JETPACK AND COMPONENTS

COMP 41690

DAVID COYLE > D.COYLE@UCD.IE

Android Jetpack

Jetpack is a collection of Android software components to make it easier for you to develop great Android apps. These components help you follow best practices, free you from writing boilerplate code, and simplify complex tasks, so you can focus on the code you care about.

Jetpack comprises the androidx.* package libraries, unbundled from the platform APIs. This means that it offers backward compatibility and is updated more frequently than the Android platform, making sure you always have access to the latest and greatest versions of the Jetpack components.

GET STARTED

WATCH THE INTRO VIDEO

Accelerate development

Components are individually adoptable but built to work together while taking advantage of Kotlin language features that make you more productive.

Eliminate boilerplate code

Android Jetpack manages tedious activities like background tasks, navigation, and lifecycle management, so you can focus on what makes your app great.

Build high quality, robust apps

Built around modern design practices, Android Jetpack components enable fewer crashes and less memory leaked with backwards-compatibility baked in.

https://www.youtube.com/watch?v=r8U5Rtcr5UU&list=PLWz5rJ2EKKc9mxIBd0DRw9gwXuQshgmn2&index=2

Foundation

Foundation components provide cross-cutting functionality like backwards compatibility, testing and Kotlin language support.

AppCompat

Degrade gracefully on older versions of Android

Android KTX

Write more concise, idiomatic Kotlin code

Multidex

Provide support for apps with multiple DEX files

Test

An Android testing framework for unit and runtime UI tests

Architecture

Architecture components help you design robust, testable and maintainable apps.

Data Binding

Declaratively bind observable data to UI elements

Lifecycles

Manage your activity and fragment lifecycles

LiveData

Notify views when underlying database changes

Navigation

Handle everything needed for in-app navigation

Paging

Gradually load information on demand from your data source

Room

Fluent SQLite database access

ViewModel

Manage UI-related data in a lifecycle-conscious way

WorkManager

Manage your Android background jobs

Behavior

Behavior components help your app integrate with standard Android services like notifications, permissions, sharing and the Assistant.

Download manager

Schedule and manage large downloads

Media & playback

Backwards compatible APIs for media playback and routing (including Google Cast)

Notifications

Provides a backwards-compatible notification API with support for Wear and Auto

Permissions

Compatibility APIs for checking and requesting app permissions

Sharing

Provides a share action suitable for an app's action bar

Slices

Create flexible UI elements that can display app data outside the app

Preferences

Create interactive settings screens

UI

UI components provide widgets and helpers to make your app not only easy, but delightful to use.

Animation & transitions

Move widgets and transition between screens

Auto

Components to help develop apps for Android Auto

Emoji

Enable an up-to-date emoji font on older platforms

Fragment

A basic unit of composable UI

Layout

Lay out widgets using different algorithms

Palette

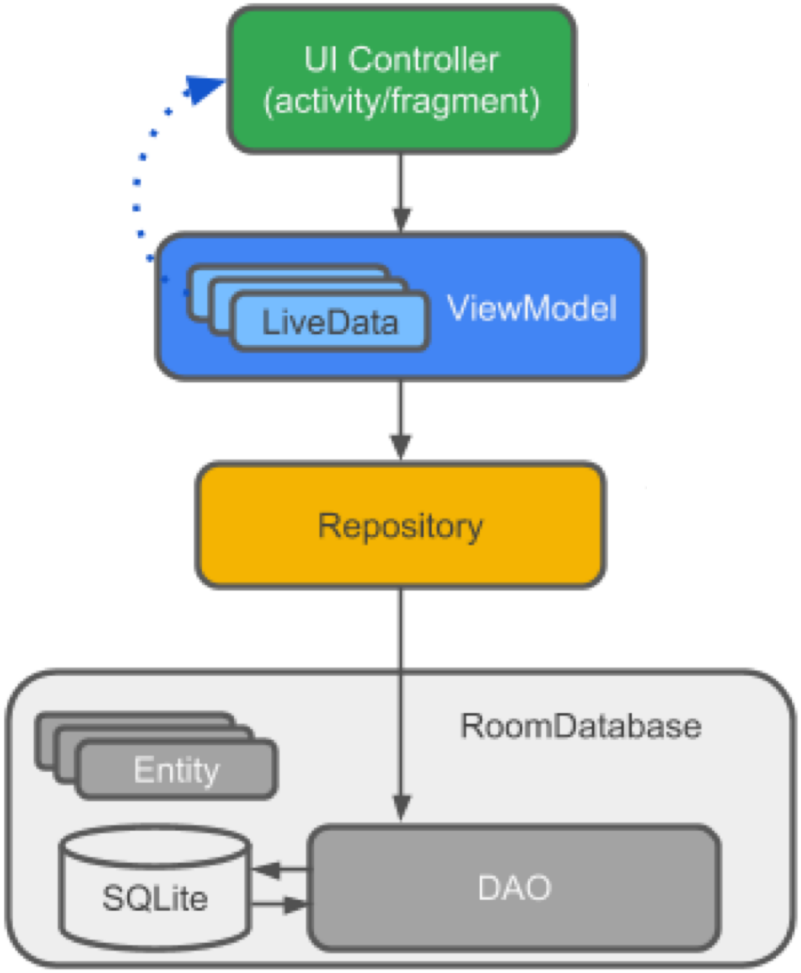
Pull useful information out of color palettes

