Kotlin Ecosystem: Android

JUNO fitbit SPACE JET BRAINS

school.kt

Kotlin is first-class language for Android apps

- Kotlin friendly Android SDK
- Code samples in Kotlin
- Android Studio and tooling support
- Learn: Kotlin Bootcamp for Programmers by Google
- Android KTX

Android KTX

school.kt

android-ktx

- androidx.core:core-**ktx**
- androidx.activity:activity-**ktx**
- androidx.fragment:fragment-**ktx**
- androidx.dynamicanimation:dynamicanimation-**ktx**
- androidx.palette:palette-**ktx**
- androidx.sqlite:sqlite-**ktx**
- androidx.collection:collection-**ktx**
- androidx.lifecycle:lifecycle-viewmodel-**ktx**
- androidx.lifecycle:lifecycle-reactivestreams-**ktx**
- androidx.lifecycle:lifecycle-livedata-core-**ktx**
- androidx.paging:paging-*-**ktx**
- androidx.preference:preference-**ktx**
- androidx.slice:slice-builders-**ktx**
- androidx.test:test-core-**ktx**
- android.arch.navigation:navigation-*-**ktx**
- android.arch.work:work-runtime-**ktx**

android-ktx

```
sharedPreferences.edit()
    .putBoolean("key", value)
    .apply()
```

android-ktx

```kotlin
sharedPreferences.edit {
    putBoolean("key", value)
}
```

android-ktx

```
fragmentManager.beginTransaction()
    .replace(R.id.container, myFragment, FRAGMENT_TAG)
    .commitAllowingStateLoss()
```

android-ktx

```
fragmentManager.transaction(allowStateLoss = true) {
    replace(R.id.container, myFragment, FRAGMENT_TAG)
}
```

Room 2.0 Alpha Coroutines

```kotlin
@Dao
interface UsersDao {

    @Query("UPDATE users SET age = age + 1 WHERE userId = :userId")
    suspend fun incrementUserAge(userId: String)

    @Insert
    suspend fun insertUser(user: User)

    @Update
    suspend fun updateUser(user: User)

    @Delete
    suspend fun deleteUser(user: User)
}
```

school.kt

Kotlin Android Extensions

school.kt

Setup

```
// Add plugin to a module build.gradle
apply plugin: 'kotlin-android-extensions'
```

View Binding

school.kt

View Binding

```
<TextView
    android:id="@+id/textView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    />
```

View Binding

```
class MyActivity : Activity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_call)

        val textView = findViewById<TextView>(R.id.text)
        textView.text = "Hello, world!"
    }
}
```

View Binding

```
import kotlinx.android.synthetic.main.activity_main.*

class MyActivity : Activity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_my)

        // Instead of findViewById<TextView>(R.id.textView)
        textView.text = "Hello, world!"
    }
}
```

Parcelize

school.kt

```
class User(
    val firstName: String,
    val lastName: String,
    val age: Int
) : Parcelable {

    constructor(source: Parcel) : this(
        source.readString(),
        source.readString(),
        source.readInt()
    )

    override fun writeToParcel(dest: Parcel, flags: Int)
{
    override fun writeToParcel(dest: Parcel, flags: Int)
{
        dest.writeString(firstName)
        dest.writeString(lastName)
        dest.writeInt(age)
    }

    override fun describeContents() = 0

    companion object {

        @JvmField
        val CREATOR = object : Parcelable.Creator<User> {

            override fun createFromParcel(source: Parcel)
= User(source)
```

school.kt

```
    companion object {

        @JvmField
        val CREATOR = object : Parcelable.Creator<User> {

            override fun createFromParcel(source: Parcel)
 = User(source)

            override fun newArray(size: Int) =
 arrayOfNulls<User>(size)
        }
    }
}
```

```
class User(
    val firstName: String,
    val lastName: String,
    val age: Int
) : Parcelable {

    constructor(source: Parcel) : this(
        source.readString(),
        source.readString(),
        source.readInt()
    )

    override fun writeToParcel(dest: Parcel, flags: Int)
{
```

Kotlin Android Ext Parcelable

```kotlin
@Parcelize
class User(
    val firstName: String,
    val lastName: String,
    val age: Int
) : Parcelable
```

school.kt

Third party libraries with Kotlin support

- Retrofit 2
- PermissionDispatcher
- Kotlin Anko
- Kotter Knife - Butterknife for Kotlin
- Barista - Espresso in Kotlin way
- Koin Android

insert-Koin.io

school.kt

Koin

```
// just declare it
val myModule = module {
    single { Controller(get()) }
    single { BusinessService() }
}



class Controller(val service : BusinessService)
class BusinessService()
```

Koin

```
class MyActivity : Activity() {

    // lazy inject BusinessService into property
    val service : BusinessService by inject()

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        // or directly get any instance
        val service : BusinessService = get()
    }
}
```

Koin

```
class MyApp : Application() {

    override fun onCreate() {
        super.onCreate()

        startKoin {
            // Android context
            androidContext(this@MyApp)
            // modules
            modules(myModule)
        }
    }
}
```

ViewModel

school.kt

Koin ViewModel

```
class MyViewModel(val repo : MyRepository) : ViewModel()

class MyActivity : AppCompatActivity() {

    val myViewModel : MyViewModel = …
}
```

Koin ViewModel

```
// declared ViewModel using the viewModel keyword
val myModule = module {
    viewModel { MyViewModel(get()) }
    single { MyRepository() }
}
```

Koin ViewModel

```kotlin
// Just get it
class MyActivity : AppCompatActivity() {

    // lazy inject MyViewModel
    val myViewModel : MyViewModel by viewModel()
}
```

Koin

- + Pure Kotlin
- + DSL for declare dependencies
- + No compile time code generation or reflection
- + Unit test support from the box
- – No graph validation at compile time
- No advanced level features

Thanks!!!

school.kt