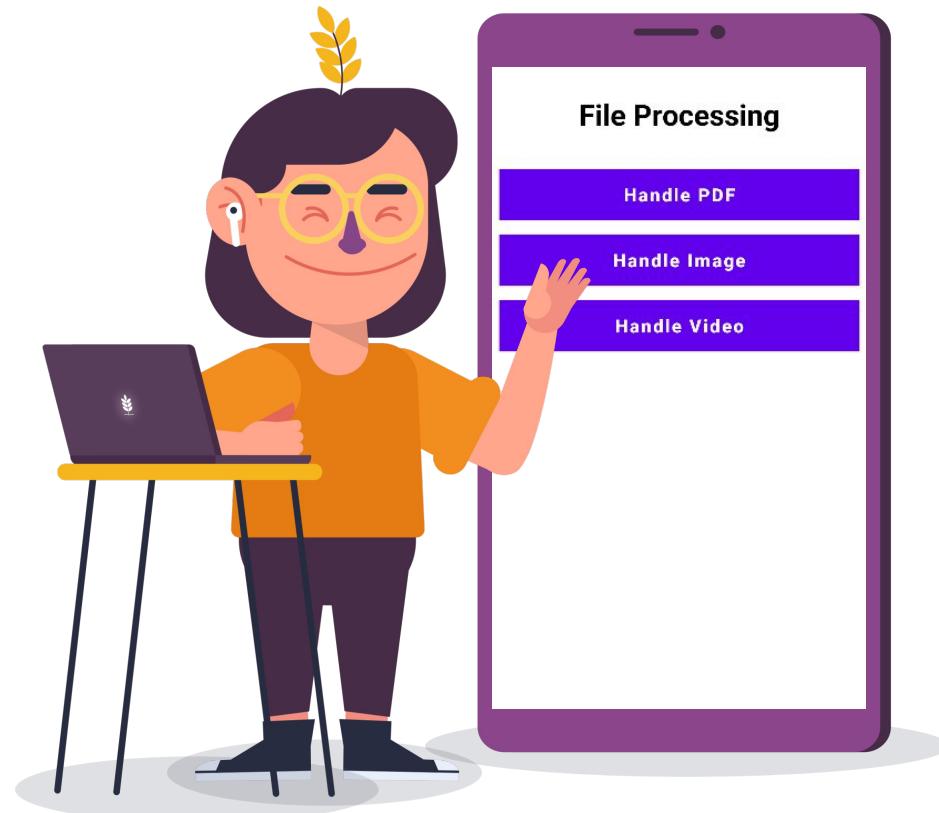
<Button
    android:id="@+id/btnImageHandle"
    android:layout_width="match_parent"
    android:layout_height="56dp"
    android:layout_margin="8dp"
    android:background="@color/colorPrimary"
    android:text="Handle Image"
    android:textAllCaps="false"
    android:textColor="@color/color_white"
    android:textSize="20sp"
    android:textStyle="bold" />

<Button
    android:id="@+id/btnVideoHandle"
    android:layout_width="match_parent"
    android:layout_height="56dp"
    android:layout_margin="8dp"
    android:background="@color/colorPrimary"
    android:text="Handle Video"
    android:textAllCaps="false"
    android:textColor="@color/color_white"
    android:textSize="20sp"
    android:textStyle="bold" />
```

```
</LinearLayout>
```

Tadaaa! Kalau per step sebelumnya kamu melakukan kodenya dengan benar, berarti tampilannya jadi kayak disamping ini ☺☺

Abis ini selesai? Dikit lagii nih. Kita migrasi dulu ke step tiga setelah slide ini yaa~





Step 3: Tambah Layout lagi~

Selanjutnya pada **strings.xml** kita tambahkan value dibawah :

```
● ● ●  
  
<resources>  
    <string name="app_name">MyFileProcessingPractice</string>  
  
    <string name="title">File Processing</string>  
    <string name="web_view">Buka dgn WebView</string>  
    <string name="assets">Buka dari Assets</string>  
    <string name="storage">Buka dari Penyimpanan</string>  
    <string name="internet">Buka dari Internet</string>  
</resources>
```



Step 4: Buat Halaman “Handle PDF”

Pertama-tama kita akan membuat sebuah fitur yang dapat membaca pdf-nya dulu..

Buatlah sebuah Activity baru dengan nama **PdfHandleActivity**.

Setelah itu kita isi layoutnya dengan syntax seperti gambar di samping. Eitss.. tapi syntax ini belum selesai. Berhubung masih panjang, diilanjut ke halaman berikutnya ya~

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".PdfHandleActivity">

    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="56dp"
        android:layout_margin="8dp"
        android:gravity="center"
        android:text="Handle PDF"
        android:textColor="@color/color_black"
        android:textSize="32sp"
        android:textStyle="bold" />

    <Button
        android:id="@+id/buttonWebView"
        android:layout_width="match_parent"
        android:layout_height="56dp"
        android:layout_margin="8dp"
        android:background="@color/colorPrimary"
        android:text="@string/web_view"
        android:textAllCaps="false"
        android:textColor="@android:color/white"
        android:textSize="20sp" />
```



Lanjut



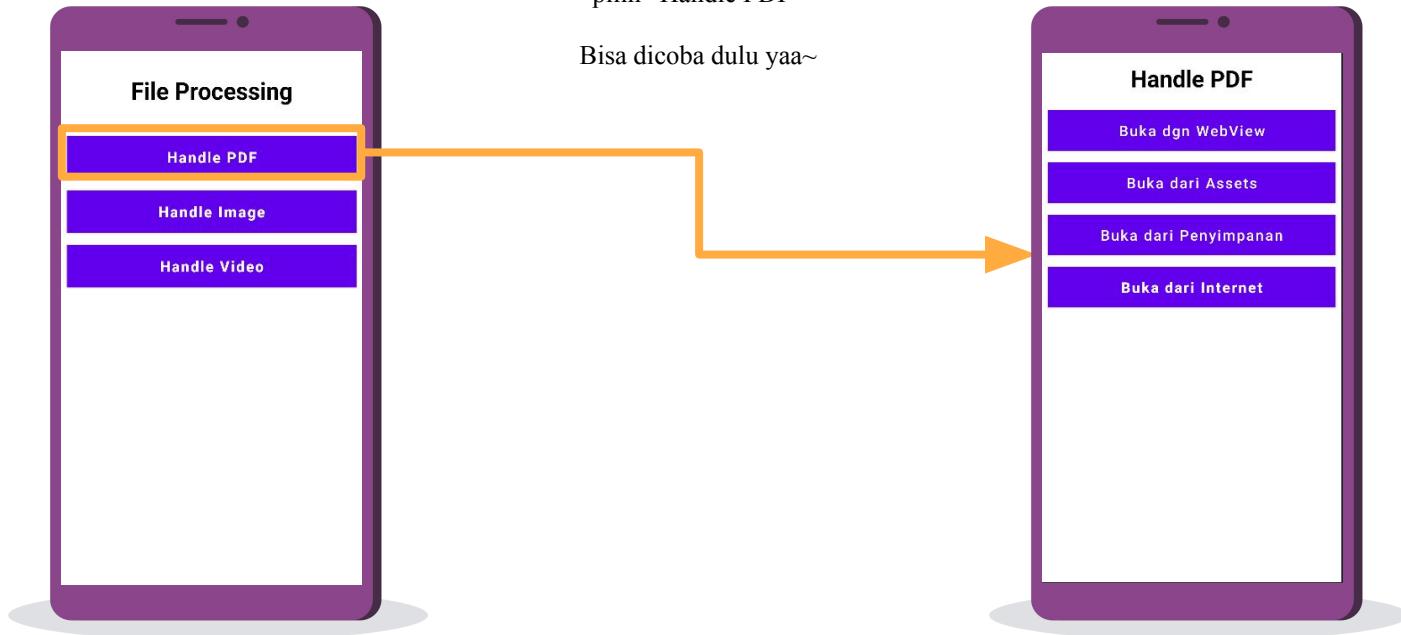
Ini nih buat syntax lanjutannya. Bisa cek gambar di samping

```
<Button  
    android:id="@+id/buttonAssets"  
    android:layout_width="match_parent"  
    android:layout_height="56dp"  
    android:layout_margin="8dp"  
    android:background="@color/colorPrimary"  
    android:text="@string/assets"  
    android:textAllCaps="false"  
    android:textColor="@color/color_white"  
    android:textSize="20sp" />  
  
<Button  
    android:id="@+id/buttonStorage"  
    android:layout_width="match_parent"  
    android:layout_height="56dp"  
    android:layout_margin="8dp"  
    android:background="@color/colorPrimary"  
    android:text="@string/storage"  
    android:textAllCaps="false"  
    android:textColor="@color/color_white"  
    android:textSize="20sp" />  
  
<Button  
    android:id="@+id/buttonInternet"  
    android:layout_width="match_parent"  
    android:layout_height="56dp"  
    android:layout_margin="8dp"  
    android:background="@color/colorPrimary"  
    android:text="@string/internet"  
    android:textAllCaps="false"  
    android:textColor="@color/color_white"  
    android:textSize="20sp"  
    android:textStyle="bold" />  
  
</LinearLayout>
```



