DISTRIBUTED SYSTEMS

Lab 9

João Leitão, Sérgio Duarte, Pedro Camponês

GOALS

In the end of this lab you should be able to:

- Understand what is Oauth
- How to register an application with Imgur
- How to generate the credentials needed for Oauth in Imgur
- How to take advantage of the REST API documentation of Imgur
- Know how to make requests to Imgur using Oauth using the library ScribeJava

GOALS

In the end of this lab you should be able to:

- **Understand what is Oauth**
- How to register an application with Imgur
- How to generate the credentials needed for Oauth in Imgur
- How to take advantage of the REST API documentation of Imgur
- Know how to make requests to Imgur using Oauth using the library ScribeJava

OAUTH

Many online services only allow access through secure channels with client authentication.

- Secure channels are provided by SSL/TLS
- Client authentication is provided by Oauth

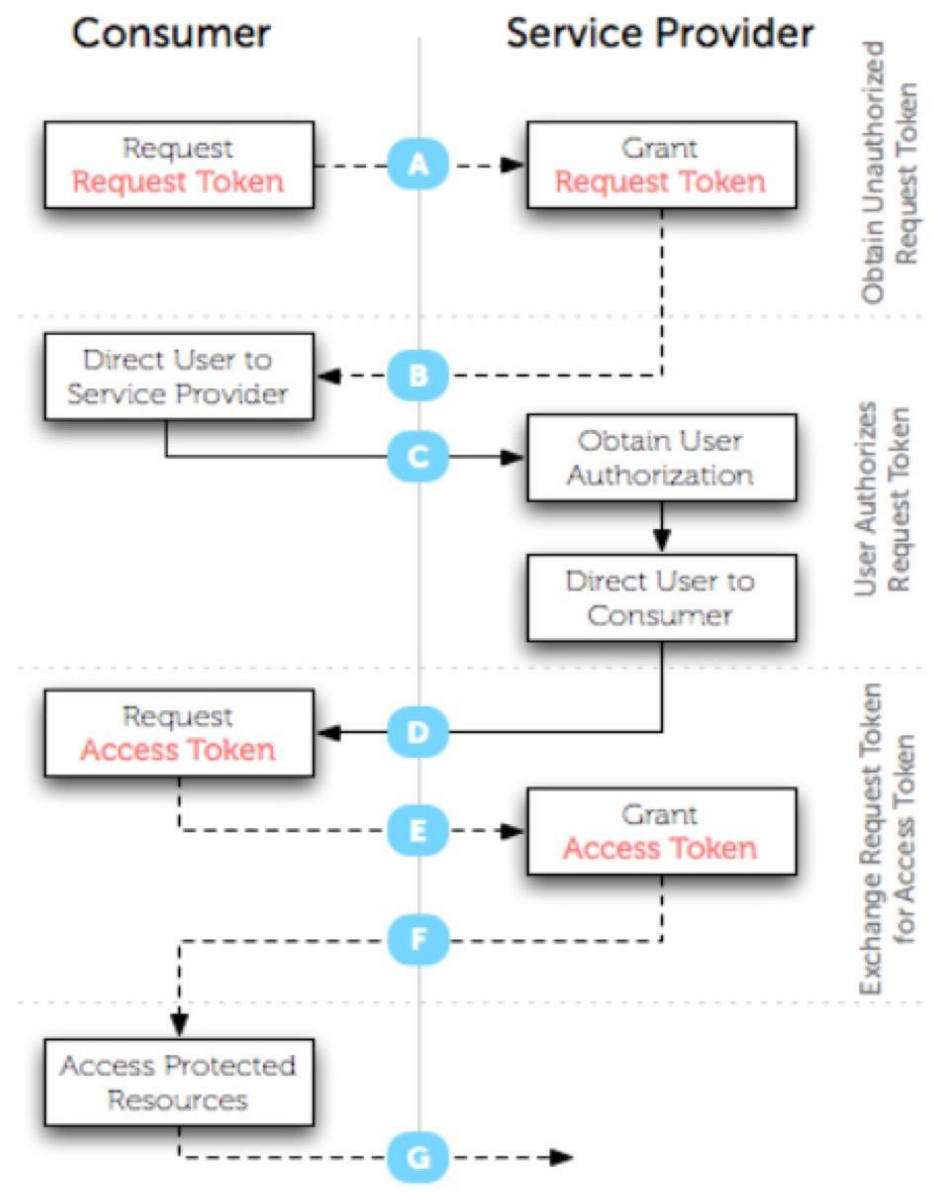
Oauth is also used to allow users of a given application to access their resources in an external service to the application (e.g., authentication with Google or Facebook, or storing your application files in Google Drive, or images in Imgur) without sharing the users' credentials.

OAUTH TYPICAL WORKFLOW

——→ Person Using Web Browser / Manual Entry

- - - - → Consumer / Service Provider

OAUTH AUTHENTICATION FLOW v1.0a



Consumer
Service Provider
Request Request Token
A
Grant Request Token
Obtain Unauthorized Request Token
Direct User to Service Provider
B
C
Obtain User Authorization
Direct User to Consumer
User Authorizes Request Token
Request Access Token
D
E
Grant Access Token
F
Exchange Request Token for Access Token
Access Protected Resources
G

A Consumer Requests Request Token

Request includes
oauth_consumer_key
oauth_signature_method
oauth_signature
oauth_timestamp
oauth_nonce
oauth_version (optional)
oauth_callback

B Service Provider Grants Request Token

Response includes
oauth_token
oauth_token_secret
oauth_callback_confirmed

C Consumer Directs User to Service Provider

Request includes
oauth_token (optional)

D Service Provider Directs User to Consumer

Request includes
oauth_token
oauth_verifier

E Consumer Requests Access Token

Request includes
oauth_consumer_key
oauth_token
oauth_signature_method
oauth_signature
oauth_timestamp
oauth_nonce
oauth_version (optional)
oauth_verifier

F Service Provider Grants Access Token

Response includes
oauth_token
oauth_token_secret

G Consumer Accesses Protected Resources

Request includes
oauth_consumer_key
oauth_token
oauth_signature_method
oauth_signature
oauth_timestamp
oauth_nonce
oauth_version (optional)

OAUTH IN THE CONTEXT OF DISTRIBUTED SYSTEMS

Applications that wish to use user’s resources in some external service must register with that service

- (e.g., a web application wants to allow their users to store/access files in their own Imgur account -> application must be registered to Imgur)
- This step creates the authentication pieces for the applications: API KEY and API SECRET

OAUTH IN THE CONTEXT OF DISTRIBUTED SYSTEMS

A final user when interacting with the application, is required to authenticate to the external service and allow access to their resources in that service.

- (e.g., users using the application will be redirected to a login page of Imgur, authenticate with their own account, and authorize the application to access their Imgur albums and images)
- This creates the authentication piece for that (user, application) pair: ACCESS TOKEN
- This token included in all API requests made from the application, authenticating both the web application and user.
- The final user credentials (e.g., Imgur) are never shared with the application!

OAUTH IN THE CONTEXT OF DISTRIBUTED SYSTEMS

A final user when interacting with the application, is required to authenticate to the external service and allow access to their reso

- (e n pa au im
- Th ion) pa
- Th ap er.

In the context of the Distributed Systems course we are going to simplify this process. Hence the account of the external service being used (i.e., Imgur) is the account of the application developer instead of the final user. This means that the ACCESS TOKEN will also be created by the application developer (i.e., You ☺).

- The final user credentials (e.g., Imgur) are never shared with the application!

GOALS

In the end of this lab you should be able to:

- Understand what is Oauth
- **How to register an application with Imgur**
- How to generate the credentials needed for Oauth in Imgur
- How to take advantage of the REST API documentation of Imgur
- Know how to make requests to Imgur using Oauth using the library ScribeJava

Register You application with Imgur

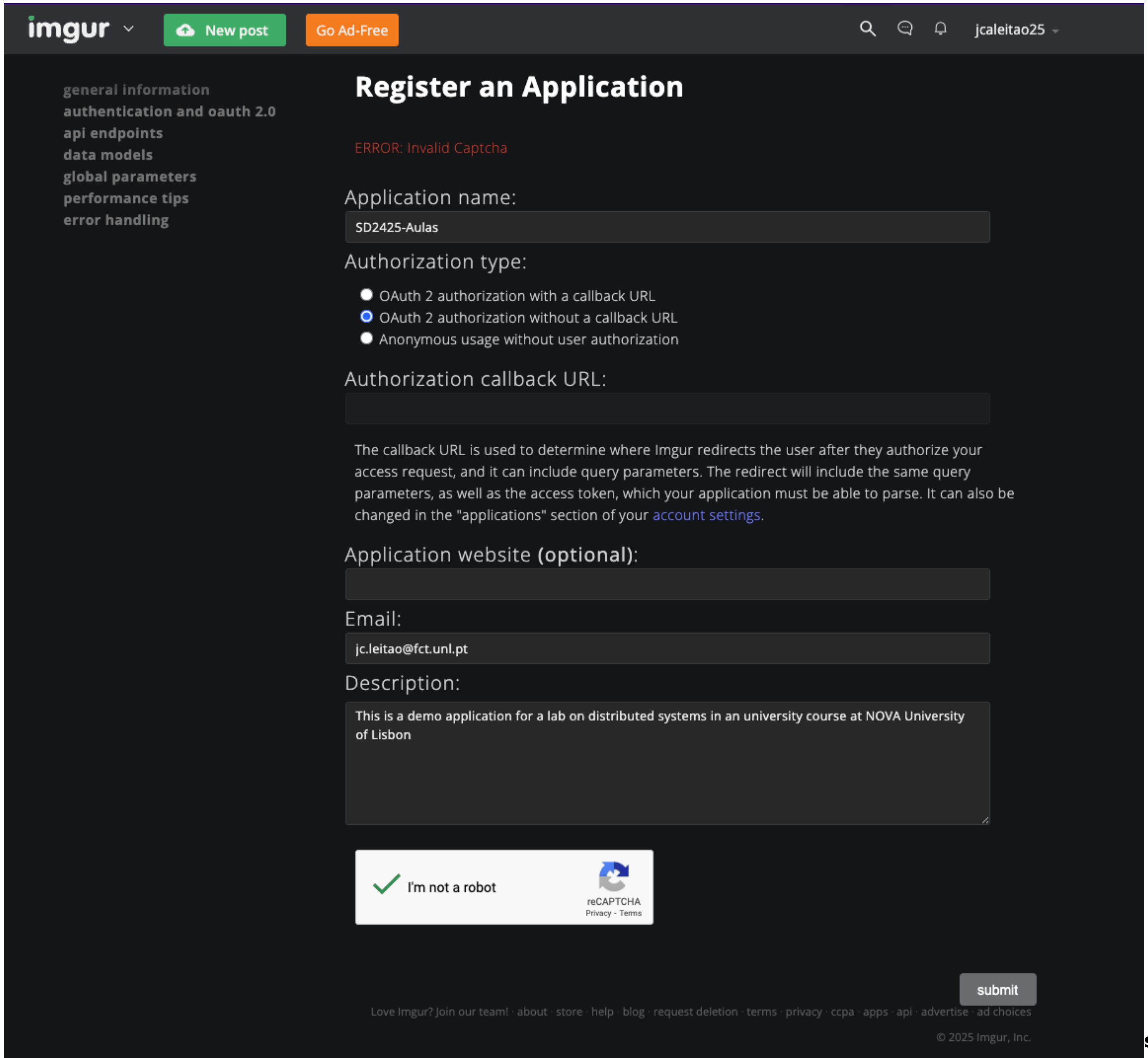
1. You need to create an account with Imgur

   https://imgur.com/register

2. After that you will have to register your application with Imgur by accessing the following URL (after having logged in into the Imgur account)

   https://api.imgur.com/oauth2/addclient

Register You application with Imgur



Register an Application

ERROR: Invalid Captcha

Application name: SD2425-Aulas

Authorization type:

- ○ OAuth 2 authorization with a callback URL
- [✓ OAuth 2 authorization without a callback URL]
- ○ Anonymous usage without user authorization

Authorization callback URL:

The callback URL is used to determine where Imgur redirects the user after they authorize your access request, and it can include query parameters. The redirect will include the same query parameters, as well as the access token, which your application must be able to parse. It can also be changed in the "applications" section of your account settings.

Application website **(optional)**:

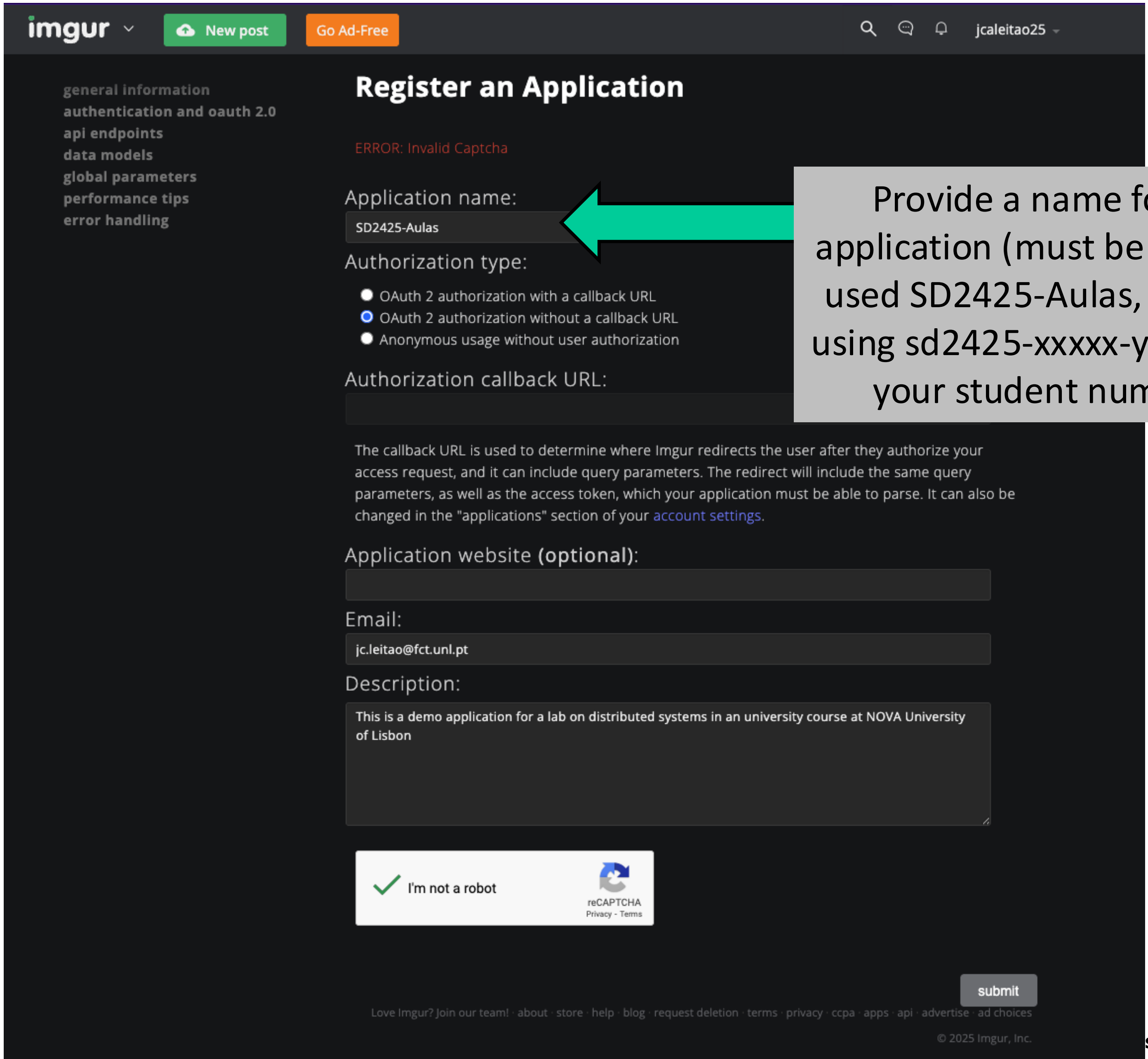
Email: jc.leitao@fct.unl.pt

Description: This is a demo application for a lab on distributed systems in an university course at NOVA University of Lisbon

[✓ I'm not a robot]

reCAPTCHA
Privacy - Terms
submit
Love Imgur? Join our team! · about · store · help · blog · request deletion · terms · privacy · ccpa · apps · api · advertise · ad choices
© 2025 Imgur, Inc.

Register You application with Imgur



Register an Application

ERROR: Invalid Captcha

Application name: SD2425-Aulas

Authorization type:
- OAuth 2 authorization with a callback URL
- [✓ OAuth 2 authorization without a callback URL]
- Anonymous usage without user authorization

Authorization callback URL:

The callback URL is used to determine where Imgur redirects the user after they authorize your access request, and it can include query parameters. The redirect will include the same query parameters, as well as the access token, which your application must be able to parse. It can also be changed in the "applications" section of your account settings.

Application website **(optional)**:

Email: jc.leitao@fct.unl.pt

Description: This is a demo application for a lab on distributed systems in an university course at NOVA University of Lisbon

I'm not a robot ✓

reCAPTCHA
Privacy - Terms

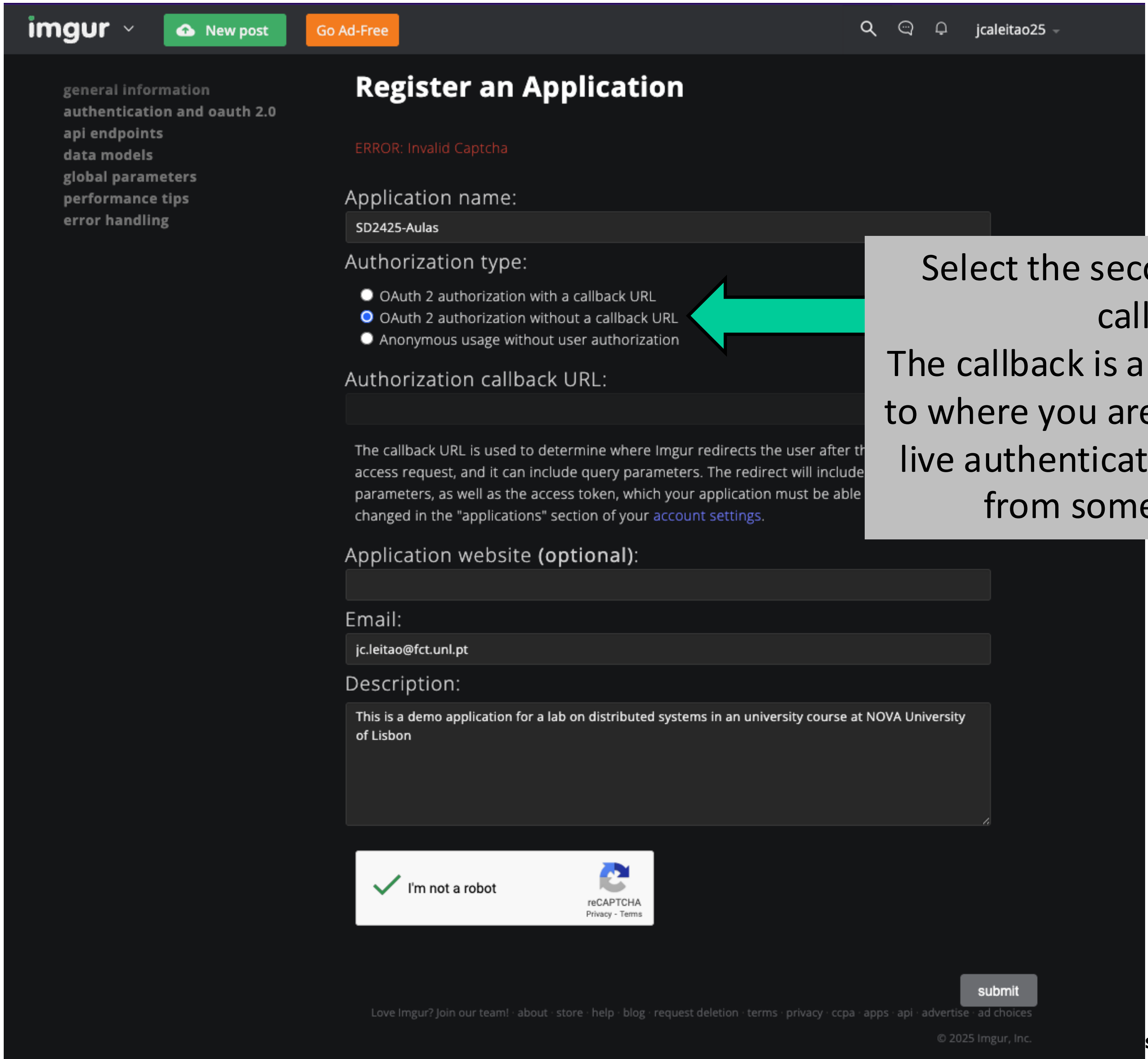
submit

Love Imgur? Join our team! · about · store · help · blog · request deletion · terms · privacy · ccpa · apps · api · advertise · ad choices

© 2025 Imgur, Inc.

Provide a name for the application (must be unique) I used SD2425-Aulas, consider using sd2425-xxxxx-yyyyy with your student numbers.

Register You application with Imgur



imgur · New post · Go Ad-Free · jcaleitao25

general information
authentication and oauth 2.0
api endpoints
data models
global parameters
performance tips
error handling

Register an Application

ERROR: Invalid Captcha

Application name:
SD2425-Aulas

Authorization type:
- ○ OAuth 2 authorization with a callback URL
- [✓ OAuth 2 authorization without a callback URL]
- ○ Anonymous usage without user authorization

Authorization callback URL:

The callback URL is used to determine where Imgur redirects the user after th access request, and it can include query parameters. The redirect will include parameters, as well as the access token, which your application must be able changed in the "applications" section of your account settings.

Application website **(optional)**:

Email:
jc.leitao@fct.unl.pt

Description:
This is a demo application for a lab on distributed systems in an university course at NOVA University of Lisbon

✓ I'm not a robot — reCAPTCHA Privacy - Terms

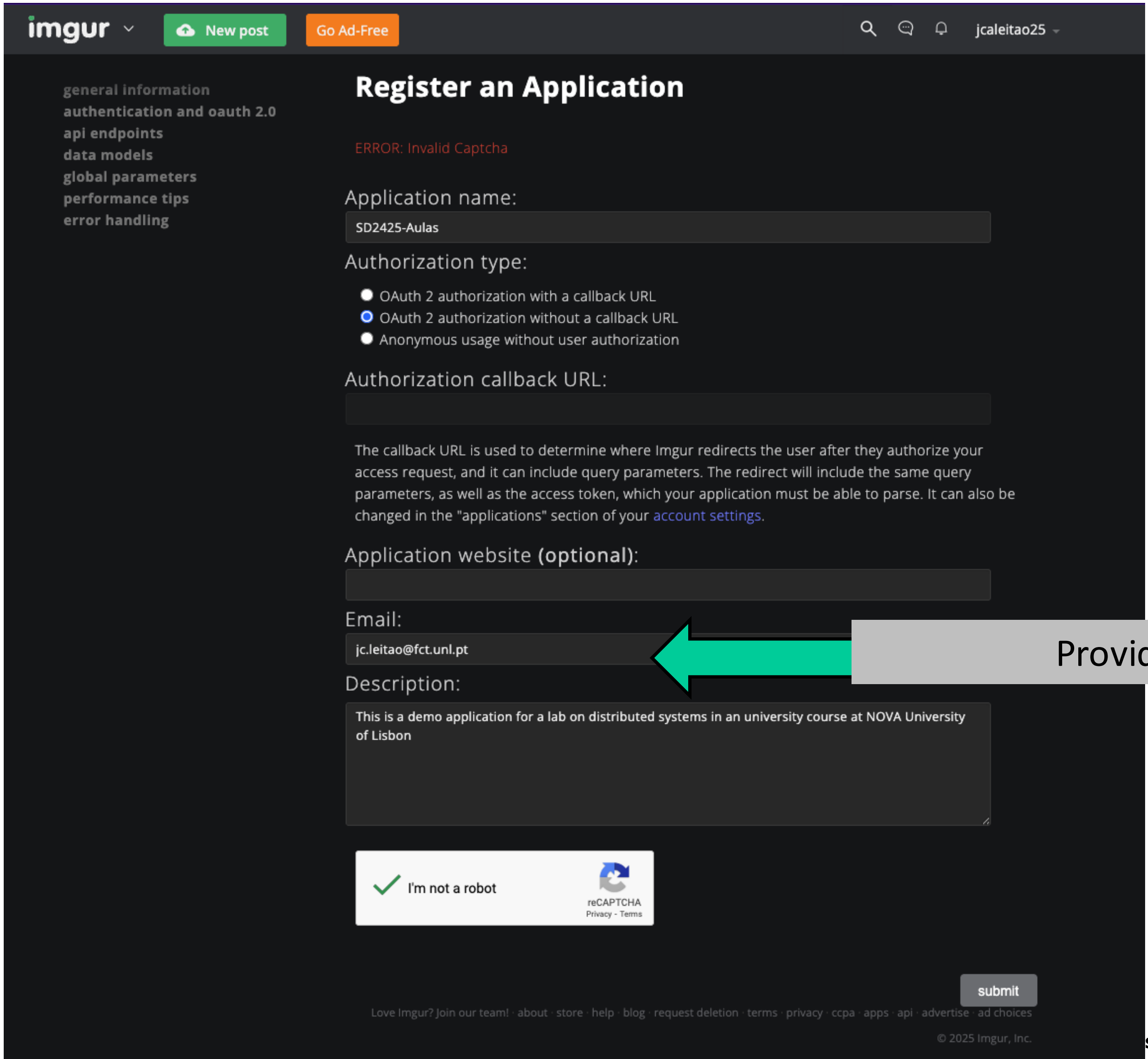
submit

Love Imgur? Join our team! · about · store · help · blog · request deletion · terms · privacy · ccpa · apps · api · advertise · ad choices

© 2025 Imgur, Inc.

Select the second option (without a callback URL).
The callback is a URL of your application to where you are redirected when doing live authentication of users originating from some web application.

REGISTER YOU APPLICATION WITH IMGUR



Register an Application

ERROR: Invalid Captcha

Application name: SD2425-Aulas

Authorization type:

- ○ OAuth 2 authorization with a callback URL
- [✓ OAuth 2 authorization without a callback URL]
- ○ Anonymous usage without user authorization

Authorization callback URL:

The callback URL is used to determine where Imgur redirects the user after they authorize your access request, and it can include query parameters. The redirect will include the same query parameters, as well as the access token, which your application must be able to parse. It can also be changed in the "applications" section of your account settings.

Application website **(optional)**:

Email: jc.leitao@fct.unl.pt

Description: This is a demo application for a lab on distributed systems in an university course at NOVA University of Lisbon

✓ I'm not a robot — reCAPTCHA Privacy - Terms

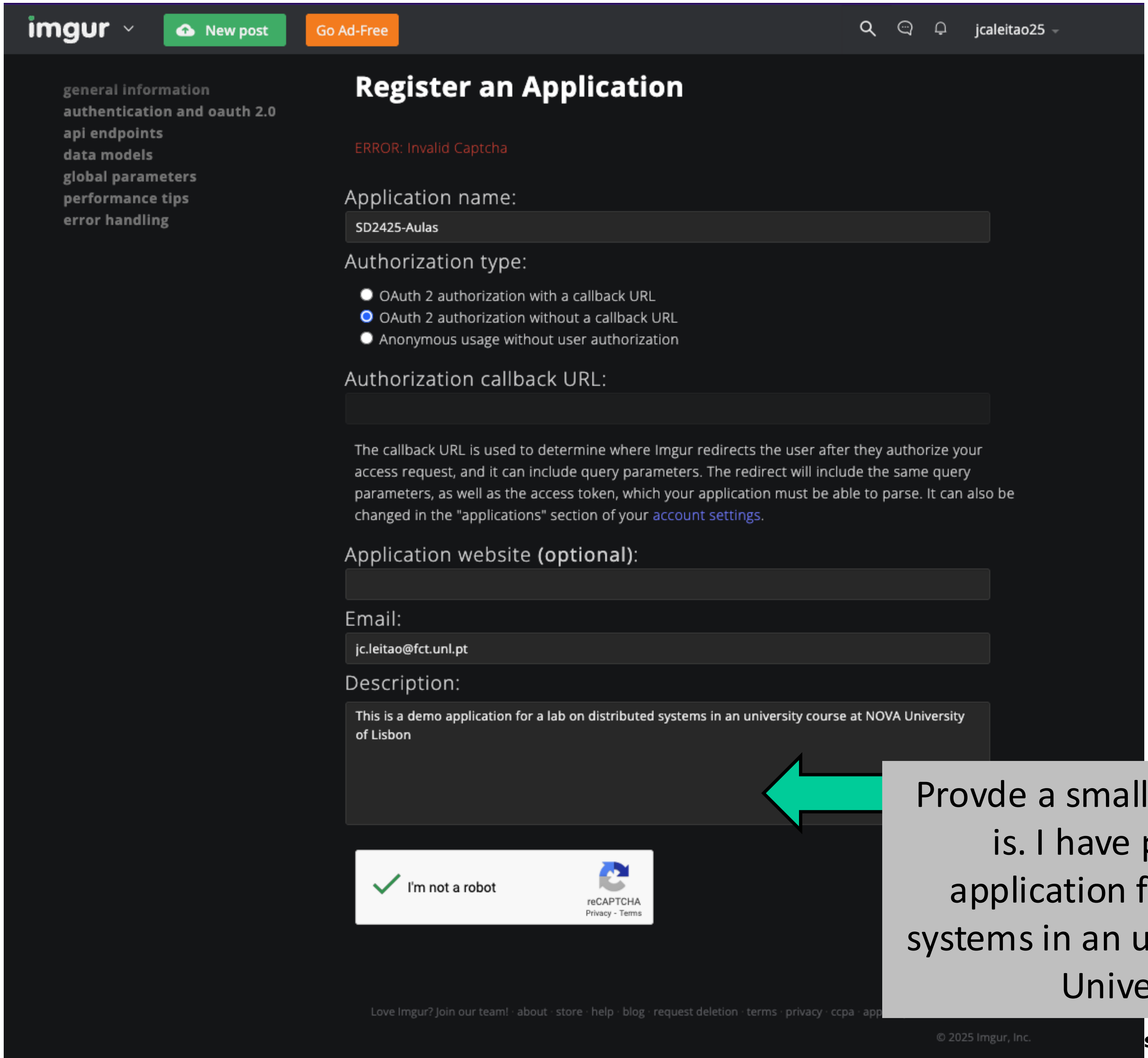
submit

Love Imgur? Join our team! · about · store · help · blog · request deletion · terms · privacy · ccpa · apps · api · advertise · ad choices

© 2025 Imgur, Inc.

Provide an e-mail

Register You application with Imgur



Register an Application

ERROR: Invalid Captcha

Application name: SD2425-Aulas

Authorization type:

- ○ OAuth 2 authorization with a callback URL
- [✓ OAuth 2 authorization without a callback URL]
- ○ Anonymous usage without user authorization

Authorization callback URL:

The callback URL is used to determine where Imgur redirects the user after they authorize your access request, and it can include query parameters. The redirect will include the same query parameters, as well as the access token, which your application must be able to parse. It can also be changed in the "applications" section of your account settings.

Application website **(optional)**:

Email: jc.leitao@fct.unl.pt

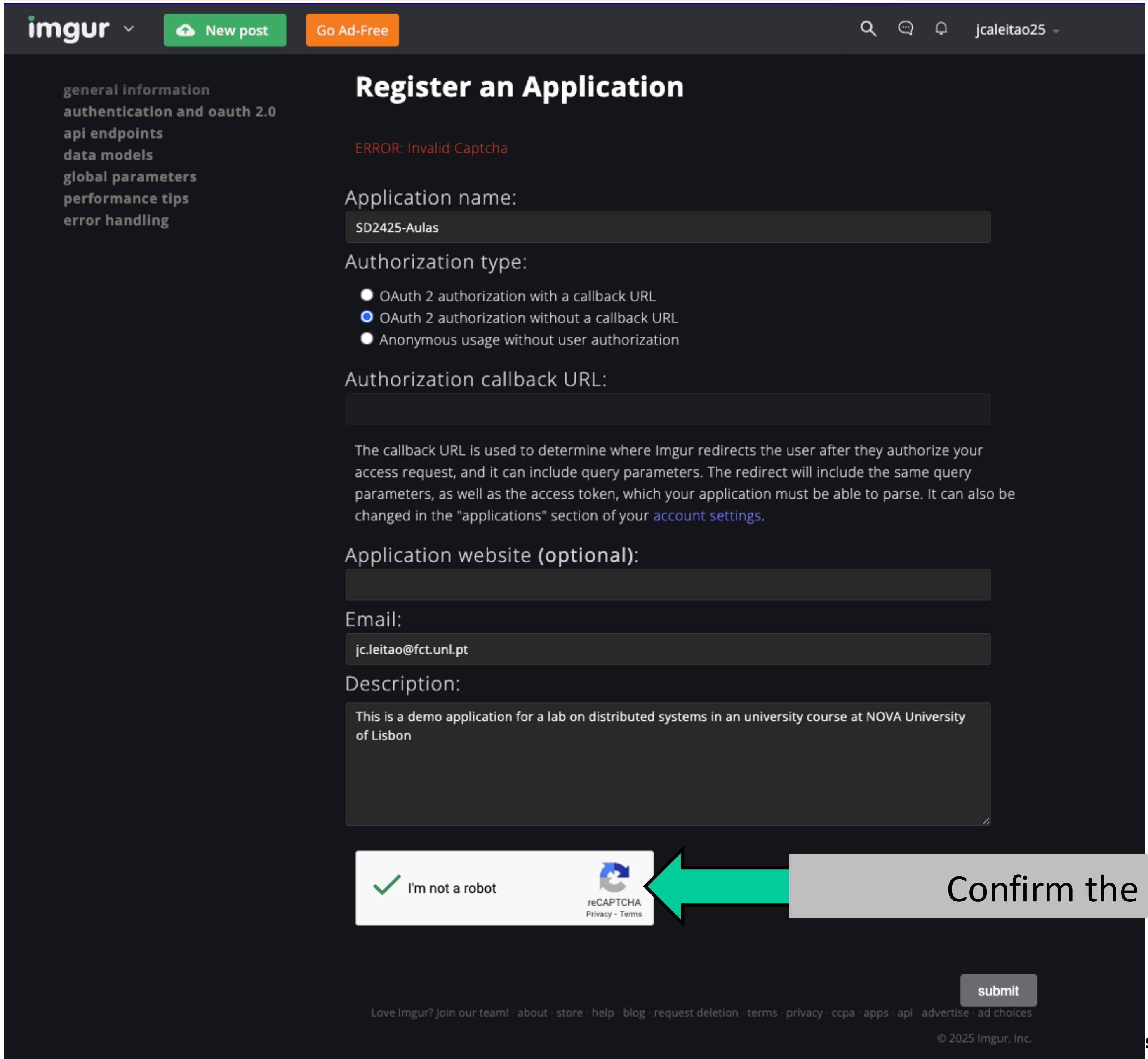
Description: This is a demo application for a lab on distributed systems in an university course at NOVA University of Lisbon

✓ I'm not a robot

reCAPTCHA
Privacy - Terms
Love Imgur? Join our team! · about · store · help · blog · request deletion · terms · privacy · ccpa · app
© 2025 Imgur, Inc.

Provde a small description of what this is. I have put: "This is a demo application for a lab on distributed systems in an university course at NOVA University of Lisbon"

Register You application with Imgur



imgur New post Go Ad-Free jcaleitao25

general information
authentication and oauth 2.0
api endpoints
data models
global parameters
performance tips
error handling

Register an Application

ERROR: Invalid Captcha

Application name:
SD2425-Aulas

Authorization type:
- OAuth 2 authorization with a callback URL
- [✓ OAuth 2 authorization without a callback URL]
- Anonymous usage without user authorization

Authorization callback URL:

The callback URL is used to determine where Imgur redirects the user after they authorize your access request, and it can include query parameters. The redirect will include the same query parameters, as well as the access token, which your application must be able to parse. It can also be changed in the "applications" section of your account settings.

Application website **(optional)**:

Email:
jc.leitao@fct.unl.pt

Description:
This is a demo application for a lab on distributed systems in an university course at NOVA University of Lisbon

✓ I'm not a robot reCAPTCHA Privacy - Terms

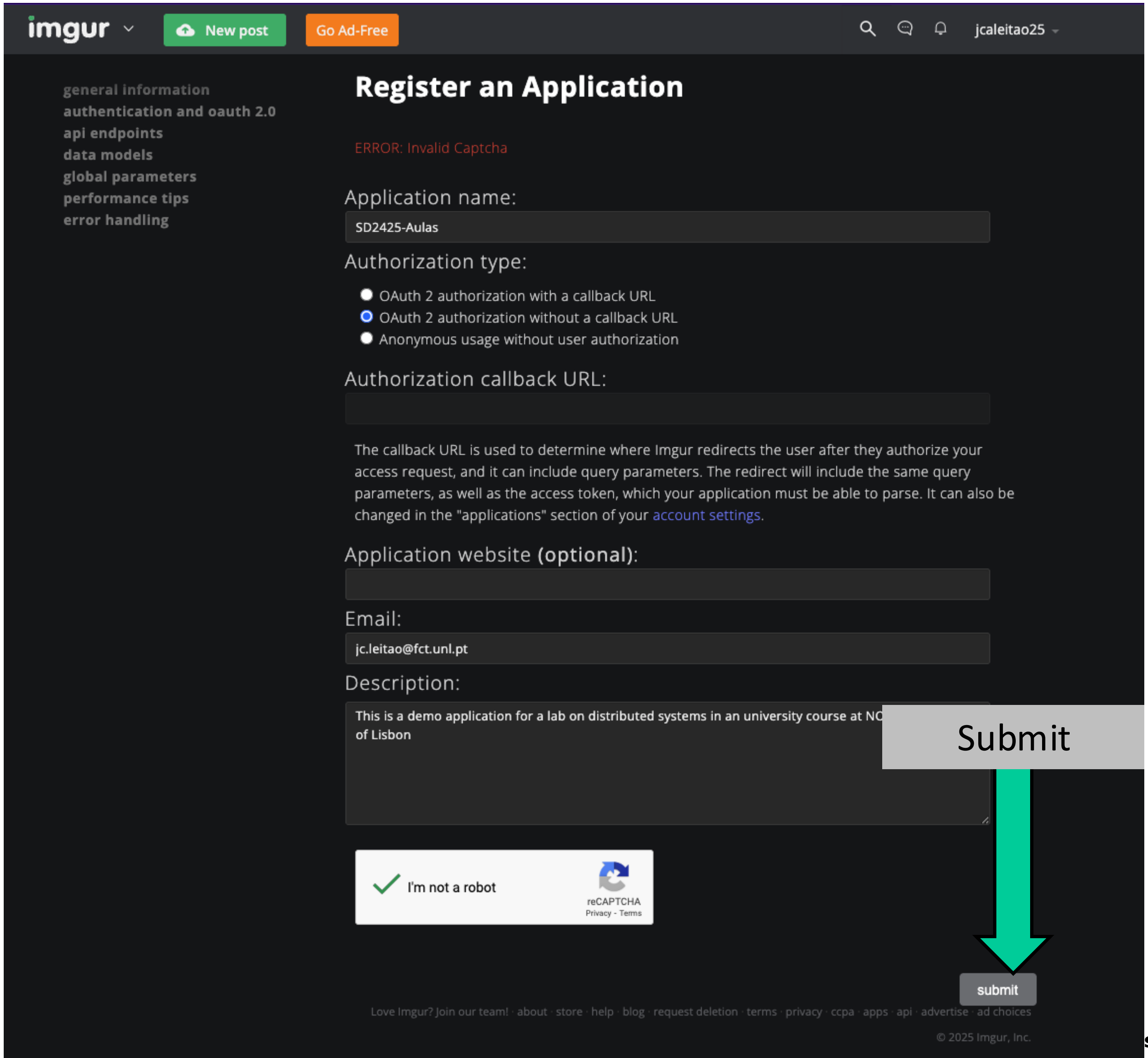
submit

Love Imgur? Join our team! · about · store · help · blog · request deletion · terms · privacy · ccpa · apps · api · advertise · ad choices

© 2025 Imgur, Inc.

Confirm the Captcha ☺

REGISTER YOU APPLICATION WITH IMGUR


imgur
New post
Go Ad-Free
jcaleitao25
general information
authentication and oauth 2.0
api endpoints
data models
global parameters
performance tips
error handling
Submit
submit
Love Imgur? Join our team! · about · store · help · blog · request deletion · terms · privacy · ccpa · apps · api · advertise · ad choices
© 2025 Imgur, Inc.

Register an Application

ERROR: Invalid Captcha

Application name: SD2425-Aulas

Authorization type:

- ○ OAuth 2 authorization with a callback URL
- [✓ OAuth 2 authorization without a callback URL]
- ○ Anonymous usage without user authorization

Authorization callback URL:

The callback URL is used to determine where Imgur redirects the user after they authorize your access request, and it can include query parameters. The redirect will include the same query parameters, as well as the access token, which your application must be able to parse. It can also be changed in the "applications" section of your account settings.

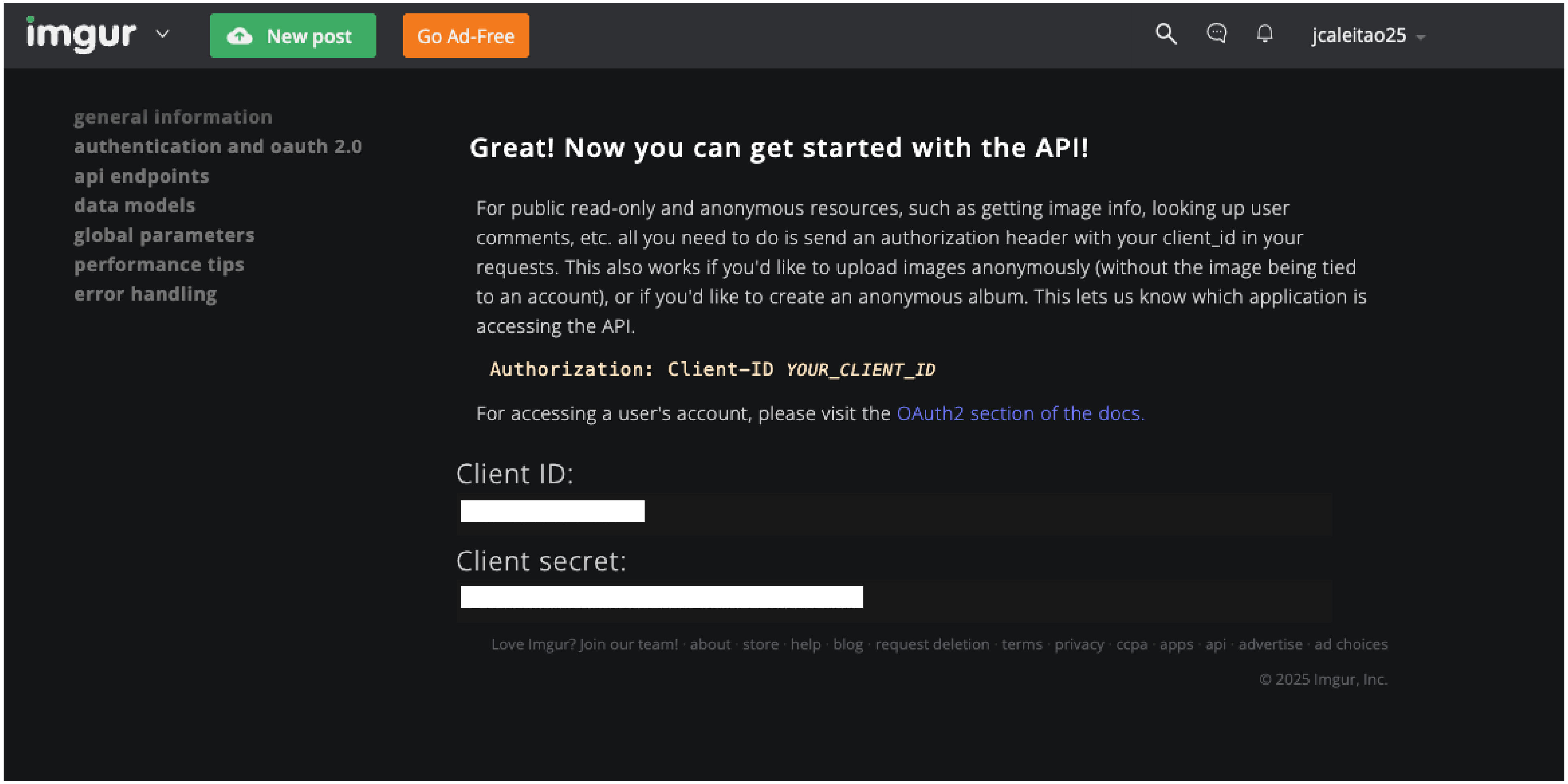
Application website **(optional)**:

Email: jc.leitao@fct.unl.pt

Description: This is a demo application for a lab on distributed systems in an university course at NO of Lisbon

✓ I'm not a robot — reCAPTCHA Privacy - Terms

Register You application with Imgur



Great! Now you can get started with the API!

