Formation Kotlin

Ihab ABADI / UTOPIOS

SOMMAIRE

SOMMAIRE PARTIE 1

SOMMAIRE PARTIE 2

1. Présentation de l'environnement Android
2. Structure de projet Android
3. Notion Activité
4. Manifest
5. Les principaux composants
6. Gestion des événements
7. Gestion des ressources
8. Gestion des droits
9. D'autres éléments
9. Programmation asynchrone android
10. Librairies incontournables

Variable muable et mutable

Kotlin permet la déclaration de variabes:
1- Muable en lecture seul avec le mot clé val
2- Mutable en lecture et écriture avec le mot clé var

Typage et cast

- Tout est objet en Kotlin.
- Kotlin possède également une répresentation primitive de certain types (numbers, boolean, characters, strings
- Kotlin possède any et null any
- Kotlin offre plusieurs opérateur pour vérfier le type d’une variable (is, as)

Contrôle de flux (Structures)

- Structure if else
- Structure when
- Structure d’itération for
- Structure d’itération while
- Structure d’itération do while

Contrôle de flux (Retour et continue)

- Return: par défaut renvoie de la fonction englobante la plus proche ou de la fonction anonyme
- Break: termine la boucle englobante la plus proche
- Continue: passe à l'étape suivante de la boucle englobante la plus proche
- Toute expression dans Kotlin peut être marquée avec une étiquette.
- Les étiquettes ont la forme d'un identifiant suivi du signe @, par exemple : abc@, fooBar@.
- Pour étiqueter une expression, ajoutez simplement une étiquette devant elle.

Package et auto-import

- Chaque élément en Kotlin peut être définit dans un package.
- Par défaut, dans Kotlin, dans chaque fichier un nombre de package et auto-importer.
- On peut importer un élément d'un package.
- On peut importer l'ensemble d'un package.

Kotlin et fonctions

- Kotlin permet la programmation fonctionnelles.
- La création d'un Kotlin peut se faire à l'aide du mot clé fun.
- Le Kotlin permet de déclarer des fonctions en une seule expression.
- Le Kotlin permet de déclarer un nombre variable de paramètres.
- Le Kotlin permet de déclarer des fonctions Infix.

Kotlin et fonctions

- Kotlin permet de créer des fonctions de Higher-order.
- Une fonction de Higher-order est une fonction qui accepte comme argument une fonction et qui renvoie une fonction.
- Kotlin permet de déclarer des fonctions anonymes et en une seul ligne avec les expressions lambda.

Kotlin et Objets (Classes et constructeurs)

- Kotlin permet de faire de la POO.
- Kotlin permet de déclarer des classes
- Chaque classe en Kotlin peut avoir un constructeur primaire.
- Un constructeur primaire ne contient pas d'instruction.
- L'initialisation peut se faire à l'aide de la méthode init.
- Chaque classe en Kotlin peut avoir plusieurs constructeurs secondaires.
- Chaque constructeur secondaire doit invoquer le constructeur primaire si déclarer.

Kotlin et Objets (Héritage)

- Kotlin permet de mettre en place de l'héritage.
- Kotlin permet surcharger des méthodes des classes parentes.

Kotlin et Objets (Propriétés)

- Kotlin remplace le mécanisme de méthodes accesseurs par des propriétés.
- Chaque propriété peut définir une partie get et set

Kotlin et Objets (Interface)

- Kotlin permet d'implémenter des interfaces.
- Les méthodes dans les interfaces peuvent avoir une implémentation par défaut.
- L'implémentation multiple est autorisée.
- Dans le cadre d'implémentation multiples, on peut avoir un conflit lors de la surcharge.
- Kotlin permet de résoudre ce conflit.
- Kotlin permet également d'implémenter des interfaces fonctionnelles.

Kotlin et Objets (Visibilité des membres)

- Kotlin propose plusieurs niveau de visibilités.
- Private, protected, internal, et public

Kotlin et Objets (Data class)

- Kotlin permet de créer des classes de types data qui ne contiennent pas de logique métier.
- Chaque data class possède une implémentation des méthodes equals, hashcode, toString

Kotlin et Objets (générique et collections)

- Kotlin permet de mettre en place des génériques.
- Kotlin propose une multitude de collections.
- List, Set, Map
- Chaque collection possède une version muable et mutable.

Opérateur de null safety

- Par défaut tout élément dans kotlin est non null.
- Dans le cadre ou on force à un élément à être nullable, on peut utiliser des opérateurs pour vérifier que l'élément est non null.
- Opérateur elvis
- Opérateur !!

