3日で作るtwitterクライアント

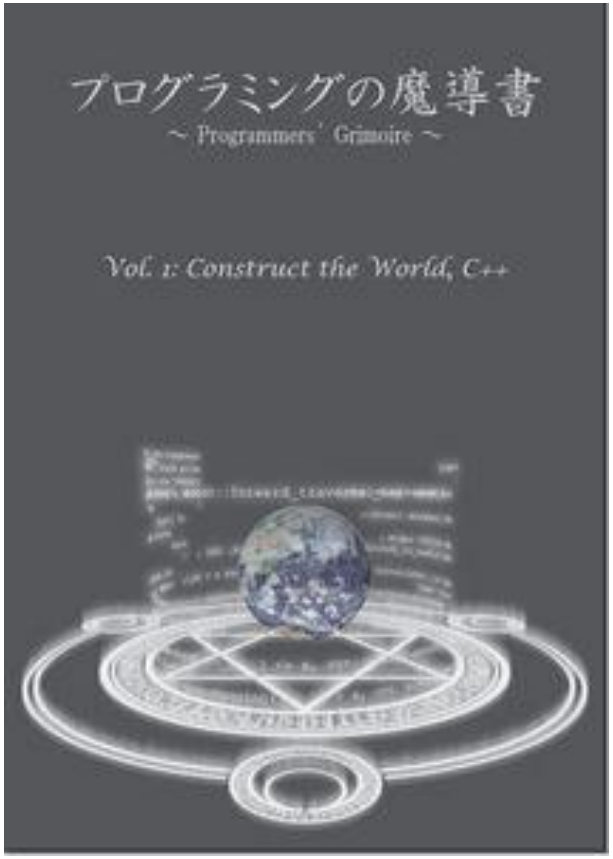
clown (tt_clown)
tt.clown@gmail.com

資料: http://cielquis.net/download/201010_boost.pdf

自己紹介

- tt_clown
  - tt_ プレフィクスは要らない子
  - ベンチャー企業で働いてます
    - 最近はフリーソフト作成
- コード
  - CLX C++ Libraries (http://clx.cielquis.net/)
  - github (http://github.com/clown/)
- その他
  - http://d.hatena.ne.jp/tt_clown/
  - http://twitter.com/tt_clown

魔導書に記事かきました



- Boost.Asio のお話
- 周りの反応
  - Boost.Asio の使い方の説明用としてはサンプルコードがしんどい
  - もうちょっと説明に特化したようなのが良かった
  - ・・・と言うか説明する気ないだろ

http://longgate.co.jp/products.html

デスヨネー

言い訳(と発表内容の動機)

- 単体で完結させたくなかった
  - 「魔導書に載せたコードを使った何か」をやりたかった
- http 通信を利用した何か・・・
  - 最近？ twitter が人気
  - 何かおーおーす？とか言う認証方式に統一されたらしい

OAuth / xAuth 認証クラスを作ろう

背景と動機 (もうちょっと真面目に)

- ここ 5 年くらい Web(サービス型)アプリが人気
  - SaaS, PaaS, IaaS, …
    - … AaaS から ZaaS まで命名するつもりか！
  - クライアント側の端末が Web ブラウザに固定される
    - cgi (Perl, Ruby, Python, PHP) + JavaScript が主流に
- Web アプリのメリット・デメリット
  - メリット: OS 非依存, インストール不要, データの共有, …
    - ブラウザ依存問題はあるが, 先達がかなり頑張ってくれてる
  - デメリット: 人気になると死ぬ
    - 503 もうずっと犬人大杉
- 個人レベルでは体力 (金銭) 的に辛い

twitter クライアント

- twitter クライアントへの需要
  - ここ数年でかなりの数の twitter クライアントがリリース
    - 例: Twitterクライアント – Twitter まとめ Wiki
      - http://bit.ly/bxkSMv
  - Web アプリではなく，デスクトップアプリと言う形（が多い）
- デスクトップ・アプリへの需要も若干復調か？

