Rakamin Academy

schoters

Virtual Internship Experience

Kotlin Fundamental

- Mengenal apa itu bahasa pemrograman Kotlin
- Memanfaatkan Data dan View Binding pada Kotlin

1. Mengenal apa itu bahasa pemrograman Kotlin

Kotlin adalah bahasa pemrograman modern namun sudah matang yang bertujuan untuk membuat pengembang lebih bahagia. Ringkas, aman, dapat dioperasikan dengan Java dan bahasa lain, dan menyediakan banyak cara untuk menggunakan kembali kode di antara berbagai platform untuk pemrograman yang produktif.

Kotlin adalah sebuah bahasa pemrograman dengan pengetikan statis yang berjalan pada Mesin Virtual Java ataupun menggunakan kompiler LLVM yang dapat pula dikompilasikan kedalam bentuk kode sumber JavaScript. Pengembang utamanya berasal dari tim programer dari JetBrains yang bermarkas di Rusia. Meskipun sintaksisnya tidak kompatibel dengan bahasa Java, Kotlin didesain untuk dapat bekerja sama dengan kode bahasa Java dan bergantung kepada kode bahasa Java dari Kelas Pustaka Java yang ada, seperti berbagai framework Java yang ada. Tim Pengembang memutuskan menamakannya Kotlin dengan mengambil nama dari sebuah pulau di Rusia, sebagaimana Java yang mengambil nama dari pulau Jawa di Indonesia. Setelah Google mengumumkan bahwa Kotlin menjadi bahasa kelas satu bagi Android, maka bersama Java dan C++, Kotlin menjadi bahasa resmi untuk pengembangan aplikasi-aplikasi Android

Versi Kotlin

Kotlin versi 1.0 dirilis pada 15 Februari 2016 [5] Versi ini secara resmi ditetapkan sebagai versi rilis stabil pertama dan JetBrains telah menetapkan dukungan versi sebelumnya untuk jangka panjangan dengan versi ini.

Pada Google I/O 2017, Google mengumumkan dukungan kelas pertama untuk Kotlin pada Android.

Kotlin versi 1.2 dirilis pada 28 November 2017.[6] Fitur berbagi kode antara JVM dan platform Javascript baru ditambahkan pada versi rilis ini.

Kotlin versi 1.3 dirilis pada 29 Oktober 2018, membawa coroutines pada pemrograman asynchronous

Pendekatan Android yang mengutamakan Kotlin

Di Google I/O 2019, kami telah mengumumkan bahwa pengembangan Android akan semakin mengutamakan Kotlin, dan kami telah mempertahankan komitmen tersebut. Kotlin adalah bahasa pemrograman ekspresif dan ringkas yang dapat mengurangi error kode umum, serta mudah diintegrasikan ke dalam aplikasi yang sudah ada. Jika Anda ingin mem-build aplikasi Android, sebaiknya mulai dengan Kotlin untuk memanfaatkan fitur-fiturnya yang terbaik di kelasnya.

Sebagai upaya untuk mendukung pengembangan Android menggunakan Kotlin, kami bersama-sama mendirikan Kotlin Foundation dan melakukan investasi yang berkelanjutan dalam meningkatkan performa compiler dan kecepatan build. Untuk mempelajari lebih lanjut komitmen Android dalam mengutamakan Kotlin, lihat Komitmen Android terhadap Kotlin.

Mengapa pengembangan Android lebih mengutamakan Kotlin?

Kami telah meninjau masukan yang berasal dari developer secara langsung di konferensi, Dewan Penasihat Pelanggan (CAB), Pakar Developer Google (GDE), dan melalui riset developer kami. Banyak developer yang lebih suka menggunakan Kotlin, dan permintaan untuk dukungan Kotlin lainnya menjadi suatu hal yang jelas. Berikut adalah hal-hal yang disukai developer terkait kesan mereka saat menulis dalam Kotlin:

- Ekspresif dan ringkas: Anda dapat melakukan lebih banyak hal dengan lebih mudah. Tuangkan ide Anda dan kurangi jumlah kode boilerplate. 67% developer profesional yang menggunakan Kotlin mengatakan bahwa Kotlin telah meningkatkan produktivitas mereka.
- Kode yang lebih aman: Kotlin memiliki berbagai fitur bahasa untuk membantu Anda menghindari kesalahan pemrograman yang umum

seperti pengecualian pointer null. Aplikasi Android yang berisi kode Kotlin memiliki kemungkinan error 20% lebih rendah.

- Interoperabilitas: Panggil kode berbasis Java dari Kotlin, atau panggil Kotlin dari kode berbasis Java. Kotlin memiliki interoperabilitas 100% dengan bahasa pemrograman Java, yang berarti Anda dapat menggunakan sesedikit atau sebanyak mungkin Kotlin dalam project Anda sesuai keinginan.
- Serentak dan Terstruktur: Coroutine Kotlin menjadikan kode asinkron berfungsi dengan mudah sebagai kode pemblokir. Coroutine menyederhanakan pengelolaan tugas latar belakang secara dramatis untuk semua hal, mulai dari panggilan jaringan hingga mengakses data lokal.

