Introduction à la plateforme Android

Cours 2 - http://www.tutos-android.com/MTI/2018/

Passage de données entre vues

Passage de données entre des vues - Bundles

- S’effectue à l’aide des extras lors de la création d’un Intent.

```
public static final String USER_NAME_EXTRA = "USER_NAME";

final Intent intent = new Intent(LoginActivity.this, SignupActivity.class);
intent.putExtra(USER_NAME_EXTRA, "nazim");
startActivity(intent);
```

Passage de données entre des vues - Bundles

- La récupération des datas dans l'activité cible s'effectue à l'aide des Intents.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_signup);

    final Intent intent = getIntent();
    if (intent != null) {
        if (intent.hasExtra(USER_NAME_EXTRA)) {
            userName = intent.getStringExtra(USER_NAME_EXTRA);
        }
    }
}
```

Passage de données entre des vues - Bundles

- Peux contenir que des types basiques (boolean, char, byte, double, float, int, long, String, …).

Passage de données entre des vues - Parcelable

- Si on souhaite passer des données plus complexes / personnalisées, par exemple un utilisateur.

```
public class User {

    private final String login;
    private final String password;

    public User(String login, String password) {
        this.login = login;
        this.password = password;
    }
}
```

Passage de données entre des vues - Parcelable

- [ ] Implémenter la classe Parcelable.
- [ ] Comment convertir la classe User en Parcelable.

```
@Override
public int describeContents() {
    return 0;
}

@Override
public void writeToParcel(Parcel dest, int flags) {
    dest.writeString(login);
    dest.writeString(password);
}
```

Passage de données entre des vues - Parcelable

- [ ] Créer un CREATOR pour convertir la classe Parcelable en classe User.

```
public static final Parcelable.Creator<User> CREATOR = new Parcelable.Creator<User>()
{
    @Override
    public User createFromParcel(Parcel source)
    {
        return new User(source);
    }

    @Override
    public User[] newArray(int size)
    {
        return new User[size];
    }
};

private User(Parcel source) {
    login = source.readString();
    password = source.readString();
}
```

Passage de données entre des vues - Parcelable

- [ ] **Passage des données**

```
User user = new User("Nazim", "android");

final Intent intent = new Intent(LoginActivity.this,
SignupActivity.class);
intent.putExtra(USER_EXTRA, user);
startActivity(intent);
```

Passage de données entre des vues - Parcelable

- Récupération des données

```
final Intent intent = getIntent();
if (intent != null) {
    if (intent.hasExtra(USER_EXTRA)) {
        userName = intent.getParcelableExtra(USER_NAME_EXTRA);
    }
}
```

Appeler d’autres applications

Appeler une application connue

- Lancer une activité du système Android (Dialer, Calendar …)
- Plus d'exemples ici : https://developer.android.com/training/basics/intents/sending.html

```
Uri number = Uri.parse("tel:5551234");
Intent callIntent = new Intent(Intent.ACTION_DIAL, number);
startActivity(callIntent);

Uri webpage = Uri.parse("http://www.android.com");
Intent webIntent = new Intent(Intent.ACTION_VIEW, webpage);
startActivity(webIntent);
```

Appeler une application connue

- Lancer une activité dont vous connaissez le nom de package.

```
Intent launchIntent =
getPackageManager().getLaunchIntentForPackage("com.package.address");
if (launchIntent != null) {
    startActivity(launchIntent);
}
```

Appeler une action

```
File file = new File(« myfile.pdf");

Intent target = new Intent(Intent.ACTION_VIEW);
target.setDataAndType(Uri.fromFile(file), "application/pdf");

Intent intent = Intent.createChooser(target, "Open File");

PackageManager packageManager = getPackageManager();
List activities = packageManager.queryIntentActivities(intent,
PackageManager.MATCH_DEFAULT_ONLY);
boolean isIntentSafe = activities.size() > 0;

