eBook Gratuit

APPRENEZ twitch

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#twitch

Table des matières

À propos

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: twitch

It is an unofficial and free twitch ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official twitch.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapitre 1: Commencer avec twitch

Versions

Version	Date de sortie
1.0.0	2016-04-14

Examples

Demander un jeton

Le flux de subvention implicite est le mieux adapté aux applications Web. Il s'intègre facilement à un site Web en utilisant JavaScript et ne nécessite pas de serveur pour stocker le code d'autorisation pour récupérer un jeton.

Vous allez d'abord envoyer l'utilisateur au noeud final d'autorisation Twitch. Cette URL est composée d'une URL d'autorisation de base (`https://api.twitch.tv/kraken/oauth2/authorize`) et de paramètres de chaîne de requête qui définissent ce que vous demandez. Les paramètres requis sont `response_type` , `client_id` , `redirect_uri` et `scope` .

Pour le flux de subvention implicite, le paramètre `response_type` est toujours défini sur `token` . Cela signifie que vous demandez directement un jeton OAuth.

`redirect_uri` est l'endroit où l'utilisateur sera redirigé après avoir approuvé les portées demandées par votre application. Cela doit correspondre à ce que vous avez enregistré sur la page Connexions de votre compte Twitch.

L' `client_id` est un identifiant unique pour votre application. Vous pouvez également trouver votre identifiant client sur la page Connexions.

Le paramètre `scope` définit ce à quoi vous avez accès au nom de l'utilisateur. Vous ne devez demander que le minimum requis pour que votre application fonctionne. Vous pouvez trouver la liste des étendues sur l' API Twitch GitHub .

Le paramètre `state` est également pris en charge pour vous protéger contre les attaques par script intersite. Lorsque l'utilisateur est redirigé après autorisation, cette valeur sera incluse dans le `redirect_uri` .

Rediriger l'utilisateur vers cette URL:

```
https://api.twitch.tv/kraken/oauth2/authorize
    ?response_type=token
    &client_id=[your client ID]
    &redirect_uri=[your registered redirect URI]
    &scope=[space separated list of scopes]
```

```
    &state=[your provided unique token]
```

Récupère le jeton OAuth à partir du fragment d'URL

Si l'utilisateur autorise votre application, elle sera redirigée vers l'URL suivante:

```
https://[your registered redirect URI]/#access_token=[an access token]
        &scope=[authorized scopes]
```

Notez que le jeton d'accès se trouve dans le fragment d'URL et non dans la chaîne de requête. Cela signifie que la valeur ne sera pas affichée dans les requêtes HTTP à votre serveur. Les fragments d'URL sont accessibles à partir de JavaScript avec `document.location.hash` .

Lire Commencer avec twitch en ligne: https://riptutorial.com/fr/twitch/topic/464/commencer-avec-twitch

Chapitre 2: Appel des API Twitch

Remarques

Cette rubrique est destinée à montrer une manière générale d'appeler l'API Twitch sans OAuth. Vous pouvez appeler toutes les API trouvées dans la documentation de l'API REST Twitch en utilisant ce modèle. Vous devez simplement modifier l'URL sur le point de terminaison correct.

Un ID client est requis pour tous les appels à l'API Twitch. Dans ces exemples, l'ID client est ajouté en tant qu'en-tête à chaque appel. Vous pouvez également l'ajouter avec le paramètre de chaîne de requête `client_id` . Si vous utilisez un jeton OAuth, l'API Twitch résoudra automatiquement l'ID client pour vous.

Vous pouvez enregistrer une application de développeur sur la nouvelle page client sur Twitch .

Examples

PHP

Ce qui suit récupérera un objet de `channel` pour le canal de `twitch` et répercutera la réponse.

```
$channelsApi = 'https://api.twitch.tv/kraken/channels/';
 $channelName = 'twitch';
 $clientId = '...';
 $ch = curl_init();

 curl_setopt_array($ch, array(
    CURLOPT_HTTPHEADER=> array(
    'Client-ID: ' . $clientId
    ),
    CURLOPT_RETURNTRANSFER=> true,
    CURLOPT_URL => $channelsApi . $channelName
 ));

 $response = curl_exec($ch);
 curl_close($ch);
 echo $response;
```