For public read-only and anonymous resources, such as getting image info, looking up user comments, etc. all you need to do is send an authorization header with your client_id in your requests. This also works if you'd like to upload images anonymously (without the image being tied to an account), or if you'd like to create an anonymous album. This lets us know which application is accessing the API.

```
Authorization: Client-ID YOUR_CLIENT_ID
```

For accessing a user's account, please visit the OAuth2 section of the docs.

Client ID:

[redacted]

Client secret:

[redacted]

Love Imgur? Join our team! · about · store · help · blog · request deletion · terms · privacy · ccpa · apps · api · advertise · ad choices
© 2025 Imgur, Inc.

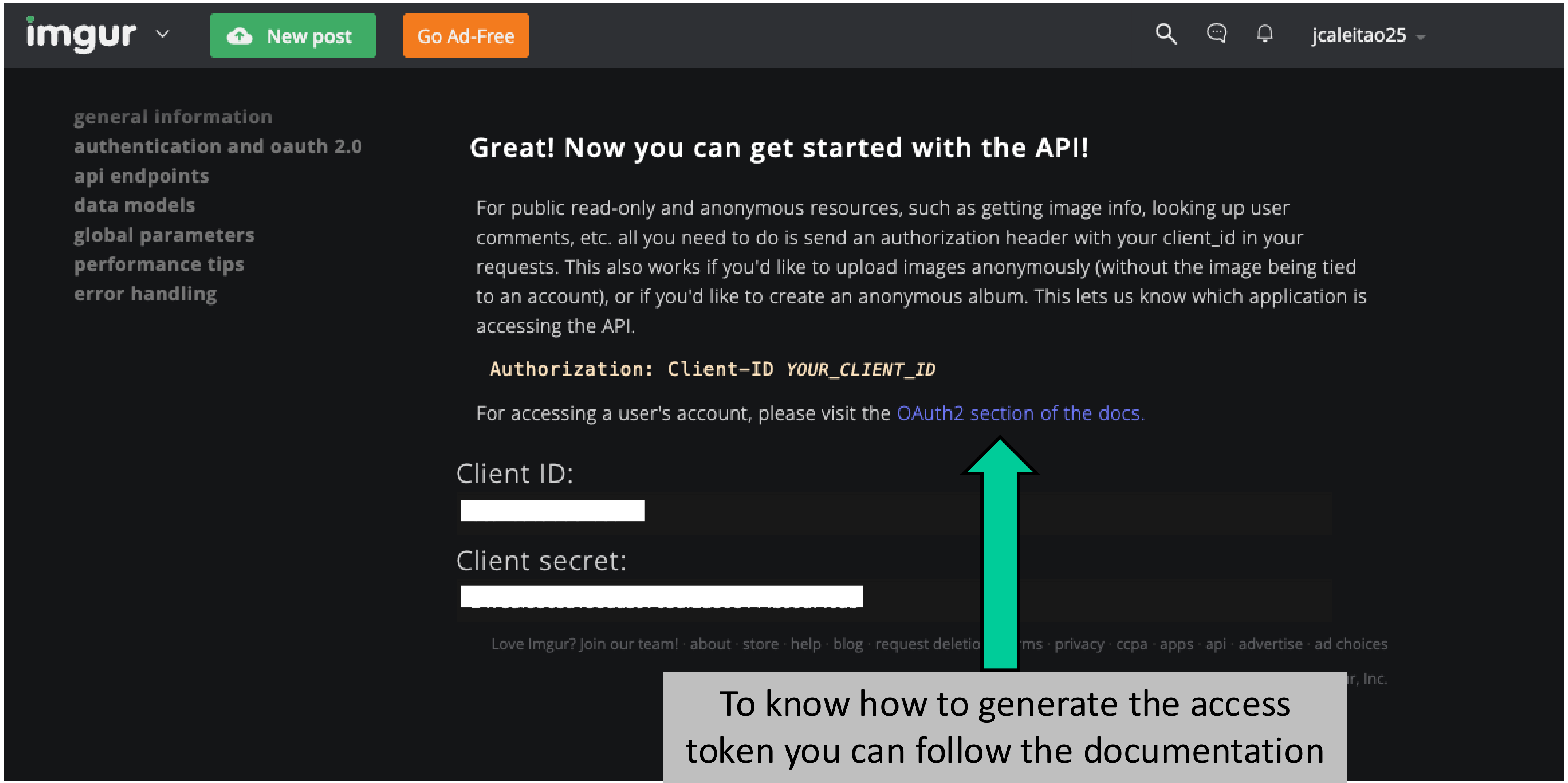
You will be provided your Client ID (API KEY) and Client Secret (API SECRET)

GOALS

In the end of this lab you should be able to:

- Understand what is Oauth
- How to register an application with Imgur
- **How to generate the credentials needed for Oauth in Imgur**
- How to take advantage of the REST API documentation of Imgur
- Know how to make requests to Imgur using Oauth using the library ScribeJava

Get your Access Token to access Imgur



Great! Now you can get started with the API!

For public read-only and anonymous resources, such as getting image info, looking up user comments, etc. all you need to do is send an authorization header with your client_id in your requests. This also works if you'd like to upload images anonymously (without the image being tied to an account), or if you'd like to create an anonymous album. This lets us know which application is accessing the API.

```
Authorization: Client-ID YOUR_CLIENT_ID
```

For accessing a user's account, please visit the OAuth2 section of the docs.

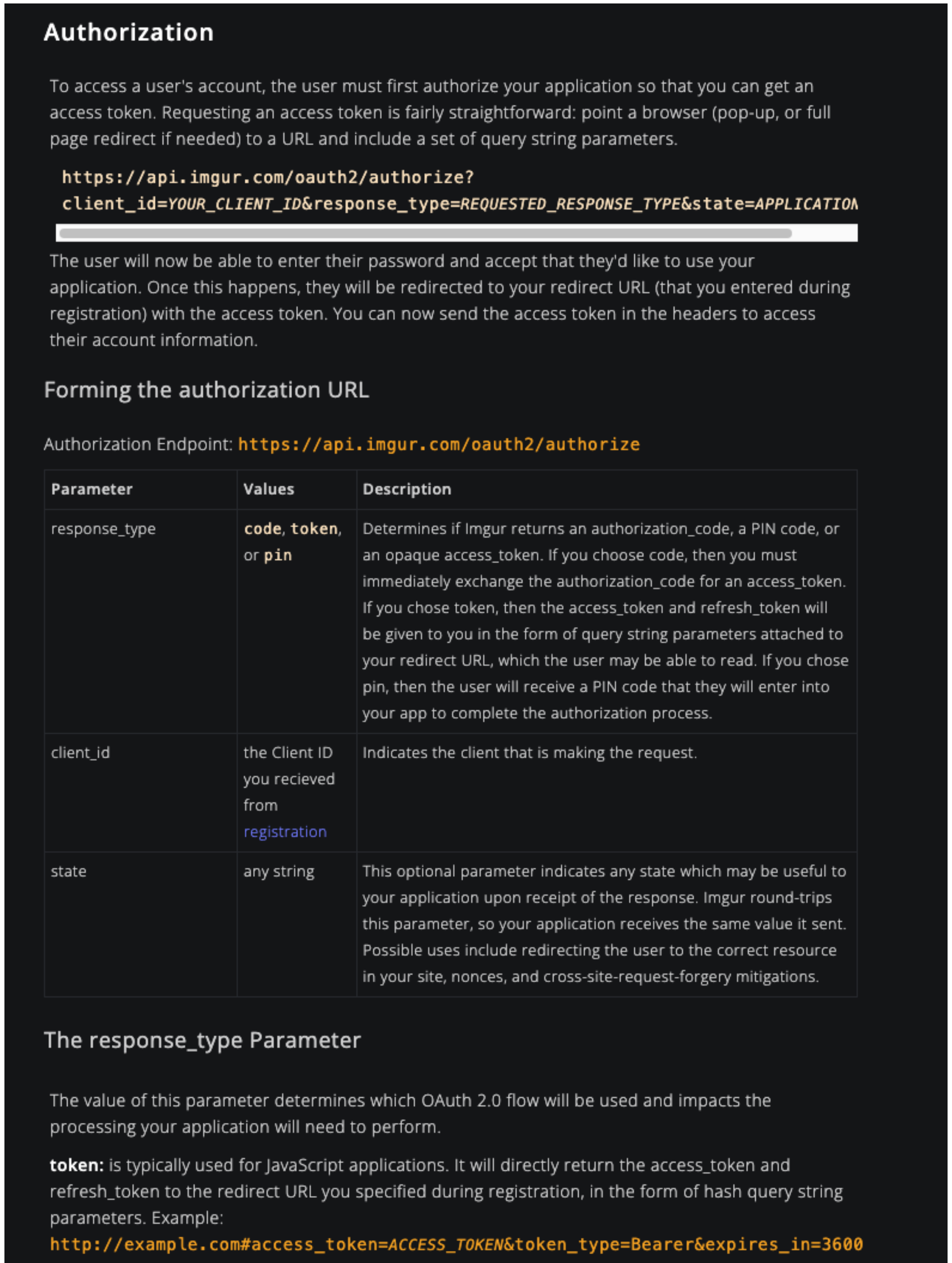
Client ID: [redacted]

Client secret: [redacted]

Love Imgur? Join our team! · about · store · help · blog · request deletio · rms · privacy · ccpa · apps · api · advertise · ad choices
r, Inc.

To know how to generate the access token you can follow the documentation by clicking here

GET YOUR ACCESS TOKEN TO ACCESS IMGUR



Authorization

To access a user's account, the user must first authorize your application so that you can get an access token. Requesting an access token is fairly straightforward: point a browser (pop-up, or full page redirect if needed) to a URL and include a set of query string parameters.

```
https://api.imgur.com/oauth2/authorize?
client_id=YOUR_CLIENT_ID&response_type=REQUESTED_RESPONSE_TYPE&state=APPLICATION
```

The user will now be able to enter their password and accept that they'd like to use your application. Once this happens, they will be redirected to your redirect URL (that you entered during registration) with the access token. You can now send the access token in the headers to access their account information.

Forming the authorization URL

Authorization Endpoint: `https://api.imgur.com/oauth2/authorize`

Parameter	Values	Description
response_type	**code**, **token**, or **pin**	Determines if Imgur returns an authorization_code, a PIN code, or an opaque access_token. If you choose code, then you must immediately exchange the authorization_code for an access_token. If you chose token, then the access_token and refresh_token will be given to you in the form of query string parameters attached to your redirect URL, which the user may be able to read. If you chose pin, then the user will receive a PIN code that they will enter into your app to complete the authorization process.
client_id	the Client ID you recieved from registration	Indicates the client that is making the request.
state	any string	This optional parameter indicates any state which may be useful to your application upon receipt of the response. Imgur round-trips this parameter, so your application receives the same value it sent. Possible uses include redirecting the user to the correct resource in your site, nonces, and cross-site-request-forgery mitigations.

The response_type Parameter

The value of this parameter determines which OAuth 2.0 flow will be used and impacts the processing your application will need to perform.

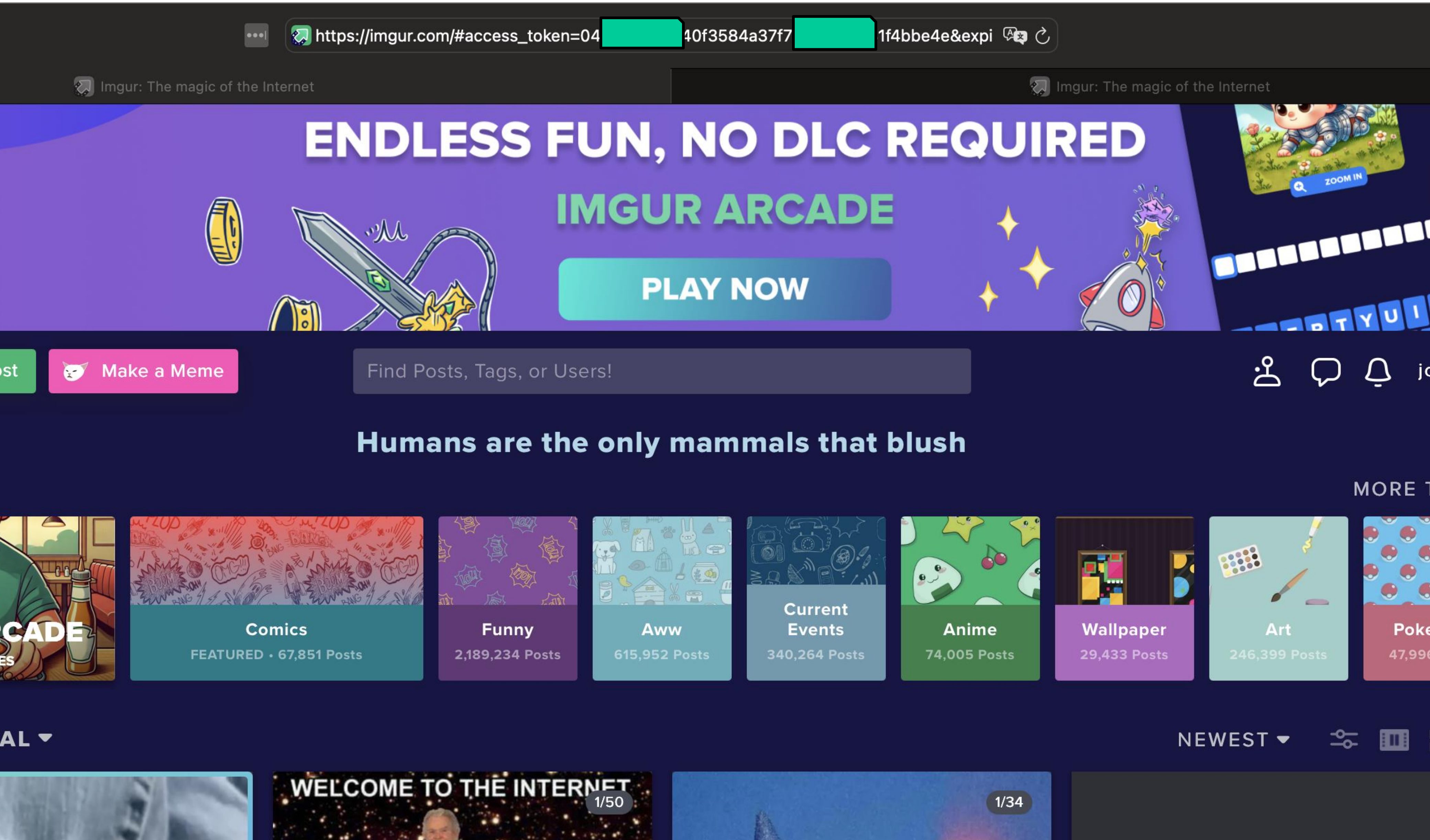
token: is typically used for JavaScript applications. It will directly return the access_token and refresh_token to the redirect URL you specified during registration, in the form of hash query string parameters. Example:
`http://example.com#access_token=ACCESS_TOKEN&token_type=Bearer&expires_in=3600`

This explains the process; you can use and endpoint to generate your access token (that will be returned on the URL for where you will be redirected after confirming that you are willing to give access to the application.

The URL to get the access token is (replacing YOUR_CLIENT_ID with the CLIENT_ID (i.e., API_KEY) you got previously:

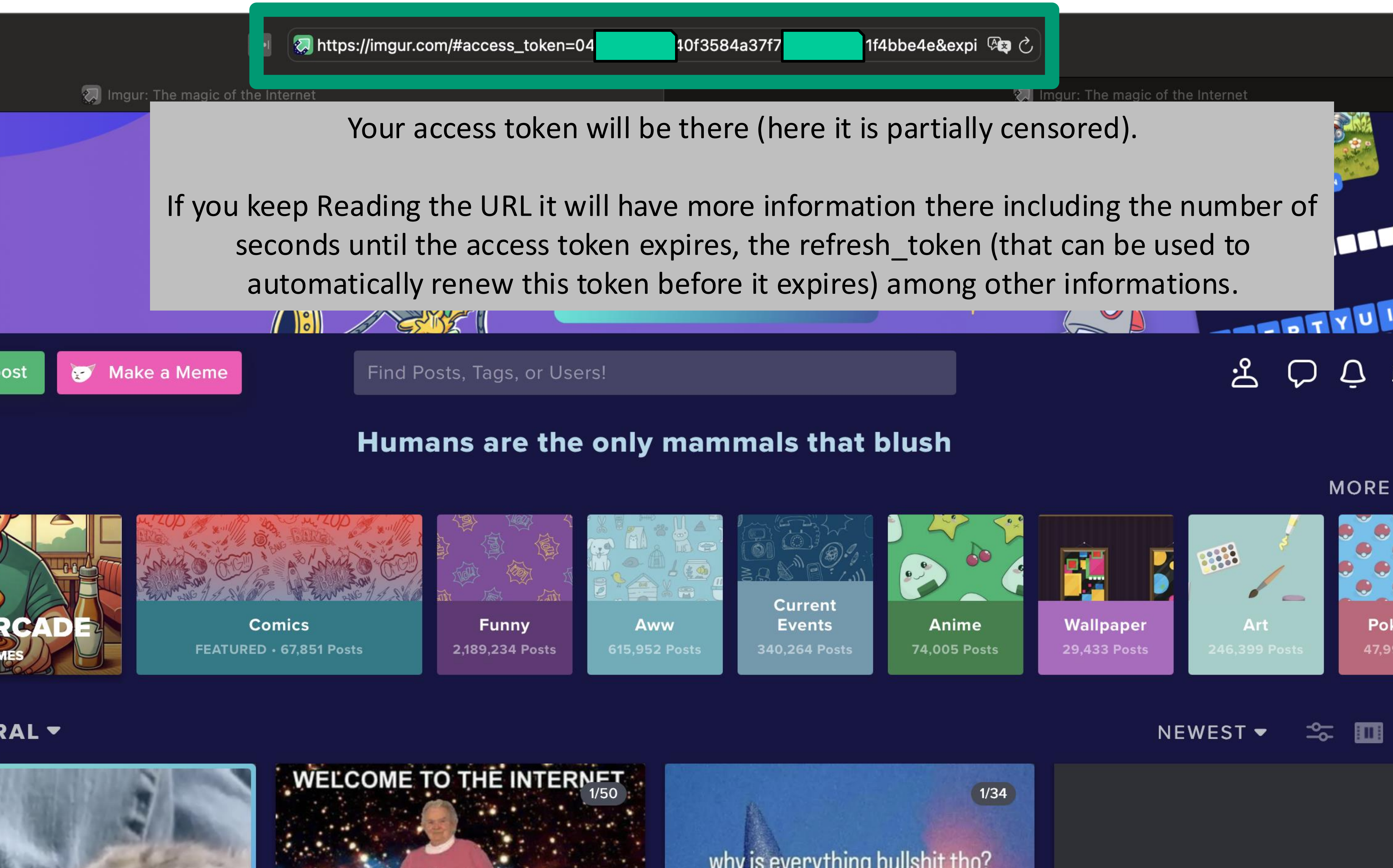
https://api.imgur.com/oauth2/authorize?client_id=*YOUR_CLIENT_ID*&response_type=*token*

Get your Access Token to access Imgur



https://imgur.com/#access_token=04[redacted]40f3584a37f7[redacted]1f4bbe4e&expi

Get your Access Token to access Imgur



https://imgur.com/#access_token=04[redacted]40f3584a37f7[redacted]1f4bbe4e&expi

Your access token will be there (here it is partially censored).

If you keep Reading the URL it will have more information there including the number of seconds until the access token expires, the refresh_token (that can be used to automatically renew this token before it expires) among other informations.

Get your Access Token to access Imgur

https://imgur.com/#access_token=04[redacted]0f3584a37f7[redacted]f4bbe4e&expires_in=315360000&token_type=bearer&refresh_token=c41095599[redacted]176567c741741fdaa1fa&account_username=jcaleitao25&account_id=191062675

The returned URL provides information about:

- Access Token
- Expiration of the access token (in seconds)
- Type of token (bearer, whoever has it can use it in the context of this application)
- Refresh Token
- Username and Identifier of the user account.

GOALS

In the end of this lab you should be able to:

- Understand what is Oauth
- How to register an application with Imgur
- How to generate the credentials needed for Oauth in Imgur
- **How to take advantage of the REST API documentation of Imgur**
- Know how to make requests to Imgur using Oauth using the library ScribeJava

Imgur API Documentation

One of the benefits of Imgur is that the REST API is well documented and it follows most of the conventions specified in the REST model that we have been discussing in lectures and throughout the semester.

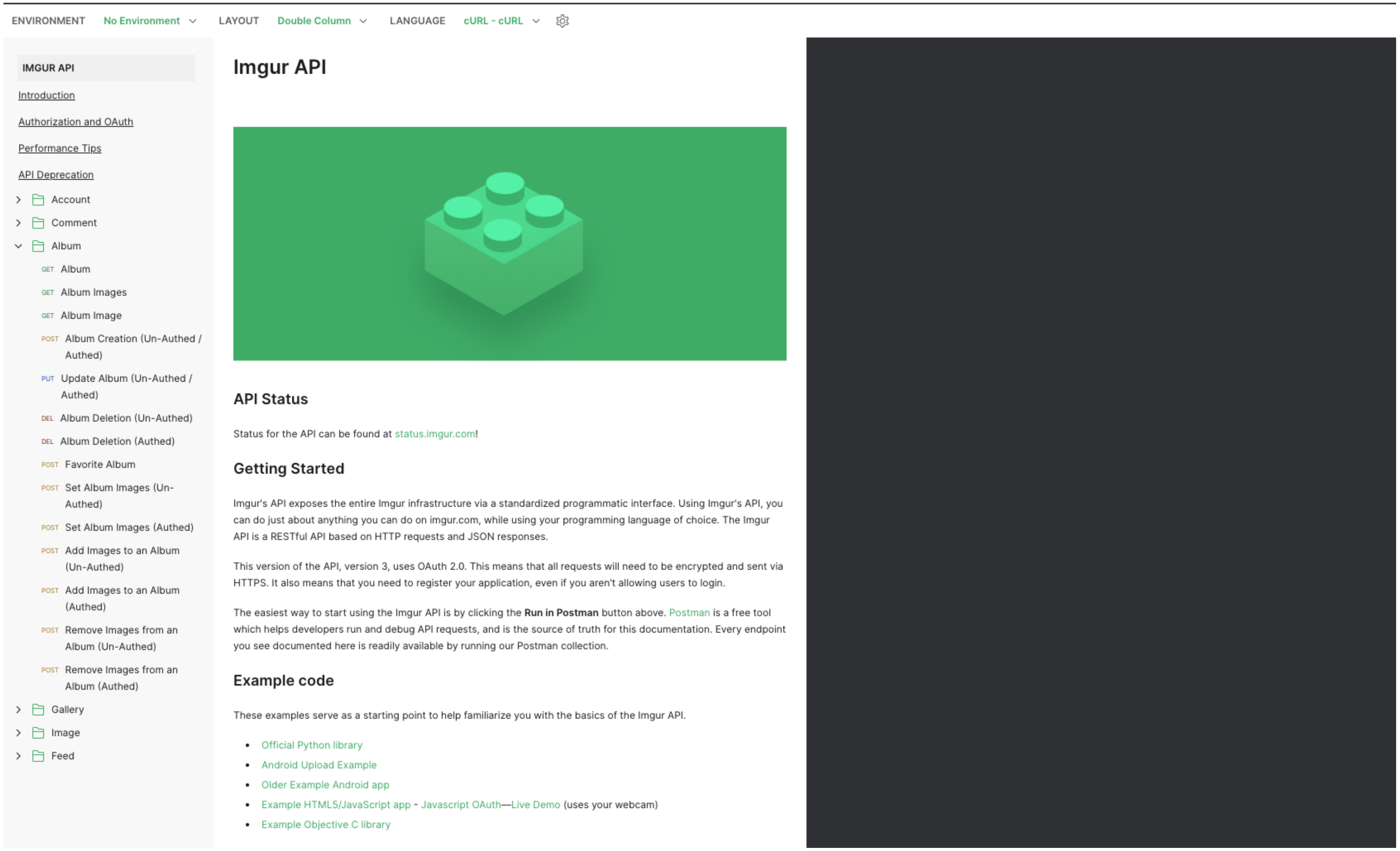
It can be found here: https://apidocs.imgur.com/#intro

Some importante observations:

- If you want to send arguments in the body in Json (encoded from Java objects) you need to add a header to your requests
- You will have to study and decide which operations of the API are more suitable for your Project. Here we only discuss a few operations that might be useful while doing the Project.

Imgur API Documentation

https://apidocs.imgur.com/#intro



Imgur API

API Status

Status for the API can be found at status.imgur.com!

Getting Started

Imgur's API exposes the entire Imgur infrastructure via a standardized programmatic interface. Using Imgur's API, you can do just about anything you can do on imgur.com, while using your programming language of choice. The Imgur API is a RESTful API based on HTTP requests and JSON responses.

This version of the API, version 3, uses OAuth 2.0. This means that all requests will need to be encrypted and sent via HTTPS. It also means that you need to register your application, even if you aren't allowing users to login.

The easiest way to start using the Imgur API is by clicking the **Run in Postman** button above. Postman is a free tool which helps developers run and debug API requests, and is the source of truth for this documentation. Every endpoint you see documented here is readily available by running our Postman collection.

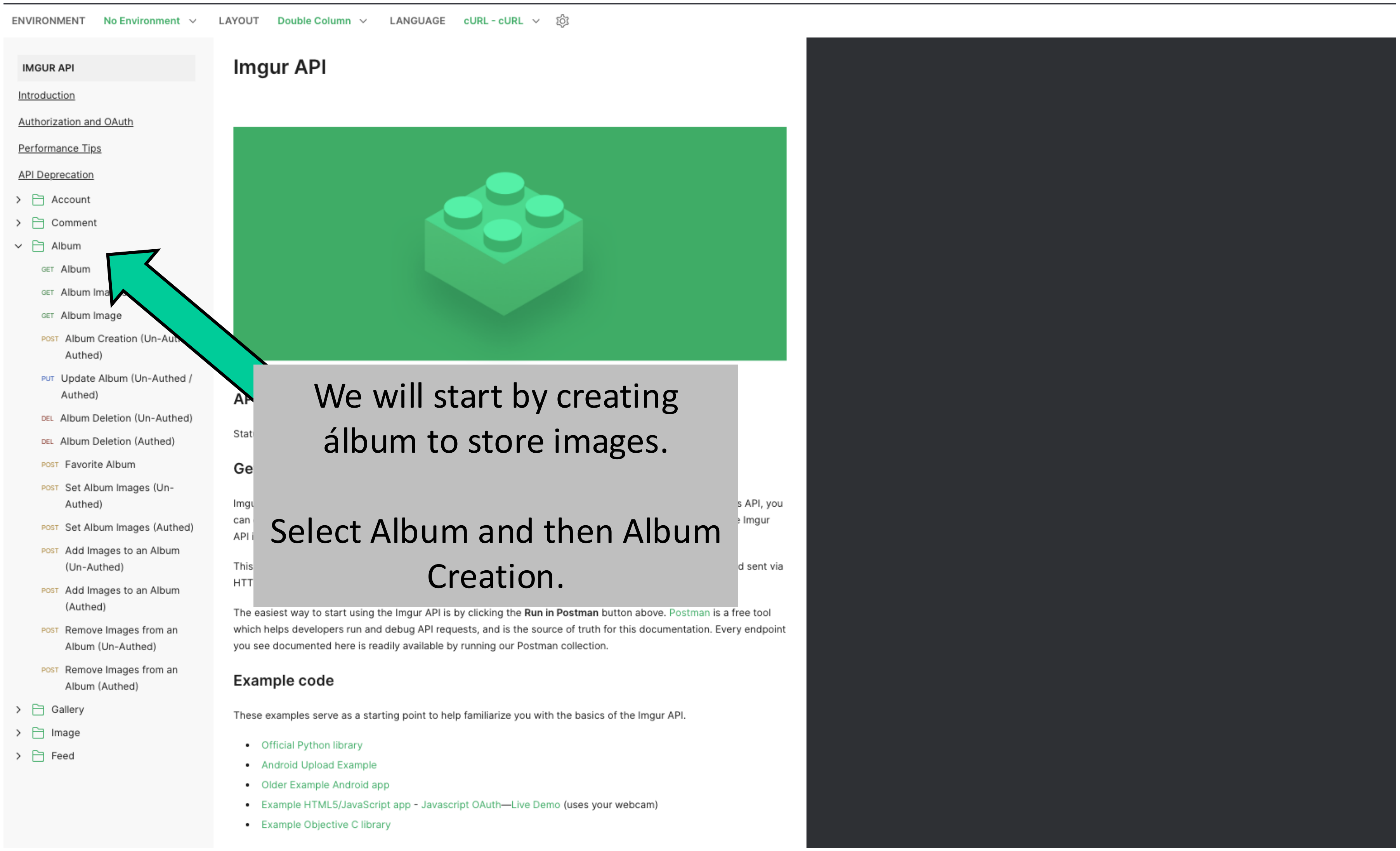
Example code

These examples serve as a starting point to help familiarize you with the basics of the Imgur API.

- Official Python library
- Android Upload Example
- Older Example Android app
- Example HTML5/JavaScript app - Javascript OAuth—Live Demo (uses your webcam)
- Example Objective C library

IMGUR API DOCUMENTATION

https://apidocs.imgur.com/#intro


ENVIRONMENT
No Environment
LAYOUT
Double Column
LANGUAGE
cURL - cURL
IMGUR API
Introduction
Authorization and OAuth
Performance Tips
API Deprecation
Account
Comment
Album
GET Album
GET Album Ima
GET Album Image
POST Album Creation (Un-Au Authed)
PUT Update Album (Un-Authed / Authed)
DEL Album Deletion (Un-Authed)
DEL Album Deletion (Authed)
POST Favorite Album
POST Set Album Images (Un-Authed)
POST Set Album Images (Authed)
POST Add Images to an Album (Un-Authed)
POST Add Images to an Album (Authed)
POST Remove Images from an Album (Un-Authed)
POST Remove Images from an Album (Authed)
Gallery
Image
Feed

Imgur API

The easiest way to start using the Imgur API is by clicking the **Run in Postman** button above. Postman is a free tool which helps developers run and debug API requests, and is the source of truth for this documentation. Every endpoint you see documented here is readily available by running our Postman collection.

Example code

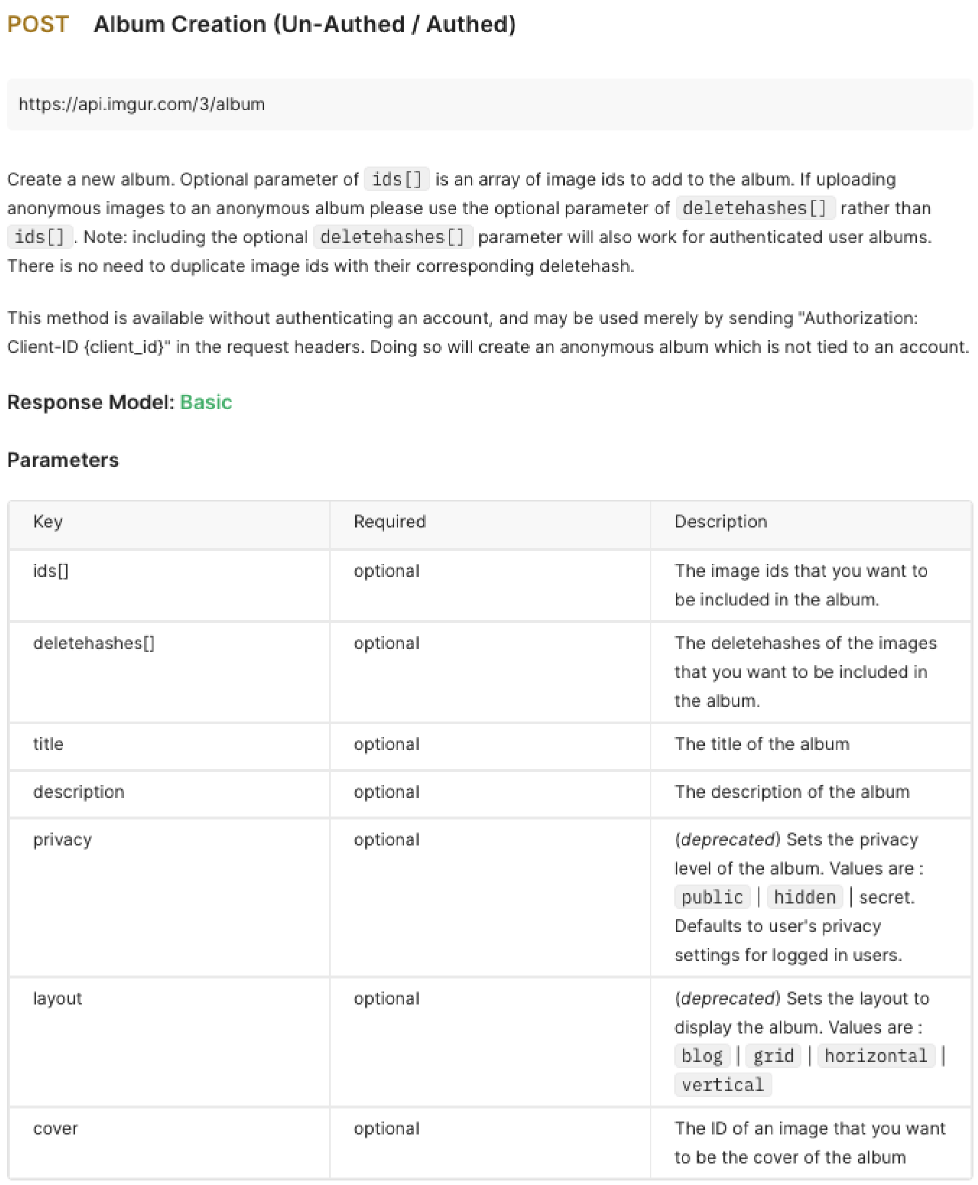
These examples serve as a starting point to help familiarize you with the basics of the Imgur API.

- Official Python library
- Android Upload Example
- Older Example Android app
- Example HTML5/JavaScript app - Javascript OAuth—Live Demo (uses your webcam)
- Example Objective C library

We will start by creating álbum to store images.

Select Album and then Album Creation.

ALBUM CREATION (1/2)



POST **Album Creation (Un-Authed / Authed)**

```
https://api.imgur.com/3/album
```

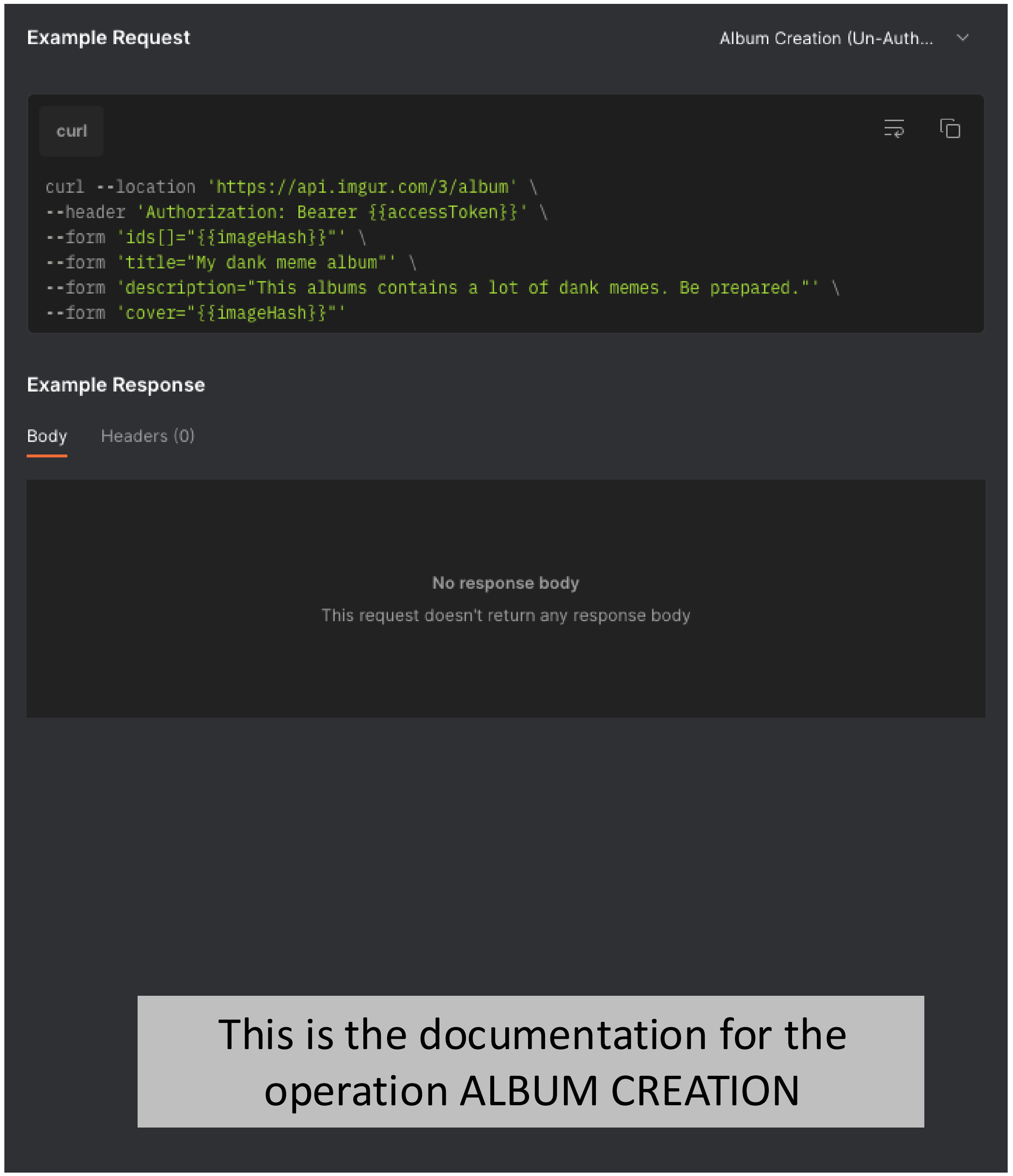
Create a new album. Optional parameter of `ids[]` is an array of image ids to add to the album. If uploading anonymous images to an anonymous album please use the optional parameter of `deletehashes[]` rather than `ids[]`. Note: including the optional `deletehashes[]` parameter will also work for authenticated user albums. There is no need to duplicate image ids with their corresponding deletehash.

This method is available without authenticating an account, and may be used merely by sending "Authorization: Client-ID {client_id}" in the request headers. Doing so will create an anonymous album which is not tied to an account.

Response Model: Basic

Parameters

Key	Required	Description
ids[]	optional	The image ids that you want to be included in the album.
deletehashes[]	optional	The deletehashes of the images that you want to be included in the album.
title	optional	The title of the album
description	optional	The description of the album
privacy	optional	*(deprecated)* Sets the privacy level of the album. Values are : `public` | `hidden` | secret. Defaults to user's privacy settings for logged in users.
layout	optional	*(deprecated)* Sets the layout to display the album. Values are : `blog` | `grid` | `horizontal` | `vertical`
cover	optional	The ID of an image that you want to be the cover of the album



Example Request

Album Creation (Un-Auth...

curl

```
curl --location 'https://api.imgur.com/3/album' \
--header 'Authorization: Bearer {{accessToken}}' \
--form 'ids[]="{{imageHash}}"' \
--form 'title="My dank meme album"' \
--form 'description="This albums contains a lot of dank memes. Be prepared."' \
--form 'cover="{{imageHash}}"'
```

Example Response

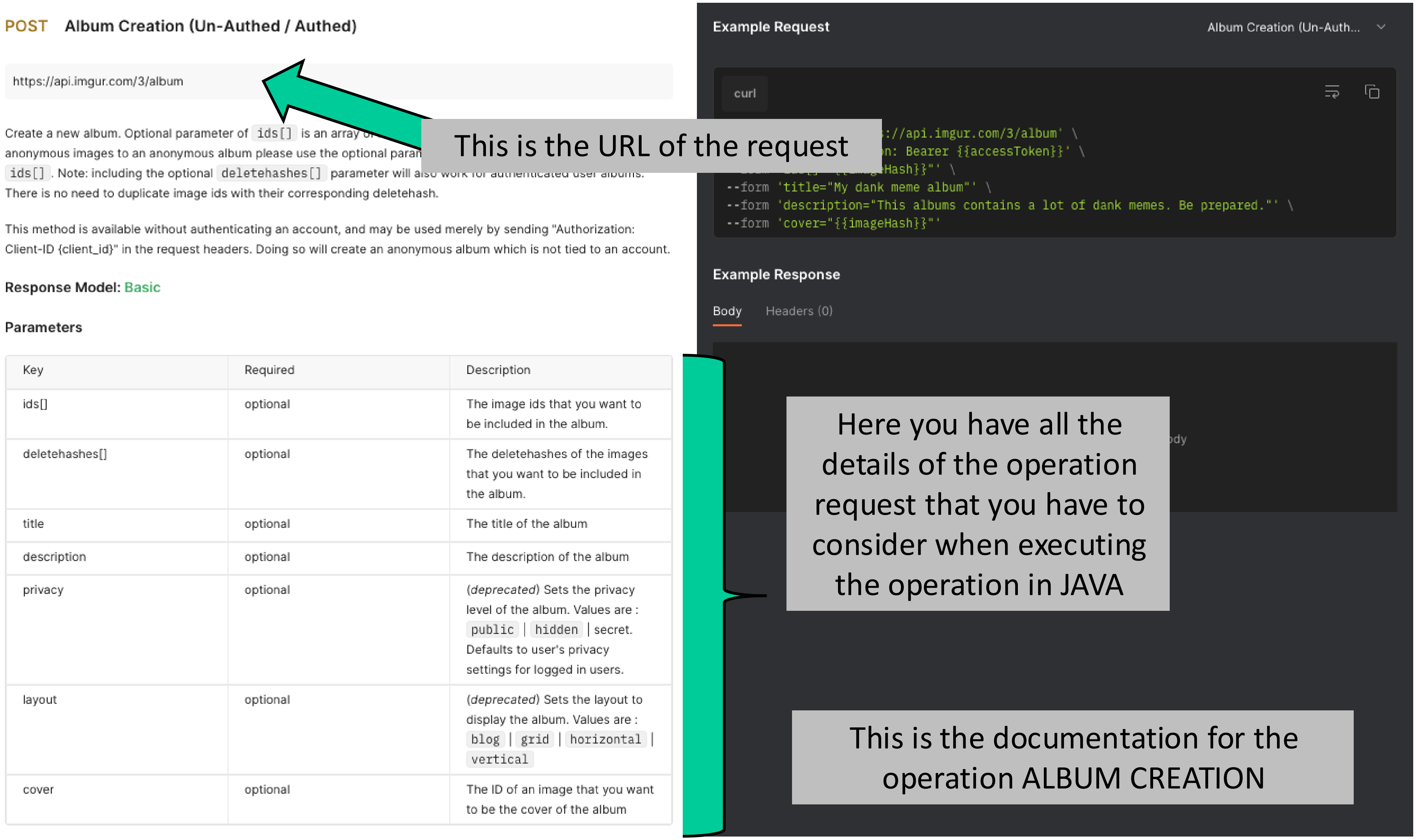
Body   Headers (0)

No response body

This request doesn't return any response body

This is the documentation for the operation ALBUM CREATION

ALBUM CREATION (1/2)



POST Album Creation (Un-Authed / Authed)

https://api.imgur.com/3/album

This is the URL of the request

Create a new album. Optional parameter of `ids[]` is an array o
anonymous images to an anonymous album please use the optional param
`ids[]`. Note: including the optional `deletehashes[]` parameter will al
There is no need to duplicate image ids with their corresponding deletehash.

