付録 システム開発

付録では実際に業務で使えるWebシステムの開発について学びます。有用なシステムを自作することによって、今後のシステム開発を可能にし、システム開発を外注する場合の知識を獲得します。

データ解析とシステム開発

昨今のデータ解析は手動でデータ収集やレポート作成をするのではなく、それらを自動で実行するシステムを利用することが多々あります。なぜデータ解析に置いてWebシステムを利用するかというと、日々の定型作業を自動化し、レポーティングを容易にすることによって運用のコストを下げるためです。ここでいうレポーティングとは、KPIや必要なデータをまとめたレポートの定期的(多くは日次)な作成・配布までを指します。日々行うレポーティングは馬鹿にできないコストになります。データ解析者が毎日定型作業に時間を取られていては、分析や施策提案にあてる時間が無くなってしまいます。実際様々な企業のデータ解析者から、データ解析の部署やチームを作ったはいいが、やっていることはほぼ定型のレポーティングばかり…という話は非常によく耳にします。そこで、定型作業をWebシステム化することで日々のレポーティングコストを下げ、効率化を図ることにチャレンジしてみましょう。
また、Webシステム化は単に定型作業が楽になるだけではなく、レポートに誤りがあった際の改修や再配布が楽かつ確実になるというメリットもあります。紙やファイルでレポートを受け渡しする場合、レポートに誤まりがあればレポート閲覧者に対して正しいレポートを再配布して最新の状況に更新しなくてはなりません。しかし、閲覧者の中には更新作業を怠って旧ファイルを持ったままそれを正しいデータだとしてしまうケースがあります。常に最新のレポート(※脚注：ここでいう最新のデータとは、日付が直近のデータと言う意味ではなく、データや分析手法を改修した最新バージョンのデータという意味です。)を利用して頂こうと思うなら、各利用者のローカル環境（PCやスマホなど）に置かせないようにしましょう。改修前のデータやフォーマットのせいで問題になるというケースはよくあります。ミーティングする際、各々の参加者の手持ち資料の数字が異なるのでは混乱を招きます。常に最新のレポートは同一の場所にあり関係者の誰もがアクセスするだけで閲覧できるような状況にしておけば、古い誤まったレポートを参照してしまう事を防げます。
さらに、ブラウザさえあれば誰でも閲覧可能であるのもメリットでしょう。ExcelファイルやPowerPointファイルで分析結果やデータを受け渡しする際、Linuxユーザやmacユーザでofficeソフトを入れていないケースもあり得ます。携帯端末から閲覧・操作するのも困難な場合があるでしょう(※このデメリットに関しては、Googleドライブを使うのも良いでしょう)。

本書はプログラミングを学ぶ本ではないため、各技術に関して深入りすることなく「ここはこう設定すれば動きます、この通り実行して下さい」で済ませるところが多々あり、本章で作成したシステムをきちんと理解したり改造したりするためにはプログラミングの専門書を読む必要があります。ここではさしあたって最低限動くシステムを作ることを目的として作りましょう。その程度であっても、一度実際に動くシステムを作ることによって、安定してシステムを稼働させることや要件をシステムに落とし込むことの困難さを知ることが出来るため、今後自分でシステム開発を学ぶためのモチベーションを高めたり、外部に発注する時どのようなことに気を付け何を伝えなければならないかについて重要な経験を積むことが出来るでしょう。

どのようなサービスを作るか

twitterから「東京」と「天気」を含むツイートを自動で収集し、その内容を表示するシステムを作ります。さらに、単に表示するだけではなく、6章で学んだKWIC検索とツイート数の時系列推移を計測するという機能を実装しましょう。これら機能があれば話題になっている情報を一覧できて大変便利です。これはあくまで習作ではありますが、収集するキーワード次第で市場調査や評判分析にも応用可能で実用的なシステムです。

今回のWebシステムを作る流れ

1. サーバを立てる
2. データベース管理システムを用意する
3. twitter API利用登録
4. ツイート収集プログラムを作る
5. データベースにツイートを格納するプログラムを作る
6. ツイートをブラウザ上で表示できるようにする
7. KWIC検索機能を作る
8. 時系列データを作成する
9. データ出力APIを作る
10. 時系列の件数をグラフ表示する機能を作る
11. 自動化する

開発の準備

付録の内容はサンプルコードを読みながら実践します。まずはサンプルコードを取得します。
https://github.com/AntiBayesian/DataAnalysisForPractice/ の「Download ZIP」ボタンを左クリックするとサンプルをダウンロードできます。
なお、以降の説明ではWindows7以上を利用することを想定しています。他の環境の場合は適宜読み替えて下さい。

1. サーバを立てる

Webシステムを作るにはまずWebサーバを構築しなければなりません。Webサーバとは、ブラウザやアプリなどからURLを指定するなどの要求(リクエスト)を受けて、それに応じて画像やHTMLなどのデータを返すプログラムの事です。本来であればサーバを構築するのはかなり手間がかかる上に専門知識が必要な作業ですが、XAMPPという必要な機能と設定を一括でインストール出来る便利なツールがあり、これを利用するとわずかな設定を行うだけでWebサーバを構築できます。
XAMPPを次のサイトからダウンロードして下さい。
https://www.apachefriends.org/jp/index.html

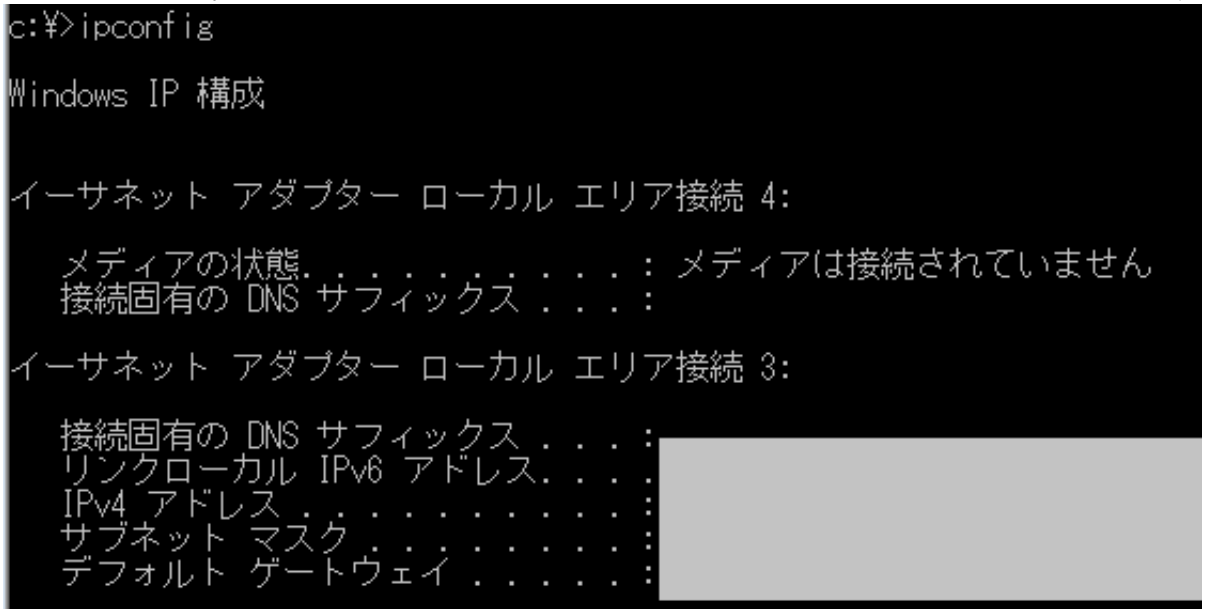
Windows環境の場合はWindowsのボタンを押します。すると2015年6月現在"xampp-win32-5.6.8-

0-VC11-installer.exe”というファイルがダウンロードされます。基本的にインストール作業はこのファイルをダブルクリックし、ダイアログに合わせてOKを押すだけで完了します。筆者の環境ではインストールに5分程度掛かりました。途中コマンドプロンプト画面が何度か開くかもしれませんが、Enterキーを押して先に進めば問題ありません。インストール作業が終了したら、ブラウザのURL欄に http://localhost/ と打ち込んでください。インストールが問題無く完了したら、次のような画面がブラウザ上に表示されます。