JavaScript

Ce qui suit enregistrera la réponse JSON de l'API à la console si la demande a réussi, sinon elle enregistrera l'erreur.

```
var xhr = new XMLHttpRequest();

xhr.open('GET', 'https://api.twitch.tv/kraken', true);

xhr.setRequestHeader('Client-ID', '...');

xhr.onload = function(data){
```

```
  console.log(data);
};

xhr.onerror = function(error){
  console.log(error.target.status);
};

xhr.send();
```

jQuery

Ce qui suit récupérera un objet de `channel` pour le canal de `twitch` . Si la requête a réussi, l'objet de `channel` sera enregistré sur la console.

```
$.ajax({
  type: 'GET',
  url: 'https://api.twitch.tv/kraken/channels/twitch',
  headers: {
    'Client-ID': '...'
  },
  success: function(data) {
    console.log(data);
  }
});
```

Lire Appel des API Twitch en ligne: https://riptutorial.com/fr/twitch/topic/760/appel-des-api-twitch

Chapitre 3: Lecteur vidéo interactif intégré

Examples

LIVE Streaming Video Player

Mise en œuvre de base:

```
<script src= "http://player.twitch.tv/js/embed/v1.js"></script>
<div id="PLAYER_DIV_ID"></div>
<script type="text/javascript">
    var options = {
        width: 854,
        height: 480,
        channel: "monstercat",
    };
    var player = new Twitch.Player("PLAYER_DIV_ID", options);
    player.setVolume(0.5);
</script>
```

Avec les contrôles cachés:

```
<script src= "http://player.twitch.tv/js/embed/v1.js"></script>
<div id="PLAYER_DIV_ID"></div>
<script type="text/javascript">
    var options = {
        width: 854,
        height: 480,
        channel: "monstercat",
        controls: false,
    };
    var player = new Twitch.Player("PLAYER_DIV_ID", options);
    player.setVolume(0.5);
</script>
```

Lecteur vidéo enregistré (pas en direct)

```
<script src= "http://player.twitch.tv/js/embed/v1.js"></script>
<div id="PLAYER_DIV_ID"></div>
<script type="text/javascript">
    var options = {
        width: 854,
        height: 480,
        video: "v53336925",
    };
    var player = new Twitch.Player("PLAYER_DIV_ID", options);
    player.setVolume(0.5);
</script>
```

L'extrait ci-dessus diffusera la vidéo suivante: twitch.tv/general_mittenz/ **v / 53336925**

Commencez avec un joueur en sourdine

```
<script src= "http://player.twitch.tv/js/embed/v1.js"></script>
<div id="{PLAYER_DIV_ID}"></div>
<script type="text/javascript">
    var options = {
        width: 854,
        height: 480,
        channel: "{CHANNEL}"
    };
    var player = new Twitch.Player("{PLAYER_DIV_ID}", options);
    player.setMuted(true);
</script>
```

Lire Lecteur vidéo interactif intégré en ligne: https://riptutorial.com/fr/twitch/topic/470/lecteur-video-interactif-integre

Chapitre 4: Listes des Streamers par jeu

Examples

Obtenir la première page en Ruby

Cet exemple Ruby utilise Mechanize , une bibliothèque pour automatiser les interactions Web.

`client_id` est un identifiant client OAuth.

`game` est le répertoire de jeu à lister.

```
require 'mechanize'
master_agent = Mechanize.new

client_id = "123"
game = "Minecraft"

url = "https://api.twitch.tv/kraken/streams?game=#{game}&client_id=#{client_id}"
final_list = []
master_agent.get(url) do |page|
    master_list = JSON.parse(page.body)
    master_list["streams"].each do |stream|
        final_list << stream["channel"]["name"]
    end
end
```

Lire Listes des Streamers par jeu en ligne: https://riptutorial.com/fr/twitch/topic/552/listes-des-streamers-par-jeu

Chapitre 5: Obtenir un jeton OAuth à l'aide du flux de code d'autorisation

Examples

Envoyer l'utilisateur au noeud final d'autorisation pour obtenir le code d'autorisation

Vous allez d'abord envoyer l'utilisateur au noeud final d'autorisation Twitch. Cette URL est composée d'une URL d'autorisation de base (`https://api.twitch.tv/kraken/oauth2/authorize`) et de paramètres de chaîne de requête qui définissent ce que vous demandez. Les paramètres requis sont `response_type` , `client_id` , `redirect_uri` et `scope` .

Pour le flux de code d'autorisation, le paramètre `response_type` est toujours défini sur `code` . Cela signifie que vous demandez un code d'autorisation à l'API Twitch.

`redirect_uri` est l'endroit où l'utilisateur sera redirigé après avoir approuvé les portées demandées par votre application. Cela doit correspondre à ce que vous avez enregistré sur la page Connexions de votre compte Twitch.

L' `client_id` est un identifiant unique pour votre application. Vous pouvez également trouver votre identifiant client sur la page Connexions.

La `scope` définit ce à quoi vous avez accès au nom de l'utilisateur. Vous ne devez demander que le minimum requis pour que votre application fonctionne. Vous pouvez trouver la liste des étendues sur l' API Twitch GitHub .

