Kotlin

Le langage moderne pour développer sur Android

Kotlin

- 100 % compatible avec Java
- Pleinement intégré dans Android Studio
- Autant, voir plus performant que Java
- Similitudes avec les nouveaux langages comme Swift ou TypeScript
- Permet de réduire et clarifier énormément le code par rapport à Java

Historique

- Développé en 2011 par Jetbrains
- 1ère version stabe en 2016
- Google annonce en Mai 2017 adopter le langage pour Android
- Officiellement intégré à Android Studio 3.0 en Octobre 2017
- 2ème langage le plus aimé par les développeurs en 2018
  (selon une enquête de Stack OverFlow)

Kotlin vs Java

Quelques exemples pour vous montrer pourquoi Kotlin va changer votre vie de développeur Android

Data class

```java
public class Movie {

    private int id;
    private String name;
    private Date releaseDate;

    public Movie() {
    }

    public Movie(int id, String name, Date releaseDate) {
        this.id = id;
        this.name = name;
        this.releaseDate = releaseDate;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public Date getReleaseDate() {
        return releaseDate;
    }

    public void setReleaseDate(Date releaseDate) {
        this.releaseDate = releaseDate;
    }

    @Override
    public String toString() {
        return "id :" + id +
                ", name : " + name +
                ", releaseDate: " + releaseDate;
    }
}
```

```kotlin
data class MovieK(
    var id: Int = 0,
    var name: String = "Unknown",
    var releaseDate: Date = Date()
)
```

- **Constructeur avec valeurs par défaut**
- **Getters & Setters automatiquement générés**
- **Méthodes toString() & equals() automatiquement générés (à partir du constructeur)**

Constructeur

```java
Movie movie1 = new Movie();

System.out.println("Movie 1 : " + movie1);
```

```
Movie 1 : id :0, name : null, releaseDate: null
```

```java
Calendar calendar = Calendar.getInstance();
calendar.set(2012, 3, 25);

Movie movie2 = new Movie(2, "Avengers", calendar.getTime());

System.out.println("Movie 2 : " + movie2);
```

```
Movie 2 : id :2, name : Avengers, releaseDate: Wed Apr 25 18:12:52 GMT+02:00 2012
```

```kotlin
val movie1 = MovieK()

println("Movie 1 : $movie1")
```

```
Movie 1 : MovieK(id=0, name=Unknown, releaseDate=Sun Dec 16 17:58:16 GMT+01:00 2018)
```

```kotlin
val calendar = Calendar.getInstance()
calendar.set(2012, 3, 25)

val movie2 = MovieK(2, "Avengers", calendar.time)

println("Movie 2 : $movie2")
```

```
Movie 2 : MovieK(id=2, name=Avengers, releaseDate=Wed Apr 25 17:58:16 GMT+02:00 2012)
```

```kotlin
val movie3 = MovieK(id = 3)

println("Movie 3 : $movie3")
```

```
Movie 3 : MovieK(id=3, name=Unknown, releaseDate=Sun Dec 16 17:58:16 GMT+01:00 2018)
```

Getters & Setters

```java
Movie movie3 = new Movie();
movie3.setId(3);
movie3.setName("Avengers");
movie3.setReleaseDate(calendar.getTime());
```

```kotlin
val movie4: MovieK = MovieK()
movie4.id = 4
movie4.name = "Avengers"
movie4.releaseDate = calendar.time
```

Fonctions de scope

```kotlin
with(movie4) {
    id = 4
    name = "Avengers"
    releaseDate = calendar.time
}
```

```kotlin
movie4.apply {
    id = 4
    name = "Avengers"
    releaseDate = calendar.time
}
```

```kotlin
movie4.run {
    id = 4
    name = "Avengers"
    releaseDate = calendar.time
}
```

```kotlin
movie4.let {
    it.id = 4
    it.name = "Avengers"
    it.releaseDate = calendar.time
}

movie4.let { movie ->
    movie.id = 4
    movie.name = "Avengers"
    movie.releaseDate = calendar.time
}
```

Listes

```java
ArrayList<Movie> movies = new ArrayList<>();
movies.add(movie1);
movies.add(movie2);
movies.add(movie3);

for (Movie movie: movies) {
    System.out.println("Movie : " + movie);
}

int sumOfIds = 0;

for (Movie movie: movies) {
    sumOfIds += movie.getId();
}
System.out.println("sumOfIds : " + sumOfIds);
```

```kotlin
val movies = arrayListOf(movie1, movie2, movie3)

movies.forEach { movie -> println("Movie: $movie") }

println("sumOfIds : ${movies.sumBy { it.id } }")
```