Step 5: Cek hasil Layoutnya

Kalau udah jadi, kamu bakal dapetin hasil kayak dibawah ini. Layout ini akan di-setting muncul apabila di menu utama tadi, kamu pilih “Handle PDF”



Okey, Layout settingnya udah jadi. Terus apa lagi nih?

Nah untuk menghandle file PDF terdapat beberapa cara untuk menampilkannya. Untuk bahasan ini, kita akan eksplor setidaknya 4 cara, yaitu:

- **WebView**
- **Assets**
- **Penyimpanan (Local Storage)**
- **Internet**

Yuk kita eksplor satu-satu!





Bener banget! dalam handle file PDF, ada empat cara sebagaimana gambar disamping ini.

Kita bahas sesuai urutannya dan kita meluncur ke urutan pertama.

Gimana sih cara Handle File PDF dengan WebView?





Buka PDF dengan WebView

Cara pertama dan termudah untuk membaca atau menampilkan file PDF adalah dengan menampilkannya di WebView.

Yang perlu kita lakukan cukup dengan memasukkan WebView ke dalam layout dan masukkan URL yang diinginkan dengan menggunakan `webView.loadUrl()`.

Caranya kayak apa tuh? Tenang Broda Brodi, buka tirai abis ini~





Step 1: Proses File PDF

Oke, langsung saja kita buat terlebih dahulu activity baru dengan nama **WebViewActivity**.

Setelah itu kita isi layoutnya dengan syntax di samping:

```
<androidx.constraintlayout.widget.ConstraintLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    tools:context=".WebViewActivity">  
  
    <WebView  
        android:id="@+id/webView"  
        android:layout_width="match_parent"  
        android:layout_height="match_parent" />  
  
</androidx.constraintlayout.widget.ConstraintLayout>
```



Step 2: Buat Fungsi `getPdfUrl()`

Buka class **Utils.kt** yang berada di package utils.

Kita tambahkan fungsi bernama `getPdfUrl()` yang berfungsi untuk **mengembalikan URL PDF** yang akan kita lihat di “WebView”.

Di sini kita menggunakan “link media kit” dari “Kotlinlang”. Kita tambahkan fungsinya ke dalam class **Utils.kt** seperti berikut □□



```
fun getPdfUrl(): String =  
    "https://kotlinlang.org/assets/kotlin-media-kit.pdf"
```



Step 3: Masukan URL ke WebView

Sekarang, kamu buka URL yang sebelumnya dimasukan ke dalam “WebView” dengan memanggil **method webView.loadUrl**.

Kita tambahkan kode seperti yang ada di samping ke dalam file **WebViewActivity.kt**. Kita masukkan di dalam **onCreate()**.

Jadi deh salah satu fungsi kita~

```
override fun onCreate(savedInstanceState: Bundle?) {  
    ....  
    binding.apply {  
        webView.webViewClient = WebViewClient()  
        webView.settings.setSupportZoom(true)  
        webView.settings.javaScriptEnabled = true  
        val url = Utils.getPdfUrl()  
        webView.loadUrl("https://docs.google.com/gview?  
        embedded=true&url=$url")  
    }  
}
```



Step 4: Tambahkan Permission Internet

Sebelum menjalankan fungsi **onCreate()** tadi, **jangan lupa** menambahkan **permission internet**. Karena kita **menggunakan URL** yang mengharuskan kita terkoneksi dengan internet.

Kalo udah, tinggal kita **jalankan** aja deh dan langsung kita lihat reaksinya ☺



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



Keliatan kan File PDF-nya?

Oh iya, karena kita membuka PDF di **WebView**, **kecepatan internet** bakal mempengaruhi cepat atau lambatnya PDF tersebut dibuka.

Jadi **pastikan koneksi internet kamu stabil atau cepat**. Bahkan kalau dibutuhin, bisa minta tetricing sama teman kita □





Sipp deh, Chingu!

Barusan banget, masih anget, dibahas caranya proses file dengan WebView.

Lanjut~ kedua nih, kita bahas cara Handle File PDF dengan Assets, Storage, dan Internet!





Buka PDF lewat Assets, Storage dan Internet

Ada **berbagai library** yang dapat digunakan untuk memudahkan kita menampilkan file PDF di aplikasi.

Pada praktik kali ini kita akan menggunakan **library AndroidPdfViewer**.

Kita juga akan menggunakan **PRDownloader** yang digunakan **untuk mendownload** file dari internet dan membukanya menggunakan **AndroidPdfViewer**.



Kita bisa menggunakan library “AndroidPdfViewer” untuk membuka PDF dari:

- **Folder aset (dari Android Studio)**
- **Penyimpanan telepon**
- **Internet**

Untuk menggunakan library ini, tulis kode-kode yang menghubungkan “button click” pada “PdfHandleActivity.kt” dengan event di atas. Kayak apa caranya? cekidutt~





Step 1: Aktifkan Permission untuk Akses Storage HP

First thing first, kita tambahkan dulu nih permission di bawah supaya kita bisa mengakses storage pada handphone.

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



Step 2: Tambahkan Activity “PDFViewActivity”

Setelah itu kita tambahkan lagi activity baru dengan **PdfViewActivity**. Selanjutnya kita juga perlu **mendadani layout** dari **activity_pdf_view.xml** dulu supaya bisa menampilkan file PDF di sana.

Karena kita menggunakan library **AndroidPdfViewer**, kita dapat dengan mudah untuk menampilkan file PDF di dalamnya.

Kita cukup menambahkan library tersebut di dalam file xml-nya, seperti yang ada di samping.

```
<androidx.constraintlayout.widget.ConstraintLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:app="http://schemas.android.com/apk/res-auto"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    tools:context=".PdfViewActivity">  
  
    <com.github.barteksc.pdfviewer.PDFView  
        android:id="@+id/pdfView"  
        android:layout_width="match_parent"  
        android:layout_height="match_parent" />  
  
    <ProgressBar  
        android:id="@+id/progressBar"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:visibility="gone"  
        app:layout_constraintBottom_toBottomOf="parent"  
        app:layout_constraintEnd_toEndOf="parent"  
        app:layout_constraintStart_toStartOf="parent"  
        app:layout_constraintTop_toTopOf="@+id/pdfView" />  
  
</androidx.constraintlayout.widget.ConstraintLayout>
```



Step 2: Buat Method checkPdfAction()

Kita buat fungsi bernama **checkPdfAction()** di dalam file **PdfViewActivity.kt** dan masukkan kode seperti di samping.

