Retrofit 2.0 how to get deserialised error response.body

Asked 9 years, 3 months ago Modified 1 year, 5 months ago Viewed 242k times

213

I'm using **Retrofit 2.0.0-beta1**.

In tests i have an alternate scenario and expect error HTTP 400

I would like to have `retrofit.Response<MyError> response` but `response.body() == null`

MyError is not deserialised - i see it only here

```
response.errorBody().string()
```

but it doesn't give me MyError as object

java retrofit retrofit2

Share Follow

edited Mar 6, 2016 at 16:41
chubao
6,011 ● 6 ● 43 ● 64

asked Sep 11, 2015 at 9:03
Piotr Boho
2,748 ● 2 ● 14 ● 21

10 its simple futurestud.io/tutorials/retrofit-2-simple-error-handling – ahmadalibaloch Oct 3, 2016 at 14:17

is it a good practice to deserialise the error response? since the response might be a webserver error which is html. – Hossein Shahdoost Mar 5, 2018 at 9:28

1 thx @ahmadalibaloch, that link is really really helpfull. – Ravi Vaniya Oct 26, 2018 at 10:45

31 Answers

Sorted by: Highest score (default)

1 2 Next

221

I currently use a very easy implementation, which does not require to use converters or special classes. The code I use is the following:

```
public void onResponse(Call<ResponseBody> call, Response<ResponseBody>
response) {
    DialogHelper.dismiss();

    if (response.isSuccessful()) {
        // Do your success stuff...
    } else {
        try {
            JSONObject jObjError = new
JSONObject(response.errorBody().string());
            Toast.makeText(getContext(),
jObjError.getJSONObject("error").getString("message"),
Toast.LENGTH_LONG).show();
        } catch (Exception e) {
            Toast.makeText(getContext(), e.getMessage(),
Toast.LENGTH_LONG).show();
        }
    }
}
```

A point to note here is that `response.errorBody().string()` will return the correct value only once. If you call it again, it will return an empty string. So in case you want to reuse it, store the value in a variable with the first call.

There is a way to get the error body string from the response without making it empty on the next call, by rolling your own implementation of `toString()` that does not update the `errorBody` Buffer's read-pointer. See this answer for more info.

Share Follow

edited Sep 26, 2022 at 20:06
ErikE
50.1k ● 23 ● 154 ● 200

answered Jul 7, 2016 at 10:56
Saif Bechan
17.1k ● 23 ● 85 ● 125

7 Your solution doesn't show an error response content. – CoolMind Sep 18, 2016 at 17:21

2 Check my edit, dunno why I made it so unclear in the first place. – Saif Bechan Sep 19, 2016 at 7:57

8 Finally, a simple answer to an android question that works (most android answers are comically complex). – Doug Voss Jul 7, 2017 at 21:08

This is definitely not answering the question. It simply return the error message and not the Error Enum Object. Please follow this response : stackoverflow.com/a/21103420/2914140 – Tobliug Aug 18, 2017 at 13:01

In the answer it's looking for a mapping to "message", but my error response didn't have that. It had a mapping to "error". So everyone reading, it depends on the response you get! – mco Aug 15, 2018 at 20:57

104

ErrorResponse is your custom response object

Kotlin

```
val gson = Gson()
val type = object : TypeToken<ErrorResponse>() {}.type
var errorResponse: ErrorResponse? =
gson.fromJson(response.errorBody()!!.charStream(), type)
```

Java

```
Gson gson = new Gson();
Type type = new TypeToken<ErrorResponse>() {}.getType();
ErrorResponse errorResponse =
gson.fromJson(response.errorBody.charStream(),type);
```

Share Follow

edited Oct 29, 2018 at 16:30
Arshak
3,225 ● 1 ● 26 ● 33

answered Feb 13, 2018 at 11:35
Shahab Rauf
3,911 ● 1 ● 31 ● 42

6 you shouldn't be force unwrapping optionals: `gson.fromJson(response.errorBody()?.charStream(), type)` – hopeman Mar 26, 2019 at 10:11