English / Deutsch / Francais / Nederlands / Polski / Italiano / Norwegian / Español / 中文 / Português (Brasil) / 日本語

> [Appendix.1 サーバ起動確認画面]

以下、c:\xamppフォルダにxamppをインストールしたとします。デフォルト設定ではc:\xampp\htdocsフォルダが存在します。その中に適当なファイルを入れてアクセスして見ましょう。そのフォルダにtest.txtファイルを置いた場合(ファイルパスとしてc:\xampp\htdocs\test.txtとなる場合）、 http://localhost/test.txt とブラウザのURL欄に打ち込むと、test.txtの内容が表示されます。また、このサーバにイントラネット（職場や大学などの内部だけでアクセス出来るネットワーク）上の他のパソコンや携帯電話からアクセスしたい場合は、サーバを起動しているパソコンのIPアドレスを伝えます。コマンドプロンプト(※コマンドプロンプトを起動するにはWindowsのスタートメニューから「プログラムとファイルの検索」の検索欄に「cmd.exe」と打ち込むか、)にipconfigと打ち込むと様々なネットワーク関連の情報が表示されますので、その中からIPv4 Addressと書かれたピリオド区切りになった4つの数字がそのパソコンのIPアドレスです。



> [Appendix.2 IP確認]
> 例えばIPアドレスが111.22.333.444の場合、相手には http://111.22.333.444/ にアクセスするよう伝えます。インターネット上に公開する場合はレンタルサーバを借りるのが良いでしょう。例として一つ取り上げると、さくらのレンタルサーバ(http://www.sakura.ne.jp/)であれば、最も安いプランでは月額129円で利用出来ます（2015年6月現在）。XAMPPの設定はあくまでテスト環境として利用するモノであっ

て、そのままインターネットに公開するのはセキュリティ上リスクがあるのでお勧めできません。自分でインターネット向けのサーバを構築するのは専門書をお読み下さい。
一点だけ後々のために設定ファイルを書き換えます。C:\xampp\phpフォルダ内にあるphp.iniファイルをテキストエディタで開き、"date.timezone=Europe/Berlin"という箇所を"date.timezone=Asia/Tokyo"に書き換えます。この書き換えを行うことによってタイムゾーンという時間の設定を日本に合わせます。
これでWebシステムを作る土台は整いました。htdocsフォルダの下にtwitterというフォルダを作成し、今後はそこで作業を行います。付録のサンプルコードが入ったフォルダであるAppendixの中にあるtwitterフォルダをc:\xampp\htdocsにコピーし、c:\xampp\htdocs\twitter以下にサンプルコードが下の画像の様に配置されている状態にしてください。

```
c:¥xampp¥htdocs¥twitter>dir
 ドライブ C のボリューム ラベルは Windows7_OS です
 ボリューム シリアル番号は A2AC-F375 です

 c:¥xampp¥htdocs¥twitter のディレクトリ

2015/02/23  17:28    <DIR>          .
2015/02/23  17:28    <DIR>          ..
2014/12/28  15:39             2,488 get_tweet.php
2014/11/23  23:23             2,949 insert_tweet_to_db.php
2014/11/23  23:52             1,856 kwic_tweet.php
2014/11/23  23:52               803 print_tweet.php
2014/12/09  22:33             1,230 series.html
2014/12/09  21:33               712 series.php
2015/02/03  13:58               605 series_api.php
2015/02/23  16:53            65,536 tweet.db
               8 個のファイル              76,179 バイト
               2 個のディレクトリ  17,722,966,016 バイトの空き領域
```

[Appendix.3 フォルダ構成]

2. データベース管理システムを用意する

取得したデータはそのままの形ではなく、用途に合わせて形式を決め、検索や抽出などの再利用をし易いように管理すると、その後のシステムの拡張や分析が容易になります。このように管理されたデータの集まりをデータベースと言います。また、前述のデータの再利用を実現するシステムとしてデータベース管理システム(DataBase Management System。以下DBMSと略します)があります。例えば、テキストファイルにデータを格納する場合は、検索や抽出などの機能を自力で実装(※プログラムを作ること)する必要があります。DBMSを利用すればそれらの機能が組み込まれているため、面倒な操作を簡単にかつ高速で実行できるようになります。
ここではSQLiteというDBMSを利用します。SQLiteは良く使われるDBMSの一つ(※他にMySQLやMongoDBなどがあります)です。SQLiteの特徴は、そのシステムがたった一つのファイル(※Windowsの場合SQLite3.exe)だけで構成されていることです。また、SQLiteで管理するデータも単独のファイルで管理されます。殆どのデータベースシステムはインストール作業が必要だったりDBMS内のデータを受け渡しするのに手間が掛ったりしますが、SQLiteならファイル一つ渡すだけで簡単にやりとり出来るのが魅力です。

次のページからSQLiteをダウンロードします。
http://www.sqlite.org/download.html

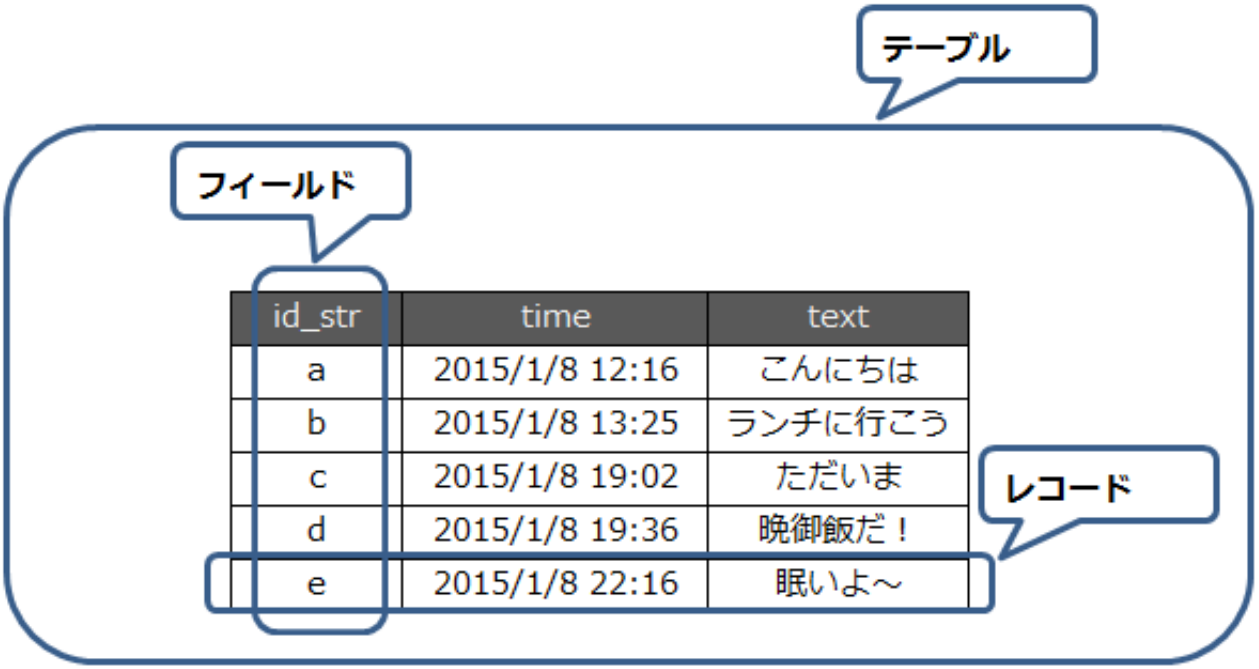
Windowsをお使いの場合は"Precompiled Binaries for Windows"の下にあるリンクをクリックしてファイルをダウンロードします。zipファイルを展開すると、中にsqlite3.exeが一つだけ入っています。これをc:\xampp\htdocs\twitterにコピーして下さい。コマンドプロンプトにcd c:\xampp\htdocs\twitterと打ち込んで作業フォルダまで移動します。cd とはchange directoryの略で、フォルダを移動するコマンドのことです。ここまでで下準備は終わりです。
SQLiteにデータを格納して利用できるようにするには、まずデータを格納するデータベースファイルを作成します。コマンドプロンプトに sqlite3 tweet.db と打ち込むことで、ツイート情報を格納するデータベースファイルを作成出来ます。フォルダにtweet.dbが生成されていることを確認してください。そしてコマンドプロンプト上では sqlite> と表示されているのを確認してください
(※sqliteを終了するにはCtrl + zを押下します。再度tweet.dbを利用する場合は、同じようにsqlite3 tweet.dbと打ち込みます)。この状態になればsqliteのコマンドを入力できます。その状態で下記コマンドを入力します。

```
create table tweet (
    id_str text,
    time text,
    text text
)
```

