Formation

ANDROID Débutants

Session sur 4 journées

Ce document n'est qu'un résumé « brut »

Objet : Résumé « brut »

Votre formateur :
Lionel BEAUDOIN - Tél. : +33 4 27 46 00 46 - projet@cinrel.net
CINREL – Rue de la Mairie - 69440 SAINT ANDRE LA COTE - FRANCE

Table des Matières

1. PRÉSENTATION LANGAGE JAVA....3
1.1. Principes de base de la Programmation Objet....3
1.2. Glossaire....3
2. PRÉSENTATION ANDROID....4
2.1. Le Manifeste (AndroidManifest.xml)....4
2.2. Le contexte....4
2.3. L'activité (android.app.Activity)....4
2.4. Le Bundle....5
2.5. Le Service (android.app.Service)....5
2.6. L'intention (intent)....5
2.7. Broadcast receiver (android.content.BroadcastReceiver)....6
2.8. Content Provider (android.content.ContentProvider)....6
2.9. L'Interface Graphique Utilisateur (GUI)....7
2.10. Les Vues (view)....7
2.11. Les Fragments (android.app.Fragment)....7
2.12. Les Activités Dédiées....7
2.13. Les Dialogues et Messages....7
2.14. Les Menus....8
2.15. Les barres d'Actions Simples ou Rapides....8
2.16. Les Thèmes et les Styles....8
2.17. L'Instanciation des Vues....8
2.18. Les Ecouteurs (listener)....8
2.19. Les Adaptateurs....8
2.20. Les Notifications....8
2.21. Les Applications Multi-Ecrans....8
2.22. Les Application Multi-Langues....9
2.23. Les Variables globales....9
2.24. Persistance, Threading, Partage des Données....9
2.25. Gestion de fichiers....9
2.26. Géolocalisation : (android.location.LocationManager)....9
2.27. Les Services Systèmes....9
2.28. Les Alarmes et Timers....10
2.29. Linkify et TextWatcher....10

1. Présentation Langage Java

1.1. Principes de base de la Programmation Objet

- La programmation structurée
- Les types, structures et les classes
- Les méthodes
- Le mécanisme d'héritage
- Les types de données : char, int, long, boolean, float, double, string
- Durée de vie des variables
- Variables tableaux
- Opérateurs logiques, exemple : if (chaine.equals("valeur") || condition_2)
- Structure itératives

1.2. Glossaire

- classe : structure contenant des attributs et des méthodes
- méthodes : définissent les actions qu'une classe peut effectuer
- attributs : décrivent la classe par des variables de différents types
- objet : permet de mettre en œuvre une classe
- instance : allocation en mémoire des éléments d'une classe (attributs et méthodes) : new()
- private : visible que dans la classe
- protected : visible dans les classes héritées
- public : l'objet ou la méthode peut être utilisable depuis d'autres classe
- static : indique qu'il n'est pas nécessaire de créer une instance pour utiliser la méthode
- final : dans une déclaration de variable, l'affectation n'est valable qu'une fois = constante
- constructeurs : méthodes exécutées lors de l'instanciation (du même nom que la classe)
- extends : permet de définir l'héritage d'une classe (class Poisson extends Animaux {} ;)
- opérateur conditionnel : valeur = condition ? Valeur_vrai : Valeur_faux ;
- switch : switch(variable) { case 'A' : ….. break ; case 'B' : …. break ; }
- main : méthode principale : public static void main(String[] args) { … }
- this : permet de définir l'objet courant
- import : inclusion de paquetages Java, exemple : import javax.swing.* ;

2. Présentation Android

2.1. Le Manifeste (AndroidManifest.xml)

- Nomme le paquetage Java pour identification unique
- Déclare les composants applicatifs
- Déclare les permissions de l'application et le niveau minimum de compatibilité