乗るしかない　このビッグ？ウェーブに

そう言う訳で OAuth/xAuth のお話

OAuth 認証とは？

BASIC 認証は ID やパスワードが一時的にせよ API 利用側に預けられてしまう
いう問題です。これを回避するために、アカウントの認証処理自体をついったー
などAPI提供側に代わりにやってもらい、API 利用側は認証された結果だけをも
らうという形にしたのが OAuth です。

OAuthプロトコルの中身をざっくり解説してみるよ –ゆろよろ日記
http://d.hatena.ne.jp/yuroyoro/20100506/1273137673

- ・・・だそうです
- とりあえずサンプルコードを探そう

サンプルコードをググる

まぁでも一応書いて判断すべきかなと思って、僕的な答えとして、さらーーーっと書いた。CONSUMER_KEY と CONSUMER_SECRET を書き換えて実行して下さい。xAuth認証を使ってtwitterにステータスをポストします。

CLX C++ Library で xAuth し てみた。 - Big Sky
http://mattn.kaoriya.net/software/lang/c/20100911222528.htm

- 作者より使いこなしてる凄い人
- このコードを参考に http 通信部分を Boost.Asio 化
  - 本当は他にも Boost 化できそうだけど保留・・・

※余談: OAuth クラスを作ろうと思った(本当の)きっかけ
http://petitbanca.blogspot.com/2009/11/oauthtwitter.html

リクエスト用文字列の生成

- リクエスト (POST/GET) 用の文字列の生成が大変
  - 以下の (key, value) で HMAC-SHA1 を計算する
    - key: consumer_secret_ + "&" + oauth_token_secret_
    - value: 必須パラメータ+Options を辞書順に並べる
      - 必須パラメータ: oauth_consumer_key, oauth_signature_method, oauth_timestamp, oauth_nonce, oauth_version
    - 計算値を base64 でエンコードし oauth_signature の値に
  - 各パラメータは RFC3986 の %xx 形式でエスケープ
    - 非制限文字: ALPHA, DIGIT, '-', '.', '_', '~'

リクエスト用文字列の生成コード

```
http_response post(const string_type& path, const parameter_map& parameters) {
    parameter_map v = this->merge(parameters); // ソートは std::map に任せる
    std::basic_stringstream<char_type> ss;
    bool first = true;
    for (parameter_map::const_iterator pos = v.begin(); pos != v.end(); ++pos) {
        if (!first) ss << '&';
        else first = false;
        ss << pos->first << '=' << pos->second;
    }
    string_type key = consumer_secret_ + "&" + oauth_token_secret_;
    clx::uri_encoder f("-._~", false, false);
    string_type uri = traits::protocol() + string_type("://") + traits::domain();
    string_type port = traits::port();
    if (port != "80" && port != "443") uri += ":" + port;
    uri += path;
    string_type val = "POST&" + clx::convert(uri, f) + "&" + clx::convert(ss.str(), f);

    const clx::sha1& hmac = clx::hmac<clx::sha1>(key.c_str(), key.size(), val.c_str(), val.size());
    string_type hm = clx::base64::encode(reinterpret_cast<const char*>(hmac.code()), 20);
    ss << "&oauth_signature=" << clx::convert(hm, f);
    return session_.post(clx::uri::encode(path), ss.str());
}
```

http://github.com/clown/cloost/blob/master/oauth_base.h

OAuth/xAuth に必要なもの

- 最終的に必要なものは以下の 4 パラメータ
  - oauth_consumer_key
  - consumer_secret
  - access_token
  - access_token_secret
- oauth_consumer_key, consumer_secret は手動で取得

