

Retrofit 2

What the IOException?!

Retrofit 2

What the IOException?!

What the HttpException?!

Ash Davies

@DogmaticCoder

**"Retrofit 2 will be out by
the end of this year"**

— Jake Wharton (*Droidcon NYC 2014*)

**"Retrofit 2 will be out by
the end of this year"**

— Jake Wharton (*Droidcon NYC 2015*)

"Retrofit 2.0.0 is now released!"

– ***@JakeWharton (11 Mar 2016)***

A woman with long, wavy brown hair is smiling warmly at the camera. She is wearing a dark green turtleneck sweater over a light-colored collared shirt. Her hands are clasped together in front of her, holding a large, lush bouquet of red and pink roses. The background is a soft-focus outdoor scene with greenery and a building.

Finally!



IMMOBILIEN
SCOUT24

**BRILLIANT IDEA
MAYOR.**



Retrofit2: Dependencies

```
dependencies {  
  
    // Retrofit 1.9  
    compile 'com.squareup.retrofit:retrofit:1.9.0'  
  
    // Retrofit 2.1  
    compile 'com.squareup.retrofit2:retrofit:2.1.0'  
  
}
```

Method Counts

Retrofit2: Dependencies

Retrofit 1.9 (1997)

Retrofit (776)

Gson (1231)

Retrofit 2.1 (3349)

retrofit2 (508)

OkHttp3 (2180)

OkIo (661)

Retrofit2: Dependencies

Retrofit 1.9 (4848)

Retrofit (776)

OkHttp3 (2180)

Gson (1231)

OkIo (661)

Retrofit2: Dependencies

Retrofit 2.1 (3496)

Retrofit2 (508)

Converter-Gson (32)

Adapter-RxJava (115)

OkHttp3 (2180)

OkIo (661)

Retrofit2: Dependencies

Retrofit 1.9 & 2.1 (+655)

Retrofit 1.9 > 2.1 (-1352)

OkHttp

Awesome!

What's Next?

Retrofit2: RestAdapter

```
// Retrofit 1.9
RestAdapter adapter = new RestAdapter.Builder()
    .setClient(new Ok3Client(okHttpClient))
    .setEndpoint("...")
    .build();
```

```
// Retrofit 2.1
Retrofit retrofit = new Retrofit.Builder()
    .client(okHttpClient)
    .baseUrl("...")
    .build();
```

Converters

Retrofit2: Converters

```
compile 'com.squareup.retrofit2:converter-gson:2.1.0'
```

Retrofit2: Converters

```
compile 'com.squareup.retrofit2:converter-gson:2.1.0'
```

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

Retrofit2: Converters

```
compile 'com.squareup.retrofit2:converter-gson:2.1.0'
```

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

moshi, scalars, simplexml, wire, jackson, protobuf

Multiple Converters

Retrofit2: Multiple Converters

→ Checks every converter sequentially

Retrofit2: Multiple Converters

- Checks every converter sequentially
- Register converter factories in order

Retrofit2: Multiple Converters

- Checks every converter sequentially
- Register converter factories in order
- Create custom retrofit2.Converter.Factory

**"I love
Factories!"**

— @alosdev



TRUE STORY

A photograph showing a large pile of cut logs and branches in a dense forest. The logs are piled high, filling the lower half of the frame. The background is filled with tall trees and green foliage. The overall scene conveys a sense of deforestation and industrial logging.

Logging

Retrofit2: Logging

```
dependencies {  
    compile 'com.squareup.okhttp3:logging-interceptor:3.4.1'  
}
```

Retrofit2: Logging

```
HttpLoggingInterceptor logger = new HttpLoggingInterceptor();  
logger.setLevel(HttpLoggingInterceptor.Level.BODY);
```

```
OkHttpClient client = new OkHttpClient.Builder()  
    .addInterceptor(logger)  
    .build();
```

Adapters

Adapters

Call -> Asynchronous Consumer

Retrofit2: Adapters

```
// RxJava: Observable<T>
com.squareup.retrofit2:adapter-rxjava
```

