ZAAWANSOWANE TECHNOLOGIE MOBILNE

RETROFIT

type-safe HTTP client for Android and Java

Dawid Talaga & Michał Kolbusz

KILKA SŁÓW O RETROFIT

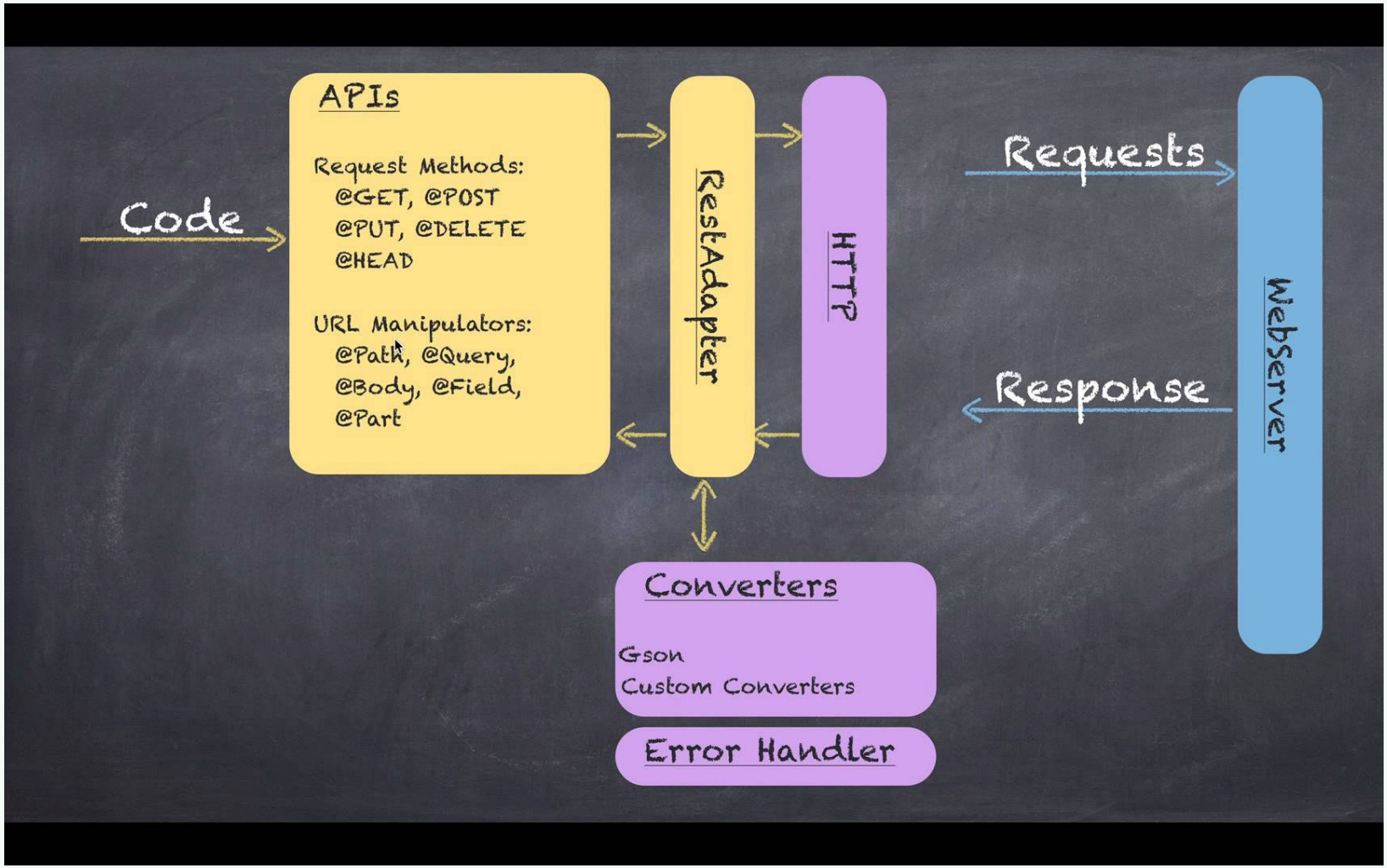
> A type-safe HTTP client
> for Android and Java

Adnotacje do opisu zapytań HTTP

Łatwy sposób na obsługę REST-owych serwisów

Korzysta z konwerterów do
serializacji/deserializacji obiektów

JAK DZIAŁA RETROFIT?



ADNOTACJE METOD

@GET - zapytanie o zasób. Wiąże się tylko z otrzymaniem danych.

@HEAD - zapytanie identyczne jak @GET, ale nie może zwracać żadnej zawartości zasobu poza nagłówkami.

@POST - wysłanie zawartości zasobu na serwer. Wiąże się często z modyfiakcją na serwerze.