if (isIntentSafe) {
    startActivity(intent);
}
```

Débogage et Logs

Les logs - Utilisation

- Api permettant d’afficher des logs selon différents niveaux d’importance.
- Log.e : Erreur
- Log.w : Warning
- Log.d : Debug
- Log.i : Informations
- Log.v : Verbose

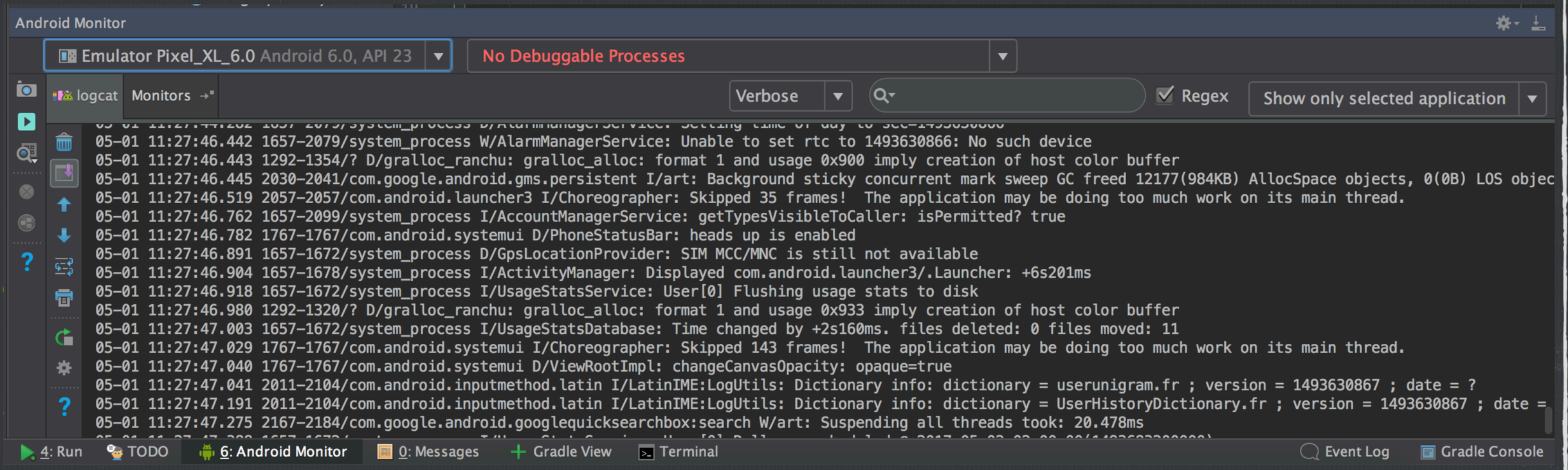
Les logs - Utilisation

```
public static final String TAG = "SignupActivity";

......

for (int i = 0; i < 100; ++i) {
    Log.v(TAG, "i = " + i);
    ......
}
```

Les logs - Utilisation

`static int`	`e(String tag, String msg)` Send an `ERROR` log message.
`static int`	`e(String tag, String msg, Throwable tr)` Send a `ERROR` log message and log the exception.

LogCat


Android Monitor
Emulator Pixel_XL_6.0 Android 6.0, API 23
No Debuggable Processes
logcat
Monitors
Verbose
Regex
Show only selected application
05-01 11:27:46.442 1657-2079/system_process W/AlarmManagerService: Unable to set rtc to 1493630866: No such device
05-01 11:27:46.443 1292-1354/? D/gralloc_ranchu: gralloc_alloc: format 1 and usage 0x900 imply creation of host color buffer
05-01 11:27:46.445 2030-2041/com.google.android.gms.persistent I/art: Background sticky concurrent mark sweep GC freed 12177(984KB) AllocSpace objects, 0(0B) LOS objec
05-01 11:27:46.519 2057-2057/com.android.launcher3 I/Choreographer: Skipped 35 frames! The application may be doing too much work on its main thread.
05-01 11:27:46.762 1657-2099/system_process I/AccountManagerService: getTypesVisibleToCaller: isPermitted? true
05-01 11:27:46.782 1767-1767/com.android.systemui D/PhoneStatusBar: heads up is enabled
05-01 11:27:46.891 1657-1672/system_process D/GpsLocationProvider: SIM MCC/MNC is still not available
05-01 11:27:46.904 1657-1678/system_process I/ActivityManager: Displayed com.android.launcher3/.Launcher: +6s201ms
05-01 11:27:46.918 1657-1672/system_process I/UsageStatsService: User[0] Flushing usage stats to disk
05-01 11:27:46.980 1292-1320/? D/gralloc_ranchu: gralloc_alloc: format 1 and usage 0x933 imply creation of host color buffer
05-01 11:27:47.003 1657-1672/system_process I/UsageStatsDatabase: Time changed by +2s160ms. files deleted: 0 files moved: 11
05-01 11:27:47.029 1767-1767/com.android.systemui I/Choreographer: Skipped 143 frames! The application may be doing too much work on its main thread.
05-01 11:27:47.040 1767-1767/com.android.systemui D/ViewRootImpl: changeCanvasOpacity: opaque=true
05-01 11:27:47.041 2011-2104/com.android.inputmethod.latin I/LatinIME:LogUtils: Dictionary info: dictionary = userunigram.fr ; version = 1493630867 ; date = ?
05-01 11:27:47.191 2011-2104/com.android.inputmethod.latin I/LatinIME:LogUtils: Dictionary info: dictionary = UserHistoryDictionary.fr ; version = 1493630867 ; date =
05-01 11:27:47.275 2167-2184/com.google.android.googlequicksearchbox:search W/art: Suspending all threads took: 20.478ms
4: Run
TODO
6: Android Monitor
0: Messages
Gradle View
Terminal
Event Log
Gradle Console

Débogage - Breakpoint

Démo

Android Support Library

A quoi ça sert ?

- [ ] D’accéder aux nouvelles fonctionnalités / composants sur les anciennes version d’Android.
- [ ] D’accéder à des fonctionnalités / composants qui se comportent pareil quelques soit là version d’Android.
- [ ] Aussi quelques classes Helper et de Tests.

Example de classe incluse dedans

- Plusieurs version de support library (v4, v7 et design).
- Contient par exemple : ActivityCompat, Fragment, RecyclerView, ViewPager, CardView, SearchView, FloatingButton, SnackBar …

Comment inclure dans le projet