Le paramètre `state` est également pris en charge pour vous protéger contre les attaques par script intersite. Le paramètre `state` sera inclus dans le `redirect_uri` lorsque l'utilisateur autorise votre application.

```
https://api.twitch.tv/kraken/oauth2/authorize
  ?response_type=code
  &client_id=[your client ID]
  &redirect_uri=[your registered redirect URI]
  &scope=[space separated list of scopes]
  &state=[your provided unique token]
```

Récupère le code d'autorisation de la chaîne de requête

Lorsque l'utilisateur accède au point de terminaison d'autorisation, il lui est demandé d'autoriser votre application aux portées demandées. Ils peuvent refuser cela, vous devez donc vous assurer de prendre cela en considération dans votre code. Après avoir autorisé l'accès à votre application, l'utilisateur sera redirigé vers l'URL spécifiée dans `redirect_uri` . La chaîne de requête aura maintenant un paramètre de `code` , qui est le code d'autorisation que vous pouvez échanger contre un jeton OAuth.

```
<?php
  $authCode = $_GET['code'];
?>
```

Echangez le code pour le jeton OAuth

Maintenant que vous disposez d'un code d'autorisation, vous pouvez créer un POST sur le noeud final du jeton (`https://api.twitch.tv/kraken/oauth2/token`) pour obtenir un jeton OAuth. Vous recevrez un jeton d'accès codé JSON, un jeton d'actualisation et une liste des étendues approuvées par l'utilisateur. Vous pouvez maintenant utiliser ce jeton pour effectuer des demandes authentifiées au nom de l'utilisateur.

```
<?php
  $authCode = $_GET['code'];

  $parameterValues = array(
    'client_id' => '...',
    'client_secret' => '...',
    'grant_type' => 'authorization_code',
    'redirect_uri' => 'http://localhost/',
    'code' => $authCode
  );

  $postValues = http_build_query($parameterValues, '', '&');

  $ch = curl_init();

  curl_setopt_array($ch, array(
    CURLOPT_RETURNTRANSFER => true,
    CURLOPT_URL => 'https://api.twitch.tv/kraken/oauth2/token',
    CURLOPT_POST => 1,
    CURLOPT_POSTFIELDS => $postValues
  ));

  $response = curl_exec($ch);
  curl_close($ch);

  echo $response;
?>
```

Lire Obtenir un jeton OAuth à l'aide du flux de code d'autorisation en ligne: https://riptutorial.com/fr/twitch/topic/6624/obtenir-un-jeton-oauth-a-l-aide-du-flux-de-code-d-autorisation

Chapitre 6: Twitch Chat (IRC) Bot

Remarques

Twitch Chat est un simple chat IRC. Pour tout développement sérieux, il existe plusieurs documents, y compris la ressource la plus complète et générale: http://ircdocs.horse/

Connexion, poignée de main

IRC est un protocole TCP de base en clair. La connexion à Twitch fonctionne comme n'importe quel service IRC standard avec une authentification différente:

Initiation de la connexion> Handshake> Usage

La poignée de main est régulièrement la partie la plus difficile à corriger:

Après avoir établi la connexion au serveur, vous devez fournir `PASS` , **puis** un `NICK` , où `PASS` est un OAuth-Token (que vous pouvez générer ici) et `USER` est le nom d'utilisateur de ce jeton OAuth.

Le handshake est alors le suivant (`<` étant envoyé du client au serveur, `>` envoyé du serveur au client):

```
< PASS oauth:your_oauth_token
< NICK your_username
> :tmi.twitch.tv 001 your_username :connected to TMI
> :tmi.twitch.tv 002 your_username :your host is TMI
> :tmi.twitch.tv 003 your_username :this server is pretty new
> :tmi.twitch.tv 004 your_username tmi.twitch.tv 0.0.1 w n
> :tmi.twitch.tv 375 your_username :- tmi.twitch.tv Message of the day -
> :tmi.twitch.tv 372 your_username :- not much to say here
> :tmi.twitch.tv 376 your_username :End of /MOTD command
```

Une fois que vous avez reçu l'un de ces `MODE` , `376` ou `422` , vous êtes prêt à partir et pouvez envoyer au serveur de contraction toutes les commandes, telles que:

```
> JOIN :#gamesdonequick
> PRIVMSG #gamesdonequick :Hello world!
```

Un guide plus détaillé des commandes client-serveur peut être trouvé ici .