TV

Components to help develop apps for Android TV

Wear OS by Google

Components to help develop apps for Wear

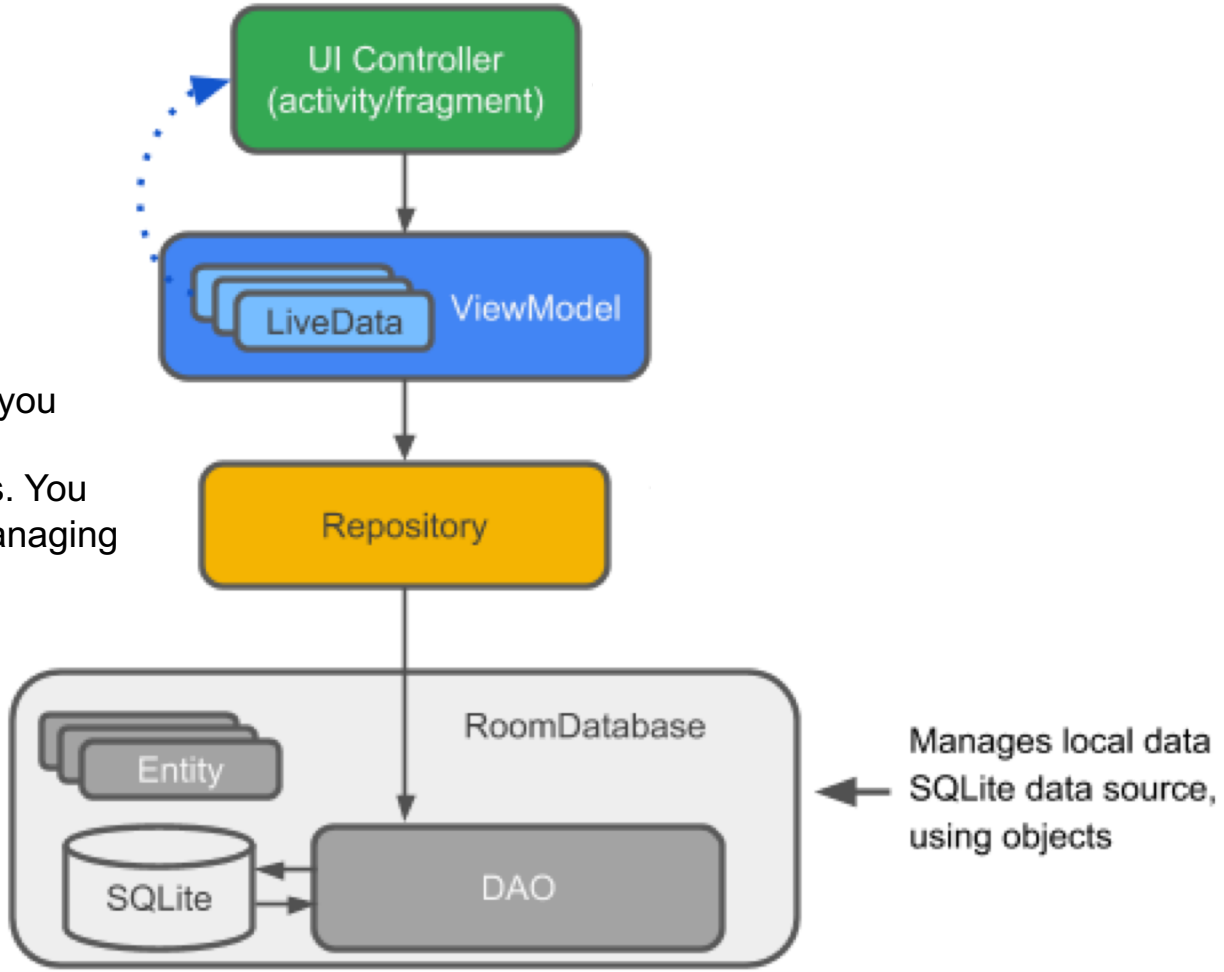


Entity: When working with Architecture Components, this is an annotated class that describes a database table.