46

I solved it by:

```
 if(!response.isSuccessful()){
       Gson gson = new Gson();
       MyErrorMessage
message=gson.fromJson(response.errorBody().charStream(),MyErrorMessage.class);
       if(message.getCode()==ErrorCode.DUPLICATE_EMAIL_ID_CODE){
                  //DO Error Code specific handling
        }else{
                 //DO GENERAL Error Code Specific handling
        }
    }
```

MyErrorMessage Class:

```
  public class MyErrorMessage {
     private int code;
     private String message;

     public int getCode() {
        return code;
     }

     public void setCode(int code) {
        this.code = code;
     }

     public String getMessage() {
         return message;
     }
```

```
    public void setMessage(String message) {
        this.message = message;
    }
}
```

Share Follow

edited Sep 8, 2017 at 5:19
Amarjit
4,357 ● 2 ● 37 ● 53

answered Feb 15, 2017 at 20:30
Pooja Gupta
461 ● 4 ● 2

2 java.lang.IllegalStateException: Expected BEGIN_OBJECT but was STRING at line 1 column 2 path $ – Ronel Gonzales Sep 19, 2017 at 9:03

Use `.addConverterFactory(ScalarsConverterFactory.create())` @RonelGonzales – Pratik Butani Nov 22, 2018 at 8:45

43

It's actually very straight forward.

Kotlin:

```
val jsonObj = JSONObject(response.errorBody()!!.charStream().readText())
responseInterface.onFailure(jsonObj.getString("msg"))
```

Java:

```
    if(response.errorBody()!=null){
    JSONObject jsonObj = new
JSONObject(TextStreamsKt.readText(response.errorBody().charStream()));
        responseInterface.onFailure(jsonObj.getString("msg"));
    }else{
        responseInterface.onFailure("you might want to return a generic error
message.");
    }
```

Tested on retrofit:2.5.0. Read the text from the charStream which will give you a String, then parse to JSONObject.

Adios.

Share Follow

edited Oct 13, 2020 at 8:20

answered May 15, 2020 at 0:32
Wale
1,796 ● 16 ● 31

2 there's no `readText()` extension on java, use `TextStreamsKt.readText(response.errorBody().charStream())` if you still on java – mochadwi Jun 5, 2020 at 4:52

Update: Retrofit 2.6.2 -> `val errorMessage=jsonObj.getJSONArray("errors").getJSONObject(0).getString("message")` – Darshan Miskin Aug 15, 2020 at 15:01

@Wale, Why this doesn't work for body without an error? **JSONObject(response.body()!!.charStream().readText())** – J A S K I E R Jan 26, 2021 at 9:04

1 @Oleksandr I'm not sure I get your question but I don't think your response.body has a byteStream function, it could come as a String if you using a scala converter or related or it could simple come as an object you passed if you're using google gson converter. So, what you're trying to do might be this=> JSONObject (response.errorBody()!!.charStream().readText()) – Wale Jan 26, 2021 at 14:58

@TaslimOseni I'm glad it worked for you too. – Wale Feb 16, 2021 at 16:52

33

In Retrofit 2.0 beta2 this is the way that I'm getting error responses:

1. Synchronous

```
try {
   Call<RegistrationResponse> call =
backendServiceApi.register(data.in.account, data.in.password,
           data.in.email);
   Response<RegistrationResponse> response = call.execute();
   if (response != null && !response.isSuccess() && response.errorBody() !=
null) {
       Converter<ResponseBody, BasicResponse> errorConverter =

MyApplication.getRestClient().getRetrofitInstance().responseConverter(BasicR
new Annotation[0]);
       BasicResponse error = errorConverter.convert(response.errorBody());
       //DO ERROR HANDLING HERE
       return;
   }
   RegistrationResponse registrationResponse = response.body();
   //DO SUCCESS HANDLING HERE
} catch (IOException e) {
   //DO NETWORK ERROR HANDLING HERE
}
```

2. Asynchronous

```
Call<BasicResponse> call = service.loadRepo();
call.enqueue(new Callback<BasicResponse>() {
    @Override
    public void onResponse(Response<BasicResponse> response, Retrofit retrof
{
        if (response != null && !response.isSuccess() && response.errorBody(
!= null) {
            Converter<ResponseBody, BasicResponse> errorConverter =
                retrofit.responseConverter(BasicResponse.class, new
Annotation[0]);
            BasicResponse error = errorConverter.convert(response.errorBody(
            //DO ERROR HANDLING HERE
            return;
        }
        RegistrationResponse registrationResponse = response.body();
        //DO SUCCESS HANDLING HERE
    }

    @Override
    public void onFailure(Throwable t) {
        //DO NETWORK ERROR HANDLING HERE
    }
});
```

