Retrofit and Coroutines

목표 :

나라 이름의 리스트를 endpoint 로부터 받아온다. 코루틴, Retrofit, MVVM 을 사용한다. 이 리스트를 RecyclerView 에다가 띄워준다.

0. 레이아웃 및 그래이들 설정하기 (뷰 홀더 레이아웃도 만들어주기)

build.gradle

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation"org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin_version"
    implementation 'androidx.appcompat:appcompat:1.1.0'
    implementation 'androidx.core:core-ktx:1.1.0'
    implementation 'androidx.constraintlayout:constraintlayout:1.1.3'

    implementation "androidx.recyclerview:recyclerview:1.1.0"
    implementation "androidx.swiperefreshlayout:swiperefreshlayout:1.0.0"

    implementation 'com.github.bumptech.glide:glide:4.8.0'

    implementation 'com.squareup.retrofit2:retrofit:2.6.0'
    implementation 'com.squareup.retrofit2:converter-gson:2.6.0'

    implementation 'android.arch.lifecycle:extensions:1.1.1'

    //coroutines
    implementation 'org.jetbrains.kotlinx:kotlinx-coroutines-core:1.3.0'
    implementation 'org.jetbrains.kotlinx:kotlinx-coroutines-android:1.3.0'

    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test:runner:1.2.0'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.2.0'
}
```

1. **모델 클래스 만들기**

```
package com.devtides.androidcoroutinesretrofit.model

import com.google.gson.annotations.SerializedName

data class Country(
    @SerializedName("name")
    val countryName: String?,

    @SerializedName("capital")
    val capital: String?,

    @SerializedName("flagPNG")
    val flag: String?
)
```

2. Retrofit API → Service object class → Retrofit 객체 만들기

CountriesApi.kt

```
package com.devtides.androidcoroutinesretrofit.model

import retrofit2.Response
import retrofit2.http.GET

interface CountriesApi {
    @GET("DevTides/countries/master/countriesV2.json")
```

```
    suspend fun getCountries() : Response<List<Country>>
}
```

CountriesService.kt

```
package com.devtides.androidcoroutinesretrofit.model

import retrofit2.Retrofit
import retrofit2.converter.gson.GsonConverterFactory

//object -> 싱글톤이 된다.
object CountriesService {
    private val BASE_URL = "https://raw.githubusercontent.com/"

    fun getCountriesService():CountriesApi{
        return Retrofit.Builder()
            .baseUrl(BASE_URL)
            .addConverterFactory(GsonConverterFactory.create())
            .build()
            .create(CountriesApi::class.java)
    }
}
```

- **object class 사용 참고하기.**

3. RecyclerView Adapter 와 ViewHolder 클래스 작성하기

```
package com.devtides.coroutinesretrofit.view

import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.recyclerview.widget.RecyclerView
import com.devtides.androidcoroutinesretrofit.R
import com.devtides.androidcoroutinesretrofit.model.Country
import kotlinx.android.synthetic.main.item_country.view.*

class CountryListAdapter(var countries: ArrayList<Country>): RecyclerView.Adapter<CountryListAdapter.CountryViewHolder>() {

    fun updateCountries(newCountries: List<Country>) {
        countries.clear()
        countries.addAll(newCountries)
        notifyDataSetChanged()
    }

    override fun onCreateViewHolder(parent: ViewGroup, p1: Int) = CountryViewHolder(
        LayoutInflater.from(parent.context).inflate(R.layout.item_country, parent, false)
    )

    override fun getItemCount() = countries.size

    override fun onBindViewHolder(holder: CountryViewHolder, position: Int) {
        holder.bind(countries[position])
    }

    class CountryViewHolder(view: View): RecyclerView.ViewHolder(view) {

        private val imageView = view.imageView
        private val countryName = view.name
        private val countryCapital = view.capital

        fun bind(country: Country) {
            countryName.text = country.countryName
            countryCapital.text = country.capital
            imageView.loadImage(country.flag)
        }
    }
}
```

4. ImageView extension function 작성하기

Util.kt

```
package com.devtides.coroutinesretrofit.view

import android.widget.ImageView
import com.bumptech.glide.Glide
import com.bumptech.glide.request.RequestOptions
import com.devtides.androidcoroutinesretrofit.R


fun ImageView.loadImage(uri: String?) {
    val options = RequestOptions()
        .error(R.mipmap.ic_launcher_round)
    Glide.with(this.context)
        .setDefaultRequestOptions(options)
        .load(uri)
        .into(this)
}
```

5. ViewModel 클래스 작성하기