Kalau udah beres, kita **panggil method** tersebut di dalam **onCreate()**.

```
private fun checkPdfAction(intent: Intent) {  
    when (intent.getStringExtra("ViewType")) {  
        "assets" -> {  
            // menampilkan pdf dari assets  
        }  
        "storage" -> {  
            // menampilkan pdf dari storage  
        }  
        "internet" -> {  
            // menampilkan pdf dari internet  
        }  
    }  
}
```

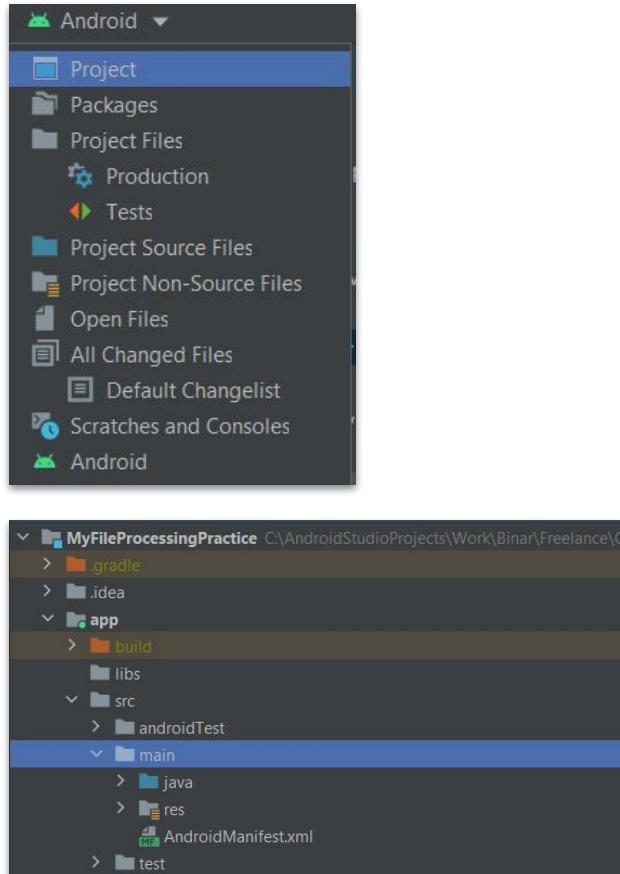
```
override fun onCreate(savedInstanceState: Bundle?) {  
    ....  
    checkPdfAction(intent)  
}
```



Step 3: Buat Folder Assets untuk diisi PDF

Karena fungsi pertama yang ingin dibuat adalah membaca file PDF dari assets, tentu saja kita perlu siapin folder assets yang terdapat di dalam project kita.

Dalam skeleton project yang sudah di-download, terdapat **folder assets** dan juga **file PDF** yang akan digunakan sebagai contoh.

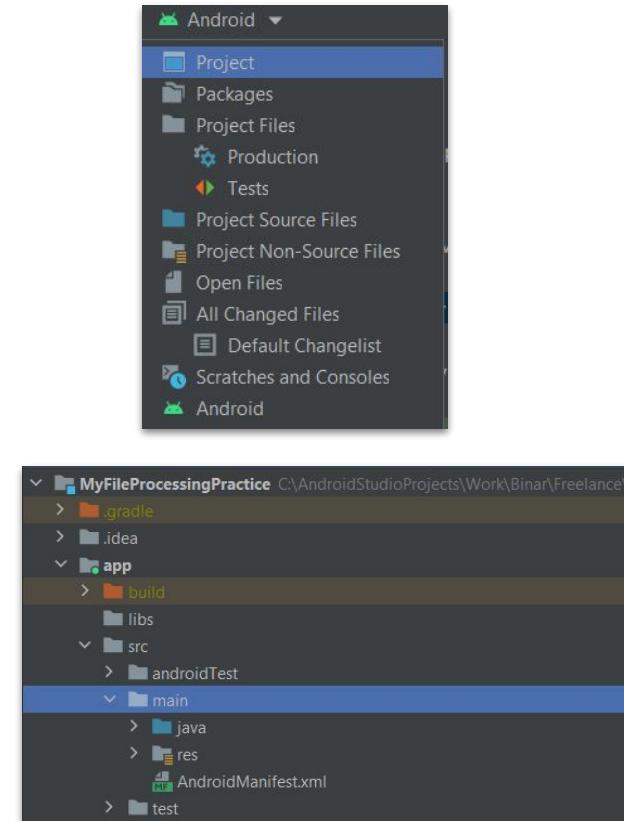




Untuk menambah pengetahuan kita, cara untuk membuat folder assets adalah :

Klik kanan pada root project > New Folder > Assets Folder.

Selanjutnya masukkan lah file pdf di dalam sana, di sini kita akan mencoba memasukkan file pdf dengan nama **Example_PDF_Kotlin_Wikipedia.pdf**.





Step 4: Buat Folder Assets untuk diisi PDF

Setelah itu kita perlu membuat method dengan nama `getPdfNameFromAssets` di dalam class `Utils.kt` seperti yang dibawah ini

```
fun getPdfNameFromAssets(): String {  
    return "Example_PDF_Kotlin_Wikipedia.pdf"  
}
```



Step 5: Tambahkan fungsi `showPdfFromAssets`

Sekarang kita tambahkan kode berikut ke dalam `PdfViewActivity.kt`.

Method ini nih yang nantinya bertugas untuk mengambil nama file dalam format String dan menggunakan method `fromAssets()` dari library `AndroidPdfViewer` untuk menampilkan PDF.

```
private fun showPdfFromAssets(pdfName: String) {
    pdfView.fromAsset(pdfName)
        .password(null)
        // jika memiliki password, masukkan password di sini
        .defaultPage(0) // set halaman default
        .onPageError { page, _ ->
            Toast.makeText(
                this@PdfViewActivity,
                "Error at page: $page", Toast.LENGTH_LONG
            ).show()
        }
        .load()
}
```



Step 6: Tambah Method `checkPdfAction`

Nah terakhir jangan lupa untuk memanggil method tersebut di dalam method `checkPdfAction`.

Kita tambahkan kode seperti gambar disamping:

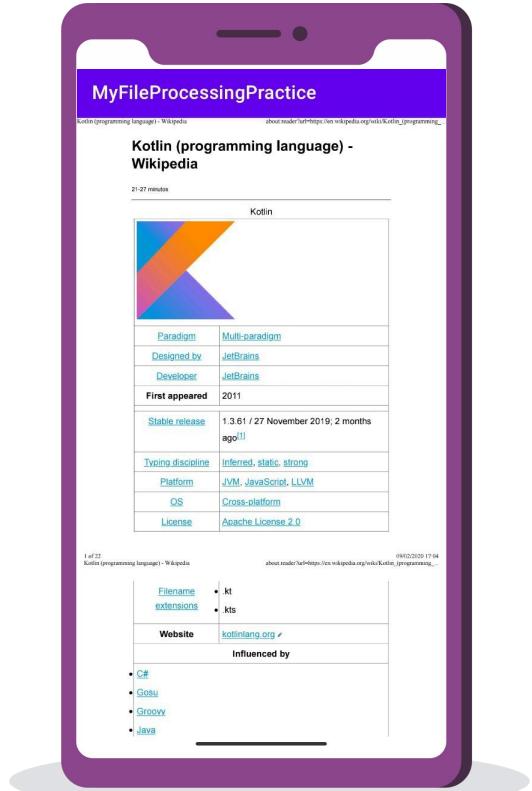
```
private fun checkPdfAction(intent: Intent) {
    when (intent.getStringExtra("ViewType")) {
        "assets" -> {
            // perform action to show pdf from assets
            showPdfFromAssets(Utils.getPdfNameFromAssets())
        }
        ....
    }
}
```



Fungsi Buka PDF via Assets Selesai~

Yoo, fungsi yang kita buat sudah bisa digunakan.

Yuk kita coba~





Wohoo! KBL alias Keren Banget Loch ☺

Okeh, tadi kita bahas cara proses file dengan Assets, Storage, dan Internet.

