ANDROİD

LOCAL DATABASE

ROOM

Room Nedir?

"Room, Android uygulamaları için yerel SQLite veritabanlarını kolayca oluşturmak ve yönetmek için kullanılan bir veritabanı kütüphanesidir."

- Room, geliştiricilere SQLite veritabanını kullanırken yaşanan karmaşıklığı azaltarak, veritabanı işlemleriyle daha etkili bir şekilde başa çıkmalarını sağlar.
- Veri güvenliği ve bütünlüğüne odaklanan Room, SQaL enjeksiyonu veya hata durumlarından kaynaklanabilecek sorunları önceden önler.
- Hızlı ve etkili veri erişimi sunan Room, kullanıcı deneyimini iyileştirir ve uygulamanın genel performansını artırır.
- Veri ilişkilendirmeyi kolaylaştıran ve verileri izlemeyi sağlayan Room, daha karmaşık uygulama mantığı oluşturmayı mümkün kılar, böylece kullanıcı etkileşimini ve uygulama duyarlılığını artırır.

Room ve SQLITE?

Derleme zamanı doğrulaması, boilerplate kodu azlığı ve şema değişikliklerinin otomatik yönetimi gibi özellikler, Room'u Android uygulamaları için iyi bir seçim haline getirir.

Özellik	**SQLite**	**Room**
Veri tabanı tipi	Yerel, dosya tabanlı	Yerel, dosya tabanlı
Veritabanı dili	SQL	SQL
ORM desteği	Hayır	Evet
Derleme zamanı doğrulaması	Hayır	Evet
Boilerplate kodu	Çok fazla	Az
Şema değişiklikleri	Manuel olarak yönetilir	Otomatik olarak yönetilir
LiveData ve RxJava ile entegrasyon	Hayır	Evet

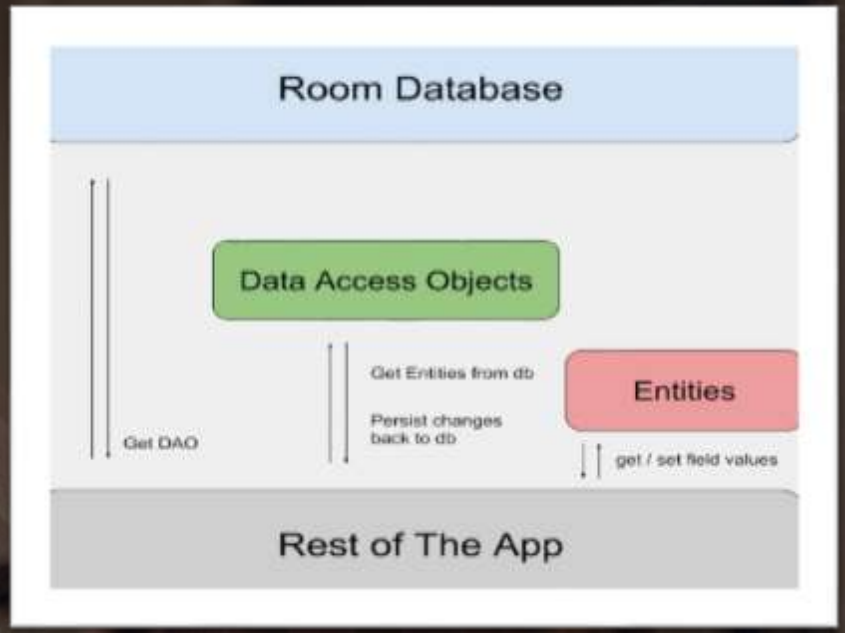
Room Kullanım Alanı	Açıklama
Veri önbelleği	İlgili veri parçalarını önbelleğe alarak, kullanıcıların çevrimdışıyken bile uygulamaya erişmesine ve içeriği görüntülemesine olanak tanır. Örneğin, bir alışveriş uygulamasında, kullanıcıların çevrimdışıyken ürünleri görüntülemesine veya sepetlerini gözden geçirmesine izin vermek için Room kullanılabilir.
Çevrimdışı işlemler	Çevrimdışı işlemler gerçekleştirerek, kullanıcıların uygulamayı çevrimdışıyken bile kullanmalarına olanak tanır. Örneğin, bir not uygulamasında, kullanıcıların çevrimdışıyken notlar oluşturmasına veya düzenlemesine izin vermek için Room kullanılabilir.
Karmaşık veri mantığı	Verileri ilişkilendirmeyi ve izlemeyi kolaylaştırarak, daha karmaşık veri mantığı oluşturmayı mümkün kılar. Örneğin, bir müşteri yönetimi uygulamasında, kullanıcıların müşteri bilgilerini yönetmesine ve müşteriler arasındaki ilişkileri izlemesine olanak tanımak için Room kullanılabilir.
Güvenli ve bütünleşik veri yönetimi	Verileri korumak için çeşitli güvenlik özellikleri sağlayarak, güvenli ve bütünleşik veri yönetimi sağlar. Örneğin, bir finans uygulamasında, kullanıcı verilerini korumak için Room kullanılabilir.
Veriye hızlı ve etkili erişim	Verileri önbelleğe almak ve verilere erişmek için akıllı algoritmalar kullanarak, verilere hızlı ve etkili erişim sağlar. Örneğin, bir arama uygulamasında, kullanıcıların arama sonuçlarına hızlı bir şekilde erişmesini sağlamak için Room kullanılabilir.

