Introduction au développement sous Android
TP4

Master WeDSCi/I2L - 2^{me} année

année 2021-2022

Introduction

L'objectif de ce TP est d'expérimenter l'utilisation des intentions implicites dans le cadre des communications inter-composants. L'application développée dans le cadre de l'exercice 2, sera à rendre selon les modalités habituelles en fin de TP.

Exercice 1

Créez un nouveau projet Android nommé `TestImplicite`, puis complétez celui-ci avec les étapes successives suivantes :

1. Configurez l'interface graphique avec les boutons tels qu'ils apparaissent dans la figure 1. Vous penserez à faire figurer l'intitulé des boutons dans le fichier `strings.xml`.
2. Complétez votre activité principale de telle sorte que l'appui sur chacun d'un de ces boutons provoque l'affichage du texte du bouton dans la console. Il vous est demandé de gérer tous les boutons dans la même fonction *callback*;

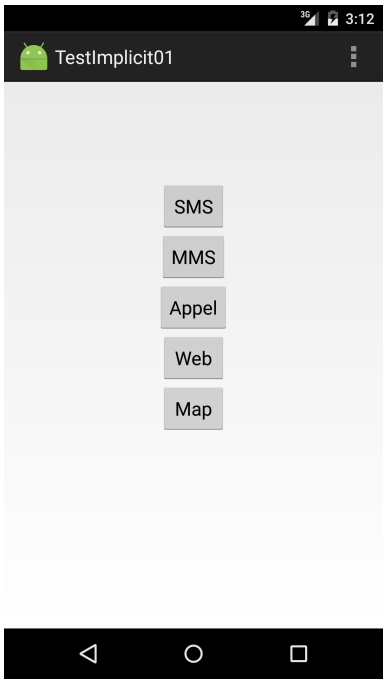

FIGURE 1 – Vue de l'interface graphique de l'exercice 1.

3. Modifiez le code de gestion de l'appui sur le bouton `SMS` de telle sorte que cet évenement déclenche la génération d'une intention visant l'envoi d'un sms. L'action à utiliser est `ACTION_SENDTO` et l'*URI* à créer doit utiliser le schema `sms` et une information correspondant à un numéro de téléphone quelconque. Testez cette fonctionnalité ;
4. Faites de même pour le bouton `MMS` ;

5. Modifiez le code de gestion de l'appui sur le bouton `Appel` de telle sorte que cet évenement déclenche la génération d'une intention visant l'appel vers le numéro fourni. L'action à utiliser est `ACTION_DIAL` et l'*URI* à créer doit utiliser le schema `tel` et une information correspondant à un numéro de téléphone quelconque ;
6. Complétez le code de gestion du bouton `web`, de manière à ce qu'il ouvre la page web
   `http://www-lisic.univ-littoral.fr`
   en cas d'appel. L'action à utiliser est `ACTION_VIEW` ;
7. Faites de même pour le bouton `Map`, en utilisant l'URI suivante (extraite de la documentation android) :
   `geo:0,0?q=1600+Amphitheatre+Parkway,+Mountain+View,+CA+94043`

Exercice 2

L'objectif de cet exercice est d'écrire une application nommée TPimageXXX, où XXX représente votre nom de *login*, permettant de charger une image stockée sur le périphérique et de l'afficher. Une vue de l'application à réaliser est donnée sur la figure 2 :

— l'image à charger se trouve au centre de l'interface ;
— le bouton de chargement se trouve en bas de l'écran ;
— le texte figurant en haut de l'écran correspond à l'*URI* de l'image chargée.

FIGURE 2 – Vue de l'interface graphique de l'exercice 2.

Avant de pouvoir charger une image, il est nécessaire que des images soient présentes sur le périphérique virtuel ... Pour ce faire, la méthode la plus simple est d'utiliser l'appareil de photo embarqué sur votre émulateur :

— Vérifiez, en utilisant le *Android Virtual Device manager*, que la caméra de votre périphérique est bien émulée. Dans le cas contraire, activez cette option ;
— Vous pouvez ensuite utiliser l'appareil photo pour générer quelques images de test ;
— Sur certaines versions d'émulateur se trouve une option *scène virtuelle* qui permet de prendre des images d'une scène un peu plus complexe qu'un simple damier.

étape 1

Dans un premier temps, créez l'interface graphique telle qu'elle apparaît sur la figure 2. On précise que le *widget* à utiliser pour l'image est `ImageView`. Compléter le code de l'application de manière à pouvoir gérer l'appui sur le bouton de chargement, en faisant afficher un message dans la console.

étape 2

Complétez à présent l'application de telle sorte qu'elle émette une intention dont l'action sera `ACTION_GET_CONTENT` et le type celui correspondant à une image, qu'elle récupère l'image retournée et l'affiche dans le *widget* correspondant.

On précise les points suivants :

— la classe à utiliser pour l'image est `ImageView` ;

— l'image à récupérer sera stockée dans une instance de la classe `Bitmap`, que vous récupérerez en utilisant le code suivant [1] :

```
//  ----- préparer les options de chargement de l'image
BitmapFactory.Options option = new BitmapFactory.Options();
option.inMutable = true; // l'image pourra être modifiée
// ------ chargement de l'image - valeur retournée null en cas d'erreur
Bitmap bm = BitmapFactory.decodeStream(getContentResolver().openInputStream(imageUri), null, option)
```

où `imageUri` représentera l'*URI* de l'image retournée par l'application extérieure ;

— pour pouvoir charger une image, votre application doit disposer de la permission `READ_EXTERNAL_STORAGE`.

étape 3

Lorsque le chargement de vos images fonctionne, ajoutez des boutons à votre application, permettant d'effectuer (i) un miroir horizontal de l'image (ii) un miroir vertical de l'image. Pour récupérer le bitmap mémorisé dans le *ImageView*, vous pourrez utiliser le code suivant :

```
// image est une instance de ImageView
 Bitmap bitmap = ((BitmapDrawable)image.getDrawable()).getBitmap();
```

On précise que ces opérations devront se faire pixel par pixel, et non pas en utilissant les opérations géométriques sur les images (solution la plus fréquemment rencontrée sur les forums ...) telles que `setScaleX`, `setScaleY`, utilisation d'une matrice de transformation, etc.

Vous utiliserez au contraire les méthodes `getPixel` et `setPixel` qui permettent un accès direct aux pixels d'un bitmap.

Exercice optionnel

Modifiez votre application `TestImplicite` de telle sorte qu'elle comporte :

— un champ de saisie permettant de récupérer le numéro de téléphone à utiliser pour les sms, mms et appels ;

— un champ de saisie pour l'url à ouvrir ;

— deux champs de saisie pour les coordonnées géographiques (latitude et longitude) d'un lieu à visualiser.

Il vous appartient de tester que les données saisies par l'utilisateur sont correctes ...

1. davantage d'informations seront données sur ce code lors de l'étude des `Content Provider`.