2.2. Le contexte

Essentiellement, un contexte est une référence à des ressources gérées par l'application.
Il permet également de récupérer les informations sur le système.
Il fournit aussi l'accès à des services de l'Android.
Le contexte permet aussi de gérer d'autres aspects de l'application.
Il doit d'ailleurs être employé pour pouvoir lancer une nouvelle activité, réceptionner les intentions ou écouter les événements :

- création de nouveaux objets (TextView, Adapter...)
- accès à des ressources (String, Array...)
- accès implicite à des composants (par exemple *getApplicationContext().getContentResolver()* pour *content resolver*)

Méthodes pour récupérer le contexte :

- getApplicationContext()
- getContext()
- getBaseContext()
- this - mais disponible uniquement quand on récupère le contexte depuis une activité

- le contexte modélise les informations globales sur l'environnement de l'application
- Il possède les méthodes d'accès aux ressources de l'application
- les accès au contexte depuis une Activity ou un Service (this), un BroadcastReceiver, un ContentProvider

2.3. L'activité (android.app.Activity)

En analogie avec le monde des applications web, on peut dire qu'elle est une page web.
Elle représente des commandes que l'utilisateur peut faire à un moment donné.
Les activités sont gérées dans une pile d'activités. Chaque fois où une nouvelle activité est lancée, elle arrive à la première position de la pile et devient l'activité qui est actuellement exécutée. L'activité précédente reste dans la pile et ne pourra pas être lancée une fois la première activité terminée.
L'activité est représentée par la classe *Activity* dont 7 méthodes illustrent le cycle de vie qu'elle peut prendre :

- Une activité = un écran. Une application est formée de n activités
- Cycle de vie d'une activité au travers de 7 méthodes callback :
  onCreate, onStart, onRestart, onResume, onPause, onStop, onDestroy
  - onCreate : la méthode d'initialisation des vues, des paramètres et d'autres données. Elle prend en paramètre l'instance de la classe *Bundle*
  - onStart : appelée quand l'activité est rendue visible à l'utilisateur.
  - onRestart : appelée à un nouveau démarrage de la même activité (quand l'activité était arrêtée)
  - onResume : appelée quand l'activité commence à interagir avec l'utilisateur.
  - onPause : méthode qui sert à arrêter une activité temporairement.
  - onStop : cette fonction est utilisée quand l'activité n'est plus visible à l'utilisateur. Elle est cachée soit à cause d'une nouvelle activité lancée, soit parce que l'activité en cours s'apprête à être détruite.

- onDestroy : est invoquée quand l'activité est détruite. La destruction opère quand quelqu'un appelle la méthode *finish()* ou quand c'est le système qui décide de tuer l'activité pour économiser de l'espace.

- Lancement d'une activité :
  - doit avoir été déclarée dans le manifeste, dont une désignée comme initiale
  - lancement par startActivity ou startActivityForResult pour un résultat en retour
- La pile des activités :
  - empilée quand elle démarre et dépilée par la touche 'Back'
  - le bouton 'Home' ne dépile pas l'activité mais la met en arrière-plan
  - plusieurs piles d'activités coexistent sur Android

2.4. Le Bundle

Cette classe est une sorte de conteneur pour les données transmissibles d'une activité à l'autre.
Elle permet de récupérer tout type de données : long, char, ArrayList ...
Mais attention, c'est une des deux méthodes de transmission de données.
L'autre se base sur les données stockées dans des intentions.

2.5. Le Service (android.app.Service)

Les services sont des tâches qui peuvent être lancées avec ou sans intervention de l'utilisateur. Elles s'exécutent dans le background de l'application et peuvent se terminer soit après la finalisation de la tâche, soit à travers une intervention externe.
Les services représentent également une fonctionnalité d'une application exposée à d'autres applications.
Il est important de mentionner que le service ne fournit pas d'interface graphique (User Interface).
Un service concret hérite de la classe *Service* et surcharge des méthodes suivantes :

