Data Storage: SQLite, Room, ViewModel, LiveData and Observer

Sensor Based Mobile Applications

Patrick Ausderau, Ulla Sederlöf, Jarkko Vuori

Helsinki Metropolia University of Applied Science

2022

Outline

Room SQLite Database

LiveData, ViewModel, Observer

Lab

For more info:

https://developer.android.com/training/data-storage/room/

https://www.sqlite.org/

https:
//developer.android.com/topic/libraries/architecture/livedata

https://developer.android.com/jetpack/guide

https://developer.android.com/jetpack/compose/lists

Room SQLite Database

- Support for creating and using local SQLite databases on the device
  - A database is private to its application
  - Databases are stored on the device (or emulator) in `/data/data/<package>/databases`
- Room provides an abstraction layer over SQLite and consit of 3 major components
  - `@Database` extends `RoomDatabase`, provide access to the database
  - `@Dao` contains the methods used for accessing the database (CRUD operations)
  - `@Entity` represents a table (and relations) within the database

Dependencies

- in project build gradle, add `google()` in the list of repositories (should already be there by default)
- in module app build gradle, add in dependencies:

```
apply plugin: "kotlin-kapt"
dependencies {
  def roomVersion = "2.4.3"
  implementation "androidx.room:room-runtime:$roomVersion"
  annotationProcessor
      "androidx.room:room-compiler:$roomVersion"

  // To use Kotlin annotation processing tool (kapt)
  kapt "androidx.room:room-compiler:$roomVersion"
  // Kotlin Extensions and Coroutines support for Room
  implementation "androidx.room:room-ktx:$roomVersion"
  //...
}
```

@Entity

- data class with @Entity annotation

```
@Entity
data class User(
    @PrimaryKey(autoGenerate = true)
    val uid: Long,
    val firstname: String,
    val lastname: String) {
  //constructor, getter and setter are implicit :)
  override fun toString() = "$firstname $lastname
      ($uid)"
}
```

@Entity with relation

```kotlin
@Entity(foreignKeys = [ForeignKey(
        entity = User::class,
        onDelete = CASCADE,
        parentColumns = ["uid"],
        childColumns = ["user"])])
data class ContactInfo(
    val user: Long,
    val type: String, //e.g. phone, email, fb, twitter,...
    @PrimaryKey
    val value: String)
```

```kotlin
class UserContact {
    @Embedded
    var user: User? = null
    @Relation(parentColumn = "uid", entityColumn = "user")
    var contacts: List<ContactInfo>? = null
}
```

@Dao

```kotlin
@Dao
interface UserDao {
  @Query("SELECT * FROM user")
  fun getAll(): LiveData<List<User>>

  @Query("SELECT * FROM user WHERE user.uid = :userid")
  // the @Relation do the INNER JOIN for you ;)
  fun getUserWithContacts(userid: Long): LiveData<UserContact>

  @Insert(onConflict = OnConflictStrategy.REPLACE)
  suspend fun insert(user: User): Long

  @Update
  suspend fun update(user: User)

  @Delete
  suspend fun delete(user: User)
}

@Dao
interface ContactInfoDao { /* ... */ }
```

@Database

```kotlin
@Database(entities = [(User::class), (ContactInfo::class)],
    version = 1)
abstract class UserDB: RoomDatabase() {
  abstract fun userDao(): UserDao
  abstract fun contactDao(): ContactInfoDao

  companion object{
    private var sInstance: UserDB? = null
    @Synchronized
    fun get(context: Context): UserDB {
      if (sInstance == null) {
        sInstance =
            Room.databaseBuilder(context.applicationContext,
            UserDB::class.java, "users.db").build()
      }
      return sInstance!!
    }
  }
}
```

LiveData and ViewModel

- `LiveData` is an observable data holder class
- LiveData is lifecycle-aware, it will only updates app component observers that are in an active lifecycle state
- ViewModel is a class that is responsible for preparing and managing the data for a Composable function or an Activity or a Fragment.

LiveData and ViewModel

- in app build config, add dependencies:

```
def lifecycle_version = "2.5.1"
// ViewModel
implementation
    "androidx.lifecycle:lifecycle-viewmodel-ktx:$lifecycle_version"
// ViewModel utilities for Compose
implementation
    "androidx.lifecycle:lifecycle-viewmodel-compose:$lifecycle_version'
// LiveData
implementation
    "androidx.lifecycle:lifecycle-livedata-ktx:$lifecycle_version"
implementation
    "androidx.compose.runtime:runtime-livedata:$compose_version"
```

