

Programação Para Dispositivos Móveis I

ROOM

2024/_25 CTeSP – Desenvolvimento para a Web e Dispositivos Móveis

Ricardo Barbosa , rmb@estg.ipp.pt

Carlos Aldeias, cfpa@estg.ipp.pt

Índice

- Android Architecture Components;
- Room;
- Leitura Adicional.

Android Architecture Components

Uma coleção de bibliotecas recomendadas que ajudam no desenvolvimento de aplicações robustas

Principais componentes:

- LiveData
- Handling Lifecycles
- ViewModel
- Room

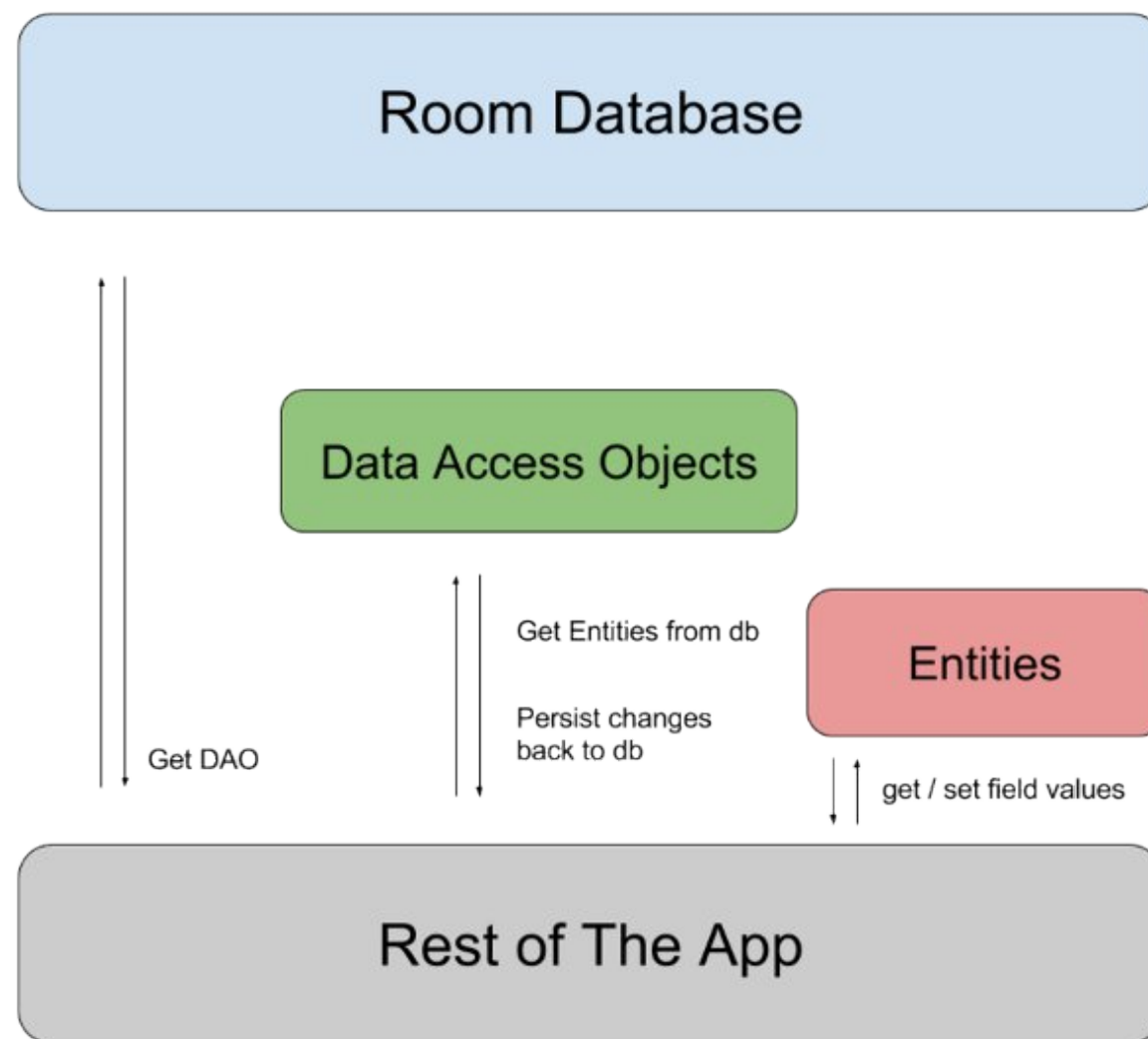
!! Vamos nos focar na biblioteca Room !!