- onStartCommand : si un autre composant (par exemple une activité) fait appel à un service via *startService()*, c'est la méthode onStartCommande du service qui est appelée en premier.
- onBind : pareil qu'onStartCommand. La seule différence repose dans le fait que cette méthode est appelée quand un autre component appelle le service avec *bindService()*. onBind doit retourner *null* si l'on souhaite que l'invocation du service depuis une autre application soit impossible.
- onCreate : elle est appelée après onStartCommand ou onBind. Cette méthode contient toutes les procédures nécessaires à l'initialisation du service.
- onDestroy : cette fonction est invoquée quand l'exécution du service doit être terminée. L'implémentation de cette méthode est nécessaire car chaque fin de l'exécution libère des ressources de l'Android

- Un service = une opération ou un calcul sans interface utilisateur local ou distant
- Appel d'un service en mode Unbounded : startService et Stopservice
- Appel d'un service en mode ou Bounded : bindService, unbindService

2.6. L'intention (intent)

Les intentions sont des opérations qui permettent, entre autres, de naviguer entre les écrans. Grâce à elles on peut également passer des paramètres d'un écran à l'autre de notre application.
Cependant, la navigation n'est pas la seule fonctionnalité des intentions.
L'autre, aussi importante, est la possibilité d'écouter et de réagir à des événements extérieurs à l'application.
Par exemple, grâce à des intentions, on peut capter le moment où notre connexion WiFi n'est plus disponible et exécuter une action dans notre application.

On utilisera pour cela les broadcast receivers qu'on abordera plus loin dans la série.
Une intention est donc définie par :
- une liste d'événements auxquels elle doit répondre
- un composant de l'application qui doit s'occuper de gérer l'événement capté

On distingue deux types d'intentions :
- explicite : on sait quelle activité on veut démarrer (par exemple pour passer à une activité "ShowSite" après le click sur le bouton "afficher le site")
- implicite : on ne sait pas quelle activité doit être invoquée.

En occurrence, c'est Android qui va se charger de trouver l'activité adéquate à lancer (par exemple pour consulter un site web, il va lancer un des navigateurs disponibles).
Les intentions sont accompagnés de *filtres* qui regroupent les conditions que doit remplir une activité ou un broadcast receiver pour être exécuté.

- Les intentions : Activity, Service, BroadcastReceiver fonctionnent à travers un bus à messages
- Cycle de vie d'un service au travers de 4 méthodes callback
  onCreate, onStartCommand, onBind, onUnbind, onDestroy
- Nature directe si le composant cible est activé par le composant source
- Nature par procuration si le composant cible est activé par un élément tiers
- Les extras :
  - l'intent peut convoyer des données supplémentaires
  - Stockage par une hash table
  - Récupération par des méthodes propres à chaque type

2.7. Broadcast receiver (android.content.BroadcastReceiver)

Ce sont des classes implémentant *BroadcastReceiver* qui reçoivent des intentions et effectuent des actions spécifiques. (la batterie est faible, un sms vient d'arriver, etc …)
Ils vivent uniquement le temps qui est nécessaire pour traiter l'intention.
Le traitement s'effectue dans la méthode *onReceive()* qui prend en paramètre les instances du Context et de l'Intent.
Les broadcast receivers sont destinés à exécuter de petites tâches, comme l'envoi d'une nouvelle localisation GPS, traitement rapide d'un mail envoyé.
Android attribue à des broadcast receivers un laps de temps (10 seconds) dans lequel la tâche définie doit être terminée.
Si au bout de ce temps le traitement n'est pas terminé, le broadcast receiver est immédiatement disponible à être tué si l'application a besoin de plus de mémoire.
Ils ne sont donc pas adaptés à des tâches dont la durée est indéterminée ou très longue (plus que 10 secondes).
Dans ce cas de figure on utilisera la conjonction du Service à broadcast receiver.
Il réagit aux annonces diffusées par sendBroadcast
Il ne nécessite pas d'interface graphique

2.8. Content Provider (android.content.ContentProvider)

Certaines données d'une application doivent être partagées avec d'autres application.
Par exemple d'un carnet d'adresses qui peut être facilement utilisé par d'autres applications pour pouvoir, par exemple, envoyer un SMS à un de ses contacts.
Sous le principe de partage se cache la notion du Content Provider.
L'accès à ces données se fait via l'instance de la classe *ContentResolver*.
Elle permet d'effectuer toutes les opérations CRUD (create, retreive, update, delete).