Odada(room) 3 temel bileşen mevcuttur,

- Veritabanını barındıran ve uygulamanızın kalıcı verilerine yapılan temel bağlantı için ana erişim noktası görevi gören veritabanı sınıfı.

- Uygulamanızın veritabanındaki tabloları temsil eden veri varlıkları.

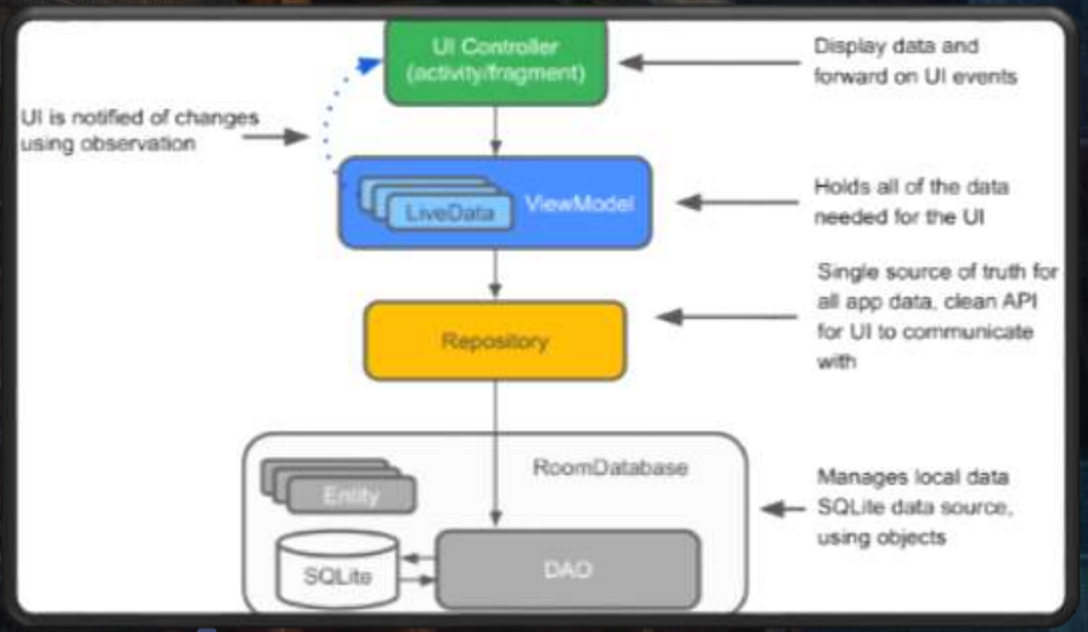
- Uygulamanızın veritabanındaki verileri sorgulamak, güncellemek, eklemek ve silmek için kullanabileceği yöntemler sunan veri erişim nesneleri (DAO'lar).