Si paling ketiga banget nih, kita bahas cara Handle File PDF di Storage Lokal!





Gimana tuh cara panggil PDF dari Storage?

Jadi, kita harus memarsing data menggunakan intent untuk menemukan file yang memiliki format PDF

File yang dipilih nantinya akan ditampilkan dalam **PDFView** dengan memanggil method **fromUri**.





Step 1: Pilih PDF dari Penyimpanan Lokal

Kita buat dulu nih sebuah fungsi **selectedPdfFromStorage()** seperti kode di samping:

```
companion object {
    private const val PDF_SELECTION_CODE = 99
}

-----

private fun selectPdfFromStorage() {
    Toast.makeText(this, "selectPDF", Toast.LENGTH_LONG).show()
    val browseStorage = Intent(Intent.ACTION_GET_CONTENT)
    browseStorage.type = "application/pdf"
    browseStorage.addCategory(Intent.CATEGORY_OPENABLE)
    startActivityForResult(
        Intent.createChooser(browseStorage, "Select PDF"), PDF_SELECTION_CODE
    )
}
```



Step 2: Activity Pemilihan file PDF

Setelah itu kita juga harus menyiapkan sebuah method yang bertugas untuk mengurus fungsi ketika user memilih file PDF.

Tambahkan kode seperti yang ada di bawah ini.

```
override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    super.onActivityResult(requestCode, resultCode, data)
    if (requestCode == PDF_SELECTION_CODE && resultCode == Activity.RESULT_OK && data != null) {
        val selectedPdfFromStorage = data.data
        showPdfFromUri(selectedPdfFromStorage)
    }
}
```



Step 3: Buat Method untuk Tampilkan PDF

Setelah itu kita perlu membuat sebuah method yang bertugas untuk menampilkan PDF dari Uri yang dikirimkan.

Kita buat sebuah method baru dengan nama **showPdfFromUri()**,

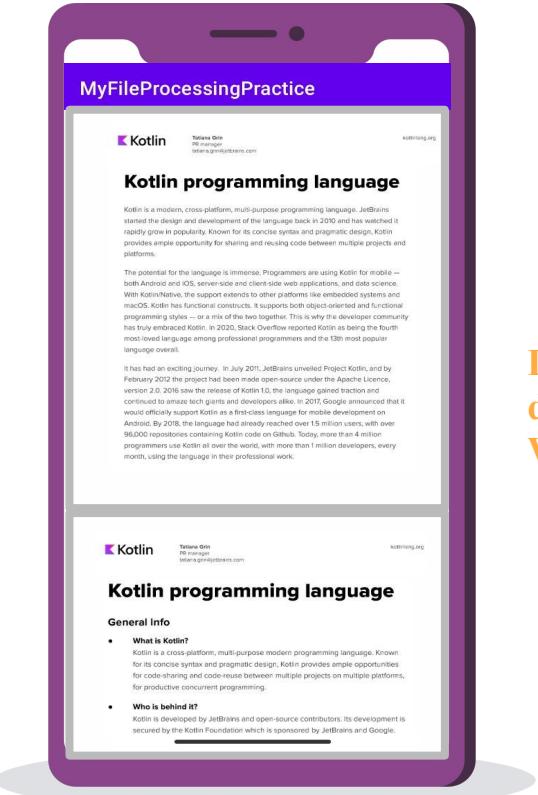
Lalu, kita tambahkan seperti kode di samping.

```
private fun showPdfFromUri(uri: Uri?) {  
    pdfView.fromUri(uri)  
        .defaultPage(0)  
        .spacing(10)  
        .load()  
}
```



Tadaa! Fungsi yang kita buat sudah bisa digunakan. Kita **coba jalankan** aplikasi dan lihat hasilnya seperti apa □

Sebelumnya, jangan lupa nih untuk **pastikan dulu** bahwa kita mempunyai file PDF ya. Karena kalau nggak punya, pasti nggak bisa menampilkan sesuatu **XD**.



Dibuka
dengan
WebView



Wuih!! Apa masih happy dan kiyowo?

Kita meluncur ke bahasan keempat, nih. Jangan lesu karena I'll always be there for you~

Fungsi terakhir adalah **membaca PDF download dari internet.**





Buka PDF dari Internet, gimana caranya?

Kita akan download terlebih dahulu file PDF menggunakan sebuah URL atau link menggunakan **PRDownloader**.

File akan digunakan ini untuk menampilkan file PDF pada **PdfViewActivity.kt** dengan menggunakan proses yang sama seperti di **fungsi assets dan storage**.

Tetapi, di sini kita harus menggunakan **fromFile()** untuk menambahkan tampilan PDF.



```
PRDownloader.initialize(applicationContext)
```



Step 1: Inisiasi PRDownloader

Yuk kita buat fungsi untuk download file PDF tersebut!

Inisialisasikan kode berikut di **PdfViewActivity.kt** pada bagian **onCreate()**



```
PRDownloader.initialize(applicationContext)
```



Step 2: Buat method Download dari PRDownloader

Selanjutnya kita perlu mendownload file dari internet menggunakan method **download()** dari **PRDownloader**.

Jadi, kita buat dulu nih sebuah fungsi dengan nama **downloadPdfFromInternet()** di dalam file **PdfViewactivity.kt**.

Fungsi ini akan mengambil URL, jalur direktori, dan nama file dari file yang akan diunduh. Untuk hal ini, coba tambahkan kode seperti di samping, yaa~

```
private fun downloadPdfFromInternet(url: String, dirPath: String, fileName: String) {
    PRDownloader.download(
        url,
        dirPath,
        fileName
    ).build()
        .start(object : OnDownloadListener {
            override fun onDownloadComplete() {
                Toast.makeText(this@PdfViewActivity, "downloadComplete", Toast.LENGTH_LONG)
                    .show()
                val downloadedFile = File(dirPath, fileName)
                progressBar.visibility = View.GONE
                showPdfFromFile(downloadedFile)
            }

            override fun onError(error: Error?) {
                Toast.makeText(
                    this@PdfViewActivity,
                    "Error in downloading file : $error",
                    Toast.LENGTH_LONG
                )
                    .show()
            }
        })
}
```



Step 3: Tambahkan Fungsi untuk “Download Selesai”

Sekarang kita akan membuat sebuah fungsi untuk menangani ketika download telah selesai atau berhasil.

Coba deh lirik gambar disamping sebentar, ada method yang berwarna ungu.

Kita buat sebuah fungsi dengan nama **showPdfFromFile()**, lalu tambahkan kode seperti di samping.

```
private fun showPdfFromFile(file: File) {
    pdfView.fromFile(file)
        .password(null)
        .defaultPage(0)
        .enableSwipe(true)
        .swipeHorizontal(false)
        .enableDoubletap(true)
        .onPageError { page, _ ->
            Toast.makeText(
                this@PdfViewActivity,
                "Error at page: $page", Toast.LENGTH_LONG
            ).show()
        }
        .load()
}
```



Step 4: Buat Fungsi `getRootDirPath()`

Untuk mendownload PDF dari internet dan membukanya di aplikasi, kita membutuhkan URL, nama direktori, dan nama file dari file yang akan didownload.

Kita bisa mendapatkan URL dengan memanggil method `getPdfUrl()` dari kelas **Utils.kt**.

Sekarang, kita buat sebuah fungsi dengan nama `getRootDirPath()` di kelas **Utils.kt** yang akan bertugas untuk mengembalikan direktori root.

```
fun getRootDirPath(context: Context): String {  
    return if (Environment.MEDIA_MOUNTED ==  
        Environment.getExternalStorageState()) {  
        val file: File = ContextCompat.getExternalFilesDirs(  
            context.applicationContext,  
            null  
        )[0]  
        file.getAbsolutePath  
    } else {  
        context applicationContext.filesDir.getAbsolutePath  
    }  
}
```



Step 5: Panggil Method untuk Download PDF

Terakhir, kita panggil method **downloadPdfFromInternet** dari **checkPdfAction** di **PdfViewActivity.kt**

Kalau udah, abis itu kita coba jalankan aplikasinya. Kita munculkan si PDF satu ini.