```
compile 'com.android.support:appcompat-v7:25.3.1'
compile 'com.android.support:appcompat-v4:25.3.1'
```

Les permissions

Principe

- Chaque application s’exécute isolée dans une sandbox.
- Si elle veux accéder à une ressources extérieure, elle demande une permission.
- Deux types de permissions :
  - Permissions sensibles : Par exemple, lire les contacts utilisateurs.
  - Permissions non sensibles : Accéder à internet.

Demander une permission

- La première étape consiste à déclarer la permission dans le fichier AndroidManifeste.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
          package="com.example.tp">

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


    <application ...>
        ...
    </application>

</manifest>
```

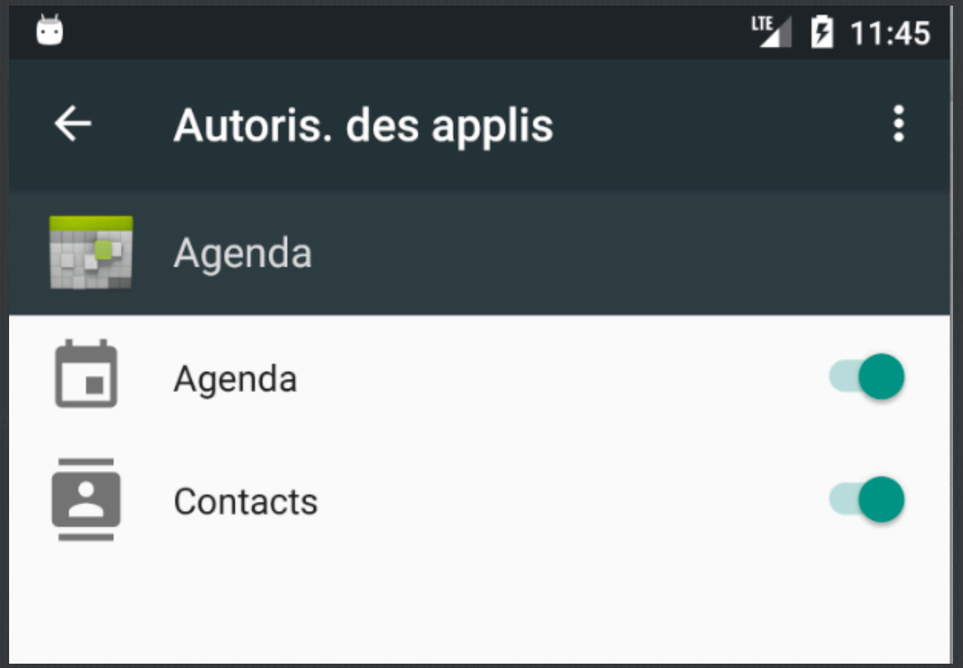
Demander une permission

- Si la permission est non sensible elle saura accordé sans étapes supplémentaire
- Liste des permissions sensibles : https://developer.android.com/guide/topics/permissions/requesting.html#normal-dangerous

Les permissions sensibles

- Depuis Android 6, les permissions sensibles nécessitent une demande au RunTime.
- Vous devez vérifier si vous possédez la permission avant d'exécuter du code nécessitant une permission.
- Un utilisateur peux accorder une permission et pas une autre. Il peux aussi désactiver les permissions dans les paramètres.

Les permissions sensibles



Les permissions sensibles

- [ ] **Etape 1 : Vérifier si une permission est accordée**

```
final int permissionCheck =
ContextCompat.checkSelfPermission(LoginActivity.this,
    Manifest.permission.WRITE_CALENDAR);

if (permissionCheck == PackageManager.PERMISSION_GRANTED) {
    //Permissions accordées
} else {
    //Permissions non accordées
}
```

Les permissions sensibles

- Etape 2 : Faut-il une explication de pourquoi cette permission ?
- Dans certaines circonstance, vous aurez peut être besoin d'expliquer à l'utilisateur pourquoi votre application à besoin de cette permission.
- La méthode « shouldShowRequestPermissionRationale » retourne « true » si l'utilisateur à déjà refusé l'accès à cette permission. Peux-être faut-il afficher une explication ?.

```
boolean shouldNeedExplanation =
ActivityCompat.shouldShowRequestPermissionRationale(this,
Manifest.permission.WRITE_CALENDAR);
```

Les permissions sensibles

- [ ] Etape 3 : Demander une permission

- [ ] Le request code est utile pour récupérer le résultat de la demande.

```
public static final int READ_CONTACT_PERMISSION_REQUEST_CODE = 12;

ActivityCompat.requestPermissions(this,
    new String[]{Manifest.permission.READ_CONTACTS},
READ_CONTACT_PERMISSION_REQUEST_CODE);
```

Les permissions sensibles

- [ ] **Etape 4 : Récupérer le résultat de la demande**

```
@Override
public void onRequestPermissionsResult(int requestCode,
    String permissions[], int[] grantResults) {
    switch (requestCode) {
        case READ_CONTACT_PERMISSION_REQUEST_CODE: {
            if (grantResults.length > 0
                && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                // Permission garantie
            } else {
                // Permission refusée
            }
            break;
        }
    }
}
```

Gestions des ressources

Principe

- Le dossier « res » est le dossier qui contiendra les ressources de l’application.
- Tout son contenu sera accessible depuis votre code (xml ou java).