Kasus penggunaan Multiplatform Kotlin

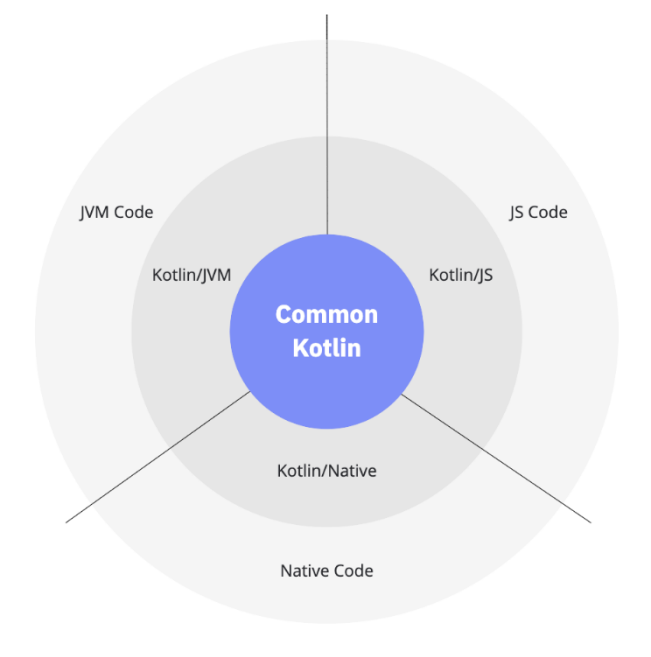
Aplikasi Android dan iOS

Berbagi kode antar platform seluler adalah salah satu kasus penggunaan Kotlin Multiplatform utama. Dengan Kotlin Multiplatform Mobile, Anda dapat membangun aplikasi seluler lintas platform dan berbagi kode umum antara Android dan iOS, seperti logika bisnis, konektivitas, dan banyak lagi.

Aplikasi web full-stack

Skenario lain ketika berbagi kode dapat membawa manfaat adalah aplikasi yang terhubung di mana logika dapat digunakan kembali di server dan sisi klien yang berjalan di browser. Hal ini juga diungkapkan oleh Kotlin Multiplatform.

Cara kerja Kotlin Multiplatform



Gambar 1. How kotlin works

- **Common Kotlin** mencakup bahasa, pustaka inti, dan alat dasar. Kode yang ditulis bersama Kotlin bekerja di mana-mana di semua platform.

- Dengan pustaka Kotlin Multiplatform, Anda dapat menggunakan kembali logika multiplatform dalam kode umum dan khusus platform. Kode umum dapat mengandalkan satu set pustaka yang mencakup tugas sehari-hari seperti HTTP, serialisasi, dan mengelola coroutines.

- Untuk interop dengan platform, gunakan versi Kotlin khusus platform. Versi khusus platform Kotlin (Kotlin/JVM, Kotlin/JS, Kotlin/Native) menyertakan ekstensi ke bahasa Kotlin, serta pustaka dan alat khusus platform.

- Melalui platform ini Anda dapat mengakses kode asli platform (JVM, JS, dan Native) dan memanfaatkan semua kemampuan asli.

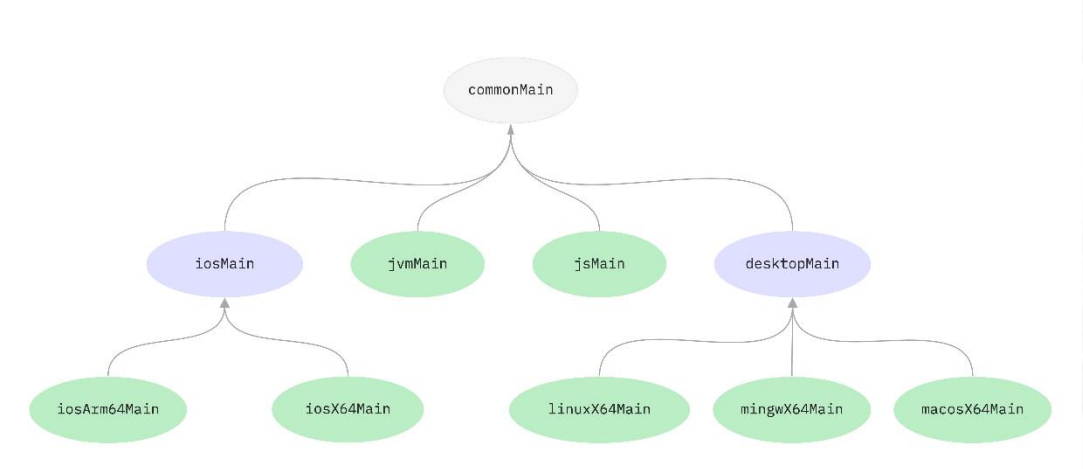
Berbagi kode antar platform

Dengan Kotlin Multiplatform, habiskan lebih sedikit waktu untuk menulis dan memelihara kode yang sama untuk platform yang berbeda - cukup bagikan menggunakan mekanisme yang disediakan Kotlin:

Bagikan kode di antara semua platform yang digunakan dalam proyek Anda. Gunakan untuk berbagi logika bisnis umum yang berlaku untuk semua platform.