Update for Retrofit 2 beta3:

1. Synchronous - not changed
2. Asynchronous - Retrofit parameter was removed from onResponse

```
Call<BasicResponse> call = service.loadRepo();
call.enqueue(new Callback<BasicResponse>() {
    @Override
    public void onResponse(Response<BasicResponse> response) {
        if (response != null && !response.isSuccess() && response.errorBody(
!= null) {
            Converter<ResponseBody, BasicResponse> errorConverter =

MyApplication.getRestClient().getRetrofitInstance().responseConverter(BasicR
new Annotation[0]);
            BasicResponse error = errorConverter.convert(response.errorBody(
            //DO ERROR HANDLING HERE
            return;
        }
        RegistrationResponse registrationResponse = response.body();
        //DO SUCCESS HANDLING HERE
    }

    @Override
    public void onFailure(Throwable t) {
        //DO NETWORK ERROR HANDLING HERE
    }
});
```

Share Follow

edited Jan 21, 2016 at 9:03

answered Oct 1, 2015 at 20:13

JFreeman
685 ● 6 ● 17

4 what do you have in BasicResponse ? – Jemshit Nov 25, 2015 at 8:18

2 Just a basic Jackson annotated class that contains message and error code. In any case you can have any annotated class there that matches your server response type. Try using jsonschema2pojo to generate one matching your needs. – JFreeman Nov 25, 2015 at 10:08

For others, you may use this instead: Converter<ResponseBody, <Message> errorConverter = retrofit.responseBodyConverter(Message.class, new Annotation[0]); – Kim Montano Mar 1, 2016 at 3:44

@JFreeman, what if I want to deserialize `List<BasicResponse>` ? – azizbekian Mar 31, 2016 at 12:37

can you for me see class MyApplication – dungtv Apr 15, 2016 at 3:42

29

Create a model of the Error response & user Gson to convert the response to it. This will just work fine.

APIError.java

```
public class APIError {
    private String message;

    public String getMessage() {
        return message;
    }
}
```

MainActivity.java (inside request onResponse)

```
if (response.isSuccessful()) {
    // Do your success stuff...

} else {
    APIError message = new Gson().fromJson(response.errorBody().charStream(),
APIError.class);
    Toast.makeText(MainActivity.this, "" + message.getMessage(),
Toast.LENGTH_SHORT).show();
}
```

Share Follow

answered Feb 19, 2020 at 6:08

Sreekant Shenoy
1,618 ● 16 ● 25

This by far is the most elegant way of doing it.. thanks for the answer... – DragonFire Jan 8, 2021 at 22:37

Thank you so much, I was using errorBody().toString instead of errorBody().charStream(). Thanks again – Pemba Tamang Apr 29, 2023 at 5:50

12

If you use Kotlin another solution could be just create extension function for Response class:

```
inline fun <reified T>Response<*>.parseErrJsonResponse(): T?
{
    val moshi = MyCustomMoshiBuilder().build()
    val parser = moshi.adapter(T::class.java)
    val response = errorBody()?.string()
    if(response != null)
        try {
            return parser.fromJson(response)
        } catch(e: JsonDataException) {
            e.printStackTrace()
        }
    return null
}
```

Usage

```
val myError = response.parseErrJsonResponse<MyErrorResponse>()
if(myError != null) {
   // handle your error logic here
   // ...
}
```

Share Follow   edited Apr 5, 2019 at 2:30   answered Mar 20, 2019 at 6:57
Arsenius
5,612 ● 6 ● 30 ● 45

1 Finally someone used Kotlin power to make code easy to read ! – Vince Apr 5, 2019 at 1:14

11

```
 @Override
 public void onResponse(Call<Void> call, retrofit2.Response<Void> response) {
            if (response.isSuccessful()) {

            //Do something if response is ok
            } else {

                JsonParser parser = new JsonParser();
                JsonElement mJson = null;
                try {
                    mJson = parser.parse(response.errorBody().string());
                    Gson gson = new Gson();
                    MyError errorResponse = gson.fromJson(mJson,
MyError.class);
                } catch (IOException ex) {
                    ex.printStackTrace();
                }
            }
```