Principe

- Exemple de dossiers :
  - anim : Contient les animations.
  - color : Contient la declaration des couleurs.
  - drawable : Contient les images.
  - mipmap : Contient les icônes de l’app.
  - layout : Contient les vues.
  - menu : Contient le menu de l’action bar.
  - values : Contient les différentes valeurs (strings, dimens …).

Gestions de images & icônes

- Drawable : Contient toutes les images que vous allez utiliser dans votre app.
- Mipmap : Contient les icônes de l’application.
- Cela dans différentes résolutions (lhdpi, mhdpi, hdpi, xhdpi, xxhdpi ..).

Gestions de chaînes de caractères

- Permet de déclarer toutes les chaînes de caractère utilisées dans l'app.

```
<resources>
    <string name="app_name">TP1</string>
    <string name="adresse_e_mail">Adresse e-mail</string>
    <string name="mot_de_passe">Mot de passe</string>
    <string name="se_connecter">Se connecter</string>
    <string name="inscription">S\'inscrire</string>
</resources>
```

- Utilisation simple « @string/nom » depuis un fichier xml et « R.string.nom » depuis un fichier java.

Gestion du pluriel

```
<resources>
    <plurals name="plural_name">
        <item
          quantity=["zero" | "one" | "two" | "few" | "many" |
"other"]>text_string
        </item>
    </plurals>
</resources>
```

```
Resources res = getResources();
String songsFound =
res.getQuantityString(R.plurals.numberOfSongsAvailable);
```

Echaper les apostrophes

```
<string name="good_example">This\'ll work</string>
<string name="good_example_2">"This'll also work"</string>
<string name="good_example_3">This is a \"good string\".</string>
```

Formater des chaînes de caractère

```
<string name="welcome_messages">Hello, %1$s! You have %2$d new
messages.</string>

Resources res = getResources();
String text = res.getString(R.string.welcome_messages, username,
mailCount);
```

Internationalisation

- Vous pouvez créer des fichiers values pour internationaliser votre application.
- Par exemple, vous voulez que votre application supporte le français, l'anglais, l'espagnol et le japonais. Vous aurez 4 dossier values :
  - values (pour l'anglais), values-fr, values-es et values-jp
- Android choisira la langue adaptée à celle de l'appareil de l'utilisateur.

Gestions des dimensions

- Les dimensions, tailles seront stockées dans le fichier dimens.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <dimen name="font_size">16sp</dimen>
</resources>
```

- Vous pouvez par exemple, avoir un dossier « values-large » qui contiendra les dimensions pour la tablette.

Gestions des couleurs

- Centralise les couleurs utilisées dans votre application.

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#3F51B5</color>
    <color name="colorPrimaryDark">#303F9F</color>
    <color name="colorAccent">#FF4081</color>
</resources>
```

Autres constantes

- Vous pouvez aussi stocker :
  - Des tableaux de chaines de caractères.
  - Des booleans.
  - Des entiers.
  - Des tableaux d’entiers….etc