- Seul moyen pour partager des données inter-applications
- Techniques : fichier binaire, XML, JSON, base de données embarquée ou distante
- L'interface pour manipuler les données via getContentResolver()
- Données accédées par URI ou format MIME

2.9. L'Interface Graphique Utilisateur (GUI)

- Ressources définies dans le répertoire /res impactent le fichier R.java (à ne pas toucher)
- Types de ressources obtenues par context.getResources() :
  - menu/ : fichiers XML décrivant les menus de l'application
  - anim/ : fichiers XML compilés en objets d'animation
  - raw/ : fichiers bruts (ex : .mp3)
  - color/ : fichiers XML de description des couleurs
  - values-[qualifier]/ : fichiers XML pour différentes ressources (textes, styles, …)
  - drawable-[qualifier]/ : fichiers images ou fichiers XML des objets dessinables
  - xml/ : fichiers XML de de configuration de l'application
  - layout/ : fichiers XML compilés en vues (écrans, fragments, etc …)
- Accès aux ressources :
  - depuis le code : obj.setColor(R.color.rose_bonbon) ;
  - depuis d'autres ressources : <EditText android:textColor='@android:color/rose'/>

2.10.Les Vues (view)

- View :
  - Affecter à un écran : Activity.setContentView()
  - Rechercher : Activity.findViewById() ou View.findViewById()
- ViewGroup = Noeud (LinearLayout, TableLayout, RelativeLayout,FrameLayout, Scrollview)
- Widget = Feuille (Button, EditText, TextView, Spinner, CheckBox)
- Propriétés :
  - Orientation : android:orientation = vertical | horizontal
  - Orientation : android:layout_width / android:layout_height = ??px | fill_parent | wrap_content
  - Orientation : android:layout_weight = ?? (0 par défaut)
  - Espacement : android:padding = top | left | right | bottom / android:margin =??px
  - Gravité : android:layout_gravity = top | left | right | bottom | center_horizontal

2.11.Les Fragments (android.app.Fragment)

- Pour gérer les écrans larges (ex :tablettes)

2.12.Les Activités Dédiées

- ListActivity : si l'activité présente une liste d'items
- ExpandableListActivity : si l'activité présente une liste d'items extensibles
- TabActivity : si l'activité présente des onglets
- FragmentActivity : si l'activité contient des fragments
- MapActivity : si l'activité présente une carte Google

2.13.Les Dialogues et Messages

- Boîte de dialogue (ex :confirmation)
- Barre de progression
- Toast : message compact et rapide à l'intention de l'utilisateur

2.14.Les Menus

- Option Menu activé par la touche Menu du téléphone
- Context Menu activé par un clic long sur un élément
- Popup Menu

2.15.Les barres d'Actions Simples ou Rapides

2.16.Les Thèmes et les Styles

- Un thème correspond à une feuille de style
- Un style est une ressource : System-defined, User-defined (R.style.bleu) ou héritage
- Chaque vue est stylisable par un seul style avec les propriétés

2.17.L'Instanciation des Vues

- Obligatoire pour obtenir les objets
- Récupérer les vues depuis le code - exemple : findViewById(R.id.my_button)
- Récupérer toute une vue depuis le code :
  - dé-sérialiser (inflate) un fichier XML de layout ou menu et le greffer à un sous arbre
  - View my_view = LayoutInflater.inflate(R.layout.main,null) ;
  - MenuInflater.inflate(R.menu.control,my_menu) ;

2.18.Les Ecouteurs (listener)

