Kotlin to jest coś

Mike Dunn

Java jest prawdopodobnie najbardziej dojrzałym i sprawdzonym językiem, który nadal jest w powszechnym użyciu i jest mało prawdopodobne, aby zmieniło się to radykalnie w dającej się przewidzieć przyszłości. Aby ułatwić współczesne wyobrażenie o tym, co powinien robić język programowania, niektórzy sprytni ludzie zdecydowali się napisać nowy język, który obsługuje wszystkie rzeczy Javy, a także kilka fajnych nowych, których nauczenie się jest dość łatwe i które są w dużej mierze interoperacyjne. Ktoś taki jak ja, który od lat pracuje nad tą samą ogromną aplikacją na Androida, może zdecydować się na napisanie jednej klasy w Kotlinie bez angażowania się w pełną migrację.

Zadaniem Kotlina jest umożliwienie pisania krótszego, czystszego i nowocześniejszego kodu. Chociaż współczesne oraz zapowiadane wersje Javy rozwiązują wiele problemów, które również usprawnia Kotlin, może on być szczególnie przydatny dla programistów Androida, którzy utknęli gdzieś między Javą 7 a Javą 8.

Spójrzmy na kilka przykładów, takich jak wzorzec konstruktora właściwości dla modeli. Zacznijmy od prostego przykładu, jak może wyglądać taki model w Javie:

```
public class Person {
   private String name;
   private Integer age;
   public String getName() {
      return name;
   }
   public void setName(String name) {
      this.name = name;
   }
   public Integer getAge() {
      return age;
   }
   public void setAge(int age) {
      this.age = age;
   }
}
```

Moglibyśmy utworzyć specjalny konstruktor, który będzie pobierał pewne wartości początkowe:

```
public class Person {
   public Person(String name, Integer age) {
      this.name = name;
      this.age = age;
   }
   ...
}
```

Nieźle, ale prawdopodobnie widzisz, jak kilka dodatkowych właściwości może sprawić, że definicja tej dość prostej klasy będzie bardzo szybko się rozrastać. Rzućmy okiem na tę klasę w Kotlinie:

```
class Person(val name:String, var age:Int)
```

Otóż to! Innym fajnym przykładem jest delegacja. Delegacje w Kotlinie pozwalają utworzyć logikę dla dowolnej liczby operacji odczytu. Jednym z przykładów jest leniwa inicjalizacja, koncepcja z pewnością znana programistom Javy. Może to wyglądać tak:

```
public class SomeClass {
   private SomeHeavyInstance someHeavyInstance = null;
   public SomeHeavyInstance getSomeHeavyInstance() {
      if (someHeavyInstance == null) {
         someHeavyInstance = new SomeHeavyInstance();
      }
      return someHeavyInstance;
   }
}
```

Znów niezbyt to straszne, a nawet zrobione dość prosto i bez konfiguracji, ale są szanse, że będziesz powtarzał ten sam fragment kilka razy w swoim kodzie, naruszając zasadę DRY (*Don't Repeat Yourself* – nie powtarzaj się). Nie jest też bezpieczne w przypadku wątków. Oto wersja w Kotlinie:

```
val someHeavyInstance by lazy {
   return SomeHeavyInstance()
}
```

Krótkie, przyjemne i czytelne. Cały ten nadmiar kodu jest ładnie schowany pod spodem. Aha, i jest też bezpieczne dla wątków. Dużym ulepszeniem jest również bezpieczeństwo związane z `null`. W Kotlinie możesz zobaczyć wiele operatorów znaków zapytania po odwołaniu do wartości, która może przyjmować null:

```
val something = someObject?.someMember?.anotherMember
```