Fragment

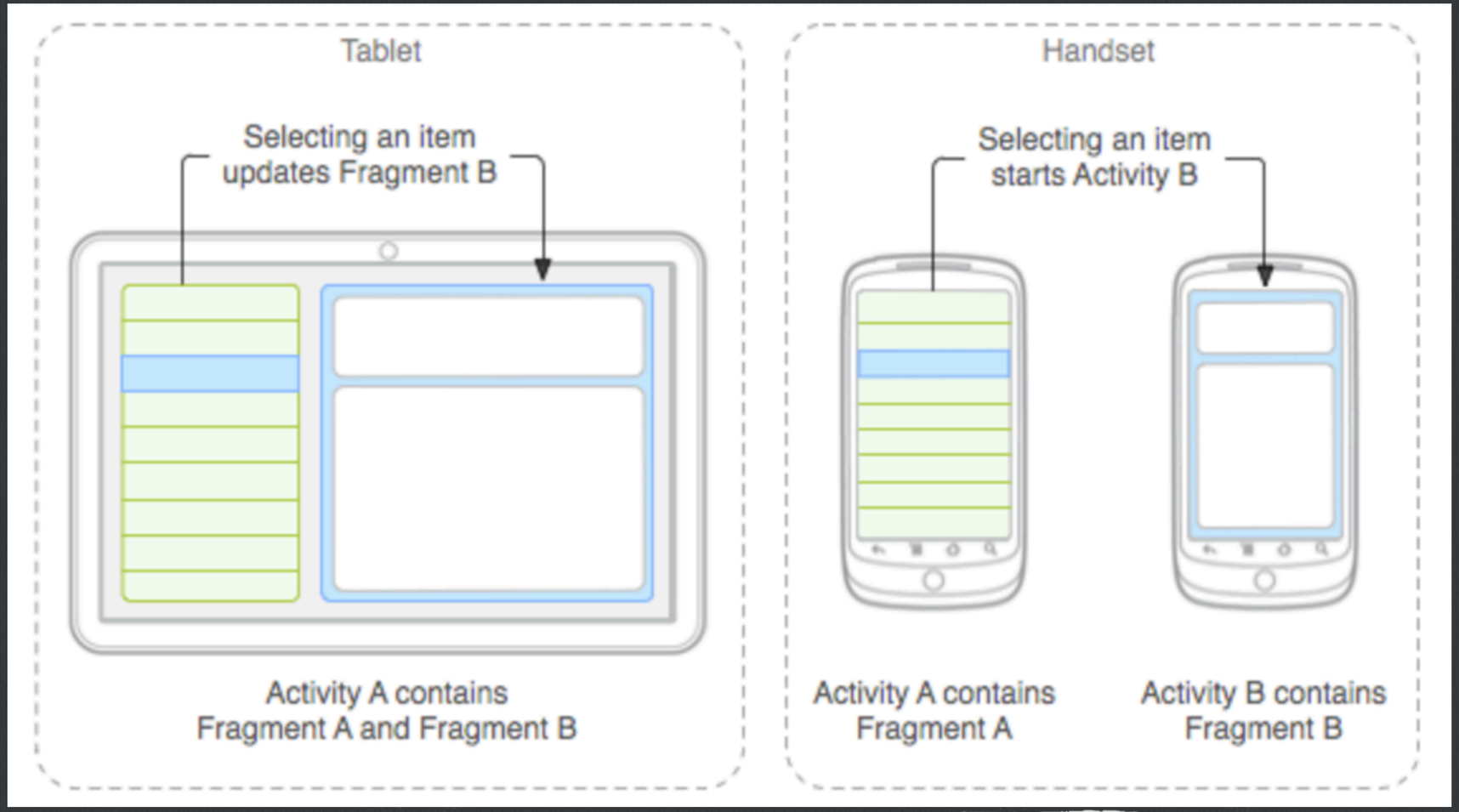
Principe

- ☐ Représente une portion d’une interface (activité).
- ☐ Combiner plusieurs fragments dans une activité.
- ☐ Cycle de vie, événement indépendant de l’activité.
- ☐ Un fragment doit toujours faire partie d’une activité.
- ☐ Quand une activité est en pause, ces fragments sont en pause aussi.

Principe

- Disponible depuis Android 3.0
- Utiliser de préférence la version disponible dans la bibliothèque de support v4.

Principe


Tablet
Selecting an item
updates Fragment B
Activity A contains
Fragment A and Fragment B
Handset
Selecting an item
starts Activity B
Activity A contains
Fragment A
Activity B contains
Fragment B

Principe

- Une classe qui hérite de la classe Fragment.
- Un fragment possède une vue, spécifiée dans la méthode onCreateView.

```
import android.support.v4.app.Fragment;

public class LoginFragment extends Fragment {

    @Nullable
    @Override
    public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup
container,
        @Nullable Bundle savedInstanceState) {
        return inflater.inflate(R.layout.activity_login, container, false);
    }
}
```

Principe

- La vue à charger.
- Le vue « root ».
- Spécifie si la vue chargée, doit être ajouter à la vue Parent.

```
public View inflate(@LayoutRes int resource, @Nullable ViewGroup root,
boolean attachToRoot)
```

Ajouter un fragment à une activité (Méthode 1)

- ☐ Directement dans le layout.
- ☐ Modification des fragments de façon dynamique est impossible.
- ☐ Spécifie la classe du fragment à l’aide de la méthode « android:name ».
- ☐ Identifiant unique par fragment

Ajouter un fragment à une activité (Méthode 1)

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

<fragment
    android:name="com.example.news.ArticleListFragment"
    android:id="@+id/list"
    android:layout_weight="1"
    android:layout_width="0dp"
    android:layout_height="match_parent"
    />
<fragment
    android:name="com.example.news.ArticleReaderFragment"
    android:id="@+id/viewer"
    android:layout_weight="2"
    android:layout_width="0dp"
    android:layout_height="match_parent"
    />

</LinearLayout>
```

Ajouter un fragment à une activité (Méthode 2)

- ☐ Directement dans le fichier Java.
- ☐ Ajout et Modification des fragments de façon dynamique.
- ☐ Une transaction = une opération sur les fragments.
- ☐ Le fragment est ajouté à un layout (par exemple un FragmeLayout).

Ajouter un fragment à une activité (Méthode 2)

```
FragmentManager fragmentManager = getSupportFragmentManager();

