



Análise e Desenvolvimento de Sistemas

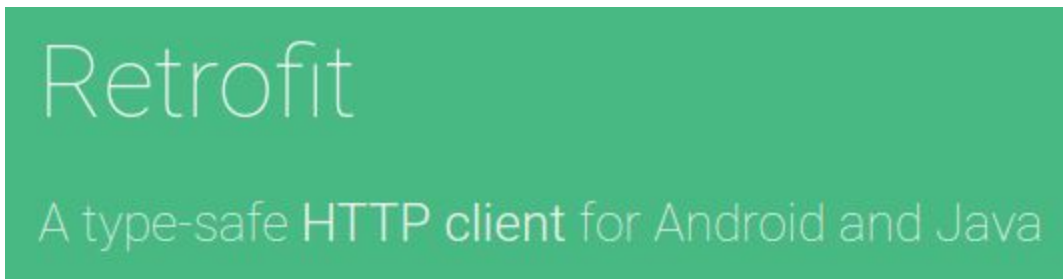
Desenvolvimento Mobile e IOT - Android

Integração backend - JSON e Cliente RESTFul - Retrofit



Prof. Douglas Cabral <douglas.cabral@fiap.com.br>
<https://www.linkedin.com/in/douglascabral/>

Retrofit é um client REST para Android que permite uma grande interação com APIs e o envio de requisições através do protocolo HTTP.



De uma forma simples, o Retrofit realiza o download do conteúdo da API em formato JSON ou XML e então realiza a conversão para um objeto POJO (Plain Old Java Object).

OBS: Não esquecer de inserir a permissão para acessar a internet

Setup:

Inserir as dependências necessárias do Retrofit em nosso arquivo Gradle

```
compile 'com.google.code.gson:gson:2.7'
compile 'com.squareup.retrofit2:retrofit:2.2.0'
compile 'com.squareup.retrofit2:converter-gson:2.2.0'
```

Inserir permissão no AndroidManifest.xml para acesso da internet

```
<uses-permission android:name="android.permission.INTERNET" />
```

Setup:

Criar um POJO representando a entidade retornada pela API. (Nosso exemplo será um Contato)

```
public class Contato implements Serializable {  
    private String nome;  
    private String telefone;  
    private String status;  
    private String imagem;  
  
    //GET e SET aqui....  
}
```

Após a execução do setup, precisamos criar uma instância do Retrofit, informando qual será a URL base da API. Por questões didáticas, vamos fazer em nossa **MainActivity**.

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("https://douglascabral.com.br/aula-api/")
    .addConverterFactory(GsonConverterFactory.create())
    .build();
```

Como organização, o Retrofit organiza todas as rotas de sua API dentro de uma interface, no qual, através de **Annotations** descrevemos quais serão os parâmetros e método da requisição.

Exemplo:

```
public interface MyApiEndpointInterface {  
    // Request method and URL specified in the annotation  
    // Callback for the parsed response is the last parameter  
  
    @GET("users/{username}")  
    Call<User> getUser(@Path("username") String username);  
  
    @GET("group/{id}/users")  
    Call<List<User>> groupList(@Path("id") int groupId, @Query("sort") String sort);  
  
    @POST("users/new")  
    Call<User> createUser(@Body User user);  
}
```

Em nosso projeto, vamos criar uma interface chamada **RetrofitContatosInterface** que possuirá um método **getContatos** anotado com **@GET("contatos.php")**.

```
public interface RetrofitContatosInterface {  
    @GET("contatos.php")  
    Call<List<Contato>> getContatos();  
}
```



Em nossa **MainActivity** vamos fazer nossa integração funcionar e então, retornar os dados, exibindo-os no console.

Passos:

Após a instanciação do Retrofit, criar uma variável chamada **api** (ou o nome que preferir) do tipo **RetrofitContatosInterface**. Exemplo:

```
RetrofitContatosInterface api = retrofit.create(RetrofitContatosInterface.class);
```

```
public class MainActivity extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        Retrofit retrofit = new Retrofit.Builder()  
            .baseUrl("https://douglascabral.com.br/aula-api/")  
            .addConverterFactory(GsonConverterFactory.create())  
            .build();  
  
        RetrofitContatosInterface api = retrofit.create(RetrofitContatosInterface.class);  
    }  
}
```



Com nosso objeto instanciado, basta chamarmos o método **getContatos()** e então em seguida o método **enqueue(<Callback>)** que de forma assíncrona, irá retornar um **List<Contato>** dentro de um callback que possuirá um método para sucesso (**onResponse**) e um em caso de falha (**onFailure**).

```
RetrofitContatosInterface api = retrofit.create(RetrofitContatosInterface.class);

api.getContatos().enqueue(new Callback<List<Contato>>() {
    @Override
    public void onResponse(Call<List<Contato>> call, Response<List<Contato>> response) {
        //Lógica em caso de sucesso
    }

    @Override
    public void onFailure(Call<List<Contato>> call, Throwable t) {
        //Lógica em caso de erro
    }
});
```

Incluir a seguinte lógica dentro de **onResponse**:

```
List<Contato> contatos = response.body();  
for ( int i = 0; i < contatos.size(); i++) {  
    Log.i("CONTATO", contatos.get(i).getNome());  
}
```

O Retrofit é bastante flexível, permitindo realizar GET, POST, DELETE, HEAD... e permite o envio de dados pelo header ou body da requisição, como querystring ou substituindo uma parte da URL do endpoint por uma outra informação.

Exemplos:

```
@Headers({"Cache-Control: max-age=640000", "User-Agent: My-App-Name"})
```

```
@GET("/some/endpoint")
```

```
@Multipart
```

```
@POST("/some/endpoint")
```

```
Call<SomeResponse> someEndpoint(@Header("Cache-Control") int maxAge)
```

Para saber mais sobre todas as opções disponíveis:

<https://square.github.io/retrofit/>

https://github.com/codepath/android_guides/wiki/Consuming-APIs-with-Retrofit

<https://github.com/square/retrofit>

Com os dados retornados pela API de exemplo nesta apresentação, liste-os em um ListView.

Dica:

Para baixar a imagem informada pela API de maneira fácil, use uma biblioteca chamada Picasso.

<https://square.github.io/picasso/>



Copyright © 2017 Prof. Douglas Cabral <douglas.cabral@fiap.com.br> <https://www.linkedin.com/in/douglascabral/>

Todos direitos reservados. Reprodução ou divulgação total ou parcial deste documento é expressamente proibido sem o consentimento formal, por escrito, do Professor (autor).