LEONARDO YONGUK KIM @ KAKAO BANK

PAGING PAGED LIST ADAPTER

가장 간단한 형태의 PAGEDLISTADAPTER만 다룹니다.

- PagedListAdapter를 어떻게 사용하는지만 다룹니다.
- PagedListAdapter의 다양한 기능을 다루지 않습니다.
- PagedListAdapter의 구조도 다루지 않습니다.
- 간단한 REST 예제로 시작합니다.

Tweet

Home About Documentation

Pokéapi V2 BETA

The RESTful Pokémon API

Over 221,236,600 API calls received!

Finally; all the Pokémon data you'll ever need, in one place, and easily accessible through a modern RESTful API.

Whats new in V2? Check out the docs!

Try it now!

http://pokeapi.co/api/v2/ pokemon/1/ submit

Need a hint? try *pokemon/1/* or *type/3/* or *ability/4/*

아래 JSON을 파싱합니다.

```json
{
  "count": 949,
  "previous": null,
  "results": [
    {
      "url": "https:\/\/pokeapi.co\/api\/v2\/pokemon\/1\/",
      "name": "bulbasaur"
    },
    ...
    {
      "url": "https:\/\/pokeapi.co\/api\/v2\/pokemon\/20\/",
      "name": "raticate"
    }
  ],
  "next": "https:\/\/pokeapi.co\/api\/v2\/pokemon\/?limit=20&offset=20"
}
```

RETROFIT을 하던대로 씁시다

```kotlin
data class Response(
        val count: Int,
        val previous: String,
        val next: String,
        val results: List<Result>
)

data class Result(
        val url: String,
        val name: String
)
```

```json
{
  "count": 949,
  "previous": null,
  "results": [
    {
      "url": "https:\/\/pokeapi.co\/api\
      "name": "bulbasaur"
    },
    ...
    {
      "url": "https:\/\/pokeapi.co\/api\
      "name": "raticate"
    }
  ],
  "next": "https:\/\/pokeapi.co\/api\/v2
}
```

RETROFIT을 하던대로 씁시다

```kotlin
interface PokeAPI {
    @GET("pokemon/")
    fun listPokemons(): Call<Response>

    @GET("pokemon/")
    fun listPokemons(
            @Query("offset") offset: String,
            @Query("limit") limit: String
    ): Call<Response>
}
```

데이터소스에게 무엇이 변경되었는지 알려줍니다. ITEMS은 아이디 비교, CONTENTS는 내용 비교.

```kotlin
private class DiffItemCallback : DiffUtil.ItemCallback<Result>() {
    override fun areItemsTheSame(oldItem: Result, newItem: Result): Boolean =
            oldItem.url == newItem.url

    override fun areContentsTheSame(oldItem: Result, newItem: Result): Boolean =
            oldItem.name == newItem.name && oldItem.url == newItem.url
}
```

이전(PREVIOUS) 페이지와 이후(NEXT) 페이지를 주목합시다.

```json
{
  "count": 949,
  "previous": null,
  "results": [
    {
      "url": "https:\/\/pokeapi.co\/api\/v2\/pokemon\/1\/",
      "name": "bulbasaur"
    },
    ...
    {
      "url": "https:\/\/pokeapi.co\/api\/v2\/pokemon\/20\/",
      "name": "raticate"
    }
  ],
  "next": "https:\/\/pokeapi.co\/api\/v2\/pokemon\/?limit=20&offset=20"
}
```

```kotlin
private class DataSource(private val pokeAPI: PokeAPI) : PageKeyedDataSource<String, Result>() {

    override fun loadInitial(params: LoadInitialParams<String>, callback: LoadInitialCallback<String,
Result>) {
        val body = pokeAPI.listPokemons().execute().body()
        callback.onResult(body!!.results, body.previous, body.next)
    }

    override fun loadBefore(params: LoadParams<String>, callback: LoadCallback<String, Result>) {
        val map = handleKey(params.key)
        val body = pokeAPI.listPokemons(map["offset"]!!, map["limit"]!!).execute().body()
        callback.onResult(body!!.results, body.previous)
    }

    override fun loadAfter(params: LoadParams<String>, callback: LoadCallback<String, Result>) {
        val map = handleKey(params.key)
        val body = pokeAPI.listPokemons(map["offset"]!!, map["limit"]!!).execute().body()
        callback.onResult(body!!.results, body.next)
    }

    private fun handleKey(key: String): MutableMap<String, String> {
        val (_, queryPart) = key.split("?")
        val queries = queryPart.split("&".toRegex()).dropLastWhile
{ it.isEmpty() }.toTypedArray()
        val map = mutableMapOf<String, String>()
        for (query in queries) {
            val (k, v) = query.split("=".toRegex()).dropLastWhile
{ it.isEmpty() }.toTypedArray()
            map[k] = v
        }
```

페이지드 리스트 어댑터에게 변경을 확인할 객체를 전달합니다.

```kotlin
private class Adapter : PagedListAdapter<Result, VieHolder>(DiffItemCallback()) {

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): VieHolder =
            VieHolder(LayoutInflater.from(parent.context).inflate(R.layout.item_recyclerview, parent,
false))

    override fun onBindViewHolder(holder: VieHolder, position: Int) {
        getItem(position)?.let { (_, name) ->
            holder.title = name
        }
    }
}
```

데이터소스를 위한 팩토리, 설정을 만든 후 라이브 데이터를 반환받습니다.

```kotlin
private fun createLiveData(): LiveData<PagedList<Result>> {
    val config = PagedList.Config.Builder()
            .setInitialLoadSizeHint(20)
            .setPageSize(20)
            .setPrefetchDistance(10)
            .build()

    return LivePagedListBuilder(object : android.arch.paging.DataSource.Factory<String,
Result>() {
        override fun create(): android.arch.paging.DataSource<String, Result> {
            return MainActivity.DataSource(pokeAPI)
        }
    }, config).build()
}
```

라이브데이터로 들어온 페이지드 리스트를 페이지드 리스트 어댑터에게 전달.

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
    recyclerView.apply {
        adapter = this@MainActivity.adapter
        layoutManager = LinearLayoutManager(this@MainActivity)
    }
    createLiveData().observe(this, Observer(adapter::submitList))
}
```

https://github.com/dalinaum/paged-list-adpater-demo/