- Veritabanı sınıfını bir odanın zeminine ve duvarlarına benzetebiliriz varlıklar ve daolarla bitişik haldedir ve bu sınıf tanımlanırken diğer bileşenleri içerebilir.

- Veri varlıklarını odanın içindeki tablolar ve eşyalara benzetilebilir kullanıcıyla oda arasındaki en çok etkileşimin bulunduğu alandır burada tabloların tasarımı gayet önemlidir ve diğer varlıkların bütünlüğü de.

- DAO'ları ise odadaki pencere ve kapılara benzetebiliriz nesneleri almak atmak ve değiştirmek için odanın dışında(veritabanı-kullanıcı) olan ilişkisinde baş roldedirler



Annotations	Purpose
@Entity	Creates a SQLite table in the database using a data model class.
@Dao	Create a Data Access Object in the database using an interface class.
@Database	A class with this annotation will create an abstraction for the Data Access Object.
@PrimaryKey	A variable with this annotation will set a primary key for the table.
@Insert	Inserts parameters into the table.
@Update	Updates parameters of an existing table.
@Delete	Deletes parameters of an existing table
@Query	Running SQL query method within the table
@Ignore	Ignores the parameter form the Room database

DATABASE CLASS(VERİTABANI SINIFI)

```
// Song and Album are classes annotated with @Entity.
@Database(version = 1, entities = [Song::class, Album::class])
abstract class MusicDatabase : RoomDatabase {
  // SongDao is a class annotated with @Dao.
  abstract fun getSongDao(): SongDao

  // AlbumDao is a class annotated with @Dao.
  abstract fun getArtistDao(): ArtistDao

  // SongAlbumDao is a class annotated with @Dao.
  abstract fun getSongAlbumDao(): SongAlbumDao
```

•RoomDatabase, Android Jetpack'in bir parçası olan bir veritabanı kalıtımı sınıfıdır.

- Database anatasyonuyla veritabanı versiyonu tanımlayabaliriz
- Database anatosyonu entities parametresi ile tablo oluşturmamıza olanak sağlar
- Yandaki örnekte görüldüğü üzere 2 tabloya ve 3 daoya sahibiz.

• AutoMigration anatasyonu, veritabanı sürümlerini yükseltirken manuel olarak geçişler yazmaktan sizi kurtararak zamandan ve çabadan tasarruf etmenizi sağlar. Örneğin: yeni sutunlar eklediğinizde

• RoomDatabase.Builder sınıfının exportSchema() yöntemi veritabanı şemasını bir File nesnesi olarak döndürür. Bu nesneyi, istediğiniz klasöre kaydedebilirsiniz.

VARLIKLAR(ENTITIES)

```
Entity(
    tableName: String = "",
    indices: Array<Index> = [],
    inheritSuperIndices: Boolean = false,
    primaryKeys: Array<String> = [],
    foreignKeys: Array<ForeignKey> = [],
    ignoredColumns: Array<String> = []
)
```

Data Entity, Kotlin Room kütüphanesinde veritabanında saklanan bir veri modelini temsil eden bir sınıftır. Her Data Entity sınıfı, veritabanında saklanan bir tabloyu temsil eder.

Data Entity sınıfları, aşağıdaki özellikleri içerirler:

- @Entity anotasyonu: Sınıfı bir Data Entity olarak işaretler.
- IgnoredColumns parametresi: bir tablodaki sütunu göz ardı etmeye yarar

@PrimaryKey anotasyonu: Tablonun birincil anahtarını temsil eden bir veya daha fazla niteliği işaretler.