Jika Anda perlu mengakses API khusus platform dari kode bersama, gunakan mekanisme Kotlin dari deklarasi yang diharapkan dan aktual.

Bagikan kode di antara beberapa platform yang disertakan dalam proyek Anda tetapi tidak semua. Lakukan ini ketika Anda dapat menggunakan kembali banyak kode di platform serupa:



Gambar 2. Sharing code multiplatform

Sintaks dasar

Definisi dan impor paket
Spesifikasi paket harus berada di bagian atas file sumber.

```
package my.demo
```

```
import kotlin.text.*
```

Titik masuk program
Titik masuk aplikasi Kotlin adalah fungsinya.main

```
fun main() {
    println("Hello world!")
}
```

Cetak ke output standar
print mencetak argumennya ke output standar.

```
print("Hello ")
print("world!")
```

println mencetak argumennya dan menambahkan jeda baris, sehingga hal berikutnya yang anda cetak muncul di baris berikutnya.

```
println("Hello world!")

println(42)
```

Fungsi
Fungsi dengan dua parameter dan tipe return.IntInt

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

Variabel
Variabel lokal baca-saja didefinisikan menggunakan kata kunci . Mereka dapat diberi nilai hanya sekali. val

```
val a: Int = 1  // immediate assignment
val b = 2   // `Int` type is inferred
val c: Int  // Type required when no initializer is provided
c = 3       // deferred assignment
```

2. Memanfaatkan Data dan View Binding pada Kotlin

Apa Itu View Binding?

View Binding adalah fitur yang dapat membantu kita agar lebih mudah untuk berinteraksi dengan View yang ada pada Layout XML kita (contohnya findViewById, Kotlin Synthetic, dan lain-lain).

Cara Menggunakan View Binding

Hal pertama yang harus dilakukan adalah mengecek versi Android Studio yang dimiliki teman-teman. Apakah sudah versi terbaru Android Studio sekarang? Minimal Android Studio 3.6 dan Android Gradle Plugin 3.6 atau di atasnya. Apabila sudah, sila ikuti langkah di bawah ini.

1. Setting enable View Binding di dalam build.gradle
   Untuk dapat menggunakan View Binding, cukup setting enable View Binding pada build.gradle. Teman-teman otomatis dapat menggunakan View Binding. Dengan ini, setiap ada layout XML, otomatis akan di-generate oleh Android Studio menjadi file binding.

```
android {
 ...
 viewBinding {
  enabled = true
 }
}
```

2. Menggunakan View Binding pada Activity
   Pada Activity, kita harus membuat variable binding terlebih dahulu. Untuk membuat variable binding, cukup panggil nama file binding yang berformat nama layout xml misalkan activity_main.xml, maka menjadi MainActivityBinding (nama dibalik dan tambahkan binding di belakangnya).

```
// Membuat variable binding
private lateinit var binding: MainActivityBinding

override fun onCreate(savedInstanceState: Bundle) {
    super.onCreate(savedInstanceState)
    binding = MainActivityBinding.inflate(layoutInflater)
    val view = binding.root
    setContentView(view)
}
```

Gambar 3. Membuat variable binding

3. Mengakses View dengan View Binding
   Setelah kita sudah membuat variable dan mengisinya dengan binding layout yang kita pilih, maka kita sudah bisa mengakses semua View yang ada di dalam layout tersebut. Kita juga dapat langsung menggunakannya, dengan cara seperti ini.

```
binding.name.text = "Bukan kaleng-kaleng"
binding.button.setOnClickListener { viewModel.userClicked() }
```

Gambar 4. Akses View

Mencegah View Binding Generate XML yang Tidak Kita Inginkan

Mungkin teman-teman tidak ingin meng-generate XML tertentu karena alasan suatu hal. Nah, kita bisa menambahkan kode berikut ini.

```
tools:viewBindingIgnore="true"
```

Tambahkan pada root view layout; contohnya seperti ini.

```
<LinearLayout
    ...
    tools:viewBindingIgnore="true" >
  ...
</LinearLayout>
```

Maka, layout XML teman-teman tidak akan di-generate oleh Android Studio dan untuk layout tersebut teman-teman tidak bisa menggunakan View Binding.

Daftar Pustaka

https://kotlinlang.org/docs/getting-started.html#create-your-powerful-application-with-kotlin
https://id.wikipedia.org/wiki/Kotlin_(bahasa_pemrograman)
https://developer.android.com/kotlin/first
https://medium.com/gits-apps-insight/view-binding-membuat-aplikasi-android-menjadi-paten-bukan-kaleng-kaleng-7dca4b5f4739