PROGRAMAÇÃO ANDROID

Configuração do Retrofit

Universidade Federal de Sergipe
Departamento de Sistemas de Informação
Prof. Andrés Menéndez
ammenendez@gmail.com

Previously...

- Temos que criar uma interface para fazer o mapeamento da chamada HTTP

```
public interface GithubUserAPI {

    @GET("/users/{usuario}")
    Call<Usuario> getUsuario(@Path("usuario") String usuario);

    public static final Retrofit retrofit = new Retrofit.Builder()
            .baseUrl("https://api.github.com/")
            .addConverterFactory(GsonConverterFactory.create())
            .build();
}
```

Retrofit

- Podemos fazer a chamada de duas formas
  - Síncrona
  - Assíncrona
- Vejamos inicialmente a chamada síncrona

Retrofit

- Muito simples!
- O problema deste código é que não funciona

```
Button sincrono = (Button) findViewById(R.id.btnSincrono);
sincrono.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        GithubUserAPI githubUser = GithubUserAPI.retrofit.create(GithubUserAPI.class);
        final Call<Usuario> call = githubUser.getUsuario("ammenendez");
        try {
            Usuario usuario = call.execute().body();
            Toast.makeText(getBaseContext(), "Nome do usuário: " + usuario.name,
                Toast.LENGTH_LONG).show();
        } catch (IOException e) {
            Toast.makeText(getBaseContext(), e.getMessage(), Toast.LENGTH_LONG).show();
        }
    }
});
```

Retrofit

- O problema não é do código em si, mas uma restrição da plataforma Android
- Não podemos fazer uma chamada ”longa” pois isto poderá trazer uma experiência desagradável ao usuário

```
09-15 14:39:16.748 10834-10834/br.ufs.tep.retrofit D/AndroidRuntime: Shutting down VM
09-15 14:39:16.757 10834-10834/br.ufs.tep.retrofit E/AndroidRuntime: FATAL EXCEPTION: main
                                                                     Process: br.ufs.tep.retrofit, PID: 10834
                                                                     android.os.NetworkOnMainThreadException
```

- E agora?
  - Precisamos tornar a chamada assíncrona

Retrofit

- No Retrofit é simples tornar a chamada assíncrona
- Ao invés de chamar como vimos anteriormente:

```
Usuario usuario = call.execute().body();
```

- Chamamos assim

```
call.enqueue(new Callback<Usuario>() {
    @Override
    public void onResponse(Call<Usuario> call, Response<Usuario> response) {

        }
    }
    @Override
    public void onFailure(Call<Usuario> call, Throwable t) {

    }
});
```

Retrofit

- Vejamos como ficaria o código completo sendo disparado pelo click de um Button

```
Button carga = (Button) findViewById(R.id.btnCarregar);
carga.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        GithubUserAPI githubUser = GithubUserAPI.retrofit.create(GithubUserAPI.class);
        final Call<Usuario> call = githubUser.getUsuario("ammenendez");
        call.enqueue(new Callback<Usuario>() {
```

Retrofit

```
        @Override
        public void onResponse(Call<Usuario> call, Response<Usuario> response) {
            int code = response.code();
            if (code == 200) {
                Usuario usuario = response.body();
                Toast.makeText(getBaseContext(), "Nome do usuário: " +
                        usuario.name, Toast.LENGTH_LONG).show();
            } else {
                Toast.makeText(getBaseContext(), "Falha: " + String.valueOf(code),
                        Toast.LENGTH_LONG).show();
            }
        }

        @Override
        public void onFailure(Call<Usuario> call, Throwable t) {
            Toast.makeText(getBaseContext(), t.getMessage(),
                    Toast.LENGTH_LONG).show();
        }
    });
  }
});
```

Retrofit

- Agora vamos chamar o método que retorna vários objetos
  - https://api.github.com/users/ammenendez/followers
- Neste caso o método definido na interface será feito da seguinte forma:

```
@GET("/users/{usuario}/followers")
Call<List<Usuario>> getSeguidores(@Path("usuario") String usuario);
```

- Note que o retorno será do tipo List<Usuario>

Retrofit

- Vejamos como fica a chamada ao web service

```
call.enqueue(new Callback<List<Usuario>>() {
    @Override
    public void onResponse(Call<List<Usuario>> call,
                           Response<List<Usuario>> response) {
        List<Usuario> lista = response.body();
        for (Usuario usuario : lista) {
            Log.d("MainActivity", usuario.login);
        }

    }

    @Override
    public void onFailure(Call<List<Usuario>> call, Throwable t) {


    }
```