Module - II

Android Resources (R)

* A resource in An) is a file / a value that is bound to an executable appli⁻ in such a way that you can change them.
* eg → Strings, colors, bitmaps & layouts.

I String Resources :-

* Provides txt strings for appli⁻ with optional txt styling & formatting.
* 3 types —

a) String – XML (R) that provides a single str.

b) String Array – XML (R) that provides an array of str.

c) Quantity Strings (plurals) – carries diff str for pluralization.

a) String :
- It is a single String that can be refrnced from the appli⁻ / from othr (R) file.
- located in res/values/strings.xml.
- getString() used to retrieve strings.
- Syn → <?xml version="1.0" encoding="utf-8"?>
  <resources>
  <string name="str_name"> txt str </string>

</resources>.

* eg →
```
xml {
<?xml ~~
<resources>
    <string name="hello">hello world
    </string>
</resources>.
```

to retrieve a str —

```
java { String name = res.getString(R.string.hello);
```
(res. → resources; R → string → str name)

```
display content {
<TextView
    android:layout_height="wrap_content"
    " width = " " "
    " text="@string/hello" />.
```

b) String array:

- used to describe the array of strings that can be refrenced from the appli~
- getStringArray() used to retrieve strings.

eg →
```
<?xml ~
<resources>
    <string-array name="test_array">
        <item> one </item>
        <item> Two </item>
        !
    </string-array>
</resources>
```

<item> : to define the individual items within the array. Each <item> represents a specific value/string in the arr.

to retrieve -

```
//get access to res obj
Resources res = getResources();
String strings[] = res.getStringArray(R.
        array.test_array);
```

c) plurals:

- Also → quantity strings. i.e. is a set of str.
- These str are various ways of expressing a numerical quantity
- eg → how many eggs are in a nest.
  - → There is 1 egg
  - → There are 2 eggs
  - → " are 100 eggs
- getQuantityString() used to retrieve str.
- eg →
```
<resources>
    <plurals name="EggNest">
        <item quantity="one">There is 1 egg
        </item>
        <item quantity="other">There are
                 %d eggs </item>
    </plurals>
</resources>
```

to retrieve —

```
int count = getEggNest();
Resources res = getResources();
String eggs = res.getQuantityString(R.
    plurals.EggNest, count, count);
```
[quantity "one" or "other" ...] [%d ... value]

s/x → <plurals name="pluralname">
<item quantity=["zero"|"one"|"two"|
"few"|"many"|"other"]>txt str
</item> </plurals>.

o/p (If we give 3 it goes to other)
There are 3 eggs.

* Formatting & Styling :

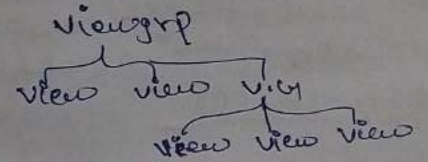
① Escaping apostrophes & quotes :
(apostrophes not work, bt we can format it)
<string name="ex1">"this'll work"</string
([illegible] work [illegible])

② Styling with html :
<?XML ~
<resources>
<string name="welcome"> welcome to
 Android
</string>
</resources>

① Layout Resources :
* defines the archi~ for the UI in an activity of a UI.
* located in res/layout/filename.XML
* In (An), the view for a screen is often loaded from an XML file as a (R).

~~s/x → <?XML ~ <viewGroup>~~

* All the elemnts in a layout are built using a hierarchy of view & viewgroup.
· view : defined as the UI which is used to create interactive UI components like txtview, imagview, EditText, RadioButton..
· viewGroup : Act as a base c/s for layouts & layouts paramtr that hold othr views & to define the layout (pro) & it is root elemnt



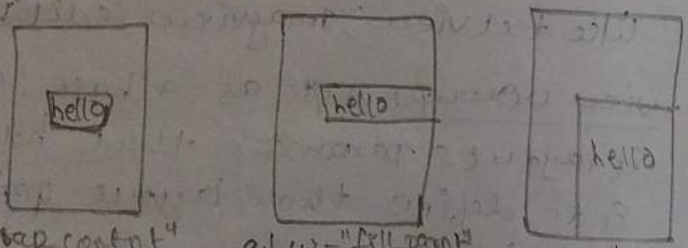
* Types — Linear (L), Relative (L), frame (L), table (L), Constraint (L).