This method is available without authenticating an account, and may be used merely by sending "Authorization: Client-ID {client_id}" in the request headers. Doing so will create an anonymous album which is not tied to an account.

Response Model: Basic

Parameters

Key	Required	Description
ids[]	optional	The image ids that you want to be included in the album.
deletehashes[]	optional	The deletehashes of the images that you want to be included in the album.
title	optional	The title of the album
description	optional	The description of the album
privacy	optional	*(deprecated)* Sets the privacy level of the album. Values are : `public` | `hidden` | secret. Defaults to user's privacy settings for logged in users.
layout	optional	*(deprecated)* Sets the layout to display the album. Values are : `blog` | `grid` | `horizontal` | `vertical`
cover	optional	The ID of an image that you want to be the cover of the album

Example Request

Album Creation (Un-Auth...

curl

```
                           ://api.imgur.com/3/album' \
                           n: Bearer {{accessToken}}' \
                           Hash}}"' \
--form 'title="My dank meme album"' \
--form 'description="This albums contains a lot of dank memes. Be prepared."' \
--form 'cover="{{imageHash}}"'
```

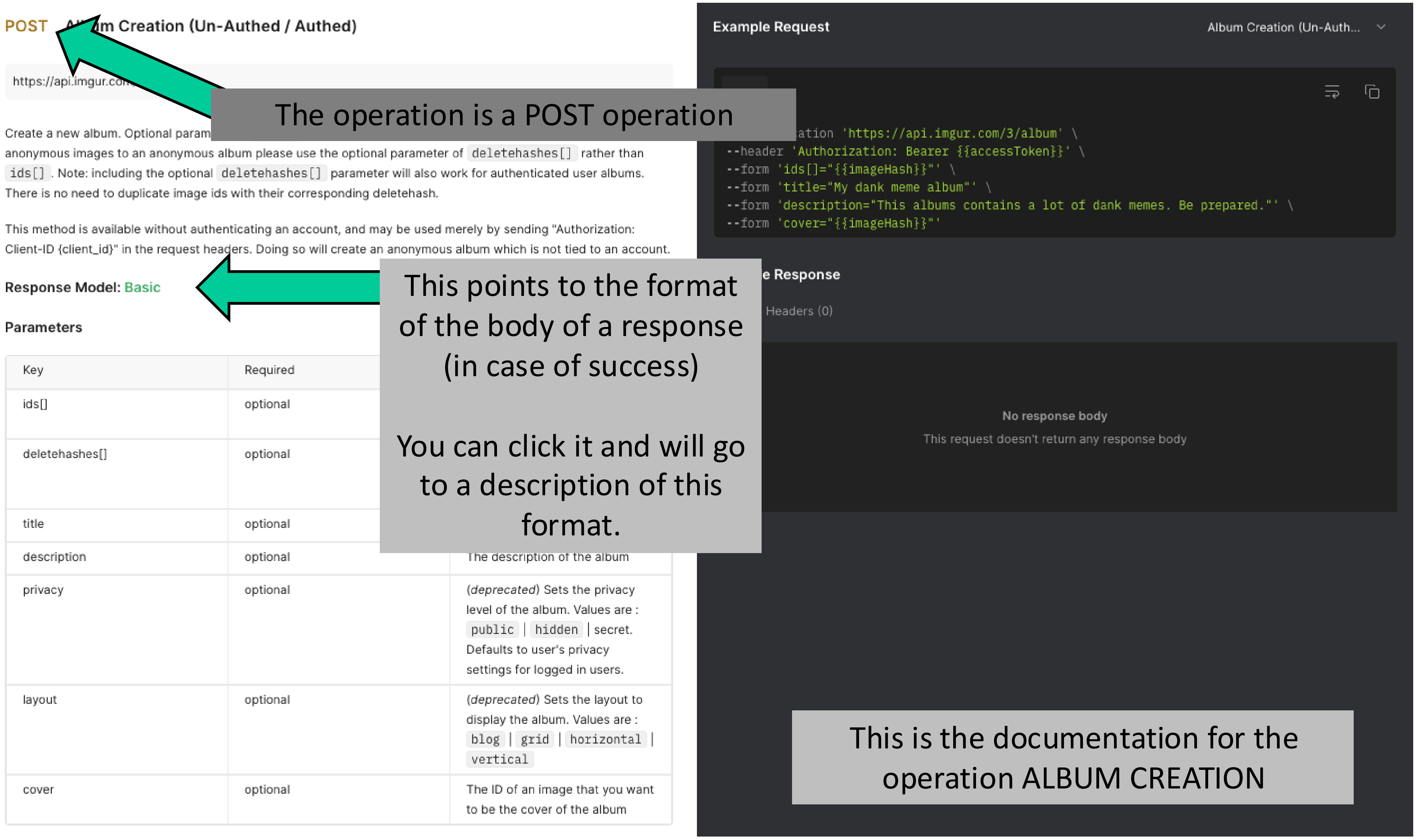
Example Response

Body Headers (0)

Here you have all the details of the operation request that you have to consider when executing the operation in JAVA

This is the documentation for the operation ALBUM CREATION

ALBUM CREATION (1/2)



POST Al[…]m Creation (Un-Authed / Authed)

https://api.imgur.co

Create a new album. Optional param
anonymous images to an anonymous album please use the optional parameter of `deletehashes[]` rather than `ids[]`. Note: including the optional `deletehashes[]` parameter will also work for authenticated user albums. There is no need to duplicate image ids with their corresponding deletehash.

This method is available without authenticating an account, and may be used merely by sending "Authorization: Client-ID {client_id}" in the request headers. Doing so will create an anonymous album which is not tied to an account.

Response Model: Basic

Parameters

Key	Required	
ids[]	optional	
deletehashes[]	optional	
title	optional	
description	optional	The description of the album
privacy	optional	*(deprecated)* Sets the privacy level of the album. Values are : `public` | `hidden` | secret. Defaults to user's privacy settings for logged in users.
layout	optional	*(deprecated)* Sets the layout to display the album. Values are : `blog` | `grid` | `horizontal` | `vertical`
cover	optional	The ID of an image that you want to be the cover of the album

Example Request Album Creation (Un-Auth...

```
ation 'https://api.imgur.com/3/album' \
--header 'Authorization: Bearer {{accessToken}}' \
--form 'ids[]="{{imageHash}}"' \
--form 'title="My dank meme album"' \
--form 'description="This albums contains a lot of dank memes. Be prepared."' \
--form 'cover="{{imageHash}}"'
```

e Response

Headers (0)

No response body
This request doesn't return any response body

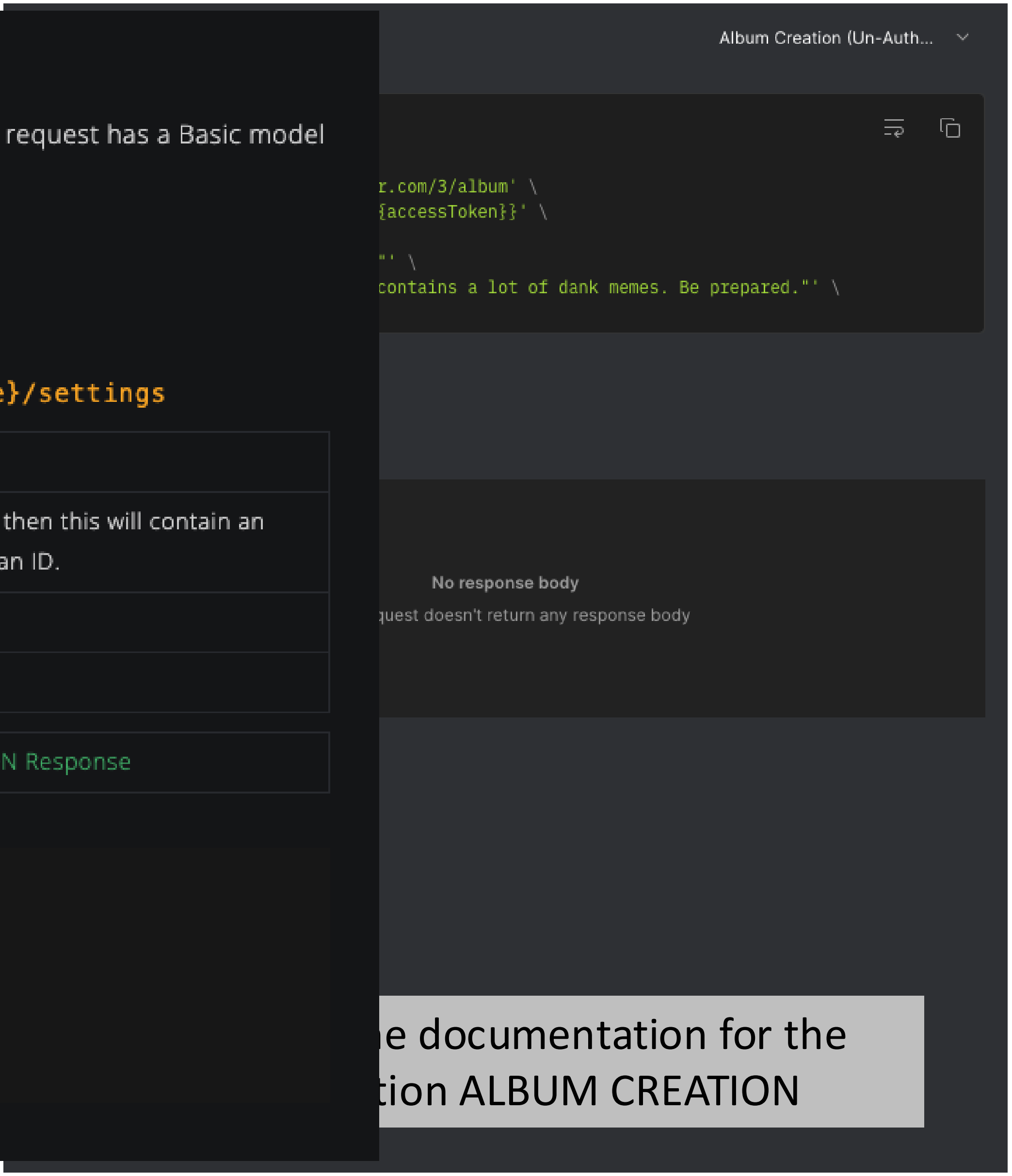
The operation is a POST operation

This points to the format of the body of a response (in case of success)

You can click it and will go to a description of this format.

This is the documentation for the operation ALBUM CREATION

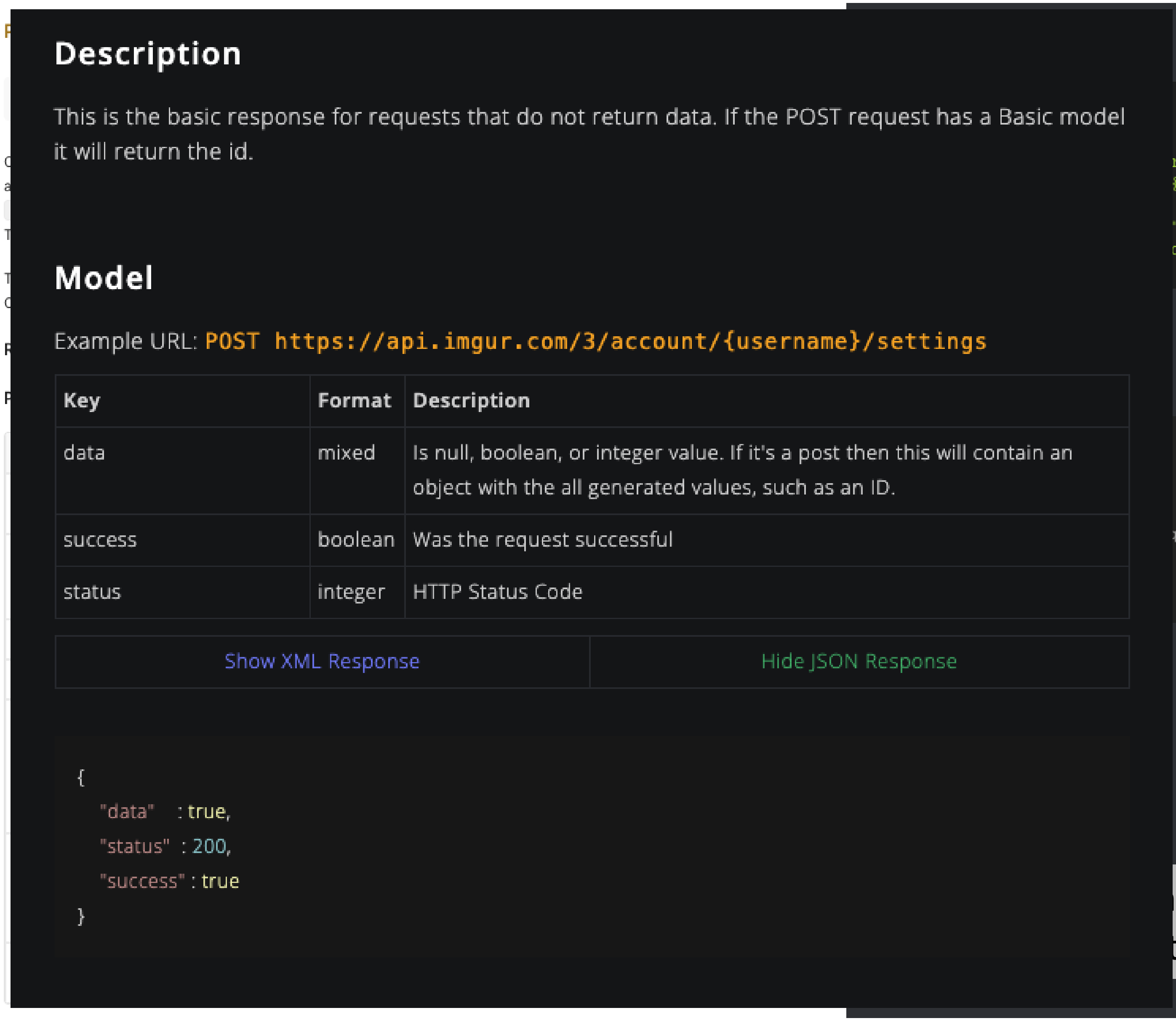
ALBUM CREATION (1/2)



```
r.com/3/album' \
{accessToken}}' \
"' \
contains a lot of dank memes. Be prepared."' \
```

No response body

quest doesn't return any response body



Description

This is the basic response for requests that do not return data. If the POST request has a Basic model it will return the id.

Model

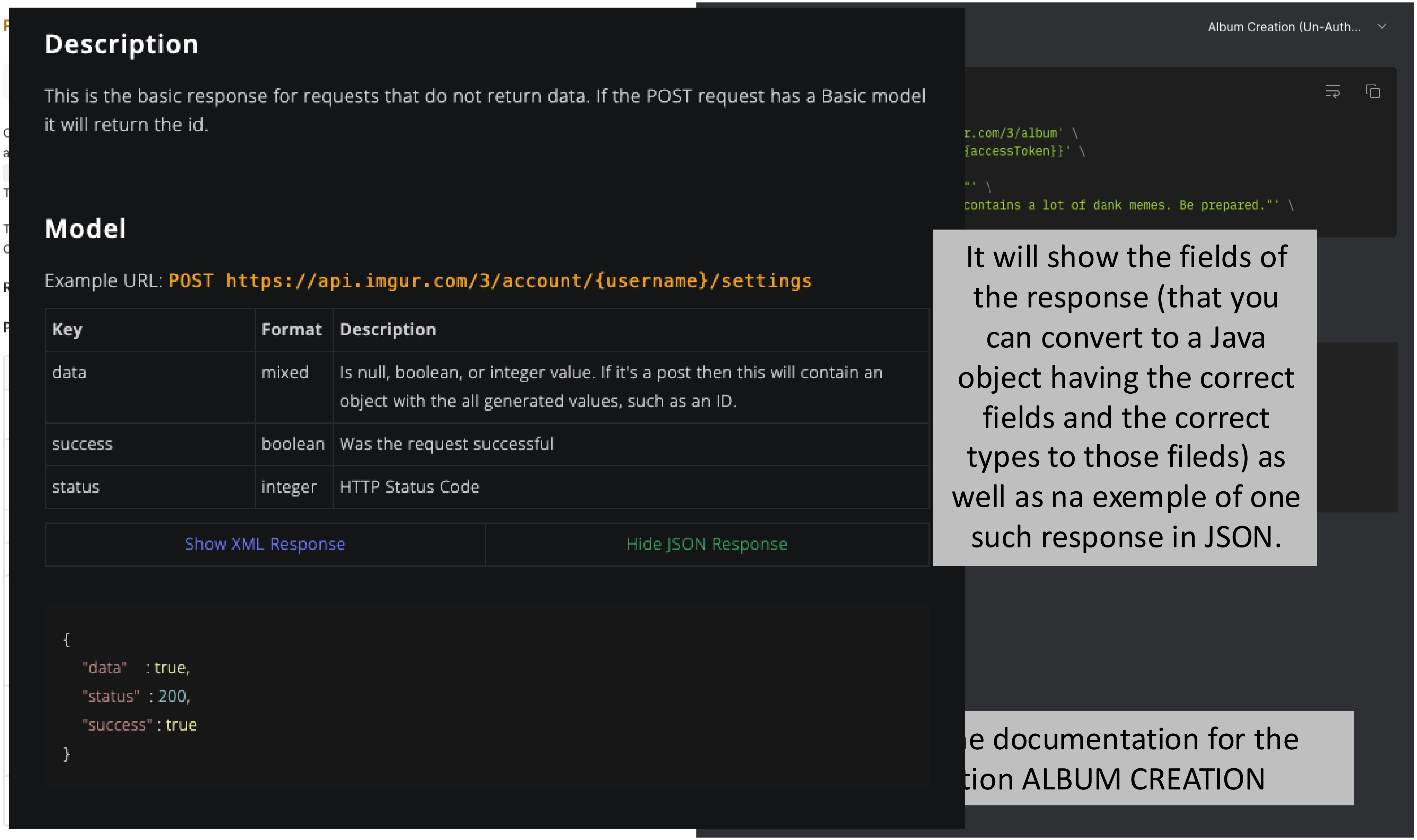
Example URL: `POST https://api.imgur.com/3/account/{username}/settings`

Key	Format	Description
data	mixed	Is null, boolean, or integer value. If it's a post then this will contain an object with the all generated values, such as an ID.
success	boolean	Was the request successful
status	integer	HTTP Status Code

```
{
   "data"    : true,
   "status"  : 200,
   "success" : true
}
```

e documentation for the
tion ALBUM CREATION

ALBUM CREATION (1/2)



Album Creation (Un-Auth...

```
r.com/3/album' \
{accessToken}}' \

"' \
contains a lot of dank memes. Be prepared."' \
```

Description

This is the basic response for requests that do not return data. If the POST request has a Basic model it will return the id.

Model

Example URL: `POST https://api.imgur.com/3/account/{username}/settings`

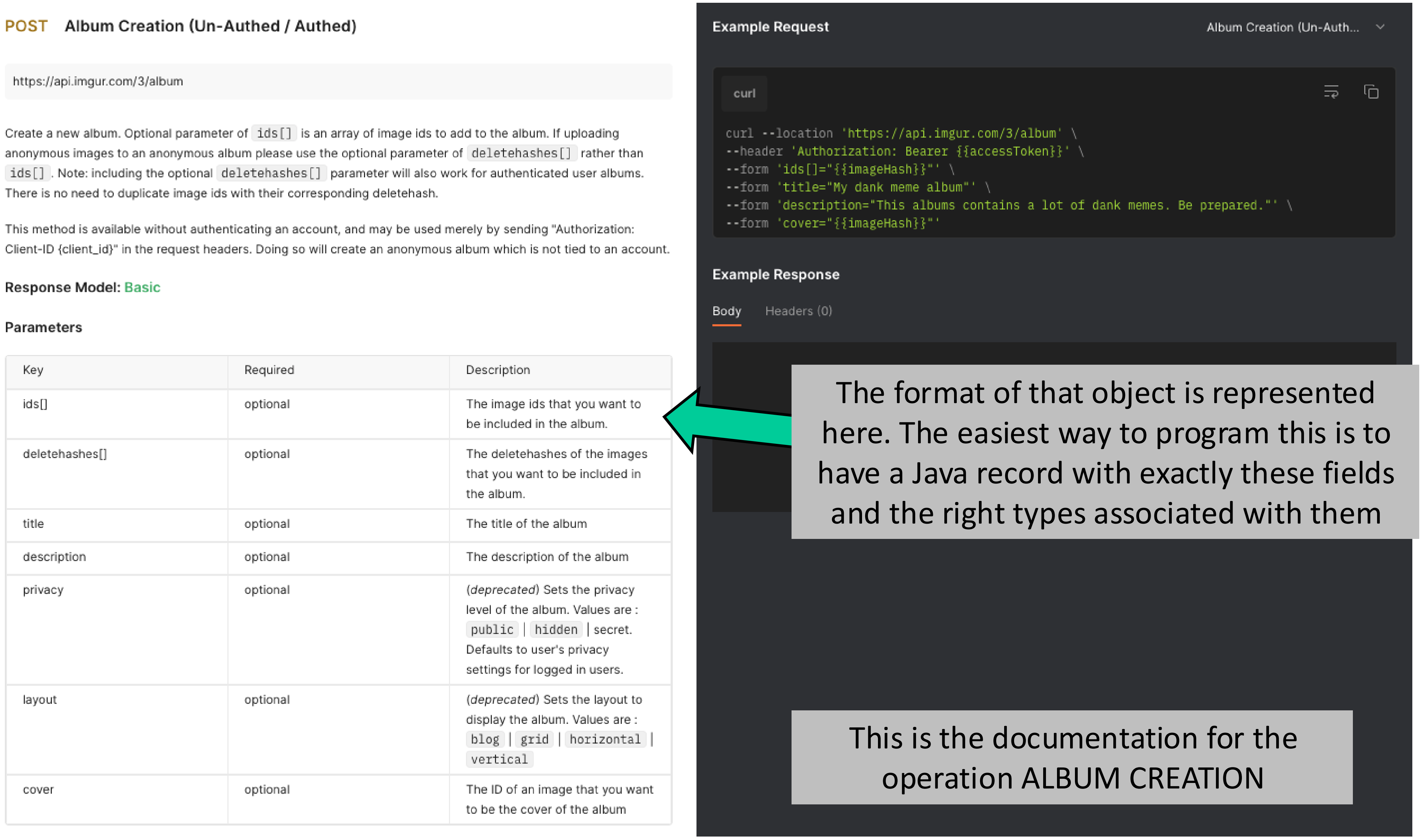
Key	Format	Description
data	mixed	Is null, boolean, or integer value. If it's a post then this will contain an object with the all generated values, such as an ID.
success	boolean	Was the request successful
status	integer	HTTP Status Code

```
{
  "data"    : true,
  "status"  : 200,
  "success" : true
}
```

It will show the fields of the response (that you can convert to a Java object having the correct fields and the correct types to those fileds) as well as na exemple of one such response in JSON.

e documentation for the
tion ALBUM CREATION

Album Creation (1/2)



POST **Album Creation (Un-Authed / Authed)**

https://api.imgur.com/3/album

Create a new album. Optional parameter of `ids[]` is an array of image ids to add to the album. If uploading anonymous images to an anonymous album please use the optional parameter of `deletehashes[]` rather than `ids[]`. Note: including the optional `deletehashes[]` parameter will also work for authenticated user albums. There is no need to duplicate image ids with their corresponding deletehash.

This method is available without authenticating an account, and may be used merely by sending "Authorization: Client-ID {client_id}" in the request headers. Doing so will create an anonymous album which is not tied to an account.

Response Model: Basic

Parameters

Key	Required	Description
ids[]	optional	The image ids that you want to be included in the album.
deletehashes[]	optional	The deletehashes of the images that you want to be included in the album.
title	optional	The title of the album
description	optional	The description of the album
privacy	optional	*(deprecated)* Sets the privacy level of the album. Values are : `public` | `hidden` | secret. Defaults to user's privacy settings for logged in users.
layout	optional	*(deprecated)* Sets the layout to display the album. Values are : `blog` | `grid` | `horizontal` | `vertical`
cover	optional	The ID of an image that you want to be the cover of the album

Example Request

Album Creation (Un-Auth...

curl

```
curl --location 'https://api.imgur.com/3/album' \
--header 'Authorization: Bearer {{accessToken}}' \
--form 'ids[]="{{imageHash}}"' \
--form 'title="My dank meme album"' \
--form 'description="This albums contains a lot of dank memes. Be prepared."' \
--form 'cover="{{imageHash}}"'
```

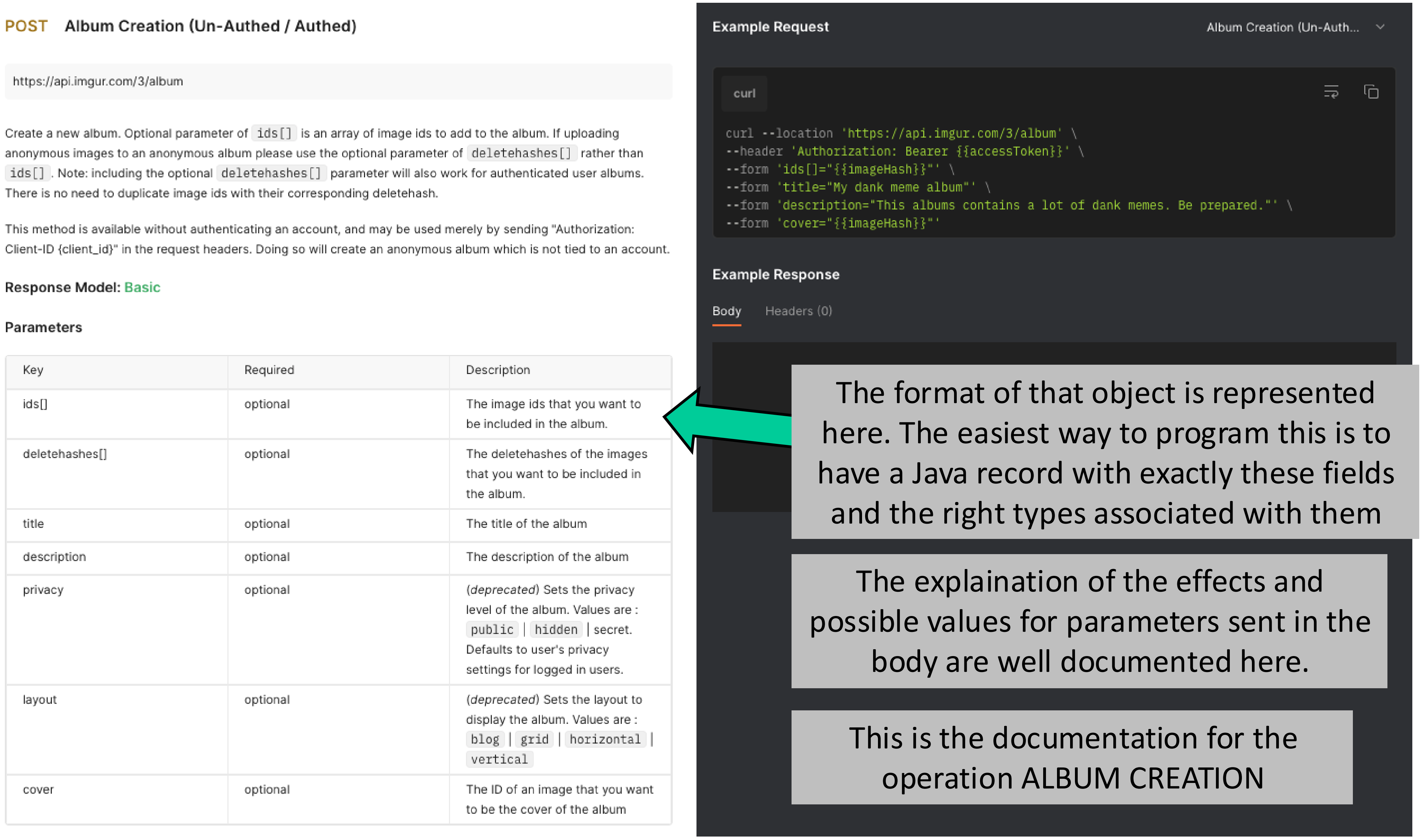
Example Response

Body   Headers (0)

The format of that object is represented here. The easiest way to program this is to have a Java record with exactly these fields and the right types associated with them

This is the documentation for the operation ALBUM CREATION

ALBUM CREATION (1/2)



POST **Album Creation (Un-Authed / Authed)**

https://api.imgur.com/3/album

Create a new album. Optional parameter of `ids[]` is an array of image ids to add to the album. If uploading anonymous images to an anonymous album please use the optional parameter of `deletehashes[]` rather than `ids[]`. Note: including the optional `deletehashes[]` parameter will also work for authenticated user albums. There is no need to duplicate image ids with their corresponding deletehash.

This method is available without authenticating an account, and may be used merely by sending "Authorization: Client-ID {client_id}" in the request headers. Doing so will create an anonymous album which is not tied to an account.

Response Model: Basic

Parameters

Key	Required	Description
ids[]	optional	The image ids that you want to be included in the album.
deletehashes[]	optional	The deletehashes of the images that you want to be included in the album.
title	optional	The title of the album
description	optional	The description of the album
privacy	optional	*(deprecated)* Sets the privacy level of the album. Values are : `public` | `hidden` | secret. Defaults to user's privacy settings for logged in users.
layout	optional	*(deprecated)* Sets the layout to display the album. Values are : `blog` | `grid` | `horizontal` | `vertical`
cover	optional	The ID of an image that you want to be the cover of the album

Example Request Album Creation (Un-Auth...

curl

```
curl --location 'https://api.imgur.com/3/album' \
--header 'Authorization: Bearer {{accessToken}}' \
--form 'ids[]="{{imageHash}}"' \
--form 'title="My dank meme album"' \
--form 'description="This albums contains a lot of dank memes. Be prepared."' \
--form 'cover="{{imageHash}}"'
```

Example Response

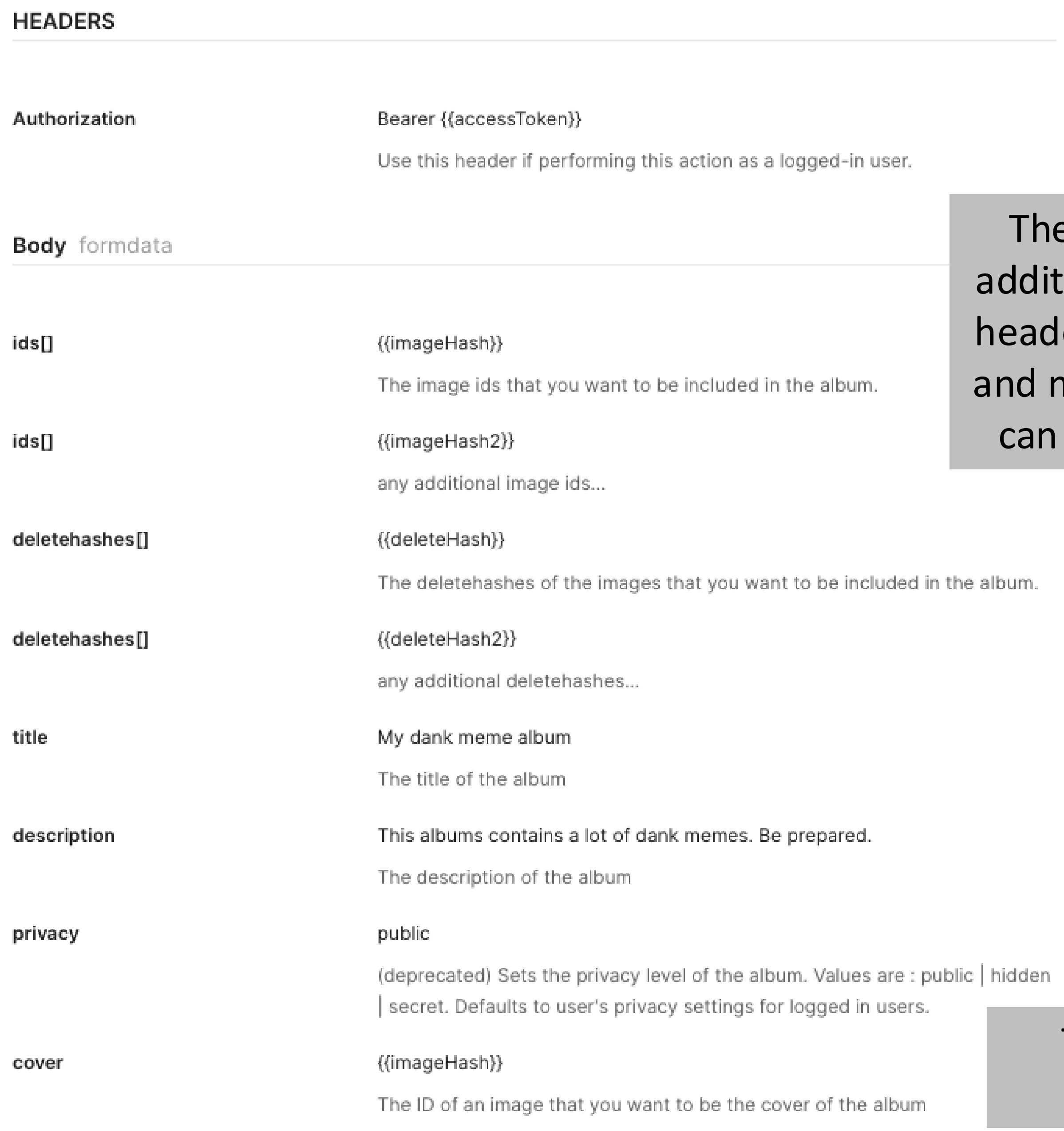
Body Headers (0)

The format of that object is represented here. The easiest way to program this is to have a Java record with exactly these fields and the right types associated with them

The explaination of the effects and possible values for parameters sent in the body are well documented here.

This is the documentation for the operation ALBUM CREATION

ALBUM CREATION(2/2)



HEADERS

Authorization	Bearer {{accessToken}} Use this header if performing this action as a logged-in user.

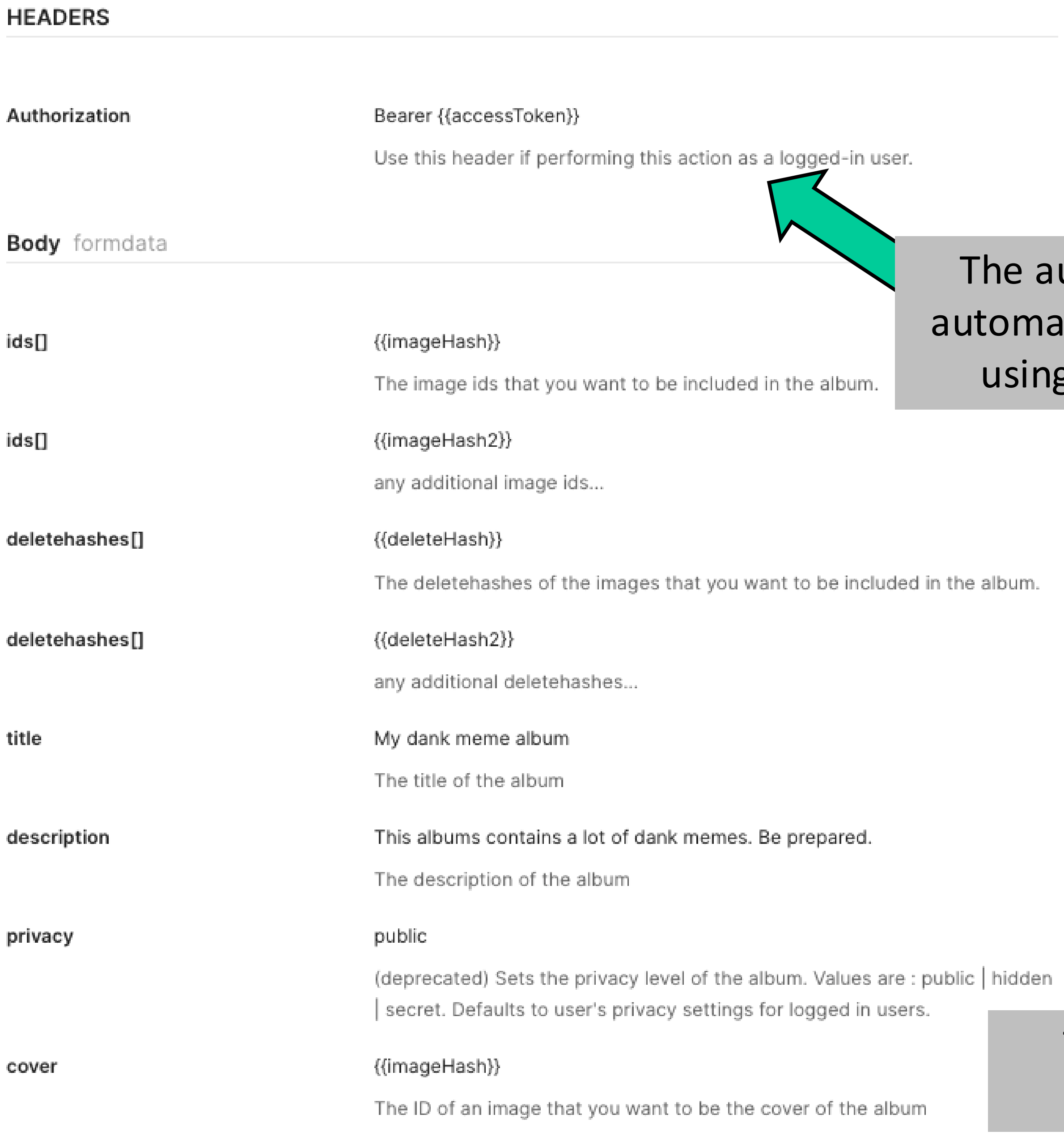
Body formdata

ids[]	{{imageHash}} The image ids that you want to be included in the album.
ids[]	{{imageHash2}} any additional image ids...
deletehashes[]	{{deleteHash}} The deletehashes of the images that you want to be included in the album.
deletehashes[]	{{deleteHash2}} any additional deletehashes...
title	My dank meme album The title of the album
description	This albums contains a lot of dank memes. Be prepared. The description of the album
privacy	public (deprecated) Sets the privacy level of the album. Values are : public | hidden | secret. Defaults to user's privacy settings for logged in users.
cover	{{imageHash}} The ID of an image that you want to be the cover of the album

The rest of the documentation covers additional aspects of the request, namely headers that can be added to the request and more details about the elements that can be sent in the body of the request.

This is the documentation for the operation ALBUM CREATION

ALBUM CREATION(2/2)



HEADERS

Authorization	Bearer {{accessToken}} Use this header if performing this action as a logged-in user.

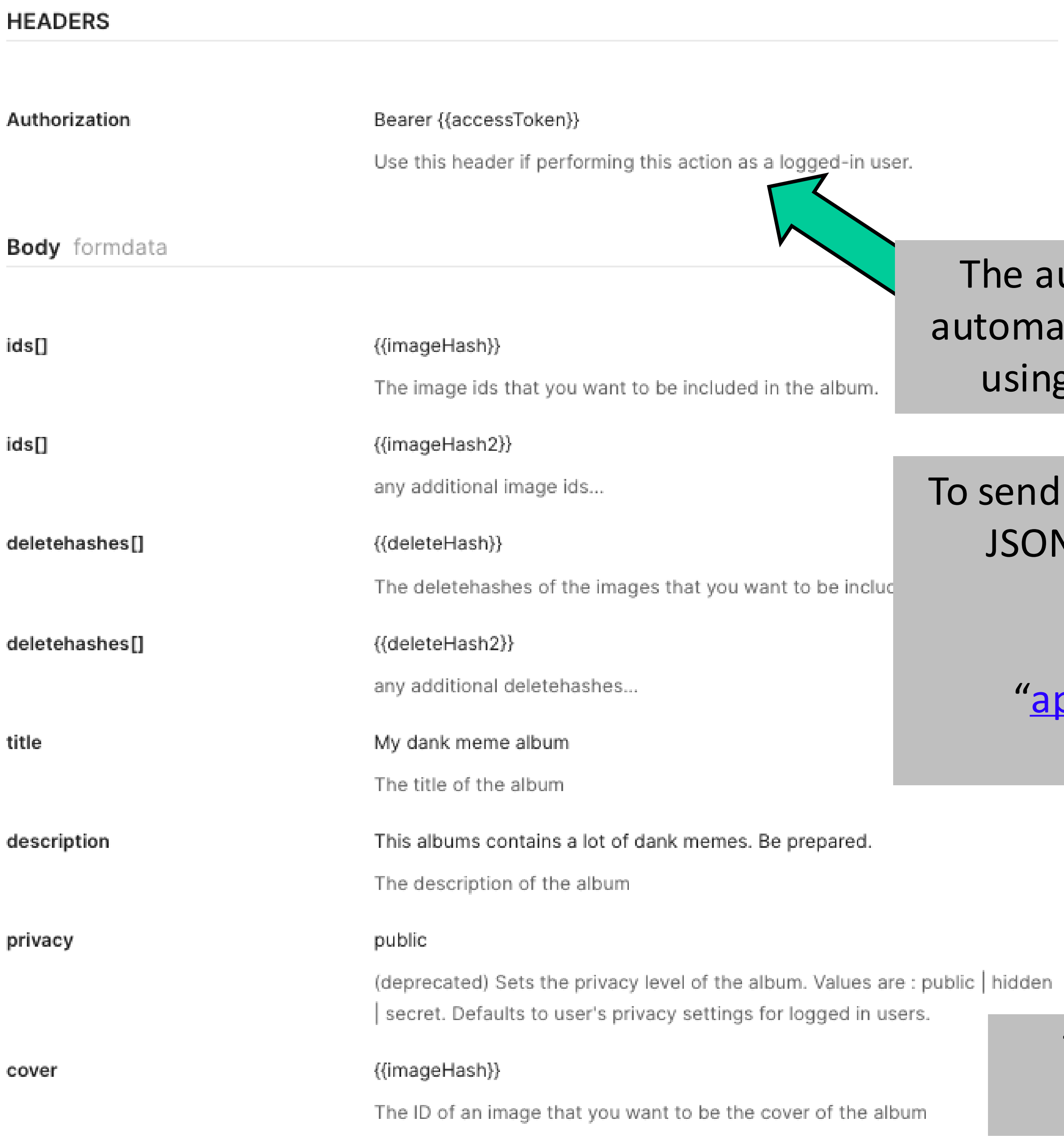
Body formdata

ids[]	{{imageHash}} The image ids that you want to be included in the album.
ids[]	{{imageHash2}} any additional image ids...
deletehashes[]	{{deleteHash}} The deletehashes of the images that you want to be included in the album.
deletehashes[]	{{deleteHash2}} any additional deletehashes...
title	My dank meme album The title of the album
description	This albums contains a lot of dank memes. Be prepared. The description of the album
privacy	public (deprecated) Sets the privacy level of the album. Values are : public | hidden | secret. Defaults to user's privacy settings for logged in users.
cover	{{imageHash}} The ID of an image that you want to be the cover of the album

The authorization header will be added automatically by the library that we will be using to make these requests in Java.

This is the documentation for the operation ALBUM CREATION

ALBUM CREATION(2/2)



HEADERS

Authorization	Bearer {{accessToken}} Use this header if performing this action as a logged-in user.

Body formdata

ids[]	{{imageHash}} The image ids that you want to be included in the album.
ids[]	{{imageHash2}} any additional image ids...
deletehashes[]	{{deleteHash}} The deletehashes of the images that you want to be incluc
deletehashes[]	{{deleteHash2}} any additional deletehashes...
title	My dank meme album The title of the album
description	This albums contains a lot of dank memes. Be prepared. The description of the album
privacy	public (deprecated) Sets the privacy level of the album. Values are : public | hidden | secret. Defaults to user's privacy settings for logged in users.
cover	{{imageHash}} The ID of an image that you want to be the cover of the album

The authorization header will be added automatically by the library that we will be using to make these requests in Java.

