P4 : Les composants

1. Les activités
2. Les services
3. Les Intents
4. Les broadcast receiver
5. Notifications de l'utilisateur (toast, boite de dialogue, barre notification)

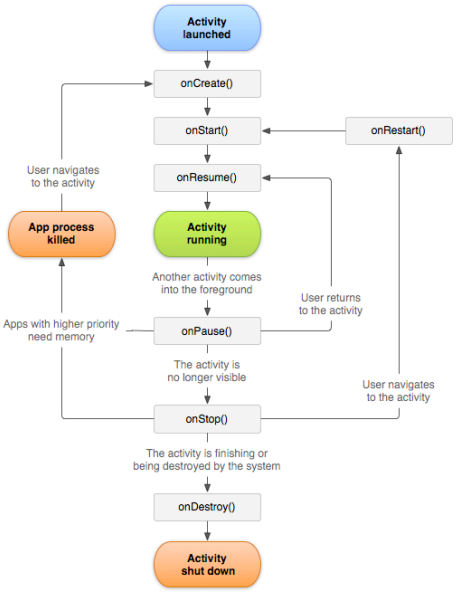
1 – Les activités

Une activité est la base pour la création d'une interface utilisateur.

L'activité permet d'interagir avec l'application.

Plusieurs activités peuvent composer une même application.

Chaque activité doit être déclarée dans le manifest.



2 – Les services

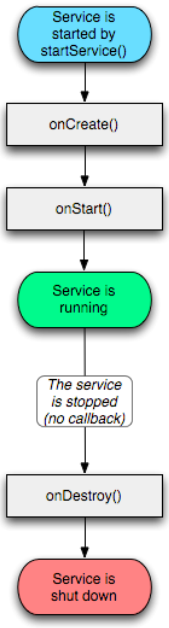
Le service ne dispose pas d'interface utilisateur (contrairement à l'activité).

Le service tourne en tâche de fond (thread séparé) pour une période définie ou non par le développeur (lecture de musique, collecte de données, suivi GPS, vérification de mise à jour ...).

Chaque service doit être déclaré dans le manifest.

2 – Les services

Cycle de vie :



3 – Les Intents

Les intents permettent de communiquer entre les Activités, les Services et les Broadcast receiver.

Il est possible de passer des informations entre activités grâce aux intents.

Il est également possible de solliciter une autre application (par exemple envoyer un e-mail, la lecture d'un document pdf).

3 – Les Intents

Démarrer une activité :

```
Intent edit = new Intent(this, Activity.class);
edit.putExtra("categorie", "la categorie");


startActivity(edit);
```

Récupérer les informations envoyées :

```
if(this.getIntent().getExtras() != null) {
    categorie   = this.getIntent().getExtras().getString("categorie");

}
```

3 – Les Intents

Démarrer une activité en passant un objet :

```
Intent edit = new Intent(getParent(), QuestionsReponses.class);

edit.putExtra("position", position);
edit.putExtra("categorie", categorie.toString());

Bundle extras = new Bundle();
extras.putSerializable("questionreponse", (Serializable) items);
edit.putExtras(extras);

startActivity(edit);
```

L'objet devra implémenter « Serializable » .

3 – Les Intents

Démarrer une activité et récupérer le résultat (MainActivity):

```
Intent intent = new Intent(activity, HomeActivity.class);

activity.startActivityForResult(intent, 1000);
```

Définir un résultat (FormActivity) :

```
Intent dataIntent = new Intent();

dataIntent.putExtra("cle", "valeur");

activity.setResult(Activity.RESULT_OK, dataIntent);
```

Récupérer le résultat (MainActivity) :

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {

    if (resultCode == RESULT_OK && requestCode == 1000) {

    }

    super.onActivityResult(requestCode, resultCode, data);
}
```

3 – Les Intents

Solliciter l'application Mail :

```
Intent i = new Intent(Intent.ACTION_SEND);
i.setType("message/rfc822");
i.putExtra(Intent.EXTRA_SUBJECT, getString(R.string.partageemailapropostitre));
i.putExtra(Intent.EXTRA_TEXT, getString(R.string.partageemailapropos));
startActivity(Intent.createChooser(i, "Titre:"));
```

Solliciter une application de lecture de document pdf :

```
File file = new File("/sdcard/fichier.pdf");
Uri path = Uri.fromFile(file);

Intent intent = new Intent(Intent.ACTION_VIEW);
intent.setDataAndType(path, "application/pdf");
startActivity(intent);
```

Solliciter un navigateur Internet :

```
startActivity(new Intent(Intent.ACTION_VIEW, Uri.parse("http://www.google.fr")));
```

3 – Les Intents

Démarrer ou arrêter un service :

```
startService(new Intent(TabsActivity.this, ApplicationService.class));

stopService(new Intent(this, ApplicationService.class));
```

Service :