するとデータベースファイル内にtweetテーブルが作られます。create tableコマンドはテーブルを新規作成するコマンドで、

```
create table テーブル名 (
    フィールド名 フィールドの型,
    フィールド名 フィールドの型,
    フィールド名 フィールドの型
)
```

と指定します。
データベースにデータを格納する領域をテーブルと言います。テーブルはデータベースファイルの中に複数持つことが出来ます。テーブルはレコードと呼ばれる行と、フィールドと呼ばれる列から構成されます。このtweetテーブルはid_strとtime, textというフィールドを持ちます。Excelで例えると、データベースファイルがExcelファイル(ブック)に相当するもので、各テーブルがExcelシートに相当します。今回のテーブル定義に従うと、Excelで例えれば各行に対してid_strとtime, textの3列が存在することになります。テーブル、レコード、フィールドは下図のような関係です。そしてデータベースの中に複数のテーブルが格納されます。



> [Appendix.4 データベース、テーブル、レコード、フィールドの関係図]

各フィールドは各々型情報を持ちます。型情報とは、フィールドに格納されているデータが文字列なのか数値なのかなどを表すメタデータです。フィールドに数値型を指定しているのに文字列を格納しようとすればエラーになります。型情報のおかげで誤まった挿入をすれば即座に判明するという強い利点があります。数値の型を指定する場合はinteger(整数値のみ)、real(小数点も可能)の2つがあります。realの方が小数点も格納できるため便利だと考えられるでしょうが、その分容量を大きく取るという欠点もあります。必要に応じて使い分けて下さい。今回は全て文字列型のみを利用します。id_strはtwitterのユーザにシステムで自動的に割り振られるIDです。これは名前やアカウントIDとは異なり、ユーザ側では変更できません。そのため、アカウントを特定するのに役立ちます。例えばあるキーワードについて一日何人の人がツイートしているのかを知りたい場合に名前やアカウントIDで集計してしまうと、名前やアカウントIDを変更された時に別人として重複カウントしてしまう可能性があります。また、ある人のツイートを時系列に渡って計測したいと言う場合も名前やアカウントIDでは変更されてしまう可能性があるため不適切です。timeにはツイートを投稿した時間を、textにはツイート内容を格納します。
テーブルの定義が正しいかどうかはSQliteの場合、.schemaコマンドを利用して確認することが出来ます。.schema テーブル名と入力することで、指定したテーブルの定義が出力されます。ここでは .schema tweet と入力すると、先ほどの create tweet の定義がそのまま出力されます。

```
sqlite> .schema tweet
CREATE TABLE tweet (
    id_str text,
    time text,
    text text
);
```

> [Appendix.5 .schema実行結果]
> データベースファイル上にどのようなテーブルが存在するかは .tablesコマンドで確認できます。現時点で.tablesコマンドを入力するとtweetテーブルのみが表示されます。

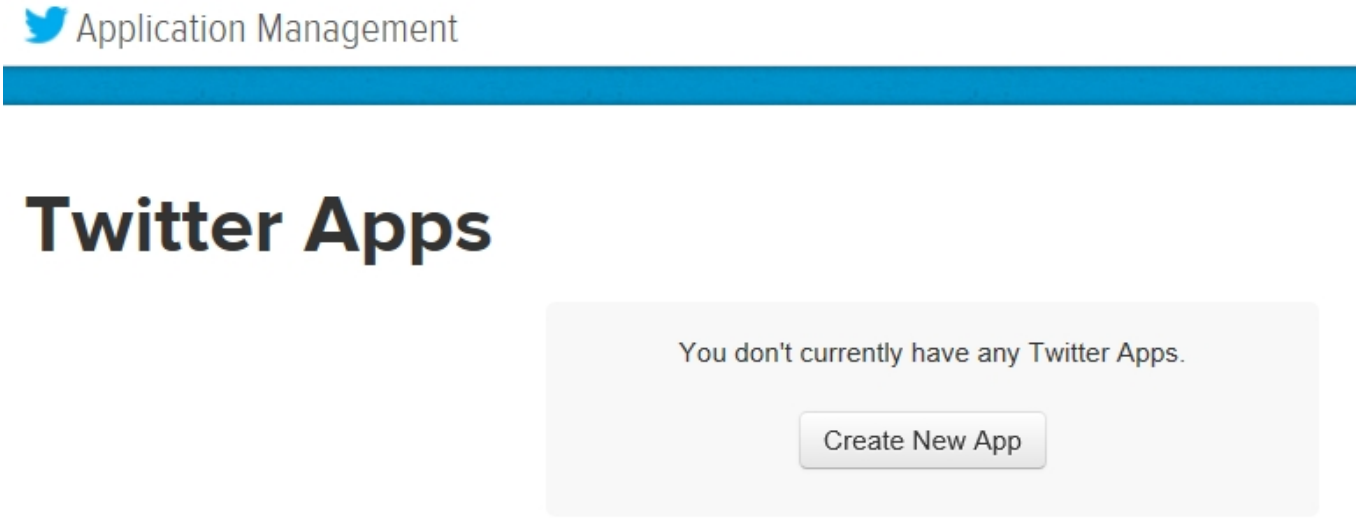
```
sqlite> .tables
tweet
```

> [Appendix.6 .tables実行結果]
> ここまで出来ればデータベースの準備は完了です。

3. twitter API利用登録

まずはデータを取得するための仕組みを作ります。twitterが提供しているAPIを利用しデータを取得する方法について学びます。ここではPHPというプログラミング言語を用いてデータを取得します。
APIを利用してデータを取得する場合は利用登録やユーザ認証が必要なケースもあります。twitterの場合はアプリケーション登録を行って初めてAPIが利用可能になります。twitterのアプリケーション登録の手順を学び、実際にAPIからツイートを取得してみましょう。
(※twitterに限らずWeb APIの仕様は突然変更されることがあります。ここでの説明は、出来る限り最新の仕様に準拠するよう努めますが、動かない場合は各サービスのAPIの説明ページをご参照下さい)
まずtwitterアカウントを取得し、電話番号認証を実施します(※2014年末以降、新規登録したアカウントでは電話番号認証をしないとAPIを利用出来なくなったようです。)。既にアカウントをお持ちの場合はそのtwitter IDとパスワードを控えておいてください。アカウントを登録出来たら下記のtwitterの開発者向けサイトに行きます。
https://dev.twitter.com/