To send the body of requests encoded as a JSON object you will nedd to add an additional header named "Content-Type" with value "application/json; charset=utf-8"

This is the documentation for the operation ALBUM CREATION

GOALS

In the end of this lab you should be able to:

- Understand what is Oauth
- How to register an application with Imgur
- How to generate the credentials needed for Oauth in Imgur
- How to take advantage of the REST API documentation of Imgur
- **Know how to make requests to Imgur using Oauth using the library ScribeJava**

OAUTH REQUESTS TO IMGUR IN JAVA

To execute Oauth requests we are going to use an additional Java library called ScribeJava

- The examples shown here are very simple client applications that have an over-simplified user interface in the command line and execute requests (and process replies) to Imgur
- In the project a variant of your server will be responsible to execute operations over Imgur (and those servers will act as clients to Imgur) without any command line user interface.
- As usual, all code shown here is available in an Eclipse project in the course webpage

OAUTH REQUESTS TO IMGUR IN JAVA

Since we are using additional libraries you will need to add new dependencies to your pom.xml file. The pom.xml file in the example project already include these:

```
<dependency>
    <groupId>com.google.code.gson</groupId>
    <artifactId>gson</artifactId>
    <version>2.13.1</version>
</dependency>
<dependency>
    <groupId>com.github.scribejava</groupId>
    <artifactId>scribejava-apis</artifactId>
    <version>8.3.3</version>
</dependency>
<dependency>
    <groupId>org.pac4j</groupId>
    <artifactId>pac4j-oauth</artifactId>
    <version>6.1.2</version>
</dependency>
```

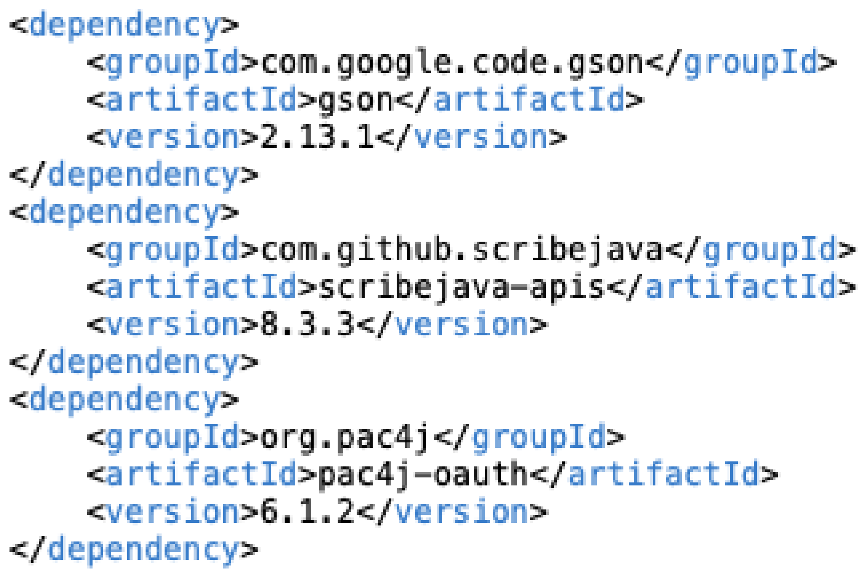
We will use Gson to manipuate json objects

The scribejava main APIs are provided by this

pom.xml (Dependencies Section)

OAUTH REQUESTS TO IMGUR IN JAVA

Since we are using additional libraries you will need to add new dependencies to your pom.xml file. The pom.xml file in the example project already include these:



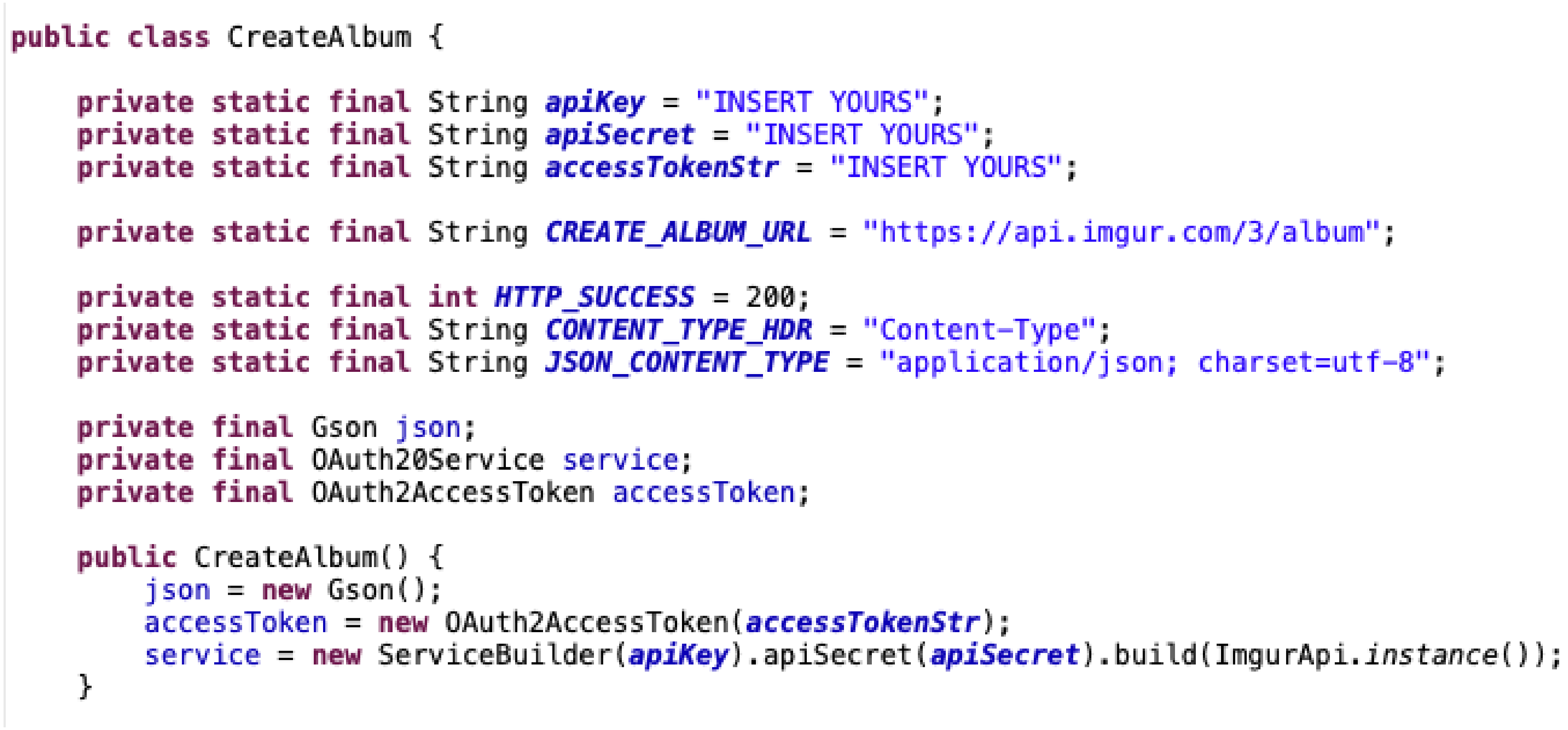
```xml
<dependency>
    <groupId>com.google.code.gson</groupId>
    <artifactId>gson</artifactId>
    <version>2.13.1</version>
</dependency>
<dependency>
    <groupId>com.github.scribejava</groupId>
    <artifactId>scribejava-apis</artifactId>
    <version>8.3.3</version>
</dependency>
<dependency>
    <groupId>org.pac4j</groupId>
    <artifactId>pac4j-oauth</artifactId>
    <version>6.1.2</version>
</dependency>
```

pom.xml (Dependencies Section)

We are going to use the latest version of each of these libraries

OAuth Requests to Imgur in JAVA
Create Album (1/3)

Class CreateAlbum (preamble and constructor)

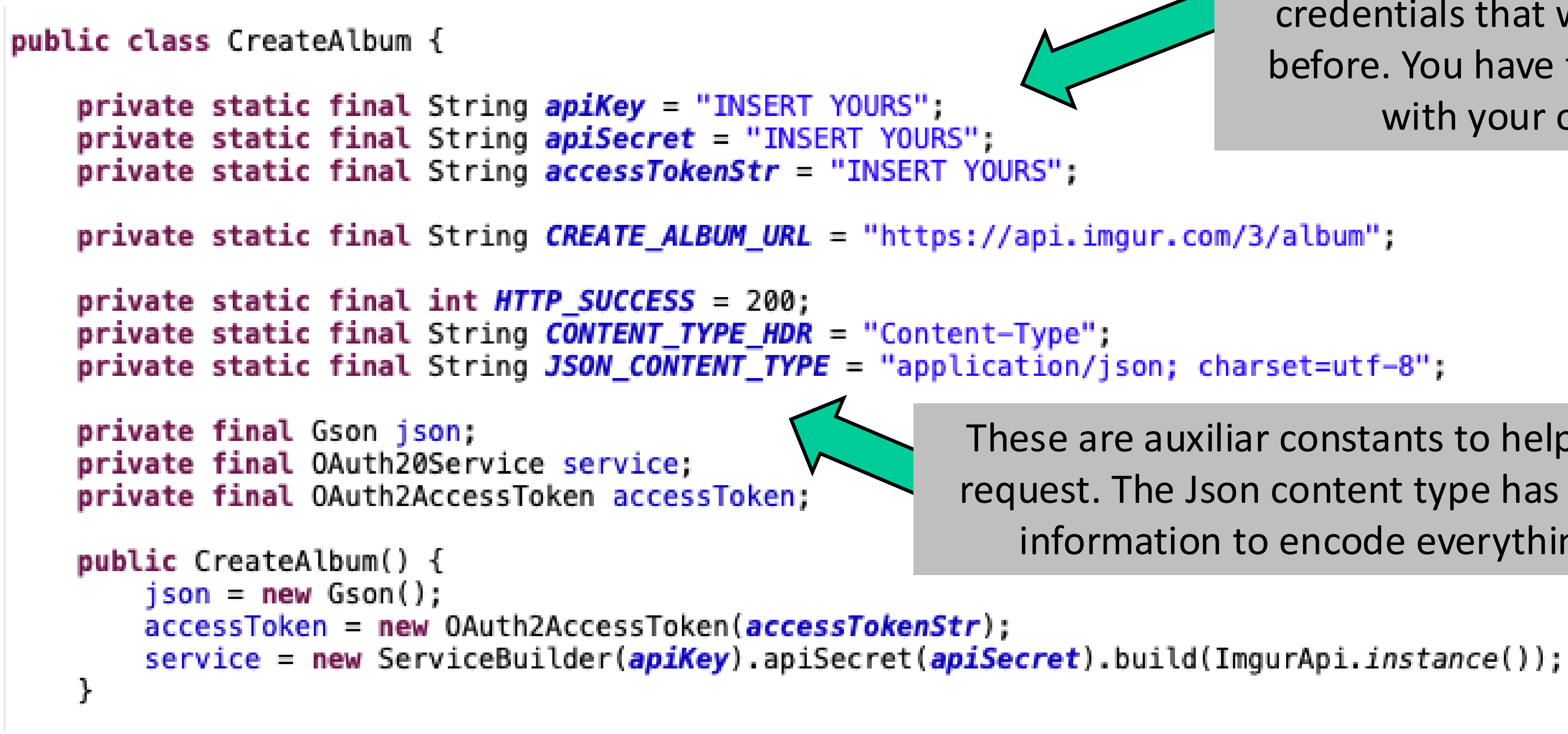


```
public class CreateAlbum {

    private static final String apiKey = "INSERT YOURS";
    private static final String apiSecret = "INSERT YOURS";
    private static final String accessTokenStr = "INSERT YOURS";

    private static final String CREATE_ALBUM_URL = "https://api.imgur.com/3/album";

    private static final int HTTP_SUCCESS = 200;
    private static final String CONTENT_TYPE_HDR = "Content-Type";
    private static final String JSON_CONTENT_TYPE = "application/json; charset=utf-8";

    private final Gson json;
    private final OAuth20Service service;
    private final OAuth2AccessToken accessToken;

    public CreateAlbum() {
        json = new Gson();
        accessToken = new OAuth2AccessToken(accessTokenStr);
        service = new ServiceBuilder(apiKey).apiSecret(apiSecret).build(ImgurApi.instance());
    }
```

OAUTH REQUESTS TO IMGUR IN JAVA
CREATE ALBUM (1/3)

Class CreateAlbum (preamble and constructor)

These are the autentication credentials that we created before. You have to fill them with your own.

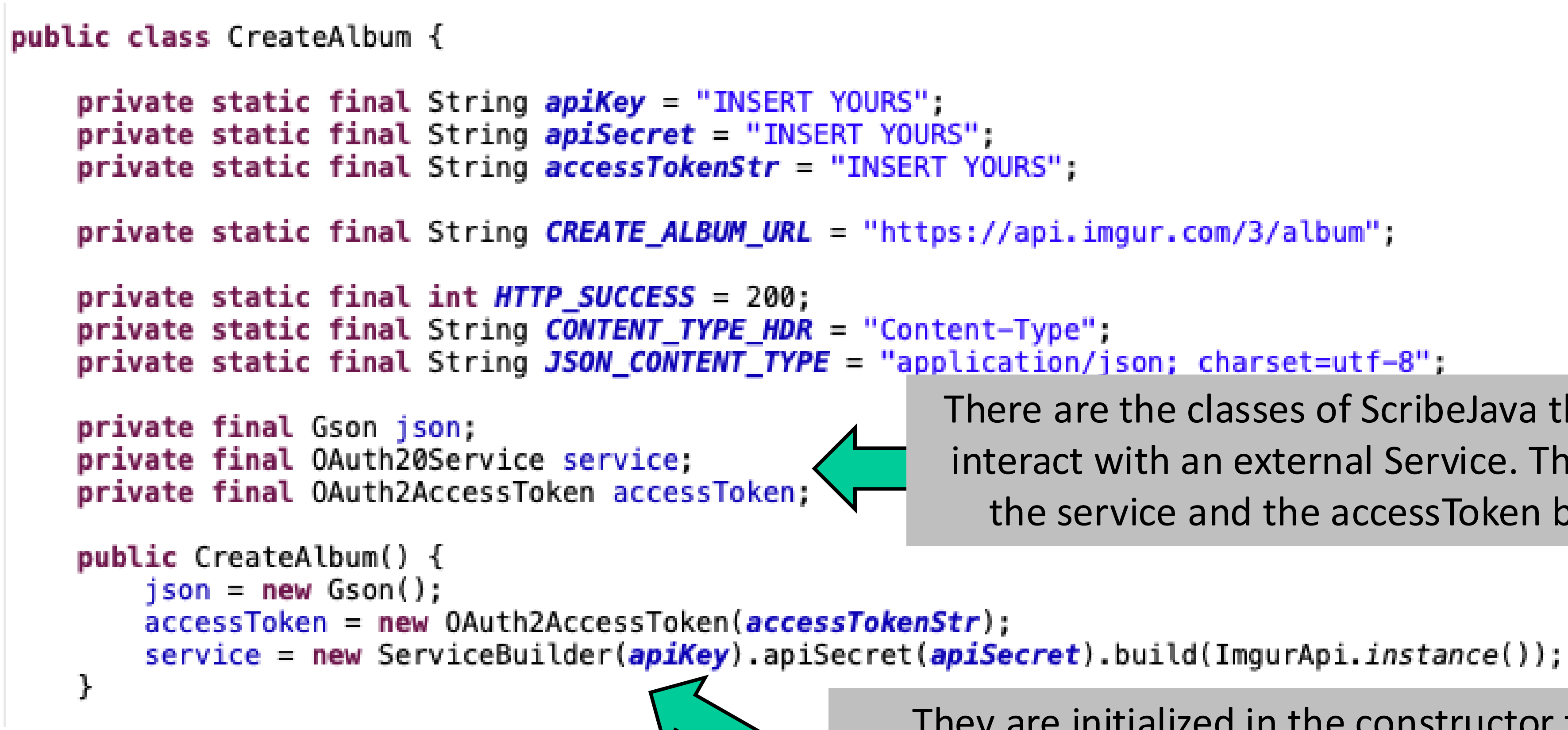


```
public class CreateAlbum {

    private static final String apiKey = "INSERT YOURS";
    private static final String apiSecret = "INSERT YOURS";
    private static final String accessTokenStr = "INSERT YOURS";

    private static final String CREATE_ALBUM_URL = "https://api.imgur.com/3/album";

    private static final int HTTP_SUCCESS = 200;
    private static final String CONTENT_TYPE_HDR = "Content-Type";
    private static final String JSON_CONTENT_TYPE = "application/json; charset=utf-8";

    private final Gson json;
    private final OAuth20Service service;
    private final OAuth2AccessToken accessToken;

    public CreateAlbum() {
        json = new Gson();
        accessToken = new OAuth2AccessToken(accessTokenStr);
        service = new ServiceBuilder(apiKey).apiSecret(apiSecret).build(ImgurApi.instance());
    }
```

These are auxiliar constants to help us build the request. The Json content type has an additional information to encode everything in utf-8

OAuth Requests to Imgur in JAVA
Create Album (1/3)

Class CreateAlbum (preamble and constructor)



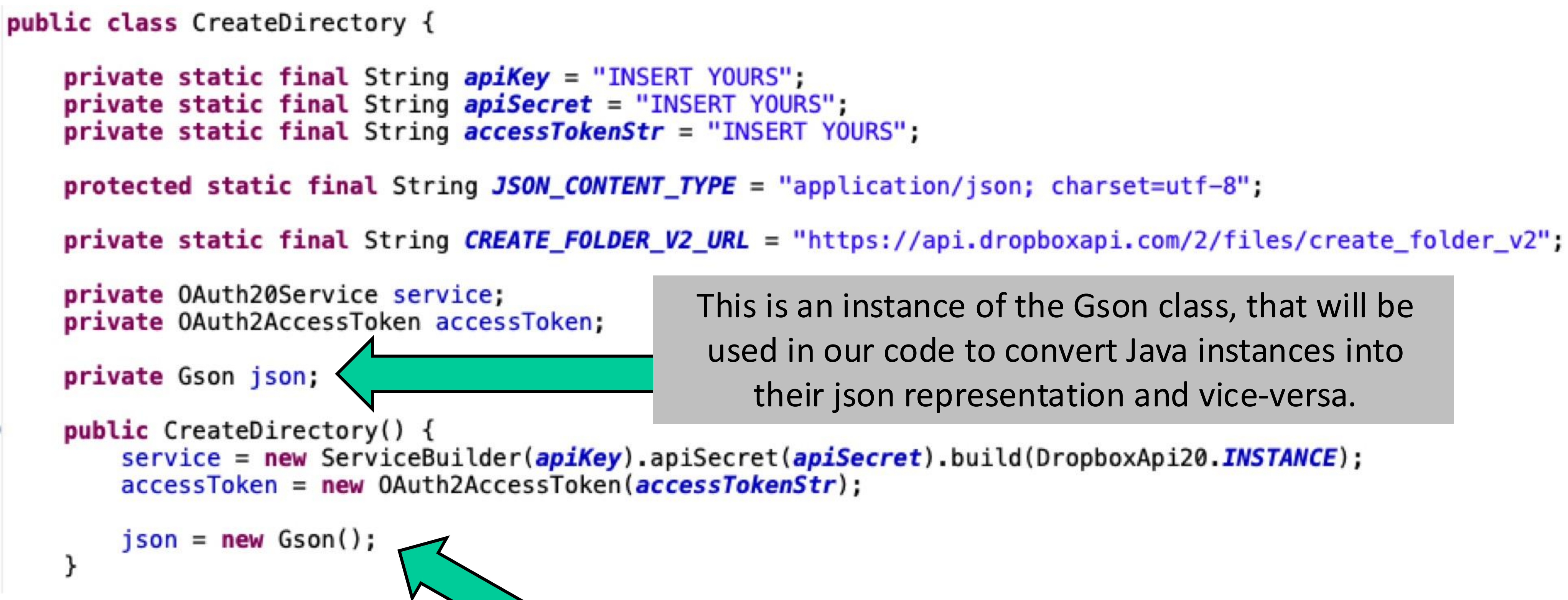
```
public class CreateAlbum {

    private static final String apiKey = "INSERT YOURS";
    private static final String apiSecret = "INSERT YOURS";
    private static final String accessTokenStr = "INSERT YOURS";

    private static final String CREATE_ALBUM_URL = "https://api.imgur.com/3/album";

    private static final int HTTP_SUCCESS = 200;
    private static final String CONTENT_TYPE_HDR = "Content-Type";
    private static final String JSON_CONTENT_TYPE = "application/json; charset=utf-8";

    private final Gson json;
    private final OAuth20Service service;
    private final OAuth2AccessToken accessToken;

    public CreateAlbum() {
        json = new Gson();
        accessToken = new OAuth2AccessToken(accessTokenStr);
        service = new ServiceBuilder(apiKey).apiSecret(apiSecret).build(ImgurApi.instance());
    }
```

There are the classes of ScribeJava that we use to interact with an external Service. They represent the service and the accessToken being used.

They are initialized in the constructor taking advantage of the constants defined before. We configure the service instance to interact with Imgur by using the appropriate wrapper class.

OAUTH REQUESTS TO IMGUR IN JAVA
CREATE ALBUM (1/3)

Class CreateAlbum (preamble and constructor)



```
public class CreateDirectory {

    private static final String apiKey = "INSERT YOURS";
    private static final String apiSecret = "INSERT YOURS";
    private static final String accessTokenStr = "INSERT YOURS";

    protected static final String JSON_CONTENT_TYPE = "application/json; charset=utf-8";

    private static final String CREATE_FOLDER_V2_URL = "https://api.dropboxapi.com/2/files/create_folder_v2";

    private OAuth20Service service;
    private OAuth2AccessToken accessToken;

    private Gson json;

    public CreateDirectory() {
        service = new ServiceBuilder(apiKey).apiSecret(apiSecret).build(DropboxApi20.INSTANCE);
        accessToken = new OAuth2AccessToken(accessTokenStr);

        json = new Gson();
    }
```

This is an instance of the Gson class, that will be used in our code to convert Java instances into their json representation and vice-versa.

We initialize this auxiliary class in the constructor as well.

OAUTH REQUESTS TO IMGUR IN JAVA
CREATE ALBUM (2/3)

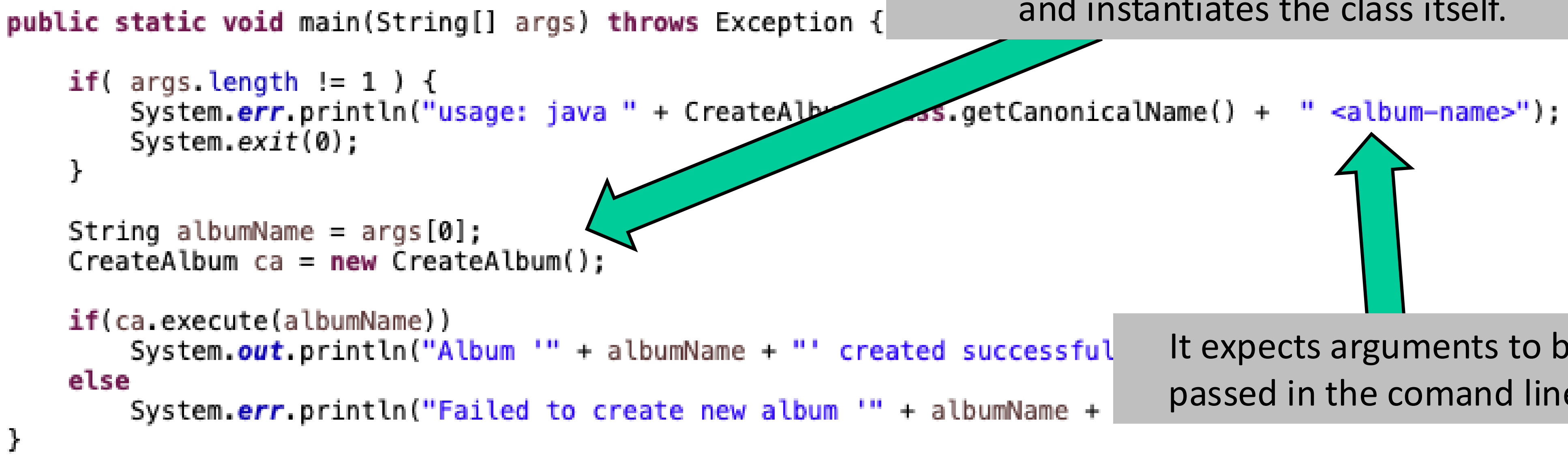
Class CreateAlbum (main -- basic user interface)

```
public static void main(String[] args) throws Exception {

    if( args.length != 1 ) {
        System.err.println("usage: java " + CreateAlbum.class.getCanonicalName() +  " <album-name>");
        System.exit(0);
    }

    String albumName = args[0];
    CreateAlbum ca = new CreateAlbum();

    if(ca.execute(albumName))
        System.out.println("Album '" + albumName + "' created successfuly.");
    else
        System.err.println("Failed to create new album '" + albumName + "'");
}
```

OAUTH REQUESTS TO IMGUR IN JAVA
CREATE ALBUM (2/3)

Class CreateAlbum (main -- basic user interface)

The main of the class interacts with the user and instantiates the class itself.



```
public static void main(String[] args) throws Exception {

    if( args.length != 1 ) {
        System.err.println("usage: java " + CreateAlb         s.getCanonicalName() +  " <album-name>");
        System.exit(0);
    }

    String albumName = args[0];
    CreateAlbum ca = new CreateAlbum();

    if(ca.execute(albumName))
        System.out.println("Album '" + albumName + "' created successful
    else
        System.err.println("Failed to create new album '" + albumName +
}
```

It expects arguments to be passed in the comand line.

OAUTH REQUESTS TO IMGUR IN JAVA
CREATE ALBUM (2/3)

Class CreateAlbum (main -- basic user interface)

```java
public static void main(String[] args) throws Exception {

    if( args.length != 1 ) {
        System.err.println("usage: java " + CreateAlbum.class.getCanonicalName() +  " <album-name>");
        System.exit(0);
    }

    String albumName = args[0];
    CreateAlbum ca = new CreateAlbum();

    if(ca.execute(albumName))
        System.out.println("Album '" + albumName + "' created successfuly.");
    else
        System.err.println("Failed to create new album '" + albumName + "'");
}
```

And uses the provided album name to execute the Oauth operation.

And reports the success or failure of the operation.

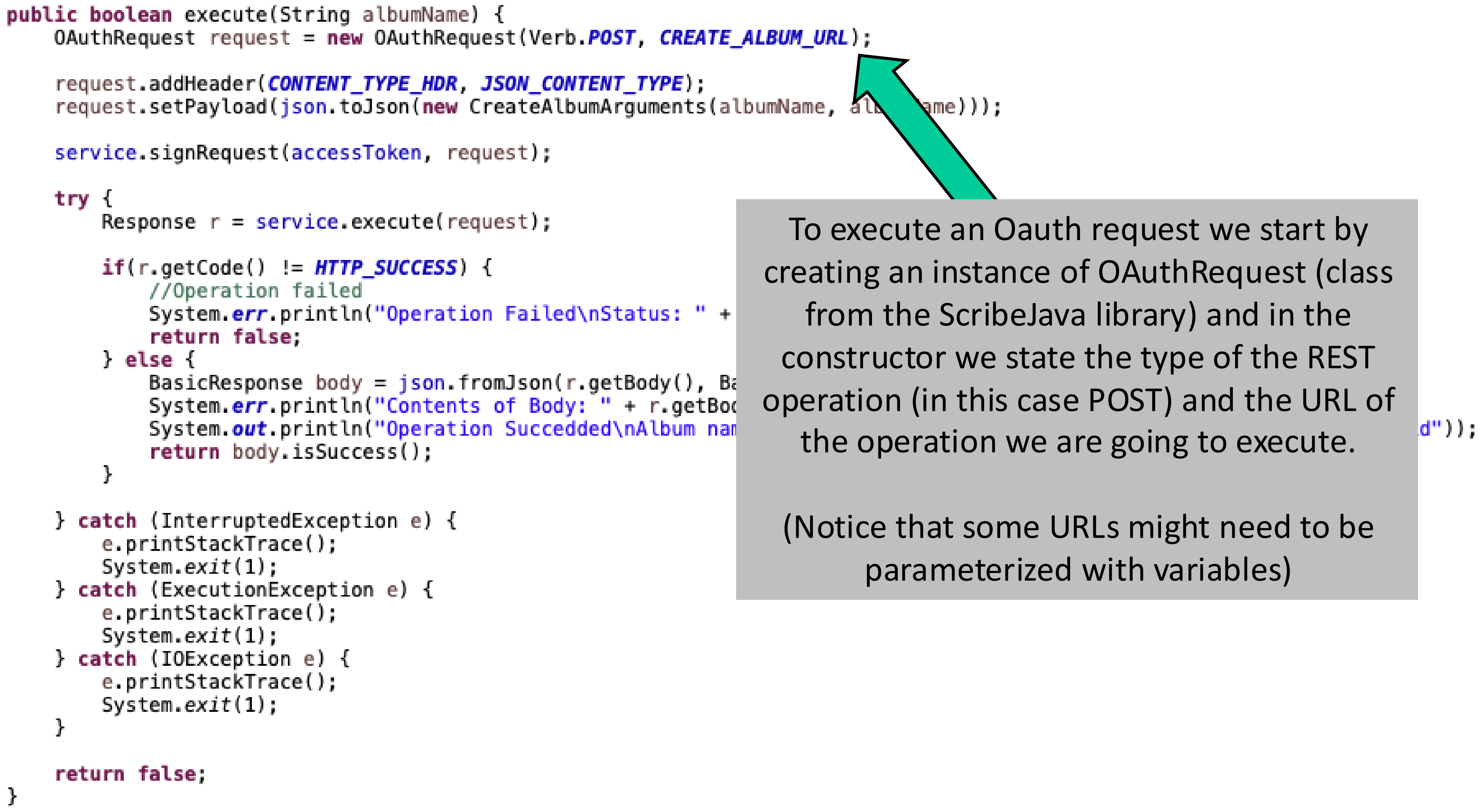
OAUTH REQUESTS TO IMGUR IN JAVA
CREATE ALBUM (3/3)

Class CreateAlbum (oauth request execution)

```java
public boolean execute(String albumName) {
    OAuthRequest request = new OAuthRequest(Verb.POST, CREATE_ALBUM_URL);

    request.addHeader(CONTENT_TYPE_HDR, JSON_CONTENT_TYPE);
    request.setPayload(json.toJson(new CreateAlbumArguments(albumName, albumName)));

    service.signRequest(accessToken, request);

    try {
        Response r = service.execute(request);

        if(r.getCode() != HTTP_SUCCESS) {
            //Operation failed
            System.err.println("Operation Failed\nStatus: " + r.getCode() + "\nBody: " + r.getBody());
            return false;
        } else {
            BasicResponse body = json.fromJson(r.getBody(), BasicResponse.class);
            System.err.println("Contents of Body: " + r.getBody());
            System.out.println("Operation Succedded\nAlbum name: " + albumName + "\nAlbum ID: " + body.getData().get("id"));
            return body.isSuccess();
        }

    } catch (InterruptedException e) {
        e.printStackTrace();
        System.exit(1);
    } catch (ExecutionException e) {
        e.printStackTrace();
        System.exit(1);
    } catch (IOException e) {
        e.printStackTrace();
        System.exit(1);
    }

    return false;
}
```

OAUTH REQUESTS TO IMGUR IN JAVA
CREATE ALBUM (3/3)

Class CreateAlbum (oauth request execution)



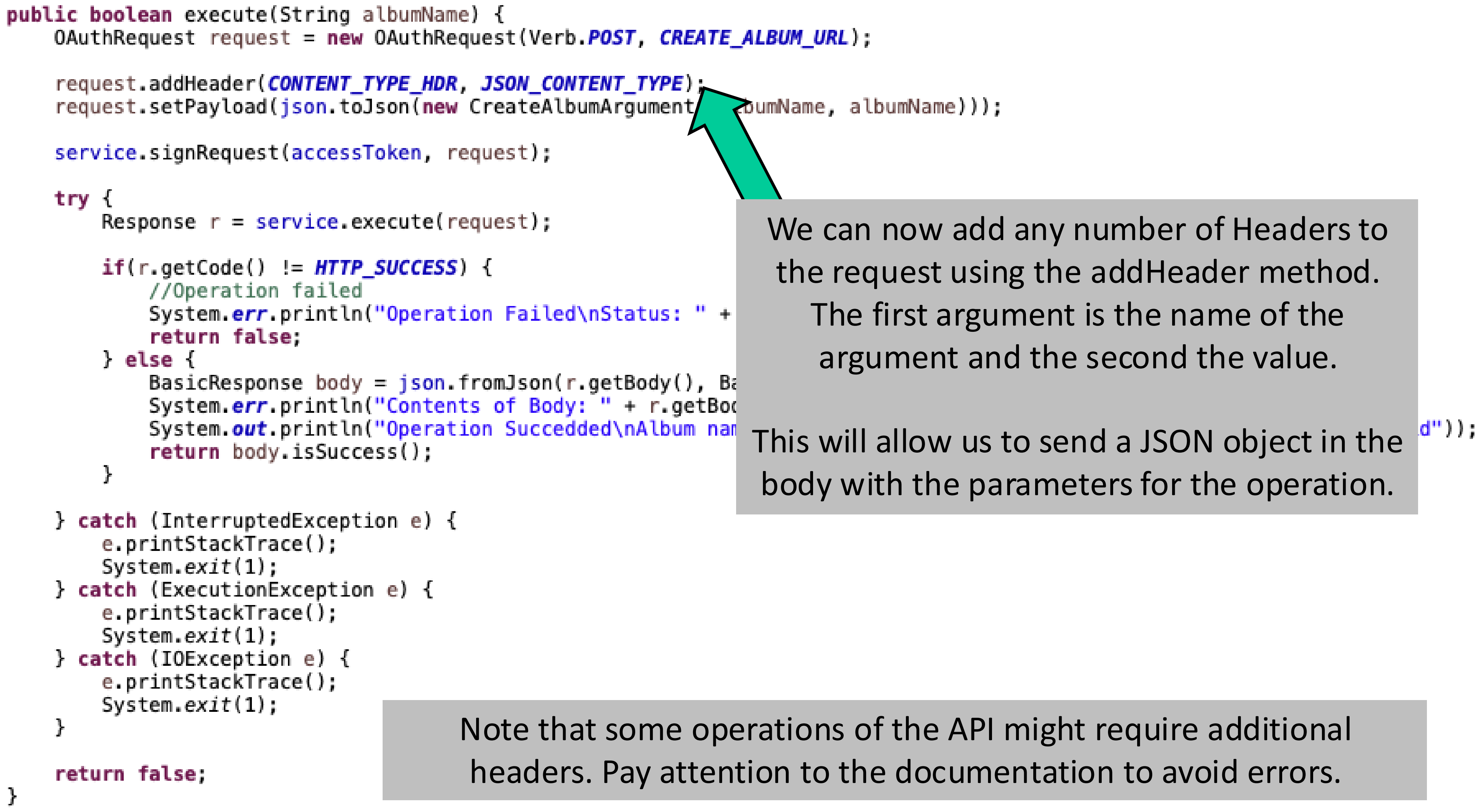
```
public boolean execute(String albumName) {
    OAuthRequest request = new OAuthRequest(Verb.POST, CREATE_ALBUM_URL);

    request.addHeader(CONTENT_TYPE_HDR, JSON_CONTENT_TYPE);
    request.setPayload(json.toJson(new CreateAlbumArguments(albumName, al      ame)));

    service.signRequest(accessToken, request);

    try {
        Response r = service.execute(request);

        if(r.getCode() != HTTP_SUCCESS) {
            //Operation failed
            System.err.println("Operation Failed\nStatus: " +
            return false;
        } else {
            BasicResponse body = json.fromJson(r.getBody(), Ba
            System.err.println("Contents of Body: " + r.getBod
            System.out.println("Operation Succedded\nAlbum nam                              d"));
            return body.isSuccess();
        }

    } catch (InterruptedException e) {
        e.printStackTrace();
        System.exit(1);
    } catch (ExecutionException e) {
        e.printStackTrace();
        System.exit(1);
    } catch (IOException e) {
        e.printStackTrace();
        System.exit(1);
    }

    return false;
}
```

To execute an Oauth request we start by creating an instance of OAuthRequest (class from the ScribeJava library) and in the constructor we state the type of the REST operation (in this case POST) and the URL of the operation we are going to execute.

(Notice that some URLs might need to be parameterized with variables)

OAUTH REQUESTS TO IMGUR IN JAVA
CREATE ALBUM (3/3)

Class CreateAlbum (oauth request execution)



```
public boolean execute(String albumName) {
    OAuthRequest request = new OAuthRequest(Verb.POST, CREATE_ALBUM_URL);

    request.addHeader(CONTENT_TYPE_HDR, JSON_CONTENT_TYPE);
    request.setPayload(json.toJson(new CreateAlbumArgument    bumName, albumName)));

    service.signRequest(accessToken, request);

    try {
        Response r = service.execute(request);

        if(r.getCode() != HTTP_SUCCESS) {
            //Operation failed
            System.err.println("Operation Failed\nStatus: " +
            return false;
        } else {
            BasicResponse body = json.fromJson(r.getBody(), Ba
            System.err.println("Contents of Body: " + r.getBod
            System.out.println("Operation Succedded\nAlbum nam                                   d"));
            return body.isSuccess();
        }

    } catch (InterruptedException e) {
        e.printStackTrace();
        System.exit(1);
    } catch (ExecutionException e) {
        e.printStackTrace();
        System.exit(1);
    } catch (IOException e) {
        e.printStackTrace();
        System.exit(1);
    }

    return false;
}
```

We can now add any number of Headers to the request using the addHeader method. The first argument is the name of the argument and the second the value.

This will allow us to send a JSON object in the body with the parameters for the operation.

Note that some operations of the API might require additional headers. Pay attention to the documentation to avoid errors.

OAUTH REQUESTS TO IMGUR IN JAVA
CREATE ALBUM (3/3)

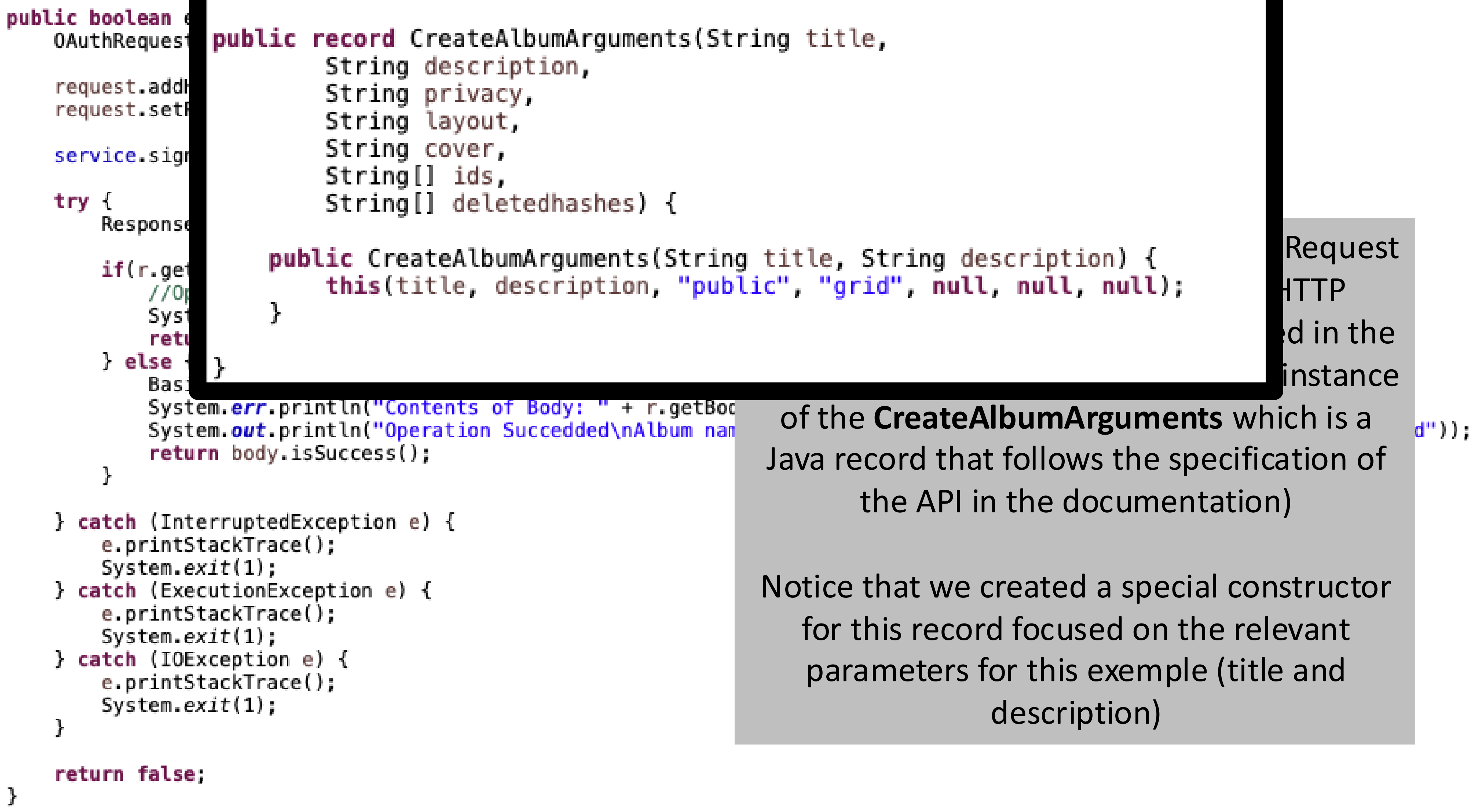
Class CreateAlbum (oauth request execution)

```java
public boolean execute(String albumName) {
    OAuthRequest request = new OAuthRequest(Verb.POST, CREATE_ALBUM_URL);

    request.addHeader(CONTENT_TYPE_HDR, JSON_CONTENT_TYPE);
    request.setPayload(json.toJson(new CreateAlbumArguments(albumName, albumName)));

    service.signRequest(accessToken, request);

    try {
        Response r = service.execute(request);

        if(r.getCode() != HTTP_SUCCESS) {
            //Operation failed
            System.err.println("Operation Failed\nStatus: " +
            return false;
        } else {
            BasicResponse body = json.fromJson(r.getBody(), Ba
            System.err.println("Contents of Body: " + r.getBoc
            System.out.println("Operation Succedded\nAlbum nam                                                d"));
            return body.isSuccess();
        }

    } catch (InterruptedException e) {
        e.printStackTrace();
        System.exit(1);
    } catch (ExecutionException e) {
        e.printStackTrace();
        System.exit(1);
    } catch (IOException e) {
        e.printStackTrace();
        System.exit(1);
    }

    return false;
}
```

The setPayload method of the OAuthRequest allows to define the body of the HTTP request (only one value can be carried in the body). In this case we are passing an instance of the **CreateAlbumArguments** which is a Java record that follows the specification of the API in the documentation)

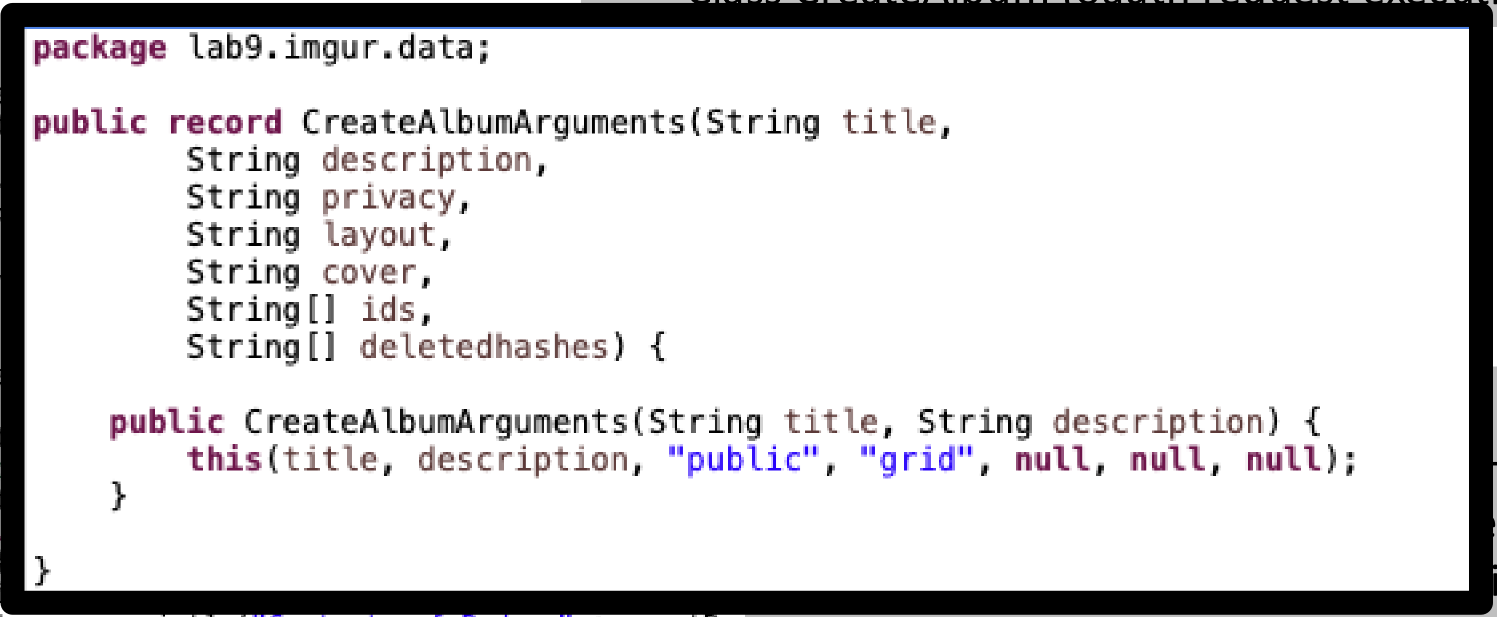
OAUTH REQUESTS TO IMGUR IN JAVA
CREATE ALBUM (3/3)