* eg → <?XML ~~
<LinearLayout xmlns:android="http://
schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fill_parent"
" : " _height ="fill_parent">

<Text view
android:id="@+id/text"
" : layout_width ="wrap_content"
" : " _height = "wrap_content"
" : text = "@string/hello" />.

```
<Button
   android:id="@+id/Button
   "   :layout_width="fill_parent"
   "      " _height="wrap_content"
   "   :text="hello" />
</LinearLayout>
```

btn ⟶



a.l_w="wrap content"   a.l_w="fill parent"   a.l_w="f.p"
a.l_h=" – – "   a.l.h="w.c"   a.l_h="f.p"

Fill_parent = match_parent

III color Resource:

* A color value defined in XML.
* color is specified with an RGB value & alpha channel
* value always begins with # & followed by Alpha - Red - Green - Blue info in 1 of the follow formats -
  - #RGB   - #RRGGBB
  - #ARGB   - #AARRGGBB.
- file is located in res/values/colors.xml.
- Syx ⟶

```
<?xml ..
  <resources>
    <color name="color_name"> hex_color</color>
  </resources>
```

- eg ⟶

```
<?xml> ...
<resources>
   <color name="opaque_red"> #f00 </color>
   <color name="translucent_red">#80ff0000
                                   </color>.
</resources>.
```

to retreives the color —

```
Resource res = getResources();
int color = res.getColor(R.color.opaque_red);
```

to display content —

```
<TextView
   android:layout_width=" f.p "
   "  :   " _height = "w.c"
   "  :textColor="@color/translucent_red"
   android:text="Hello" />.
```

([illegible] Hello [illegible] Str-[illegible] translucent color apply [illegible])

IV Dimension Resources:

* used to style & localize (An) UIs without changing the source code.
* dimen is specified with a no. followed by a unit of measure —
  - px - pixels
  - in - inches
  - mm - millimtrs
  - pt - points
  - dp - density independent pixels based on 160 dpi (pixel density per inch) screen

SP - Scale independent pixels
(dimen that allow for user sizing)

* $tx → <?xml —
<resources>
<dimen name="dimn_name>dimn
</dimen>
</resources>.

* eg → <xml —
<resources>
<dimen name="txtview_height">25dp
</dimen>
<dimen name="textview_width>150dp
</dimen>
<dimen name="font_size">16sp
</dimen>
</resources>
(font_size always be in SP unit)

to retrieve —
Resources res = getResources();
float fontSize = res.getDimension(R.
dimen.font_size);

layout xml —
<TextView
android:layout_height="@dimen/
txtview_height"
" :l_w="@dimen/textview_width"
" : text size="@dimen/font_size"/>

V Image Resources :

* A drawable (R) is a general concept for a graphic that can be drawn to screen.
* can be retrieved using getDrawble()
* img files are placed in res/drawable directory.
* Sprted img types — .gif, .jpg and .png.
* eg → <imageview
Android:layout_h="w.c"
" :layout_w="w.c"
" :src="@drawble/myimage"/>

to retrieve —
Resources res = getResources();
Drawble drawble = res.getDrawble(R.
drawable.myimage); →

✗ (R) refrnce Syntx :

* Regardless of the type of (R) all (R) are identified by their IDEs in java code.
* The $tx that is used to allocate an ID to a (R) in XML file → R.r.str.
* This $tx is not limited to allocating just aids: it is a way to identify any (R) like str, layout file / an img.
* has follow formats —
@[package:]type/name — name gvn to the (R).

If you don't specify any package, it takes R.java pckge.

* you can use any java pkge name in place of the pkge.
* resource-type namespaces available in R.java - R.id, R.drawable, R.layout, R.String, R.attr, R.plural, R.array
* eg → <TextView
  android:id="@+id/text1" ... />
  ↓ type ↓ name
  here no pkg, so it take R.java (local pkg)
  + → indicate txt1 name id exists or not, if yes go on, otherwise it creates text1 as id.

VI Defining own (R) IDs:

* General pattern for allocating an ID is either to create a new 1 / to use the 1 created by the (An) pkg.
* It is possible to create IDs beforehand & use them later in yr own pkg
* The line <TextView android:id="@+id/text" indicates that an ID named txt is used if it already exists (+ ...)
  If the ID doesn't exist, a new one is created.

Predefining an ID –

```
<resources>
    <item type="id" name="text" />
</resources>
```

* Reusing a predefined ID –

```
<TextView android:id="@id/text">
    ... </TextView>
```

(Here + not req, bcz it is already been created).

