Fast Android Networking

vdcn & lcc4

Estrutura da apresentação

1. Sobre o Fast Android Networking e por que usá-la
2. Demonstração.
3. Como é feito a conectividade sem/com a lib.

Sobre o Fast Android Networking e por que usá-la?

– – –

- Lib feita sobre o OkHttp e suporta HTTP/2.
- Visa simplificar com customização o uso de rede em android.
- Remoção recente do HttpClient no Android M deixou outras libs obsoletas.
- Dizem que são os mais completos.
- Nenhuma outra lib fornece uma interface simples para uso de prioridade, cancelamento etc.
- Usa Okio e, por isso, previne overhead do GC em aplicações android.

Demonstração

mostrar get request com json

*mostrar como implementar esse objeto

mostrar get request com objeto

mostrar post

mostrar post com objeto

mostrar download

mostrar

Get JSONArray as Response

```
AndroidNetworking.get("https://fierce-cove-29863.herokuapp.com/getAllUsers/{pageNumber}")
                 .addPathParameter("pageNumber", "0")
                 .addQueryParameter("limit", "3")
                 .setPriority(Priority.LOW)
                 .build()
                 .getAsJSONArray(new JSONArrayRequestListener() {
                    @Override
                    public void onResponse(JSONArray response) {
                      // do anything with response
                    }
                    @Override
                    public void onError(ANError error) {
                      // handle error
                    }
                 });
```

JSON Object

```
data class User (
    val id : Int,
    val firstname : String,
    val lastname : String
) {
    override fun toString(): String {
        var ret = "id: " + id.toString() + "\n" +
                  "firstname: " + firstname + "\n" +
                   "lastname: " + lastname + "\n"
        return ret
    }
}
```

Get parsed object as a response

```
AndroidNetworking.get("https://fierce-cove-29863.herokuapp.com/getAnUserDetail/{userId}")
                .addPathParameter("userId", "1")
                .setTag(this)
                .setPriority(Priority.LOW)
                .build()
                .getAsObject(User.class, new ParsedRequestListener<User>() {
                    @Override
                    public void onResponse(User user) {
                       // do anything with response
                       Log.d(TAG, "id : " + user.id);
                       Log.d(TAG, "firstname : " + user.firstname);
                       Log.d(TAG, "lastname : " + user.lastname);
                    }
                    @Override
                    public void onError(ANError anError) {
                       // handle error}});
```

Post

```
AndroidNetworking.post("https://fierce-cove-29863.herokuapp.com/createAnUser")
                 .addBodyParameter("firstname", "Amit")
                 .addBodyParameter("lastname", "Shekhar")
                 .setPriority(Priority.MEDIUM)
                 .build()
                 .getAsJSONArray(new JSONArrayRequestListener() {
                    @Override
                    public void onResponse(JSONArray response) {
                      // do anything with response
                    }
                    @Override
                    public void onError(ANError error) {
                      // handle error
                    }
                 });
```

Posting object

```
User user = new User();
user.firstname = "Amit";
user.lastname = "Shekhar";

AndroidNetworking.post("https://fierce-cove-29863.herokuapp.com/createUser")
                 .addBodyParameter(user) // posting java object
                 .setTag("test")
                 .setPriority(Priority.MEDIUM)
                 .build()
                 .getAsJSONArray(new JSONArrayRequestListener() {
                    @Override
                    public void onResponse(JSONArray response) {
                      // do anything with response
                    }
                    @Override
                    public void onError(ANError error) {
                      // handle error
                    }});
```

Download

```
AndroidNetworking.download(url,dirPath,fileName)
                 .setTag("downloadTest")
                 .setPriority(Priority.MEDIUM)
                 .build()
                 .setDownloadProgressListener(new DownloadProgressListener() {
                    @Override
                    public void onProgress(long bytesDownloaded, long totalBytes) {
                      // do anything with progress
                    }
                  })
                 .startDownload(new DownloadListener() {
                    @Override
                    public void onDownloadComplete() {
                      // do anything after completion
                    }
                    @Override
                    public void onError(ANError error) {
                      // handle error
                    }});
```