FragmentTransaction fragmentTransaction =
    fragmentManager.beginTransaction();
LoginFragment fragment = new LoginFragment();

fragmentTransaction.add(R.id.fragment_container, fragment);
fragmentTransaction.commit();
```

Ajouter un fragment à une activité (Méthode 2)

- [ ] getSupportFragmentManager : Récupére un Manager permettant d'effectuer des modifications sur les fragments.
- [ ] beginTransaction : Toujours commencer par une transaction.
- [ ] Ajouter / Supprimer / Remplacer un fragment.
- [ ] Toujours finir par la méthode commit pour valider la transaction.

Liste & RecyclerView

Les listes - Principe

- Ensemble d’éléments scrollables.
- Gestion des données : adapter.
- Création d’une liste nécessite :
  - Layout global (qui contient la liste).
  - Layout représentant chaque ligne d’une liste.
  - Un adapter pour la gestion des données.

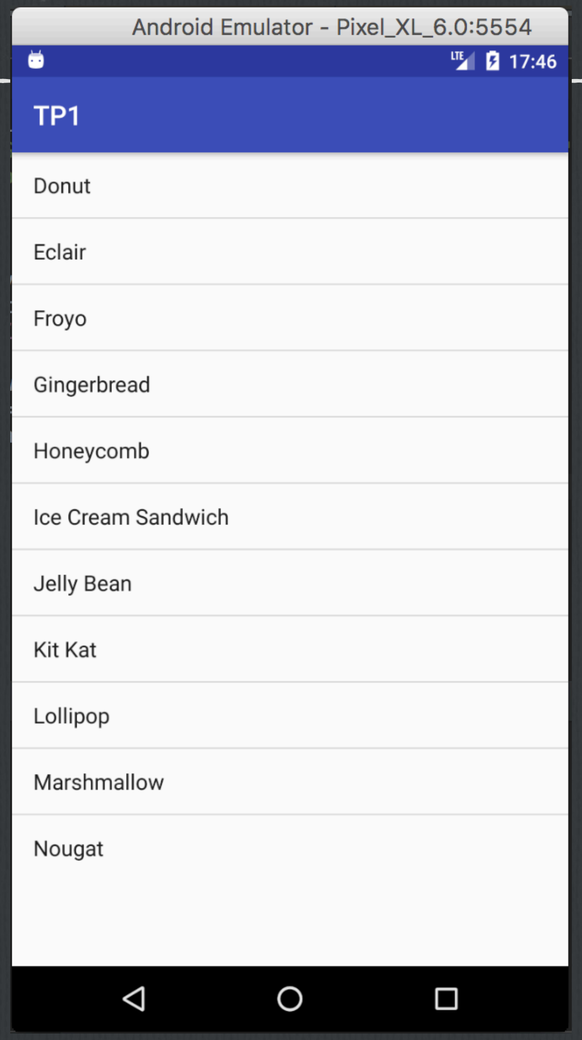
Les listes - Exemple simple

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/
android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ListView
        android:id="@+id/list"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        />
</RelativeLayout>
```

Les listes - Exemple simple

```
private ListView listView;
private String[] values = new String[] {
    "Donut", "Eclair", "Froyo", "Gingerbread", "Honeycomb", "Ice Cream
Sandwich", "Jelly Bean", "Kit Kat","Lollipop", "Marshmallow", "Nougat"
};

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_login);

    listView = (ListView) findViewById(R.id.list);
    ArrayAdapter<String> adapter =
        new ArrayAdapter<>(this, android.R.layout.simple_list_item_1,
android.R.id.text1, values);
    listView.setAdapter(adapter);
}
```

Les listes - Exemple simple



Les listes - Les adapters

- [ ] Créer un adaptateur personnalisé.
    - [ ] Personnalisation de la liste.
    - [ ] Personnalisation de l'UI.

Les listes - Les adapters

- [ ] **Layout représentant une ligne de la vue.**

```
<?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="wrap_content"
    android:orientation="horizontal"
    >

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@android:drawable/sym_def_app_icon"
        />

    <TextView
        android:id="@+id/list_text_item"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_gravity="center_vertical"
        tools:text="item"
        />
</LinearLayout>
```

