目次

初期データを投入する方法

初期データの投入方法を解説するにあたり、以下のようなUserエンティティを使用することにします。

```
@Entity
data class User(
    @PrimaryKey(autoGenerate = true)
    val id: Int,
    var name: String,
    var age: Int
)
```

User.kt hosted with ❤ by GitHub　view raw

初期データの投入は、データベースが作成されたタイミングで行います。

```
@Database(entities = [User::class], version = 1)
abstract class UserDatabase : RoomDatabase() {

    abstract fun userDao(): UserDao

    companion object {

        private var INSTANCE: UserDatabase? = null

        private val lock = Any()

        fun getInstance(context: Context): UserDatabase {
            synchronized(lock) {
                if (INSTANCE == null) {
                    INSTANCE = Room.databaseBuilder(context.applicationContext,
                        UserDatabase::class.java, "User.db")
                        .addCallback(object : RoomDatabase.Callback() {
                            override fun onCreate(db: SupportSQLiteDatabase) {
                                super.onCreate(db)

                                val sql = "INSERT INTO 'user' VALUES " +
                                        "(NULL, '田中', 20)," +
                                        "(NULL, '佐藤', 24)," +
                                        "(NULL, '山口', 30)" +
                                db.execSQL(sql)
                            }
                        })
                        .build()
                }
                return INSTANCE!!
            }
        }
    }
}
```

UserDatabase.kt hosted with ❤ by GitHub　view raw

データベースが作成されるタイミングを知るため、17行目のaddCallback()で、RoomDatabase.Callbackを実装しております。

これにより、データベースが作成されたタイミングで、onCreate()が呼ばれるので、その時にデータの投入を行います。

上記のサンプルでは、データ挿入用のSQL文を作成した後、execSQL()でSQL文を実行することで、初期データの投入を行なっております。