* Bitmap file: A bitmap graphic file (.png, .jpg or .gif)
* Layer list: A drawable that manages an array of other drawables. These are drawn in array order, so the elemnt with the largest indx is being drawn on top.
* State list: An XML file that refrences diff bitmap graphics for diffrnt states eg- use a diffrnt img when an btn is pressed
* Level list: An XML file that defines a drawable that manages a new of alternate drawable, each assigned a max numerical value.

(Types of drawables) (Png (R) continuous)

⇒ Content Providers =

* Are (An)'s central mechanism that enables to access data of other appli – mostly info stored in DB.
* Spt the 4 basic opr → CRUD opr.
* Provide data abstraction & encapsulation

& provide mechanism for defining data security.
* Supplies data from 1 appln to others on rqst.
* Such rqsts are handled by the methods of ContentResolver cls.
* can use diffrnt ways to store its data - & the data can be stored in a DB, in files / ovr a n/w.

* Need of C.P:

- DB in (An) is private to the appln that creates it.
- no common storage area in (An) that every appln can access.
- for diffrat appln to use a DB (An) needs an interface that allows inter-appln & intr-(p) data exchange.

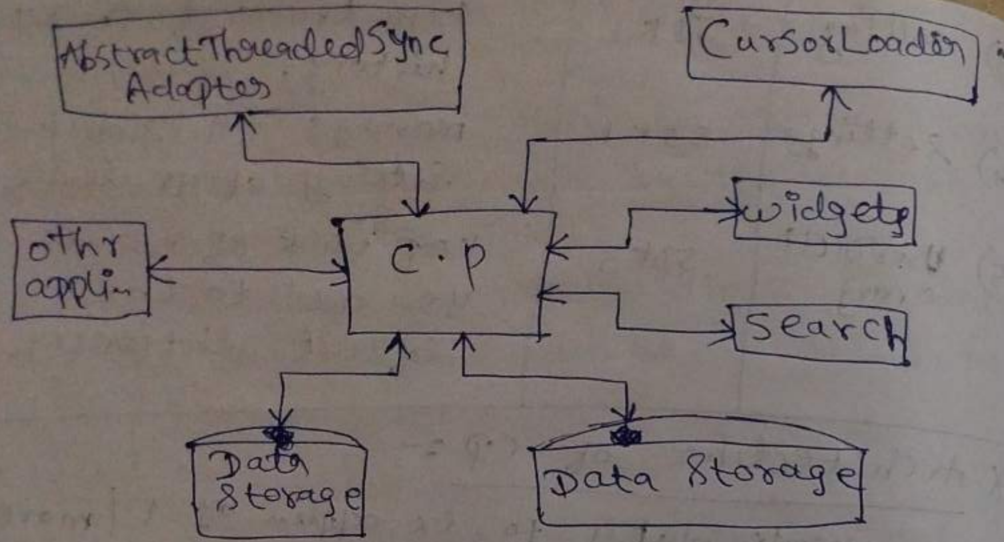
(An) Built-in providers:

Non of C.P are part of (An)'s API.
All these standard providrs are defined in the pkg android.provider.

provider	Since	usage.
Browsr	SDK 1	manags yr web searchs, history, bookmarks.
calendar Contract	SDK 14	manages the calendr on user's device.
3) calllog	SDK 1	keep track of yr call history.
4) Settings	SDK 1	manags all global settings of yr device.
5) UserDictionary	SDK 3	keep track of words you add to the dfault dictionary.

* Architecture of CP :-

- C.P prsnts data to ex appln as 1 / more tables similar to tables in relational DB.
- A C.P coordinates access to the data storage layer in yr appln for a no. of diffrnt APIS & components -
- sharing access to yr appln data with othr appln
- finding data to a widget.
- Loading data in yr UI using a 'CursorLoader'
- Synchronizing appln data with yr servr using an implemn of 'AbstractThreadedSync Adaptor'.
- follow figure shows the relationship b/w C.P and othr Components -

