Activity, Intent

Jean-Claude Tarby

Laboratoire CRIStAL

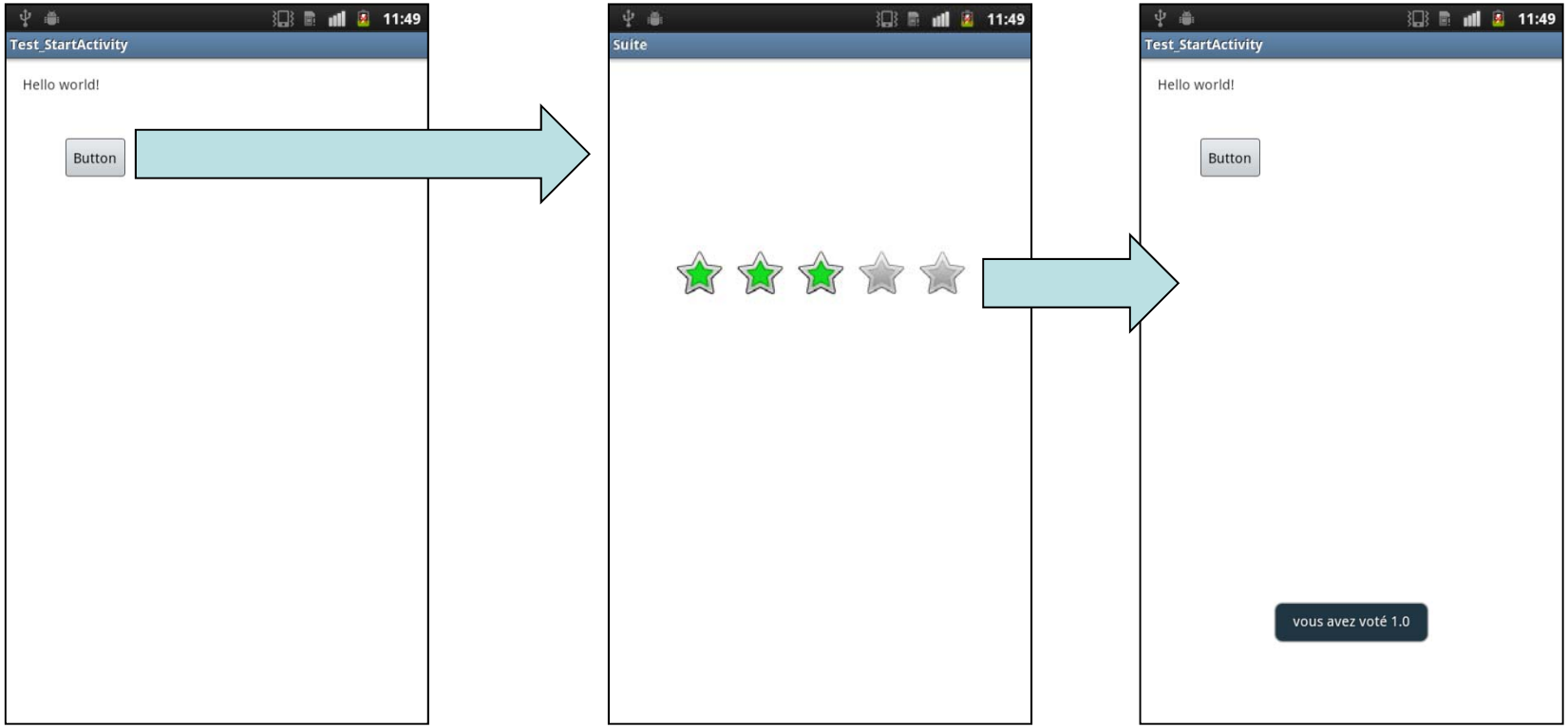
Université Lille 1

Appel d’activité

Appel d'activité

- Rappel du principe général
  - Une application Android = des activités qui appellent des activités
- TP :
  - Cherchez les **4 façons/variations** d'appeler une activité **de votre application** (cf. exemple ci-après)
    - (vous pouvez utiliser mes URLs du début de cours ou les vôtres)
    - N'appelez pas une autre application ! (+ tard)
    - Ne vous occupez pas des « filtres » (+ tard)
  - Mettez en pratique chacune d'elle sur un petit exemple

L’application à réaliser



Activité 1 | Activité 2 | Activité 1

A vous…

- Vous avez une heure…

L’application à réaliser

- Activité 1 : un bouton
- Activité 2 : une ratingBar (étoiles pour voter)