- Gestion du KeyPAd (tap, trackball) :
  onClick(View), onLongClick(View),onKeyUp(KeyEvent), onKeyDown(KeyEvent)
- Gestion du TouchScreen : onTouchEvent(MotionEvent)

2.19.Les Adaptateurs

- Classes qui lient des données aux vues : android.widget.BaseAdapter
- Récupérer des données d'une collection : SimpleAdapter, ArrayAdapter
- Récupérer des données d'une base de données : CursorAdapter

2.20.Les Notifications

- Exemples : led, son, vibreur, barre de notification

2.21.Les Applications Multi-Ecrans

- Différentes variantes d'une même image :
  /res/drawable-hdip/, drawable-mdip/, drawable-ldip/
- Différentes orientations : /res/layout-port/main.xml, /res/layout-land/main.xml
- Images redimensionnables :
  utiliser les images divisées en 9 zones en utilisant l'outil draw9patch.exe

2.22.Les Application Multi-Langues

- Différentes variantes d'une même chaîne :
  /res/values-fr/strings.xml,/res/values-en/strings.xml
- Choix automatique en fonction de la configuration du terminal (ex :LOCALE=FR_fr)
- S'applique également aux images : /res/drawable-fr/, /res/drawable-en/

2.23.Les Variables globales

- Créer une sous-classe de android.app.Aplication
- Gérer les événements propres à la vie de l'application :
  onCreate(), onTerminate(), onLowMemory(), onConfigurationChanged()
- Se déclare au niveau du nœud <application> du manifeste
- Principe du singleton : une seule instance de la classe pour tout l'application
- Les variables d'instances sont les données à partager

2.24.Persistance, Threading, Partage des Données

- Mécanisme simple et léger :
  - sauvegarde de paires clé/valeur
  - SharedPreferences : pref = getPreferences(Activity.MODE_PRIVATE)
- Sauvegarde des préférences
  - éditeur de préférence : Editor ed = pref.edit() ;
  - stocker les paires : ed.putString('teacher','Pierre Dupont') ;
    ed.putBoolean('actif',true) ;
    valider les modifications : ed.commit() ;
  - récupérer les préférences : String t = pref.getString('teacher', 'unknown ') ;

2.25.Gestion de fichiers

- Flux sortie : FileOutputStream fos = openFileOutput('fichier.dat',Context.MODE_PRIVATE)
- Flux entrée : FileInputStream fis = openFileInput('fichier.dat') ;
- Fichier statique dans res/raw/fichier.txt : openRawResource(R.raw.fichier.txt) ;
- Sérialisation d'objets : ObjectOutputStream oos = ObjectOutputStream(fos) ;
  oos.writeObject(myObject) ;
- Dé-sérialisation d'objets : ObjectInputStream ois = ObjectInputStream(fis) ;
  myObject = ois.readObject() ;
- Utilisation du SGBD embarqué SQLite : db.rawQuery(…..)

2.26.Géolocalisation : (android.location.LocationManager)

- Fournisseur de position : android.location.LocationProvider
- Evénements de position : onLocationChanged(), onProviderDisabled(), onStatusChanged()
- Cartographie :
  com.google.android.maps.MapView, com.google.android.maps.MapController,
  com.google.android.maps.Overlay
- Coordonnées : com.google.android.maps.GeoPoint
- Projection des coordonnées : com.google.android.maps.Projection : toPixels(), fromPixels()

2.27.Les Services Systèmes

- Récupération : getSystemService(ContextXXX)

2.28.Les Alarmes et Timers

- AlarmManager am = (AlarmManager) getSystemService(Context.ALARM_SERVICE) ;
- Gestionnaire d'alarmes : am.set(TYPE , TIME , pendinfIntent) ;
- Exécution des tâches : new Timer().shedule(new TimerTask() { …. } , new Date()) ;

2.29.Linkify et TextWatcher

- Transformer des textes (TextView) en liens cliquables : android.text.util.Linkify
- Contrôler la saisie d'un texte (EditText) : android.text.util.TextWatcher