DAO: Data access object. A mapping of SQL queries to functions.

Room database: Database layer on top of SQLite database that takes care of mundane tasks that you used to handle with an SQLiteOpenHelper.

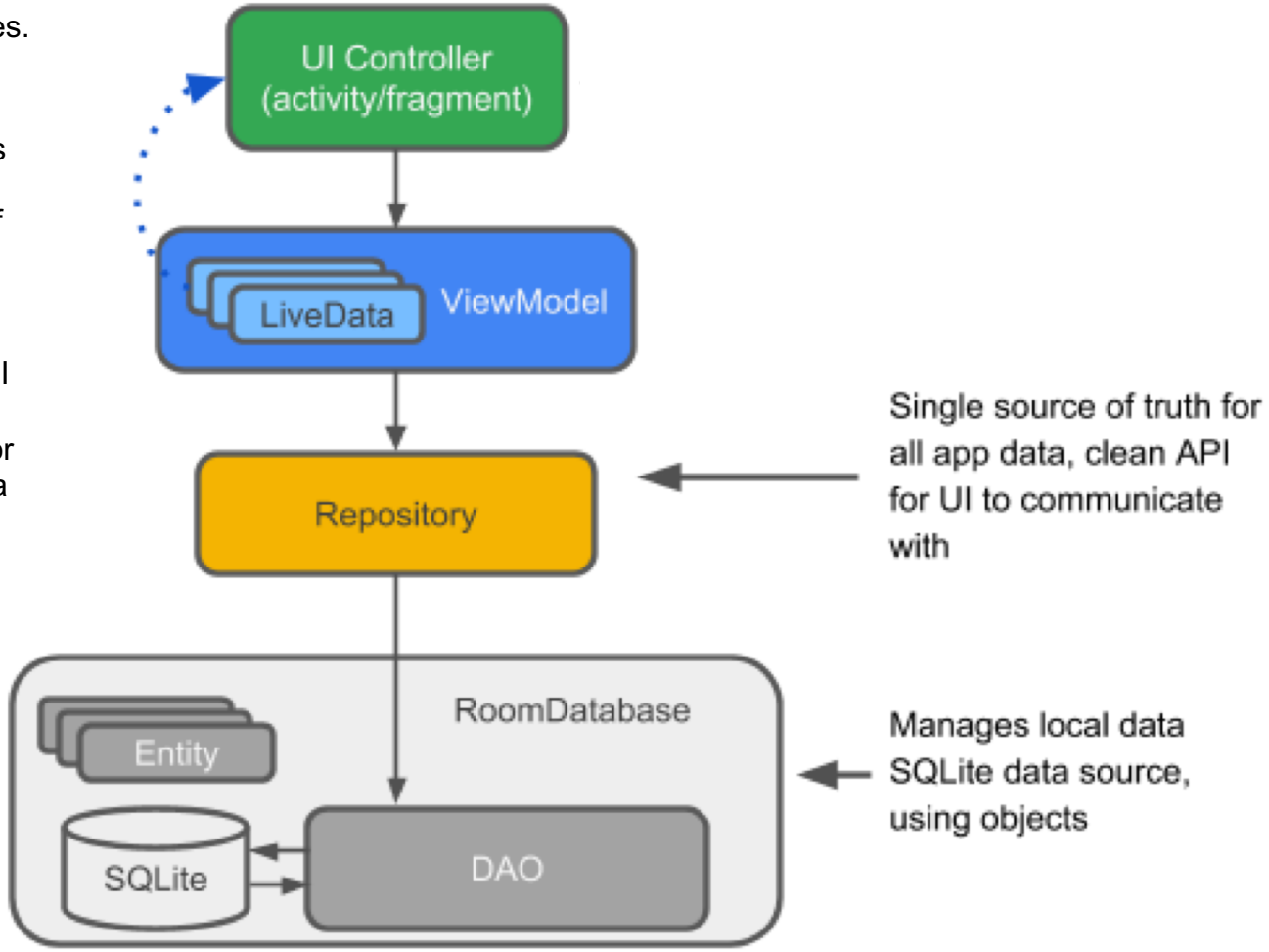
Repository: A class that you create, for example using the WordRepository class. You use the Repository for managing multiple data sources.

