

Kotlin-V

Agenda

- Android Components in Kotlin
- Best Practices to write these components
- Take advantage Kotlin properties properly.
- Use where ever you feel you have great powers

Activities

Declaration:

```
class MainActivity : AppCompatActivity() {

    companion object {
        val log = Logger.getLogger(MainActivity::class.java.name)
    }

    /**
     *
     * Code
     *
     */
}
```

Using Views in your activities:

```
package com.developers.kotlindemo

import android.support.v7.app.AppCompatActivity
import android.os.Bundle
import kotlinx.android.synthetic.main.activity_main.*

class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        //Text View with a message
        welcomeMessage.text = "Hello peeps"
    }

}
```

```
class MainActivity : AppCompatActivity() {

    companion object {
        val log = Logger.getLogger(MainActivity::class.java.name)
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        log.info("OnCreate Called")
    }

    override fun onStart() {
        super.onStart()
        log.info("onStart Called")
    }

    override fun onRestart() {
        super.onRestart()
        log.info("OnRestart Called")
    }


    override fun onResume() {
        super.onResume()
        log.info("OnResume Called")
    }

    override fun onStop() {
        super.onStop()
        log.info("OnStop Called")
    }

    override fun onPause() {
        super.onPause()
        log.info("OnPause Called")
    }

    override fun onDestroy() {
        super.onDestroy()
        log.info("OnDestroy Called")
    }
}
```

Fragment Transactions



```
supportFragmentManager.beginTransaction()  
    .setCustomAnimations(R.anim.design_bottom_sheet_slide_in,  
R.anim.design_bottom_sheet_slide_out)  
    .replace(R.id.content, fragment, fragment.javaClass.getSimpleName())  
    .commit()
```

Candidate for keeping in extensions.


Imagine: **commitFragment(HomeFragment)** will do.

Opening Activities with intents



```
fun startIntent(s : Student){  
    val intent = Intent(this, HomeActivity::class.java)  
    with(intent){  
        putExtra(NAME, s.name) // No need to use intent object anymore  
        putExtra(SUBJECT, s.subject)  
        putExtra(IMAGE, s.image)  
    }  
    startActivity(intent)  
}
```

Recycler View



```
class MyAdapter(val myAndroidOsList: List<String>, val context: Context) :
    RecyclerView.Adapter<MyAdapter.MyViewHolder>() {

    override fun getItemCount(): Int {
        return myAndroidOsList.size
    }

    override fun onBindViewHolder(holder: MyViewHolder?, position: Int) {
        holder?.bindItems(myAndroidOsList[position])
    }

    override fun onCreateViewHolder(parent: ViewGroup?, viewType: Int): MyViewHolder {
        val v = LayoutInflater.from(context).inflate(R.layout.list_row_item,
            parent, false)
        return MyViewHolder(v)
    }

    class MyViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {

        fun bindItems(myAndroidOsListName: String) {
            itemView.android_nane_text_view.text = myAndroidOsListName
        }
    }
}
```

Recycler View

```
class MyAdapter(val myAndroidOsList: List<String>, val context: Context) :  
    RecyclerView.Adapter<MyAdapter.MyViewHolder>() {  
  
    override fun getItemCount(): Int {  
        return myAndroidOsList.size  
    }  
  
    override fun onBindViewHolder(holder: MyViewHolder?, position: Int) {  
        holder?.bindItems(myAndroidOsList[position])  
    }  
  
    override fun onCreateViewHolder(parent: ViewGroup?, viewType: Int): MyViewHolder {  
        val v = LayoutInflater.from(context).inflate(R.layout.list_row_item,  
            parent, false)  
        return MyViewHolder(v)  
    }  
  
    class MyViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {  
  
        fun bindItems(myAndroidOsListName: String) {  
            itemView.android_nane_text_view.text = myAndroidOsListName  
        }  
    }  
}
```