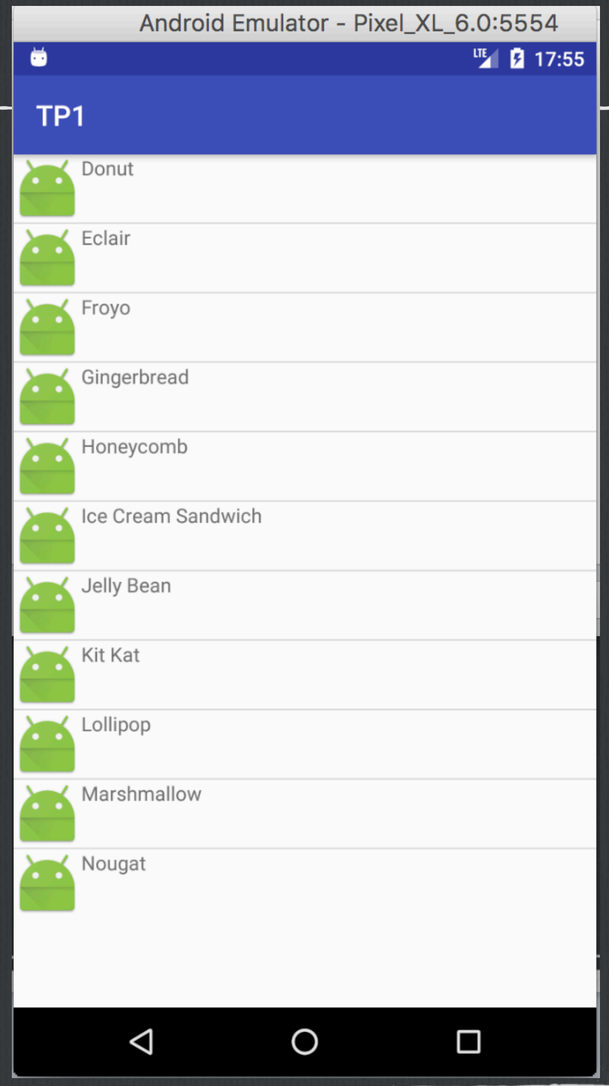
Les listes - Les adapters

```
public class MyAdapter extends ArrayAdapter<String> {

    private String[] data;
    private Context context;
    private int res;
    private LayoutInflater inflater;

    public MyAdapter(Context context, int resource, String[] objects) {
        super(context, resource, objects);
        data = objects;
        res = resource;
        this.context = context;
        inflater = (LayoutInflater) context.
            getSystemService(Context.LAYOUT_INFLATER_SERVICE);
    }

    @Override
    public int getCount() {
        if (data != null) return data.length;
        return 0;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        convertView = inflater.inflate(res, null);
        ((TextView) convertView.findViewById(R.id.list_text_item)).setText(data[position]);
        return convertView;
    }
}
```

Les listes - Les adapters

```
public class LoginActivity extends AppCompatActivity {

    private ListView listView;
    private String[] values = new String[] {
        "Donut", "Eclair", "Froyo", "Gingerbread", "Honeycomb", "Ice Cream
Sandwich", "Jelly Bean", "Kit Kat","Lollipop", "Marshmallow", "Nougat"
    };

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);

        listView = (ListView) findViewById(R.id.list);
        MyAdapter adapter = new MyAdapter(this, R.layout.listview_line, values);
        listView.setAdapter(adapter);
    }
}
```

Les listes - Les adapters



Les listes - ViewHolder

- ☐ Inflate et findViewById très couteux.
- ☐ Un appel par ligne de la liste.
- ☐ Optimiser le chargement de chaque ligne.

Les listes - ViewHolder

```
@NonNull
@Override
public View getView(int position, View convertView, @NonNull ViewGroup parent) {
    ViewHolder holder;

    if (null == convertView) {
        convertView = inflater.inflate(res, null);
        holder = new ViewHolder();
        holder.listItemText = (TextView) convertView.findViewById(R.id.list_text_item);
        convertView.setTag(holder);
    } else {
        holder = (ViewHolder) convertView.getTag();
        holder.listItemText.setText(data[position]);
    }

    return convertView;
}

private static class ViewHolder {
    TextView listItemText;
}
```

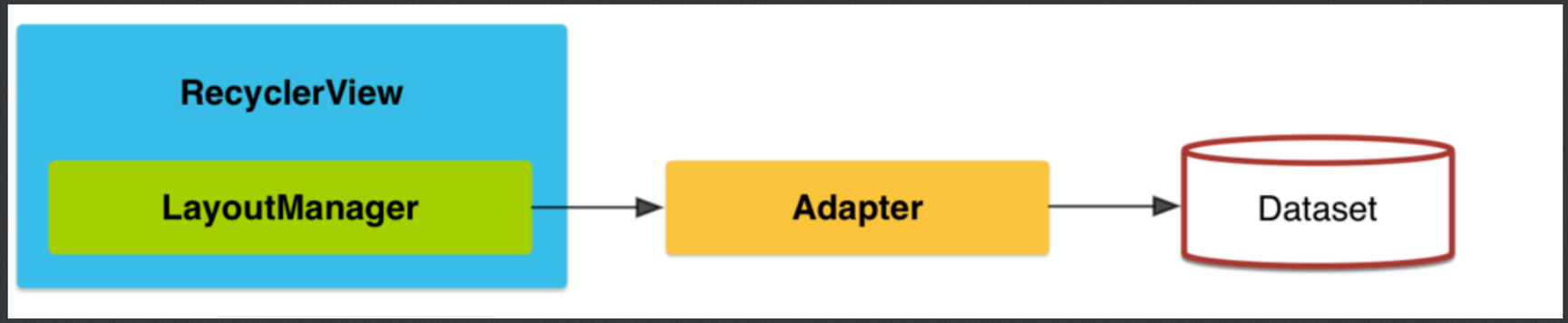
Les listes - Gestion du clic

- Ajouter un clic sur les éléments.
- Les paramètres sont : adaptateur, vue, position, identifiant.

```
listView.setOnItemClickListener(new AdapterView.OnItemClickListener()
{
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int
position, long id) {

    }
});
```

RecyclerView

- Une version améliorée, optimisée et flexible des listes.
- Maintien un nombre limité de vue dans la liste.