Room

Faz parte dos novos componentes de arquitetura do Android;

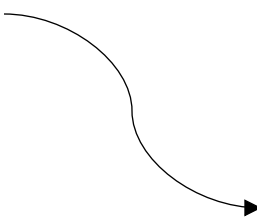
- Suporte para gerir a persistência de dados com recurso a entidades e objetos usando o componente Room;
- Evita desenvolvimentos de classes para conexão à base de dados;
- Converte tabelas em SQLite para objetos JAVA;
- Testa comandos SQLite em tempo de compilação;



Room

Instalação[build.gradle (project)]

Em principio já estão definidos estes repositórios, mas confirmar ajuda sempre



```
allprojects {  
    repositories {  
        google()  
        jcenter()  
    }  
}
```

Room

Instalação[build.gradle (module)]

```
dependencies {  
    ...  
    def room_version = "2.6.1"  
  
    implementation "androidx.room:room-runtime:$room_version"  
    annotationProcessor "androidx.room:room-compiler:$room_version"  
    ...  
}
```

Room

Componentes

Existem 3 componentes principais em ROOM:

- Base de dados – contém a base de dados e serve de ponto de acesso aos dados persistidos;
- Entidades – representa uma tabela na base de dados;
- Data Access Objects (DAO) – contém os métodos para acesso à base de dados.

Room

Entidades

Chave primária composta

Se pretendermos nomes diferentes

```
@Entity(tableName = "user")
public class Person {

    @PrimaryKey
    public int _id;

    @ColumnInfo(name = "person_name")
    public String name;

    public int age;
}
```

```
@Entity(primaryKeys = {"name", "age"})
public class Person {

    public String name;

    public int age;
}
```


Room

Relação entre entidades 1:1

```
@Entity
public class Person {


    @PrimaryKey
    public int _id;

    public String name;

    public int age;
}
```

```
@Entity
public class Playlist{
    @PrimaryKey
    public int _id;

    public int userOwnerId;
}
```



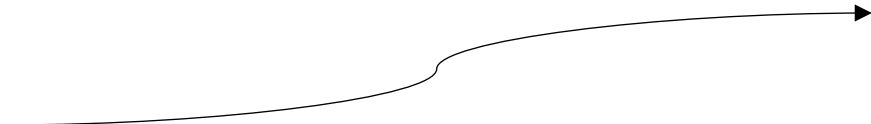
Room

Relação entre entidades

```
@Entity(foreignKeys = @ForeignKey(entity = Person.class,  
                                   parentColumns = "_id",  
                                   childColumns = "personID"))
```

```
public class Book {  
  
    @PrimaryKey  
    public int _bookID;  
  
    public String title;  
  
    public int personID;  
}
```

```
@Entity  
public class Person {  
  
    @PrimaryKey  
    public int _id;  
  
    public String name;  
  
    public int age;  
}
```



Room

Relação entre entidades 1:N

@Entity

```
public class Person {  
  
    @PrimaryKey  
    public int _id;  
  
    public String name;  
  
    public int age;  
}
```

@Entity

```
public class Playlist{  
    @PrimaryKey  
    public int _id;  
  
    public String name;  
  
    public int userOwnerId;  
}
```

```
public class UserWithPlaylist{  
    @Embedded public Person user;  
    @Relation(  
        parentColumn = "_id"  
        entityColumn = "userOwnerId"  
    )  
    public List<Playlist> playlists;  
}
```

Room

Relação entre entidades N:N

```
@Entity
public class Playlist{
    @PrimaryKey
    public int _id;

    public String name;
}
```

```
@Entity
public class Song{
    @PrimaryKey
    public int _id;

    public String name;

    public String artist;
}
```

```
@Entity(primaryKeys = {"playlistId", "songId"})
public class PlaylistSongCrossRef{
    private int playlistId;

    private int songId;
}
```

Room

Objetos aninhados

```
class Address{
    public String street;
    public String state;
    public String city;

    @ColumnInfo(name = "postal_code")
    public int postalCode;
}

@Entity
public class Person {

    @PrimaryKey
    public int _id;

    public String name;

    public int age;

    @Embedded
    public Address address;
}
```

Room

DAO

DAO (Data Access Objects) são usados para aceder a dados persistidos na base de dados como objetos JAVA.

- Acedendo à base de dados através de DAO permite separar diferentes componentes da arquitetura da base de dados;
- DAO pode ser uma interface ou uma classe abstrata;
- A implementação das interface é feita pela biblioteca DAO;

Room

DAO -> Possíveis interfaces

Inserção de dados

@Dao

```
public interface PersonDAO {
```

```
    @Insert(onConflict = OnConflictStrategy.REPLACE)  
    void insertPersons(Person ... persons);
```

Remoção de dados

@Insert

```
    void insertPerson(Person person);
```

@Update

```
    void updatePerson(Person person);
```

@Delete

```
    void deletePerson(Person person);
```

```
}
```

Atualização de dados

Room

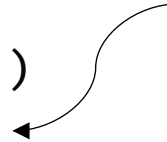
DAO -> Queries

@Dao

public interface PersonDAO {

@Query("SELECT * FROM Person")
List<Person> getAll();

Sem argumentos



Com argumentos



@Query("SELECT * FROM Person WHERE age BETWEEN :minAge AND :maxAge")
List<Person> loadPersonsBetweenAges(int minAge, int maxAge);

@Query("SELECT * FROM Person WHERE name LIKE :search")
List<Person> findPersonsWithName(String search);