Terus, coba untuk mengganti link PDF yang terdapat pada contoh sebelumnya dengan contoh yang kita punya masing-masing.

```
private fun checkPdfAction(intent: Intent) {
    when (intent.getStringExtra("ViewType")) {
        ...
        "internet" -> {
            // perform action to show pdf from the internet
            progressBar.visibility = View.VISIBLE
            val fileName = "myFile.pdf"
            downloadPdfFromInternet(
                Utils.getPdfUrl(),
                Utils.getRootDirPath(this),
                fileName
            )
        }
    }
}
```

Bimsalabim! Tadaaa! □

Empat serangkai dibahas semuanya~

Eitss.. tapi nanti dulu. Kita masih harus tambahin settingan-nya supaya makin sempurna.



Handle PDF

Buka dengan WebView

Buka dari Assets

Buka dari Penyimpanan

Buka dari Internet



Step 1: Tambahkan fungsi-fungsi tadi dalam layout setting “Handle PDF”

Setelah kita dandanin yang lainnya, kita kembali lagi pada class **PdfHandleActivity.kt**.

Kita tambahin kode seperti yang ada di samping ya~

```
override fun onCreate(savedInstanceState: Bundle?) {  
    ....  
    binding.apply {  
        buttonWebView.setOnClickListener {  
            val intent = Intent(this@PdfHandleActivity, WebViewActivity::class.java)  
            startActivity(intent)  
        }  
  
        buttonAssets.setOnClickListener {  
            val intent = Intent(this@PdfHandleActivity, PdfViewActivity::class.java)  
            intent.putExtra("ViewType", "assets")  
            startActivity(intent)  
        }  
  
        buttonStorage.setOnClickListener {  
            val intent = Intent(this@PdfHandleActivity, PdfViewActivity::class.java)  
            intent.putExtra("ViewType", "storage")  
            startActivity(intent)  
        }  
  
        buttonInternet.setOnClickListener {  
            val intent = Intent(this@PdfHandleActivity, PdfViewActivity::class.java)  
            intent.putExtra("ViewType", "internet")  
            startActivity(intent)  
        }  
    }  
}
```



Step 2 :

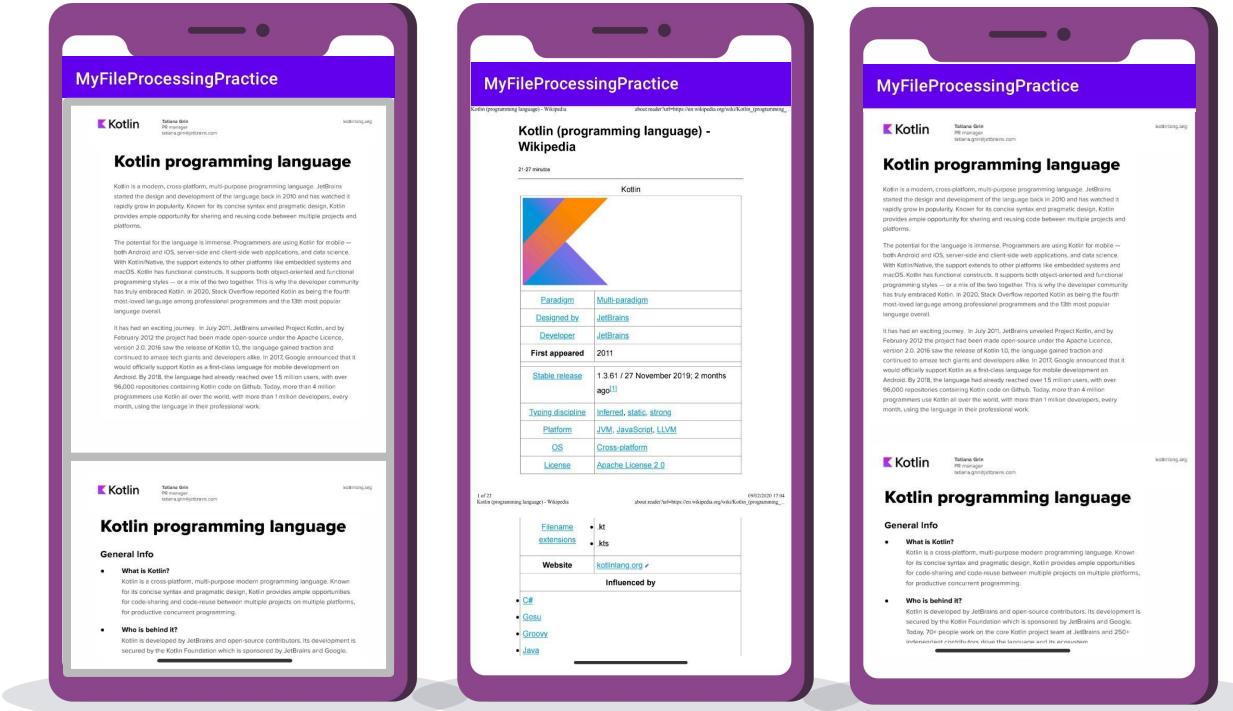
Nah terakhir banget nih, kita tambahin function juga di class MainActivity.kt -nya.

Masukkan seperti kode yang ada di bawah ini~

```
override fun onCreate(savedInstanceState: Bundle?) {  
    .....  
    binding.apply {  
        btnPdfHandle.setOnClickListener {  
            val intent = Intent(this@MainActivity, PdfHandleActivity::class.java)  
            startActivity(intent)  
        }  
    }  
}
```



Setelah semuanya sudah kita masukan, coba jalankan project yang telah dibuat dan coba jalankan masing-masing fitur udah dibuat juga □□



Dibuka dengan WebView

Dibuka dengan Assets

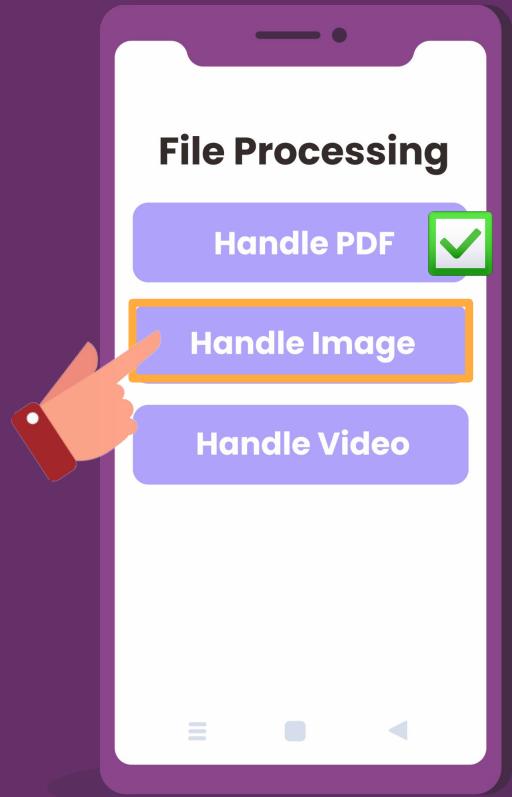
Dibuka dengan Internet



Asiik! Fitur untuk Handle PDF beres ☐

Sebelum lanjut, kita inget-inget dulu. Pada awal materi, ada tiga format file yang bakal dibahas.

Kalau Handle PDF udah beres, berarti kita lanjut ke file berikutnya, **Image**.