Demo

Comparativos

```
val root = getExternalFilesDir(DIRECTORY_DOWNLOADS)
if (root != null) {
    Log.d("DownloadService", "" + root.path.toString())
}
root?.mkdirs()
val output = File(root, i!!.data!!.lastPathSegment)
if (output.exists()) {
    output.delete()
}
val url = URL(i.data!!.toString())
val c = url.openConnection() as HttpURLConnection
val fos = FileOutputStream(output.path)
val out = BufferedOutputStream(fos)
try {
    val `in` = c.inputStream
    val buffer = ByteArray(8192)
    var len = `in`.read(buffer)
    while (len >= 0) {
        out.write(buffer, 0, len)
        len = `in`.read(buffer)
    }
    out.flush()
} finally {
    fos.fd.sync()
    out.close()
    c.disconnect()
}
```

Podcast....

```
val root = getExternalFilesDir(DIRECTORY_DOWNLOADS)
if (root != null) {
    Log.d("DownloadService", "" + root.path.toString())
}
root?.mkdirs()
val output = File(root, i!!.data!!.lastPathSegment)
if (output.exists()) {
    output.delete()
}
val url = URL(i.data!!.toString())
val c = url.openConnection() as HttpURLConnection
val fos = FileOutputStream(output.path)
val out = BufferedOutputStream(fos)
try {
    val `in` = c.inputStream
    val buffer = ByteArray(8192)
    var len = `in`.read(buffer)
    while (len >= 0) {
        out.write(buffer, 0, len)
        len = `in`.read(buffer)
    }
    out.flush()
} finally {
    fos.fd.sync()
    out.close()
    c.disconnect()
}
```

Podcast....

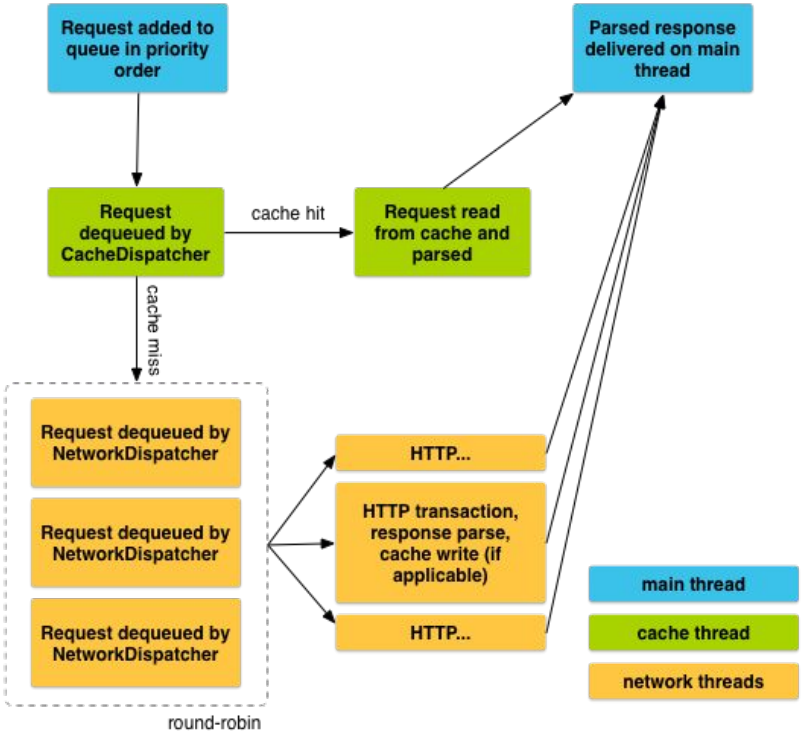
MEMLEAK!!!!

```
class NetworkFragment : Fragment() {
    private var callback: DownloadCallback<String>? = null
    private var downloadTask: DownloadTask? = null
    private var urlString: String? = null

    (...)

    override fun onDetach() {
        super.onDetach()
        // Clear reference to host Activity to avoid memory leak.
        callback = null
    }
}
```

Volley


Request added to queue in priority order
Parsed response delivered on main thread
Request dequeued by CacheDispatcher
cache hit
Request read from cache and parsed
cache miss
Request dequeued by NetworkDispatcher
Request dequeued by NetworkDispatcher
Request dequeued by NetworkDispatcher
HTTP...
HTTP transaction, response parse, cache write (if applicable)
HTTP...
round-robin
main thread
cache thread
network threads

Get request

```
val textView = findViewById<TextView>(R.id.text)
// ...

// Instantiate the RequestQueue.
val queue = Volley.newRequestQueue(this)
val url = "http://www.google.com"

// Request a string response from the provided URL.
val stringRequest = StringRequest(Request.Method.GET, url,
        Response.Listener<String> { response ->
            // Display the first 500 characters of the response string.
            textView.text = "Response is: ${response.substring(0, 500)}"
        },
        Response.ErrorListener { textView.text = "That didn't work!" })

// Add the request to the RequestQueue.
queue.add(stringRequest)
```

Cancel requests

```
val TAG = "MyTag"
val stringRequest: StringRequest // Assume this exists.
val requestQueue: RequestQueue? // Assume this exists.

// Set the tag on the request.
stringRequest.tag = TAG

// Add the request to the RequestQueue.
requestQueue?.add(stringRequest)

(...)

protected fun onStop() {
    super.onStop()
    requestQueue?.cancelAll(TAG)
}
```

Baixar um arquivo usando o FAN

```
AndroidNetworking.download(url,dirPath,fileName)
                 .setTag("downloadTest")
                 .setPriority(Priority.MEDIUM)
                 .build()
                 .setDownloadProgressListener(new DownloadProgressListener() {
                    @Override
                    public void onProgress(long bytesDownloaded, long totalBytes) {
                      // do anything with progress
                    }
                 })
                 .startDownload(new DownloadListener() {
                    @Override
                    public void onDownloadComplete() {
                      // do anything after completion
                    }
                    @Override
                    public void onError(ANError error) {
                      // handle error
                    }
                });
```

Custom requestQueue: network and cache

```
// Instantiate the cache
val cache = DiskBasedCache(cacheDir, 1024 * 1024) // 1MB cap

// Set up the network to use HttpURLConnection as the HTTP
client.
val network = BasicNetwork(HurlStack())

// Instantiate the RequestQueue with the cache and network. Start
the queue.
val requestQueue = RequestQueue(cache, network).apply {
    start()
}
```

Caching in FAN

```
.doNotCacheResponse()
.getResponseOnlyIfCached()
.getResponseOnlyFromNetwork()
.setMaxAgeCacheControl(0, TimeUnit.SECONDS)
.setMaxStaleCacheControl(365, TimeUnit.SECONDS)
```

Prefetch a GET request

```
AndroidNetworking.get(url)
        .addPathParameter("pageNumber", "0")
        .addQueryParameter("limit", "3")
        .build()
        .prefetch();
```

Set timeout globally

```
OkHttpClient okHttpClient = new OkHttpClient().newBuilder()
                .connectTimeout(120, TimeUnit.SECONDS)
                .readTimeout(120, TimeUnit.SECONDS)
                . writeTimeout(120, TimeUnit.SECONDS)
                .build();

AndroidNetworking.initialize(getApplicationContext(),okHttpClient);
```

Timeout for each request

```
OkHttpClient okHttpClient = new OkHttpClient().newBuilder()
                    .connectTimeout(120, TimeUnit.SECONDS)
                    .readTimeout(120, TimeUnit.SECONDS)
                    . writeTimeout(120, TimeUnit.SECONDS)
                    .build(); // set this okHttpClient into your request
```

Pontos NEGATIVOS

- Péssima documentação
- Exemplos quebrados
- Não há explicação sobre alguns parâmetros
- “Pouco” utilizada → poucas pessoas usando → falta de material para aprender a usar
- Não responde issues do seu repositório

Pontos POSITIVOS

- Não precisamos nos preocupar com memory leak
- Cada request podemos setar individualmente o tempo de timeout (connection, read, write)
- Caching
- Executor
- Prefetch
- Logging usando um Interceptor

Obrigado!