Share Follow

edited Jun 8, 2017 at 7:23

answered May 26, 2017 at 15:35

Vins
385 ● 4 ● 9

10

In https://stackoverflow.com/a/21103420/2914140 and https://futurestud.io/tutorials/retrofit-2-simple-error-handling this variant is shown for Retrofit 2.1.0.

```
call.enqueue(new Callback<MyResponse>() {
    @Override
    public void onResponse(Call<MyResponse> call, Response<MyResponse>
response) {
        if (response.isSuccessful()) {
            ...
        } else {
            Converter<ResponseBody, MyError> converter
                    = MyApplication.getRetrofit().responseBodyConverter(
                    MyError.class, new Annotation[0]);
            MyError errorResponse = null;
            try {
                errorResponse = converter.convert(response.errorBody());
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
```

Share Follow

edited May 23, 2017 at 12:17
Community Bot
1 ● 1

answered Sep 18, 2016 at 17:45
CoolMind
28.7k ● 18 ● 205 ● 236

9

There are many valid answers already. This is just an addition for a use case, when you need to consume same Retrofit response more than once. Neither of below can be used, as you can read response body only once, as it will be closed afterwards and you will get `null` each next time, when you try to read from the same response object:

```
response()?.errorBody()?.charStream()?.readText()
response()?.errorBody()?.string()
```

Instead, you can get read-only copy of response string (while the response itself can be passed over and eventually consumed later):

```
response()?.errorBody()?.source()?.buffer?.snapshot()?.utf8()
```

Share Follow

answered Aug 9, 2021 at 17:11
Myroslav
1,217 ● 16 ● 25

This should be higher! – dicarlomagnus Aug 19, 2022 at 22:13

7

I was facing same issue. I solved it with retrofit. Let me show this...

If your error JSON structure are like

```
{
"error": {
    "status": "The email field is required."
}
}


My ErrorRespnce.java

public class ErrorResponse {

   @SerializedName("error")
   @Expose
   private ErrorStatus error;

   public ErrorStatus getError() {
      return error;
   }

   public void setError(ErrorStatus error) {
      this.error = error;
   }
}
```

And this my Error status class

```
public class ErrorStatus {

  @SerializedName("status")
  @Expose
  private String status;

  public String getStatus() {
      return status;
  }

  public void setStatus(String status) {
      this.status = status;
  }
}
```

Now we need a class which can handle our json.

```
  public class ErrorUtils {

   public static ErrorResponse parseError (Response<?> response){
      Converter<ResponseBody , ErrorResponse> converter =
ApiClient.getClient().responseBodyConverter(ErrorResponse.class , new
Annotation[0]);
    ErrorResponse errorResponse;
    try{
        errorResponse = converter.convert(response.errorBody());
    }catch (IOException e){
        return new ErrorResponse();
    }
    return errorResponse;
}
}
```

Now we can check our response in retrofit api call

```
private void registrationRequest(String name , String email , String password ,
String c_password){


    final Call<RegistrationResponce> registrationResponceCall =
apiInterface.getRegistration(name , email , password , c_password);
    registrationResponceCall.enqueue(new Callback<RegistrationResponce>() {
        @Override
        public void onResponse(Call<RegistrationResponce> call,
Response<RegistrationResponce> response) {



            if (response.code() == 200){


            }else if (response.code() == 401){


                ErrorResponse errorResponse = ErrorUtils.parseError(response);
                Toast.makeText(MainActivity.this,
""+errorResponse.getError().getStatus(), Toast.LENGTH_SHORT).show();
            }
        }

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

        }
    });
}
```

That's it now you can show your Toast

Share Follow

edited Mar 8, 2020 at 13:16
Ali Noureddine
323 ● 5 ● 20

answered Oct 1, 2018 at 7:46
pavel
1,671 ● 23 ● 19

App Crash ErrorResponse$ErrorStatus.getStatus()' on a null object reference
– Faizan Haidar Khan Nov 23, 2020 at 9:59