- **Solution 1** : on appelle simplement l’activité 2
- **Solution 2** : on appelle l’activité 2 avec un code OK / PAS_OK en retour
- **Solution 3** : on appelle l’activité 2 avec un code OK / PAS_OK + le résultat du vote en retour
- **Solution 4** : comme la solution 3 avec initialisation du vote par l’activité 1 (passage de paramètre)

- Cf. TP_StartActivity_RatingBar, solution 1 (les autres solutions seront vues plus loin avec d’autres exemples…)
  - https://gitlab.com/m2eservices/TP_StartActivity_RatingBar.git (commit 1)

Les Intent…

- Dans méthode de l'activité
  - Intent intent = new Intent(**this**, Suite.class);
  - Intent intent = new Intent(**getApplicationContext()**, Suite.class);
  - … cf plus loin pour les contextes…

- Dans un « onClick(View v) »
  - Intent intent = new Intent(**v.getContext()**, Suite.class);

N’oubliez pas !

- N’empilez pas vos activités à l’infini
  - Activité A → B → C → A → B → C !!!
- Quand on fait un « back », on « finish » l’activité
  - donc on passe par les onStop…onDestroy…
  - donc ne pas oublier de « nettoyer » avant de quitter
    - Arrêter les listeners (par exemple pour le GPS…)
- On peut empêcher le « back »
  - on le verra plus loin
- On peut dépiler plusieurs activités en une seule fois (on le verra dans quelques slides…)
  - on le verra plus loin

Concept d'Intent

- Communication par messages (issus de la classe Intent)
- Principe fondamental en Android !
- Evite les dépendances !
- Permet à Android de choisir le meilleur correspondant (mode implicite) sauf si on choisit explicitement à qui envoyer le message.
- Les applications peuvent filtrer les Intents pour ne répondre qu'à ce qu'elles veulent.

- Intent, Activité, Vue : « métaphore de personnes avec habits »

Intent

- Un Intent a essentiellement 4 usages :
  - démarrer une activité dans l'application (navigation entre écrans et appel de boite de dialogue) ; en mode explicite !
  - solliciter une autre application (lecture QRCode par exemple) ; c'est Android qui « résoud » l'intention (mode implicite généralement).
  - envoyer des informations
  - démarrer un service
  - peut aussi être diffusé à plusieurs applications (par exemple pour informer que la batterie est faible)

Intent

- Un Intent contient :
  - le nom du composant ciblé (optionnel)
  - l'action
  - les données (en MIME tel que ACTION_VIEW + url)
  - les données supplémentaires sous la forme de paires clé/valeur
  - la catégorie de l'Intent (par exemple CATEGORY_BROWSABLE ou CATEGORY_LAUNCHER)
  - les drapeaux : principalement utilisé pour dire comment démarrer une activité ; par exemple FLAG_ACTIVITY_NO_ANIMATION
- On peut créer ses propres Intent en créant une classe héritant de Intent et en redéfinissant les constructeurs et méthodes nécessaires.

Quel « context » utiliser ?

- http://stackoverflow.com/questions/10347184/difference-and-when-to-use-getapplication-getapplicationcontext-getbasecon

- **Toast** and **Intent**, both requires reference to *context*. And **getApplication**, **getApplicationContext**, **MyActivity.this** and **getBaseContext**, they all offer reference to the context.

- Now the thing confuses is the declaration of different contexts and their specific-usage. To make things simple, you should count two types of context available in the Android framework.
  - Application Context
  - Activity Context
- **Application** context is attached to the application's life-cycle and will always be same throughout the life of application. So if you are using *Toast*, you can use application context or even activity context (both) because a toast can be raised from anywhere with in your application and is not attached to a window.
- **Activity** context is attached to the Activity's life-cycle and can be destroyed if the activity's onDestroy() is raised. If you want to launch a new activity, you must need to use activity's context in its *Intent* so that the new launching activity is connected to the current activity (in terms of activity stack). However, you may use application's context too to launch a new activity but then you need to set flag Intent.FLAG_ACTIVITY_NEW_TASK in intent to treat it as a new task.

- LoginActivity.this though its referring to your own class which extends Activity class but the base class (Activity) also extends Context class, so it can be used to offer activity context.
- getApplication() though its referring to Application object but the Application class extends Context class, so it can be used to offer application context.
- getApplicationContext() offers application context.
- getBaseContext() offers activity context.