Recycler View

```
class MyAdapter(val myAndroidOsList: List<String>, val context: Context) :  
    RecyclerView.Adapter<MyAdapter.MyViewHolder>() {  
  
    override fun getItemCount(): Int {  
        return myAndroidOsList.size  
    }  
  
    override fun onBindViewHolder(holder: MyViewHolder?, position: Int) {  
        holder?.bindItems(myAndroidOsList[position])  
    }  
  
    override fun onCreateViewHolder(parent: ViewGroup?, viewType: Int): MyViewHolder {  
        val v = LayoutInflater.from(context).inflate(R.layout.list_row_item,  
            parent, false)  
        return MyViewHolder(v)  
    }  
  
    class MyViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {  
  
        fun bindItems(myAndroidOsListName: String) {  
            itemView.android_nane_text_view.text = myAndroidOsListName  
        }  
    }  
}
```

Recycler View

```
class MyAdapter(val myAndroidOsList: List<String>, val context: Context) :  
    RecyclerView.Adapter<MyAdapter.MyViewHolder>() {  
  
    override fun getItemCount(): Int {  
        return myAndroidOsList.size  
    }  
  
    override fun onBindViewHolder(holder: MyViewHolder?, position: Int) {  
        holder?.bindItems(myAndroidOsList[position])  
    }  
  
    override fun onCreateViewHolder(parent: ViewGroup?, viewType: Int): MyViewHolder {  
        val v = LayoutInflater.from(context).inflate(R.layout.list_row_item,  
            parent, false)  
        return MyViewHolder(v)  
    }  
  
    class MyViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {  
  
        fun bindItems(myAndroidOsListName: String) {  
            itemView.android_nane_text_view.text = myAndroidOsListName  
        }  
    }  
}
```

Intent Service

```
/**
 * An [IntentService] subclass for handling asynchronous task requests in
 * a service on a separate handler thread.
 *
 *
 * TODO: Customize class - update intent actions and extra parameters.
 */
class MyIntentService : IntentService("MyIntentService") {

    val log = Logger.getLogger(MyIntentService::class.java.name)

    override fun onHandleIntent(intent: Intent?) {
        if (intent != null) {
            val data = intent.getStringExtra("data")
            log.info("Message is : ${data}")
        }
    }

}
```

Fragments



```
class PopularFragment : Fragment() {  
    companion object {  
        fun newInstance(): PopularFragment {  
            val fragmentHome = PopularFragment()  
            val args = Bundle()  
            fragmentHome.arguments = args  
            return fragmentHome  
        }  
    }  
  
    override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,  
                              savedInstanceState: Bundle?): View? {  
        // Inflate the layout for this fragment  
        return inflater.inflate(R.layout.fragment_popular, container, false)  
    }  
}
```

Api Calls

- RxKotlin
- Retrofit (Normal)
- OkHttp (Normal)

RxKotlin ==> RxJava + Kotlin Extensions



- Less Code
- More Powers

Simple Example

```
data class MovieResult(  
    @SerializedName("page") var page: Int = 0,  
    @SerializedName("total_results") var totalResults: Int = 0,  
    @SerializedName("total_pages") var totalPages: Int = 0,  
    @SerializedName("results") var results: List<Result> = listOf()  
)  
  
data class Result(  
    @SerializedName("vote_count") var voteCount: Int = 0,  
    @SerializedName("id") var id: Int = 0,  
    @SerializedName("video") var video: Boolean = false,  
    @SerializedName("vote_average") var voteAverage: Double = 0.0,  
    @SerializedName("title") var title: String = "",  
    @SerializedName("popularity") var popularity: Double = 0.0,  
    @SerializedName("poster_path") var posterPath: String = "",  
    @SerializedName("original_language") var originalLanguage: String = "",  
    @SerializedName("original_title") var originalTitle: String = "",  
    @SerializedName("genre_ids") var genreIds: List<Int> = listOf(),  
    @SerializedName("backdrop_path") var backdropPath: String = "",  
    @SerializedName("adult") var adult: Boolean = false,  
    @SerializedName("overview") var overview: String = "",  
    @SerializedName("release_date") var releaseDate: String = ""  
)
```

Model Class



```
dataManager.searchMovieQuery(key, query)
    .subscribeOn(Schedulers.io())
    .observeOn(AndroidSchedulers.mainThread())
    .subscribe({ result ->
        val movieList = result.results
        log.info(" " + movieList[0].title)
        view?.showData(movieList)
        view?.hideLoading()
    }, { e -> view?.showError(e.message.toString()) })
```

onNext() and onError() implementations

Thank You

