

# ANDROID

Web Services



## Conteúdo

- Web Services – REST e JSON
- Cliente REST - Retrofit





# Web Services

## Web Services

- Integrar a aplicação com uma aplicação servidora na Web
  - Não ter acesso direto ao seu Banco de Dados
  - Ter uma interface com serviços disponibilizados
- REST
  - Web Services criados sobre o protocolo HTTP
  - Utilizam os métodos GET, POST, PUT, DELETE para o acesso
  - Utilizam formatos XML ou JSON para transitar informações





# RETROFIT

## Retrofit

- Biblioteca para criar consumidores REST
  - Passos para o uso:
    - Adicionar permissão para uso da Internet no MANIFEST
- ```
<uses-permission android:name="android.permission.INTERNET" />
```
- Adicionar o Retrofit e Conversor Gson no `build.gradle`
- ```
compile 'com.squareup.retrofit2:retrofit:2.1.0'  
compile 'com.squareup.retrofit2:converter-gson:2.1.0'
```
- Criar classes para representação dos dados
    - Beans, setters/getters
  - Definir uma interface para o serviço
    - Contém os métodos que serão invocados e o método HTTP, parâmetros, etc para invocação do WS
    - Estes métodos são chamados no código



# Retrofit

- Invocar os métodos do Retrofit para chamar o WS
  - `Retrofit.Builder()`
  - `retrofit.create()`
  - `API.métodoDalInterface()`
  - `call.enqueue(Callback)` : Faz a chamada de forma assíncrona
  - Tratar os resultados no Callback: retorno HTTP e dados
    - `Callback.onResponse()` – retorno com sucesso
    - `Callback.onFailure()` – retorno com erro
  - `call.execute()`: Faz a chamada de forma síncrona
    - Retorno é um Response<Tipo>



## Retrofit – Exemplo Bean

```
public class Boi {  
    private int id;  
    private String nome;  
    private String raca;  
  
    // setters/getters  
}
```



## Retrofit – Exemplo Interface (API)

```
import retrofit2.Call;
import retrofit2.http.GET;

public interface BoiService {
    @GET("boi")
    Call<Boi> carregarBoi();
}
```



## Retrofit – Métodos do Retrofit

```
import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;
import retrofit2.Retrofit;
...
Retrofit retrofit = new Retrofit.Builder()
        .baseUrl("http://www.razer.net.br/app/")
        .addConverterFactory(GsonConverterFactory.create())
        .build();
BoiService boiService = retrofit.create(BoiService.class);
Call<Boi> call = boiService.carregarBoi();

// chama enqueue/execute para tratar o resultado
```



## Retrofit – Métodos do Retrofit

```
call.enqueue(new Callback<Boi>() {
    // Se sucesso
    public void onResponse(Call<Boi> call, Response<Boi> response) {
        if (response.isSuccessful()) {
            Boi boi = response.body();
            // ...
        }
        else {
            System.out.println(response.errorBody());
        }
    }
    // Se falha
    public void onFailure(Call<Boi> call, Throwable t) {
        t.printStackTrace();
    }
});
```



## Retrofit – Métodos do Retrofit

```
Response<Boi> response = call.execute();
int status = response.code();
if (response.isSuccessful()) {    // código de 200-300
    Boi boi = response.body();
}
```



## Retrofit – Acesso à rede em outra Thread

```

new Thread() {
    public void run() {
        try {
            Retrofit retrofit = new Retrofit.Builder()
                .baseUrl("http://razer.net.br/app/")
                .addConverterFactory(GsonConverterFactory.create())
                .build();
            BoiService boiService = retrofit.create(BoiService.class);
            Call<Boi> call = boiService.carregarBoi();

            Response<Boi> resp = call.execute();
            Boi boi = resp.body();
        }
        catch(IOException e) {
            Log.e("APP_BOI", "Erro obtendo Boi: " + e.getMessage());
        }
    }
};

```



## Retrofit – Outros Métodos do HTTP

- Pode-se usar também:
  - Passagem de parâmetros na invocação
  - Receber código HTTP de retorno
  - GET : retornando uma lista
  - POST : para inserir uma informação, passando um JSON no corpo da mensagem
  - PUT : para atualizar uma informação, passando um JSON no corpo da mensagem
  - DELETE : para remover uma informação



## Retrofit – Passagem de Parâmetros

- No caminho dentro de @GET usa-se "{" e "}" para delimitar um parâmetro
- Nos parâmetros do método definido, usa-se @Path para referenciar o parâmetro

```
public interface BoiService {  
    @GET("boi/{id}")  
    Call<Boi> carregarBoi(@Path("id") int id);  
}
```



## Retrofit – Passagem de Parâmetros

- A invocação fica:

```
...  
Call<Boi> call = boiService.carregarBoi(10);  
...
```



## Retrofit – Passagem de Parâmetros

- Mais de um parâmetro

```
public interface BoiService {  
    @GET("boi/{nome}/{raca}")  
    Call<Boi> carregarBoi(@Path("nome") String nome  
                           @Path("raca") String raca);  
}
```



## Retrofit – Passagem de Parâmetros

- A invocação fica:

```
...  
Call<Boi> call = boiService.carregarBoi("mimosa", "holandesa");  
...
```



## Retrofit – Código de Retorno HTTP

- No Callback `onResponse()`, o objeto `Response` encapsula o retorno
- No retorno de `execute()` também se obtém um objeto `Response`
- Métodos
  - `response.isSuccessful()` : Se o retorno foi 2XX-3XX
  - `response.code()` : Retorna o status code

```
public void onResponse(Call<Boi> call, Response<Boi> response) {
    if (response.isSuccessful()) {
        // ...
    }
    else {
        if (response.code() == 400) {
            // ...
        }
        else {
        }
    }
}
```