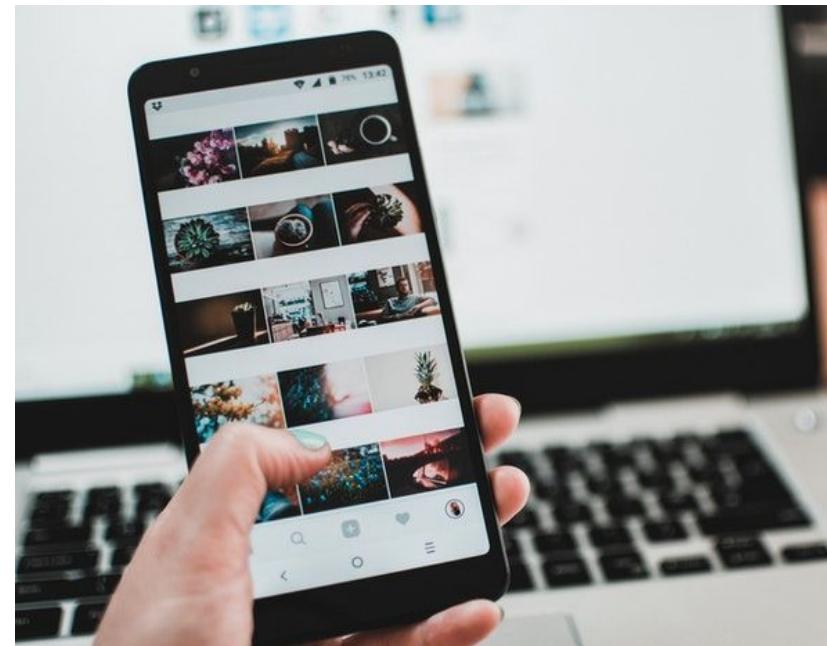


Cara Akses Gambar di Aplikasi

Pastinya kamu nggak asing, kan dengan fitur ini? Biasanya ada di Instagram, Facebook, Whatsapp, dan aplikasi sosial media lainnya.

Untuk mengupload atau akses foto, biasanya aplikasi perlu mengakses galeri handphone kita, atau bahkan bisa langsung ambil gambar lewat kamera..

Pasti kepo dong, kok bisa gitu caranya gimana? Yuk simak step by stepnya~





Step 1: Buat Activity Baru untuk Handle Gambar

Kita akan lanjutkan pembuatan activity dari layout “File Processing” sebelumnya.

Buatlah sebuah activity baru lagi dengan nama **ImageHandleActivity.kt**, setelah itu kita perlu dandani **layoutnya** terlebih dahulu.

Tambahkan syntax seperti di samping □□

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center_horizontal"
    android:orientation="vertical"
    tools:context=".ImageHandleActivity">

    <ImageView
        android:id="@+id/ivImage"
        android:layout_width="match_parent"
        android:layout_height="240dp"
        android:scaleType="fitXY"
        android:src="@drawable/ic_launcher_background"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/btnChoose"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:padding="12dp"
        android:text="Add Image"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/ivImage" />
</LinearLayout>
```



Step 2: Aktifkan Permission akses Kamera HP

Selanjutnya kita tambahkan permission dibawah ini supaya dapat membuka kamera handphone kita



```
<uses-permission android:name="android.webkit.PermissionRequest" />
<uses-permission android:name="android.permission.CAMERA" />
```



Step 3: Buat Fungsi Request User Permission untuk Kamera

Balik lagi ke class **ImageHandleActivity.kt**.

Di sini, kita akan membuat terlebih dahulu function untuk meminta user mengijinkan permission yang kita ajukan.

```
● ● ●

private fun checkingPermissions() {
    // isGranted berfungsi untuk meminta user untuk mengijinkan permission kita
    // di sini kita meminta permission camera, read/write storage
    if (isGranted(
        this, Manifest.permission.CAMERA,
        arrayOf(
            Manifest.permission.CAMERA,
            Manifest.permission.READ_EXTERNAL_STORAGE,
            Manifest.permission.WRITE_EXTERNAL_STORAGE
        ),
        REQUEST_CODE_PERMISSION,
    ))
    ) {
        chooseImageDialog() // jika permission diberikan akan membuka sebuah dialog
    }
}
```



Step 4: Buat Fungsi Request User Permission untuk Kamera

Di sini perlu menambahkan function **isGranted()** yang berfungsi untuk memberitahu kita apakah user sudah memberikan ijin atau belum.

Jika user tidak memberikan ijin, maka function **showPermissionDeniedDialog()** akan dijalankan.

```
private fun isGranted(
    activity: Activity,
    permission: String,
    permissions: Array<String>,
    request: Int,
): Boolean {
    val permissionCheck = ActivityCompat.checkSelfPermission(activity, permission)
    return if (permissionCheck != PackageManager.PERMISSION_GRANTED) {
        if (ActivityCompat.shouldShowRequestPermissionRationale(activity, permission)) {
            showPermissionDeniedDialog()
        } else {
            ActivityCompat.requestPermissions(activity, permissions, request)
        }
        false
    } else {
        true
    }
}
```

Function **isGranted()** akan me-return value Boolean yang menandakan function tersebut sudah diijinkan oleh user atau belum

Jika permission ditolak, akan memanggil function ini



Step 5: Buat Method untuk Dialog saat Permission di tolak User

Ketika user menolak semua cara untuk mengizinkan kita mengakses permission yang dinginkan, cara terakhir untuk “membujuk” user adalah dengan menginfokan kalau permission ini dibutuhkan aplikasi.

Kita dapat menyarankan user untuk mengaktifkannya di “App Settings”.

```
private fun showPermissionDeniedDialog() {
    AlertDialog.Builder(this)
        .setTitle("Permission Denied")
        .setMessage("Permission is denied, Please allow permissions from App Settings.")
        .setPositiveButton(
            "App Settings"
        ) { _, _ ->
            val intent = Intent()
            intent.action = Settings.ACTION_APPLICATION_DETAILS_SETTINGS
            val uri = Uri.fromParts("package", packageName, null)
            intent.data = uri
            startActivity(intent)
        }
        .setNegativeButton("Cancel") { dialog, _ -> dialog.cancel() }
        .show()
}
```



Step 6: Buat Fungsi Dialog Opsi “Gallery or Camera?”

Ketika semua permission diberikan oleh user, Kita perlu menambahkan fungsi Dialog yang memberikan opsi pada user:

“Manakah yang akan dibuka, apakah Gallery atau Camera?”

Buatlah function dengan sintaks di samping.