@PUT - wysłanie zawartości zasobu na serwer w celu aktulizacji lub zapisu nowej encji

@DELETE - usunięcie zasobu

```
@GET("users")
Call<List<User>> getUsers();


@GET("users/list")
Call<List<User>> getUsersList();
```

ADNOTACJE PARAMETRÓW

@Path - oznaczenie parametru metody jako fragmentu URL

```
@GET("users/{name}/friends")
Call<List<User>> getUserFriends(@Path("name") String username);
```

@Query - oznaczenie parametru metody jako parametr URL (tzw. query parameters)

/users/{name}/friends?sort=[desc|asc]

```
@GET("users/{name}/friends")
Call<List<User>> getUserFriends(@Path("name") String username, @Query("sort") String sort);
```

@QueryMap -w przypadku bardziej skomplikowanych zapytań

```
@GET("users/{name}/friends")
Call<List<User>> getUserFriends(@Path("name") String username, @QueryMap Map<String, String> options);
```

ADNOTACJE PARAMETRÓW

@Body - oznaczenie zawartości zapytania (w połączeniu z @POST/@PUT)

```
@POST("user/{id}")
Call<User> postUser(@Path("id") int id, @Body User user);
```

@FormUrlEncoded - dane w URL w postaci *par1=val1&par2=val2*
@Field - oznaczenie pary klucz-wartość zakodowanych w URL

```
@FormUrlEncoded
@POST("user/edit")
Call<User> postUser(@Field("firstname") String first, @Field("lastname") String last);
```

@FieldMap -w przypadku bardziej skomplikowanych zapytań

```
@FormUrlEncoded
@POST("user/edit")
Call<User> postUser(@FieldMap Map<String, String> fields);
```

ADNOTACJE PARAMETRÓW

@Multipart - określenie zapytania składającego się z wielu części (zazwyczaj gdy wysłamy plik i tekst)

@Part - określenie części zapytania

```
@Multipart
@POST("user/photo")
Call<User> postPhoto(@Part("photo") RequestBody photo, @Part("description") RequestBody description);
```

ADNOTACJE NAGŁÓWKÓW

@Headers - określenie nagłówków zapytania HTTP

```
@Headers({
    "Accept: application/json",
    "Cache-Control: max-age=640000"
})
@GET("users/{id}")
Call<User> getUser(@Path("id") String email);
```

@Header - dynamiczne określenie wartości nagłówka za pomocą parametru metody

```
@GET("users/{id}")
Call<User> getUser(@Path("id") String email, @Header("Authorization") String authoorization);
```

KONWERTERY

Służą do deserializacji danych HTTP do obiektów

Domyślnie używana jest deserializacja zawartości HTTP do RequestBody biblioteki OkHTTP

Możliwość tworzenia własnych konwerterów rozszrzając klasę **Converter.Factory**

KONWERTERY

Gson: com.squareup.retrofit2:converter-gson

Jackson: com.squareup.retrofit2:converter-jackson

Moshi: com.squareup.retrofit2:converter-moshi

Protobuf: com.squareup.retrofit2:converter-protobuf

Wire: com.squareup.retrofit2:converter-wire

Simple XML: com.squareup.retrofit2:converter-simplexml

Scalars (primitives, boxed, and String): com.squareup.retrofit2:converter-scalars

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

GitHubService service = retrofit.create(GitHubService.class);
```

UŻYCIE SERWISU

```java
public class ApiBuilder {
    static String API_ENDPOINT = "http://10.0.2.2:2000/";
    public static UserService userService = null;

    public static UserService getUserService() {
        if(userService == null) {
            Retrofit retrofit = new Retrofit.Builder().baseUrl(API_ENDPOINT)
                    .addConverterFactory(GsonConverterFactory.create())
                    .build();
            userService = retrofit.create(UserService.class);
        }
        return userService;
    }
}
```

```java
ApiBuilder.getUserService().getStudentsList().enqueue(new Callback<List<Student>>() {

    @Override
    public void onResponse(Call<List<Student>> call, Response<List<Student>> response) {
        students = response.body();
        adapter = new ListViewAdapter(students);
        recyclerView.setAdapter(adapter);
    }

    @Override
    public void onFailure(Call<List<Student>> call, Throwable t) {
        Log.d("Error",t.getMessage());
    }
});
```

PRZYKŁAD AUTENTYKACJI

Dla wskazanej metody interfejsu poprzez adnotacje **@Header("Authentication")**

```
public interface UserService {
    @GET("user")
    Call<User> getUser(@Header("Authentication") String authHeader);
}

String base = username + ":" + password;
String authHeader = "Basic" + Base64.encodeToString(base.getBytes(), Base64.NO_WRAP);

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

UserService userClient = retrofit.create(UserService.class);
Call<User> call = userClient.getUser(authHeader);
```

PRZYKŁAD AUTENTYKACJI

Dla każdej metody interfejsu za pomocą mechanizmu Interceptor biblioteki OkHttp

```
// Define the interceptor, add authentication headers
Interceptor interceptor = new Interceptor() {
    @Override
    public okhttp3.Response intercept(Chain chain) throws IOException {
        Request newRequest = chain.request().newBuilder()
        .addHeader("User-Agent", "Retrofit-Sample-App").build();
            return chain.proceed(newRequest);
        }
    };

// Add the interceptor to OkHttpClient
OkHttpClient.Builder builder = new OkHttpClient.Builder();
builder.interceptors().add(interceptor);
OkHttpClient client = builder.build();

// Set the custom client when building adapter
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl(API_ENDPOINT)
    .addConverterFactory(GsonConverterFactory.create())
    .client(client)
    .build();
```

DZIĘKUJEMY ZA UWAGĘ

- TALAGA DAWID & KOLBUSZ MICHAŁ