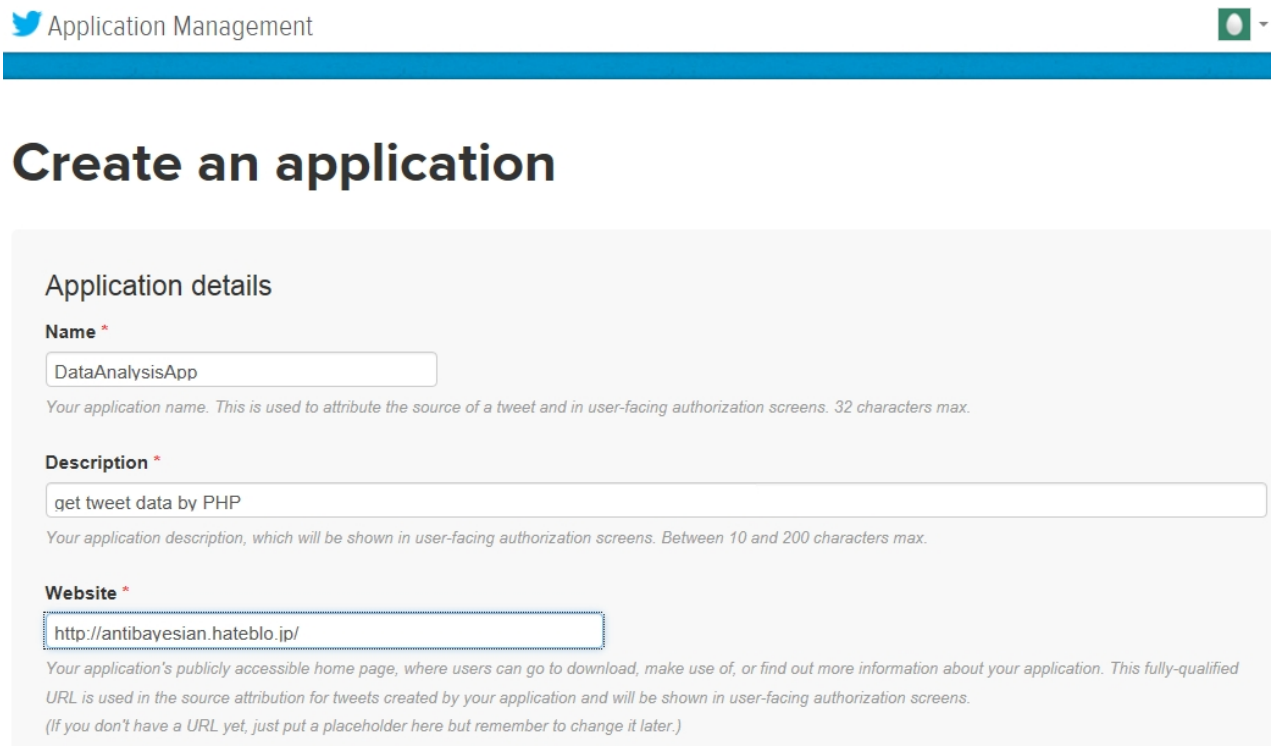
このページの画面下部にある「TOOLS Manage Your Apps」のリンクからtwitter API登録ページに移動します。
https://apps.twitter.com/
Create New Appsボタンをクリックします。
https://apps.twitter.com/app/new



> [Appendix.7 twitter apps登録画面]

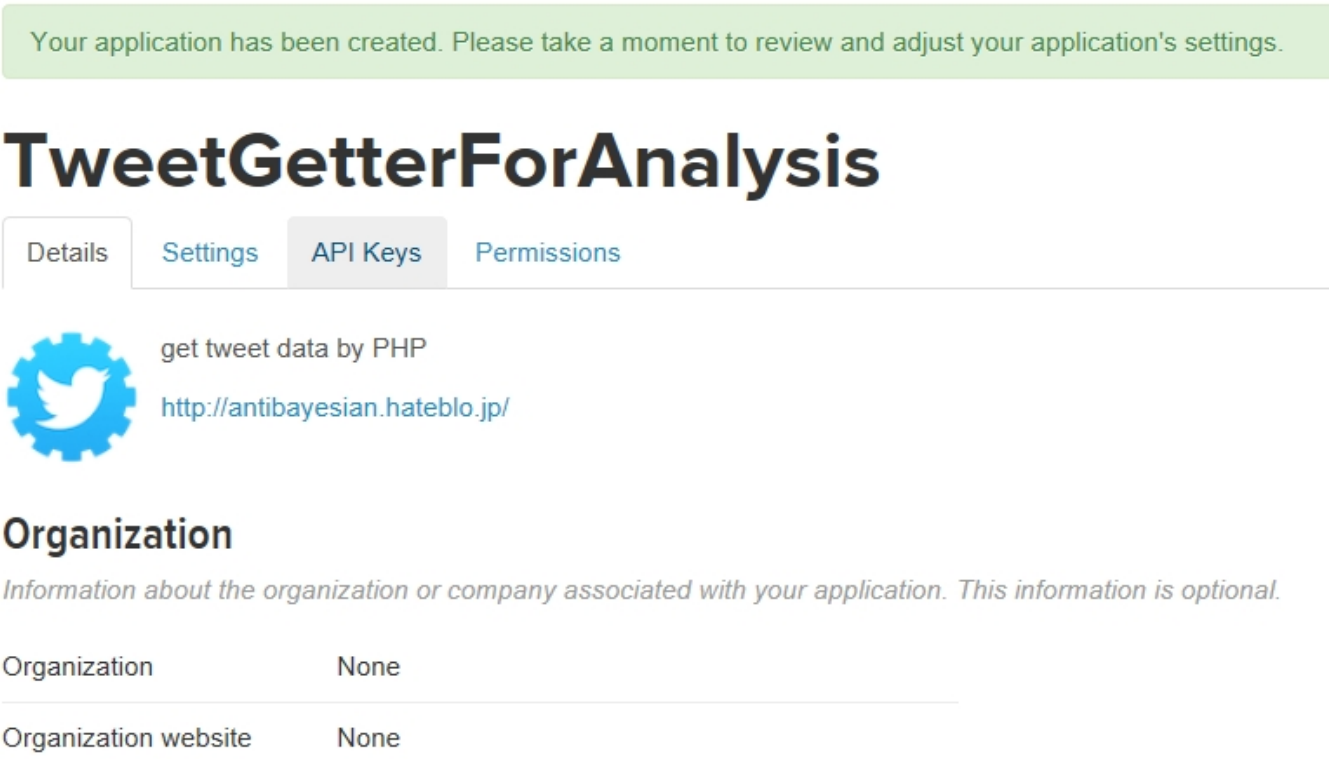
Application details内のテキストボックスに必要な情報を記入します。

- Name:入力必須。他の登録されたNameと被らなければ自由に名前を付けられます。
- Description:入力必須。アプリの説明書きで、自由に記入してかまいません。
- Website:入力必須。ご自分のウェブサイトを登録します。特に無ければ自分のtwitterホームなどでも構いません。



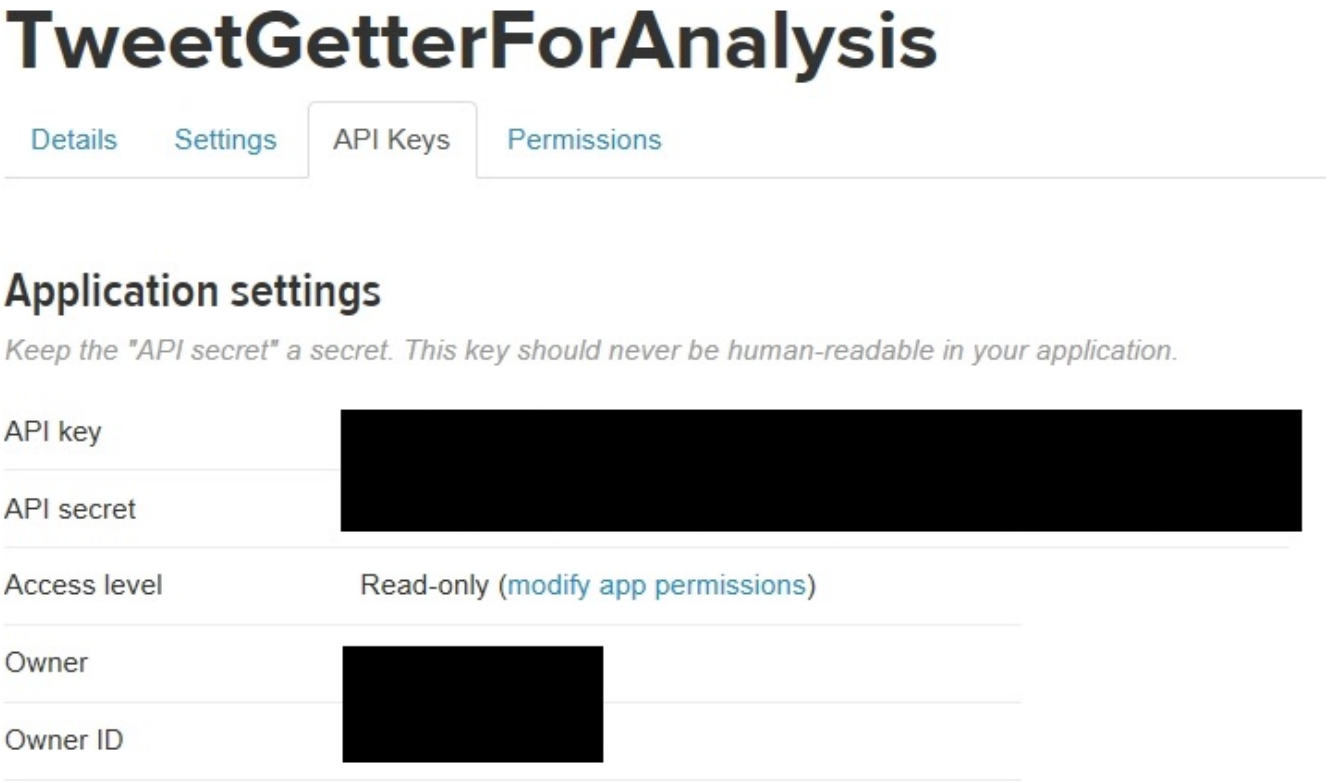
> [Appendix.8 twitter apps設定入力画面]