Retrofit2: Adapters

// RxJava: Observable<T>

com.squareup.retrofit2:adapter-rxjava

// RxJava2: Flowable<T>

com.jakewharton.retrofit:retrofit2-rxjava2-adapter

Retrofit2: Adapters

```
// RxJava: Observable<T>
com.squareup.retrofit2:adapter-rxjava

// RxJava2: Flowable<T>
com.jakewharton.retrofit:retrofit2-rxjava2-adapter

// Java8: CompletableFuture<T>
com.squareup.retrofit:converter-java8
```

Retrofit2: Adapters

// RxJava: Observable<T>

com.squareup.retrofit2:adapter-rxjava

// RxJava2: Flowable<T>

com.jakewharton.retrofit:retrofit2-rxjava2-adapter

// Java8: CompletableFuture<T>

com.squareup.retrofit:converter-java8

// Guava: ListenableFuture<T>

com.squareup.retrofit:converter-guava



Services

Retrofit2: Services

```
/* Retrofit 1.9 */
public interface Service {
    // Synchronous
    @GET("/articles")
    List<Article> articles();

    // Asynchronous
    @GET("/articles")
    void articles(Callback<List<Article>> callback);
}
```

Retrofit2: Services

```
/* Retrofit 2.1 */
public interface Service {
    @GET("articles")
    Call<List<Article>> articles();
}
```

Retrofit2: Services

```
/* Retrofit 1.9 */  
@GET("/articles")
```

```
/* Retrofit 2.1 */  
@GET("articles")
```

Retrofit2: Services

```
public interface AwsService {  
  
    @GET  
    public Call<File> getImage(@Url String url);  
}
```

Retrofit2: Services

```
public interface AwsService {  
  
    @GET  
    public Call<File> getImage(@Url String url);  
}
```

Useful when working with external Services

Retrofit2: Services

```
/* Retrofit 2.1 */
Call<List<Article>> call = service.articles();

// Synchronous
call.execute();

// Asynchronous
call.enqueue();
```

Retrofit2: Cancel Requests

```
/* Retrofit 2.1 */  
  
Call<ResponseBody> call = service.get("...");  
  
call.enqueue(new Callback<ResponseBody>() { ... });  
  
call.cancel();
```



Citytv



Rx all the things!

A scene from The Simpsons where the entire family is relaxing in a hot tub. Homer is on the right, Marge is on the left, Bart is in the center, and the two youngest children are floating in the water. The hot tub is situated on a wooden deck next to a green lawn and a wooden fence. A window with a blue frame is visible in the background.

Life is Good

(ALL MOANING HAPPILY)



**What happens when
things go wrong?**

Retrofit2: Errors

Retrofit2: Errors

```
{  
    statusCode: 400,  
    message: "Malformed email"  
}
```

```
/* Retrofit 1.9 */
service.login(username, password)
.subscribe(user -> {
    session.storeUser(user);
    view.gotoProfile();
}, throwable -> {
    if (throwable instanceof RetrofitError) {
        processServerError(
            ((RetrofitError) throwable).getBodyAs(ServerError.class)
        );
    }
    view.onError(throwable.getMessage());
});
```

FAILURE: Build failed with an exception.
`error: cannot find symbol class RetrofitError`



RetrofitError is dead.

Long live HttpException.

RetrofitError

Retrofit: RetrofitError

```
/* Retrofit 1.9: RetrofitError */
public Object getBodyAs(Type type) {
    if (response == null) {
        return null;
    }
    TypedInput body = response.getBody();
    if (body == null) {
        return null;
    }
    try {
        return converter.fromBody(body, type);
    } catch (ConversionException e) {
        throw new RuntimeException(e);
    }
}
```

***"How do you know if it
was a conversion error,
network error, random
error inside Retrofit?
Yuck."***

– Jake Wharton (Jul 25, 2013)

Retrofit: RetrofitError

```
/** Identifies the event kind which triggered a {@link RetrofitError}. */
public enum Kind {
    /** An {@link IOException} occurred while communicating to the server. */
    NETWORK,
    /** An exception was thrown while (de)serializing a body. */
    CONVERSION,
    /** A non-200 HTTP status code was received from the server. */
    HTTP,
    /**
     * An internal error occurred while attempting to execute a request. It is best practice to
     * re-throw this exception so your application crashes.
     */
    UNEXPECTED
}
```

Introduced in Retrofit v1.7 (Oct 8, 2014)

*"I find that API to awful
(which I'm allowed to say
as the author)"*

— Jake Wharton (Nov 7, 2015)

