Institut Spécialisé en Nouvelles Technologies de l'Information et de la Communication

Béni Mellal

M202
Programmation KOTLIN

Yassine AFOUDI

yassine.afoudi@ofppt.ma

2024-2025

Rappel:

Découvrir les fondamentaux de Kotlin

CHAPITRE N°2
PRÉPARER L'ENVIRONNEMENT DE DÉVELOPPEMENT

Ce que vous allez apprendre dans ce chapitre :

1. Configuration et première ligne de code
2. Structure d'une application Kotlin
3. Les Fondamentaux de Kotlin

CHAPITRE N°1
PRÉPARER L'ENVIRONNEMENT DE DÉVELOPPEMENT

PLAN

Configuration et première ligne de code

Configuration Kotlin en Android Studio

In your Android studio, Go to Tools -> Kotlin -> Configure Kotlin in Project.

Première ligne de code

Créer un nouveau projet -> Choisir langage Kotlin -> configurer kotlin dans votre projet -> Créer un nouveau fichier Kotlin

```
fun main()
{
    println("Hello World")
}
```

CHAPITRE N°1
PRÉPARER L'ENVIRONNEMENT DE DÉVELOPPEMENT

PLAN

1. Configuration et première ligne de code
2. **Structure d'une application Kotlin**
3. Les Fondamentaux de Kotlin

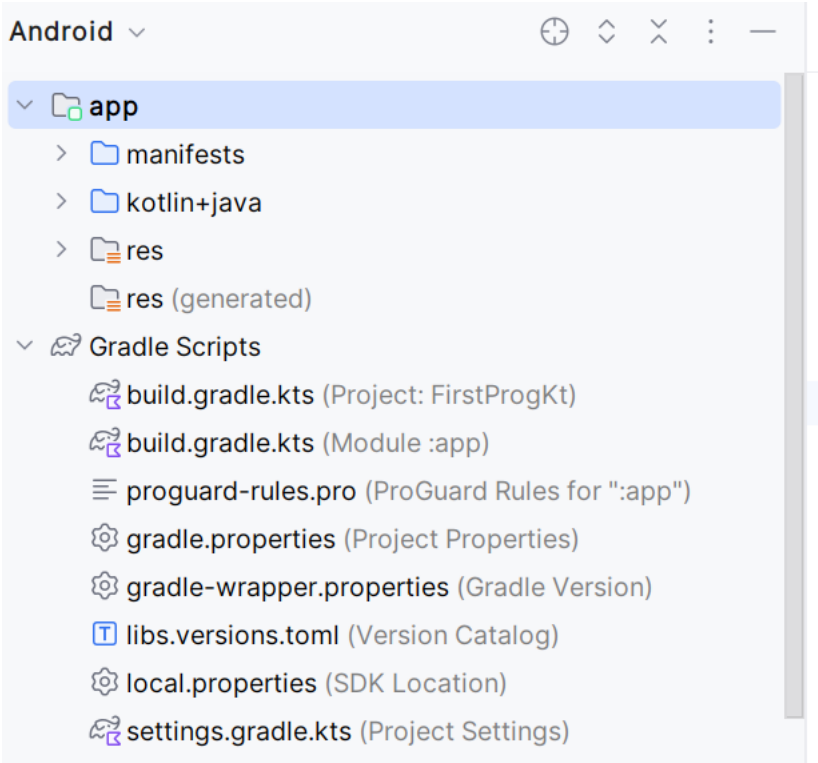
Structure d'une application Kotlin

• Chaque projet dans Android Studio contient un ou plusieurs modules avec des fichiers de code source ou des fichiers ressource.

• Vous pouvez trouvez des modules tels que :

  • Le module d'application d'Android
  • Les modules des librairies
  • Le module Google App Engine

• Par défaut Android Studio affiche vos fichiers de projet dans la vue Android qui se présente comme dans l'image suivante :



Structure d'une application Kotlin

Tous les fichiers de construction du projet sont visibles sous Gradle Script et chaque module d'application contient les fichiers suivants :

1. Le Fichier manifeste : le fichier manifeste (AndroidManifest.xml) contient toutes les informations de votre application telles que :