- **Tips: Whenever you need to manipulate Views then go for *Activity-Context*, else *Application-Context* would be enough.**

Démarrer une activité sans se préoccuper de la suite (pas de retour)

- **ERREUR DU DÉBUTANT : Exception lors du démarrage d'une activité**
  - Avant de pouvoir démarrer l'activité, il faut au préalable la déclarer dans le fichier de configuration AndroidManifest.xml de l'application. Sans cela, une erreur du type ActivityNotFoundException sera générée lors de l'appel de la méthode startActivity.

- Exemple de déclaration dans un Manifest :

```
<application ...>
    <activity android:name=".MonActivite" />
</application>
```

- Exemple d'appel :

```
Intent intent=new Intent(this, SecondActivity.class);
startActivity(intent);
```

- **A retenir : Spécifier le nom de la classe dans l'attribut android:name**
  - L'attribut android:name attend le nom complet de la classe. Cela comprend l'intégralité de l'espace de nom, le nom du paquetage et le nom de la classe. Si une classe MaClasse est dans le paquetage maCompagnie.monApplication, alors le nom complet sera maCompagnie.monApplication.MaClasse.

- Cf. projet « TP_StartActivitySimple » ou https://gitlab.com/m2eservices/TP_StartActivitySimple.git
  - Notez la mémorisation des données en cas de récréation de l'activité
  - Notez que quand on fait « back », on retrouve l'activité de départ (fonctionnement normal d'Android).

Démarrer une activité avec retour

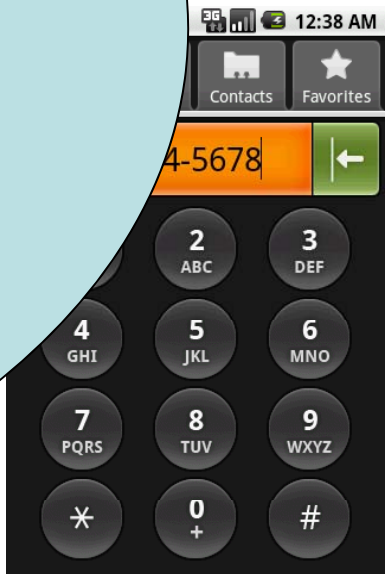
- Au lieu de **startActivity**, on utilise **startActivityForResult** qui **avertit le parent** quand l'activité est finie et lui **renvoie une valeur** de retour.

- Cf. projet TP_StartActivityWithResult
  - https://gitlab.com/m2eservices/TP_StartActivityWithResult.git

Lancer une autre application (1)

- Basé s. Les
  inf t le type de

TP à venir ☺



Lancer une autre application (2)

- Pour la

```
Un
                                                    tivities");
```

TP à venir ☺

```
                                                    es")
```

-

```
                _                    _
In
start
```

d market, vous devez avoir
un compte Google sur le téléphone

Lancer une autre application (3)

- pour la
  u

TP à venir ☺

Lancer une autre application (4)

- pour affich
  // M
  u
  e-
  is zoom
- h
  –

TP à venir ☺

Activité

```
<activity android:allowEmbedded=["true" | "false"]
          android:allowTaskReparenting=["true" | "false"]
          android:alwaysRetainTaskState=["true" | "false"]
          android:autoRemoveFromRecents=["true" | "false"]
          android:banner="drawable resource"
          android:clearTaskOnLaunch=["true" | "false"]
          android:configChanges=["mcc", "mnc", "locale",
                    "touchscreen", "keyboard", "keyboardHidden",
                    "navigation", "screenLayout", "fontScale",
                    "uiMode", "orientation", "screenSize",
                    "smallestScreenSize"]
          android:documentLaunchMode=["intoExisting" | "always" |
                    "none" | "never"]
          android:enabled=["true" | "false"]
          android:excludeFromRecents=["true" | "false"]
          android:exported=["true" | "false"]
          android:finishOnTaskLaunch=["true" | "false"]
          android:hardwareAccelerated=["true" | "false"]
          android:icon="drawable resource"
          android:label="string resource"
          android:launchMode=["multiple" | "singleTop" |
                    "singleTask" | "singleInstance"]
          android:maxRecents="integer"
          android:multiprocess=["true" | "false"]
          android:name="string"
          android:noHistory=["true" | "false"]
          android:parentActivityName="string"
          android:permission="string"
          android:process="string"
          android:relinquishTaskIdentity=["true" | "false"]
          android:screenOrientation=["unspecified" | "behind" |
                    "landscape" | "portrait" |
                    "reverseLandscape" | "reversePortrait" |
                    "sensorLandscape" | "sensorPortrait" |
                    "userLandscape" | "userPortrait" |
                    "sensor" | "fullSensor" | "nosensor" |
                    "user" | "fullUser" | "locked"]
          android:stateNotNeeded=["true" | "false"]
          android:taskAffinity="string"
          android:theme="resource or theme"
          android:uiOptions=["none" | "splitActionBarWhenNarrow"]
          android:windowSoftInputMode=["stateUnspecified",
                    "stateUnchanged", "stateHidden",
                    "stateAlwaysHidden", "stateVisible",
                    "stateAlwaysVisible", "adjustUnspecified",
                    "adjustResize", "adjustPan"] >
    . . .
</activity>
```