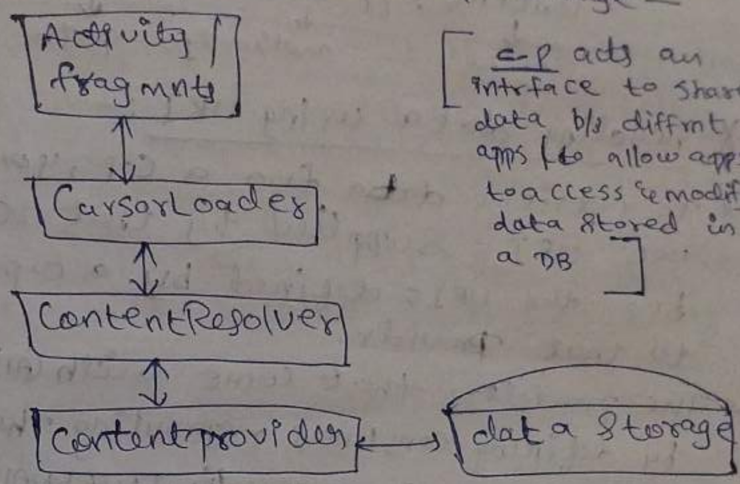


* To access data in C.P, use the <u>ContentResolver</u> obj to communicate with the provider as a client.
* provider obj receives data rqsts from clients, performs the reqsted action, & returns the rslts
* ContntResolver methods provides the basic "CRUD" ()s.
* A common pattern for accessing a Content P from UI uses a '<u>CursorLoader</u>' to run an asynchronous query in the background. The activity in UI call a CursorLoader to the query, which in turn gets the C.P using the ContentResolver.

~~This interaction pattern involves a no. of~~

This pattern involves the intera^n of a no. of diffrnt objs.

* Intera^n b/w C.P, othr cles & Storage —



[<u>C.P</u> acts an interface to share data b/w diffrnt apps / to allow apps to access & modify data stored in a DB]

→ <u>Structure of (An) content URI :-</u>

* whnever you want to access data frm a C.P you hve to specify a URI.
  [whstpp-ന് ഒരു data വേണമെങ്കിൽ അത് c.p ആണ് തരുന്നത്. c.p-ൽ നിന്ന് content എടുക്കണമെങ്കിൽ URI വേണം)
* To query a C.P, you specify the query str in the form of a URI.
* Has follow format —

  <scheme> : //<authority> /<path> / <id>

- scheme → constant value: content.
- Authority → symbolic name of provider for browsr
- path → helps distinguish the required data frm complete DB.
- id → specific record reqsted, should be numric.

* eg → looking for contact no. 5 in contacts then URI is —

Content : // contacts / people / 5 .
(Content ↓ scheme; contacts ↓ authority; people ↓ path; 5 ↓ id)

* Reading data using URI :-

* To retrieve data frm a C.P, you need to use URIs supplied by that C.P. bcz the URIs defined by a C.P are unique to that provider.
* The providers that come with (An) do this by defining constnts representing these URI strings
* eg → ① MediaStore.Images.Media.INTERNAL_CONTENT_URI (↓ for gallery imgs)
  (helper classes in (An) SDK) ② ContactsContract.Contacts.CONTENT_URI

  equivalent textual URI strings —
  ① content://media/internal/images/
  ② content://com.android.contacts/contacts/ (com.android.contacts ↓ authority)

* code to retrieve a single row of people frm the contacts provider —

(base obj created)
```
Uri peopleBaseUri = ContactsContract.Contacts.
                     CONTENT_URI;
(datatype)
Uri mypersonUri = Uri.withAppendedPath(
                     peopleBaseUri, "23");
```
(23rd contact-ന്റെ [illegible] data mypersonUri-ലേക്ക് [illegible])

then we should give access permission for the provider using <user-permission> element.

→ Using (An) cursor :-

* A cursor represents the rslt of a query & basically points to 1 row of query rslt.
* It is an interface which represents a 2D table of any DB.
* when retrieving data using SELECT (8), DB will 1st create CURSOR obj & return its rfrnce.
  The pointer of this returned rfrnce is pointing to the 0^{th} loc.
* To retrieve data frm cursor, we have to move to the 1st rec using moveToFirst()
* methods using cursor :

1) moveToFirst() takes the cursor pointer to the 1st loc & data in the 1st rec can be accessed.
2) moveToNext() - to get the next data frm cursr
3) moveToLast() - last data in the rslt.
4) moveToPrevious().
5) isAfterLast() - checks whthr the end of the queryrslt has been reached.
6) isBeforeFirst() - cursor default position-ൽ [illegible]
7) getCount() → find no. of rows in a cursor.