- Les permissions
- L'icône de l'application
- Le nom de l'application
- Les composants(Activité, service, BroadcastReceiver, ContenProvider) de votre application
- Les méta données etc.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="premierprojet.com.premierprojet">
  <application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name">
    <activity android:name=".MainActivity">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
  </application>
</manifest>
```

Structure d'une application Kotlin

2. Le répertoire Java/Kotlin : ce répertoire contient tous les fichiers de code source de votre application, y compris les codes de teste JUnit.

3. Le répertoire res : ce répertoire contient toutes les ressources de votre application telles que :

- Les fichiers drawable
- Une ou plusieurs chaines de caractère utilisées dans votre application
- Un ou plusieurs fichiers layout
- Les couleurs etc.

Structure d'une application Kotlin

Le répertoire res contient plusieurs autres sous-répertoires tels que :

- **drawable :** ce répertoire contient les images, les fichiers drawable de votre application
- **layout :** ce répertoire contient tous les fichiers layout qui définissent les différentes vues de votre application
- **mipmap :** ce répertoire contient plusieurs icônes utilisées dans l'application telles que l'icône de l'application
- **values :** ce répertoire contient plusieurs fichiers tels que le fichier :
- **colors.xml :** ce fichier contient les couleurs utilisées dans votre application
- **strings.xml :** ce fichier contient les chaines de caractère utilisées dans votre application
- **style.xml :** ce fichier contient tous les style de votre application

Structure d'une application Kotlin

- Android Studio utilise le gradle comme système de construction de votre projet.
- Le gradle utilise plusieurs fichiers pour définir l'ensemble des configurations de construction qui seront appliquées dans le module application de votre projet ou tous les modules de votre projet

CHAPITRE N°1
PRÉPARER L'ENVIRONNEMENT DE DÉVELOPPEMENT

PLAN

1. Configuration et première ligne de code
2. Structure d'une application Kotlin
3. **Les Fondamentaux de Kotlin**

Les Fondamentaux de Kotlin

Syntaxe de Base

Voici un exemple simple d'un programme Kotlin qui affiche "Hello, World!" :

```kotlin
fun main() {
    println("Hello, World!")
}
```

Les Fondamentaux de Kotlin

Variables

Kotlin a deux types de variables :

val : pour les variables immuables (similaire à final en Java).

var : pour les variables mutables.

```
val immutableVariable = 10  // Ne peut pas être modifiée
var mutableVariable = 20    // Peut être modifiée
mutableVariable = 30        // Valide
```

Les Fondamentaux de Kotlin

Types de Données

Kotlin prend en charge les types de données suivants :

- **Nombres** : Int, Double, Float, etc.
- **Booléens** : Boolean (true ou false)
- **Caractères** : Char
- **Chaînes** : String

```kotlin
val age: Int = 25
val name: String = "Alice"
val isStudent: Boolean = false
```

Les Fondamentaux de Kotlin

Fonctions

Les fonctions en Kotlin sont définies à l'aide du mot-clé **fun**. Vous pouvez spécifier le type de retour de la fonction.

```kotlin
fun add(a: Int, b: Int): Int {
    return a + b
}
```

```kotlin
fun greet(name: String) {
    println("Hello, $name!")
}
```

Les Fondamentaux de Kotlin

Contrôle de Flux

- **Conditions**

Kotlin utilise des expressions if et when pour le contrôle de flux.

```kotlin
val number = 10
if (number > 0) {
    println("Positive")
} else {
    println("Negative or Zero")
}

when (number) {
    1 -> println("One")
    2 -> println("Two")
    else -> println("Other")
}
```

Les Fondamentaux de Kotlin

Contrôle de Flux

- **Boucles**

Kotlin prend en charge les boucles for, while et do-while.

```kotlin
for (i in 1..5) {
    println(i)
}

var count = 0
while (count < 5) {
    println(count)
    count++
}
```

Les Fondamentaux de Kotlin

Contrôle de Flux

- **Boucles**

Kotlin prend en charge les boucles for, while et do-while.

```kotlin
fun main() {
    val fruits = listOf("Apple", "Banana", "Cherry")

    for (fruit in fruits) {
        println(fruit)  // Affiche chaque fruit dans la liste
    }
}
```

Les Fondamentaux de Kotlin

Lire une Chaîne de Caractères

Pour lire une chaîne, vous pouvez simplement utiliser readLine().

```
fun main() {
    println("Entrez votre nom :")
    val name = readLine()  // type : String?
    println("Bonjour, $name!")
}
```

Les Fondamentaux de Kotlin

Lire un Entier

Pour lire un entier, vous devez convertir la chaîne lue en Int en utilisant toInt(). Assurez-vous de gérer les exceptions pour éviter des erreurs si l'utilisateur entre une valeur qui ne peut pas être convertie.

```
fun main() {
    println("Entrez un nombre entier :")
    val input = readLine()

    // Utilisation d'un bloc try-catch pour gérer les exceptions
    val number = input?.toIntOrNull() // Convertit la chaîne en Int ou retourne null si éc
    if (number != null) {
        println("Vous avez entré le nombre : $number")
    } else {
        println("Ce n'est pas un nombre entier valide.")
    }
}
```

Les Fondamentaux de Kotlin

Lire un Entier

? : L'opérateur **Safe Call** en Kotlin. Cet opérateur permet d'appeler une méthode ou d'accéder à une propriété sur un objet qui peut être null sans provoquer une NullPointerException.

Les Fondamentaux de Kotlin

Lire un Entier

? : L'opérateur **Safe Call** en Kotlin. Cet opérateur permet d'appeler une méthode ou d'accéder à une propriété sur un objet qui peut être null sans provoquer une NullPointerException.

Les Fondamentaux de Kotlin

fonctions sur String

- **Length :** Renvoie la longueur de la chaîne.