Beaucoup de possibilités !
http://developer.android.com/guide/topics/manifest/activity-element.html

Quelques éléments d'activité

- La balise **activity** peut contenir :
  - **android:label**
    - texte qui sera présenté à l'utilisateur pour tout affichage. Généralement c'est le nom en haut de l'écran correspondant à l'activité. S'il n'y a pas de label, le texte est la valeur du label de l'application.
  - **android:name**
    - le nom exact de la classe Java correspondante à l'activité en question. Commencer par un point indique qu'on se place dans le package spécifié dans la balise *application*.
  - **android:icon**
    - le nom d'un fichier (sans l'extension) se trouvant dans un des répertoires drawable (ou mipmap). Ce fichier contient un icône qui sera affichée pour représenter l'activité. Si celle-ci est de type LAUNCHER et qu'elle dispose d'un icône, elle apparaîtra dans le launcher avec l'icône spécifié ici. Sinon c'est l'icône de l'application.

Quelques éléments d'activité

- **android:launchMode**
  - 4 valeurs possibles :
    - **standard** : l'activité peut être instanciée plusieurs fois et chaque instance peut se trouver n'importe où et dans n'importe quelle pile d'activités (d'une tâche).
    - **singleTop** : comme le mode **standard**, mais si une instance de cette activité est déjà en haut de la pile et qu'on souhaite exécuter par dessus une activité identique (c-a-d instance de la même classe), alors il n'y aura pas de nouvelle instance et l'activité en cours aura sa méthode *onNewIntent* invoquée.
    - **singleTask** : l'activité doit être au démarrage d'une tâche, sous-entendu elle démarre une nouvelle tâche.
    - **singleInstance** : comme **singleTask**, sauf qu'elle ne permet aucune autre activité dans sa tâche

Quelques éléments d'activité (exemple)

- Between the Browser and Alarm Clock applications, you cover all four launch modes:
  - **BrowserActivity** uses **singleTask**. There is only one browser activity at a time and it doesn't become part tasks that send it intents to open web pages. While it might return to whatever most recently launched it when you hit back it is actually fixed at the bottom of its own task activity stack. It will share its task with activities that it launches like bookmarks.
  - **BrowswerBookmarksPage** uses **singleTop**. While there can be multiple instances of this activity, if there is already one at the top of the task's activity stack it will be reused and onNewIntent()will be called. This way you only have to hit back once to return to the browser if the bookmarks activity is started multiple times.
  - **AlarmClock** uses **standard**. The user can launch multiple instances of this activity and these instances can be part of any task and anywhere in the activity stack. As a fairly simple application it doesn't really demand tight control of its activity.
  - **AlarmAlert** uses **singleInstance**. Only one alert activity at a time and it is always its own task. Anything it launches (if anything) becomes part of its own new task.

- Source : http://stackoverflow.com/questions/2626218/examples-for-android-launch-modes

Quelques éléments d'activité

- **android:screenOrientation**
  - indique l'orientation de l'interface utilisateur pour cette activité.
  - 4 valeurs intéressantes :
    - **unspecified** : valeur par défaut. L'orientation dépend des réglages de l'OS.
    - **landscape** : paysage.
    - **portrait** : portrait.
    - **sensor** : selon l'orientation du téléphone.
  - (rappel) liste complète = ["unspecified" | "behind" |
    "landscape" | "portrait" |
    "reverseLandscape" | "reversePortrait" |
    "sensorLandscape" | "sensorPortrait" |
    "userLandscape" | "userPortrait" |
    "sensor" | "fullSensor" | "nosensor" |
    "user" | "fullUser" | "locked"]
- **android:alwaysRetainTaskState**
  - fixée à vrai (faux par défaut), le système sauvegarde l'état de la tâche. Ainsi même après un certain temps d'inutilisation (qui normalement signifie une suppression de la mémoire), un retour à l'application signifie un retour à l'état précédent.