}

Room

DAO -> Queries

```
@Dao
public interface PersonDAO {

    @Query("SELECT * FROM Book"
        + "INNER JOIN Loan ON loan.book_id = book._id"
        + "INNER JOIN Person ON person._id = loan.person_id"
        + "WHERE Person.name LIKE :name")
    List<Book> findBooksBorrowedByPerson(String name);

    @Query("SELECT Person.name, Pet.name AS petName"
        + "FROM Person, Pet"
        + "WHERE person._id = pet.person_id")
    public LiveData<List<PersonPet>> loadPersonsAndPets();

    static class PersonPet {
        public String personName;
        public String petName;
    }
}
```

Query com múltiplas tabelas



Query com múltiplas tabelas para objetos personalizados



Room

DAO

```
@Dao
public interface PersonDAO {
    @Query("SELECT * FROM person")
    List<Person> getAll();

    @Query("SELECT * FROM Person WHERE name LIKE :search")
    List<Person> findPersonsWithName(String search);

    @Query("SELECT * FROM person WHERE name LIKE :name LIMIT 1")
    Person findByName(String name);

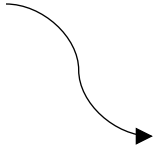
    @Insert
    void insertAll(Person... persons);

    @Delete
    void delete(Person person);
}
```

Room

Criação de Base de Dados

Uma classe abstrata não pode ser instanciada



```
@Database(entities = {Person.class}, version = 1)
public abstract class AppDatabase extends RoomDatabase {

    public abstract PersonDAO personDAO();
}
```

Podemos obter uma instancia da base de dados criada através deste código (ex. MainActivity.java)



```
AppDatabase db = Room.databaseBuilder(getApplicationContext(),
    AppDatabase.class, "database-name").build();
```

Room

Utilização

O presente código dá erro de execução!!
Operações sobre a base de dados **não são permitidas na thread da UI.**
É necessário recorrer a **threads em background** para operações sobre a interface DAO.
No futuro iremos abordar possíveis soluções.

```
public class MainActivity extends AppCompatActivity {
```

```
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);
```

```
        AppDatabase db = Room.databaseBuilder(getApplicationContext(),  
            AppDatabase.class,  
            "database-name")  
            .build();
```

```
        Person person = new Person();  
        person.name = "John";  
        person.age = 35;
```

Aceder à interface DAO e fazer a chamada do método

```
        db.personDAO().insertPerson(person);
```

```
    }
```

```
}
```

Room

Utilização

```
public class MainActivity extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        AppDatabase db = Room.databaseBuilder(getApplicationContext(),  
            AppDatabase.class,  
            "database-name")  
            .allowMainThreadQueries() ←  
            .build();  
  
        Person person = new Person();  
        person.name = "John";  
        person.age = 35;  
  
        db.personDAO().insertPerson(person);  
    }  
}
```

Para efeitos de teste podemos
adicionar esta propriedade

Room

Migração de Base de Dados

À medida que novos parâmetros são adicionadas à base de dados, existe a necessidade de implementar métodos corretivos para a aplicação não apagar os dados ao fazer o update à base de dados.

- Necessário fazer export dos Schemas atualizando o ficheiro `build.gradle (module)`

```
android {  
    ...  
  
    defaultConfig {  
        ...  
  
        javaCompileOptions{  
            annotationProcessorOptions{  
                arguments = ["room.schemaLocation":  
                            "$projectDir/schemas".toString()]  
            }  
        }  
    }  
    ...  
}
```

Room

Migração de Base de Dados

Adicionar os métodos de migração à classe da base de dados

- Cada versão da versão da base de dados deve originar um método de migração correspondente

```
@Database(entities = {Person.class}, version = 1)
public abstract class AppDatabase extends RoomDatabase {

    public abstract PersonDAO personDAO();

    private static AppDatabase INSTANCE;
    private static final Object sLock = new Object();

    static final Migration MIGRATION_1_2 = new Migration(1,2){
        @Override
        public void migrate(@NonNull SupportSQLiteDatabase database) {
            //TODO: Implementar código para alterar as tabelas da versão 1 para a 2
        }
    };
};

...
```

Room

Migração de Base de Dados (continuação)

...

```
static final AppDatabase getInstance(Context context){
    synchronized (sLock){
        if(INSTANCE == null){
            INSTANCE = Room.databaseBuilder(context.getApplicationContext(),
                AppDatabase.class, "sample.db")
                .addMigrations(MIGRATION_1_2)
                .build();
        }
        return INSTANCE;
    }
}
```


Leitura Adicional

Room Tutorial

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

Room Dependencies

<https://developer.android.com/jetpack/androidx/releases/room>

AndroidX Room

<https://developer.android.com/reference/androidx/room/package-summary>

Programação Para Dispositivos Móveis I

ROOM

2024/_25 CTeSP – Desenvolvimento para a Web e Dispositivos Móveis

Ricardo Barbosa , rmb@estg.ipp.pt

Carlos Aldeias, cfpa@estg.ipp.pt

Adaptação do conteúdo dos slides de João Ramos jrmr@estg.ipp.pt e Fábio Silva fas@estg.ipp.pt