It goes into catch block of ErrorUtils and says End of input at line 1 column 1 path $
– Faizan Haidar Khan Nov 26, 2020 at 7:55

7

```
if(!response.isSuccessful()) {
    StringBuilder error = new StringBuilder();
    try {
        BufferedReader bufferedReader = null;
        if (response.errorBody() != null) {
            bufferedReader = new BufferedReader(new InputStreamReader(
                    response.errorBody().byteStream()));

            String eLine = null;
            while ((eLine = bufferedReader.readLine()) != null) {
                error.append(eLine);
            }
            bufferedReader.close();
        }

    } catch (Exception e) {
        error.append(e.getMessage());
    }

    Log.e("Error", error.toString());
}
```

Share Follow

edited Sep 19, 2020 at 13:21

answered Aug 19, 2019 at 5:37
Riyas PK
3,217 ● 1 ● 25 ● 29

This works.. with laravel api response of 422 – DragonFire Jan 8, 2021 at 7:43

This is the only one that worked for me for retrofit 2.9.0 – dicarlomagnus Sep 21, 2021 at 16:34

6

I did it this way for asynchronous calls using Retrofit 2.0-beta2:

```
@Override
public void onResponse(Response<RegistrationResponse> response,
                       Retrofit retrofit) {
    if (response.isSuccess()) {
        // Do success handling here
    } else {
        try {
            MyError myError = (MyError)retrofit.responseConverter(
                    MyError.class, MyError.class.getAnnotations())
                .convert(response.errorBody());
            // Do error handling here
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Share Follow

edited Dec 9, 2015 at 12:06
JJD
51.7k ● 61 ● 212 ● 348

answered Oct 21, 2015 at 15:32
Shantanu
692 ● 5 ● 20

What will be MyError class? – Dhrupal Aug 9, 2016 at 10:24

I thought onResponse was supposed to contain a Call parameter and a Response parameter. How is it that yours has a Retrofit parameter? – Marty Miller Jun 1, 2017 at 22:44

@MartyMiller this was done for the following version of retrofit Retrofit 2.0-beta2 – Shantanu Jun 2, 2017 at 16:17

6

Here is elegant solution using `Kotlin` extensions:

```
data class ApiError(val code: Int, val message: String?) {
    companion object {
        val EMPTY_API_ERROR = ApiError(-1, null)
    }
}

fun Throwable.getApiError(): ApiError? {
    if (this is HttpException) {
        try {
            val errorJsonString = this.response()?.errorBody()?.string()
            return Gson().fromJson(errorJsonString, ApiError::class.java)
        } catch (exception: Exception) {
            // Ignore
        }
    }
    return EMPTY_API_ERROR
}
```

and usage:

`showError(retrofitThrowable.getApiError()?.message)`

Share Follow

edited May 4, 2020 at 9:06

answered Apr 28, 2020 at 7:42
Antonis Radz
3,097 ● 1 ● 19 ● 36

Love this. Could also easily be made generic so that you could pass in an error type: `fun <T:Exception> Throwable.getCustomException(classType: Class<T>): T?` then update the GSON line to `Gson().fromJson(errorJsonString, classType)`. Used as `e.getCustomException(CustomException::class.java)?` – Bueno Aug 19, 2020 at 15:48

5

json response

```
{
    "success": false,
    "status_code": 32,
    "status_message": "Email not verified: Your email address has not been
verified."
}
```

Error class

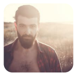
```
data class ResponseError(
    @SerializedName("status_code")
    val statusCode: Int,
    @SerializedName("status_message")
    val statusMessage: String,
    @SerializedName("success")
    val success: Boolean
)
```

get error message

```
fun <T : Any> getResultOrError(response: Response<T>): T? {
    if (response.isSuccessful) {
        return response.body()
    } else {
        try {
            val responseError = Gson().fromJson(
                response.errorBody()?.string(),
                ResponseError::class.java
            )
            throw Throwable(responseError.statusMessage)
        } catch (e: Exception) {
            throw Throwable("Unknown error")
        }
    }
}
```

Share Follow

answered Jan 29, 2022 at 19:07



Ruslan Grigoriev
93 ● 1 ● 6

3

This way you do not need a Retrofit instance if you only are injecting a service created from Retrofit.