Actions et catégories standards

- http://developer.android.com/reference/android/content/Intent.html

- ACTION_MAIN
- ACTION_VIEW
- ACTION_ATTACH_DATA
- ACTION_EDIT
- ACTION_PICK
- ACTION_CHOOSER
- ACTION_GET_CONTENT
- ACTION_DIAL
- ACTION_CALL
- ACTION_SEND
- ACTION_SENDTO
- ACTION_ANSWER
- ACTION_INSERT
- ACTION_DELETE
- ACTION_RUN
- ACTION_SYNC
- ACTION_PICK_ACTIVITY
- ACTION_SEARCH
- ACTION_WEB_SEARCH
- ACTION_FACTORY_TEST

détails slide suivant

- CATEGORY_DEFAULT
- CATEGORY_BROWSABLE
- CATEGORY_TAB
- CATEGORY_ALTERNATIVE
- CATEGORY_SELECTED_ALTERNATIVE
- CATEGORY_LAUNCHER
- CATEGORY_INFO
- CATEGORY_HOME
- CATEGORY_PREFERENCE
- CATEGORY_TEST
- CATEGORY_CAR_DOCK
- CATEGORY_DESK_DOCK
- CATEGORY_LE_DESK_DOCK
- CATEGORY_HE_DESK_DOCK
- CATEGORY_CAR_MODE
- CATEGORY_APP_MARKET

Actions standards

ACTION_ANSWER	Prendre en charge un appel entrant
ACTION_CALL	Appeler un numéro de téléphone. Cette action lance une activité affichant l'interface pour composer un numéro puis appelle le numéro contenu dans l'URI spécifiée en paramètre.
ACTION_DELETE	Démarrer une activité permettant de supprimer une donnée identifiée par l'URI spécifiée en paramètre
ACTION_DIAL	Afficher l'interface de composition des numéros. Celle-ci peut être pré-remplie par les données contenues dans l'URI spécifiée en paramètre
ACTION_EDIT	Éditer une donnée
ACTION_SEARCH	Démarrer une activité de recherche. L'expression de recherche de la pourra être spécifier dans la valeur du paramètre SearchManager.QUERY envoyé en extra de l'action
ACTION_SEND	Envoyer des données texte ou binaire par courriel ou SMS. Les paramètres dépendront du type d'envoi
ACTION_SENDTO	Lancer une activité capable d'envoyer un message au contact défini par l'URI spécifiée en paramètre
ACTION_VIEW	Démarrer une action permettant de visualiser l'élément identifié par l'URI spécifiée en paramètre. C'est l'action la plus commune. Par défaut les adresses commençant par *http:* lanceront un navigateur web, celles commençant par *tel:* lanceront l'interface de composition de numéro et celles débutant par *geo:* lanceront Google Map
ACTION_WEB_SEARCH	Effectuer une recherche sur Internet avec l'URI spécifiée en paramètre comme requête

Comment traiter la demande d'Intent pour l'application qui reçoit la demande ?

- Une fois le filtre d'Intents défini et opérationnel, quelques lignes de code suffiront pour traiter l'Intent transmis.
- Si le composant n'est pas démarré, le système s'occupera de créer automatiquement une nouvelle instance du composant pour traiter l'intention.

- L'application qui **reçoit la demande** :

```
@Override
public void onCreate(Bundle savedInstanceState) {
...
String data = getIntent().getDataString();
if(data != null)
// Ici le code à exécuter
...
}
```

- Il est possible de récupérer la donnée sous forme d'un objet Uri tel qu'il a été envoyé initialement :

```
Uri data = getIntent().getData();
if(data != null)
// Ici le code à exécuter
```

Comment passer des données en paramètres ?

On utilise de paires clé/valeur qu'on « charge » dans l'Intent avec putExtra et qu'on récupère à l'arrivée avec getExtras.