What Now?

```
/* Retrofit 2.1 */
service.login(username, password)
.subscribe(user -> {
    session.storeUser(user);
    view.gotoProfile();
}, throwable -> {
    // Now What?!
});
```

Exceptions

Retrofit2: Exceptions

IOException: Network errors

Retrofit2: Exceptions

HttpException: Non 2xx responses

HttpException

Retrofit2: HttpException

→ Included in Retrofit2 adapters

Retrofit2: HttpException

- Included in Retrofit2 adapters
- Contains a response object

Retrofit2: HttpException

- Included in Retrofit2 adapters
- Contains a response object
- Response is not serialisable

IOException

Retrofit2: IOException

RetrofitError.Kind.NETWORK

RetrofitError.Kind.CONVERSION



Retrofit: IOExceptions

`isConnectedOrConnecting();`



```
throwable -> {
    if (throwable instanceof HttpException) {
        // non-2xx error
    }

    else if (throwable instanceof IOException) {
        // Network or conversion error
    }

    else {
        // ^\_(ツ)_/-
    }
}
```

Cool story bro...

```
public abstract class ServerError extends RuntimeException {  
  
    private final int statusCode;  
  
    public ServerError(int statusCode, String message) {  
        super(message);  
        this.statusCode = statusCode;  
    }  
  
    public int getStatusCode() {  
        return statusCode;  
    }  
}
```

```
public class ErrorProcessor {  
  
    private final Converter<ResponseBody, ServerError> converter;  
  
    public ServerErrorProcessor(Retrofit retrofit) {  
        this(retrofit.responseBodyConverter(ServerError.class, new Annotation[0]));  
    }  
  
    private ServerErrorProcessor(Converter<ResponseBody, ServerError> converter) {  
        this.converter = converter;  
    }  
  
    public <T> Function<Throwable, Observable<? extends T>> convert() {  
        return throwable -> {  
            if (throwable instanceof HttpException) {  
                Response response = ((HttpException) throwable).response();  
                return Observable.error(converter.convert(response.errorBody()));  
            }  
  
            return Observable.error(throwable);  
        };  
    }  
}
```

```
public static class UserClient {

    private final ErrorProcessor processor;
    private final UserService service;

    public UserClient(Retrofit retrofit) {
        this(retrofit.create(UserService.class), new ErrorProcessor(retrofit));
    }

    private UserClient(UserService service, ErrorProcessor processor) {
        this.service = service;
        this.processor = processor;
    }

    public Flowable<User> login(String username, String password) {
        return service.login(username, password)
            .onErrorResumeNext(processor.convert());
    }
}
```

RetrofitException

goo.gl/N5tRoH

```
public static RetrofitException from(Throwable throwable) {
    if (throwable instanceof HttpException) {
        Response response = (HttpException) throwable).response();
        Request request = response.raw().request();

        return RetrofitException.http(request.url().toString(), response, retrofit);
    }

    if (throwable instanceof IOException) {
        return RetrofitException.network((IOException) throwable);
    }

    return RetrofitException.unexpected(throwable);
}
```

```
public class RxErrorHandlerAdapterFactory extends CallAdapter.Factory {
    private final RxJavaCallAdapterFactory original = RxJavaCallAdapterFactory.create();

    @Override
    public CallAdapter<?> get(Type returnType, Annotation[] annotations, Retrofit retrofit) {
        return new RxCallAdapterWrapper(retrofit, original.get(returnType, annotations, retrofit));
    }

    private static class RxCallAdapterWrapper implements CallAdapter<Observable<?>> {
        private final Retrofit retrofit;
        private final CallAdapter<?> wrapped;

        public RxCallAdapterWrapper(Retrofit retrofit, CallAdapter<?> wrapped) {
            this.retrofit = retrofit;
            this.wrapped = wrapped;
        }

        @Override
        public Type responseType() {
            return wrapped.responseType();
        }

        @Override
        public <R> Observable<?> adapt(Call<R> call) {
            return ((Observable) wrapped.adapt(call)).onErrorResumeNext(throwable -> {
                return Observable.error(RetrofitException.from(throwable));
            });
        }
    }
}
```

RxJava Adapters

RxJava Adapters

Observable<T> call();

RxJava Adapters

Observable<Response<T>> call();

RxJava Adapters

Observable<Result<T>> call();

Thank You