Class CreateAlbum (oauth request execution)



```
public boolean e
    OAuthRequest

    request.addH
    request.setP

    service.sign

    try {
        Response

        if(r.get
            //Op
            Syst
            retu
        } else {
            Basi
            System.err.println("Contents of Body: " + r.getBoc
            System.out.println("Operation Succedded\nAlbum nam
            return body.isSuccess();
        }

    } catch (InterruptedException e) {
        e.printStackTrace();
        System.exit(1);
    } catch (ExecutionException e) {
        e.printStackTrace();
        System.exit(1);
    } catch (IOException e) {
        e.printStackTrace();
        System.exit(1);
    }

    return false;
}
```

```
d"));
```



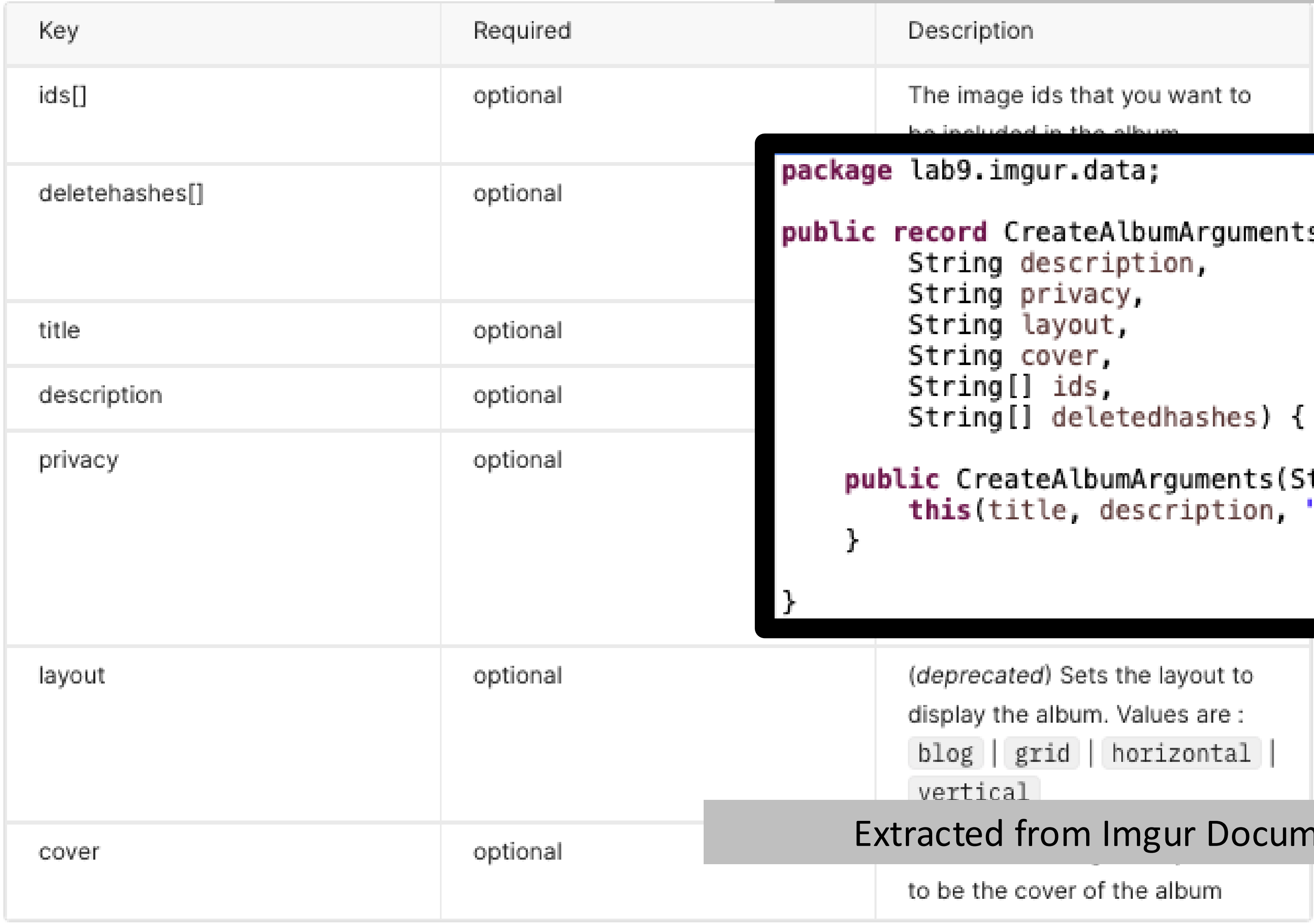
```
package lab9.imgur.data;

public record CreateAlbumArguments(String title,
        String description,
        String privacy,
        String layout,
        String cover,
        String[] ids,
        String[] deletedhashes) {

    public CreateAlbumArguments(String title, String description) {
        this(title, description, "public", "grid", null, null, null);
    }
}
```

Request
HTTP
d in the
instance
of the **CreateAlbumArguments** which is a Java record that follows the specification of the API in the documentation)

Notice that we created a special constructor for this record focused on the relevant parameters for this exemple (title and description)

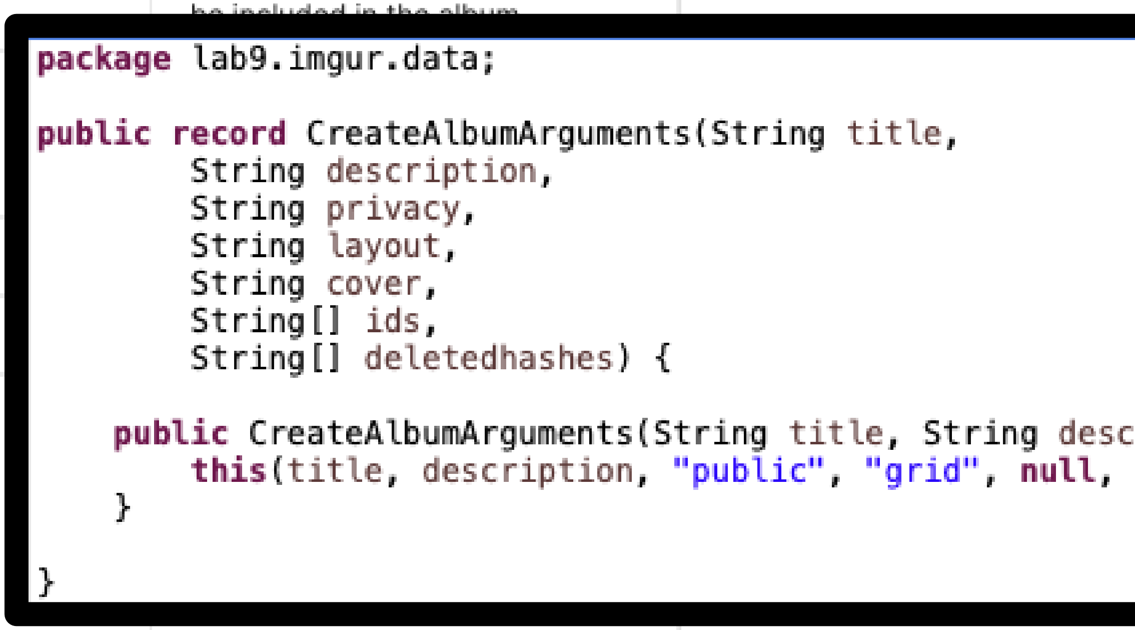
OAUTH REQUESTS TO IMGUR IN JAVA
CREATE ALBUM (AUXILIARY: CREATEALBUMARGUMENTS)

Record CreateAlbumArguments (arguments to request)



Parameters

Key	Required	Description
ids[]	optional	The image ids that you want to be included in the album
deletehashes[]	optional	
title	optional	
description	optional	
privacy	optional	
layout	optional	*(deprecated)* Sets the layout to display the album. Values are : `blog` | `grid` | `horizontal` | `vertical`
cover	optional	to be the cover of the album

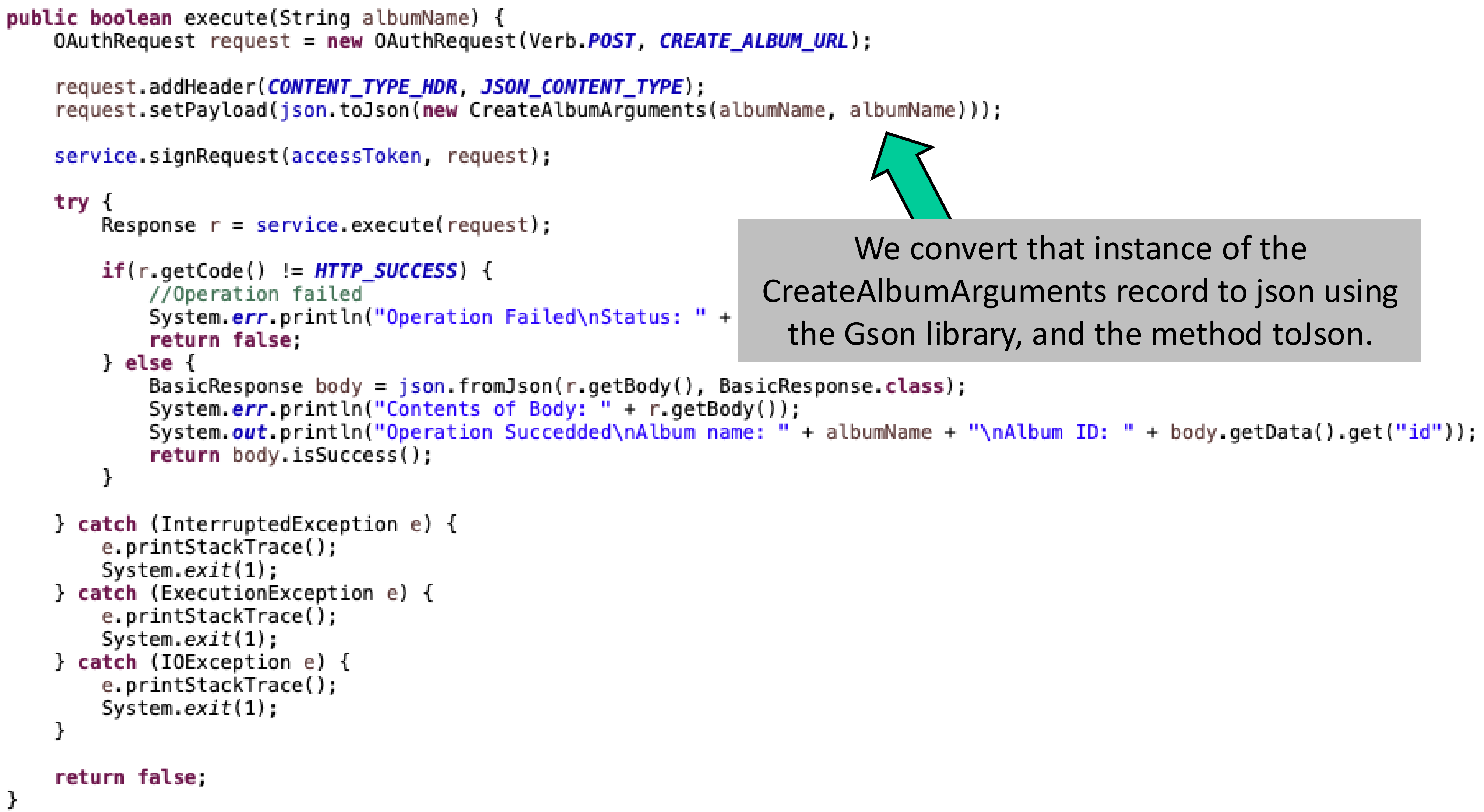


```java
package lab9.imgur.data;

public record CreateAlbumArguments(String title,
        String description,
        String privacy,
        String layout,
        String cover,
        String[] ids,
        String[] deletedhashes) {

    public CreateAlbumArguments(String title, String desc
        this(title, description, "public", "grid", null,
    }

}
```

Extracted from Imgur Documentation

OAuth Requests to Imgur in JAVA
Create Album (3/3)

Class CreateAlbum (oauth request execution)

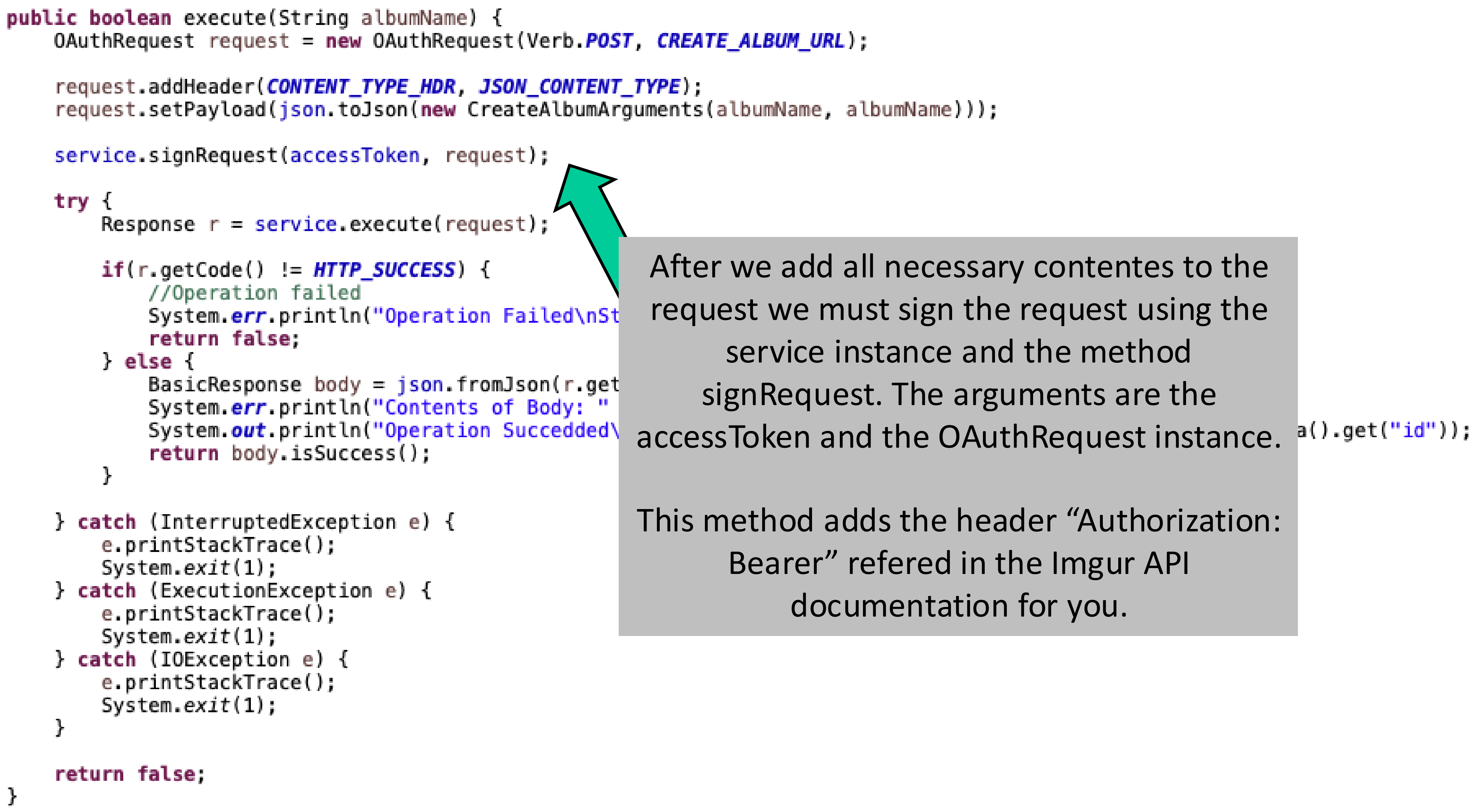


```java
public boolean execute(String albumName) {
    OAuthRequest request = new OAuthRequest(Verb.POST, CREATE_ALBUM_URL);

    request.addHeader(CONTENT_TYPE_HDR, JSON_CONTENT_TYPE);
    request.setPayload(json.toJson(new CreateAlbumArguments(albumName, albumName)));

    service.signRequest(accessToken, request);

    try {
        Response r = service.execute(request);

        if(r.getCode() != HTTP_SUCCESS) {
            //Operation failed
            System.err.println("Operation Failed\nStatus: " +
            return false;
        } else {
            BasicResponse body = json.fromJson(r.getBody(), BasicResponse.class);
            System.err.println("Contents of Body: " + r.getBody());
            System.out.println("Operation Succedded\nAlbum name: " + albumName + "\nAlbum ID: " + body.getData().get("id"));
            return body.isSuccess();
        }

    } catch (InterruptedException e) {
        e.printStackTrace();
        System.exit(1);
    } catch (ExecutionException e) {
        e.printStackTrace();
        System.exit(1);
    } catch (IOException e) {
        e.printStackTrace();
        System.exit(1);
    }

    return false;
}
```

We convert that instance of the CreateAlbumArguments record to json using the Gson library, and the method toJson.

OAUTH REQUESTS TO IMGUR IN JAVA
CREATE ALBUM (3/3)

Class CreateAlbum (oauth request execution)



```
public boolean execute(String albumName) {
    OAuthRequest request = new OAuthRequest(Verb.POST, CREATE_ALBUM_URL);

    request.addHeader(CONTENT_TYPE_HDR, JSON_CONTENT_TYPE);
    request.setPayload(json.toJson(new CreateAlbumArguments(albumName, albumName)));

    service.signRequest(accessToken, request);

    try {
        Response r = service.execute(request);

        if(r.getCode() != HTTP_SUCCESS) {
            //Operation failed
            System.err.println("Operation Failed\nSt
            return false;
        } else {
            BasicResponse body = json.fromJson(r.get
            System.err.println("Contents of Body: "
            System.out.println("Operation Succedded\                                   a().get("id"));
            return body.isSuccess();
        }

    } catch (InterruptedException e) {
        e.printStackTrace();
        System.exit(1);
    } catch (ExecutionException e) {
        e.printStackTrace();
        System.exit(1);
    } catch (IOException e) {
        e.printStackTrace();
        System.exit(1);
    }

    return false;
}
```

After we add all necessary contentes to the request we must sign the request using the service instance and the method signRequest. The arguments are the accessToken and the OAuthRequest instance.

This method adds the header “Authorization: Bearer” refered in the Imgur API documentation for you.

OAUTH REQUESTS TO IMGUR IN JAVA CREATE ALBUM (3/3)

Class CreateAlbum (oauth request execution)

```
public boolean execute(String albumName) {
    OAuthRequest request = new OAuthRequest(Verb.POST, CREATE_ALBUM_URL);

    request.addHeader(CONTENT_TYPE_HDR, JSON_CONTENT_TYPE);
    request.setPayload(json.toJson(new CreateAlbumArguments(albumName, albumName)));

    service.signRequest(accessToken, request);

    try {
        Response r = service.execute(request);

        if(r.getCode() != HTTP_SUCCESS) {
            //Operation failed
            System.err.println("Operation Failed\nStatus: " + r.get
            return false;
        } else {
            BasicResponse body = json.fromJson(r.getBody(), BasicRe
            System.err.println("Contents of Body: " + r.getBody());
            System.out.println("Operation Succedded\nAlbum name: "
            return body.isSuccess();
        }

    } catch (InterruptedException e) {
        e.printStackTrace();
        System.exit(1);
    } catch (ExecutionException e) {
        e.printStackTrace();
        System.exit(1);
    } catch (IOException e) {
        e.printStackTrace();
        System.exit(1);
    }

    return false;
}
```

After signing the request, we can execute the request using the execute method of the service instance. Exceptions can be thrown if the server cannot be contacted or if a TLS connection cannot be established (for instance because your **truststore** does not have the certificates of ROOT CAs)

OAUTH REQUESTS TO IMGUR IN JAVA
CREATE ALBUM (3/3)

Class CreateAlbum (oauth request execution)

```java
public boolean execute(String albumName) {
    OAuthRequest request = new OAuthRequest(Verb.POST, CREATE_ALBUM_URL);

    request.addHeader(CONTENT_TYPE_HDR, JSON_CONTENT_TYPE);
    request.setPayload(json.toJson(new CreateAlbumArguments(albumName, albumName)));

    service.signRequest(accessToken, request);

    try {
        Response r = service.execute(request);

        if(r.getCode() != HTTP_SUCCESS) {
            //Operation failed
            System.err.println("Operation Failed\nStatus:
            return false;
        } else {
            BasicResponse body = json.fromJson(r.getBody()
            System.err.println("Contents of Body: " + r.ge
            System.out.println("Operation Succedded\nAlbum                          t("id"));
            return body.isSuccess();
        }

    } catch (InterruptedException e) {
        e.printStackTrace();
        System.exit(1);
    } catch (ExecutionException e) {
        e.printStackTrace();
        System.exit(1);
    } catch (IOException e) {
        e.printStackTrace();
        System.exit(1);
    }

    return false;
}
```

To check the HTTP code in the response, we can use the getCode method. In this case we are verifying that it was successful by checking if it is 200 OK

(The constant HTTP_SUCCESS has a value of 200)

OAUTH REQUESTS TO IMGUR IN JAVA
CREATE ALBUM (3/3)

Class CreateAlbum (oauth request execution)

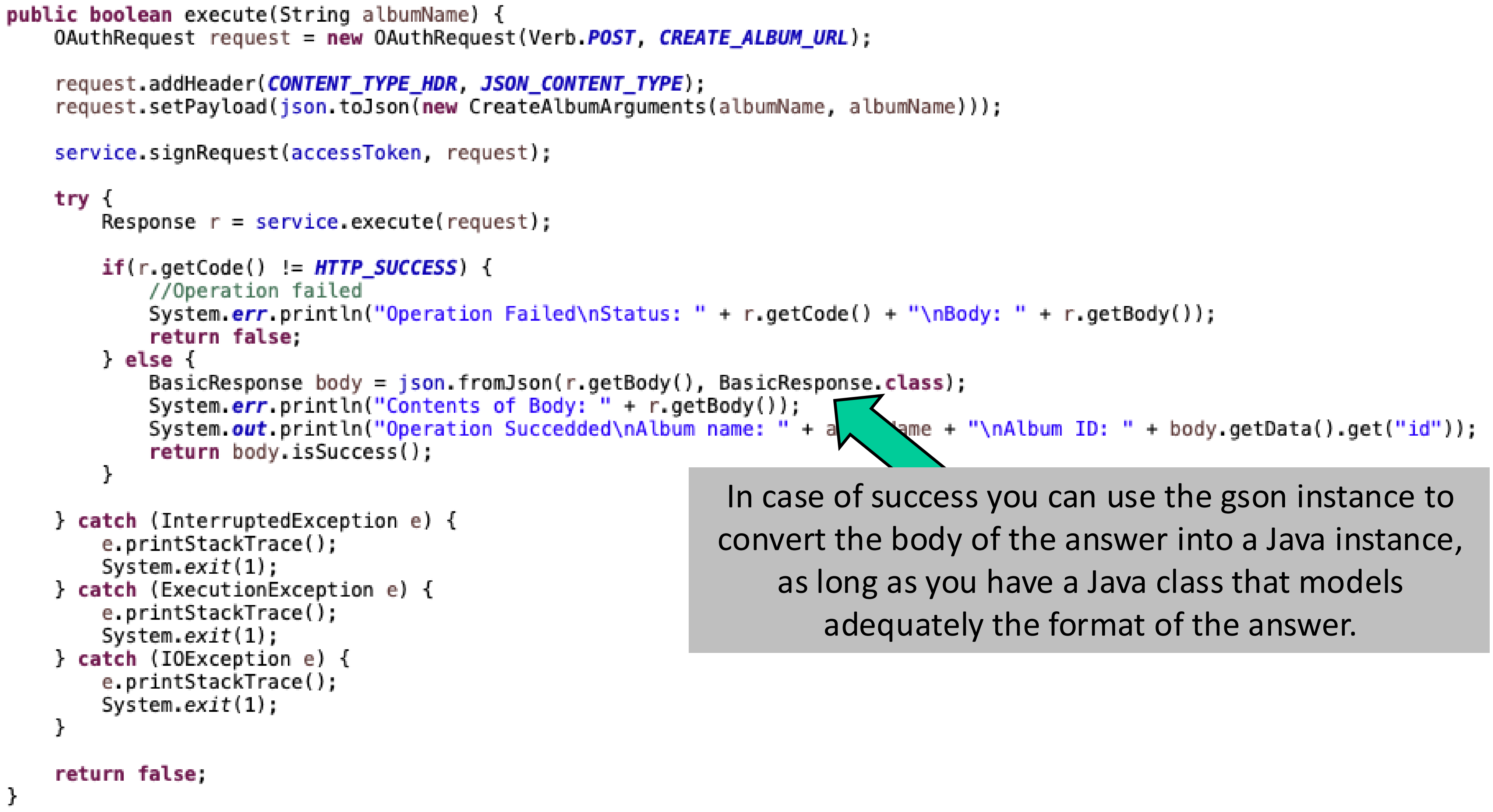
```java
public boolean execute(String albumName) {
    OAuthRequest request = new OAuthRequest(Verb.POST, CREATE_ALBUM_URL);

    request.addHeader(CONTENT_TYPE_HDR, JSON_CONTENT_TYPE);
    request.setPayload(json.toJson(new CreateAlbumArguments(albumName, albumName)));

    service.signRequest(accessToken, request);

    try {
        Response r = service.execute(request);

        if(r.getCode() != HTTP_SUCCESS) {
            //Operation failed
            System.err.println("Operation Failed\nStatus: " + r.getCode() + "\nBody: " + r.getBody());
            return false;
        } else {
            BasicResponse body = json.fromJson(r.getBody(),    rResponse.class);
            System.err.println("Contents of Body: " + r.getBod
            System.out.println("Operation Succedded\nAlbum name:      albumName + "\nAlbum ID: " + body.getData().get("id"));
            return body.isSuccess();
        }

    } catch (InterruptedException e) {
        e.printStackTrace();
        System.exit(1);
    } catch (ExecutionException e) {
        e.printStackTrace();
        System.exit(1);
    } catch (IOException e) {
        e.printStackTrace();
        System.exit(1);
    }

    return false;
}
```

If you have an error, particularly during development it is importante to check the HTTP error code and meaning (with the methods getCode and getMessage of the Response)

It is also a very good idea to check the contentes of the body of an erroneous response, since many times details about the error are there (e.g., missing headers or invalid arguments)

OAUTH REQUESTS TO IMGUR IN JAVA
CREATE ALBUM (3/3)

Class CreateAlbum (oauth request execution)

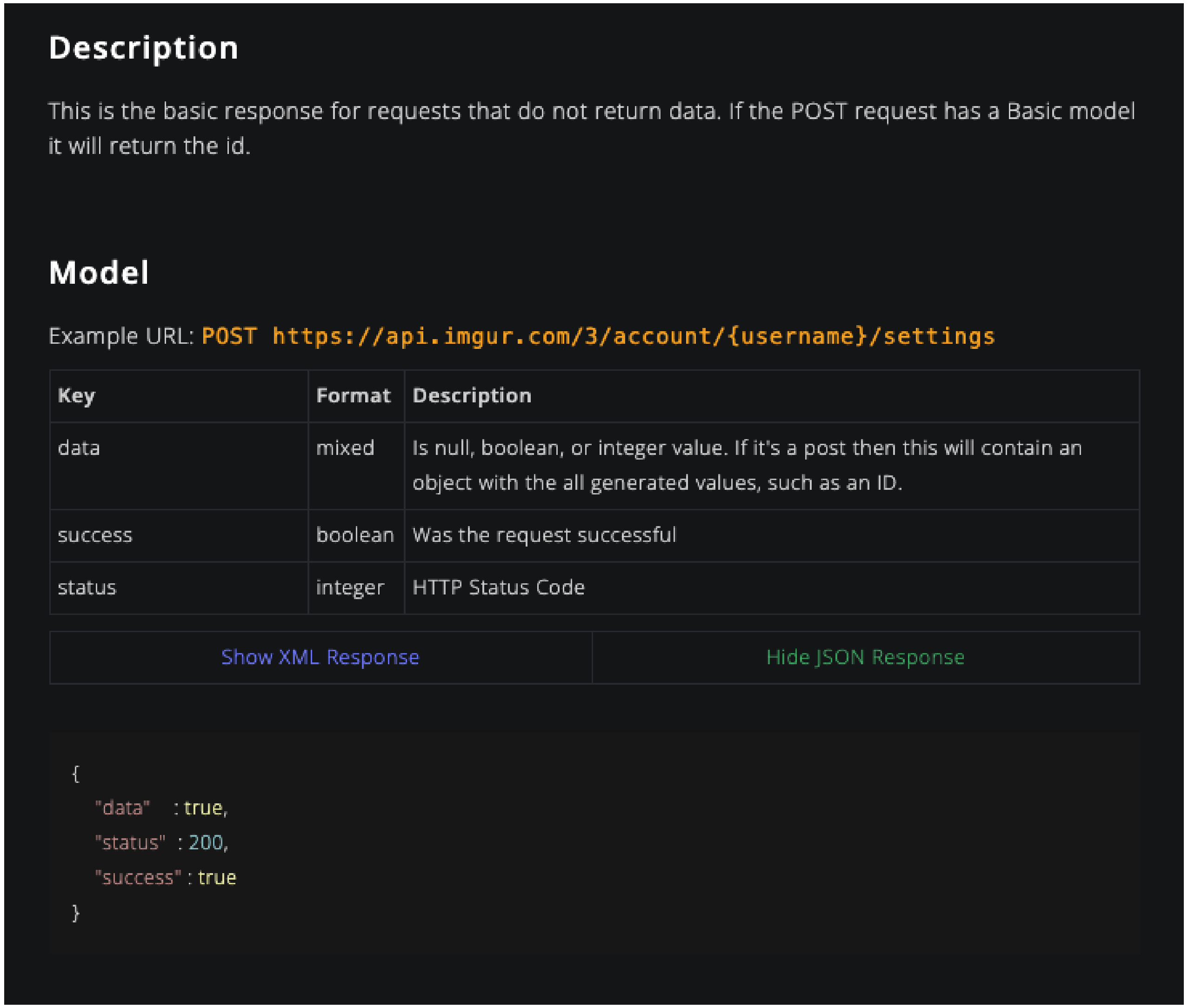


```
public boolean execute(String albumName) {
    OAuthRequest request = new OAuthRequest(Verb.POST, CREATE_ALBUM_URL);

    request.addHeader(CONTENT_TYPE_HDR, JSON_CONTENT_TYPE);
    request.setPayload(json.toJson(new CreateAlbumArguments(albumName, albumName)));

    service.signRequest(accessToken, request);

    try {
        Response r = service.execute(request);

        if(r.getCode() != HTTP_SUCCESS) {
            //Operation failed
            System.err.println("Operation Failed\nStatus: " + r.getCode() + "\nBody: " + r.getBody());
            return false;
        } else {
            BasicResponse body = json.fromJson(r.getBody(), BasicResponse.class);
            System.err.println("Contents of Body: " + r.getBody());
            System.out.println("Operation Succedded\nAlbum name: " + a      ame + "\nAlbum ID: " + body.getData().get("id"));
            return body.isSuccess();
        }

    } catch (InterruptedException e) {
        e.printStackTrace();
        System.exit(1);
    } catch (ExecutionException e) {
        e.printStackTrace();
        System.exit(1);
    } catch (IOException e) {
        e.printStackTrace();
        System.exit(1);
    }

    return false;
}
```

In case of success you can use the gson instance to convert the body of the answer into a Java instance, as long as you have a Java class that models adequately the format of the answer.

OAUTH REQUESTS TO IMGUR IN JAVA
CREATE ALBUM (3/3)

Auxiliar class to capture reply (BasicResponse where data is itself a Json object)



Description

This is the basic response for requests that do not return data. If the POST request has a Basic model it will return the id.

Model

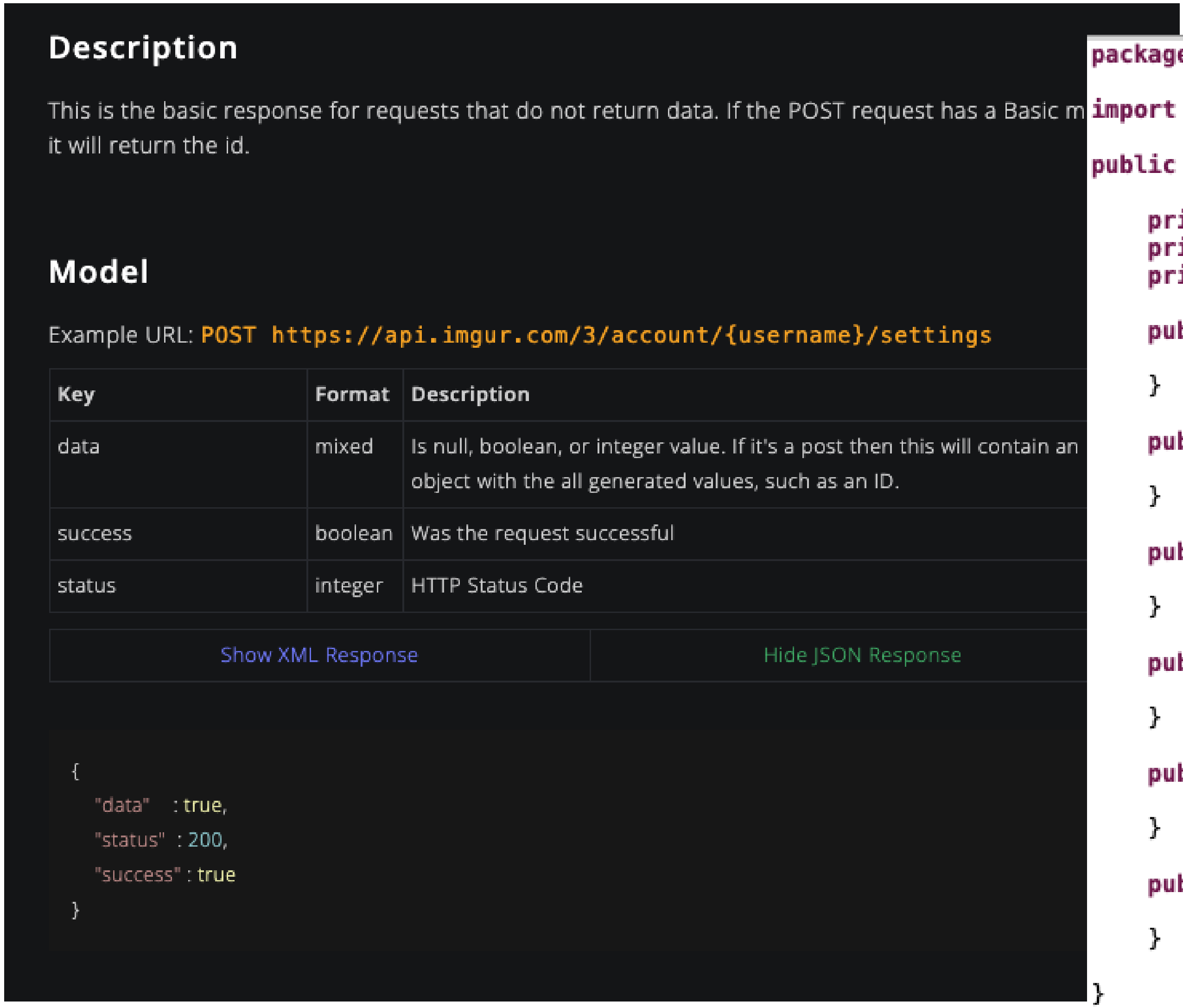
Example URL: `POST https://api.imgur.com/3/account/{username}/settings`

Key	Format	Description
data	mixed	Is null, boolean, or integer value. If it's a post then this will contain an object with the all generated values, such as an ID.
success	boolean	Was the request successful
status	integer	HTTP Status Code

```
{
    "data"    : true,
    "status"  : 200,
    "success" : true
}
```

OAUTH REQUESTS TO IMGUR IN JAVA
CREATE ALBUM (3/3)

Auxiliar class to capture reply (BasicResponse where data is itself a Json object)



Description

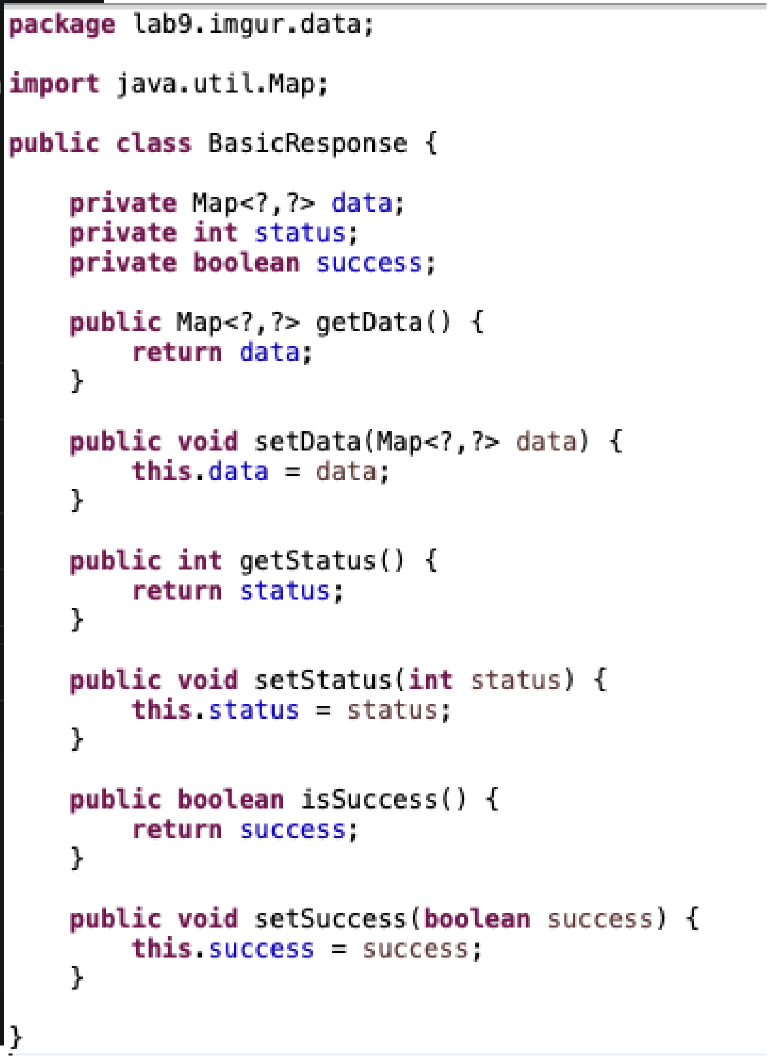
This is the basic response for requests that do not return data. If the POST request has a Basic m
it will return the id.

Model

Example URL: POST https://api.imgur.com/3/account/{username}/settings

Key	Format	Description
data	mixed	Is null, boolean, or integer value. If it's a post then this will contain an object with the all generated values, such as an ID.
success	boolean	Was the request successful
status	integer	HTTP Status Code

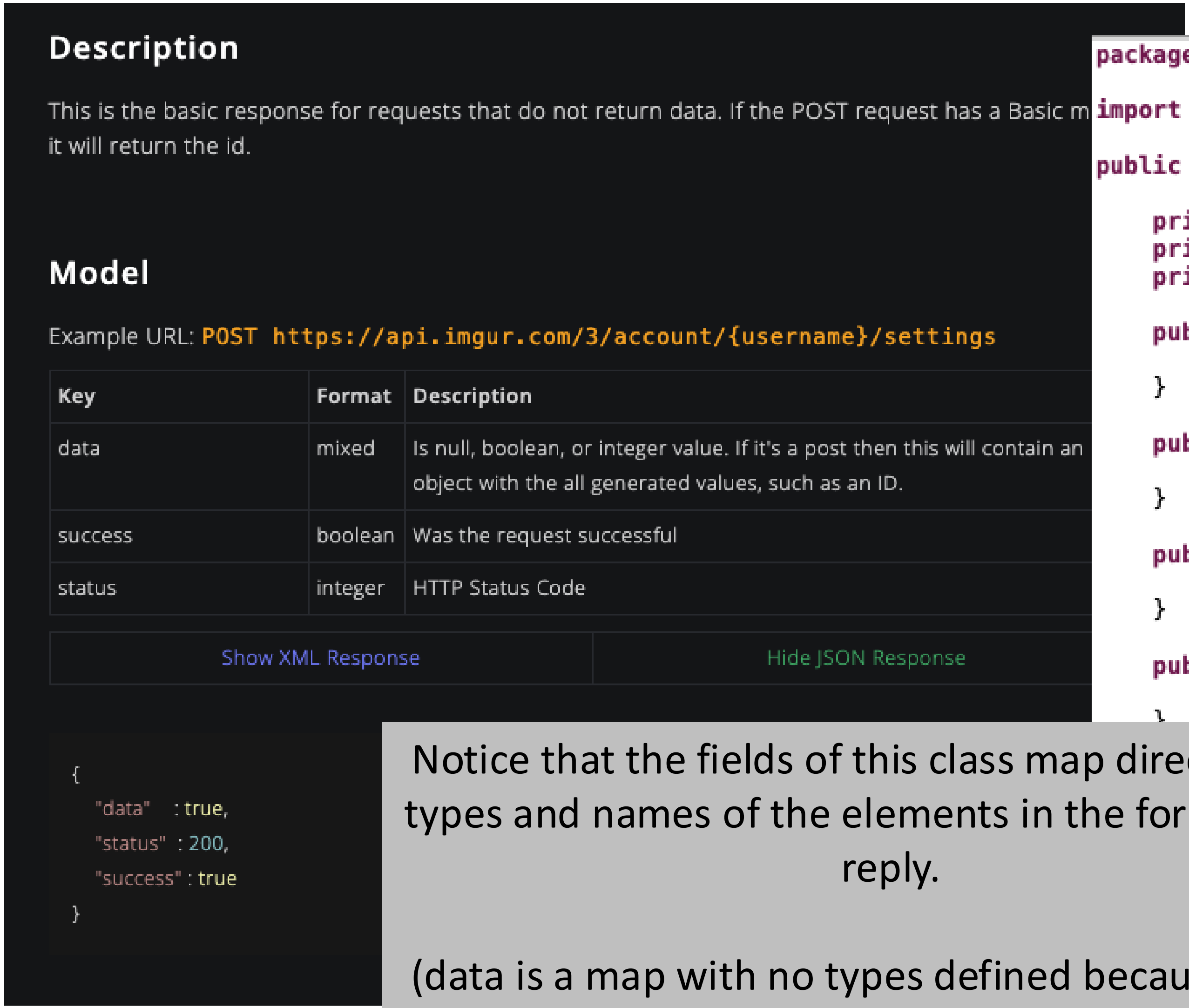
```
{
    "data"    : true,
    "status"  : 200,
    "success" : true
}
```



```java
package lab9.imgur.data;

import java.util.Map;

public class BasicResponse {

    private Map<?,?> data;
    private int status;
    private boolean success;

    public Map<?,?> getData() {
        return data;
    }

    public void setData(Map<?,?> data) {
        this.data = data;
    }

    public int getStatus() {
        return status;
    }

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

    public boolean isSuccess() {
        return success;
    }

    public void setSuccess(boolean success) {
        this.success = success;
    }

}
```

OAUTH REQUESTS TO IMGUR IN JAVA
CREATE ALBUM (3/3)

Auxiliar class to capture reply (BasicResponse where data is itself a Json object)



Description

This is the basic response for requests that do not return data. If the POST request has a Basic m
it will return the id.