DSL

- Kotlin nous donne les outils pour créer du code en quelque chose qui semble plus naturel à utiliser, via un langage spécifique à un domaine - ou DSL
- Pour créer une DSL, on utilise le pattern du builder.
- Les expressions lambda
- Les fonctions de définition de scope

Programmation asynchrone

- Kotlin possède plusieurs mécanisme pour implémenter une programmation asynchrone.
- Kotlin permet de créer des threads légers à l'aide de coroutine.
- La création de coroutine se fait à l'aide de la méthode launch
- Kotlin offre la possibilité de déclarer des fonctions qui s'exécutent dans une coroutine à l'aide du mot clé suspend
- Kotlin permet de lancer des coroutines également avec le mot clé async

SOMMAIRE PARTIE 2

1. Présentation de l'environnement Android
2. Structure de projet Android
3. Notion Activité
4. Manifest
5. Les principaux composants
6. Gestion des événements
7. Gestion des ressources
8. Gestion des droits
9. D'autres éléments
9. Programmation asynchrone android
10. Librairies incontournables

Présentation de l'environnement Android

- L'environnement minimum pour développer en Android
- SDK Android
  - Kit de développement (débogueur, bibliothèques…)
- Android Studio
  - IDE de développement

Structure projet Android

- Une application Android est composé :
- Un manifeste
- Du code source, Java ou Kotlin
- Des scripts en Gradle

Android Composantes principales

1- Activité.
2- Intention.
3- Service.
4- Manifeste.
5- Context.

Android Manifeste

Le manifeste est utilisé par les outils de compilation, par le système d'exploitation et par Google Play.
Une application contient obligatoirement un et seulement un manifeste. Un manifeste est un fichier XML. Un manifeste contient la liste des activités, services, content provider et broadcastReceiver contenus dans l'application. Un manifeste contient aussi la liste des permissions nécessaires au bon fonctionnement de l'application et les permissions nécessaires aux autres applications désireuses d'accéder aux informations de l'application.
Chaque composant présent dans le manifeste contient des attributs permettant de définir des informations de base telles que la classe représentant le composant, son intitulé…
Les intent filter sont définis dans le manifeste

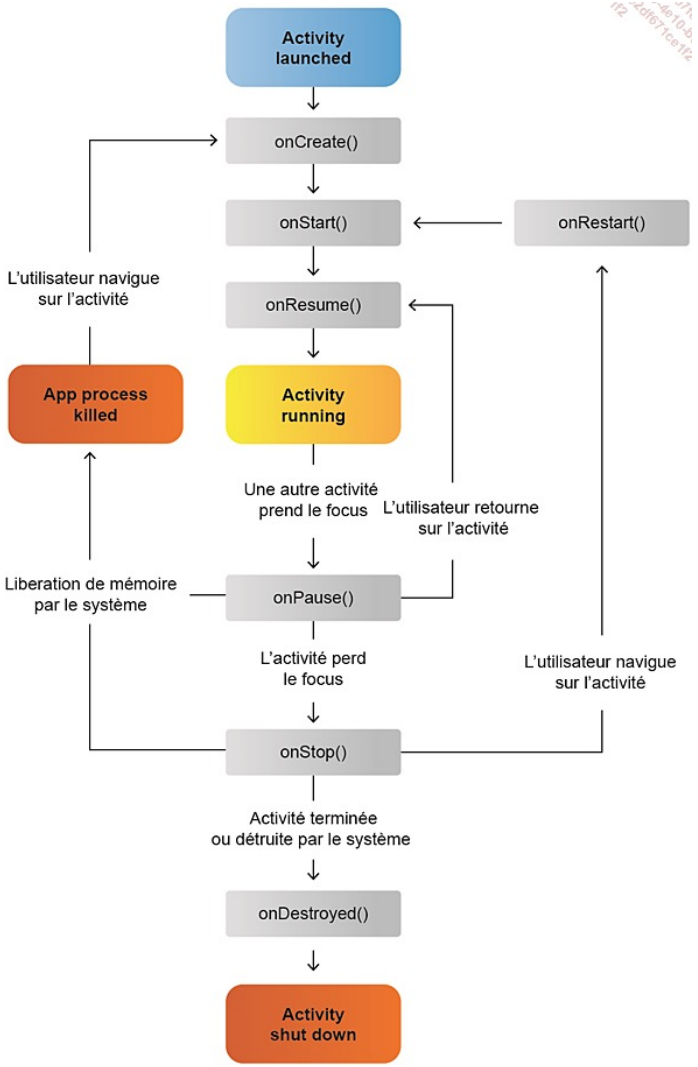
Android Activité

Une activité représente un écran d’une application.
Une application contient autant d’activités qu’il y a d’écrans.
Une seule activité peut être exécutée à la fois.
Une activité a un cycle de vie bien particulier. Plusieurs fonctions nommées "callback " s’exécutent automatiquement à des moments bien spécifiques du cycle de vie