```
public class ErrorUtils {

  public static APIError parseError(Context context, Response<?> response) {

    APIError error = new APIError();

    try {
        Gson gson = new Gson();
        error = gson.fromJson(response.errorBody().charStream(),
APIError.class);
    } catch (Exception e) {
        Toast.makeText(context, e.getMessage(), Toast.LENGTH_LONG).show();
    }

    if (TextUtils.isEmpty(error.getErrorMessage())) {
        error.setError(response.raw().message());
    }
    return error;
  }
}
```

Use it like this:

```
if (response.isSuccessful()) {

      ...

    } else {

      String msg = ErrorUtils.parseError(fragment.getActivity(),
response).getError(); // would be from your error class
      Snackbar.make(someview, msg, Snackbar.LENGTH_LONG).show();
    }
  }
```

Share Follow

answered Jul 10, 2017 at 22:45

Codeversed

9,473 ● 3 ● 45 ● 42

3

if your error response is a string you can deserialize it by using the following kotlin code :

```
val errorString = response.errorBody()?.byteStream()?.bufferedReader().use {
it?.readText() }  // defaults to UTF-8
```

Share Follow

answered Dec 19, 2021 at 11:31

a0x2
2,111 ● 1 ● 19 ● 26

you can simply use "response.message().toString()" which will give the same error string in a more readable format. – Arpit Patel May 25, 2022 at 19:51

3

In Kotlin I solved it creating a custom ResponseBody generic extension function function that converts the response body to a JSONObject. then you can use gson to customize the error response body with your custom Error Data Class.

```
inline fun <reified T> ResponseBody.getErrorObject(): T {
    val gson = Gson()
    val jsonObject = JSONObject(charStream().readText())
    return gson.fromJson(jsonObject.toString(), T::class.java)
}
```

You can then customize the error response to your custom class. For this I'm using an example

```
data class LoginError(
    val error: Error,
    val message: String,
    val success: Boolean
)

data class Error(
    val error: String,
    val status: Int
)
```

then use the extension function this way

```
val error = state.errorBody.getErrorObject<LoginError>()
```

the `state.errorBody` is my error response from retrofit of type ResponseBody

Share Follow

edited Jul 6, 2023 at 12:40

answered May 16, 2022 at 13:42

Ezzy Wachira
380 ● 2 ● 3 ● 14

2

This seems to be the problem when you use OkHttp along with Retrofit, so either you can remove OkHttp or use code below to get error body:

```
if (!response.isSuccessful()) {
 InputStream i = response.errorBody().byteStream();
 BufferedReader r = new BufferedReader(new InputStreamReader(i));
 StringBuilder errorResult = new StringBuilder();
 String line;
 try {
   while ((line = r.readLine()) != null) {
   errorResult.append(line).append('\n');
   }
 } catch (IOException e) {
    e.printStackTrace();
}
}
```

Share Follow

answered Feb 13, 2017 at 13:04

KRUPEN GHETIYA
221 ●2 ●6

2

Tested and works

```
 public BaseModel parse(Response<BaseModel> response , Retrofit retrofit){
            BaseModel error = null;
            Converter<ResponseBody, BaseModel> errorConverter =
                    retrofit.responseBodyConverter(BaseModel.class, new
Annotation[0]);
            try {
                if (response.errorBody() != null) {
                    error = errorConverter.convert(response.errorBody());
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
            return error;
        }
```

Share Follow

answered Sep 29, 2019 at 17:34

The MJ
473 ●7 ●18

2

For people using Kotlin with Moshi and coroutines, this is what I did:

Error data class

```
@JsonClass(generateAdapter = true)
data class ApiResponseNoData(
    val exito: Int,
    val error: String?
)
```

Extension

```
fun ResponseBody.getApiError(): String? {
    return try {
        Moshi
            .Builder()
            .build()
            .adapter(ApiResponseNoData::class.java)
            .fromJson(string())
            ?.error
    }catch(e: Exception) { null }
}
```

ViewModel

