RoomEx - Erstellung einer Kotlin-Bibliothek für eine vereinfachte Erstellung von Datenbankrelationen in Room

FACHHOCHSCHULE ERFURT UNIVERSITY OF APPLIED SCIENCES
Angewandte Informatik

Betreuer: Professor Dr. Steffen Avemarg und Dr. Uwe Altenburg

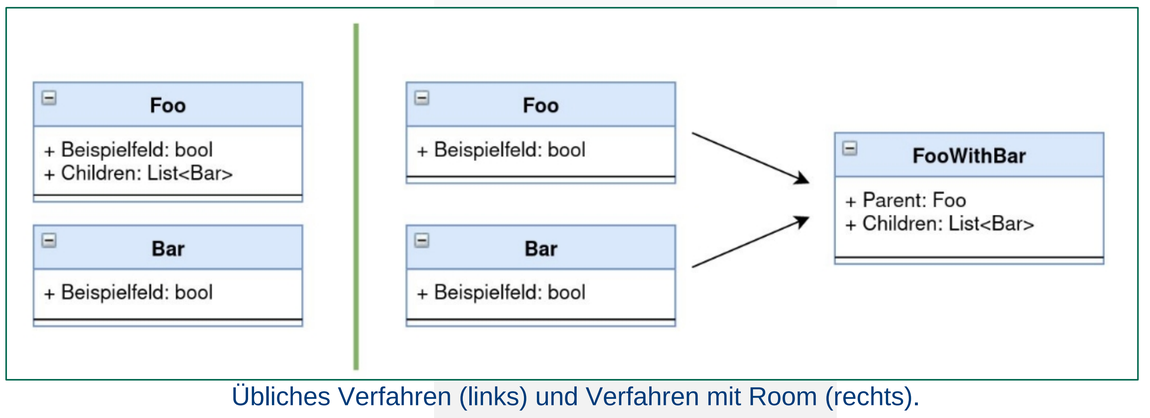
Studiengang Angewandte Informatik, Altonaer Str. 25, 99085 Erfurt, Tel. 0361 6700 642, e-mail: informatik@fh-erfurt.de

Marcel van der Heide

1990 Geboren in Drachten, Niederlande
2004-2008 Realschulabschluss, Heinrich Hertz Regelschule, Ilmenau
2011-2014 Ausbildung Fachinformatik, Berufsschulzentrum, Hermsdorf
2014-2018 Hochschulreife (beruflich qualifiziert), Berghof Systeme, Königsee
2018-2022 Bachelor Angewandte Informatik, Fachhochschule Erfurt

Problematik

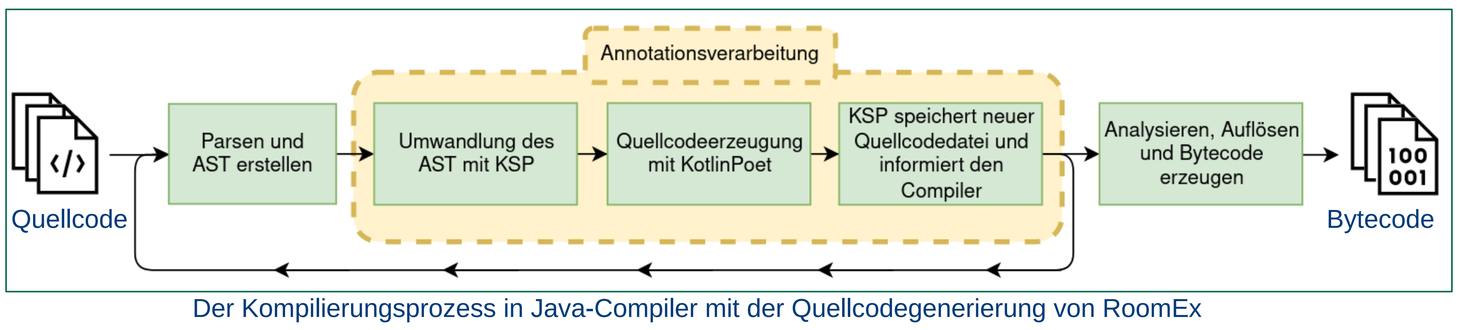
Android Applikationen, die mit Java oder Kotlin erstellt werden, können auf die von Google bereitgestellte ORM-Bibliothek Room zurückgreifen.
Eine Schwierigkeit von Room besteht allerdings in der Erstellung und Verwendung von Relationen zwischen Entitäten, da hier, konträr zu herkömmlichen ORM-Bibliotheken, Relationen über weitere Hilfsobjekte umgesetzt werden müssen.
So reicht es nicht aus, eine Liste des Kind-Objekt-Datentyps im Eltern-Objekt anzulegen, um einfache ***1:n*** Relationen zu definieren, sondern es wird ein zusätzliches Objekt benötigt, welches sowohl das Eltern-Objekt als auch eine Liste von Kind-Objekten beinhaltet. Übergeordnete Programmschichten müssen anschließend nicht nur mit den einfachen Entitäten, sondern zusätzlich mit diesen Behilfsobjekten arbeiten.