Android Activité

Lorsqu'une activité est créée, elle est empilée dans une pile de gestion appelée le stack.
Cette pile de gestion fonctionne en mode LIFO (Last In First Out).
Une activité est empilée lorsqu'elle démarre et dépilée (détruite) lorsque l'on clique sur le bouton back du téléphone ou bien lorsque la fonction finish est appelée.

Android Activité



Android View et ViewGroup

On peut définir la vue d'une activité de deux façons différentes :
statiquement dans un fichier XML,
dynamiquement via la classe de l'activité.
Tous les éléments de l'interface utilisateur d'une application Android sont construits à l'aide d'objets de types View et ViewGroup.
Un objet de type View est un objet qui dessine quelque chose sur l'écran avec lequel l'utilisateur peut interagir.
Un objet de type ViewGroup est un objet qui contient d'autres objets View afin de définir la disposition de l'interface.
Android fournit tout un panel de View et ViewGroup qui offrent la possibilité de créer des interfaces graphiques riches.
L'interface utilisateur de chaque composant de votre application est définie à l'aide d'une hiérarchie d'objets View et ViewGroup.
Chaque objet de type ViewGroup est un conteneur invisible qui organise des objets enfants de types View et ViewGroup

Android conteneurs ViewGroup

- GridLayout
- LinearLayout
- ConstraintLayout
- ScrollView

Android View

Un objet de type View est un objet qui dessine quelque chose sur l'écran avec lequel l'utilisateur peut interagir
Les principaux:
Button
CheckBox
TextView
EditText
WebView
ImageButton
ImageView

Android Class R

Chaque élément dans Android peut avoir un identifiant unique.
Cet identifiant permet d'accéder à notre objet de n'importe quel endroit de notre application.
Chaque objet est un attribut statique de classe R

Gestion des événements

Gestion des événements via XML
Gestion des événements en Kotlin

Gestion de ressources

Les composants visuels d'une IHM, les fichiers XML, fichiers image, fichiers audio, fichiers vidéo, etc. sont vus comme des ressources.
Chaque ressource a un identifiant.
Il y a deux manières d'accéder à ces identifiants, la manière d'y accéder dépend de l'endroit où l'on se trouve : soit via un fichier Kotlin , soit via un fichier XML.
La manière dont est défini un identifiant dépend du type de ressource :
soit la ressource est contenue à l'intérieur d'un fichier XML, alors c'est au développeur de définir l'identifiant
soit c'est un fichier, alors c'est le nom du fichier qui permet de définir l'identifiant
On stocke les ressources dans le dossier res
On y accède au ressources soit par la classe R, ou en xml à l'aide de @

Gestion de des droits

Pour maintenir la sécurité du système et des utilisateurs, Android exige que les applications demandent l'autorisation avant d'utiliser certaines données et fonctionnalités du système. En fonction de la sensibilité de la zone, le système peut accorder l'autorisation automatiquement ou demander à l'utilisateur d'approuver la demande.
Les permissions nécessaires à une application doivent être obligatoirement définies dans le fichier AndroidManifest.xml grâce à la balise uses-permission. Il y a autant de balises uses-permission que de permissions.
Pour les **permissions sensibles**, il faut ajouter dans le code les instructions demandant une autorisation directe à l'utilisateur. Une fois que la permission est donnée par l'utilisateur, un callback nommé onRequestPermissionsResult est appelé automatiquement.

Intentions

Une intention permet de demander au système d'exploitation Android d'exécuter une tâche spécifique. Il existe deux types d'intentions :
L'intention explicite : cette intention permet d'indiquer précisément quel composant sera exécuté : Activité, Service…
L'intention implicite : cette intention permet d'indiquer précisément quelle action doit être effectuée et alors le système d'exploitation trouve le composant adéquat.
La classe Intent permet de créer la demande. Cette demande doit être ensuite passée en paramètre à une fonction

Autre éléments

ListView
RecyclerView
Toast
AlertDialog