- foreignkeys parametresi: tablolar arası ilişkileri kurmamızda yardımcı olur

- @ColumnInfo anotasyonu: Tablonun bir sütununu temsil eden bir niteliği işaretler.
- InheritSuperIndices parametresi: üst sınıflardan miras alınan dizinleri alt sınıfa dahil etmek için kullanılır

DAO

Dao'lar, veritabanına veriler eklemek, güncellemek, silmek ve almak için yöntemler sağlar. Dao'lar, veritabanının soyut bir görünümünü sağlar, böylece uygulamanızın veritabanı şemasından bağımsız olarak çalışmasına olanak tanır.

@Insert, @Delete, @Update anotasyonlarıyla kolayca işlemleri gerçekleştirebilirsiniz, ayrıca sorguları da kullanabilirsiniz.

Bazı sorgularınız, sonucu hesaplamak için birden çok tabloya erişmeyi gerektirebilir. Room, sorgularınızda JOIN ifadelerini kullanarak birden çok tabloya erişmenizi sağlar.

DAO metodlarınızın bazıları çalışma zamanında bilinmeyen bir değişken sayısında parametreyi geçirmenizi gerektirebilir. Room, bir parametrenin bir koleksiyonu temsil ettiğini anlar ve çalışma zamanında sağlanan parametre sayısına bağlı olarak otomatik olarak genişletir.

DAO

Bazı sorgularınız sonucu hesaplamak için birden fazla tabloya erişim gerektirebilir. SQL sorgularınızda birden fazla tabloya başvurmak için JOIN ifadelerini kullanabilirsiniz.

Bazı sorgularınız, sonucu hesaplamak için birden çok tabloya erişmeyi gerektirebilir. Room, sorgularınızda JOIN ifadelerini kullanarak birden çok tabloya erişmenizi sağlar.

Room, diğer API kütüphaneleriyle entegrasyon için bazı özel dönüş tipleri sağlar.

Room, sayfalı sorguları Paging kütüphanesi ile entegrasyonu destekler. Room 2.3.0-alpha01 ve üstünde, DAO'lar Paging 3 ile kullanılmak üzere PagingSource nesnelerini döndürebilir.

Uygulamanızın mantığı, dönen satırlara doğrudan erişim gerektiriyorsa, DAO metodlarınızı bir Cursor nesnesi döndürmek üzere yazabilirsiniz.

DAO Anlatımı Demo Üzerinden Ayrıntılı Gerçeklenecektir

Room ve LiveData

•

.LiveData ve Room, Android uygulamalarında veritabanı işlemleri için kullanılan iki güçlü araçtır. Birlikte kullanıldığında, veritabanından alınan verileri kullanıcı arayüzüne yansıtmanın ve veri değişikliklerini otomatik olarak gözlemlemenin en verimli ve etkili yolundan birini sağlarlar.

•Örneğin, bir alışveriş listesi uygulamasında, Room kullanarak bir veritabanına alışveriş öğeleri ekleyebilir, silebilirsiniz ve güncelleyebilirsiniz. LiveData kullanarak bu sorguların sonuçlarını bir LiveData nesnesine dönüştürebilirsiniz. Ardından, bu LiveData nesnesini kullanıcı arayüzünüzün bir bileşenine bağlayabilirsiniz. Bileşen, LiveData'nın değişikliklerini dinleyecek ve alışveriş listeniz güncellendiğinde kullanıcı arayüzünü otomatik olarak güncelleyecektir.

•Bu yaklaşım, kullanıcı arayüzlerinizin her zaman güncel olduğundan emin olmanıza yardımcı olur. Ayrıca, veritabanı işlemlerini arka planda yürüterek kullanıcı arayüzlerinizin performansını iyileştirmenize yardımcı olur.

Teşekkürler

Alican Çağdaş

alicancagdas1@gmail.com

https://github.com/alicancagdas

Kaynakça:

- https://developer.android.com/
- https://medium.com/mindorks/using-room-database-android-jetpack-675a89a0e942