Übliches Verfahren (links) und Verfahren mit Room (rechts).

Zielsetzung

Ziel dieser Bachelorarbeit war es, die Fähigkeiten von Googles Room-Bibliothek funktional zu erweitern, um dadurch die Erstellung und Handhabung von Relationen für den Entwickler zu vereinfachen und übersichtlicher zu gestalten.
Der vom Benutzer erstellte Quellcode wird während der Kompilierung automatisch analysiert und bei Bedarf werden neue Quellcodedateien generiert, um diese neuen Funktionen einzubinden.

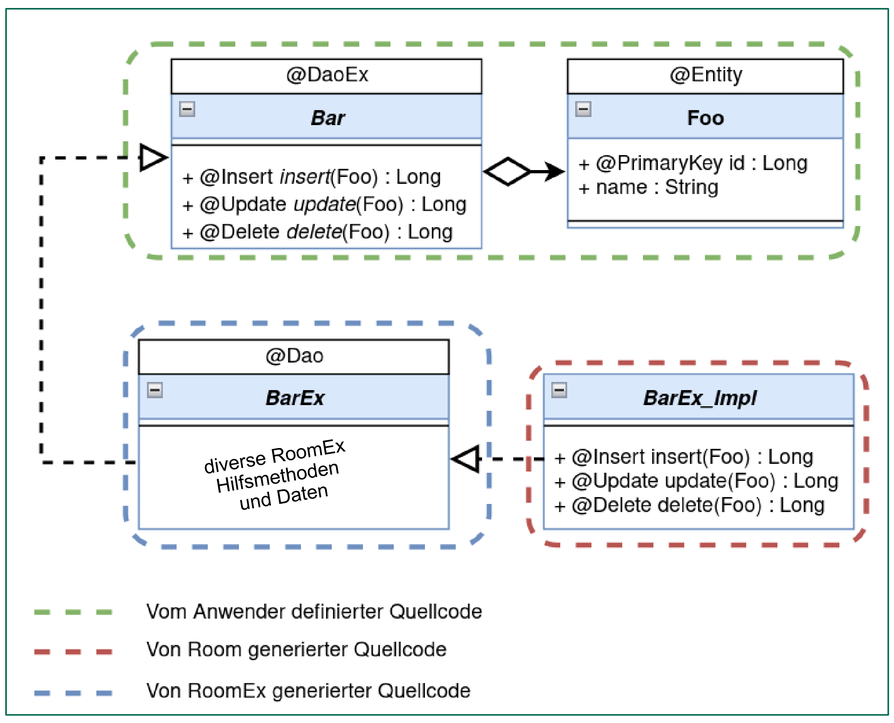


Der Kompilierungsprozess in Java-Compiler mit der Quellcodegenerierung von RoomEx

Umsetzung

Für die Umsetzung wurde besonders darauf geachtet, die Performance von Room möglichst beizubehalten. Um dies zu gewährleisten, musste zum einen weitestgehend auf Reflexion zur Laufzeit verzichtet werden. Zum anderen durften die neuen Funktionen nicht bestehende Room-Funktionen beeinflussen.

Um die Verwendung von Reflexion auf ein absolutes Minimum zu reduzieren, war es notwendig, dass wichtige Informationen, wie Relationsfelder und deren Datentypen, bereits während der Kompilierung analysiert und an den richtigen Stellen eingebettet werden. Um dies zu bewerkstelligen wurden mithilfe von Annotationsprozessoren in dem Kompilierungsprozess des Zielprojektes eingegriffen und entsprechende Änderungen vorgenommen. (s. Bild oben)
Damit die bereits vorhandenen Room-Funktionen nicht eingeschränkt werden, müssen diese Änderungen noch vor der Room-Implementierung eingebettet werden.



Hierfür wurde eine neue Annotation namens @DaoEx geschaffen, welches vom Anwender anstelle von der Room Annotation @Dao verwendet werden soll. RoomEx erzeugt darauf selbständig die @Dao-Klasse, welches von Room im nächsten Zyklus der Annotationsverarbeitung erkannt und verarbeitet wird. (s. Bild links)

Ergebnis

- Dynamische Codegenerierung während der Kompilation
- Neue Annotationen, um einfach Relationen zu erzeugen
  - @OneToOneEx, @OneToManyEx, @ManyToOneEx, @ManyToManyEx
- Bidirektionale Relationen
- Vermeidung von Hilfsobjekt für Relationen
- Kein Performanceverlust
- Neue Methoden um Relationsdaten zu erstellen, abzurufen und zu ändern