最後のDeveloper Rules of the Roadをよく読み同意したら"Yes, I agree"にチェックを入れます。すると画像のようなアプリケーション作成に成功したというメッセージが表示されます



> [Appendix.9 twitter apps登録完了画面]

これでアプリケーション登録完了です。ここからAPI設定情報を取得します。API Keysというタブを選択し、Application settingsのAPI keyとAPI secret 、Your access tokenのAccess tokenとAccess token secretの計4つの文字列を取得します。この設定情報を用いて認証し、APIを利用します。



> [Appendix.10 twitter apps設定表示画面]

4. ツイート収集プログラムを作る

ここからは実際にソースコードを眺めながら内容を理解していきましょう。サンプルコードが格納されているフォルダ、c:\xampp\htdocs\twitterを開き、**get_tweet.php**をテキストエディタで開いて下さい。これがtwitter APIを利用してツイート情報を取得するプログラムになります。順を追って理解していきましょう。プログラムのコードは一回見ただけで理解出来なくて構いません。筆者は一読目でプログラムがどのような流れで処理しているのかを把握し、二読目で各々の要素が具体的に何をしているのか理解し、三読目で流れと個々の要素を結び付けることでプログラム全体を把握するという読み方を良くします。また、何度読んでもコードの中身が分からないと言う事もあると思います。その場合は手を動かしてプログラムがどのような挙動をするのか、また、コードのどこをどう変えればどう挙動が変わるのかを確認すると言う確認を少しづつ進めていってください。

GETとはリクエストの種類の事で、リクエストにはGETとPOSTという種類があります。簡単に言うと、GETはリクエストの際パラメタをURLにつけて送ります。具体的にはURLに対しkeywordというパラメタにtestを設定してGETリクエストを送りたい場合は URL?keyword=test と指定します。POSTはURLではなくリクエストのボディ部という部分にパラメタを格納して送ります。twitter APIのドキュメントを見ると、ツイートを取得するAPIにはGETでリクエストを送るように書いているため、ここではGETでリクエストを送ります。APIを利用する場合はドキュメントを見て適切なリクエスト方法を選択しましょう。
GETとPOSTの細かい仕様の違いはさておき、GETではパラメタの内容がURLに残り、POSTでは残らないというのがシステム利用者から見てもわかりやすい違いです。ツイート閲覧ツールにおいて、今自分が見ているパラメタ（例えば検索ワードや取得期間など）の結果を相手にも渡したい時、GET方式であれば今自分が見ているURLにパラメタの内容が入っているため、そのURLをコピーして相手に渡すだけで、相手は自分と同じパラメタを設定した状態でツイートを見ることが出来ます。この特性を利用した方が便利であるため、今回のツイート閲覧ツールではGET方式でリクエストするようにします。このように、プログラミングについてしっかり学ぶと、各々の方式の特性・

利点欠点がわかるため、最適な方式を選ぶことが出来るようになります。最初から分厚くて難解なプログラミングの本を読む必要はありませんが、いつかプログラミングの本も読み通すことが出来れば、自分でプログラムを作る際も発注する際も適切なシステムを構築する助けとなるでしょう。

コマンドプロンプトに php get_tweet.php というように php <ファイル名> と打ち込むとPHPプログラムを実行できます。get_tweet.phpを置いているフォルダまでコマンドプロンプト上でcd <指定フォルダ>で移動して実行して下さい。
ライブラリを用いることによって、コメントを抜くと実質僅か30行程度でツイートを収集することが出来ました。あとはこれをデータベースに格納し、データを自由に加工して取り出せるようにし、加工済みのデータをブラウザ上でグラフや表にして閲覧できるようにしましょう。

```php
<?php
// <-これ以降の文字列はプログラムの挙動に影響を与えない。「コメント」と呼ばれる、コードに対して注記するための記法

// twitterAPIを利用するためには正式に利用登録を行った利用者であることを
// twitter側に伝える「認証処理」が必要です。
// ここではtwitter用の認証ライブラリであるtwitteroauthを利用します。
// ライブラリとは便利な機能を使えるようにまとめておいたプログラムのことです。
// ライブラリを利用することで様々な機能を実装の手間を掛けずに簡単に使えるようになります。
require_once 'twitteroauth/twitteroauth.php';

// ご自分のCONSUMER_KEY他を設定して下さい
define('CONSUMER_KEY', '***');
define('CONSUMER_SECRET', '***');
define('ACCESS_TOKEN', '***');
define('ACCESS_TOKEN_SECRET', '***');

// twitteroauthで認証するためのCONSUMER_KEYなどの情報を設定。
$twitterOAuth = new TwitterOAuth(
    CONSUMER_KEY,
    CONSUMER_SECRET,
    ACCESS_TOKEN,
    ACCESS_TOKEN_SECRET
);

// 検索キーワードを設定
// キーワードをスペースで繋げると、両方のキーワードを含むツイートを収集します。
// キーワードをorで繋ぐと、キーワードのどちらかだけでも含むツイートを収集します。
// 東京の天気に関するツイートを取集したい場合は、両方のキーワードを含んで欲しいため、スペースで繋いでいます。
$search_words = '東京 天気';

// twitter APIに渡すパラメタを指定。
// q : 検索キーワード
// lang : 言語設定。"ja"を設定することで日本語ツイートのみ取得
// count : ツイートを最大何件取得するかの設定
// result_type : 取得順番タイプの設定。recentで最新順、popularで人気順
$param = array(
    "q"=>$search_words,
    "lang"=>"ja",
    "count"=>10,
    "result_type"=>"recent");
```

```
// twitter APIを利用してパラメタの指定通りにデータを取得する処理。
// 本来ならここで認証処理が入りますが、twitteroauthが適切に行ってくれます。
// https://api.twitter.com/1.1/search/tweets.json はtwitter APIのURLです。
// このURLにparamで設定した条件でツイートを取得したいと要望を送る(これをリクエストと言います)ことでツイートを取得出来ます。
$json = $twitterOAuth->OAuthRequest(
    "https://api.twitter.com/1.1/search/tweets.json",
    "GET",
    $param);

// ツイート情報を扱いやすいように連想配列という形式に変換して$twitterの中に格納
$twitter = json_decode($json, true);

// var_dump関数を使うと変数の中身を確認できます。
// twitter APIからどのようなデータが取得できるのか一度眺めてみてください。
// この出力結果を見ると、取得したデータは多岐に渡り、ツイート内容だけではなく、
// ユーザ名や時間、ツイートした地点についてもデータを取得できることがわかります。これらを有効活用しましょう。
// var_dump($json);
// var_dump($twitter);

// 検索にヒットした複数のツイートを一つずつ取り出し、ユーザ名とツイート内容、投稿時間を表示
foreach($twitter['statuses'] as $tweet){
    echo $tweet['user']['name']; // ユーザ名
    echo $tweet['text']; //ツイート内容
    echo date( "Y-m-d H:i:s", strtotime($tweet['created_at'])); //  投稿時間。但し、twitterから直接渡される投稿時間は見辛いので整形している
    echo "\r\n"; // 出力結果を見易いように改行
}
```

> [get_tweet.php]

5. データベースにツイートを格納するプログラムを作る

では取得したツイートをSQLiteに格納していく処理を見ていきましょう。
insert_tweet_to_db.php を参照して下さい。ここではデータベース関連の処理（データベースへの接続、データベースからのデータ抽出、データベースへのデータ挿入）が追加されています。また、**プリペアード・ステートメント**というセキュリティ対策(※詳細は「体系的に学ぶ 安全なWebアプリケーションの作り方　脆弱性が生まれる原理と対策の実践」の「SQLインジェクション」について調べて下さい。)も入っています。

```
 <?php
require_once 'twitteroauth/twitteroauth.php';

define('CONSUMER_KEY', '***');
define('CONSUMER_SECRET', '***');
define('ACCESS_TOKEN', '***');
define('ACCESS_TOKEN_SECRET', '***');

$twitterOAuth = new TwitterOAuth(
```

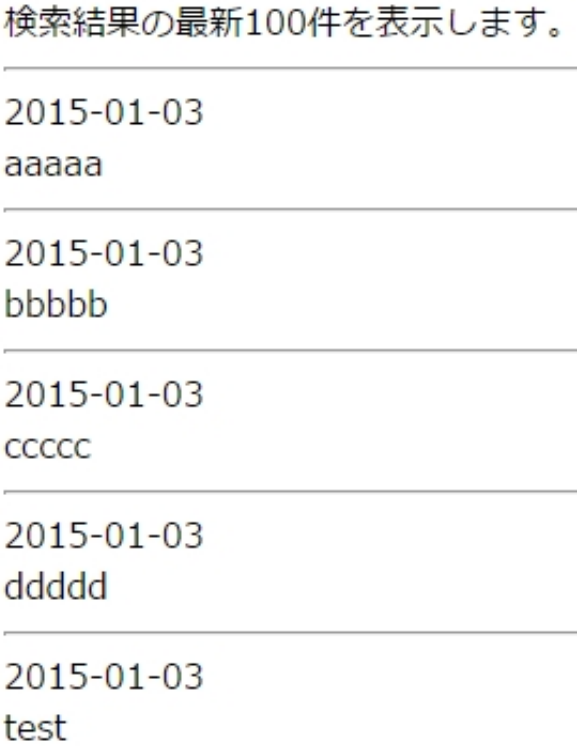
```
    CONSUMER_KEY,
    CONSUMER_SECRET,
    ACCESS_TOKEN,
    ACCESS_TOKEN_SECRET
);

$search_words = '東京 天気';

$param = array(
    "q" => $search_words,
    "lang" => "ja",
    "count" => 100,
    "result_type" => "recent"
);

$json = $twitterOAuth->OAuthRequest(
    "https://api.twitter.com/1.1/search/tweets.json",
    "GET",
    $param
);

$twitter = json_decode($json, true);

// 利用するデータベースファイルの場所(パス)を記述します。
$dsn = 'sqlite:tweet.db';

// PHPのPDOというデータベースを利用するための便利なライブラリを使う設定。
// 詳細についてはPHPの専門書籍を参照してください
$db = new PDO($dsn);
$db->setAttribute(PDO::ATTR_EMULATE_PREPARES, false);
$db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

// SQL(データベースからデータ操作を行う言語)を用いて、格納済みの最新ツイート時間を取得する処理。
// なぜこのような処理をするかは、データを重複して格納しないようにするためです。
// 単純にtwitter APIから取得したデータをデータベースに格納する処理を何度も行う場合、
// 以前取得したデータを再度格納して重複してしまう可能性があります。
// そのような重複を防ぐため、現時点でデータベースに格納されているデータのうち最新の時間を取り出し、
// その時間よりも後にツイートされたデータだけを格納するという処理を入れる必要があります。
$query = 'SELECT max(time) last_time FROM tweet';
$prepare = $db->prepare($query);
$prepare->execute();
$result = $prepare->fetch(PDO::FETCH_ASSOC);
if (isset($result['last_time'])){
    $last_time = $result['last_time'];
} else {
    // ツイートを1つも格納していなければ0を割り当てる
    $last_time = 0;
}

// php.iniを設定出来る権限が無い場合などは次のコードからコメント(//の部分)を外して利用する
// php.iniでタイムゾーン設定をしていれば次の記述は不要
// date_default_timezone_set('Asia/Tokyo');

foreach($twitter['statuses'] as $tweet){
    $time = date( "Y-m-d H:i:s", strtotime($tweet['created_at']));
    // ツイートが既に格納済みの最新ツイートよりも新しいかどうかで分岐させる処理
    // 新しい場合のみデータベースに格納する
```

```
    if ($time > $last_time) {
        // SQLを用いて取得したツイートをデータベースに格納する処理
        $query = 'INSERT INTO tweet (id_str, time, text) VALUES (:id_str, :time, :text)';
        $prepare = $db->prepare($query);
        // プリペアード・ステートメントという機能でバグやセキュリティの問題を防ぐ
        // 詳細はセキュリティ系の専門書を参照
        $prepare->bindValue(':id_str', $tweet['id_str'], PDO::PARAM_STR);
        $prepare->bindValue(':time', $time, PDO::PARAM_STR);
        $prepare->bindValue(':text', $tweet['text'], PDO::PARAM_STR);
        $prepare->execute();
    }
}
```

> [insert_tweet_to_db.php]

6. ツイートをブラウザ上で表示できるようにする

print_tweet.php を参照して下さい。コードを見ればわかるように、PHPの出力結果をブラウザ上で表示するのはとても簡単です。ここではHTMLというものが登場しています。
HTMLとはブラウザで表示する文書のマークアップ言語の一種です。マークアップ言語とは、文章の段落や章立てなどの構成、また、フォントの色やサイズなどの見た目を指定する言語のことです。そして水平線を引く `<hr>` や改行の `<br>` などのHTMLの要素をHTMLタグと言います。今回登場したモノ以外にも沢山のHTMLタグが存在します。例えば `<b></b>` というタグで囲まれた文字列はブラウザ上で太字になって表示されます。但し、Webページの見た目を変更する場合はHTMLだけではなく、CSSを用いるのが一般的です。HTMLはあくまでも文書の構造を規定するために用い、デザインする場合はCSSを用いましょう。ただ、一度にHTMLもPHPもCSSも学ぶとなると大変なので、ここではHTMLで雑に表示するのみに留めます。



[Appendix.11 kwic_tweet.phpに検索キーワードを入力した結果]

```
<!DOCTYPE html>
```

```
<html lang="ja">
<head>
<meta charset="UTF-8">
<title>ツイート表示</title>
</head>
<body>
検索結果の最新100件を表示します。
<?php
$dsn = 'sqlite:tweet.db';
$db = new PDO($dsn);
$db->setAttribute(PDO::ATTR_EMULATE_PREPARES, false);
$db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

// 最新100件のツイート情報を取得するクエリ
// order by timeでtime順にデータを並び替えることが出来ます。
// 何も指定しないと昇順、descを指定すると降順でデータが並び替えられます。
$query = 'select time, text from tweet order by time desc limit 100;';
$stmt = $db->query($query);
$results = $stmt->fetchAll(PDO::FETCH_NUM);

foreach ($results as $key => $value) {
    // <hr>, <br>はHTMLタグ。
    // <hr>で水平線を引き、<br>で改行する
    echo '<hr>'.$value[0].'<br>'.$value[1];
}
```

> [print_tweet.php]

7. KWIC検索機能を作る

ここでKWIC検索機能として盛り込みたいのは

1. 検索ワードを含むツイートだけを表示する
2. 検索ワードを含むツイートが何件あるかを表示する

の2点です。また、検索ワードを入れなければ最新100ツイートを表示するという仕様にします。そうすれば、ブラウザでこのページを開いた時に最新のツイートが見れますし、それを元に気になる検索ワードが浮かべばそれをKWIC検索に掛けてみるという利用シーンが想定出来ます。検索キーワードを入れるなど何らかの動作をしないと真っ白なページしか表示されないより、そちらの方が使い勝手が良いでしょう。KWIC検索機能を盛り込んだソースコードが **kwic_tweet.php** になります。
ここではHTMLによるフォームの作成と、GETリクエストへの対処という新しい要素が含まれています。
`<form></form>` で囲まれた部分がHTMLによるフォームを作成する部分で、フォームとはユーザが操作できるボタンやテキストボックスのことです。このテキストボックスに検索したいワードを入れ、ボタンを押すとそれに応じてKWIC検索が可能となります。そしてテキストボックスにユーザが入力した値はPHPプログラムの方で$_GET['word']という変数名で受けとります。これはこのフォームがgetリクエストで、なおかつ3行目のテキストボックスに name="word" と名前を付けたからです。このnameに設定された値を参照することでHTMLからPHPにデータを受け渡しできます。