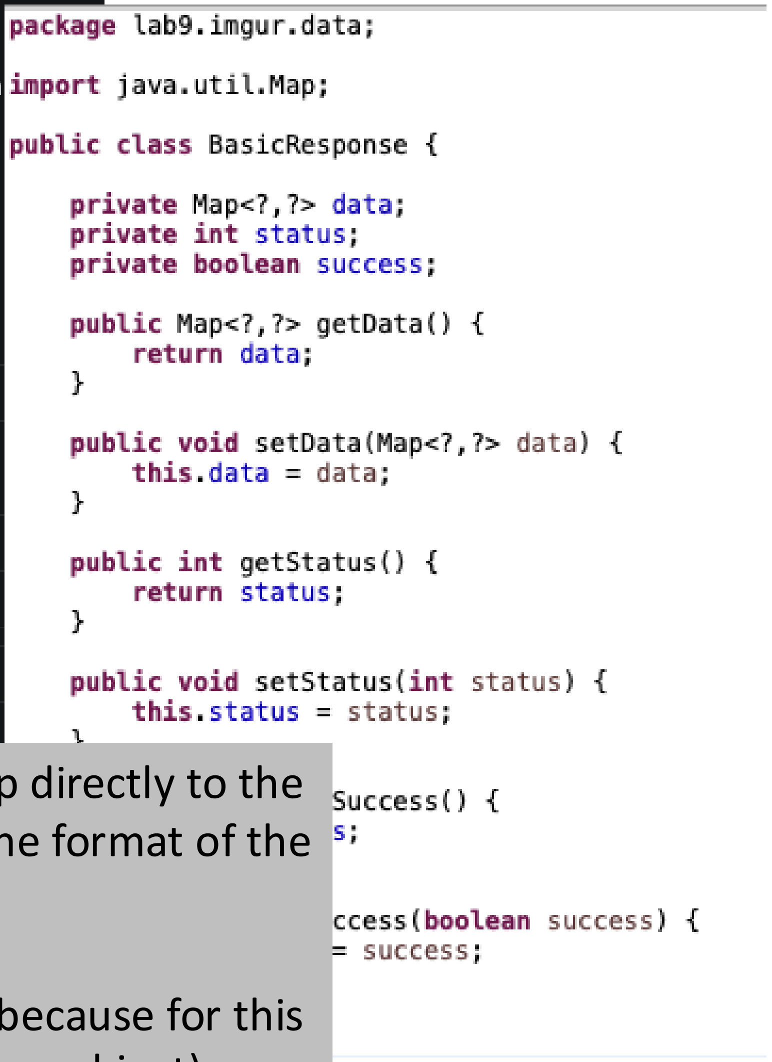
Model

Example URL: POST https://api.imgur.com/3/account/{username}/settings

Key	Format	Description
data	mixed	Is null, boolean, or integer value. If it's a post then this will contain an object with the all generated values, such as an ID.
success	boolean	Was the request successful
status	integer	HTTP Status Code

Show XML Response | Hide JSON Response

```
{
    "data"   : true,
    "status"  : 200,
    "success" : true
}
```



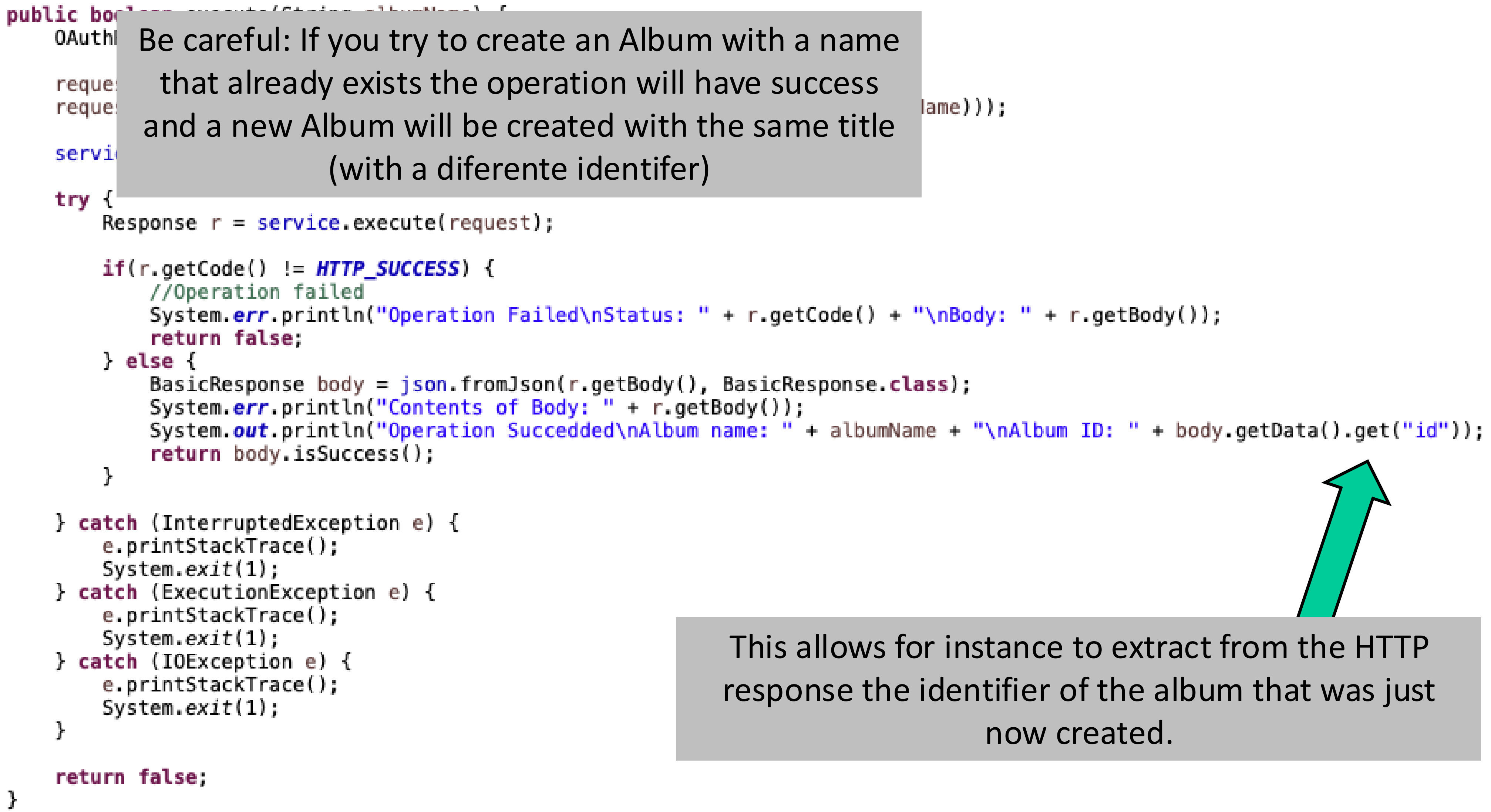
```
package lab9.imgur.data;

import java.util.Map;

public class BasicResponse {

    private Map<?,?> data;
    private int status;
    private boolean success;

    public Map<?,?> getData() {
        return data;
    }

    public void setData(Map<?,?> data) {
        this.data = data;
    }

    public int getStatus() {
        return status;
    }

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

              Success() {
              s;

              ccess(boolean success) {
              = success;
```

Notice that the fields of this class map directly to the types and names of the elements in the format of the reply.

(data is a map with no types defined because for this request it will contain another json object)

OAuth Requests to Imgur in JAVA
Create Album (3/3)

Class CreateAlbum (oauth request execution)

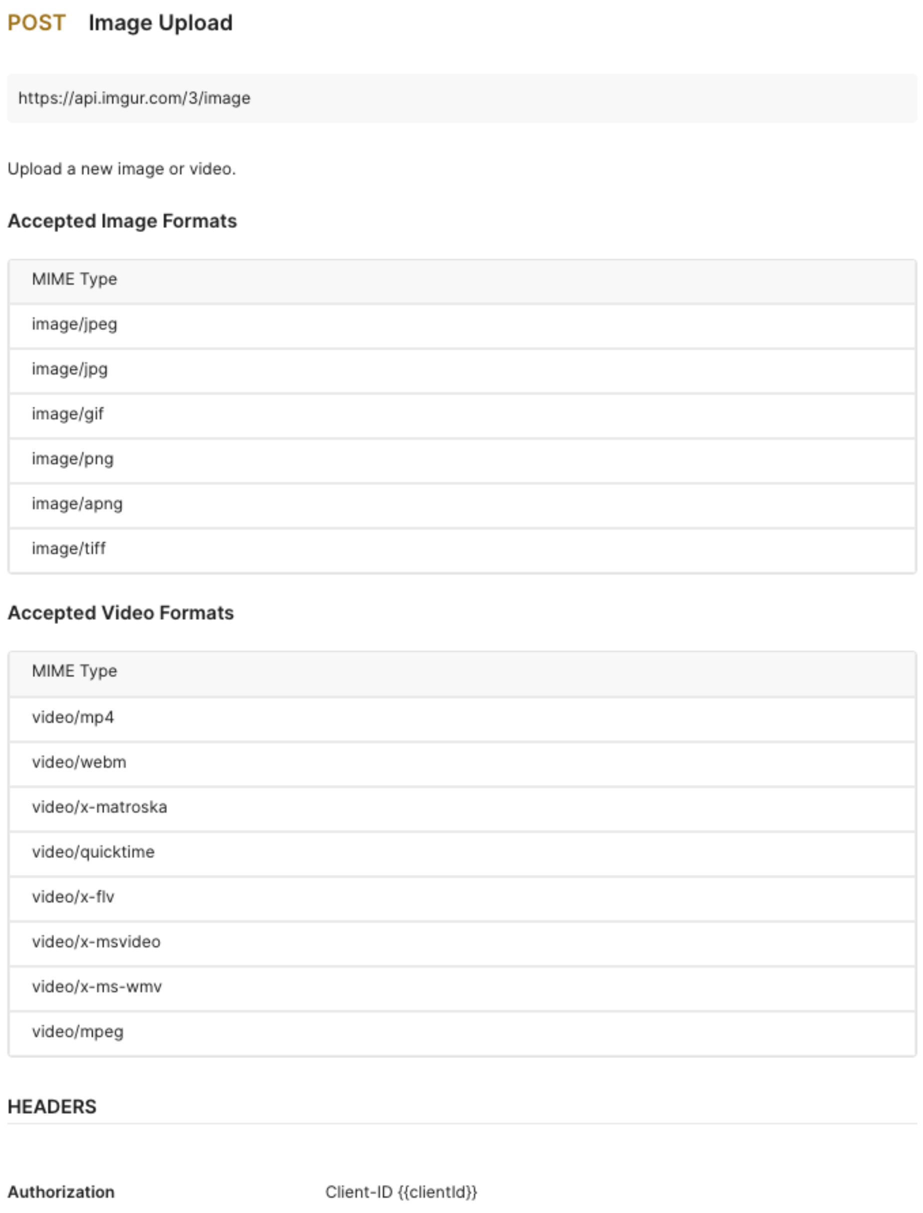


```
public boo
    OAuth

    reque
    reque                                                              ame)));

    servi

    try {
        Response r = service.execute(request);

        if(r.getCode() != HTTP_SUCCESS) {
            //Operation failed
            System.err.println("Operation Failed\nStatus: " + r.getCode() + "\nBody: " + r.getBody());
            return false;
        } else {
            BasicResponse body = json.fromJson(r.getBody(), BasicResponse.class);
            System.err.println("Contents of Body: " + r.getBody());
            System.out.println("Operation Succedded\nAlbum name: " + albumName + "\nAlbum ID: " + body.getData().get("id"));
            return body.isSuccess();
        }

    } catch (InterruptedException e) {
        e.printStackTrace();
        System.exit(1);
    } catch (ExecutionException e) {
        e.printStackTrace();
        System.exit(1);
    } catch (IOException e) {
        e.printStackTrace();
        System.exit(1);
    }

    return false;
}
```

Be careful: If you try to create an Album with a name that already exists the operation will have success and a new Album will be created with the same title (with a diferente identifer)

This allows for instance to extract from the HTTP response the identifier of the album that was just now created.

OAuth Requests to Imgur in JAVA
Image Upload (documentation)



POST Image Upload

https://api.imgur.com/3/image

Upload a new image or video.

Accepted Image Formats

MIME Type
image/jpeg
image/jpg
image/gif
image/png
image/apng
image/tiff

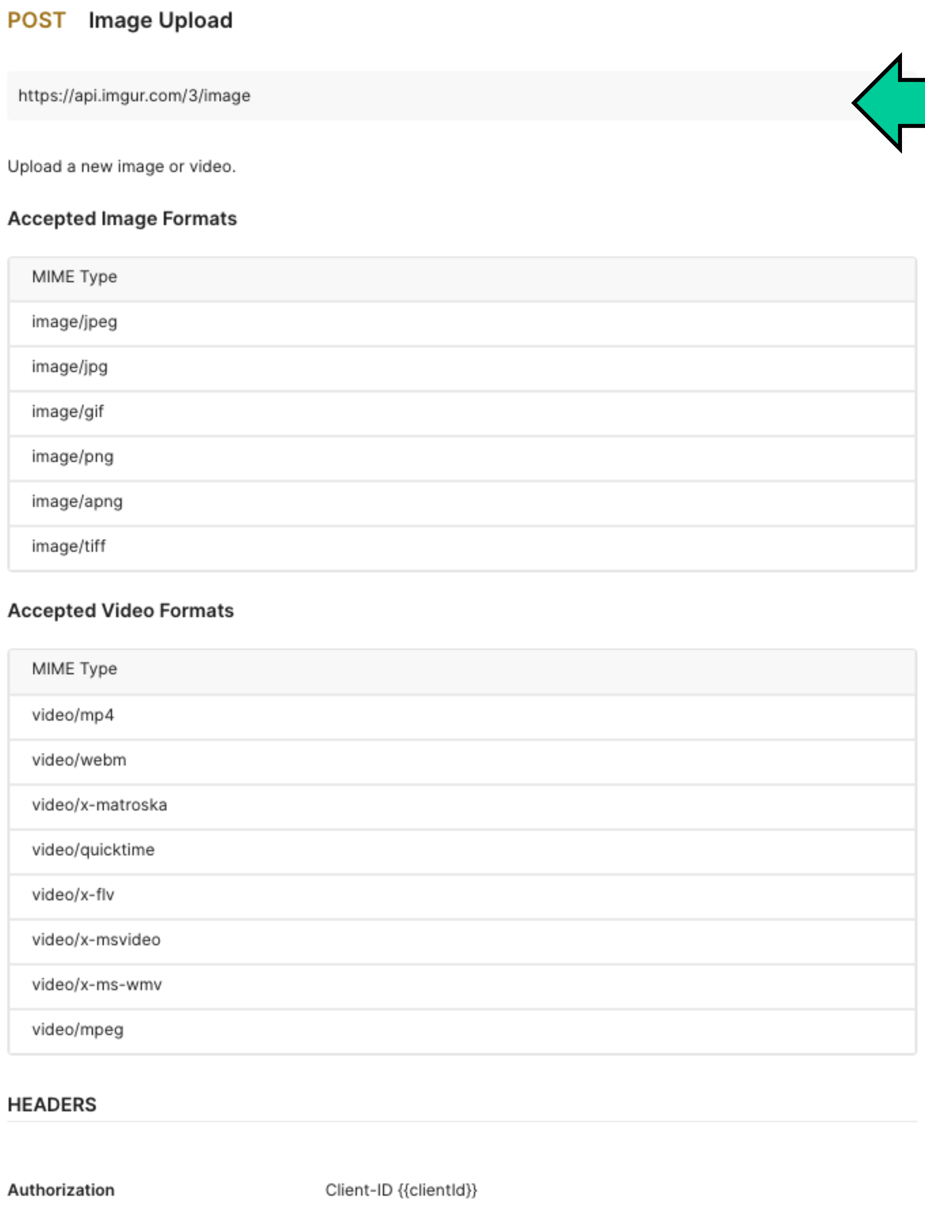
Accepted Video Formats

MIME Type
video/mp4
video/webm
video/x-matroska
video/quicktime
video/x-flv
video/x-msvideo
video/x-ms-wmv
video/mpeg

HEADERS

Authorization	Client-ID {{clientId}}

OAUTH REQUESTS TO IMGUR IN JAVA
IMAGE UPLOAD (DOCUMENTATION)



POST **Image Upload**

https://api.imgur.com/3/image

This is the URL of the request

Upload a new image or video.

Accepted Image Formats

MIME Type
image/jpeg
image/jpg
image/gif
image/png
image/apng
image/tiff

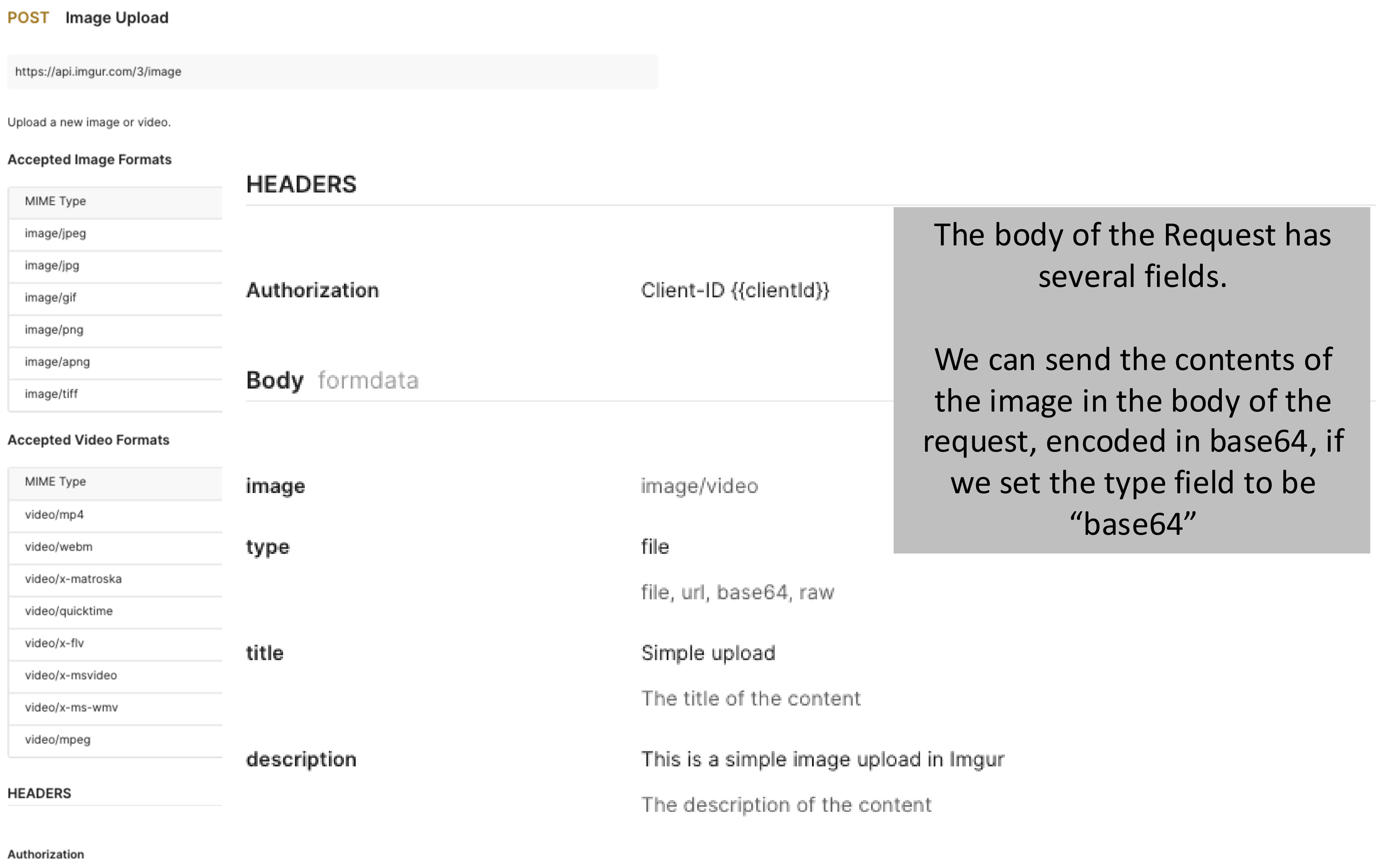
Accepted Video Formats

MIME Type
video/mp4
video/webm
video/x-matroska
video/quicktime
video/x-flv
video/x-msvideo
video/x-ms-wmv
video/mpeg

HEADERS

Authorization	Client-ID {{clientId}}

OAUTH REQUESTS TO IMGUR IN JAVA
IMAGE UPLOAD (DOCUMENTATION)



POST Image Upload

```
https://api.imgur.com/3/image
```

Upload a new image or video.

Accepted Image Formats

MIME Type
image/jpeg
image/jpg
image/gif
image/png
image/apng
image/tiff

Accepted Video Formats

MIME Type
video/mp4
video/webm
video/x-matroska
video/quicktime
video/x-flv
video/x-msvideo
video/x-ms-wmv
video/mpeg

HEADERS

Authorization

HEADERS

Authorization	Client-ID {{clientId}}

Body formdata

image	image/video
type	file file, url, base64, raw
title	Simple upload The title of the content
description	This is a simple image upload in Imgur The description of the content

The body of the Request has several fields.

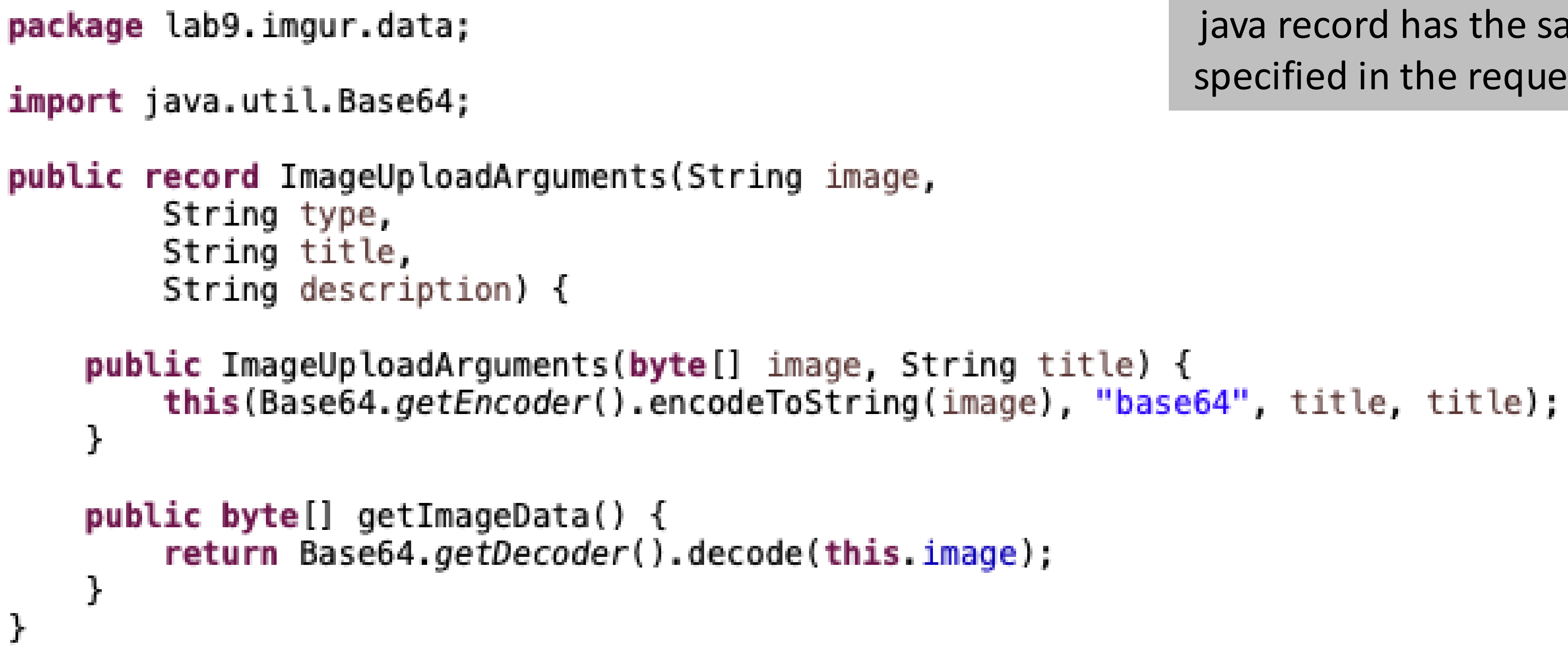
We can send the contents of the image in the body of the request, encoded in base64, if we set the type field to be "base64"

OAUTH REQUESTS TO IMGUR IN JAVA
IMAGE UPLOAD (REQUEST RECORD)

```java
package lab9.imgur.data;

import java.util.Base64;

public record ImageUploadArguments(String image,
        String type,
        String title,
        String description) {

    public ImageUploadArguments(byte[] image, String title) {
        this(Base64.getEncoder().encodeToString(image), "base64", title, title);
    }

    public byte[] getImageData() {
        return Base64.getDecoder().decode(this.image);
    }
}
```

OAUTH REQUESTS TO IMGUR IN JAVA
IMAGE UPLOAD (REQUEST RECORD)

The ImageUploadArguments java record has the same fields specified in the request format.



```java
package lab9.imgur.data;

import java.util.Base64;

public record ImageUploadArguments(String image,
        String type,
        String title,
        String description) {

    public ImageUploadArguments(byte[] image, String title) {
        this(Base64.getEncoder().encodeToString(image), "base64", title, title);
    }

    public byte[] getImageData() {
        return Base64.getDecoder().decode(this.image);
    }
}
```

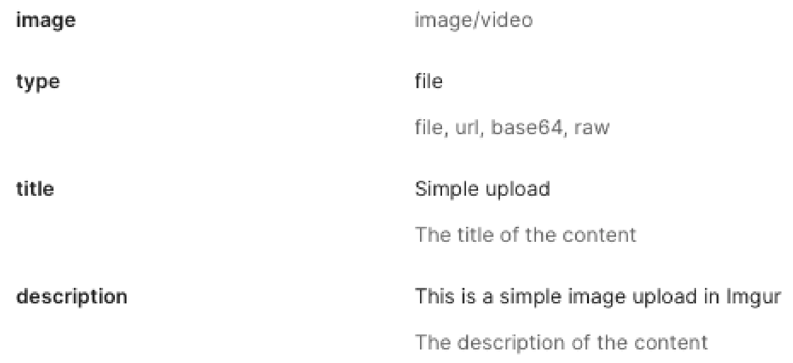


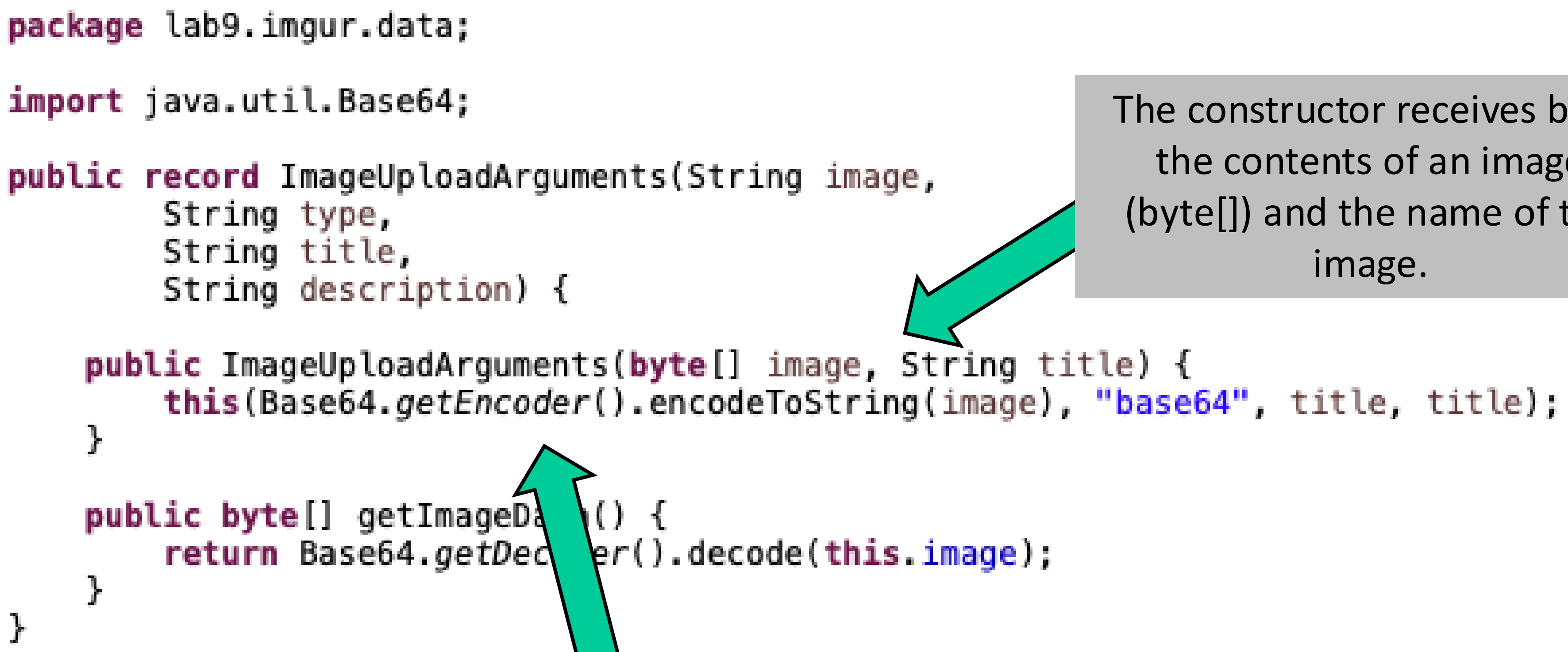
image	image/video
type	file file, url, base64, raw
title	Simple upload The title of the content
description	This is a simple image upload in Imgur The description of the content

OAUTH REQUESTS TO IMGUR IN JAVA
IMAGE UPLOAD (REQUEST RECORD)

```java
package lab9.imgur.data;

import java.util.Base64;

public record ImageUploadArguments(String image,
        String type,
        String title,
        String description) {

    public ImageUploadArguments(byte[] image, String title) {
        this(Base64.getEncoder().encodeToString(image), "base64", title, title);
    }

    public byte[] getImageData() {
        return Base64.getDecoder().decode(this.image);
    }
}
```

The constructor receives both the contents of an image (byte[]) and the name of the image.

OAUTH REQUESTS TO IMGUR IN JAVA
IMAGE UPLOAD (REQUEST RECORD)

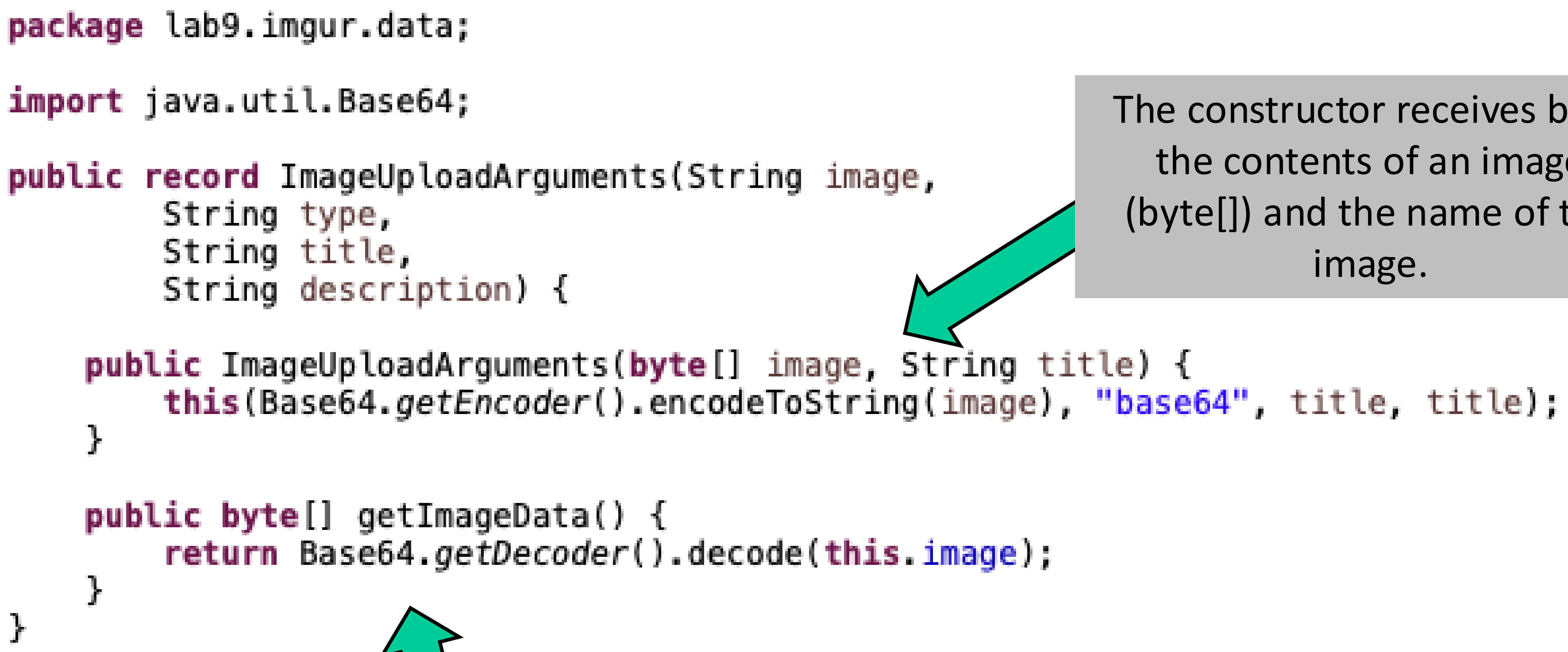


```java
package lab9.imgur.data;

import java.util.Base64;

public record ImageUploadArguments(String image,
        String type,
        String title,
        String description) {

    public ImageUploadArguments(byte[] image, String title) {
        this(Base64.getEncoder().encodeToString(image), "base64", title, title);
    }

    public byte[] getImageData() {
        return Base64.getDecoder().decode(this.image);
    }
}
```

The constructor receives both the contents of an image (byte[]) and the name of the image.

The constructor encodes the contentes of the image in a texto format that can be stored and serialized in a String format.

OAuth Requests to Imgur in JAVA
Image Upload (Request Record)



```java
package lab9.imgur.data;

import java.util.Base64;

public record ImageUploadArguments(String image,
        String type,
        String title,
        String description) {

    public ImageUploadArguments(byte[] image, String title) {
        this(Base64.getEncoder().encodeToString(image), "base64", title, title);
    }

    public byte[] getImageData() {
        return Base64.getDecoder().decode(this.image);
    }
}
```

The constructor receives both the contents of an image (byte[]) and the name of the image.

The getImageData method reverts this operation, it decodes the String back to a binary format (with the actual contents of the image).

OAuth Requests to Upload an Image in JAVA (1/3)

Class ImageUpload (preamble and constructor)

```java
public class ImageUpload {

    private static final String apiKey = "INSERT YOURS";
    private static final String apiSecret = "INSERT YOURS";
    private static final String accessTokenStr = "INSERT YOURS";

    private static final String UPLOAD_IMAGE_URL = "https://api.imgur.com/3/image";

    private static final int HTTP_SUCCESS = 200;
    private static final String CONTENT_TYPE_HDR = "Content-Type";
    private static final String JSON_CONTENT_TYPE = "application/json; charset=utf-8";

    private final Gson json;
    private final OAuth20Service service;
    private final OAuth2AccessToken accessToken;

    public ImageUpload() {
        json = new Gson();
        accessToken = new OAuth2AccessToken(accessTokenStr);
        service = new ServiceBuilder(apiKey).apiSecret(apiSecret).build(ImgurApi.instance());
    }
```

The constants and constructor in this example are almost the same as in the Create Album example. Notice however that now we have a different URL for this operation.

OAUTH REQUESTS TO UPLOAD AN IMAGE IN JAVA (2/3)

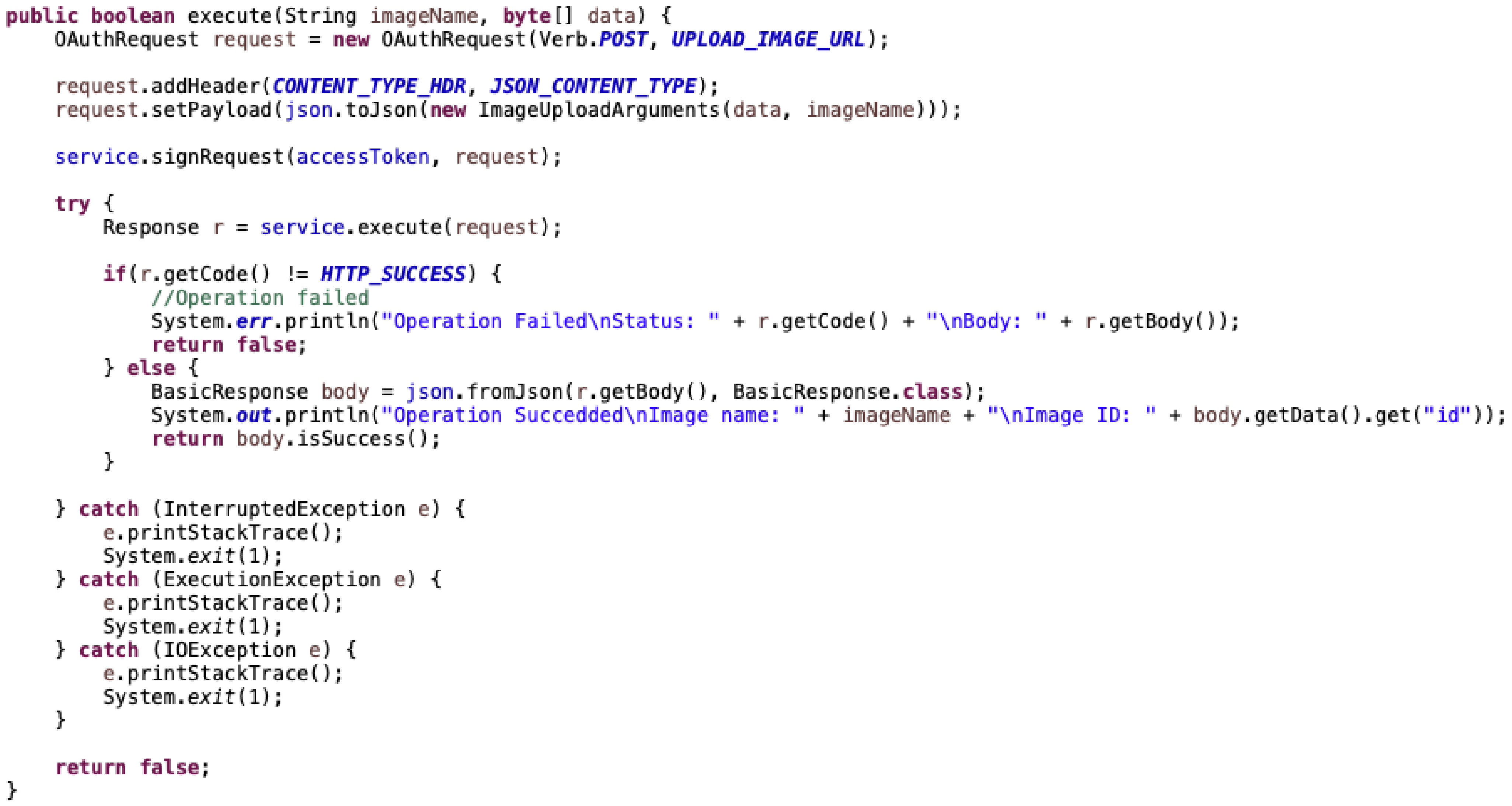
Class ImageUpload (main -- basic user interface)

```java
public static void main(String[] args) throws Exception {

    if( args.length != 1 ) {
        System.err.println("usage: java " + ImageUpload.class.getCanonicalName() + " <album-name>");
        System.exit(0);
    }

    String filename = args[0];

    byte[] data = null;

    try {
        data = Files.readAllBytes(Path.of("./", filename));
    } catch (Exception e) {
        e.printStackTrace();
        System.exit(0);
    }

    ImageUpload ca = new ImageUpload();

    if(ca.execute(filename, data))
        System.out.println("Image '" + filename + "' uploaded successfuly.");
    else
        System.err.println("Failed to upload image from '" + filename + "'");
}
```

The Main is also similar to the previous example. But here we get a filename (for an image) and then also read the bytes of the image to be able to upload them.

OAUTH REQUESTS TO UPLOAD AN IMAGE IN JAVA (3/3)

Class ImageUpload (oauth request execution)



```
public boolean execute(String imageName, byte[] data) {
    OAuthRequest request = new OAuthRequest(Verb.POST, UPLOAD_IMAGE_URL);

    request.addHeader(CONTENT_TYPE_HDR, JSON_CONTENT_TYPE);
    request.setPayload(json.toJson(new ImageUploadArguments(data, imageName)));

    service.signRequest(accessToken, request);

    try {
        Response r = service.execute(request);

        if(r.getCode() != HTTP_SUCCESS) {
            //Operation failed
            System.err.println("Operation Failed\nStatus: " + r.getCode() + "\nBody: " + r.getBody());
            return false;
        } else {
            BasicResponse body = json.fromJson(r.getBody(), BasicResponse.class);
            System.out.println("Operation Succedded\nImage name: " + imageName + "\nImage ID: " + body.getData().get("id"));
            return body.isSuccess();
        }

    } catch (InterruptedException e) {
        e.printStackTrace();
        System.exit(1);
    } catch (ExecutionException e) {
        e.printStackTrace();
        System.exit(1);
    } catch (IOException e) {
        e.printStackTrace();
        System.exit(1);
    }

    return false;
}
```

OAUTH REQUESTS TO UPLOAD AN IMAGE IN JAVA (3/3)

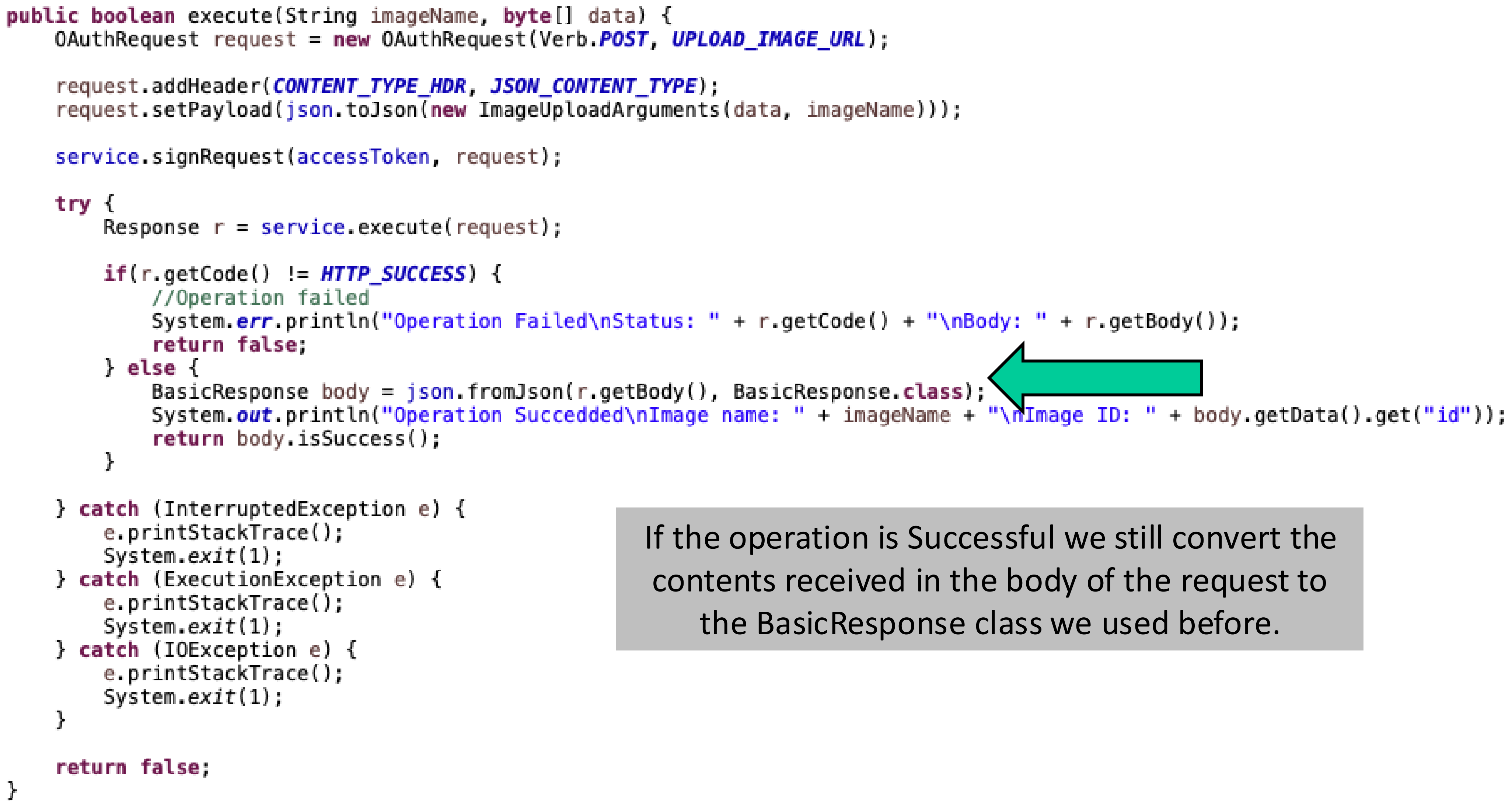
Class ImageUpload (oauth request execution)

```
public boolean execute(String imageName, byte[] data) {
    OAuthRequest request = new OAuthRequest(Verb.POST, UPLOAD_IMAGE_URL);

    request.addHeader(CONTENT_TYPE_HDR, JSON_CONTENT_TYPE);
    request.setPayload(json.toJson(new ImageUploadArguments(data, imageName)));

    service.signRequest(accessToken, request);

    try {
        Response r = service.execute(request);

        if(r.getCode() != HTTP_SUCCESS) {
            //Operation failed
            System.err.println("Operation Failed\nStatus: " + r.getCode() + "\nBody: " + r.getBody());
            return false;
        } else {
            BasicResponse body = json.fromJson(r.getBody(), BasicResponse.class);
            System.out.println("Operation Succedded\nImage name: " + imageName + "\nImage ID: " + body.getData().get("id"));
            return body.isSuccess();
        }

    } catch (InterruptedException e) {
        e.printStackTrace();
        System.exit(1);
    } catch (ExecutionException e) {
        e.printStackTrace();
        System.exit(1);
    } catch (IOException e) {
        e.printStackTrace();
        System.exit(1);
    }

    return false;
}
```

The process is very similar to the previous example. We still have to add the additional header to indicate we are going to encode the contents of the body in JSON.