⇒ working with where clause :

* cp offers 2 ways of passing a where clause to retrieve data –

a) through URI

b) through the combin" of a Str clause & Set of replaceable Str-arr argmnt. (explicit where(c))

eg → (a) Through the URI –

to retrieve a note whose ID is 23 frm the google notes DB.

```
Activity someActivity;
// initialize someActivity
String noteUri = "content://com.google.provider.Notepad/notes/23";
// cursor obj
Cursor managedCursor = someActivity.managedQuery(
        noteUri, projection, null, null);
```

(projection → which colmns to return; where clause → where (c) ; order-by clause → order-by clause.)

(Here where (c) omits, bcz id is 23 is gvn)

(From version-1 managedQuery() is used)

(Cursor obj is used to store rows, that's why it defined as cursor);

(b) using explicit where (c):

- method by which (An) find a list of explicit colmns & their corresponding values as where(c)
- Structure –

```
public final Cursor managedQuery(Uri uri,
        String[] projection,
        "    Selection,   // where(c) is gvn here
        "    SelectionArgs,  // selection arg is gvn here
        "    SortOrder);  // asc/desc order.
```

- Arg named Selection represnts a filter declaring which rows to return, passing null will return all rows for gvn URI.
- we can include '?' in Selection paramtr along with SelectionArgs is to prvnt SQL injection attacks & ensure that the query is safe & secure.

* eg → query for a note whose ID is 23 using either of these 2 methods –

```
// URI method
managedQuery("content://com.google.provider.Notepad/notes/23", null, null,
        null, null);
```

(P = projection, S = sort; null → sArg, sort)

(more common)

(or)

```
// explicit where (c) –
managedQuery("co ... /notes",
        null, "_id=?", new String[]{23}, null)
```

⇒ Inserting records to c.p =

* (An) uses a cls → android.content.ContentValues to hold the values for a single record that is to be inserted.

ContentValues is a dictionary of key/value pairs like colmn names & their values.

* Records are inserted by 1st populating a rec into ContentValues & then asking android.content.ContentResolver to insert

that rec using a URI.

* eg →
  ```
  ContentValues values = new ContentValues();
  values.put ("title", "New Note");
  values.put ("note", "this is a note");
  //values obj is now ready to be inserted.

  ContentResolver contntResolvr = activity.
                getContntResolver();
  {URI for notepad → Notepad.Notes.CONTENT_URI}
  // take this URI & contnt values & make
     a call to insert the row
  Uri uri = ContentResolvr.insert (Notepad.
                Notes.CONTENT_URI, values);
  //This call returns a URI pointing to
     newly inserted record.
  ```

⇒ Updates & Deletes =

* performing an update is similar to performing an insert, in which changed colmn values are passed through a ContentValues obj.

* s/x → for update —

```
int numberOfRowsUpdated = activity.getContent
     Resolver().update(Uri uri, ContentValues
     values, String whereClause, String[]
     SelectionArgs)
```

[illegible] update ← (whereClause)

→ [illegible] update [illegible] (SelectionArgs)

* s/x for dlt —

```
int numberofRowsDeleted = activity.getContent
     Resolver().delete (Uri uri, String
     whereClause, String[] SelectionArgs)
```

* Implemnting contnts :

To write a C.P, you hve to extend android.content.Contentprovidr & implemnt follow key methods —

1) onCreate() → method is called whn the providr is started.
2) query() — receivs a rqst frm client. Result.
3) Insert() — ins a new rec into C.P
4) Delete() — dlts an existing rec frm C.P
5) Update() — updates an " "
6) getType() — returns the MME type of the data at gvn URI.

* C.P implemn steps —

a) plan yr DB, URIs, colmn names & create a meta data cls.
b) Extnd the abstract cls C.P
c) Implemnt these methods : query, ins, update, dlt & getType
d) Registr the providr in the manifest file.

⇒ Intents = (I)

* It is an action that you can tell (An) to perform. The action (An) invokes (starts) & (data) depnds on wht is registered for that action.
* (I) facilite commun b/w components in several ways.
* 3 fundamntal use cases —

a) Starting an activity: An activity reprsnts single screen in an app. To start a new instnce of an activity, pass an (I) to startActivity.
To receive a rslt frm an activity why finishes, call startActivityForResult()

b) Starting a service: Service is a component that performs opr in the background without a user interface.
To start a service to perform 1 1-time opr, pass an intnt to startService().

c) Delivring a broadcast: broadcast is a msg that any app can receive.
To delivr a broadcast to othr apps, pass an (I) to SendBroadcast() / sendOrderedBroadcast()

I Available (I) is (An):

* A browsr applin to opn a browsr window
* A applin to call a telephn nom
* A mapping applin to show the map of the world at a gvn latitude & longitude coords.