```
package com.devtides.androidcoroutinesretrofit.viewmodel

import androidx.lifecycle.MutableLiveData
import androidx.lifecycle.ViewModel
import com.devtides.androidcoroutinesretrofit.model.CountriesService
import com.devtides.androidcoroutinesretrofit.model.Country
import kotlinx.coroutines.*
import retrofit2.HttpException
import kotlin.coroutines.coroutineContext

class ListViewModel: ViewModel() {

    val countriesService = CountriesService.getCountriesService()
    //코루틴의 라이프사이클 메서드에 접근하기 위해 Job 변수를 선언한다.
    var job : Job? = null
    //코루틴 예외 처리
    val exceptionHandler = CoroutineExceptionHandler{
        coroutineContext, throwable ->
        onError("Exception : ${throwable.localizedMessage}")
    }
//리스트를 받는다.
    val countries = MutableLiveData<List<Country>>()
//에러가 나는 경우 스트링 값, 그렇지 않은 경우의 스트링 값을 받는다.
    val countryLoadError = MutableLiveData<String?>()
//로딩이 되는 중이라면 true, 아니면 false 값을 받는다.
    val loading = MutableLiveData<Boolean>()

//swipe - refresh 레이아웃을 사용하므로
    fun refresh() {
        fetchCountries()
    }

    private fun fetchCountries() {
        loading.value = true
//네트워크 통신이므로 IO 디스패처를 사용한다.
        job = CoroutineScope(Dispatchers.IO + exceptionHandler).launch {
            val response = countriesService.getCountries()
//메인 스레드에 결과를 전달해 주기 위해서 메인 디스패처를 사용한다.
            withContext(Dispatchers.Main){
                if(response.isSuccessful){
                    countries.value = response.body()
                    countryLoadError.value = ""
                    loading.value = false
                }
                else{
                    onError("Error : ${response.message()}")
                }
            }
        }
    }

    private fun onError(message: String) {
//에러가 나는 경우 에러 메시지를 띄운다.
        countryLoadError.value = message
        loading.value = false
```

```
    }

    override fun onCleared() {
        super.onCleared()
//위에서 참조한 코루틴에 cancel 메서드를 호출한다.
        job?.cancel()
    }

}
```

- 메인 액티비티와 소통하는 뷰모델 클래스이다.
- 통신 결과에 따른 예외 처리, 성공했을 경우 데이터 처리 등을 해준다.

6. MainActivity 에서 위의 뷰모델 함수들을 호출하고, RecyclerView 를 셋팅한다.

```
package com.devtides.androidcoroutinesretrofit.view

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.View
import androidx.lifecycle.Observer
import androidx.lifecycle.ViewModelProviders
import androidx.recyclerview.widget.LinearLayoutManager
import com.devtides.androidcoroutinesretrofit.R
import com.devtides.androidcoroutinesretrofit.viewmodel.ListViewModel
import com.devtides.coroutinesretrofit.view.CountryListAdapter
import kotlinx.android.synthetic.main.activity_main.*

class MainActivity : AppCompatActivity() {

    lateinit var viewModel: ListViewModel
    private val countriesAdapter = CountryListAdapter(arrayListOf())

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        viewModel = ViewModelProviders.of(this).get(ListViewModel::class.java)
        viewModel.refresh()

        countriesList.apply {
//apply 함수로 리사이클러뷰를 셋팅해준다.
            layoutManager = LinearLayoutManager(context)
            adapter = countriesAdapter
        }

//observing 하는 것들을 한꺼번에 함수로 호출한다.
        observeViewModel()
    }

    fun observeViewModel() {
        viewModel.countries.observe(this, Observer {countries ->
            countries?.let {
                countriesList.visibility = View.VISIBLE
//어댑터에 넘겨주고 update 메서드 안에서 데이터셋이 바뀌는 처리를 해준다.
                countriesAdapter.updateCountries(it) }
        })

        viewModel.countryLoadError.observe(this, Observer { isError ->
//에러가 났을 때 "" 을 리턴받으면 텍스트뷰가 뜬다.
            list_error.visibility = if(isError == "") View.GONE else View.VISIBLE
        })

        viewModel.loading.observe(this, Observer { isLoading ->
            isLoading?.let {
                loading_view.visibility = if(it) View.VISIBLE else View.GONE
                if(it) {
                    list_error.visibility = View.GONE
                    countriesList.visibility = View.GONE
                }
            }
        })
    }
}
```

*** Repository 를 사용하지 않았음.

결과: