Network Android Libraries

Substituições para o robospice

Possíveis escolhas:

Android Volley: Introdução

Volley é uma biblioteca HTTP que facilita a criação de redes para aplicações Android.

Volley destaca-se em operações do tipo RPC usadas para preencher o UI. Integra-se facilmente a qualquer protocolo e possui suporte para strings, imagens e JSON.

Android Volley: Vantagens e Desvantagens

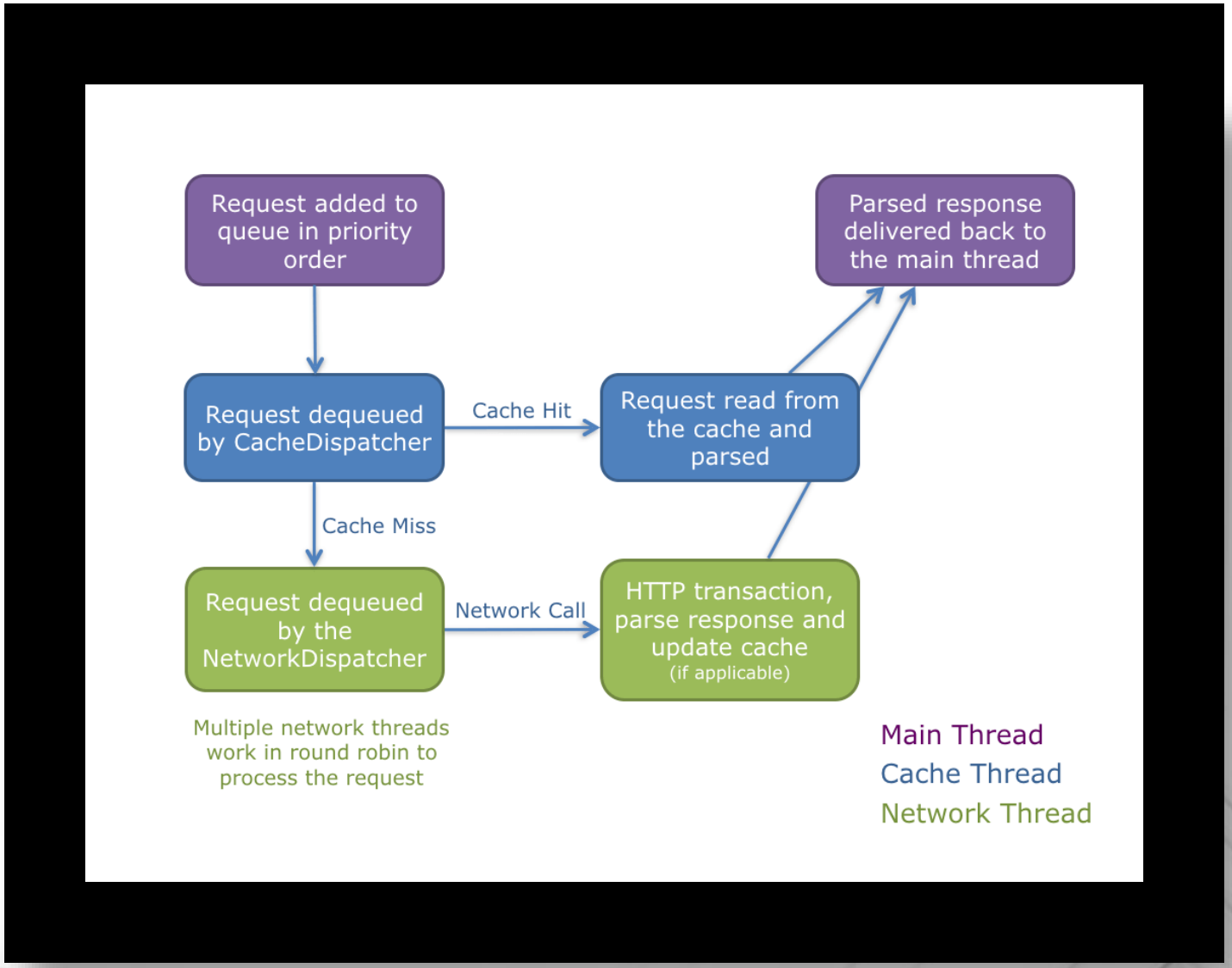
Vantagens

- Fila de Pedidos implementada com prioridades
- Possui mecanismo para nova tentativa de pedido
- Possui sistema de cache
- Possui feature para loading de imagens

Desvantagens

- Code para configuração de pedido mais complexa comparativamente ao retrofit
- Devido ao uso da memoria não é aconselhavel para grandes operações de dowload ou de streaming

Volley Request Life-Cycle



Retrofit: Introdução

Retrofit é uma biblioteca REST Client (Helper Library), criada pela Square e usada no Android e Java para criar um pedido HTTP. Processa resposta HTTP de uma API REST e também pode ser usado para receber estruturas de dados que não sejam JSON, por exemplo, SimpleXML e Jackson.

Retrofit:
Vantagens e Desvantagens

Vantagens:

- Muito simples de usar e de implementar
- Consegue receber qualquer tipo de pedido desde de Strings a Json Objects
- Auto parsing de JsonObject e JsonArray

Desvantagens:

- Não possui fila para pedidos prioritários
- Apenas corre com OkHttp
- Não possui sistema de cache
- Nova tentativa de pedido é feita colunando o pedido

Comparações

Criar RequestQueue Volley

Default Queue

```
RequestQueue queue = Volley.newRequestQueue
        (MainActivity.getApplicationContext());
queue.start();
```

Queue with cache

```
RequestQueue requestQueue;

Cache cache = new DiskBasedCache(MainActivity.getCacheDir()
        , maxCacheSizeInBytes: 1024 * 1024); // 1MB cap

Network network = new BasicNetwork(new HurlStack());

requestQueue = new RequestQueue(cache,network);

requestQueue.start();
```

Criar Interface Retrofit

```
class MyService {
    static final String API_URL =
            "http://mobile-montepio.itsector.local/";

    public interface Request {

        @POST("/public/contentByGroup")
        Call<ResponseContent> getContent(
                @HeaderMap Map<String,String> headers,
                @Body JsonObject objectToSend
                );
    }
}
```

```
MyService.Request request;

Retrofit retrofit = new Retrofit.Builder().baseUrl(MyService.API_URL).
        addConverterFactory(GsonConverterFactory.create()).build();

request = retrofit.create(MyService.Request.class);
```

Criar e chamar um Pedido Assíncrono Volley

```
final ContentToSend contentToSend = new ContentToSend( marketing: "EXAMPLE");
final Gson gson = new GsonBuilder().setPrettyPrinting().create();
JSONObject contentToSendJson = null;
try {
    contentToSendJson = new JSONObject(gson.toJson(contentToSend));
} catch (JSONException e) {
    e.printStackTrace();
}
String url = "http://exemple.com";

JsonObjectRequest request =
```

```
JsonObjectRequest request =
        new JsonObjectRequest(Request.Method.POST, url, contentToSendJson,
                new Response.Listener<JSONObject>() {
                    @Override
                    public void onResponse(JSONObject response) {
                        final ResponseContent result;
                        result = gson.fromJson(response.toString(), ResponseContent.class);
                        //Do something in Ui
                    }
                }, new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
                //Do something
            }
        }) {
            @Override
            public Map<String, String> getHeaders() { return headers; }
        };

queue.start();
queue.add(request);
```

Criar e chamar um Pedido Assíncrono Retrofit

```
protected JsonObject jsonToSend(){
    ContentToSend contentToSend = new ContentToSend( ContentModule: "MARKETING");
    Gson gson = new Gson();
    return new JsonParser().parse(gson.toJson(contentToSend)).getAsJsonObject();
}

protected void performRequest(Map<String, String> headers) {

    Call<ResponseContent> call;

    call = request.getContent(headers,jsonToSend());
    call.enqueue(new Callback<ResponseContent>() {
        @Override
        public void onResponse(@NotNull Call<ResponseContent> call, @NotNull Response<ResponseContent> response) {
            assert response.body() != null;

            final List<ResponseContent.ResponseContentResult> result =
                    response.body().getResult().getContentResult();
            //Do Something
        }
        @Override
        public void onFailure(@NotNull Call<ResponseContent> call, @NotNull Throwable t) {
            //Do Something
        }
    });
```

Criar e chamar um Pedido Síncrono Volley

```
final ContentToSend contentToSend = new ContentToSend( ContentModule: "MARKETING");
final Gson gson = new GsonBuilder().setPrettyPrinting().create();
JSONObject contentToSendJson = null;
try {
    contentToSendJson = new JSONObject(gson.toJson(contentToSend));
} catch (JSONException e) {
    e.printStackTrace();
}
String url = "http://mobile-montepio.itsector.local/public/contentByGroup";

//Synchronous Request
```

```
final RequestFuture<JSONObject> requestFuture = RequestFuture.newFuture();

JsonObjectRequest request =
        new JsonObjectRequest(Request.Method.POST, url, contentToSendJson,requestFuture, requestFuture) {
            @Override
            public Map<String, String> getHeaders() { return headers; }
        };

queue.start();
queue.add(request);

requestFuture.setRequest(request);
```

```
@SuppressLint("StaticFieldLeak") AsyncTask task =
        new AsyncTask<Object, Void,ResponseContent>() {

            @Override
            protected ResponseContent doInBackground(Object... objects) {
                String response = null;
                try {
                    response = requestFuture.get( timeout: 10,TimeUnit.SECONDS).toString();
                } catch (InterruptedException e) {
                    e.printStackTrace();
                } catch (ExecutionException e) {
                    e.printStackTrace();
                } catch (TimeoutException e) {
                    e.printStackTrace();
                }

                ResponseContent responseContent = gson.fromJson(response,ResponseContent.class);

                return responseContent;
            }

            @Override
            protected void onPostExecute(ResponseContent responseContent){
                request_response_get = SystemClock.elapsedRealtime();
                initUI(responseContent);
            }
        };

task.execute();
```

Criar e chamar um Pedido Síncrono Retrofit

```
protected JsonObject jsonToSend(){
    ContentToSend contentToSend = new ContentToSend( ContentModule: "MARKETING");
    Gson gson = new Gson();
    return new JsonParser().parse(gson.toJson(contentToSend)).getAsJsonObject();
}

protected void performRequest(Map<String, String> headers) {

    Call<ResponseContent> call;

    call = request.getContent(headers,jsonToSend());
```

```
@SuppressLint("StaticFieldLeak") AsyncTask<Void,Void,ResponseContent> task =
        new AsyncTask<Void, Void, ResponseContent>() {
    @Override
    protected ResponseContent doInBackground(Void... voids) {
        ResponseContent content = null;
        try {
            content = call.execute().body();
        } catch (IOException e) {
            e.printStackTrace();
            onRestart();
        }
        return content;
    }
            @Override
            protected void onPostExecute(ResponseContent responseContent) {

                request_response_get_timer = SystemClock.elapsedRealtime();
                initUI(responseContent.getResult().getContentResult());
            }
};

task.execute();
```

AsyncTask

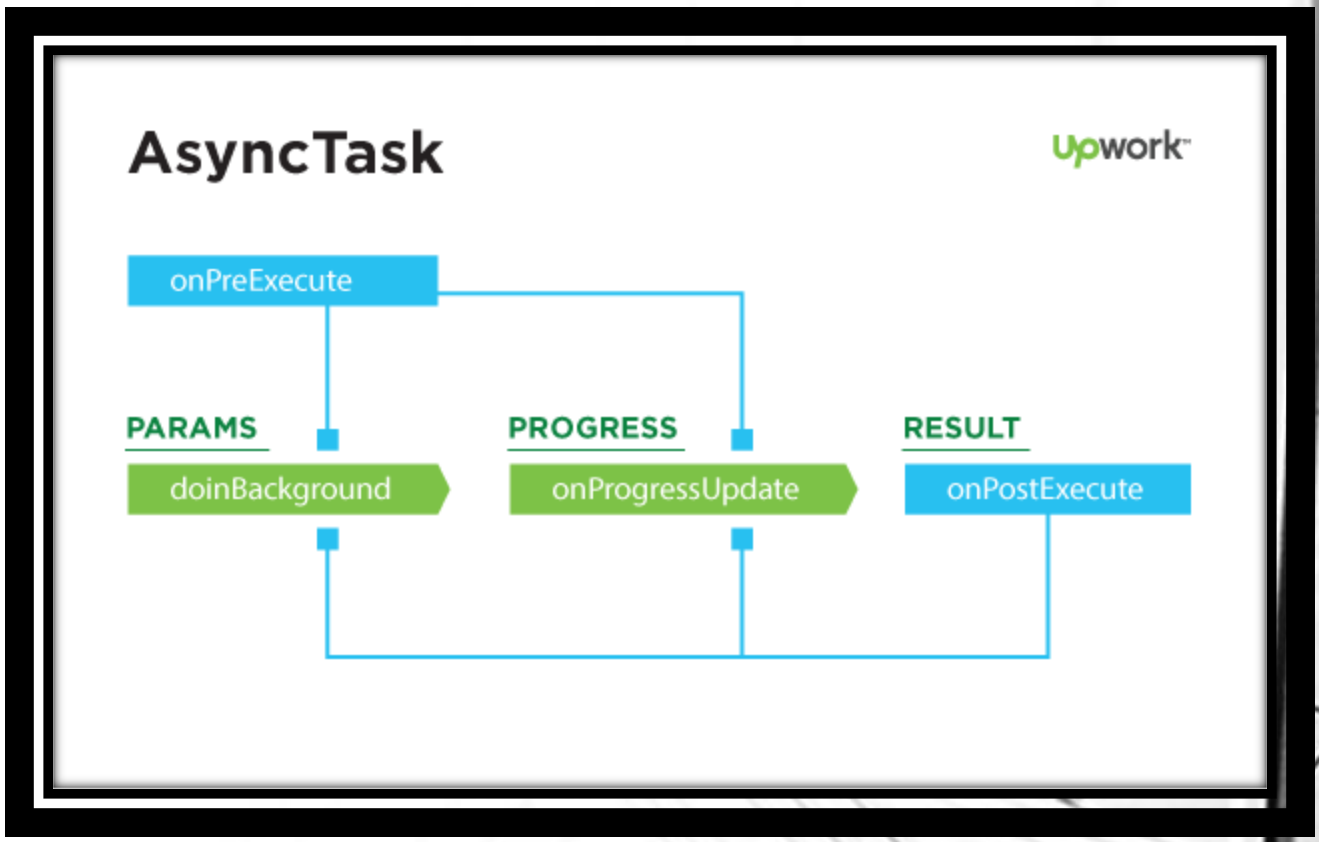
```
class DoSomeTask extends AsyncTask<String, Integer, String>{

    @Override
    protected void onPreExecute() {
        //Setup precondition to execute some task
    }

    @Override
    protected String doInBackground(String... params) {
        //Do some task
        publishProgress (1);
        return "";
    }

    @Override
    protected void onProgressUpdate(Integer... values) {
        //Update the progress of current task
    }

    @Override
    protected void onPostExecute(String s) {
        //Show the result obtained from doInBackground
    }
}
```



Conclusões

Referencias

- https://developer.android.com/training/volley
- https://square.github.io/retrofit/
- https://futurestud.io/tutorials/tag/retrofit
- https://github.com/RicardoAz15/NetWork
- https://www.smashingmagazine.com/2017/03/simplify-android-networking-volley-http-library/