LiveData and ViewModel

```kotlin
class UserViewModel(application: Application):
    AndroidViewModel(application) {
    private val userDB = UserDB.get(application)

    fun getAll(): LiveData<List<User>> =
        userDB.userDao().getAll()

    fun insert(user: User) {
        viewModelScope.launch {
            userDB.userDao().add(user)
        }
    }

    /* fun update, delete, getDetails,... */
}
```

LiveData and ViewModel

```kotlin
class MainActivity : ComponentActivity() {
    companion object{
        private lateinit var userViewModel: UserViewModel
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        userViewModel = UserViewModel(application)
        setContent {
            MainAppNav(userViewModel)
        }
    }
}
```

LiveData and ViewModel

```kotlin
@Composable
fun InsertUser(userViewModel: UserViewModel) {
    // candiate for mutableStateListOf
    var fname by remember { mutableStateOf("") }
    var lname by remember { mutableStateOf("") }
    Column {
        TextField(value = fname, label = {
            Text(stringResource(R.string.fname)) },
            onValueChange = { fname = it })
        TextField(value = lname, label = {
            Text(stringResource(R.string.lname)) },
            onValueChange = { lname = it })
        Button(onClick = {
            userViewModel.insert(User(0, fname, lname))
        }) {
            Text(stringResource(R.string.insert))
        }
    }
}
```

LiveData and ViewModel

```kotlin
@Composable
fun ListUsers(userViewModel: UserViewModel, navController:
    NavController) {
    val userList =
        userViewModel.getAll().observeAsState(listOf())
    LazyColumn {
        item {
            Row {
                Text(stringResource(R.string.header))
            }
        }
        items(userList.value) {
            Text("User: $it", Modifier.clickable {
                navController.navigate("details/${it.uid}")
            })
        }
    }
}
```

LiveData and ViewModel

```kotlin
@Composable
fun MainAppNav(userViewModel: UserViewModel) {
  val navController = rememberNavController()
  NavHost(navController, startDestination = "main") {
    composable("main") {
      Column {
        InsertUser(userViewModel)
        ListUsers(userViewModel, navController)
      }
    }
    composable("details/{userId}") {
      val id = it.arguments?.getString("userId")?.toLong() ?: 0
      DetailView(userViewModel, id, navController)
    }
  }
}
```

Build gradle and details about navigation: https://developer.android.com/jetpack/compose/navigation

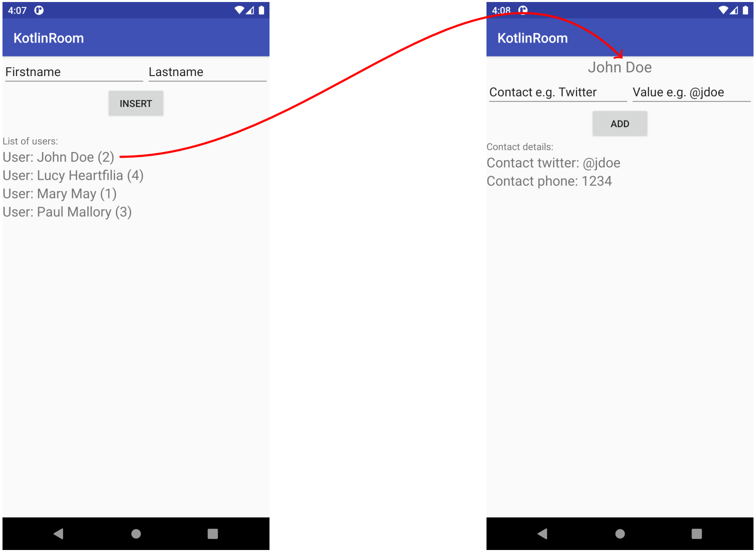
Lab_w1_d5 Room and LiveData

- Create an app that will use a simple Room SQLite database (at least two related tables). E.g.
  - Ice Hockey teams (name/created year) and their players (name/position)
  - Movies (title/year/director) and their actors (name/role)
  - Recipes (name/country of origin) and their ingredients (name/quantity)
  - Animals Families (English/Latin name) and their species (English/Latin name/area)
  - ...
- Create two views to insert data (one for each table) and LazyColumn to list the content (select from the two related tables).

Note: start easy with one table.

Hint: when modifying the database (entities, dao,...), before redeploying your app to emulator/debugging device, go to phone settings ⇒ App & notification, select your app ⇒ Storage & cache and clear both cache and storage (so you avoid database modification exception).

Lab_w1_d5 Room and LiveData


4:07

KotlinRoom

Firstname Lastname

INSERT

List of users:
User: John Doe (2)
User: Lucy Heartfilia (4)
User: Mary May (1)
User: Paul Mallory (3)

4:08

KotlinRoom

John Doe

Contact e.g. Twitter Value e.g. @jdoe

ADD

Contact details:
Contact twitter: @jdoe
Contact phone: 1234