## Retrofit – GET com Lista

- Definição da Interface

```
public interface BoiService {
    @GET("boi/{id}")
    Call<Boi> carregarBoi(@Path("id") int id);
    @GET("boi")
    Call<List<Boi>> carregarTodos();
}
```



## Retrofit – GET com Lista

- Invocação

```
...
Call<List<Boi>> call = boiService.carregarTodos();
...
```



## Retrofit – GET com Lista

- Callback

```
call.enqueue(new Callback<List<Boi>>() {
    // Se sucesso
    public void onResponse(Call<List<Boi>> call, Response<List<Boi>> response) {
        if (response.isSuccessful()) {
            List<Boi> lista = response.body();
            // ...
        }
        else {
            System.out.println(response.errorBody());
        }
    }
    // Se falha
    public void onFailure(Call<List<Boi>> call, Throwable t) {
        t.printStackTrace();
    }
});
}
```



## Retrofit – POST

- Definição da interface

```
public interface BoiService {  
    @GET("boi/{id}")  
    Call<Boi> carregarBoi(@Path("id") int id);  
    @GET("boi")  
    Call<List<Boi>> carregarTodos();  
    @POST("boi")  
    Call<Boi> inserir(@Body Boi boi);  
}
```



## Retrofit – POST

- Invocação

```
...  
Boi b = new Boi();  
// setters  
Call<Boi> call = boiService.inserir(b);  
...  
// Métodos de Callback são implementados da mesma forma
```



## Retrofit – PUT

- Definição da interface

```
public interface BoiService {
    @GET("boi/{id}")
    Call<Boi> carregarBoi(@Path("id") int id);
    @GET("boi")
    Call<List<Boi>> carregarTodos();
    @POST("boi")
    Call<Boi> inserir(@Body Boi boi);
    @PUT("boi/{id}")
    Call<Boi> alterar(@Path int id, @Body Boi boi);
}
```



## Retrofit – POST

- Invocação

```
...
Boi b = new Boi();
// setters
Call<Boi> call = boiService.alterar(10, b);
...
// Métodos de Callback são implementados da mesma forma
```



## Retrofit – DELETE

- Definição da interface

```
public interface BoiService {  
    @GET("boi/{id}")  
    Call<Boi> carregarBoi(@Path("id") int id);  
    @GET("boi")  
    Call<List<Boi>> carregarTodos();  
    @POST("boi")  
    Call<Boi> inserir(@Body Boi boi);  
    @PUT("boi/{id}")  
    Call<Boi> alterar(@Path int id, @Body Boi boi);  
    @DELETE("boi/{id}")  
    Call<Boi> remover(@Path int id);  
}
```



## Retrofit – DELETE

- Invocação

```
...  
Call<Boi> call = boiService.remover(10);  
...  
// Métodos de Callback são implementados da mesma forma
```



## Callback para Retorno

- Chamada do WS ocorre em outra Thread
- É assíncrono
- Não tem como "esperar" o término
- Deve-se fazer CALLBACK
- Várias formas de se implementar, ex:
  - Usar um Façade para esconder a lógica
  - Usar uma Interface para o Callback



## Callback para Retorno

- Interface para o Callback

```
public interface BoiCallback {  
    public void onSuccess(Boi boi);  
    public void onFailure(Throwable t);  
}
```



## Callback para Retorno

- Façade para esconder a lógica

```
public class BoiFacade {
    public static void inserir(Boi boi, final BoiCallback callback)
        Retrofit retrofit = new Retrofit.Builder()
            .baseUrl("http://servidor:8080/projeto/webresources/")
            .addConverterFactory(GsonConverterFactory.create()).build();
        BoiService service = retrofit.create(BoiService.class);
        Call<Boi> call = service.inserir(boi);
        call.enqueue(new Callback<Boi>() {
            public void onResponse(Call<Boi> call, Response<Boi> response) {
                if (response.isSuccessful()) {
                    callback.onSuccess(response.body());
                } else {
                    callback onFailure(new Exception(response.errorBody().toString()));
                }
            }
            public void onFailure(Call<Boi> call, Throwable t) {
                callback.onFailure(t);
            }
        });
    }
}
```



## Callback para Retorno

- Na Activity, ao inserir, chama-se

```
ProgressBar progress = (ProgressBar) findViewById(R.id.progress);
progress.setVisibility(View.VISIBLE);
progress.bringToFront();

BoiFacade.inserir(boi, new BoiCallback() {
    @Override
    public void onSuccess(Boi boi) {
        // Trata SUCESSO
    }
    @Override
    public void onFailure(Throwable t) {
        // Trata FALHA
    }
});
```



## Callback para Retorno

```
BoiFacade.inserir(boi, new BoiCallback() {
    @Override
    public void onSuccess(Boi boi) {
        ProgressBar progress = (ProgressBar) findViewById(R.id.progress);
        progress.setVisibility(ProgressBar.INVISIBLE);
        Toast.makeText(InsertActivity.this, "Boi inserido com sucesso.",
                      Toast.LENGTH_LONG).show();
        finish();
    }
    @Override
    public void onFailure(Throwable t) {
        ProgressBar progress = (ProgressBar) findViewById(R.id.progress);
        progress.setVisibility(ProgressBar.INVISIBLE);
        Toast.makeText(InsertActivity.this, "Erro inserindo boi: " +
                      t.getMessage(), Toast.LENGTH_LONG).show();
    }
});
```