```
private fun chooseImageDialog() {
    AlertDialog.Builder(this)
        .setMessage("Pilih Gambar")
        .setPositiveButton("Gallery") { _, _ -> openGallery() }
        .setNegativeButton("Camera") { _, _ -> openCamera() }
        .show()
}
```

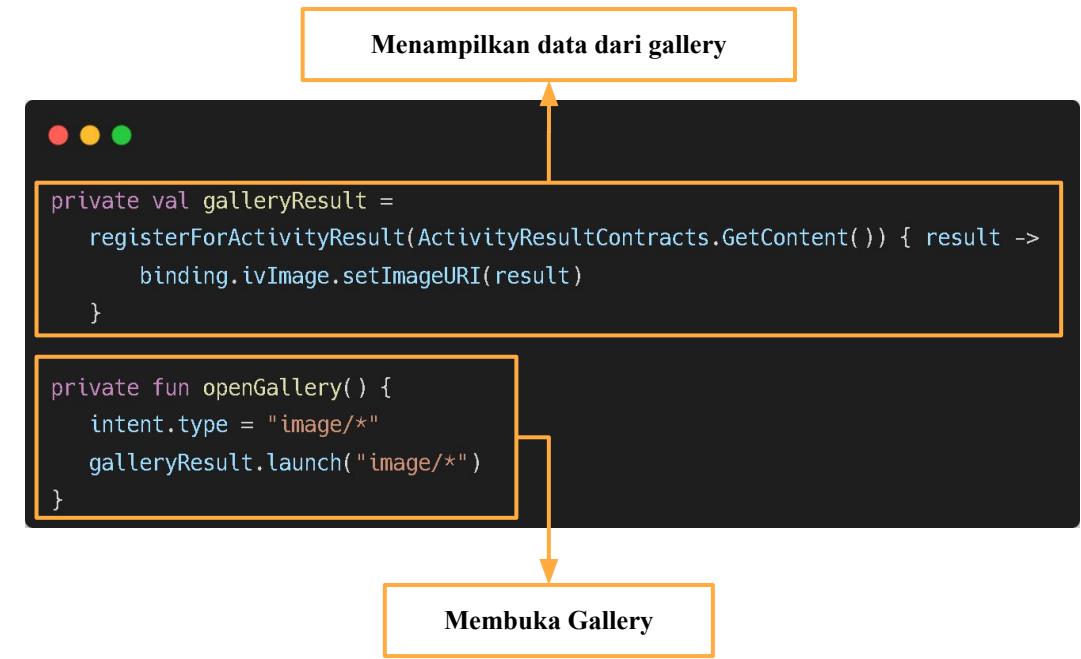


Step 7: Buat Fungsi untuk Buka Gallery dan Menampilkan Data Gambar dan

Perlu diketahui bahwa ketika membuka sebuah gallery atau camera, kita membutuhkan function **activityResult()** untuk mendapatkan data dari gallery.

Setelahnya, kita juga perlu tambahkan function untuk membuka gallery. Untuk hal ini, tambahkan variable seperti disamping, dibawah fungsi **chooseImageDialog()** sebelumnya.

Function untuk mendapatkan data dari gallery kita sudah jadi deh~





Step 8: Buat Fungsi untuk Handle Data dari Kamera

Setelah akses galeri, kita tambahkan fungsi supaya aplikasi bisa membuka kamera dan mendapatkan data dari hasil jepretan kamera kita.

Tambahkan juga variable activityResult-nya untuk camera dengan **cameraResult()** seperti pada kode disamping.

```
private val cameraResult =  
    registerForActivityResult(ActivityResultContracts.StartActivityForResult()) { result ->  
        if (result.resultCode == Activity.RESULT_OK) {  
            handleCameraImage(result.data)  
        }  
    }  
  
private fun handleCameraImage(intent: Intent?) {  
    val bitmap = intent?.extras?.get("data") as Bitmap  
    binding.ivImage.setImageBitmap(bitmap)  
}  
  
private fun openCamera() {  
    val cameraIntent = Intent(MediaStore.ACTION_IMAGE_CAPTURE)  
    cameraResult.launch(cameraIntent)  
}
```



Dibawahnya, tambah function **handleCameraImage()** untuk meng-handle data dari kamera supaya ditampilkan pada “imageView”.

Jika sudah, terakhir kita tambahkan function **openCamera()** untuk membuka camera.

```
private val cameraResult =  
    registerForActivityResult(ActivityResultContracts.StartActivityForResult()) { result ->  
        if (result.resultCode == Activity.RESULT_OK) {  
            handleCameraImage(result.data)  
        }  
    }  
  
private fun handleCameraImage(intent: Intent?) {  
    val bitmap = intent?.extras?.get("data") as Bitmap  
    binding.ivImage.setImageBitmap(bitmap)  
}  
  
private fun openCamera() {  
    val cameraIntent = Intent(MediaStore.ACTION_IMAGE_CAPTURE)  
    cameraResult.launch(cameraIntent)  
}
```



Step 9: Selesai! Coba aplikasinya~

Sudah lengkap semua function yang kita perlukan, terakhir kita tambahkan action pada MainActivity.

Kalau udah, kita coba jalankan aplikasi kita~



```
override fun onCreate(savedInstanceState: Bundle?) {
    ....
    binding.apply {
        ....
        btnImageHandle.setOnClickListener {
            val intent = Intent(this@MainActivity, ImageHandleActivity::class.java)
            startActivity(intent)
        }
    }
}
```



Fitur untuk Handle Image sampai ujung bahasan~

Terakhir bingit, kita belajar membuat fitur untuk
membuka dan menampilkan Video dari kamera
dan galeri kita □

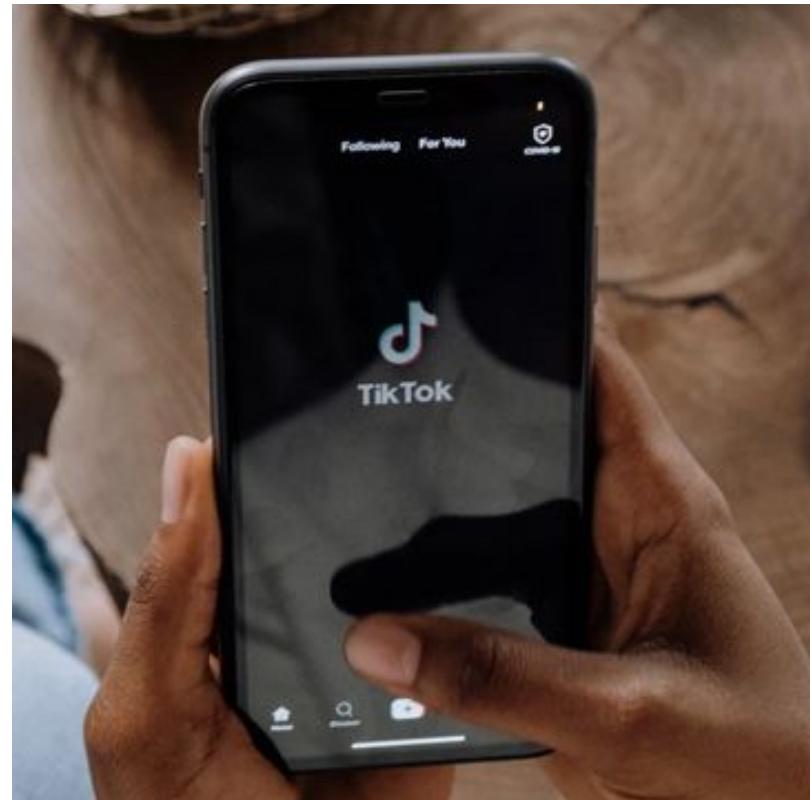




Biar aplikasi bisa akses video, gimana caranya?

Kamu juga pasti familiar dengan fitur yang satu ini. Fitur yang sedang heboh-hebohnya di media sosial, terutama kalau kita ngomongin Youtube dan Tiktok~

Supaya rasa ingin tahu kita bisa terjawab, kita simak langsung yuk langkah-langkahnya~





Step 1: Buat Activity dan Layout untuk Handle Video

Buatlah sebuah activity baru lagi dengan nama **VideoHandleActivity.kt**.

Kalau udah kita perlu dandani **layoutnya** terlebih dahulu. Untuk melakukan hal tersebut, tambahkan syntax seperti di samping.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".VideoHandleActivity">

    <com.google.android.exoplayer2.ui.PlayerView
        android:id="@+id/playerView"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```



Step 2: Buat Variable untuk panggil Exoplayer dan Link video

Setelah itu kita balik lagi ke class **VideoHandleActivity.kt**, di sini kita membuat variable global untuk menggunakan **ExoPlayer**, dan juga yang menampung **videoUrl**.

Selanjutnya pada **onCreate()** kita buat sebuah function dengan nama **initExoPlayer()**, seperti di bawah ini:

```
● ● ●

private var mPlayer: ExoPlayer? = null

private val videoURL = "https://commondatastorage.googleapis.com/gtv-videos-
bucket/sample/BigBuckBunny.mp4"

override fun onCreate(savedInstanceState: Bundle?) {
    .....
    initExoPlayer()
}
```



Step 3: Lengkapi Fungsi ExoPlayer untuk menampilkan Video

Nah, function `initExoPlayer` bertugas untuk menyiapkan `exoPlayer` supaya bisa menampilkan video.

Tambahkan sintaks seperti gambar berikut □□

```
private fun initExoPlayer() {  
    // Create a player instance.  
    mPlayer = ExoPlayer.Builder(this).build()  
  
    // Bind the player to the view.  
    binding.playerView.player = mPlayer  
  
    //setting exoplayer when it is ready.  
    mPlayer!!.playWhenReady = true  
  
    // Set the media source to be played.  
    mPlayer!!.setMediaSource(buildMediaSource())  
  
    // Prepare the player.  
    mPlayer!!.prepare()  
}  
  
//creating mediaSource  
private fun buildMediaSource(): MediaSource {  
    // Create a data source factory.  
    val dataSourceFactory: DataSource.Factory = DefaultHttpDataSource.Factory()  
  
    // Create a progressive media source pointing to a stream uri.  
    val mediaSource: MediaSource = ProgressiveMediaSource.Factory(dataSourceFactory)  
        .createMediaSource(MediaItem.fromUri(videoURL))  
  
    return mediaSource  
}
```



Step 4: Atur Lifecycle Fungsi exoPlayer

Terakhir pada activity lifecycle, kita juga perlu mengatur function exoPlayer supaya tetap berjalan walaupun pernah digunakan.