検索結果の最新100件を表示します。

e KWIC検索

2015-01-03, A
aaaaa

2015-01-03, B
bbbbb

2015-01-03, C
ccccc

2015-01-03, D
ddddd

2015-01-03, test
test

2015-01-02, D
ddddd

2015-01-02, E
eeeee

> [Appendix.11 kwic_tweet.php画面]

この状態から、検索フォームに"e"を入力して「kwic検索」ボタンを押下すると、テキストに"e"を含む"eeeee"と"test"が抽出されます。

検索結果の最新100件を表示します。

検索語を入力してください KWIC検索

検索ワード：e, 総ヒット件数:2

2015-01-03, test
test

2015-01-02, E
eeeee

> [Appendix.11 kwic_tweet.phpに検索キーワードを入力した結果]

```
<!DOCTYPE html>
<html lang="ja">
<head>
<meta charset="UTF-8">
<title>KWIC検索</title>
</head>
```

```
<body>

検索結果の最新100件を表示します。
<form action="kwic_tweet.php" method="get">
  <input type="text" name="word" placeholder="検索語を入力してください">
  <input type="submit" value="KWIC検索">
</form>

<?php
$dsn = 'sqlite:tweet.db';
$db = new PDO($dsn);
$db->setAttribute(PDO::ATTR_EMULATE_PREPARES, false);
$db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

// 検索ボックスに検索ワードが入れられていたかどうかで分岐
if (isset($_GET['word'])) {// 検索ボックスに検索ワードが入っている場合の処理
    $kwic = $_GET['word'];

    // 検索ワードにヒットした件数を取得。
    // SQLはwhereで絞り込み条件を設定できる。
    // text like "%AAA%"で、AAAを含むtextを取得する。
    $query = 'select count(*) from tweet where text like :kwic ;';
    $stmt = $db->query($query);
    $stmt->bindValue(':kwic', '%' . preg_replace('/(?=[!_%])/', '!', $kwic) . '%', PDO::P
ARAM_STR);

    $cnt = $stmt->fetch(PDO::FETCH_NUM);
    echo '検索ワード:'.$kwic.', 総ヒット件数:'.$cnt[0];

    // 検索ワードにヒットしたツイート情報を取得するクエリ
    $query = 'select time, id_str, text from tweet where text like :kwic order by time de
sc limit 100;';
    $stmt = $db->query($query);
    $stmt->bindValue(':kwic', '%' . preg_replace('/(?=[!_%])/', '!', $kwic) . '%', PDO::P
ARAM_STR);
} else {// 検索ボックスに検索ワードが入っていない場合の処理
    $query = 'select time, id_str, text from tweet order by time desc limit 100;';
    $stmt = $db->query($query);
}

$results = $stmt->fetchAll(PDO::FETCH_NUM);

// 受け取った検索ワードをそのまま利用するのではなく、htmlspecialcharsというHTMLタグを除去する関数
に通す。
// これに限らず、セキュリティ対策のため、必ずユーザ入力をそのまま利用しないこと！
foreach ($results as $key => $value) {
    echo '<hr>'.htmlspecialchars($value[0]) .', '. htmlspecialchars($value[1]) .'<br>'. h
tmlspecialchars($value[2]);
}
```

> [kwic_tweet.php]

8. 時系列データを作成する

series_count.phpを参照して下さい。ツイート数を日次で集計することによって、対象としている内容のツイートがどのように増減しているかを計測することが出来ます。広告やキャンペーンを打った際、想定通りにtwitter上で話題が盛り上がっているかなどの効果測定に利用したり、商品やサービスに対するポジティブ・ネガティブな単語を事前に決めておき、その単語の頻度に極端な増減が無いか確認することによって異常を検知したりすることが可能です。
時系列データを集計する場合、システムにアクセスするたびデータを集計して表示するとなると、データサイズが大きくなればなるほどシステムの負荷が大きくなり、ブラウザで集計データが表示されるまでにかかる時間も長くなります。事前に集計し、その結果を集計結果を取得する用のテーブルに格納し、ユーザがシステムにアクセスした際は集計済みのデータを表示することによって、負荷対策と高速な表示の両方を実現できます。
次のようなテーブルを作成します。

```
create table series(
    date text,
    cnt integer
)
```

これで日次のtweet頻度をカウントするためのテーブルを作成できます。dateには集計対象日、cntには各集計対象日のtweet頻度を格納します。

```
<?php
$dsn = 'sqlite:tweet.db';
$db = new PDO($dsn);
$db->setAttribute(PDO::ATTR_EMULATE_PREPARES, false);
$db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

$query = 'select substr(time, 1, 10), count(*) from tweet group by substr(time, 1, 10)';
$stmt = $db->query($query);
$results = $stmt->fetchAll(PDO::FETCH_NUM);

foreach ($results as $key => $value) {
    $query = 'INSERT INTO series (date, cnt) VALUES (:date, :cnt)';
    $prepare = $db->prepare($query);
    $prepare->bindValue(':date', $value[0], PDO::PARAM_STR);
    $prepare->bindValue(':cnt', $value[1], PDO::PARAM_INT);
    $prepare->execute();
}

$db = null;
```

> [series_count.php]

9. データ出力APIを作る

ここまでは表示するデータをDBから取得する機能と表示する機能を一つのプログラムに押し込んでいました。これを機能ごとに別のプログラムに切り分けてAPI化します。具体的な内容は**series_api.php** を参照して下さい。API化することによってシステムを疎結合に保つことが出来ます。疎結合とは、各機能が独立して存在し、ある機能を変更する際に他の機能に極力影響を与えないようにする設計のことです。全ての機能を一つのプログラムに押し込んでいると、何か一つ

の機能を変更しただけで他の機能まで影響を受けてしまい、予期せぬ処理結果を引き起こしてしまう可能性があります。また、API化することによって、ある画面表示部(フロントエンドと呼びます)だけではなく他の画面表示部も簡単に開発することが出来ます。もしAPI化せず一つのプログラムに落とし込んでいると、ほぼ同じデータを取得して別の画面を作ろうとするたびにその画面に応じてデータ取得部分の機能を実装しなければなりません。しかし、データ取得機能をAPI化しておけば、各画面だけを作り、データ取得に関してはそのAPIを利用するだけで済みます。そうなれば必要工数も抑えられますし、画面が増えても改修範囲が小さくなるためバグが無いかチェックする範囲も小さくすることができます。システム開発を依頼する際、担当エンジニアの方が「システムを疎結合にしたい」と言う時がよくあります。それはこのような利点があるからです。