```
fun test() {
    viewModelScope.launch(Dispatchers.IO) {
        val response = repository.test()
        withContext(Dispatchers.Main) {
            if(response.isSuccessful) {
                ...
            }else{
                val errorMsg = response.errorBody()?.getApiError() ?:
"Unexpected error occurred"
                ...
            ]
        }
    }
}
```

Share Follow

answered Jun 11, 2022 at 23:15
Ricardo Yubal
491 ● 6 ● 8

0

solved it by:

```
Converter<MyError> converter =
    (Converter<MyError>)JacksonConverterFactory.create().get(MyError.class);
MyError myError =  converter.fromBody(response.errorBody());
```

Share Follow

edited Oct 5, 2015 at 19:10
JJD
51.7k ● 61 ● 212 ● 348

answered Sep 11, 2015 at 10:06
Piotr Boho
2,748 ● 2 ● 14 ● 21

How can I convert through GsonConverterFactory ? Any idea ? – Zeeshan Sep 29, 2015 at 17:49

I found a way. I just change `JacksonConverterFactory` to `GsonConverterFactory` It converts json into my custom object but it gives warning **Unchecked cast retrofit.Converter<capture<?>>** – Zeeshan Sep 29, 2015 at 18:11

4 What MyError class have? – Dhrupal Aug 9, 2016 at 10:23

0

```
try{
                ResponseBody response = ((HttpException)
t).response().errorBody();
                JSONObject json = new JSONObject( new String(response.bytes())
);
                errMsg = json.getString("message");
            }catch(JSONException e){
                return t.getMessage();
            }
            catch(IOException e){
                return t.getMessage();
            }
```

Share Follow

answered Apr 7, 2017 at 1:47

Mike6679
6,067 ● 20 ● 67 ● 111

0

In Kotlin:

```
val call =
APIClient.getInstance().signIn(AuthRequestWrapper(AuthRequest("1234567890z",
"12341234", "nonce")))
call.enqueue(object : Callback<AuthResponse> {
    override fun onResponse(call: Call<AuthResponse>, response:
Response<AuthResponse>) {
        if (response.isSuccessful) {

        } else {
            val a = object : Annotation{}
            val errorConverter =
RentalGeekClient.getRetrofitInstance().responseBodyConverter<AuthFailureResponse
(AuthFailureResponse::class.java, arrayOf(a))
            val authFailureResponse =
errorConverter.convert(response.errorBody())
        }
    }

    override fun onFailure(call: Call<AuthResponse>, t: Throwable) {
    }
})
```

Share Follow

answered Jun 4, 2017 at 23:48

Adam Johns
36.3k ● 26 ● 128 ● 181

0

errorBody values should set APIError object in Retrofit. So that, you can use the below code structure.

```
public class APIErrorUtils {

    public static APIError parseError(Response<?> response) {
        Converter<ResponseBody, APIError> converter =
API.getClient().responseBodyConverter(APIError.class, new Annotation[0]);

        APIError error;

        try {
            error = converter.convert(response.errorBody());
            Log.d("SERVICELOG",
"****************************************************");
            Log.d("SERVICELOG", "***** SERVICE LOG");
            Log.d("SERVICELOG", "***** TIMESTAMP: " +
String.valueOf(error.getTimestamp()));
            Log.d("SERVICELOG", "***** STATUS: " +
String.valueOf(error.getStatus()));
            Log.d("SERVICELOG", "***** ERROR: " + error.getError());
            Log.d("SERVICELOG", "***** MESSAGE: " + error.getMessage());
            Log.d("SERVICELOG", "***** PATH: " + error.getPath());
            Log.d("SERVICELOG",
"****************************************************");
        } catch (IOException e) {
            return new APIError();
        }

        return error;
    }
}

APIError error = APIErrorUtils.parseError(response);
if (error.getStatus() == 400) {
    ....
}
```

Share Follow

edited Sep 25, 2019 at 5:23
Mehmed
3,040 ● 5 ● 43 ● 65

answered Mar 28, 2018 at 12:39
Egemen Mede
201 ● 1 ● 3

0

```
val error = JSONObject(callApi.errorBody()?.string() as String)
            CustomResult.OnError(CustomNotFoundError(userMessage =
error["userMessage"] as String))

open class CustomError (
    val traceId: String? = null,
    val errorCode: String? = null,
    val systemMessage: String? = null,
    val userMessage: String? = null,
    val cause: Throwable? = null
)
```