ブラウザ経由で取得

access_token が分かっていれば・・・

```
#define CLOOST_NET_MIN_LIB
#include <cloost/twitter.h>
#include <iostream>
#include <string>

int main(int argc, char* argv[]) {
    std::string consumer_key  = "b4TQaZGVzuV99e7Jn02u7w";
    std::string consumer_secret = "e9uvcsF7S6mNaLxrsIhXqGwmMMvgNIjMlplXlo42Wo";
    std::string access_token = "5397032-rnWGj5qCeKiP8w2bOSYTcHwOqWBhSCViLcYoGcw";
    std::string access_token_secret = "bSVRsqa60G4CmK1KgjtQkOVIjVw0PIj1zZHEFz3yMIk";

    boost::asio::io_service service;
    boost::asio::ssl::context ctx(service, boost::asio::ssl::context::sslv23);
    cloost::twitter::oauth_base session(service, ctx, consumer_key, consumer_secret);
    session.oauth_token(access_token);
    session.oauth_token_secret(access_token_secret);

    cloost::twitter::oauth_base::parameter_map params;
    clx::uri_encoder f("-._~", false, false);
    params[“status”] = clx::convert(“Hello, OAuth!”, f); // 実際に呟く
    cloost::http_response response = session.post("/1/statuses/update.json", params);
    return 0;
}
```

http://github.com/clown/cloost_example/blob/master/example_oauth_base.cpp

OAuth 認証の道のり

ミッション: access_token を取得せよ

1. request_token を取得する
2. ユーザに認証用 URL を提示する
3. ユーザに認証して「暗証番号」を取得してもらう
4. 暗証番号を用いて access_token を取得する

request_token の取得

```
http_response get_request_token() {
    http_response response = this->post(traits::request_token_path(), parameter_map());
    this->parse(response.body());
    return response;
}
```

http://github.com/clown/cloost/blob/master/oauth.h

- これだけ
- parse() は，key1=val1&key2=val2&…
  と言う文字列を分割
  - その後，oauth_token, oauth_token_secret キーを取得
  - この oauth_token キーが request_token と呼ばれるもの

ユーザ認証用 URL の提示

- https://twitter.com/oauth/authorize?oauth_token={request_token}
  - この URL をブラウザで開いて認証してもらう

access_token の取得

```
http_response get_access_token(const string_type& pin) {
    parameter_map params;
    params["oauth_verifier"] = pin;
    http_response response = this->post(traits::access_token_path(), params);
    this->parse(response.body());
    return response;
}
```

http://github.com/clown/cloost/blob/master/oauth.h

- パラメータが 3 つ増えている
  - oauth_verifier, oauth_token, oauth_token_secret
    - oauth_token は request_token
    - oauth_token_secret 以外は, & で連結する
- ここで parse() すると access_token が手に入る

実際に通信する

```
cloost::twitter::oauth::parameter_map params;
params[“status”] = “test”; // テストと呟く
http_response response = session.post("/1/statuses/update.json", params);
```