Capacités spécifiques à Twitch

Alors que Twitch utilise un service IRC standard, certains événements du service IRC sont liés à l'activité d'un canal sur le site Web Twitch. Les exemples ici sont l'activation ou la désactivation du mode lent, le mode réservé aux abonnés étant activé / désactivé sur le chat, l'activité d'hébergement et l'activité de gestion des bits, entre autres.

Les détails sur ces fonctionnalités spécifiques à Twitch sont répertoriés dans la documentation de GitHub pour Twitch IRC, que vous pouvez trouver ici .

Examples

Python

Voici un simple programme de ligne de commande Python qui se connecte à un canal Twitch en tant que bot et répond à quelques commandes simples.

Dépendances:

- `irc` Python lib (`pip install irc` ou `easy_install irc`)

Source: https://gist.github.com/jessewebb/65b554b5be784dd7c8d1

```
import logging
import sys

from irc.bot import SingleServerIRCBot


# config
HOST = 'irc.twitch.tv'
PORT = 6667
USERNAME = 'nickname'
PASSWORD = 'oauth:twitch_token'  # http://www.twitchapps.com/tmi/
CHANNEL = '#channel'


def _get_logger():
    logger_name = 'vbot'
    logger_level = logging.DEBUG
    log_line_format = '%(asctime)s | %(name)s - %(levelname)s : %(message)s'
    log_line_date_format = '%Y-%m-%dT%H:%M:%SZ'
    logger_ = logging.getLogger(logger_name)
    logger_.setLevel(logger_level)
    logging_handler = logging.StreamHandler(stream=sys.stdout)
    logging_handler.setLevel(logger_level)
    logging_formatter = logging.Formatter(log_line_format, datefmt=log_line_date_format)
    logging_handler.setFormatter(logging_formatter)
    logger_.addHandler(logging_handler)
    return logger_

logger = _get_logger()


class VBot(SingleServerIRCBot):
    VERSION = '1.0.0'

    def __init__(self, host, port, nickname, password, channel):
        logger.debug('VBot.__init__ (VERSION = %r)', self.VERSION)
        SingleServerIRCBot.__init__(self, [(host, port, password)], nickname, nickname)
        self.channel = channel
        self.viewers = []
```

```
    def on_welcome(self, connection, event):
        logger.debug('VBot.on_welcome')
        connection.join(self.channel)
        connection.privmsg(event.target, 'Hello world!')

    def on_join(self, connection, event):
        logger.debug('VBot.on_join')
        nickname = self._parse_nickname_from_twitch_user_id(event.source)
        self.viewers.append(nickname)

        if nickname.lower() == connection.get_nickname().lower():
            connection.privmsg(event.target, 'Hello world!')

    def on_part(self, connection, event):
        logger.debug('VBot.on_part')
        nickname = self._parse_nickname_from_twitch_user_id(event.source)
        self.viewers.remove(nickname)

    def on_pubmsg(self, connection, event):
        logger.debug('VBot.on_pubmsg')
        message = event.arguments[0]
        logger.debug('message = %r', message)
        # Respond to messages starting with !
        if message.startswith("!"):
            self.do_command(event, message[1:])

    def do_command(self, event, message):
        message_parts = message.split()
        command = message_parts[0]

        logger.debug('VBot.do_command (command = %r)', command)

        if command == "version":
            version_message = 'Version: %s' % self.VERSION
            self.connection.privmsg(event.target, version_message)
        if command == "count_viewers":
            num_viewers = len(self.viewers)
            num_viewers_message = 'Viewer count: %d' % num_viewers
            self.connection.privmsg(event.target, num_viewers_message)
        elif command == 'exit':
            self.die(msg="")
        else:
            logger.error('Unrecognized command: %r', command)

    @staticmethod
    def _parse_nickname_from_twitch_user_id(user_id):
        # nickname!username@nickname.tmi.twitch.tv
        return user_id.split('!', 1)[0]


def main():
    my_bot = VBot(HOST, PORT, USERNAME, PASSWORD, CHANNEL)
    my_bot.start()


if __name__ == '__main__':
    main()
```

Lire Twitch Chat (IRC) Bot en ligne: https://riptutorial.com/fr/twitch/topic/1847/twitch-chat--irc--bot

Crédits

S. No	Chapitres	Contributeurs
1	Commencer avec twitch	Community, DallasNChains
2	Appel des API Twitch	DallasNChains, KnottytOmo
3	Lecteur vidéo interactif intégré	Community, Tim Penner
4	Listes des Streamers par jeu	Christopher Muller, Community, DallasNChains, Josh
5	Obtenir un jeton OAuth à l'aide du flux de code d'autorisation	DallasNChains
6	Twitch Chat (IRC) Bot	awkwardpaws, Jesse Webb, Josh, Mio Bambino, Old Badman Grey