Fonctions

```java
private int getSumOfIds(ArrayList<Movie> movies) {
    int sumOfIds = 0;
    for (Movie movie: movies) {
        sumOfIds += movie.getId();
    }
    return sumOfIds;
}
```

```kotlin
fun getSumOfIds(movies: List<MovieK>) = movies.sumBy { it.id }

fun getSumOfIds2(movies: List<MovieK>): Int = movies.sumBy { it.id }

fun getSumOfIds3(movies: List<MovieK>): Int {
    return movies.sumBy { it.id }
}
```

```java
private Movie getMovieByName(ArrayList<Movie> movies, String name) {
    Movie movieToReturn = null;
    for (Movie movie: movies) {
        if (movie.getName() == name) {
            movieToReturn = movie;
        }
    }
    return movieToReturn;
}
```

```kotlin
fun getMovieByName(movies: List<MovieK>, name: String): MovieK? =
    movies.firstOrNull { it.name == name }
```

Null Safety

```
System.out.println("getMovieByName : " + getMovieByName(movies, "Avengers").getName());

System.out.println("getMovieByName : " + getMovieByName(movies, "Black Panther").getName());
```

```
getMovieByName : Avengers

FATAL EXCEPTION : NullPointerException
```

```
println("getMovieByName: ${getMovieByName(movies, "Avengers")?.name }")

println("getMovieByName: ${getMovieByName(movies, "Black Panther")?.name }")

println("getMovieByName: ${getMovieByName(movies, "Black Panther")?.name ?: "Avengers"}")

println("getMovieByName: ${getMovieByName(movies, "Black Panther")!!.name }")
```

```
getMovieByName: Avengers

getMovieByName: null

getMovieByName: Avengers

FATAL EXCEPTION : KotlinNullPointerException
```

When vs Switch

```java
private String getMonthName(int month){
    String monthName;
    switch (month){
        case 1:
            monthName = "Janvier";
            break;
        case 2:
            monthName = "Février";
            break;
        case 3:
            monthName = "Mars";
            break;
        case 4:
            monthName = "Avril";
            break;
        case 5:
            monthName = "Mai";
            break;
        case 6:
            monthName = "Juin";
            break;
        case 7:
            monthName = "Juillet";
            break;
        case 8:
            monthName = "Aout";
            break;
        case 9:
            monthName = "Septembre";
            break;
        case 10:
            monthName = "Octobre";
            break;
        case 11:
            monthName = "Novembre";
            break;
        case 12:
            monthName = "Décembre";
            break;
        default:
            monthName = "Erreur";
            break;
    }
    return monthName;
}
```

```kotlin
fun getMonthName(month: Int) = when(month) {
    1 -> "Janvier"
    2 -> "Février"
    3 -> "Mars"
    4 -> "Avril"
    5 -> "Mai"
    6 -> "Juin"
    7 -> "Juillet"
    8 -> "Aout"
    9 -> "Septembre"
    10 -> "Novembre"
    11 -> "Décembre"
    12 -> "Janvier"
    else -> "Erreur"
}
```

Kotlin: Extensions

```
fun Date.dateToString(): String {
    val formatter = SimpleDateFormat("EEEE d MMMM yyyy", Locale.getDefault())
    return "le ${formatter.format(this)}"
}

println(movie4.releaseDate.dateToString())

println(movie4.releaseDate.dateToString().capitalize())
```

```
le mercredi 25 avril 2012
Le mercredi 25 avril 2012
```

FindViewById

```java
Button actionButton = findViewById(R.id.actionButton);
```

```kotlin
val actionButton = findViewById<Button>(R.id.actionButton)
```

```kotlin
import kotlinx.android.synthetic.main.activity_main.*

actionButton
```