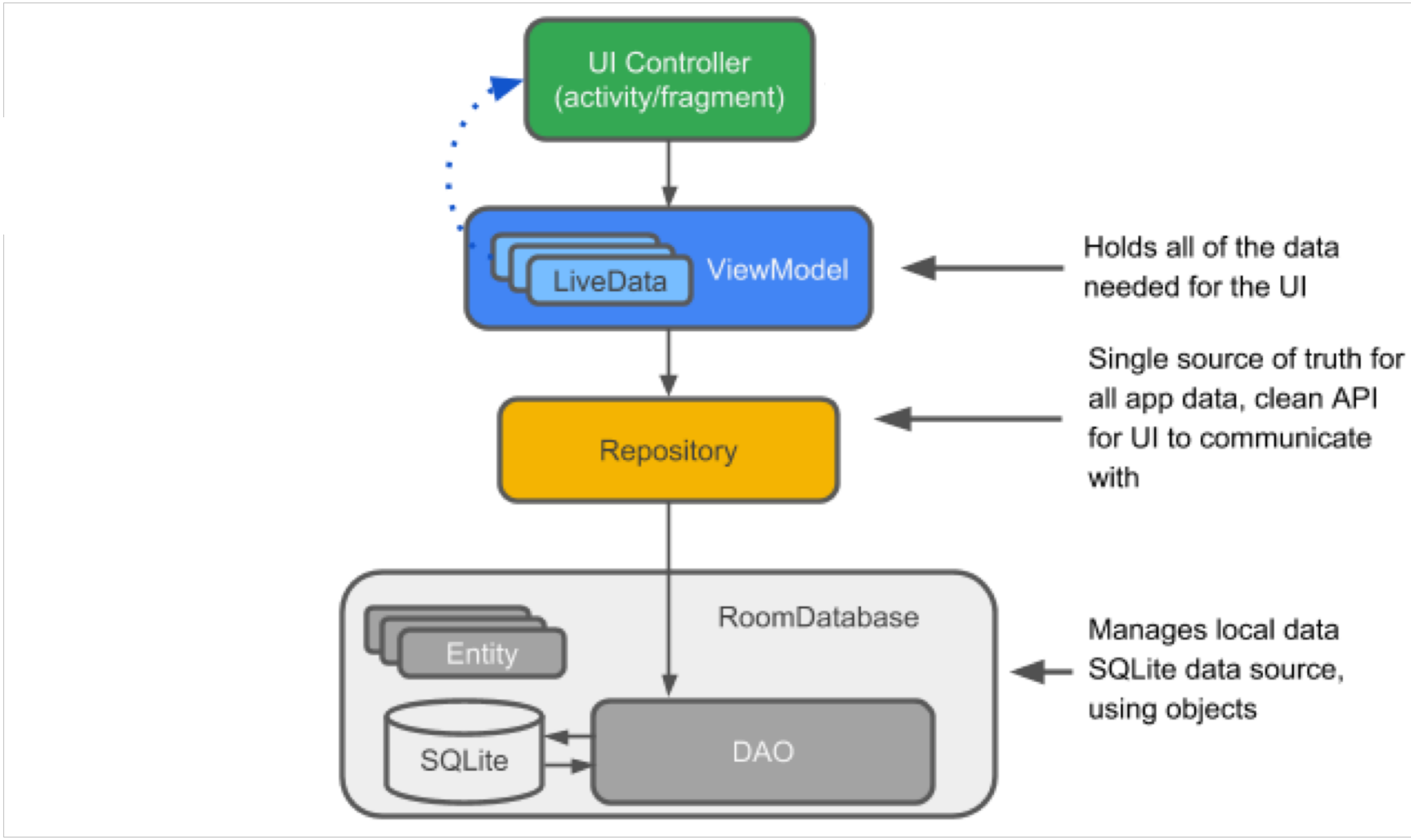


ViewModel: Provides data to the UI. Acts as a communication center between the Repository and the UI. Hides where the data originates from the UI. ViewModel instances survive configuration changes.

LiveData: A data holder class that can be observed. Always holds/caches latest version of data. Notifies its observers when the data has changed.