```
public class ApplicationService extends Service {

    @Override
    public void onCreate() {

        super.onCreate();
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {


    }

    @Override
    public void onDestroy() {

        super.onDestroy();
    }

    @Override
    public IBinder onBind(Intent intent) {
        // TODO Auto-generated method stub
        return null;
    }
```

4 – Broadcast receiver

Un broadcast receiver est un élément inactif (sans interface) qui attend un événement (batterie faible, changement de langue ...).

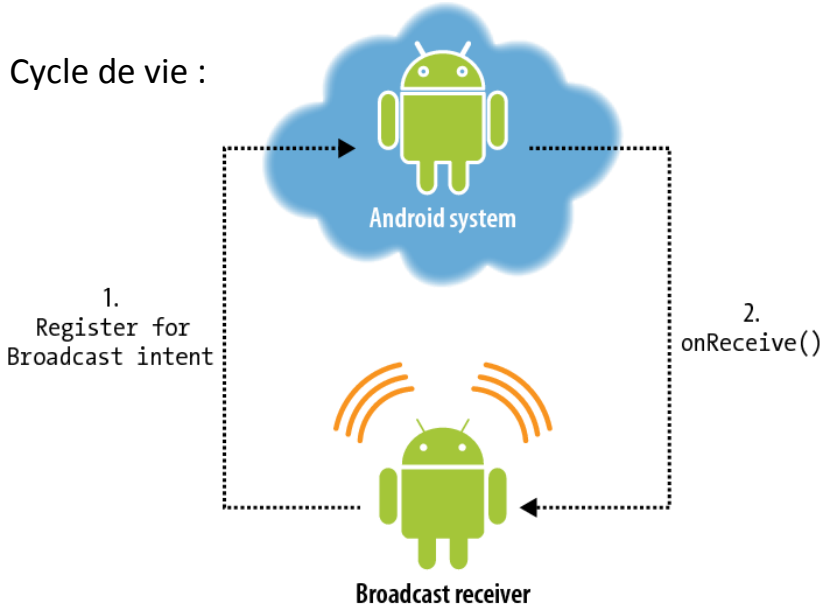
Il est possible de définir nos propres évènements.

Une application peut contenir plusieurs receiver (1 par événement important).

Il est possible de lancer des activités ou des notifications (vibration, alerte sonore ...) si nécessaire.

4 – Broadcast receiver

Cycle de vie :



4 – Broadcast receiver

Associer un Broadcast receiver (Activity) :

```
registerReceiver(receiver_SPEED, new IntentFilter("com.formation.speed"));
```

Dissocier un Broadcast receiver (Activity) :

```
unregisterReceiver(receiver_SPEED);
```

Envoyer un Broadcast (Service) :

```
sendBroadcast(new Intent("com.formation.speed").putExtra("speed", 100));
```

Class du Broadcast receiver (Activity) :

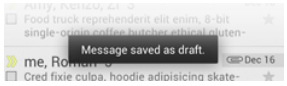
```
private class Receiver_SPEED extends BroadcastReceiver {
     @Override
     public void onReceive(Context arg0, Intent arg1) {

     }
}
```

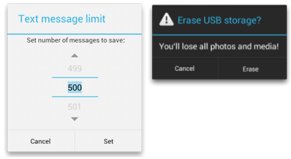
5 – Notifications de l'utilisateur

Une notification signale une information à l'utilisateur sans interrompre ses actions en cours.

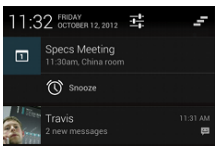
Le Toast : affiche une information pendant quelques secondes de façon à ce que l'utilisateur soit informé.



La boite de dialogue : affiche des messages d'alerte ou de demande de confirmation.



La barre de notification : alerte l'utilisateur sans utiliser d'activité.



5 – Notifications de l'utilisateur

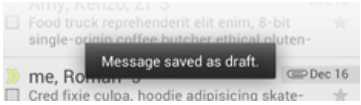
Afficher un Toast :

```
Context context = getApplicationContext();
CharSequence text = "Hello toast!";
int duration = Toast.LENGTH_SHORT;

Toast toast = Toast.makeText(context, text, duration);
toast.show();
```

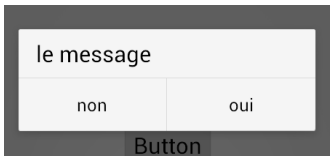
Ou plus simplement :

```
Toast.makeText(this, "Hello toast!", Toast.LENGTH_SHORT).show();
```



5 – Notifications de l'utilisateur

Afficher une boite de dialogue :

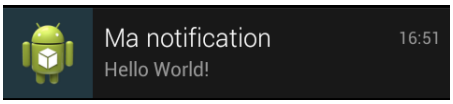



```
AlertDialog.Builder builder = new AlertDialog.Builder(this);
builder.setMessage("le message")
.setPositiveButton("oui", new DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int id) {
        // TODO action
    }
})
.setNegativeButton("non", new DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int id) {
        // TODO action
    }
});

builder.create();
```

5 – Notifications de l'utilisateur

Afficher une notification :




```
// Prepare intent which is triggered if the
// notification is selected
try {
    Intent intent = new Intent(this, SplashScreenActivity.class);
    intent.putExtra("message", message);
    intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
    PendingIntent pIntent = PendingIntent.getActivity(this, 0, intent, 0);

    // Build notification
    Notification noti = new NotificationCompat.Builder(this)
        .setContentTitle(getString(R.string.app_name))
        .setContentText(message).setSmallIcon(R.drawable.icone)
        .setContentIntent(pIntent)
        .build();
    NotificationManager notificationManager = (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
    // Hide the notification after its selected
    noti.flags |= Notification.FLAG_AUTO_CANCEL;

    notificationManager.notify(0, noti);
} catch(NoClassDefFoundError e) {
    e.printStackTrace();
}
```

7 – TP: développement d'une application …

TP « Guide » :

- Créer une nouvelle application avec une interface graphique
- Ecran « accueil » avec une bannière, un titre, une description et 2 boutons.
- Ecrans « hôtel » et « restaurant » avec une liste personnalisée.
- Ecran « détail » avec une photo, quelques informations et des boutons pour accéder au site Internet, envoyer un e-mail et lancer un appel.

7 – TP: développement d'une application ...