## Mobile Application Development



Department of Computing & Mathematics Waterford Institute of Technology http://www.wit.ie





# Introducing Android KTX Library

Android KTX Library - Part 2





A further look at the Kotlin
Extension Library for
Android





- ■Background
- ■Extension Functions
- □The KTX Library Packages (Core, Fragment, Palette, SQLite, Collections)
- ■KTX in our Case Study (Donation)





- ■Background
- ■Extension Functions
- □ The KTX Library Packages (Core, Fragment, Palette, SQLite, Collections)
- ■KTX in our Case Study (Donation)

# Recap



new addition to the Android Jetpack family

□ collection of Kotlin extension functions that enhances existing Android APIs

■takes advantage of many of the new features in Kotlin

## Official Docs (https://developer.android.com/kotlin/ktx)



Android Developers > Kotlin > Get Started





Android KTX is a set of Kotlin extensions that are included with Android Jetpack and other Android libraries. KTX extensions provide concise, idiomatic Kotlin to Jetpack, Android platform, and other APIs. To do so, these extensions leverage several Kotlin language features, including the following:

- Extension functions
- Extension properties
- Lambdas
- Named parameters
- Parameter default values
- Coroutines

# Android KTX Packages



- □ The KTX toolset is broken up into a number of different packages, so that you can import only the ones you need into your app project:
  - Core KTX: for use with framework APIs such as Toasts, Spans, and Menus
  - Fragment KTX: simplifies Fragment transactions
  - Palette KTX: for working with color palette APIs
  - SQLite KTX: simplifies SQLite database transactions
  - Collections KTX: for use with collection APIs
  - Navigation KTX: for use with Jetpack Navigation

# Android KTX Packages



- □ The KTX toolset is broken up into a number of different packages, so that you can import only the ones you need into your app project:
  - Core KTX: for use with framework APIs such as Toasts, Spans, and Menus
  - Fragment KTX: simplifies Fragment transactions
  - Palette KTX: for working with color palette APIs
  - SQLite KTX: simplifies SQLite database transactions
  - Collections KTX: for use with collection APIs
  - Navigation KTX: for use with Jetpack Navigation

# Fragment KTX

simplify the fragment api



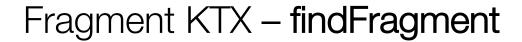
# Fragment KTX



☐ The Fragment KTX module provides a number of extensions to simplify the fragment API.

■To include this module, add the following to your app's build.gradle file:

```
dependencies {
   implementation "androidx.fragment:fragment-ktx:1.2.3"
}
```





☐ Find a **Fragment** associated with a **View**.

☐ This method will locate the Fragment associated with this view. This is automatically populated for the View returned by Fragment.onCreateView and its children.

var myFragment = view.findFragment()





□ A helper function to simplify the fragment transaction process

```
supportFragmentManager.commit {
     addToBackStack("...")
     replace (R.id.fragment container, myFragment, MY TAG)
☐ Instead of
 supportFragmentManager
    .beginTransaction()
    .addToBackStack("...")
    .replace(R.id.fragment container, myFragment, MY TAG)
    .commit()
```

\_\_\_1

# SQLite KTX

simplify database transactions



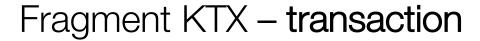
## **SQLite KTX**



provides a number of extensions to simplify the fragment API.

wraps SQL-related code in transactions, eliminating a lot of boilerplate code.

```
dependencies {
   implementation "androidx.sqlite:sqlite-ktx:2.1.0"
}
```





☐ A helper function to simplify the database transaction process

```
db.transaction { // insert data }
□Instead of
        db.beginTransaction()
        try {
            // insert data
            db.setTransactionSuccessful()
        finally {
            db.endTransaction()
```

## Collections KTX

helper functions for the collections api

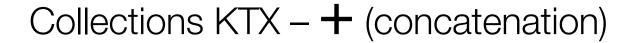


## Collections KTX



provides utility functions for working with Android's memoryefficient collection libraries, including ArrayMap, LongParseArray, LruCache, and others.

```
dependencies {
    implementation "androidx.collection:collection-ktx:1.1.0"
```

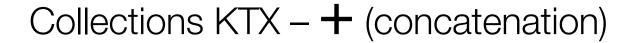




□ takes advantage of Kotlin's operator overloading to simplify collection concatenation

```
// Combine 2 ArraySets into 1.
val combined = arraySetOf(1, 2, 3) + arraySetOf(4, 5, 6)
// Combine with numbers to create a new sets.
val newArraySet = combined + 7 + 8
```

☐ Instead of





```
// Combine 2 ArraySets into 1.
var set1 = arraySetOf(1, 2, 3)
var set2 = arraySetOf(4, 5, 6)
var combined = arraySetOf<Int>()
combined.addAll(set1)
combined.addAll(set2)
// Combine with numbers to create a new sets.
val newArraySet = combined
newArraySet.add(7)
newArraySet.add(8)
```

## Android KTX Modules



Core KTX

Collection KTX

Fragment KTX

Lifecycle KTX

LiveData KTX

**Navigation KTX** 

Palette KTX

Reactive Streams KTX

Room KTX

SQLite KTX

ViewModel KTX

WorkManager KTX

Other KTX modules

Firebase KTX

Play Core KTX

## Android KTX Modules



Core KTX

Collection KTX

Fragment KTX

Lifecycle KTX

LiveData KTX

**Navigation KTX** 

Palette KTX

Reactive Streams KTX

Room KTX

SQLite KTX

ViewModel KTX

WorkManager KTX

Other KTX modules

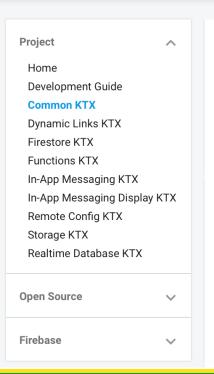
Firebase KTX

Play Core KTX



#### Documentation

GUIDES GITHUB 🗹 MAIN REFERENCE 🗹 KTX REFERENCE 🖸



#### Firebase Common Kotlin Extensions

**Contents** Getting Started

Features

#### **Getting Started**

To use the Firebase Common Android SDK with Kotlin Extensions, add the following to your app's build.gradle file:

```
// See maven.google.com for the latest versions
// This library transitively includes the firebase-common library
implementation 'com.google.firebase:firebase-common-ktx:$VERSION'
```

#### Features

Get the default FirebaseApp and FirebaseOptions

#### Kotlin

```
val defaultApp = FirebaseApp.getInstance()
val defaultOptions = defaultApp.options
```

#### Kotlin + KTX

```
val defaultApp = Firebase.app
val defaultOptions = Firebase.options
```

Initialize a FirebaseApp

#### **Kotlin**

```
val options = FirebaseApp.getInstance().options
val anotherApp = FirebaseApp.initializeApp(context, options, "myApp")
```

#### Kotlin + KTX

```
var anotherApp = Firebase.initialize(context, Firebase.options, "myApp")
```





- ■Background
- ■Extension Functions
- □ The KTX Library Packages (Core, Fragment, Palette, SQLite, Collections)
- ■KTX in our Case Study (Donation)



### References

Sources:

https://www.raywenderlich.com/5576-android-ktx-tutorial-getting-started

https://blog.mindorks.com/android-ktx-android-development-with-kotlin

https://developer.android.com/kotlin/ktx

https://codinginfinite.com/android-ktx-example-android-ktx-core-ktx/

https://ansgarlin.github.io/zh-tw/kotlin/kotlin\_vs\_ktx.html



