ToiletPaper #2

jambit
WHERE INNOVATION WORKS

Kotlin: Das bessere Java?

Von Michael Hoffmann und Andreas Scharf

✕ Problem

Java ist weltweit eine der am meisten verwendeten Programmiersprachen. Leider ist es keine moderne Programmiersprache, auch wenn Java 8 ein Schritt in die richtige Richtung darstellt. Java-Code ist anfällig für NullPointerExceptions und verlangt dem Programmierer für geringe Aufgabenstellungen viel Schreibarbeit ab (Stichwort: Boilerplate-Code). In bestimmten Android- oder Java-Projekten kann es vorkommen, dass die aktuellste Java-Version noch nicht verwendet wird und man somit auf Alternativen angewiesen ist.

✓ Lösung

Die Entwickler von JetBrains (bekannt durch die IDE IntelliJ) störten sich an den Eigenheiten von Java und erschufen kurzerhand ihre eigene Programmiersprache mit dem Namen „Kotlin“. Der Name stammt von einer kleinen Insel im Finnischen Meerbusen.

Kotlin wurde als Vielzweck-Programmiersprache (General Purpose Language) konzipiert und übernimmt einen großen Teil der Syntax von Java. Folglich ist es objektorientiert, statisch typisiert und bietet Generics. Der Quellcode wird vom Compiler in Java-Bytecode umgewandelt und läuft somit auf jeder Java-Virtual-Machine (JVM). Dadurch ist Kotlin für die Verwendung mit bekannten Frameworks wie Spring oder Hibernate sowie zur Android-Entwicklung geeignet. Die Sprache bietet 100% Interoperabilität mit Java, das heißt, man kann ohne Probleme Java-Code aus Kotlin-Code aufrufen und umgekehrt. Die Sprache steht seit Februar 2012 unter der Apache-2.0-Lizenz.

Parallel zur Sprache hat JetBrains die Kotlin-Unterstützung für IntelliJ entwickelt. Außerdem steht ein Eclipse-Plugin zur Verfügung. Bewährte Build Tools wie Maven, Gradle oder Ant können zur Entwicklung verwendet werden.

➔ Beispiel

Hier ein Vergleich zwischen der Kotlin- und Java-Syntax:

```kotlin
// Kotlin
fun main(args: Array<String>) {
        println("Hello, World!")
}
```

```java
// Java
public class App {
      public static void main(String[] args) {
      System.out.println("Hello, World!");
      }
}
```

Folgende Code-Zeile erstellt ein POJO in Kotlin, inklusive getter, setter, equals(), hashCode(), toString() und copy():

```kotlin
data class User(val name: String, val age: Int)
```

Tony Hoare, der Erfinder der Null-Referenz, bezeichnet seine Erfindung heute als billion-dollar mistake, weil die Programmierung mit null fehleranfällig ist und hohe Kosten verursacht hat. Seit Java 8 gibt es als Lösung die Optional-Klasse, welche man aus anderen Programmiersprachen kennt. Kotlin bietet eine elegantere Lösung: Es wird explizit zwischen optionalen Typen, die null enthalten dürfen, und den normalen Typen mit Pflichtwert unterschieden. Der Compiler meckert, wenn man ungeprüft auf optionale Typen referenziert:

```kotlin
//val o: String = null  // Compilerfehler
val s: String? = null

//s.hashCode()          // Compilerfehler
s?.hashCode()           // Safecall
```

+ Weiterführende Aspekte

- Offizielle Website: https://kotlinlang.org/
- Browser-Playground: http://try.kotlinlang.org/