Kotlin

Ogólne

Wspierany przez *Google*, jako język aplikacji na Androida. Obsługuje biblioteki Java'y. Kompiluje się do bytecode'u Javy (*.class) i może być uruchamiany na JVM (czyli po kompilacji niczym nie różni się od skompilowanego projektu w Java). Bezpośrednią alternatywą / konkurencją dla Kotlina jest Scala.

Frameworks: Ktor, IntelliJ, Spring;

Kotlin w Gentoo

Pakietu nie ma w oficjalnym drzewie *Gentoo*. Dlatego należy podłączyć zewnętrzne repozytorium *Spark-Overlay*.

Warunki wstępne:

- *dev-vcs/git;*
- *app-eselect/eselect-repository.*

Następnie podłączamy repozytorium:

\# **eselect repository enable spark-overlay**

\# **nano** */etc/portage/repos.conf/eselect-repo.conf*

```
[spark-overlay]
location = /var/db/repos/spark-overlay
sync-type = git
sync-uri = https://github.com/6-6-6/spark-overlay.git
```

Instalacja:

\# **emerge --sync spark-overlay**

\# **emerge kotlin-bin**

Kompilacja

Kompilacja kodu:

$ **kotlinc** *plik.kt*

Utworzony zostanie plik "*PlikKt.class*" (zawierający kod byte'owy, pośredni), który można uruchomić:

$ **kotlin** *PlikKt*

Jeśli kod nie używał *Kotlin Standard Library*, to można plik uruchomić także za pomocą:

$ **java** *PlikKt*

Aby skompilować wersję binarną uruchomieniową:

$ **kotlinc** *plik.kt* **-include-runtime -d** *plik.jar*

Aby zbudować wersję binarną, ale nieuruchomieniową (czyli zbudować bibliotekę):
$ **kotlinc** *plik.kt* **-d** *plik.jar*

Uruchamianie:
$ **java -jar** *plik.jar*

Kotlin Shell:
$ **kotlinc-jvm**

Programowanie

var *nazwaZmiennej* - zmienna, która może być przypisywana wiele razy; to, co zwykle rozumiemy pod pojęciem zmiennej;
val *nazwaZmiennej* - stała, może być przypisana tylko raz (jest tylko do odczytu); nie może być nadpisana, jednak jej atrybuty - już tak;

Przykład funkcji:
```
fun main() {
    println("To mój pierwszy program w Kotlin!")
}
```

Przekazanie argumentów do funkcji:
```
@Composable
fun WszystkieElementy() {
    Column {
        Text("Przekazywanie argumentów")
        var a = 2
        var b = 3
        Oblicz(a, b)
    }
}

@Composable
fun Oblicz(a: Int, b: Int) { // W Kotlin parametry zamieniają przekazane zmienne na stałe "val"
    // Poniższe nie zadziała, gdyż "a" jest teraz stałą "val"
    // a = 9
    // Możemy jednak ponownie zamienić "a" na zmienną „var”:
    var a = 9
    var wynik = a + b
    Text("Wynik: " + wynik)
}
```

Funkcje asynchroniczne:
```
suspend fun jakaśFunkcja() {
// Ta funkcja nie może być kompozytowa.
// Taka funkcja może zatrzymać się i poczekać na wyniki innej funkcji i dopiero wtedy wznowić
// swoje działanie. Może być wywołana tylko przez inną funcję typu „suspend” lub będąc w
```

// *coroutine*. Na przykład:
instrukcja1
instrukcja2 // Wymaga długich obliczeń lub czeka na wykonanie innej zewnętrznej funkcji
instrukcja3 // Nie zostanie wykonana dopóki instrukcja2 nie ukończy działania
}

Funkcje typu „suspend” wywołujemy za pomocą launch{ nazwaFunkcji()}.

W funkcjach asynchronicznych często używamy instrukcji wstrzymujących. W Kotlinie istnieją dwie funkcje sleep(), przy czym *TimeUnit* uważana jest za lepsze rozwiązanie:

// 120 sekund
Thread.sleep(120000)
TimeUnit.SECONDS.sleep(120)
// 1 minuta
Thread.sleep(60000)
TimeUnit.MINUTES.sleep(1)

Lista:
val FRUITS = **listOf(**"apple", "tomato", "banana"**)**

Ostatnia aktualizacja: 25 października 2023.