* Structure of an (I) —

```
public static void invokeWeb(Activity activity){
    Intent intent = new Intent(Intent.
                    ACTION_WEB_SEARCH);
```

intnt obj →

ACTION_WEB_SEARCH → opn user's web browsr with the search rslts for specified query

```
To set data →  intent.setData(Uri.parse("http://www.google.com"));
                                        (want to search)
To start (I) →  activity.startActivity(intent);
}
```

II (I) Structure =

a) primary pieces of info in an (I) —

1) ACTION: general action to be performed —
ACTION_VIEW, ACTION_EDIT, ACTION_MAIN

2) DATA: data to operate on Such as person rec in the contact DB expressed as Uri.

eg of action/data pairs —

* ACTION_VIEW — content://contacts/people/1 → display info about person where ID=1.
* ACTION_DIAL — content://contacts/people/1 → display phn dialler with person filled in.
* ACTION_DIAL tel:123 → display phn dialler with gvn nom filled in. (i.e) 123.
* ACTION_EDIT — cont.... /people/1 → EDIT info about person whose ID = 1.

(Primary pieces are, Sec are optional).

Secondary attributes of (I):-

- CATEGORY: gives additional info about kind of component that should handle the (I).
  addCategory() places a category in an (I) obj,
  removeCategory() dlts a category previously added,
  getCategories() gets the set of all categories currently in the obj.
  eg → CATEGORY_LAUNCHER, CATEGORY_BROWSABLE,
  " _ALTERNATIVE.

- EXTRAS: Bundle of any additional info.
  used to provide extended info to the component.
  Can be set & read using putExtras() & getExtras()

- FLAGS: optional part of (I) obj that instruct (An) how to launch an activity & how to treat it after it is launched.
  eg → FLAG_ACTIVITY_CLEAR_TASK.,
  " _NEW_TASK
  setFlags() → used to set flags on an (I).

III Types of (I):- (2 types)

i) Implicit (I):

Do not specify the (An) components which should be called. only specifies action to be performed. A Uri can be used with implicit (I) to specify the datatype.

eg →
```
Intent intent = new
  Intent(ACTION_VIEW, Uri.
    parse("http://www.google.com"));
```
(google.com നെ uri ആക്കി convert ചെയ്യുന്നു).

2) Explicit (I):

use Explicit (I) when you know exactly whn activity can handle the reqst. used in appli- wherein 1 activity can switch to othr activity. Also used for appli- internal msgs.

eg →
```
Intent i = new Intent(
  FirstActivity.this,
  SecondActivity.class)
startActivity(i);
```
(1st activity-ൽ നിന്ന് 2nd activity-ലേക്ക് പോകുന്നു)

IV Rules for resolving (I) to their components:-

* (An) uses multiple strategies to match (I) to their targt activities based on (I) filters.
* At the top of the hierarchy is the component name attached to an (I). If this is set, the (I) → explicit (I).

[(I) enables commu. & coordin^n b/w diffrnt components within an app / b/w diffrnt apps]

* For an explicit (I), only the component name matters, every othr aspect of the (I) is ignored.
* whn a component name is not prsnt on an (I), the (I) is said to be an implicit (I).
* whn the system receives an implicit (I) to start an activity, it searches for the bst activity for (I) based on 3 aspects -
  - Action
  - data
  - category

  } 3 tests.

* Intent filters will specify type of (I) it will accept. They are declared in manifest file.

① Action Test : If an (I) has an action in it, the (I) filter must have that action as part of its action list.

To specify accepted (I) actions, an (I) filter can declare 0/more <action> elemnts.

eg → <intent-filter>
<action android:name="android.intent.action.EDIT" />
<action " = "android.intent.action.VIEW" />
;
</intent-filter>.

- If the filter doesn't list any action, then all (I) fail the test.
- To pass this filter, the action specified in the (I) must match 1 of the actions listed in the filter.