```
open class ErrorThrowable(
    private val traceId: String? = null,
    private val errorCode: String? = null,
    private val systemMessage: String? = null,
    private val userMessage: String? = null,
    override val cause: Throwable? = null
) : Throwable(userMessage, cause) {
    fun toError(): CustomError = CustomError(traceId, errorCode, systemMessage,
userMessage, cause)
}


class NetworkError(traceId: String? = null, errorCode: String? = null,
systemMessage: String? = null, userMessage: String? = null, cause: Throwable? =
null):
    CustomError(traceId, errorCode, systemMessage, userMessage?: "Usted no
tiene conexión a internet, active los datos", cause)

class HttpError(traceId: String? = null, errorCode: String? = null,
systemMessage: String? = null, userMessage: String? = null, cause: Throwable? =
null):
    CustomError(traceId, errorCode, systemMessage, userMessage, cause)

class UnknownError(traceId: String? = null, errorCode: String? = null,
systemMessage: String? = null, userMessage: String? = null, cause: Throwable? =
null):
    CustomError(traceId, errorCode, systemMessage, userMessage?: "Unknown
error", cause)

class CustomNotFoundError(traceId: String? = null, errorCode: String? = null,
systemMessage: String? = null, userMessage: String? = null, cause: Throwable? =
null):
    CustomError(traceId, errorCode, systemMessage, userMessage?: "Data not
found", cause)`
```

Share Follow

answered Jan 27, 2020 at 21:40

Gary Loyola
34 ● 3

0

Error body handling in kotlin Android

```
catch (cause: Throwable) {
            when (cause) {
                is HttpException -> {
                    try {
                        val YourErrorResponseClassObj =
Gson().fromJson(cause.response()?.errorBody()?.charStream(),
YourErrorResponseClass::class.java)
                    } catch (e: Exception) {

                    }
                }
                else -> {
                    //Other errors like Network ...
                }
            }
        }
```

Share Follow

answered Aug 5, 2021 at 8:18

Abhishek Garg
3,242 ● 29 ● 33

0

very simple. and this save my life ever

```
public static void displayApiResponseErrorBody(Response<?> response)
{
    InputStream i = response.errorBody().byteStream();
    BufferedReader r = new BufferedReader(new InputStreamReader(i));
    StringBuilder errorResult = new StringBuilder();
    String line;
    try {
        while ((line = r.readLine()) != null)
        {
            errorResult.append(line).append('\n');
        }
        Log.d("API_RESPONSE_ERROR_BODY",String.valueOf(errorResult));
        System.out.println(errorResult);
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Share Follow

answered Nov 27, 2021 at 14:05

hamil.Dev
388 ● 3 ● 10

0

In case of retrofit error Response, You can get body using error.getResponse(), Here is the example.

```
        @Override
        public void failure(RetrofitError error){
            if(error.getResponse().getStatus()==201){
                LogUtil.INSTANCE.debug("Success : " + error.toString());
                callback.success(error.getResponse().getBody);
            }else{
                LogUtil.INSTANCE.debug("failure: " + error.toString());
                callback.failure(error);
            }
        }
```

Share Follow

answered Mar 11, 2022 at 3:42
Farid Haq
4,161 ● 1 ● 22 ● 15

0

```
val reader =
BufferedReader(response.errorBody()?.source().inputStream().reader())

val content = StringBuilder()

reader.use { readerBuffer ->
    var line = readerBuffer.readLine()
    while (line != null) {
        content.append(line)
        line = readerBuffer.readLine()
    }
}

Gson().fromJson(content.toString(), ResponseData::class.java)
```

Share Follow

edited Aug 23, 2022 at 6:02
Procrastinator
2,654 ● 42 ● 30 ● 37

answered Aug 18, 2022 at 3:26
Hả Hải Đăng Nguyễn Đăng 1 ● 1

Your answer could be improved with additional supporting information. Please edit to add further details, such as citations or documentation, so that others can confirm that your answer is correct. You can find more information on how to write good answers in the help center. – Jonas Aug 23, 2022 at 7:29