```
override fun onStart() {
    super.onStart()
    if (Util.SDK_INT >= 24) {
        initExoPlayer()
    }
}

override fun onResume() {
    super.onResume()
    if (Util.SDK_INT < 24 || mPlayer == null) {
        initExoPlayer()
    }
}

override fun onPause() {
    super.onPause()
    if (Util.SDK_INT < 24) {
        releasePlayer()
    }
}
```

```
override fun onStop() {
    super.onStop()
    if (Util.SDK_INT >= 24) {
        releasePlayer()
    }
}

private fun releasePlayer() {
    if (mPlayer == null) {
        return
    }
    //release player when done
    mPlayer!!.release()
    mPlayer = null
}
```



Step 5: Tambahkan Method Action dan Ujicoba

Sudah lengkap semua function yang diperlukan, terakhir tambahkan action pada **MainActivity**.

Kalau sudah selesai, kita coba jalankan aplikasi kita~

```
override fun onCreate(savedInstanceState: Bundle?) {  
    ....  
    binding.apply {  
        ....  
        btnVideoHandle.setOnClickListener {  
            val intent = Intent(this@MainActivity, VideoHandleActivity::class.java)  
            startActivity(intent)  
        }  
    }  
}
```

Horeee, sudah kita coba semua~

Akhirnya lengkap sudah contoh aplikasi yang dapat menghandle beberapa file yang sering digunakan pada Android.

Untuk project lengkapnya dapat kita lihat [di sini](#).

<https://github.com/Abikayusri/MyFileProcessingPractice>



Saatnya kita Quiz!





1) Apa yang dimaksud dengan File Processing?

- A. Sebuah proses dimana setiap aplikasi memiliki kemampuan untuk mengolah/membaca sebuah data tersendiri
- B. Sebuah file yang dapat memproses sesuatu
- C. Sebuah proses yang menggunakan sebuah file

1) Apa yang dimaksud dengan File Processing?

- A. Sebuah proses dimana setiap aplikasi memiliki kemampuan untuk mengolah/membaca sebuah data tersendiri
- B. Sebuah file yang dapat memproses sesuatu
- C. Sebuah proses yang menggunakan sebuah file

File Processing merupakan sebuah proses yang menggunakan prinsip setiap aplikasi memiliki kemampuan untuk mengolah/membaca sebuah data tersendiri.

2) Apa yang dimaksud dengan File?

- A. Sebuah kertas yang dimasukkan ke dalam binder
- B. Sebuah fakta mentah yang belum diolah
- C. Kumpulan data dan informasi yang saling berhubungan



2) Apa yang dimaksud dengan File?

- A. Sebuah kertas yang dimasukkan ke dalam binder
- B. Sebuah fakta mentah yang belum diolah
- C. Kumpulan data dan informasi yang saling berhubungan

File atau Berkas merupakan kumpulan data dan informasi yang saling berhubungan dan juga tersimpan di dalam sebuah ruang penyimpanan sekunder. Biasanya file juga diartikan sebagai arsip.



3) Berikut adalah jenis-jenis file yang benar bisa diproses dalam aplikasi adalah

- A. File Sistem, File Suara, File Gambar
- B. File Gambar, File Kertas, File Video
- C. File Suara, File Data, File Informasi



3) Berikut adalah jenis-jenis file yang benar bisa diproses dalam aplikasi adalah

- A. File Sistem, File Suara, File Gambar
- B. File Gambar, File Kertas, File Video
- C. File Suara, File Data, File Informasi

Pada materi sebelumnya kita telah mempelajari bahwa file memiliki beberapa jenis. Diantaranya file sistem, file video, file dokumen, file suara, dan file gambar.



4) Bagaimana cara termudah untuk menampilkan PDF di Android?

- A. Menggunakan AndroidPdfViewer
- B. Menggunakan WebView
- C. Menggunakan ViewModel

4) Bagaimana cara termudah untuk menampilkan PDF di Android?

- A. Menggunakan AndroidPdfViewer
- B. Menggunakan WebView
- C. Menggunakan ViewModel

Cara termudah untuk menampilkan PDF di Android adalah dengan menggunakan WebView. Hanya saja kekurangan dari menggunakan WebView adalah pengaruh koneksi internet akan menyebabkan hasil menjadi lama muncul.



5) Method yang dipanggil pada Main thread sebelum proses asynchronous dijalankan adalah

- A. ViewerPdfAndroid
- B. AndroidPdfViewer
- C. NitroPdf



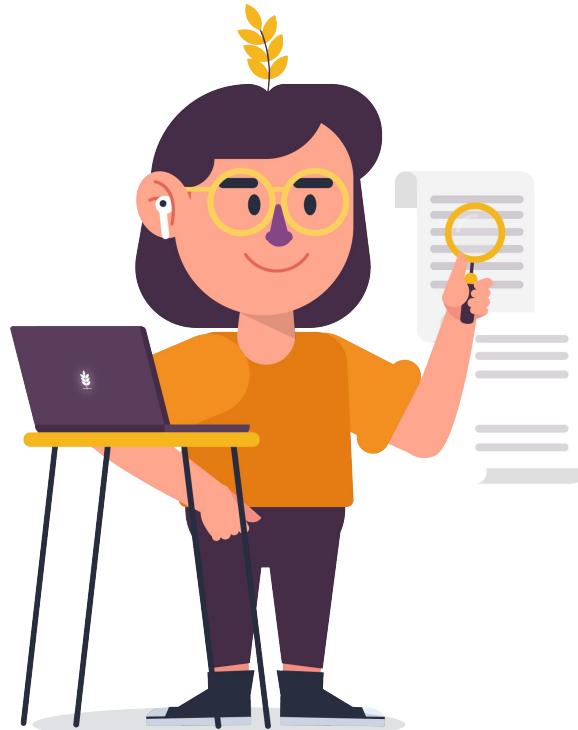
5) Method yang dipanggil pada Main thread sebelum proses asynchronous dijalankan adalah

- A. ViewerPdfAndroid
- B. AndroidPdfViewer
- C. NitroPdf

Untuk membantu kita menampilkan file PDF di Android. Kita dapat menggunakan bantuan library yaitu **AndroidPdfViewer**.

Referensi dan bacaan lebih lanjut~

1. [How to open a PDF file in Android programmatically?](#)
2. [GitHub - barteksc/AndroidPdfViewer: Android view for displaying PDFs rendered with PdfiumAndroid](#)
3. [File | Android Developers](#)
4. [Introduction to File Processing](#)
5. [SISTEM BASIS DATA TERDISTRIBUSI | aellyas](#)



Nah, selesai sudah pembahasan kita di **Chapter 6 Topic 4** ini.

Daan, Topik ini juga mengakhiri bahasan **Chapter 6** kita.

Selanjutnya, kita akan bahas Challenge yang akan dikerjakan sebagai penutup Chapter ini~



Terima Kasih!



Chapter ✓

completed