② Category Test:

To specify accepted (I) categories, an (I) filter can declare 0/more <category> elemnts.

eg → <intent-filter>
<category android:name="android.intent.category.DEFAULT" />.
<category " = " .

category.BROWSABLE" />
;
</intent-filter>

- to pass the category test, every category in the intent must match a category in the filter.
- Reverse is not necessary.
- An (I) with no categories always passes the test

③ Data test :

* To specify accepted (I) data, filter can declare 0/more <data> elemnts.

eg → <intent-filter>
<data android:mimeType="video/mpeg" android:scheme="http" />
<data " ="audio/mpeg" />.
;
</intent-filter>

mpeg → family of video & audio compression standards

mimeType → specify the format of data like video, audio, img

schemes → indicate the protocol need to access data like http, https, file ~

* each <data> elemnt contains a URI str~ & a datatype (mimeType)
* each part of URI is a separate attribute~

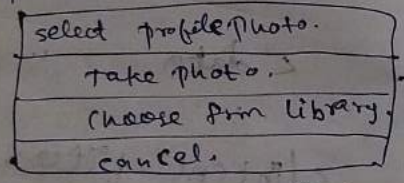
<scheme>://<host>:<port>/<path>

eg → content://com.example.project:200/folder/subfolder/~

* Data tst compares both URI & MIME type in the (I) to that of the filtr.
* If it doesn't specify both URI & MIME intent passes the tst only if filtr doesn't specify any URLs / MIME types.
* An (I) that contains URI bt no MIME passes the test only if its URI matches the filter's URI & filtr doesn't specify any MIME type.
(and vice versa.)

V. ACTION_PICK :

eg → whsapp → profile photo change option when dialog box. →



* used to start an activity that displays a list of items.
- The activity thus should allow a user to pick - item from that list.
* once the user picks the item, the activity should return the URI of the picked item to the caller.
* This allows reuse of the UI's functionality to select items of a certain type.
- hlps to pick an img item from a data source like camera / gallery.
* The activity launched depnds on the data being picked.

eg → passing content://contacts/people will invoke the native contacts list
* All you need is the URI of the data you need & required permissions to access that data.

- StartActivityForResult —

public void startActivityForResult(Intent intent, int requestCode)

(requestcode → hlps you to identify from which (I) you came back)

eg → Imagine yr Activity A (main activity) could call Activity B (camera activity), Activity C (Audio recording), Activity D (select a Contact).

* eg →

① private static final int REQUEST_PICK_IMAGE = 1; (requestcode=1)
Intent pickImageIntent = new Intent(Intent.
ACTION_PICK, android.provider.MediaStore.Images.
Media.EXTERNAL_CONTENT_URI);
(gallery uri)
startActivityForResult(pickImageIntent, REQUEST_
PICK_IMAGE);
(now gallry opns)

② private static final int PICK_CONTACT_SCRACTIVITY=2;
Uri uri = Uri.parse("content://contacts/people");
Intent intent = new Intent(Intent.ACTION_PICK, uri)
startActivityForResult(intent, PICK_CONTACT_SCRACTIVITY);

ACTION_GET_CONTENT:

* Allow the user to select a particular kind of data & return it.
* This is diffrnt than ACTION_PICK in that here we jst say wht kind of data is desired, not a URI of existing data frm which the user can pick.
* eg → taking a picture, recording sound.
* 2 ways to use this action —

1) For a specific kind of data, such as a person contact, set the MIME type to the kind of data you want

~~Intent mIntent = new Intent(Intent.ACTION_GET_CONTENT)~~
~~mIntent.setType(Contacts.Contract.Com~~

2) It is possible to wrap the GET_CONTENT intent with a chooser, which will give the propr interface for the user to pick hw to find yr data.

Pending Intent :

(An) allows a component to store an (I) for future use as a loc frm which it can be invoked again.

eg → in an alarm manager, you want to start a service whn the alarm goes off.

(An) PendingIntent is an obj that wraps up an (I) obj & it specifies an action to be taken place in future.

* PendingIntent pass a future Intnt to anthr applin & allow that applin to execute that (I)
* generally used in cases where an AlarmManager needs to be executed.
* uses methods to handle the diffrnt types of (I) —

1) pendingIntent.getActivity(): retrieve a P.I to start an Activity.
2) P.I.getBroadcast()
3) P.I.getService()

* eg → Intent intent = new Intent(this, SomeActivity.class);

PendingIntent pendingI = PendingIntent.getActivity(this, 1, intent, pendingI.FLAG_UPDATE_CURRENT);
(reqcode.)

PendingIntent.send();