http://github.com/clown/cloost_example/blob/master/example_oauth.cpp

- おつかれさまでした

xAuth 認証とは？

- 途中でブラウザ上で認証してもらうのはちょっと・・・
- 簡略化しました！
  - 暗証番号の代わりにユーザ名/パスワードで認証

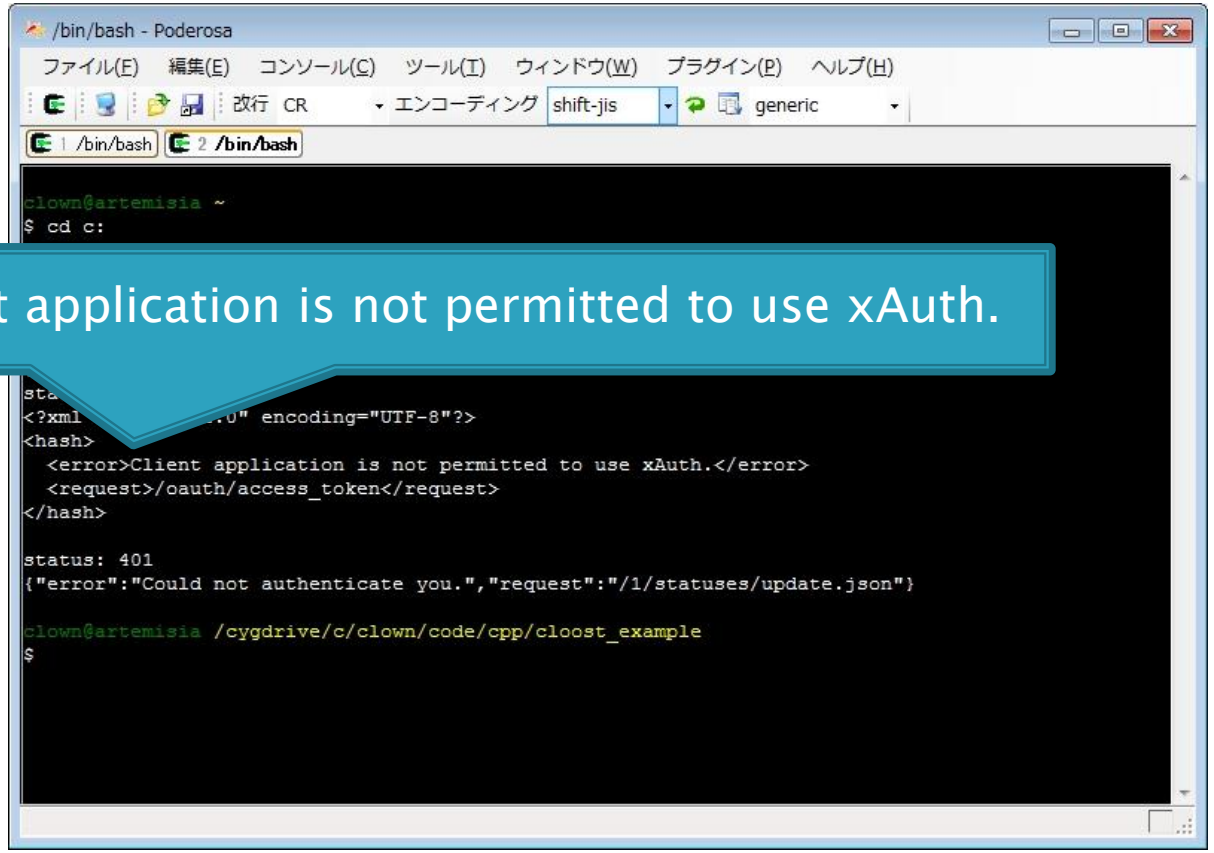
```
http_response get_access_token(const string_type& user, const string_type& pass) {
    parameter_map params;
    params["x_auth_mode"] = "client_auth";
    params["x_auth_username"] = user;
    params["x_auth_password"] = pass;

    http_response response = this->post(traits::access_token_path(), params);
    this->parse(response.body());
    return response;
}
```

xAuth 認証用のパラメータ

http://github.com/clown/cloost/blob/master/xauth.h

- パスワード・・・？

実行してみると・・・



- 通信させてくれませんでした

Boost で twitter とやり取りする場合

- Boost で実現できるもの
  - HTTP (HTTPS) 通信 -> Boost.Asio
  - JSON の解析 -> Boost.PropertyTree
- Boost で(今のところ多分)実現できないもの
  - HMAC-SHA1
  - base64 エンコーディング
  - URL 周りのエスケープ

今後の予定

- OAuth/xAuth のライブラリをもう少し真面目に作る
  - いくつか適当にしている箇所が
- GUI のサンプル作ってみる
  - 結局 twitter クライアントまでたどり着きませんでした＞＜
- cpp-netlib を見てみる
  - http://mikhailberis.github.com/cpp-netlib/
  - Boost に提案予定のネットワークプロトコル・ライブラリ

ご清聴ありがとうございました

- 今回の説明に使っていたソースコード
  - http://github.com/clown/cloost
  - http://github.com/clown/cloost_example