RecyclerView

- [ ] Un LayoutManager (LinearLayoutManager, GridLayoutManager …).
- [ ] Un adaptateur.
- [ ] Disponible dans la support library.

```
compile 'com.android.support:recyclerview-v7:25.3.1'
```

RecyclerView

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/
android" android:layout_width="match_parent"
    android:layout_height="match_parent"
    >

    <android.support.v7.widget.RecyclerView
        android:id="@+id/my_recycler_view"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        />

</RelativeLayout>
```

RecyclerView

```
public class LoginActivity extends AppCompatActivity {

    private RecyclerView mRecyclerView;
    private RecyclerView.Adapter mAdapter;
    private RecyclerView.LayoutManager mLayoutManager;

    private String[] values = new String[] {
        "Donut", "Eclair", "Froyo", "Gingerbread", "Honeycomb", "Ice Cream Sandwich", "Jelly Bean", "Kit Kat",
        "Lollipop", "Marshmallow", "Nougat"
    };

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);

        mRecyclerView = (RecyclerView) findViewById(R.id.my_recycler_view);
        mRecyclerView.setHasFixedSize(true);
        mLayoutManager = new LinearLayoutManager(this);
        mRecyclerView.setLayoutManager(mLayoutManager);
        mAdapter = new MyAdapter(values);
        mRecyclerView.setAdapter(mAdapter);
    }
}
```

RecyclerView

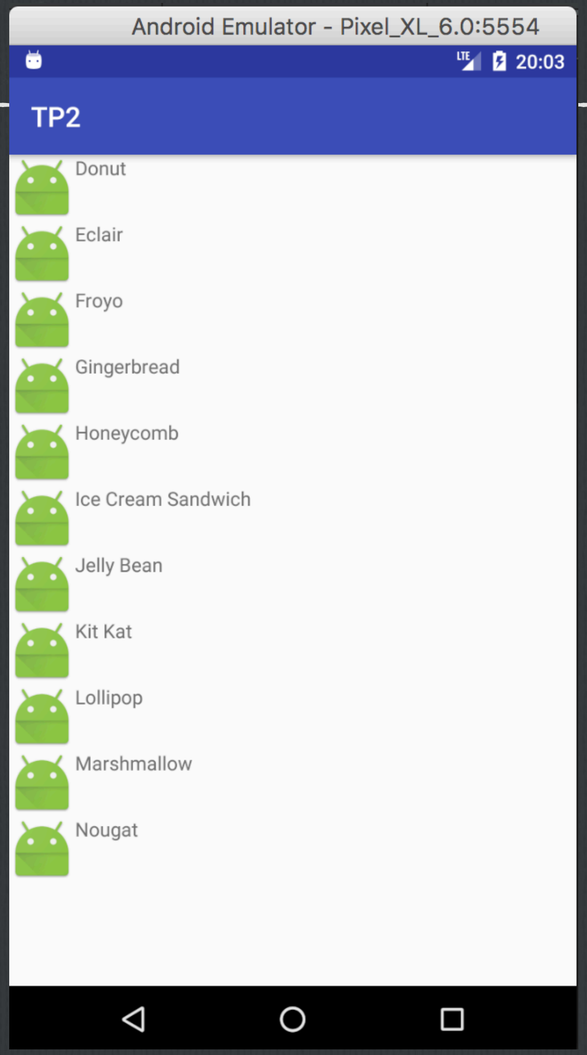
```
public class MyAdapter extends RecyclerView.Adapter<MyAdapter.ViewHolder> {

    private String[] mDataset;

    static class ViewHolder extends RecyclerView.ViewHolder {
        private TextView mTextView;

        public ViewHolder(View v) {
            super(v);
            mTextView = (TextView) v.findViewById(R.id.list_text_item);
        }
    }

    public MyAdapter(String[] myDataset) {
        mDataset = myDataset;
    }

    @Override
    public MyAdapter.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View v = LayoutInflater.from(parent.getContext()).inflate(R.layout.listview_line, parent, false);
        return new ViewHolder(v);
    }

    @Override
    public void onBindViewHolder(ViewHolder holder, int position) {
        holder.mTextView.setText(mDataset[position]);
    }

    @Override
    public int getItemCount() {
        return mDataset.length;
    }
}
```

TP

TP

- [ ] Créer un projet qui possède une activité.
- [ ] Cette activité contient un fragment, ajouté dynamiquement.
- [ ] Ce fragment affiche la liste des versions d’Android dans une recycler view.

TP


Android Emulator - Pixel_XL_6.0:5554
20:03
TP2
Donut
Eclair
Froyo
Gingerbread
Honeycomb
Ice Cream Sandwich
Jelly Bean
Kit Kat
Lollipop
Marshmallow
Nougat