OAUTH REQUESTS TO UPLOAD AN IMAGE IN JAVA (3/3)

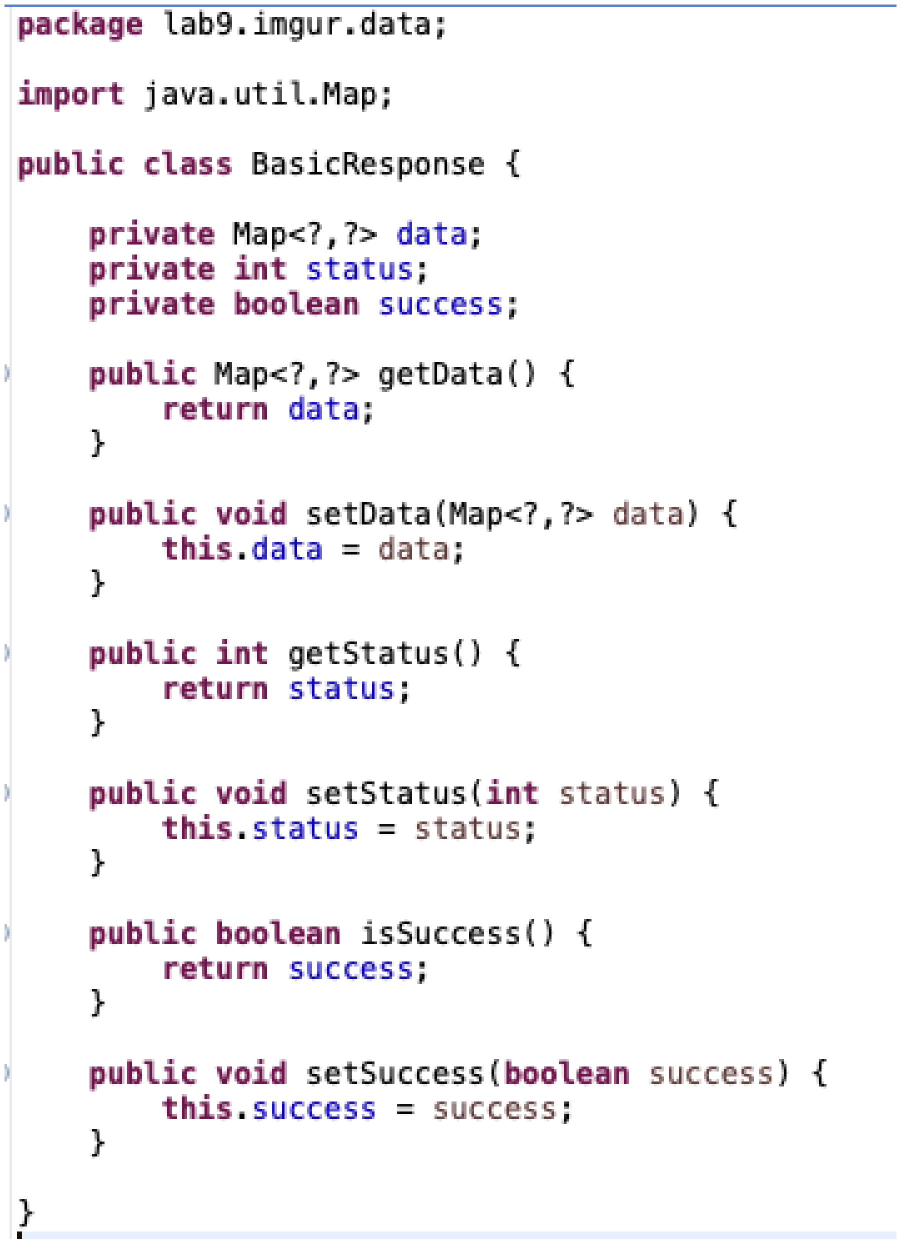
Class ImageUpload (oauth request execution)



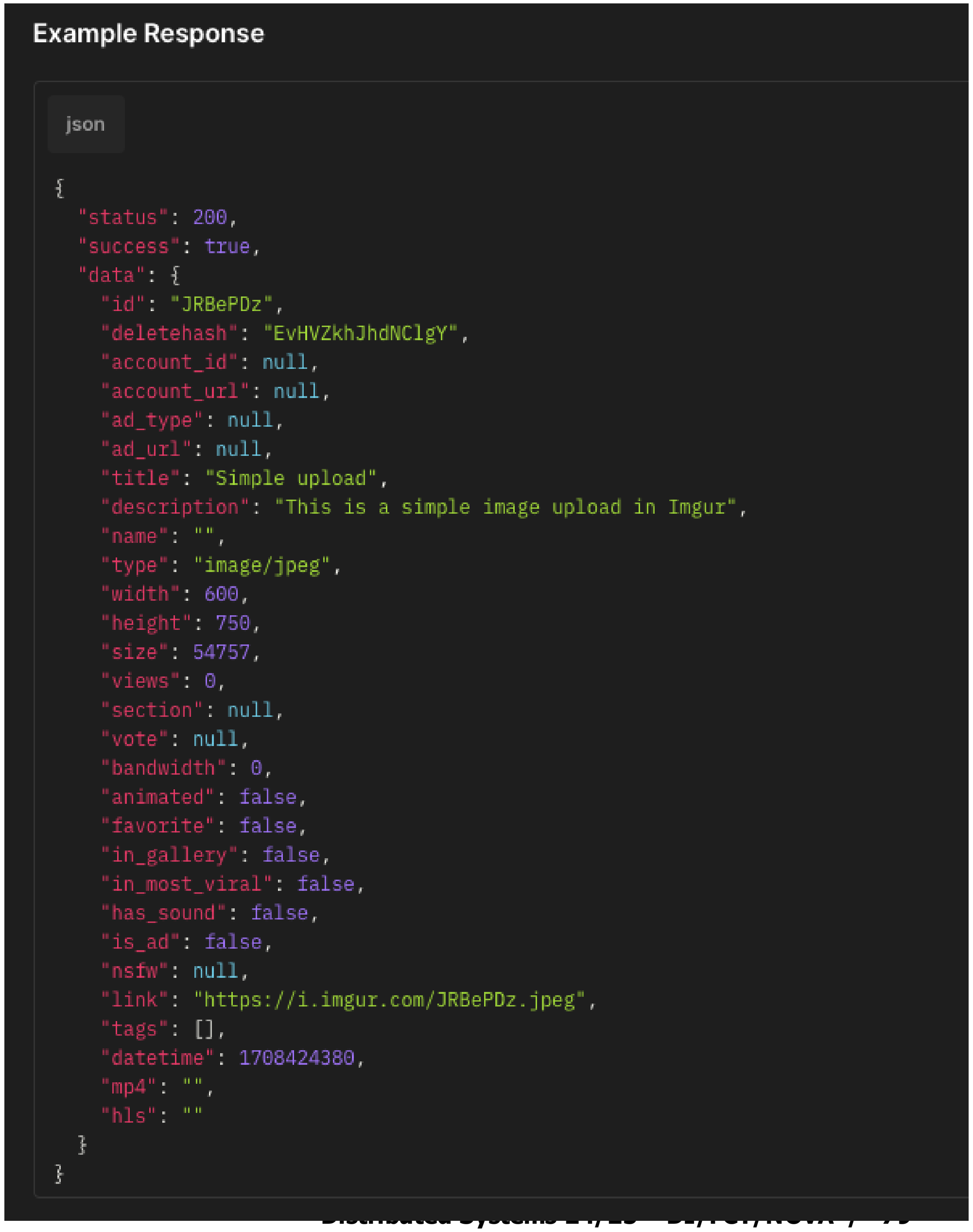
```java
public boolean execute(String imageName, byte[] data) {
    OAuthRequest request = new OAuthRequest(Verb.POST, UPLOAD_IMAGE_URL);

    request.addHeader(CONTENT_TYPE_HDR, JSON_CONTENT_TYPE);
    request.setPayload(json.toJson(new ImageUploadArguments(data, imageName)));

    service.signRequest(accessToken, request);

    try {
        Response r = service.execute(request);

        if(r.getCode() != HTTP_SUCCESS) {
            //Operation failed
            System.err.println("Operation Failed\nStatus: " + r.getCode() + "\nBody: " + r.getBody());
            return false;
        } else {
            BasicResponse body = json.fromJson(r.getBody(), BasicResponse.class);
            System.out.println("Operation Succedded\nImage name: " + imageName + "\nImage ID: " + body.getData().get("id"));
            return body.isSuccess();
        }

    } catch (InterruptedException e) {
        e.printStackTrace();
        System.exit(1);
    } catch (ExecutionException e) {
        e.printStackTrace();
        System.exit(1);
    } catch (IOException e) {
        e.printStackTrace();
        System.exit(1);
    }

    return false;
}
```

If the operation is Successful we still convert the contents received in the body of the request to the BasicResponse class we used before.

OAuth Requests to Upload an Image in JAVA (3/3) – Response Format



```java
package lab9.imgur.data;

import java.util.Map;

public class BasicResponse {

    private Map<?,?> data;
    private int status;
    private boolean success;

    public Map<?,?> getData() {
        return data;
    }

    public void setData(Map<?,?> data) {
        this.data = data;
    }

    public int getStatus() {
        return status;
    }

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

    public boolean isSuccess() {
        return success;
    }

    public void setSuccess(boolean success) {
        this.success = success;
    }

}
```

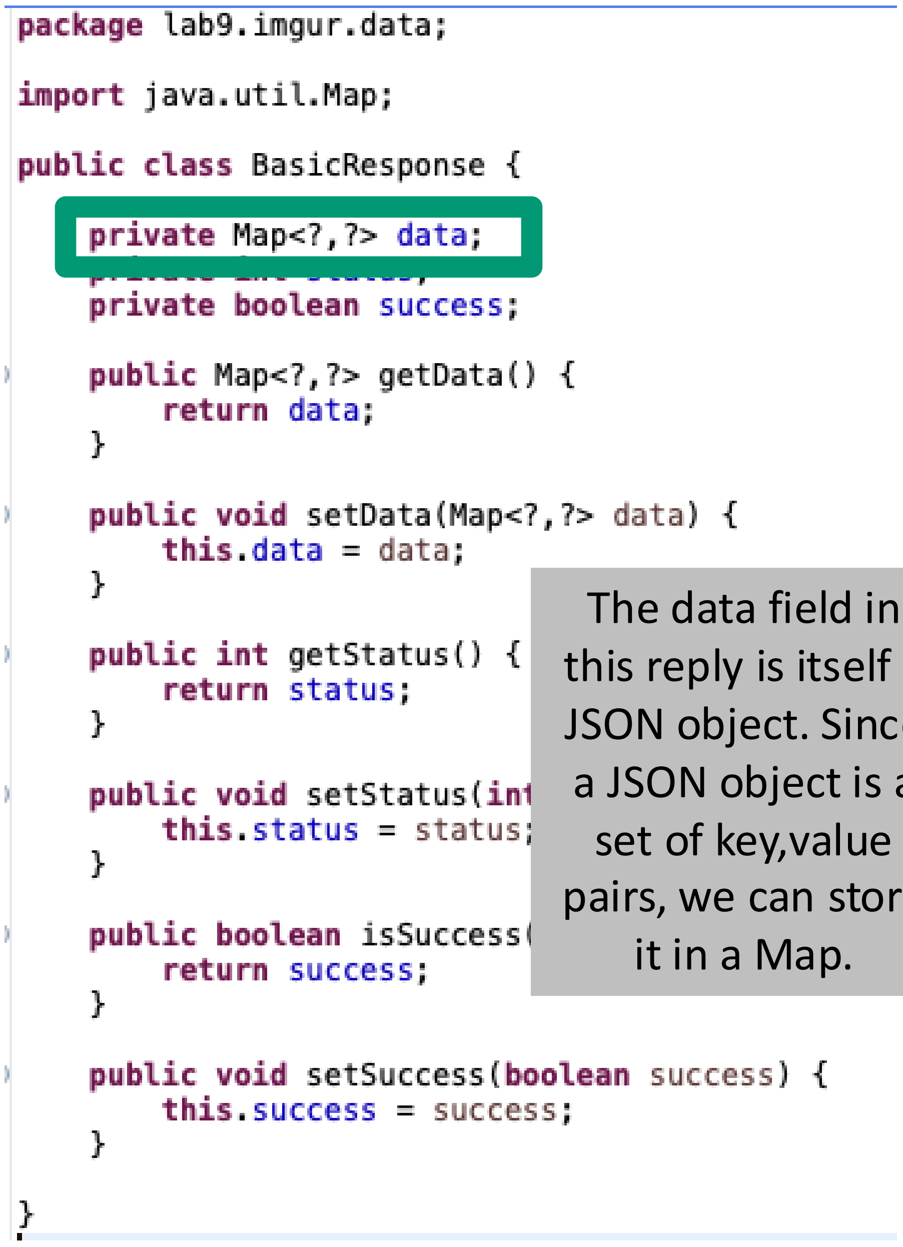


Example Response

json

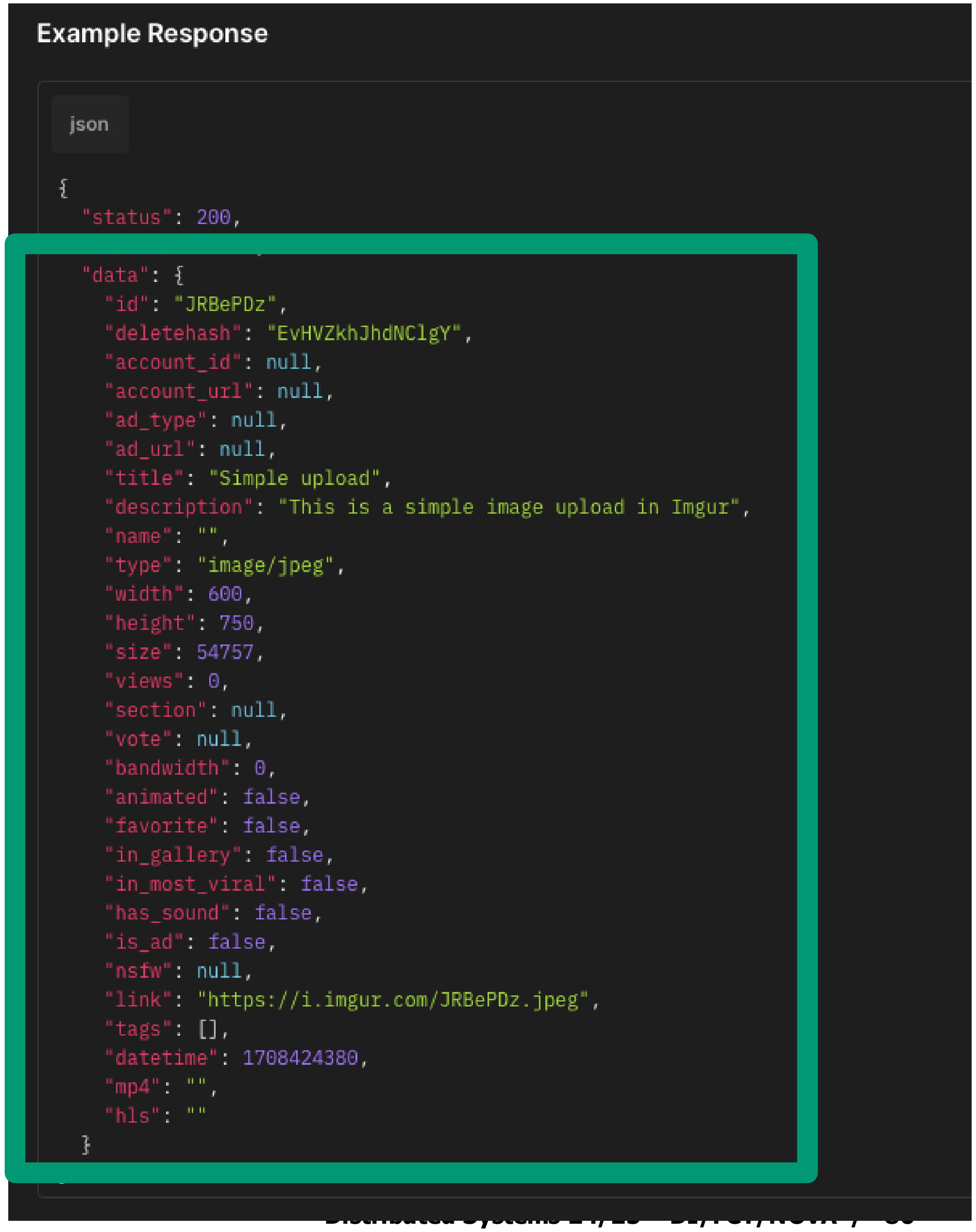
```json
{
  "status": 200,
  "success": true,
  "data": {
    "id": "JRBePDz",
    "deletehash": "EvHVZkhJhdNClgY",
    "account_id": null,
    "account_url": null,
    "ad_type": null,
    "ad_url": null,
    "title": "Simple upload",
    "description": "This is a simple image upload in Imgur",
    "name": "",
    "type": "image/jpeg",
    "width": 600,
    "height": 750,
    "size": 54757,
    "views": 0,
    "section": null,
    "vote": null,
    "bandwidth": 0,
    "animated": false,
    "favorite": false,
    "in_gallery": false,
    "in_most_viral": false,
    "has_sound": false,
    "is_ad": false,
    "nsfw": null,
    "link": "https://i.imgur.com/JRBePDz.jpeg",
    "tags": [],
    "datetime": 1708424380,
    "mp4": "",
    "hls": ""
  }
}
```

OAuth Requests to Upload an Image in JAVA (3/3) – Response Format



```java
package lab9.imgur.data;

import java.util.Map;

public class BasicResponse {

    private Map<?,?> data;
    private int status;
    private boolean success;

    public Map<?,?> getData() {
        return data;
    }

    public void setData(Map<?,?> data) {
        this.data = data;
    }

    public int getStatus() {
        return status;
    }

    public void setStatus(int
        this.status = status;
    }

    public boolean isSuccess(
        return success;
    }

    public void setSuccess(boolean success) {
        this.success = success;
    }

}
```

The data field in this reply is itself a JSON object. Since a JSON object is a set of key,value pairs, we can store it in a Map.



Example Response

json

```json
{
  "status": 200,
  "data": {
    "id": "JRBePDz",
    "deletehash": "EvHVZkhJhdNClgY",
    "account_id": null,
    "account_url": null,
    "ad_type": null,
    "ad_url": null,
    "title": "Simple upload",
    "description": "This is a simple image upload in Imgur",
    "name": "",
    "type": "image/jpeg",
    "width": 600,
    "height": 750,
    "size": 54757,
    "views": 0,
    "section": null,
    "vote": null,
    "bandwidth": 0,
    "animated": false,
    "favorite": false,
    "in_gallery": false,
    "in_most_viral": false,
    "has_sound": false,
    "is_ad": false,
    "nsfw": null,
    "link": "https://i.imgur.com/JRBePDz.jpeg",
    "tags": [],
    "datetime": 1708424380,
    "mp4": "",
    "hls": ""
  }
```

OAUTH REQUESTS TO UPLOAD AN IMAGE IN JAVA (3/3)

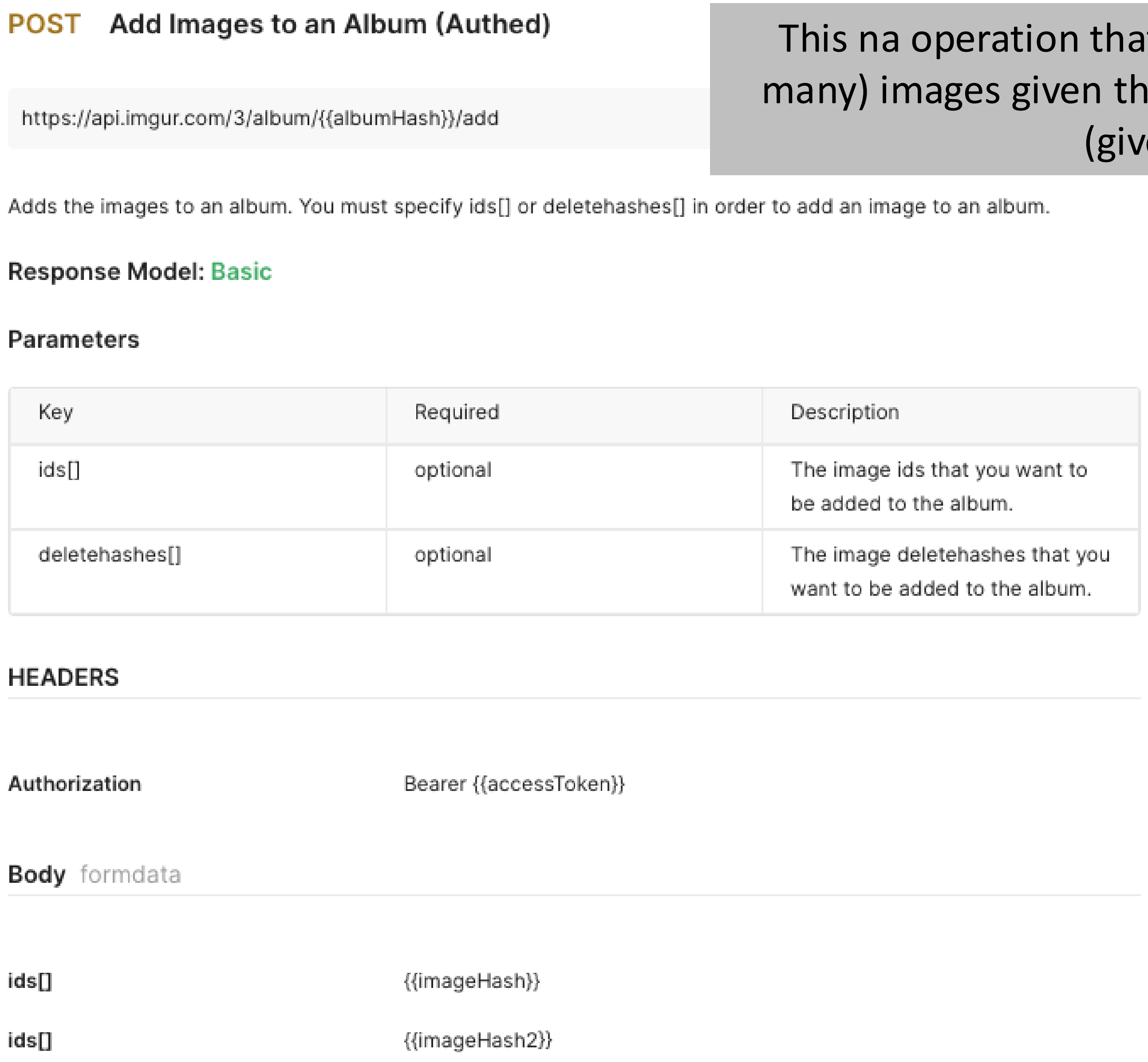
Class ImageUpload (oauth request execution)

```java
public boolean execute(String imageName, byte[] data) {
    OAuthRequest request = new OAuthRequest(Verb.POST, UPLOAD_IMAGE_URL);

    request.addHeader(CONTENT_TYPE_HDR, JSON_CONTENT_TYPE);
    request.setPayload(json.toJson(new ImageUploadArguments(data, imageName)));

    service.signRequest(accessToken, request);

    try {
        Response r = service.execute(request);

        if(r.getCode() != HTTP_SUCCESS) {
            //Operation failed
            System.err.println("Operation Failed\nStatus: " + r.getCode() + "\nBody: " + r.getBody());
            return false;
        } else {
            BasicResponse body = json.fromJson(r.getBody(), BasicResponse.class);
            System.out.println("Operation Succedded\nImage name: " + imageName + "\nImage ID: " + body.getData().get("id"));
            return body.isSuccess();
        }

    } catch (InterruptedException e) {
        e.printStackTrace();
        System.exit(1);
    } catch (ExecutionException e) {
        e.printStackTrace();
        System.exit(1);
    } catch (IOException e) {
        e.printStackTrace();
        System.exit(1);
    }

    return false;
}
```

That is why here, to obtain the identifier of the image that was just created on Imgur we access the data field (which is a map) and then obtain the value associated with the key “id” (where “id” is the name of the field in the JSON object associated with the data field).

OAuth Requests to Imgur in JAVA
Associate Image to Album (documentation)

This na operation that allows you to add one (or many) images given their ids to a particular album (given its id).



POST **Add Images to an Album (Authed)**

```
https://api.imgur.com/3/album/{{albumHash}}/add
```

Adds the images to an album. You must specify ids[] or deletehashes[] in order to add an image to an album.

Response Model: Basic

Parameters

Key	Required	Description
ids[]	optional	The image ids that you want to be added to the album.
deletehashes[]	optional	The image deletehashes that you want to be added to the album.

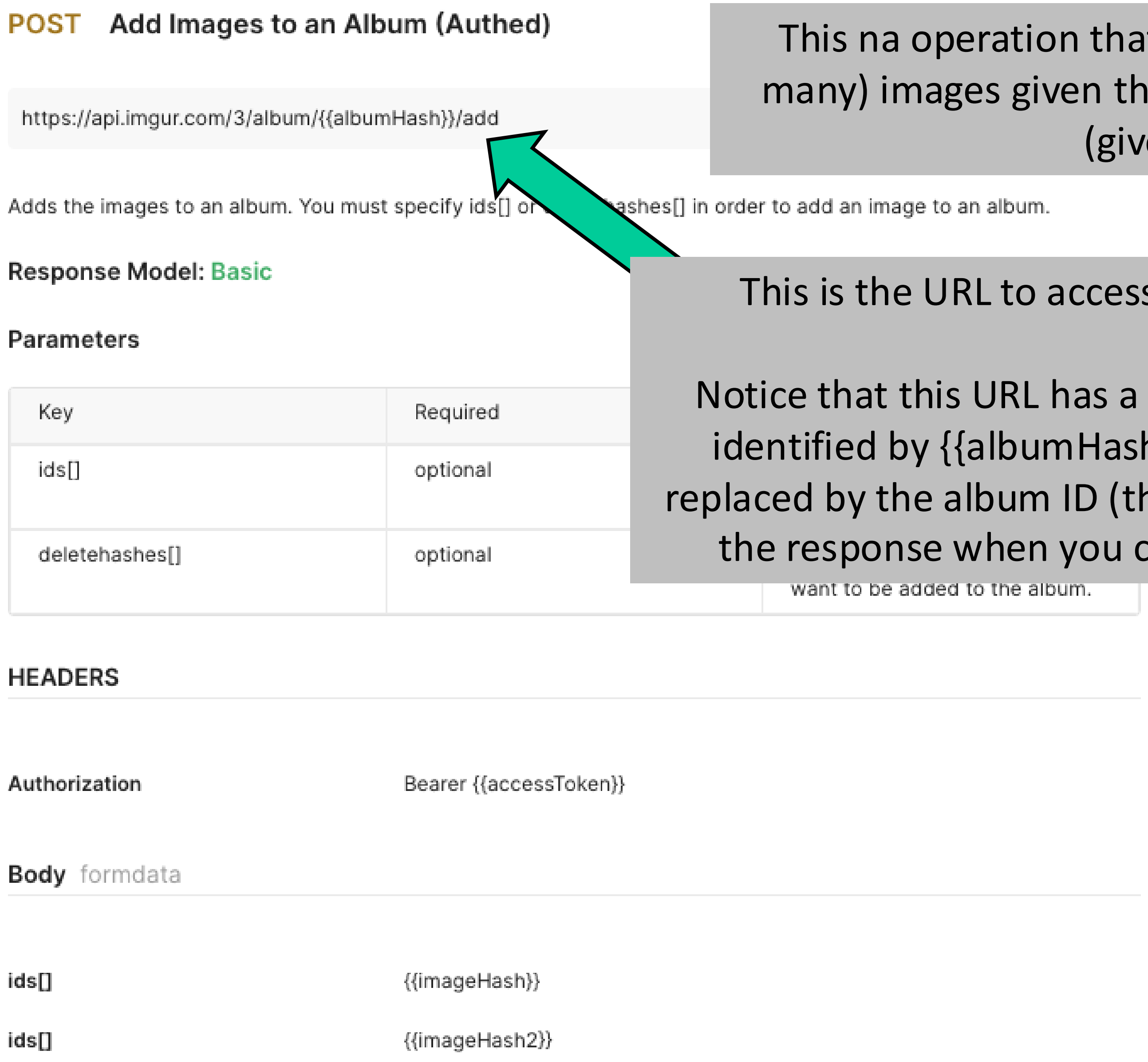
HEADERS

Authorization	Bearer {{accessToken}}

Body formdata

ids[]	{{imageHash}}
ids[]	{{imageHash2}}

OAUTH REQUESTS TO IMGUR IN JAVA
ASSOCIATE IMAGE TO ALBUM (DOCUMENTATION)



POST Add Images to an Album (Authed)

```
https://api.imgur.com/3/album/{{albumHash}}/add
```

Adds the images to an album. You must specify ids[] or ashes[] in order to add an image to an album.

Response Model: Basic

Parameters

Key	Required	
ids[]	optional	
deletehashes[]	optional	want to be added to the album.

HEADERS

Authorization	Bearer {{accessToken}}

Body formdata

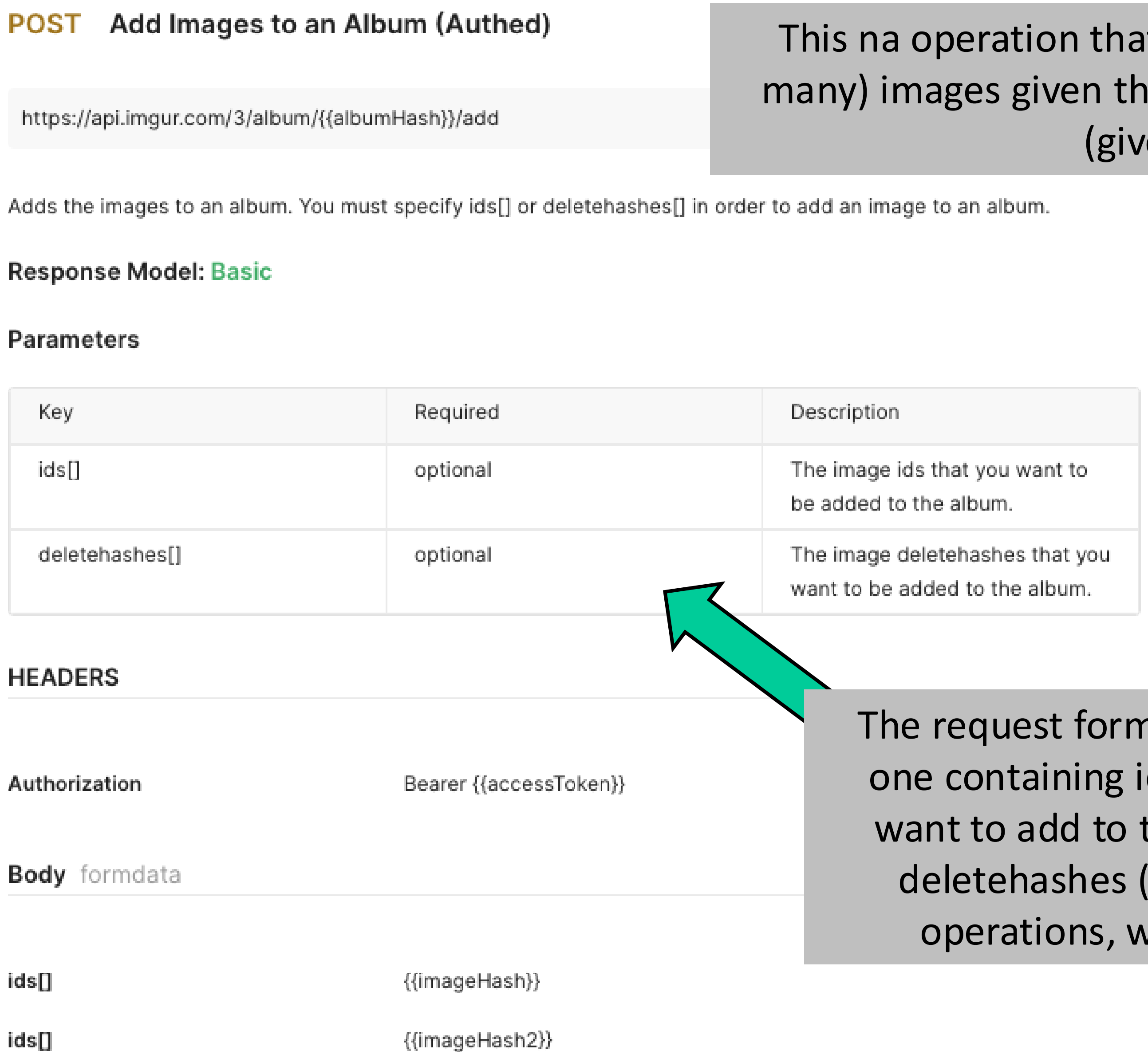
ids[]	{{imageHash}}
ids[]	{{imageHash2}}

This na operation that allows you to add one (or many) images given their ids to a particular album (given its id).

This is the URL to access this operation.

Notice that this URL has a variable argument identified by {{albumHash}} that should be replaced by the album ID (that you can get from the response when you create an Album).

OAUTH REQUESTS TO IMGUR IN JAVA
ASSOCIATE IMAGE TO ALBUM (DOCUMENTATION)



POST **Add Images to an Album (Authed)**

https://api.imgur.com/3/album/{{albumHash}}/add

Adds the images to an album. You must specify ids[] or deletehashes[] in order to add an image to an album.

Response Model: Basic

Parameters

Key	Required	Description
ids[]	optional	The image ids that you want to be added to the album.
deletehashes[]	optional	The image deletehashes that you want to be added to the album.

HEADERS

Authorization	Bearer {{accessToken}}

Body formdata

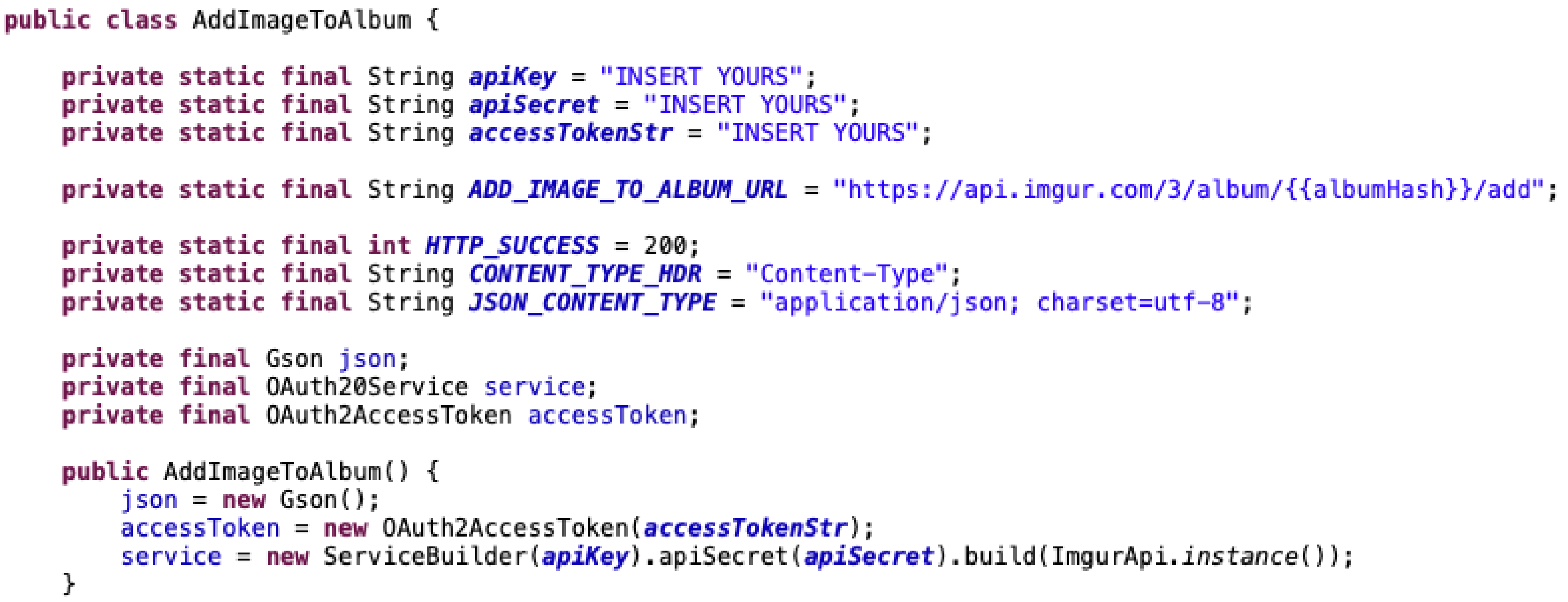
ids[]	{{imageHash}}
ids[]	{{imageHash2}}

This na operation that allows you to add one (or many) images given their ids to a particular album (given its id).

The request format is quite simple, two arrays one containing identifiers of images that you want to add to the album, and another with deletehashes (used for doing anonymous operations, we will not be using these)

OAUTH REQUESTS TO ADD IMAGE(S) TO AN ALBUM IN JAVA (1/3)

Class AddImageToAlbum (preamble and constructor)



```java
public class AddImageToAlbum {

    private static final String apiKey = "INSERT YOURS";
    private static final String apiSecret = "INSERT YOURS";
    private static final String accessTokenStr = "INSERT YOURS";

    private static final String ADD_IMAGE_TO_ALBUM_URL = "https://api.imgur.com/3/album/{{albumHash}}/add";

    private static final int HTTP_SUCCESS = 200;
    private static final String CONTENT_TYPE_HDR = "Content-Type";
    private static final String JSON_CONTENT_TYPE = "application/json; charset=utf-8";

    private final Gson json;
    private final OAuth20Service service;
    private final OAuth2AccessToken accessToken;

    public AddImageToAlbum() {
        json = new Gson();
        accessToken = new OAuth2AccessToken(accessTokenStr);
        service = new ServiceBuilder(apiKey).apiSecret(apiSecret).build(ImgurApi.instance());
    }
```

The constants and constructor in this example are again very similar to the previous examples.

OAUTH REQUESTS TO ADD IMAGE(S) TO AN ALBUM IN JAVA (1/3)

Class AddImageToAlbum (preamble and constructor)

```
public class AddImageToAlbum {

    private static final String apiKey = "INSERT YOURS";
    private static final String apiSecret = "INSERT YOURS";
    private static final String accessTokenStr = "INSERT YOURS";

    private static final String ADD_IMAGE_TO_ALBUM_URL = "https://api.imgur.com/3/album/{{albumHash}}/add";

    private static final int HTTP_SUCCESS = 200;
    private static final String CONTENT_TYPE_HDR = "Content-Type";
    private static final String JSON_CONTENT_TYPE = "application/json; charset=utf-8";

    private final Gson json;
    private final OAuth20Service service;
    private final OAuth2AccessToken accessToken;

    public AddImageToAlbum() {
        json = new Gson();
        accessToken = new OAuth2AccessToken(accessTokenStr);
        service = new ServiceBuilder(apiKey).apiSecret(apiSecret).build(ImgurApi.instance());
    }
```

The URL is different (which is expected) and has a variable {{albumHash}} that has to be replaced before executing the operation.

The constants and constructor in this example are again very similar to the previous examples.

OAUTH REQUESTS TO ADD IMAGE(S) TO AN ALBUM IN JAVA (2/3)

Class AddImageToAlbum (main -- basic user interface)

```java
public static void main(String[] args) throws Exception {

    if( args.length != 2 ) {
        System.err.println("usage: java " + AddImageToAlbum.class.getCanonicalName() +  " <album-id> <image-id>");
        System.exit(0);
    }

    String albumId = args[0];
    String imageId = args[1];
    AddImageToAlbum ca = new AddImageToAlbum();

    if(ca.execute(albumId, imageId))
        System.out.println("Added " + imageId + " to album " + albumId + " successfuly.");
    else
        System.err.println("Failed to execute operation");
}
```

The Main is also similar to the previous examples. We now expect to receive two arguments from the command line: the album id and the image id to be added to the album.