```
val text = "Hello, Kotlin!"
println("Longueur de la chaîne : ${text.length}") // Longueur de la chaîne : 15
```

- **toUpperCase() et toLowerCase() :** Convertit la chaîne en majuscules ou en minuscules.
- **split() :** Divise la chaîne en un tableau de chaînes en fonction d'un délimiteur.
- **reversed()** : Pour inverser une chaîne en Kotlin.
- **substring() :** Récupère une sous-chaîne.

```
val text = "Kotlin Programming"
val subText = text.substring(0, 6)
println(subText) // Kotlin
```

Les Fondamentaux de Kotlin

Exercice 1:

Écrire un programme qui demande à l'utilisateur d'entrer un nombre et affiche si ce nombre est pair ou impair.

Les Fondamentaux de Kotlin

Exercice 1:

```
fun main() {
    print("Entrez un nombre : ")
    val number = readLine()?.toIntOrNull()

    if (number != null) {
        if (number % 2 == 0) {
            println("Le nombre est pair.")
        } else {
            println("Le nombre est impair.")
        }
    } else {
        println("Entrée non valide.")
    }
}
```

Les Fondamentaux de Kotlin

Exercice 2:

Écrire un programme qui demande à l'utilisateur d'entrer 5 nombres et qui calcule la somme de ces nombres.

Les Fondamentaux de Kotlin

Exercice 2:

```
fun main() {
    var sum = 0

    for (i in 1..5) {
        print("Entrez un nombre : ")
        val number = readLine()?.toIntOrNull() ?: 0
        sum += number
    }

    println("La somme des nombres est : $sum")
}
```

Les Fondamentaux de Kotlin

Exercice 3:

Écrire un programme qui demande à l'utilisateur d'entrer un nombre et qui calcule le factoriel de ce nombre.

Les Fondamentaux de Kotlin

Exercice 3:

```
fun main() {
    print("Entrez un nombre : ")
    val number = readLine()?.toIntOrNull() ?: 0

    var factorial = 1

    for (i in 1..number) {
        factorial *= i
    }

    println("Le factoriel de $number est : $factorial")
}
```

Les Fondamentaux de Kotlin

Exercice 4:

Écrire un programme qui demande à l'utilisateur d'entrer trois nombres et qui trouve le plus grand.

Les Fondamentaux de Kotlin

Exercice 4:

```
fun main() {
    print("Entrez le premier nombre : ")
    val num1 = readLine()?.toIntOrNull() ?: 0

    print("Entrez le deuxième nombre : ")
    val num2 = readLine()?.toIntOrNull() ?: 0

    print("Entrez le troisième nombre : ")
    val num3 = readLine()?.toIntOrNull() ?: 0



    val max = maxOf(num1, num2, num3)
    println("Le plus grand nombre est : $max")
}
```

Les Fondamentaux de Kotlin

Exercice 5:

Écrivez un programme qui demande à l'utilisateur d'entrer une chaîne de caractères et détermine si cette chaîne est un palindrome

Les Fondamentaux de Kotlin

Exercice 5:

```
fun main() {
    println("Entrez une chaîne :")
    val input = readLine() ?: ""

    val reversed = input.reversed()
    if (input.equals(reversed, ignoreCase = true)) {
        println("$input est un palindrome.")
    } else {
        println("$input n'est pas un palindrome.")
    }
}
```

Les Fondamentaux de Kotlin

Exercice 6:

Écrivez un programme qui affiche les carrés des nombres de 1 à un nombre donné par l'utilisateur.

Les Fondamentaux de Kotlin

Exercice 6:

```
fun main() {
    println("Entrez un nombre :")
    val limit = readLine()?.toIntOrNull() ?: return

    println("Table des carrés :")
    for (i in 1..limit) {
        println("$i^2 = ${i * i}")
    }
}
```

Les Fondamentaux de Kotlin

Exercice 7:

Écrivez un programme qui demande à l'utilisateur d'entrer des nombres jusqu'à ce qu'il entre un nombre négatif. Le programme doit alors afficher le nombre total de nombres positifs entrés.

Les Fondamentaux de Kotlin

Exercice 7:

```
fun main() {
    var count = 0
    var input: Int? = 0

    do {
        println("Entrez un nombre (négatif pour arrêter) :")
        input = readLine()?.toIntOrNull() ?: continue

        if (input >= 0) {
            count++
        }
    } while (input >= 0)

    println("Vous avez entré $count nombres positifs.")
}
```