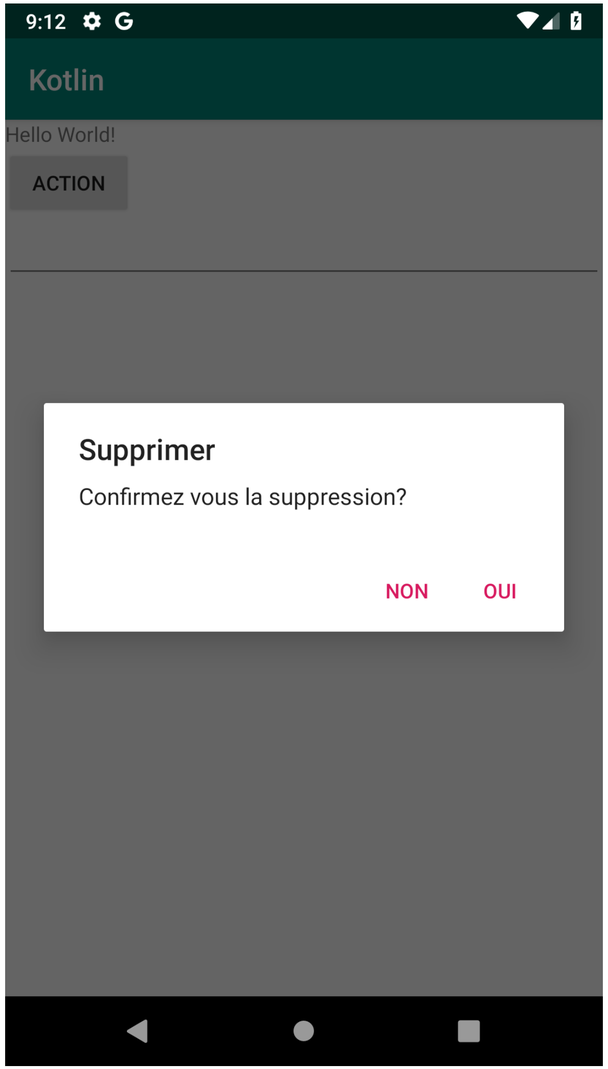
Anko

La puissance des extensions

Anko

- Librairie créée par les développeurs de Kotlin
- Rend le développement plus simple et rapide, plus propre et facile à lire
- Composé de 4 parties :
  - Commons : Librairie remplie de toute sorte d'extensions
  - Layouts : Librairie pour écrire des vues dynamiques en ~~XML~~ Kotlin
  - SQLite : Surcouche pour simplifier l'intégration de SQLite
  - Coroutines : Utilitaires basés sur les coroutines de Kotlin

Popup


9:12
Kotlin
Hello World!
ACTION
Supprimer
Confirmez vous la suppression?
NON
OUI

Popup

```java
AlertDialog.Builder builder = new AlertDialog.Builder(this);
builder.setTitle("Supprimer");
builder.setMessage("Confirmez vous la suppression?");
builder.setPositiveButton("Oui", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {
        //delete something
    }
});
builder.setNegativeButton("Non", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog, int which) {

    }
});
AlertDialog dialog = builder.create();
dialog.show();
```

Java

```kotlin
val dialog = AlertDialog.Builder(this).apply {
    setTitle("Supprimer")
    setMessage("Confirmez vous la suppression?")
    setPositiveButton("Oui") { dialog, which -> /*delete something */ }
    setNegativeButton("Non") { _, _ -> }
}.create()
dialog.show()
```

Kotlin (apply)

```kotlin
val builder = AlertDialog.Builder(this)
builder.setTitle("Supprimer")
builder.setMessage("Confirmez vous la suppression?")
builder.setPositiveButton("Oui") { dialog, which ->
    //delete something
}
builder.setNegativeButton("Non") { _, _ -> }
val dialog = builder.create()
dialog.show()
```

Kotlin

```kotlin
alert(title = "Supprimer", message = "Confirmez vous la suppression?") {
    yesButton { /*delete something */ }
    noButton { }
}.show()
```

Anko

Click listener

```java
Button actionButton = findViewById(R.id.actionButton);
actionButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        launchNextActivity();
    }
});
```

Java

```kotlin
actionButton.setOnClickListener { launchNextActivity() }
```

Kotlin

```kotlin
actionButton.onClick { launchNextActivity() }
```

Anko

Start activity

```java
Intent intent = new Intent(MainActivity.this, ActivityDetail.class);
intent.putExtra("id", 1);
startActivity(intent);
```

Java

```kotlin
val intent = Intent(this@MainActivityK, ActivityDetail::class.java)
intent.putExtra("id", 1)
startActivity(intent)
```

Kotlin

```kotlin
startActivity<ActivityDetail>("id" to 1, "id2" to 2)
```

Anko

Text Watcher

```java
EditText nameEditText = findViewById(R.id.nameEditText);
nameEditText.addTextChangedListener(new TextWatcher() {
    @Override
    public void beforeTextChanged(CharSequence s, int start, int count, int after) {

    }

    @Override
    public void onTextChanged(CharSequence charSequence, int start, int before, int count) {
        name = charSequence.toString();
    }

    @Override
    public void afterTextChanged(Editable s) {

    }
});
```

Java

```kotlin
nameEditText.addTextChangedListener(object : TextWatcher {
    override fun beforeTextChanged(s: CharSequence, start: Int, count: Int, after: Int) { }

    override fun onTextChanged(charSequence: CharSequence, start: Int, before: Int, count: Int) {
        name = charSequence.trim().toString().capitalize()
    }

    override fun afterTextChanged(s: Editable) { }
})
```

Kotlin

```kotlin
nameEditText.textChangedListener {
    onTextChanged { charSequence, _, _, _ ->
        name = charSequence?.trim().toString().capitalize()
    }
}
```

Anko

Liens

- Wikipedia Kotlin
- Documentation Kotlin
- Explications détaillées des fonctions de scope
- Github projet de démonstration Kotlin vs Java
- Github Anko