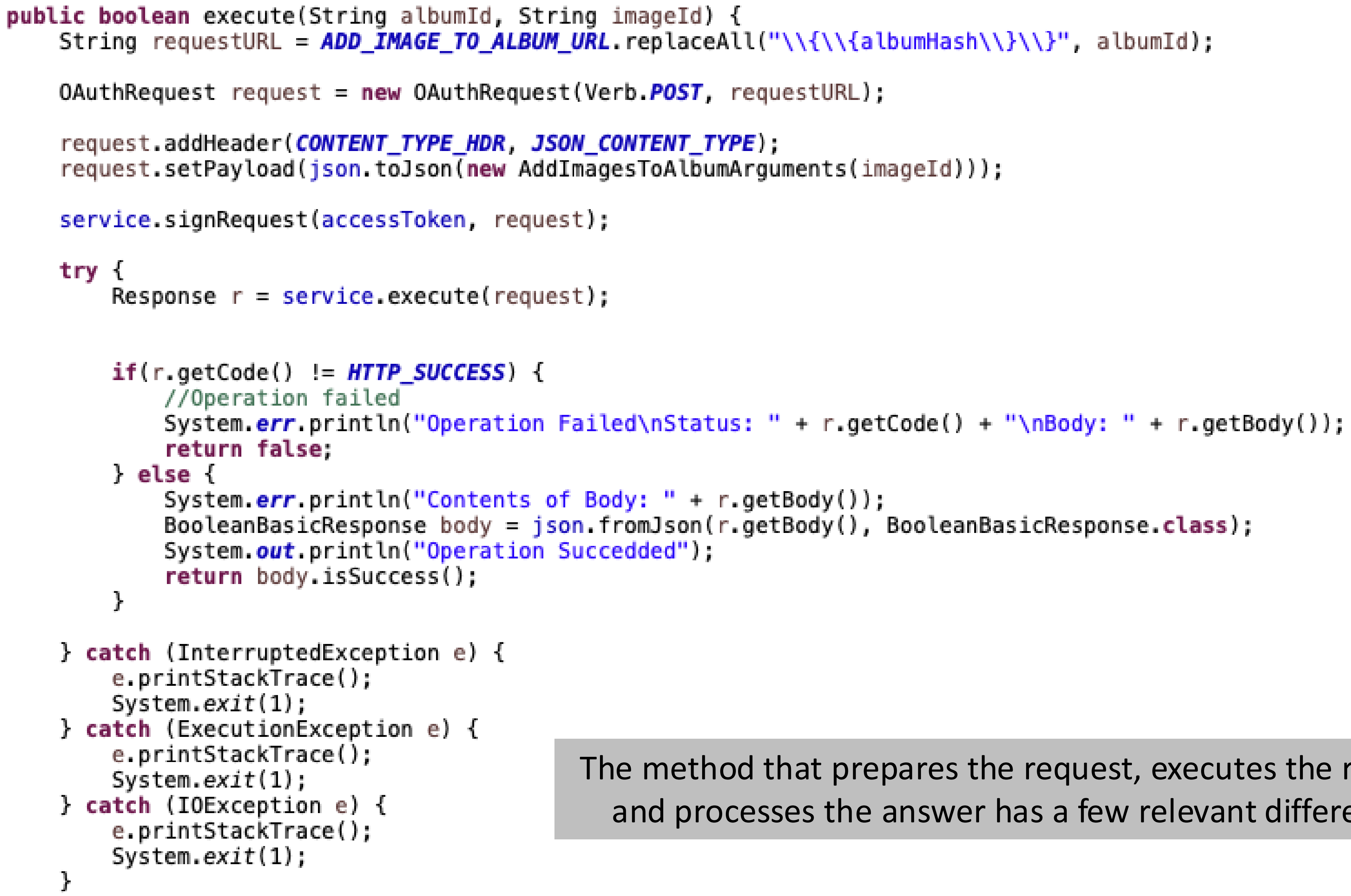
OAUTH REQUESTS TO ADD IMAGE(S) TO AN ALBUM IN JAVA (3/3)

Class AddImageToAlbum (oauth request execution)

```
public boolean execute(String albumId, String imageId) {
    String requestURL = ADD_IMAGE_TO_ALBUM_URL.replaceAll("\\{\\{albumHash\\}\\}", albumId);

    OAuthRequest request = new OAuthRequest(Verb.POST, requestURL);

    request.addHeader(CONTENT_TYPE_HDR, JSON_CONTENT_TYPE);
    request.setPayload(json.toJson(new AddImagesToAlbumArguments(imageId)));

    service.signRequest(accessToken, request);

    try {
        Response r = service.execute(request);


        if(r.getCode() != HTTP_SUCCESS) {
            //Operation failed
            System.err.println("Operation Failed\nStatus: " + r.getCode() + "\nBody: " + r.getBody());
            return false;
        } else {
            System.err.println("Contents of Body: " + r.getBody());
            BooleanBasicResponse body = json.fromJson(r.getBody(), BooleanBasicResponse.class);
            System.out.println("Operation Succedded");
            return body.isSuccess();
        }

    } catch (InterruptedException e) {
        e.printStackTrace();
        System.exit(1);
    } catch (ExecutionException e) {
        e.printStackTrace();
        System.exit(1);
    } catch (IOException e) {
        e.printStackTrace();
        System.exit(1);
    }
```

OAUTH REQUESTS TO ADD IMAGE(S) TO AN ALBUM IN JAVA (3/3)

Class AddImageToAlbum (oauth request execution)

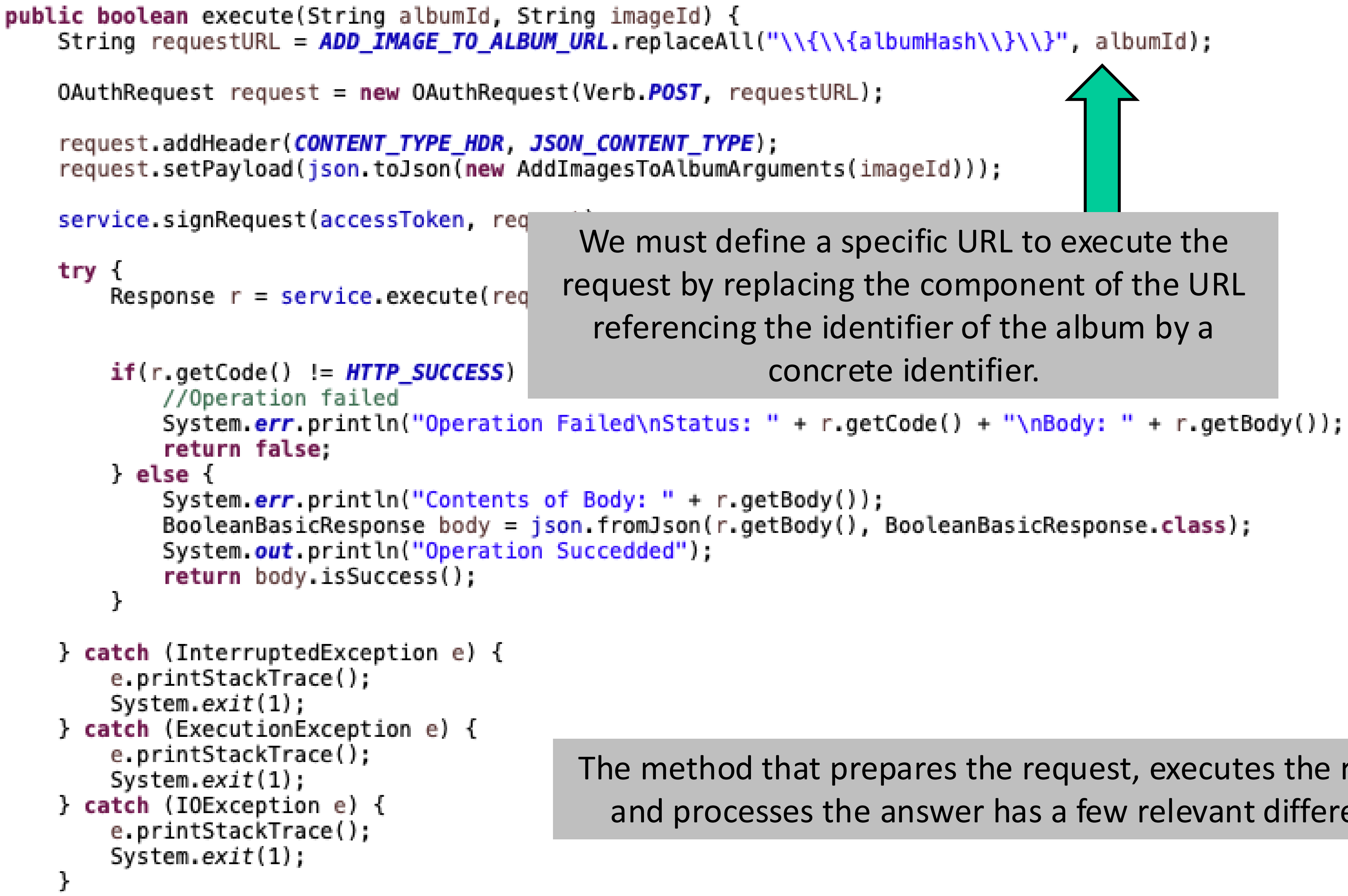


```java
public boolean execute(String albumId, String imageId) {
    String requestURL = ADD_IMAGE_TO_ALBUM_URL.replaceAll("\\{\\{albumHash\\}\\}", albumId);

    OAuthRequest request = new OAuthRequest(Verb.POST, requestURL);

    request.addHeader(CONTENT_TYPE_HDR, JSON_CONTENT_TYPE);
    request.setPayload(json.toJson(new AddImagesToAlbumArguments(imageId)));

    service.signRequest(accessToken, request);

    try {
        Response r = service.execute(request);


        if(r.getCode() != HTTP_SUCCESS) {
            //Operation failed
            System.err.println("Operation Failed\nStatus: " + r.getCode() + "\nBody: " + r.getBody());
            return false;
        } else {
            System.err.println("Contents of Body: " + r.getBody());
            BooleanBasicResponse body = json.fromJson(r.getBody(), BooleanBasicResponse.class);
            System.out.println("Operation Succedded");
            return body.isSuccess();
        }

    } catch (InterruptedException e) {
        e.printStackTrace();
        System.exit(1);
    } catch (ExecutionException e) {
        e.printStackTrace();
        System.exit(1);
    } catch (IOException e) {
        e.printStackTrace();
        System.exit(1);
    }
```

The method that prepares the request, executes the request, and processes the answer has a few relevant differentes.

OAUTH REQUESTS TO ADD IMAGE(S) TO AN ALBUM IN JAVA (3/3)

Class AddImageToAlbum (oauth request execution)



```
public boolean execute(String albumId, String imageId) {
    String requestURL = ADD_IMAGE_TO_ALBUM_URL.replaceAll("\\{\\{albumHash\\}\\}", albumId);

    OAuthRequest request = new OAuthRequest(Verb.POST, requestURL);

    request.addHeader(CONTENT_TYPE_HDR, JSON_CONTENT_TYPE);
    request.setPayload(json.toJson(new AddImagesToAlbumArguments(imageId)));

    service.signRequest(accessToken, req

    try {
        Response r = service.execute(req


        if(r.getCode() != HTTP_SUCCESS)
            //Operation failed
            System.err.println("Operation Failed\nStatus: " + r.getCode() + "\nBody: " + r.getBody());
            return false;
        } else {
            System.err.println("Contents of Body: " + r.getBody());
            BooleanBasicResponse body = json.fromJson(r.getBody(), BooleanBasicResponse.class);
            System.out.println("Operation Succedded");
            return body.isSuccess();
        }

    } catch (InterruptedException e) {
        e.printStackTrace();
        System.exit(1);
    } catch (ExecutionException e) {
        e.printStackTrace();
        System.exit(1);
    } catch (IOException e) {
        e.printStackTrace();
        System.exit(1);
    }
```

We must define a specific URL to execute the request by replacing the component of the URL referencing the identifier of the album by a concrete identifier.

The method that prepares the request, executes the request, and processes the answer has a few relevant differentes.

OAUTH REQUESTS TO ADD IMAGE(S) TO AN ALBUM IN JAVA (3/3)

Class AddImageToAlbum (oauth request execution)

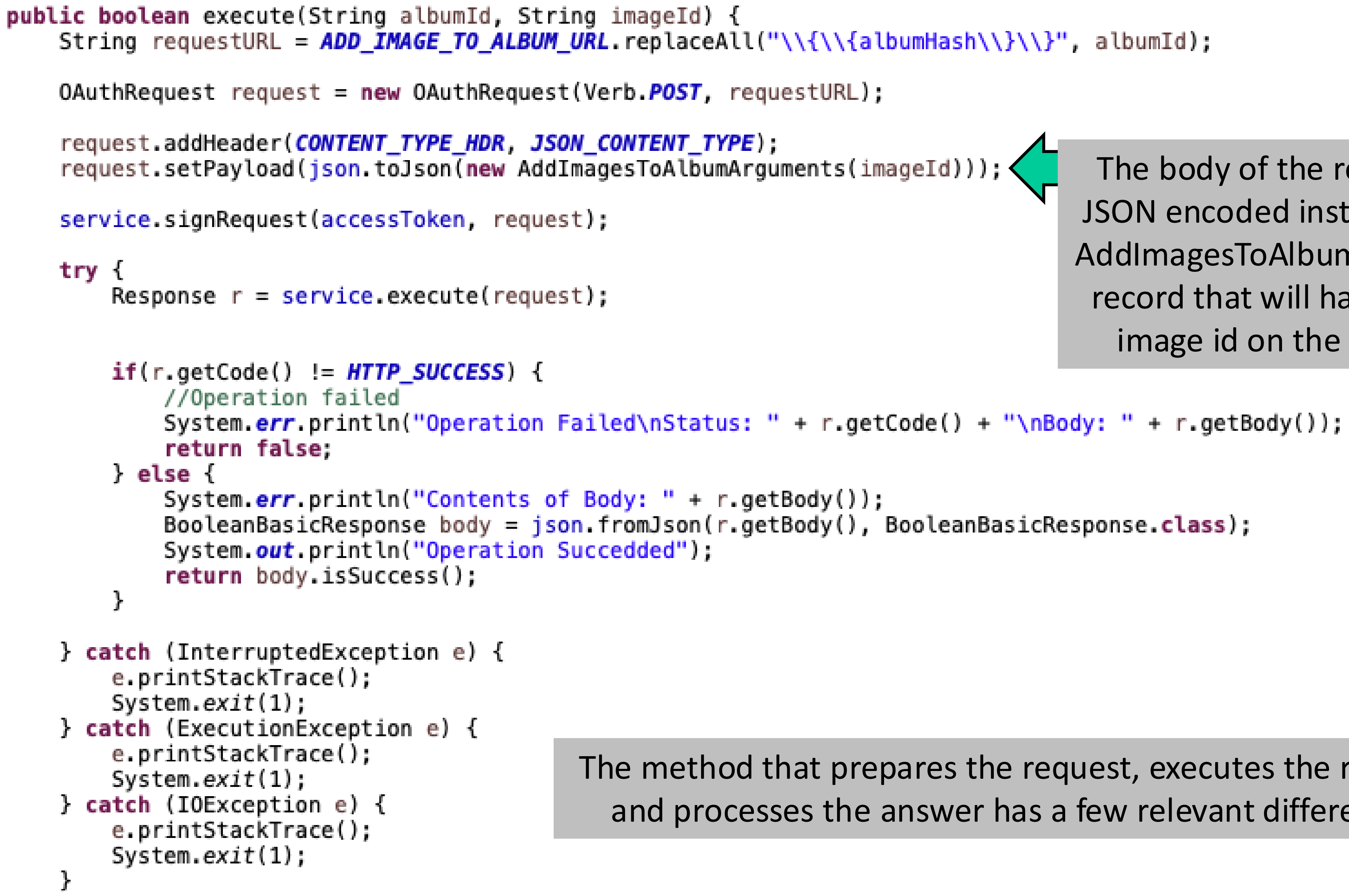
```java
public boolean execute(String albumId, String imageId) {
    String requestURL = ADD_IMAGE_TO_ALBUM_URL.replaceAll("\\{\\{albumHash\\}\\}", albumId);

    OAuthRequest request = new OAuthRequest(Verb.POST, requestURL);

    request.addHeader(CONTENT_TYPE_HDR, JSON_CONTENT_TYPE);
    request.setPayload(json.toJson(new AddImagesToAlbumArguments(imageId)));

    service.signRequest(accessToken, request);

    try {
        Response r = service.execute(request);


        if(r.getCode() != HTTP_SUCCESS) {
            //Operation failed
            System.err.println("Operation Failed\nStatus: " + r.getCode() + "\nBody: " + r.getBody());
            return false;
        } else {
            System.err.println("Contents of Body: " + r.getBody());
            BooleanBasicResponse body = json.fromJson(r.getBody(), BooleanBasicResponse.class);
            System.out.println("Operation Succedded");
            return body.isSuccess();
        }

    } catch (InterruptedException e) {
        e.printStackTrace();
        System.exit(1);
    } catch (ExecutionException e) {
        e.printStackTrace();
        System.exit(1);
    } catch (IOException e) {
        e.printStackTrace();
        System.exit(1);
    }
```

We still need to add this header to indicate that the body of the request is encoded in JSON

The method that prepares the request, executes the request, and processes the answer has a few relevant differentes.

OAUTH REQUESTS TO ADD IMAGE(S) TO AN ALBUM IN JAVA (3/3)

Class AddImageToAlbum (oauth request execution)



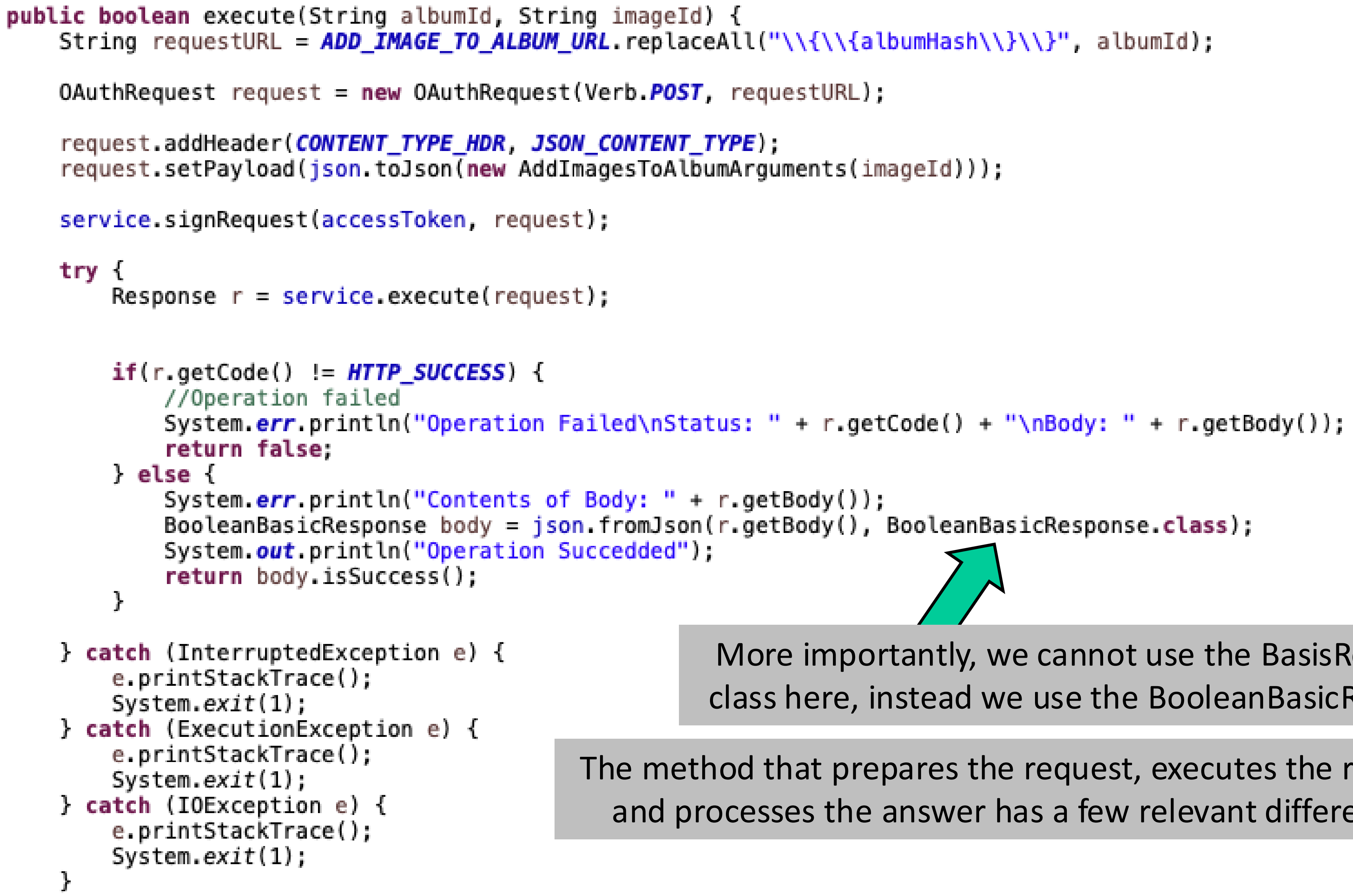
```java
public boolean execute(String albumId, String imageId) {
    String requestURL = ADD_IMAGE_TO_ALBUM_URL.replaceAll("\\{\\{albumHash\\}\\}", albumId);

    OAuthRequest request = new OAuthRequest(Verb.POST, requestURL);

    request.addHeader(CONTENT_TYPE_HDR, JSON_CONTENT_TYPE);
    request.setPayload(json.toJson(new AddImagesToAlbumArguments(imageId)));

    service.signRequest(accessToken, request);

    try {
        Response r = service.execute(request);


        if(r.getCode() != HTTP_SUCCESS) {
            //Operation failed
            System.err.println("Operation Failed\nStatus: " + r.getCode() + "\nBody: " + r.getBody());
            return false;
        } else {
            System.err.println("Contents of Body: " + r.getBody());
            BooleanBasicResponse body = json.fromJson(r.getBody(), BooleanBasicResponse.class);
            System.out.println("Operation Succedded");
            return body.isSuccess();
        }

    } catch (InterruptedException e) {
        e.printStackTrace();
        System.exit(1);
    } catch (ExecutionException e) {
        e.printStackTrace();
        System.exit(1);
    } catch (IOException e) {
        e.printStackTrace();
        System.exit(1);
    }
```

The body of the request is a JSON encoded instance of the AddImagesToAlbumArguments record that will have a single image id on the ids array.

The method that prepares the request, executes the request, and processes the answer has a few relevant differentes.

OAUTH REQUESTS TO ADD IMAGE(S) TO AN ALBUM IN JAVA (3/3)

Class AddImageToAlbum (oauth request execution)

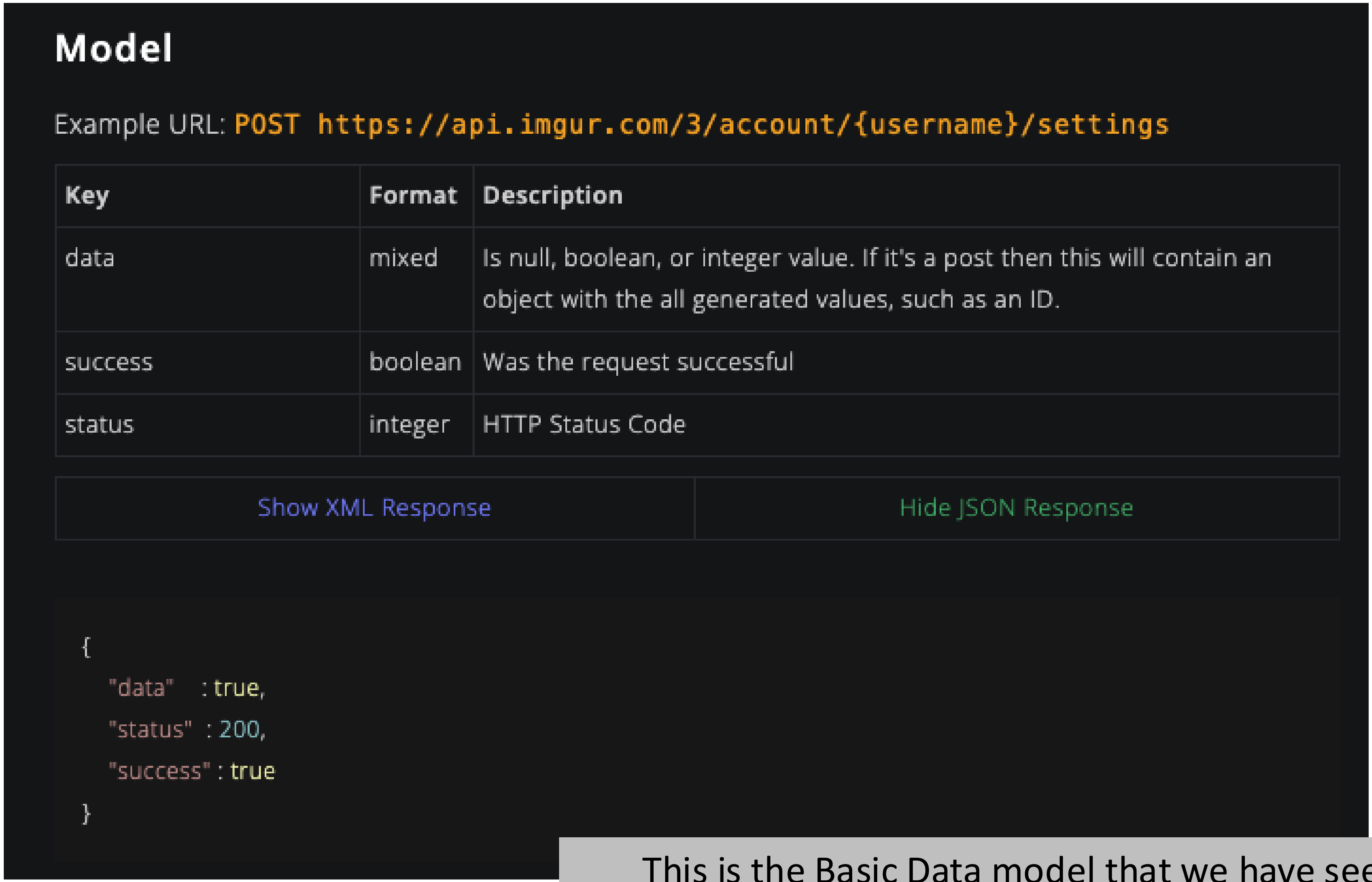


```java
public boolean execute(String albumId, String imageId) {
    String requestURL = ADD_IMAGE_TO_ALBUM_URL.replaceAll("\\{\\{albumHash\\}\\}", albumId);

    OAuthRequest request = new OAuthRequest(Verb.POST, requestURL);

    request.addHeader(CONTENT_TYPE_HDR, JSON_CONTENT_TYPE);
    request.setPayload(json.toJson(new AddImagesToAlbumArguments(imageId)));

    service.signRequest(accessToken, request);

    try {
        Response r = service.execute(request);


        if(r.getCode() != HTTP_SUCCESS) {
            //Operation failed
            System.err.println("Operation Failed\nStatus: " + r.getCode() + "\nBody: " + r.getBody());
            return false;
        } else {
            System.err.println("Contents of Body: " + r.getBody());
            BooleanBasicResponse body = json.fromJson(r.getBody(), BooleanBasicResponse.class);
            System.out.println("Operation Succedded");
            return body.isSuccess();
        }

    } catch (InterruptedException e) {
        e.printStackTrace();
        System.exit(1);
    } catch (ExecutionException e) {
        e.printStackTrace();
        System.exit(1);
    } catch (IOException e) {
        e.printStackTrace();
        System.exit(1);
    }
```

More importantly, we cannot use the BasisResponse class here, instead we use the BooleanBasicResponse

The method that prepares the request, executes the request, and processes the answer has a few relevant differentes.

OAuth Requests to Add Image(s) to an Album in JAVA – Response format (3/3)



Model

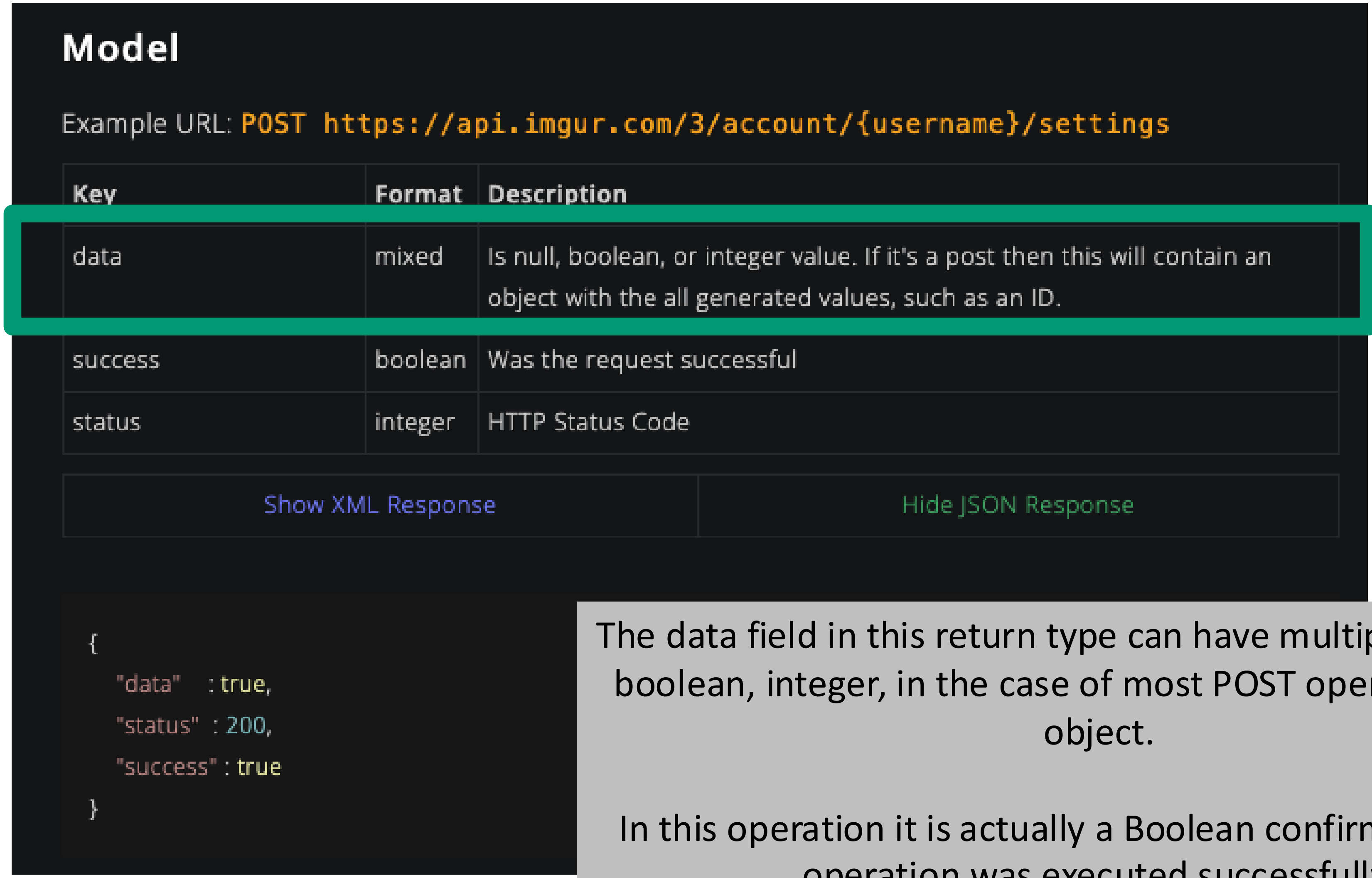
Example URL: POST https://api.imgur.com/3/account/{username}/settings

Key	Format	Description
data	mixed	Is null, boolean, or integer value. If it's a post then this will contain an object with the all generated values, such as an ID.
success	boolean	Was the request successful
status	integer	HTTP Status Code

```
{
   "data"    : true,
   "status"  : 200,
   "success" : true
}
```

This is the Basic Data model that we have seen before.

OAuth Requests to Add Image(s) to an Album in Java – Response format (3/3)



Model

Example URL: `POST https://api.imgur.com/3/account/{username}/settings`

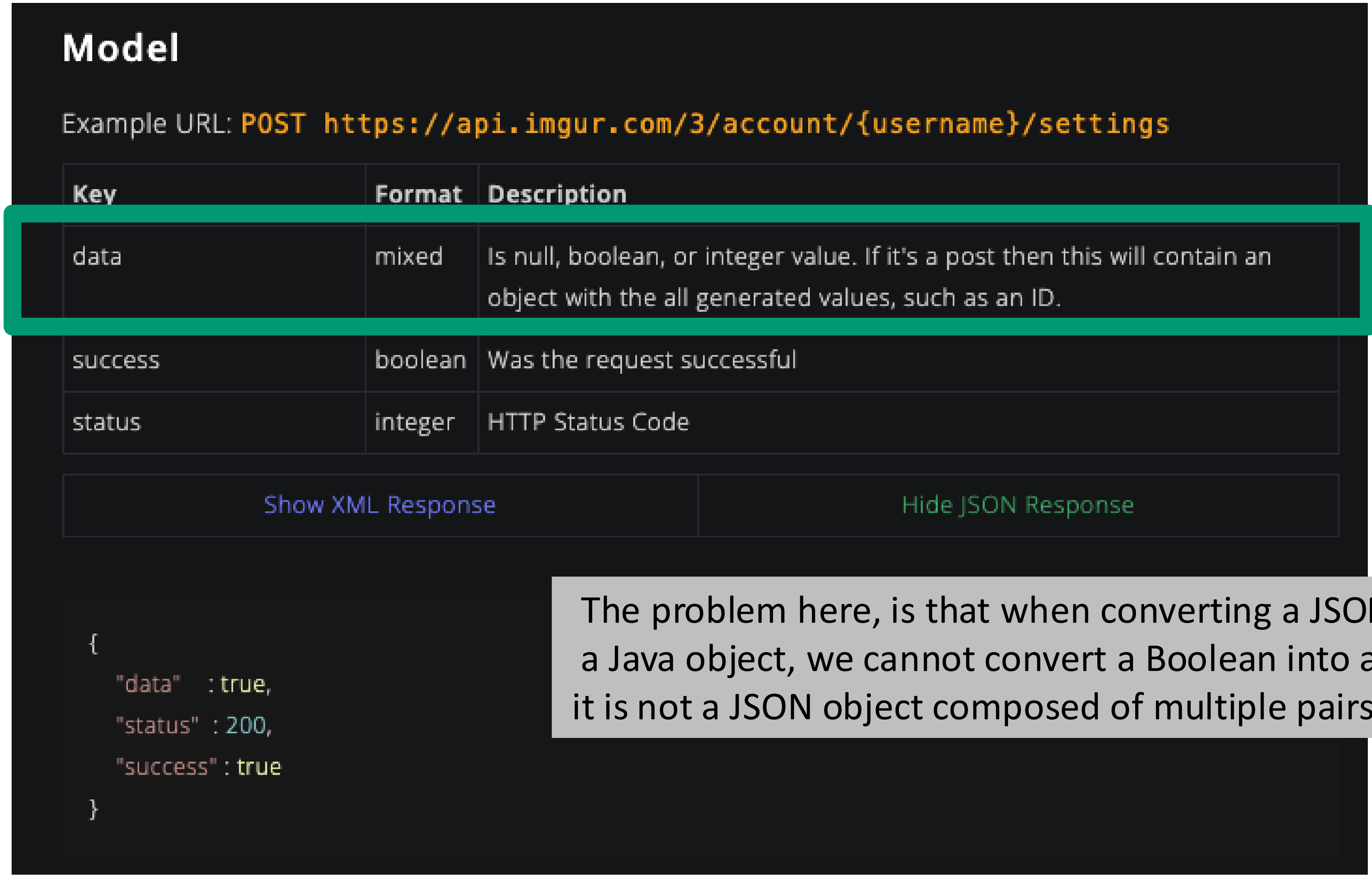
Key	Format	Description
data	mixed	Is null, boolean, or integer value. If it's a post then this will contain an object with the all generated values, such as an ID.
success	boolean	Was the request successful
status	integer	HTTP Status Code

```
{
   "data"    : true,
   "status"  : 200,
   "success" : true
}
```

The data field in this return type can have multiple types: null, boolean, integer, in the case of most POST operation a JSON object.

In this operation it is actually a Boolean confirming that the operation was executed successfully.

OAuth Requests to Add Image(s) to an Album in Java – Response format (3/3)



Model

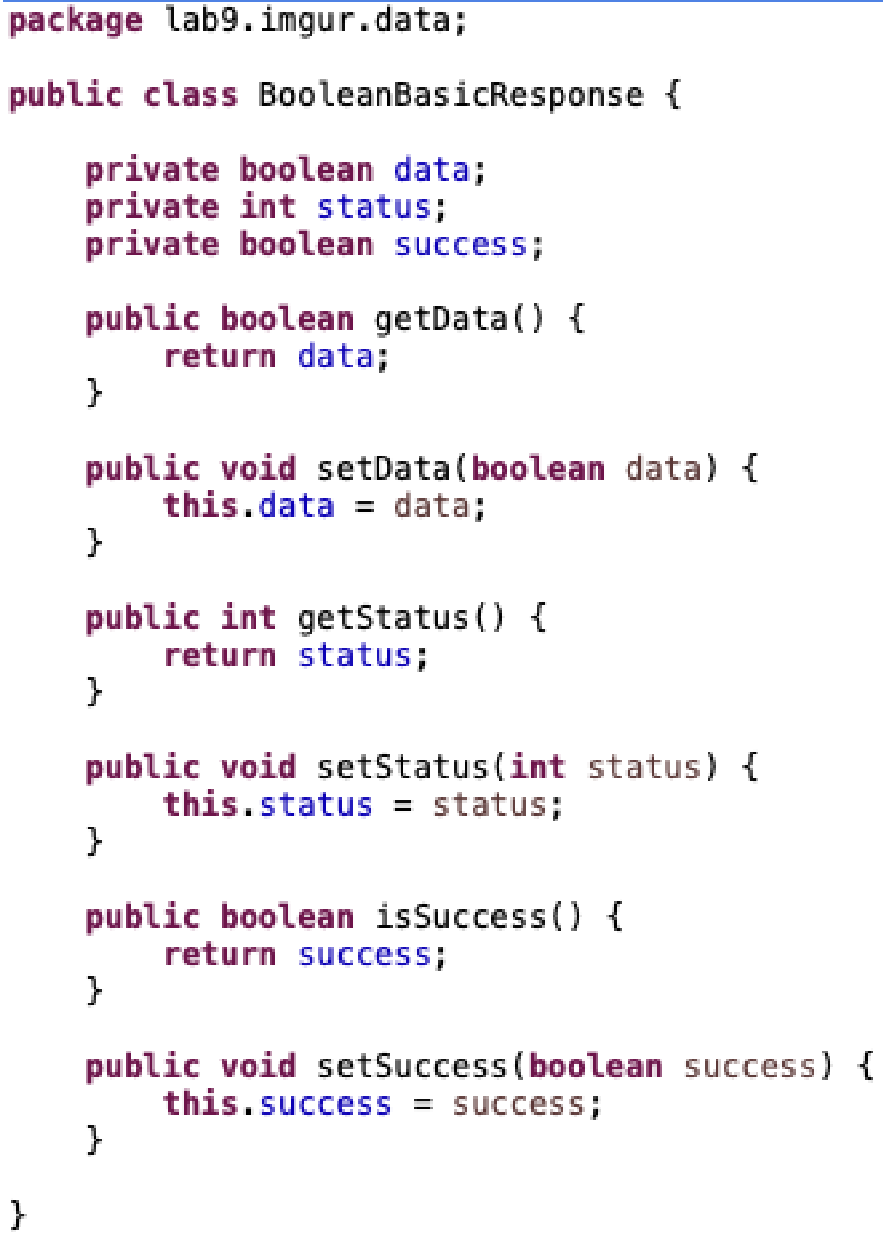
Example URL: `POST https://api.imgur.com/3/account/{username}/settings`

Key	Format	Description
data	mixed	Is null, boolean, or integer value. If it's a post then this will contain an object with the all generated values, such as an ID.
success	boolean	Was the request successful
status	integer	HTTP Status Code

```
{
   "data"    : true,
   "status"  : 200,
   "success" : true
}
```

The problem here, is that when converting a JSON objecto to a Java object, we cannot convert a Boolean into a Map (since it is not a JSON object composed of multiple pairs (key, value).

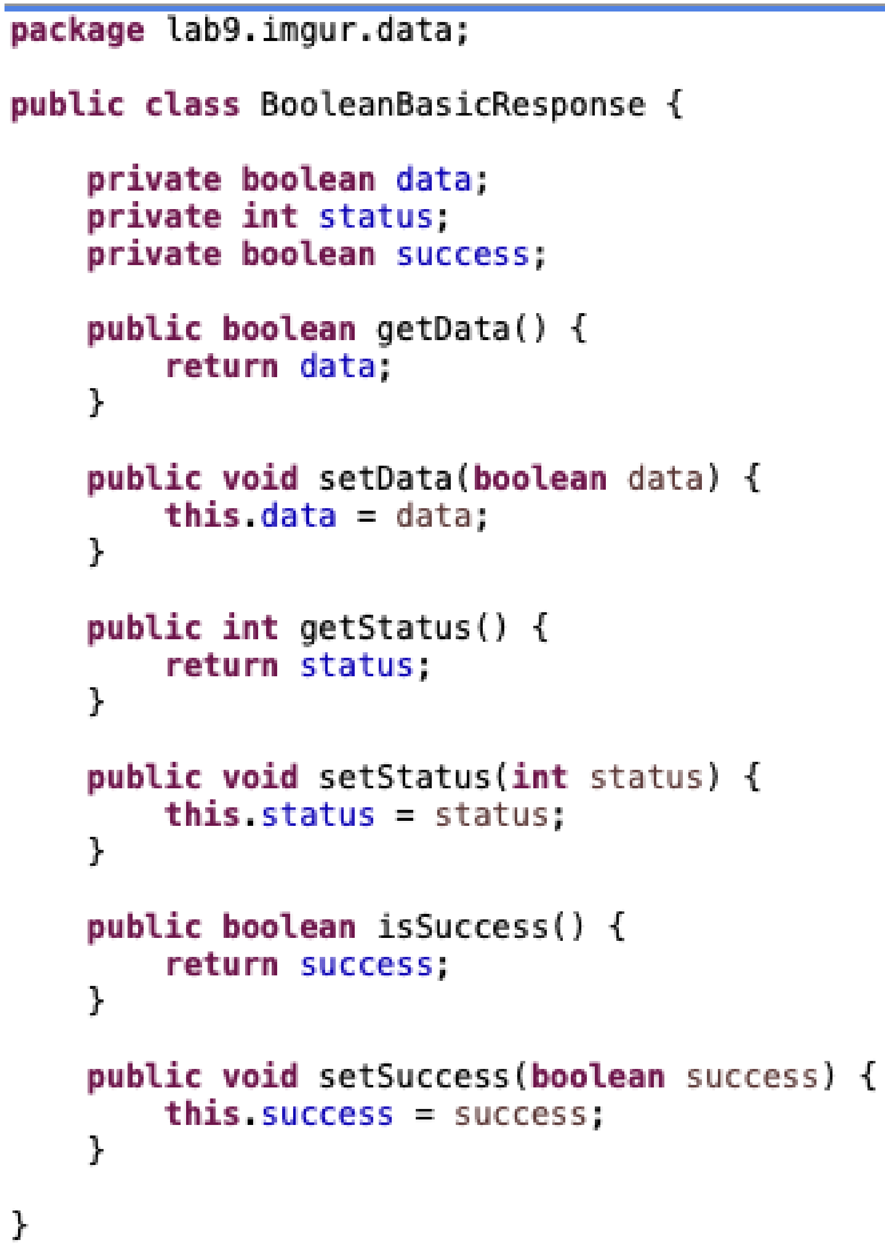
OAUTH REQUESTS TO ADD IMAGE(S) TO AN ALBUM IN JAVA – RESPONSE FORMAT (3/3)



```
package lab9.imgur.data;

public class BooleanBasicResponse {

    private boolean data;
    private int status;
    private boolean success;

    public boolean getData() {
        return data;
    }

    public void setData(boolean data) {
        this.data = data;
    }

    public int getStatus() {
        return status;
    }

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

    public boolean isSuccess() {
        return success;
    }

    public void setSuccess(boolean success) {
        this.success = success;
    }

}
```

To overcome this problem, we simply created another Java class (very similar to the BasicResponse) with the single difference that the data field now has the boolean type.

OAUTH REQUESTS TO ADD IMAGE(S) TO AN ALBUM IN JAVA – RESPONSE FORMAT (3/3)



```
package lab9.imgur.data;

public class BooleanBasicResponse {

    private boolean data;
    private int status;
    private boolean success;

    public boolean getData() {
        return data;
    }

    public void setData(boolean data) {
        this.data = data;
    }

    public int getStatus() {
        return status;
    }

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

    public boolean isSuccess() {
        return success;
    }

    public void setSuccess(boolean success) {
        this.success = success;
    }

}
```

To overcome this problem, we simply created another Java class (very similar to the BasicResponse) with the single difference that the data field now has the boolean type.

When using different endpoints in the Imgur API be careful about the body response format of those operations, and if required create new Java classes to simplify the processing of those responses.

EXERCISE

1. Create an Imgur account, register an application, generate your access Token and check that you can use the examples provided in this class (check that the effects of the operations become visible in Imgur).
2. Use what you have learned in this class to create a new version of the Image service from your project that instead of storing images in the local hard disk stores it in Imgur.
3. To allow multiple such servers to exist, it might be a good idea to store contents in a specific Album in imgur.

This is one of the mandatory aspects in the second project related with the interaction with an external service.

Exercise (Extra Details):

Implementation Suggestions:

- You can use an album with the name of the server (hostname), be careful, you have to check manually that there is not yet an Album with that name in Imgur.
- If you need to have additional information associated with images to simplify your work, you can use the description field of the image.
- It can be a good idea to simplify your implementation that the title of your image is the identifier of the image in your service.
- You will need to have a new Main class and implementation of the Resources/gRPC stub classes. These special (proxy) servers can expose either a REST or gRPC interface to your application end-clients (you can pick).
- These special proxy servers will never be replicated.