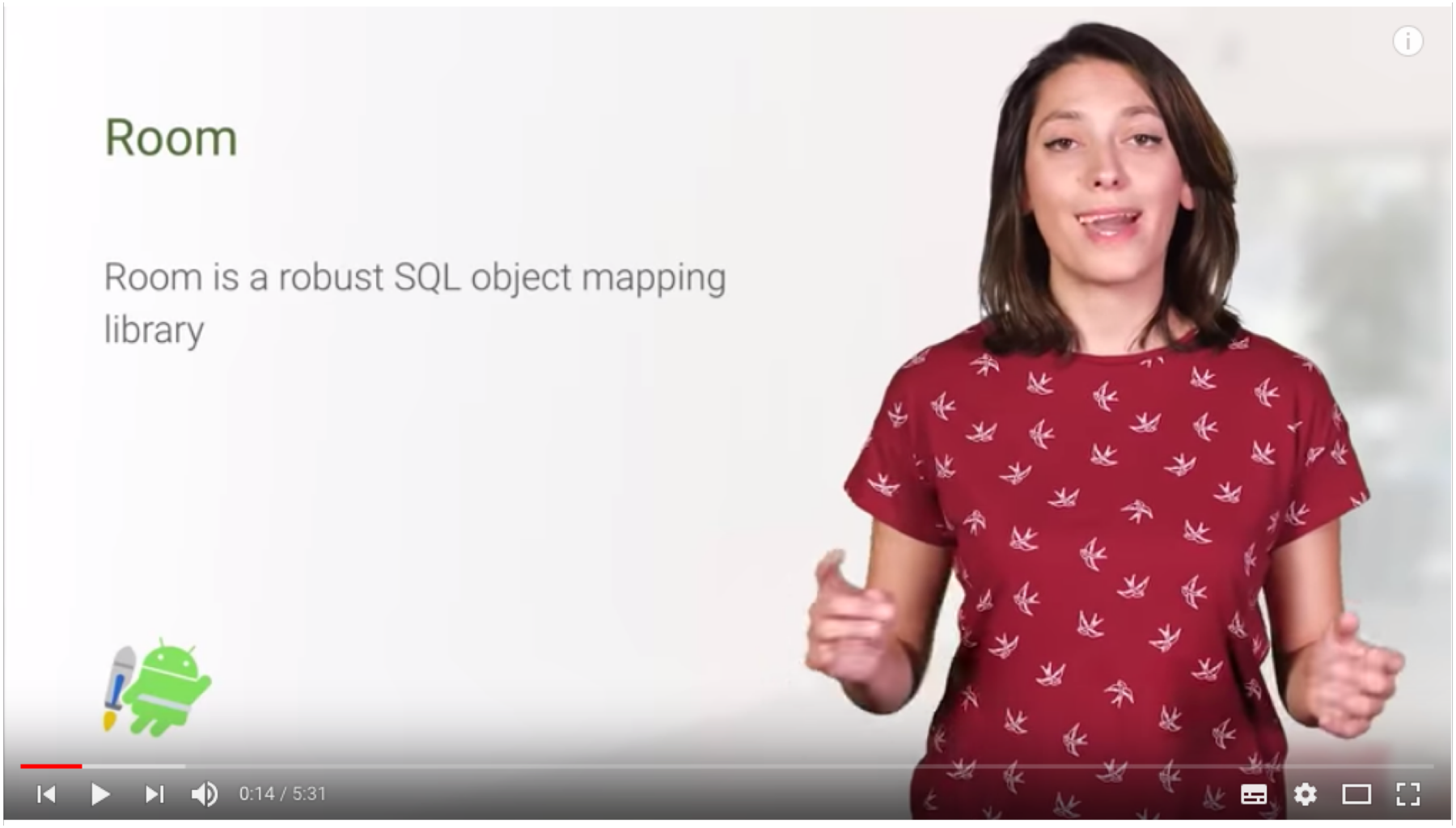
LiveData is lifecycle aware. UI components just observe relevant data and don't stop or resume observation. LiveData automatically manages all of this since it's aware of the relevant lifecycle status changes while observing.




UI Controller
(activity/fragment)
LiveData
ViewModel
Holds all of the data
needed for the UI
Repository
Single source of truth for
all app data, clean API
for UI to communicate
with
RoomDatabase
Entity
SQLite
DAO
Manages local data
SQLite data source,
using objects

https://www.youtube.com/watch?v=7p22cSzniBM

See also: https://developer.android.com/topic/libraries/architecture/



https://www.youtube.com/watch?v=SKWh4ckvFPM&list=PLWz5rJ2EKKc9mxIBd0DRw9gwXuQshgmn2&index=5

ROOMS TUTORIALS

Codelabs: Room with a View:

https://codelabs.developers.google.com/codelabs/android-room-with-a-view/#0

7 Steps to Room:

https://medium.com/androiddevelopers/7-steps-to-room-27a5fe5f99b2

QUESTIONS?

Contact:

d.coyle@ucd.ie

Next:

ROOMS