Activité qui appelle :

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    Intent intent = new Intent(this,MonActivite.class);
    intent.putExtra("maclé", "Oui ça marche !");
    startActivity(intent);
}
```

Activité appelée :

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_mon_activite);

    Bundle extra = this.getIntent().getExtras();
    if (extra != null)
    {
      String data = extra.getString("maclé");

      ((TextView)findViewById(R.id.monTexte01)).setText(
  data);
    }
}
```

- Cf. projet TP_StartActivityWithParameters
  - https://gitlab.com/m2eservices/TP_StartActivityWithParameters.git
- Cf. solution « complète » du TP RatingBar du début du cours :
  - « TP_StartActivity_RatingBar, 4 solutions »
  - https://gitlab.com/m2eservices/TP_StartActivity_RatingBar.git (commit 4)

Filtre d'Intent

- Pour définir un filtre d'Intents, déclarez l'élément intent-filter dans le Manifest.
- Exemple

```
<activity ...
        <intent-filter>
                <action android:name="android.intent.action.VIEW" />
                <category android:name="android.intent.category.DEFAULT" />
                <category android:name="android.intent.category.BROWSABLE" />
                <data android:scheme="demo" />
        </intent-filter>
</activity>
```

- Notez que ceci est écrit DANS une ACTIVITE et non pas au niveau global du Manifest ! !
- Chaque activité a donc ses propres filtres

Filtre d'Intent

- Chaque élément de la balise est important puisqu'il détermine le niveau de filtrage :
  - ***action*** *:* identifiant unique sous forme de chaîne de caractères. Il est d'usage d'utiliser la convention de nom Java ;
  - ***category*** : premier niveau de filtrage de l'action. Cette balise indique dans quelle circonstance l'action va s'effectuer ou non. Il est possible d'ajouter plusieurs balises de catégorie ;
    - Note : il existe une catégorie MONKEY qui correspond à une application qui va générer des comportements aléatoires comme si l'utilisateur faisait n'importe quoi ("stress-test"), Intéressant !!!!!
  - ***data*** : filtre l'objet Intent au niveau des données elles-mêmes.
    - Par exemple en jouant avec l'attribut android:host on peut répondre à une action comportant un nom de domaine particulier, comme *www.monsite.com*.

- Grâce à la définition de ce filtre, l'activité précédente réagira à l'action du code suivant :

```
Uri uri = Uri.parse("demo://ceci est une chaine de caractère");
Intent intent = new Intent(Intent.ACTION_VIEW, uri);
startActivity(intent);
```

- Cf. exemple d'appelant et d'appelé dans projet TP_FilterAppelant et TP_FilterAppele
  - https://gitlab.com/m2eservices/TP_FilterAppelant.git
  - https://gitlab.com/m2eservices/TP_FilterAppele.git

Vérifier que « quelqu'un » peut répondre à votre Intent

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

si **isIntentSafe** est vrai alors, il y a au moins une application qui peut répondre à votre Intent, sinon il y en a aucune

→ Lancement de l'application par défaut, ou affichage d'une liste si pas d'application par défaut

Proposer un choix à l'utilisateur

- Pour proposer un choix à l'utilisateur (même s'il a des applications « par défaut ») :

```
Intent intent = new Intent(Intent.ACTION_SEND);
...
String title = (String) getResources().getText(R.string.chooser_title);
Intent chooser = Intent.createChooser(intent, title);
startActivity(chooser);
```

Permissions associées aux actions

- Des permissions sont obligatoires pour beaucoup d'actions, par exemple pour appeler un n° de tel.
  - On ne peut pas faire que startActivity avec un ACTION_CALL, il faut aussi déclarer la permission dans le Manifest avec :

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

- Si on n'a pas déclaré les permissions, alors plantage à l'exécution !

- Pensez que depuis la version 6, Android propose des permissions à la volée, reportez-vous à la doc officielle

Reculer de plusieurs activités en une seule fois

- Main Activity -> Call Activity 1 -> Call Activity 2 -> Call Activity 3 -> Call Activity 4 -> Call Activity 5
  - Objectif : de Activity 5, on veut « revenir » directement à MainActivity « proprement »
- Indication : `FLAG_ACTIVITY_CLEAR_TOP`
- Mettez cela en pratique (30 mns)

Reculer de plusieurs activités en une seule fois

- Source : http://stackoverflow.com/questions/12566662/how-to-destroy-multiple-android-activities-at-same-time
- En fait, on empile d'une certaine façon pour que le back fasse automatiquement le dépilement de X activités.

```
Intent intent = new Intent ( this , MainActivity.class );
intent.addFlags ( Intent.FLAG_ACTIVITY_CLEAR_TOP );
startActivity ( intent );
```

- Le principe est que si on démarre une activité avec ce drapeau, et qu'elle est déjà présente dans la pile, alors on revient à cette instance en ignorant les activités intermédiaires.
  - Cf. projet TP_Back_Multiple_Activites
  - https://gitlab.com/m2eservices/TP_Back_Multiple_Activites.git