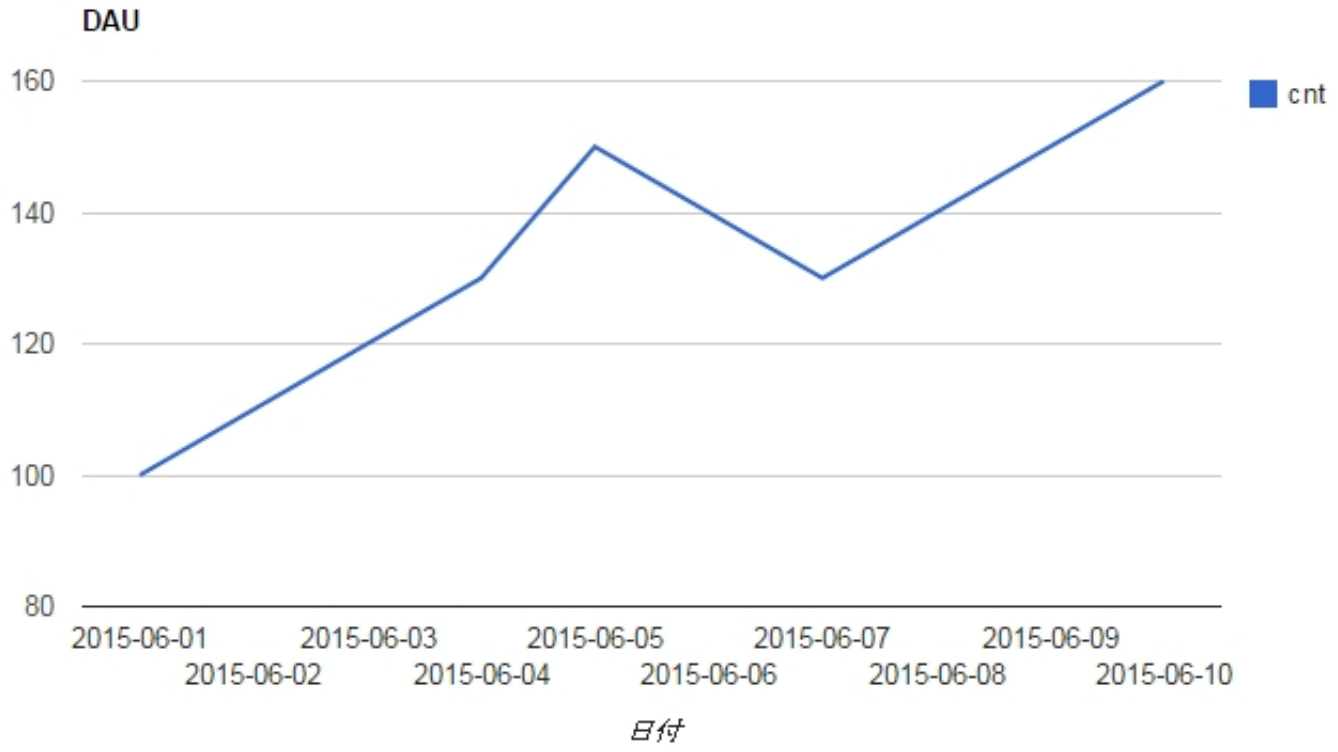
```
<?php
$dsn = 'sqlite:tweet.db';
$db = new PDO($dsn);
$db->setAttribute(PDO::ATTR_EMULATE_PREPARES, false);
$db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

// 直近10件分のデータを取得。
$query = 'select date, cnt from series order by date limit 10';

// このようにして日付を変更できるようにするとさらに便利
// 但し、前述(kwic_tweet.php)で説明したように、ユーザからの入力をそのままシステムに適用してはいけません。
// if (isset($_GET["set_date"])) {
//    $query = 'select date, cnt from series where date >= "'.$_GET["set_date"].'"order by date';
// } else {
//     $query = 'select date, cnt from series order by date limit 10';
// }

$stmt = $db->query($query);
$results = $stmt->fetchAll(PDO::FETCH_NUM);

$db = null;

$data = array();
foreach ($results as $value) {
    $data[] = array('date' => $value[0], 'cnt' => $value[1]);
}
echo header('Content-type: application/json');
echo json_encode($data);
```

> [series_api.php]

10. 時系列の件数をグラフ表示する機能を作る

先のAPIを用いて、データを折れ線グラフで時系列表示します。**draw_chart.html**を参照して下さい。ここでは折れ線グラフを描画するのにGoogle Chartsを採用します。Google ChartsはGoogleが提供しているグラフ描画APIであり、リクエストを送るだけで様々なグラフを描画してくれます。Google Chartsの使い方については「プログラマーズ雑記帳　Google Chart Tools の使い方 (http://yohshiy.blog.fc2.com/blog-category-24.html)」を参考にすると良いでしょう。また、ここではjQueryという便利なJavaScriptライブラリを利用してブラウザ上でAPIからデータを取得する仕組

みを実現しています。この仕組みは非常によく利用するものなので、覚えておいて下さい。



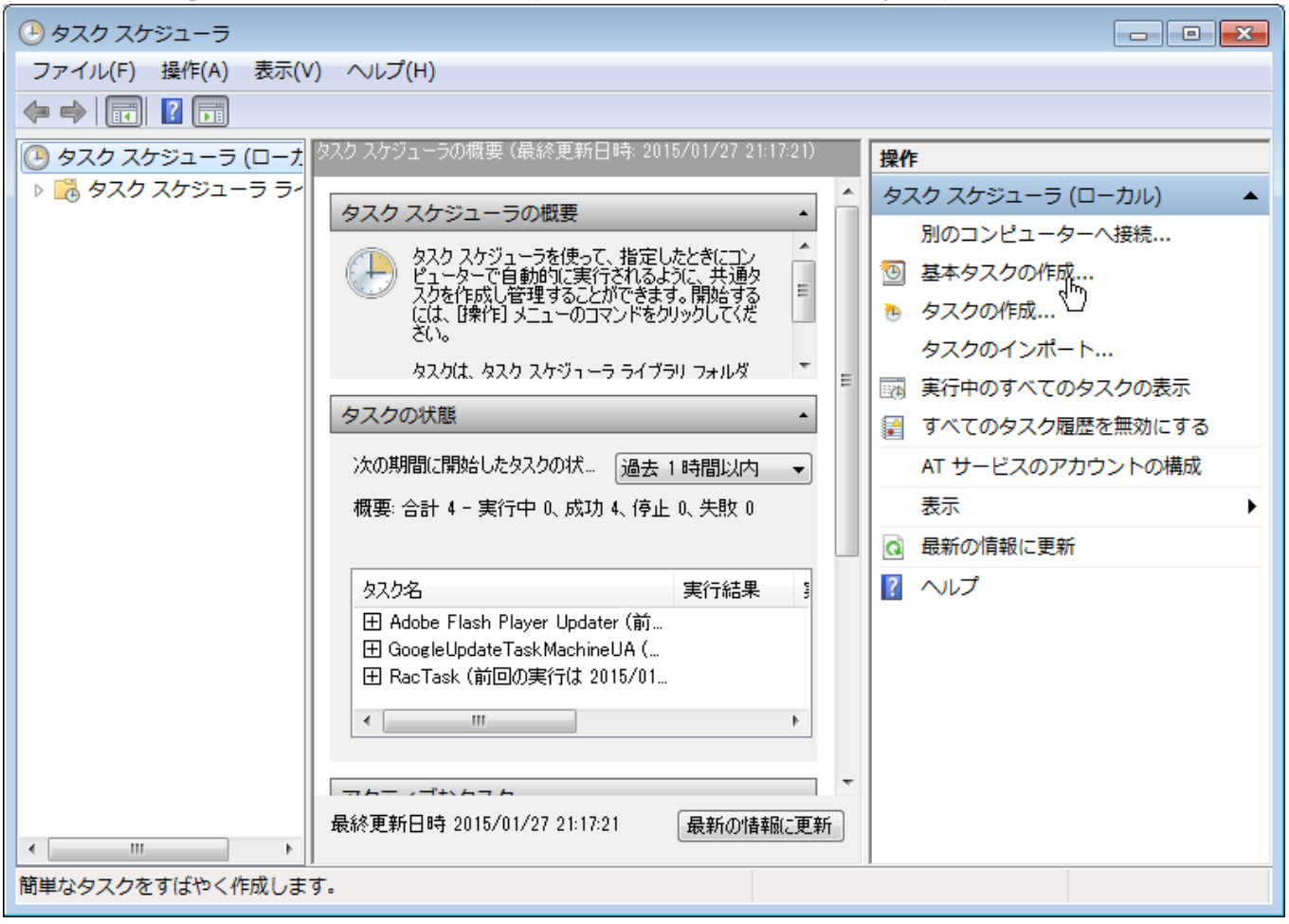

```
<html>
  <head>
    <!--Load the AJAX API-->
    <script type="text/javascript" src="https://www.google.com/jsapi"></script>
    <script type="text/javascript" src="jquery-1.11.1.min.js"></script>
    <script type="text/javascript">

    //Google Chartsを利用するための準備
    google.load('visualization', '1', {'packages':['corechart']});
    google.setOnLoadCallback(drawChart);

    function drawChart() {
      // jQueryの通信機能を利用し、APIからデータを取得
      var seriesData = $.ajax({
          url: "series_api.php",
          dataType:"json",
          async: false
          }).responseJSON;

      var drawData = [["date","cnt"]];
      for (var i=0; i<seriesData.length; i++){
        drawData.push([seriesData[i]['date'], parseInt(seriesData[i]['cnt'], 10)]);
      }
      console.log(drawData);
      var drawData = new google.visualization.arrayToDataTable(drawData);
      var chart = new google.visualization.LineChart(document.getElementById('chart_div')
);
      var options = {
                      title: 'DAU',
                      width: 800,
                      height: 400,
                      hAxis: {title: '日付', format: "####"}
                    };
```

```
      chart.draw(drawData, options);
    }

    </script>
  </head>

  <body>
    <div id="chart_div"></div>
  </body>
</html>
```

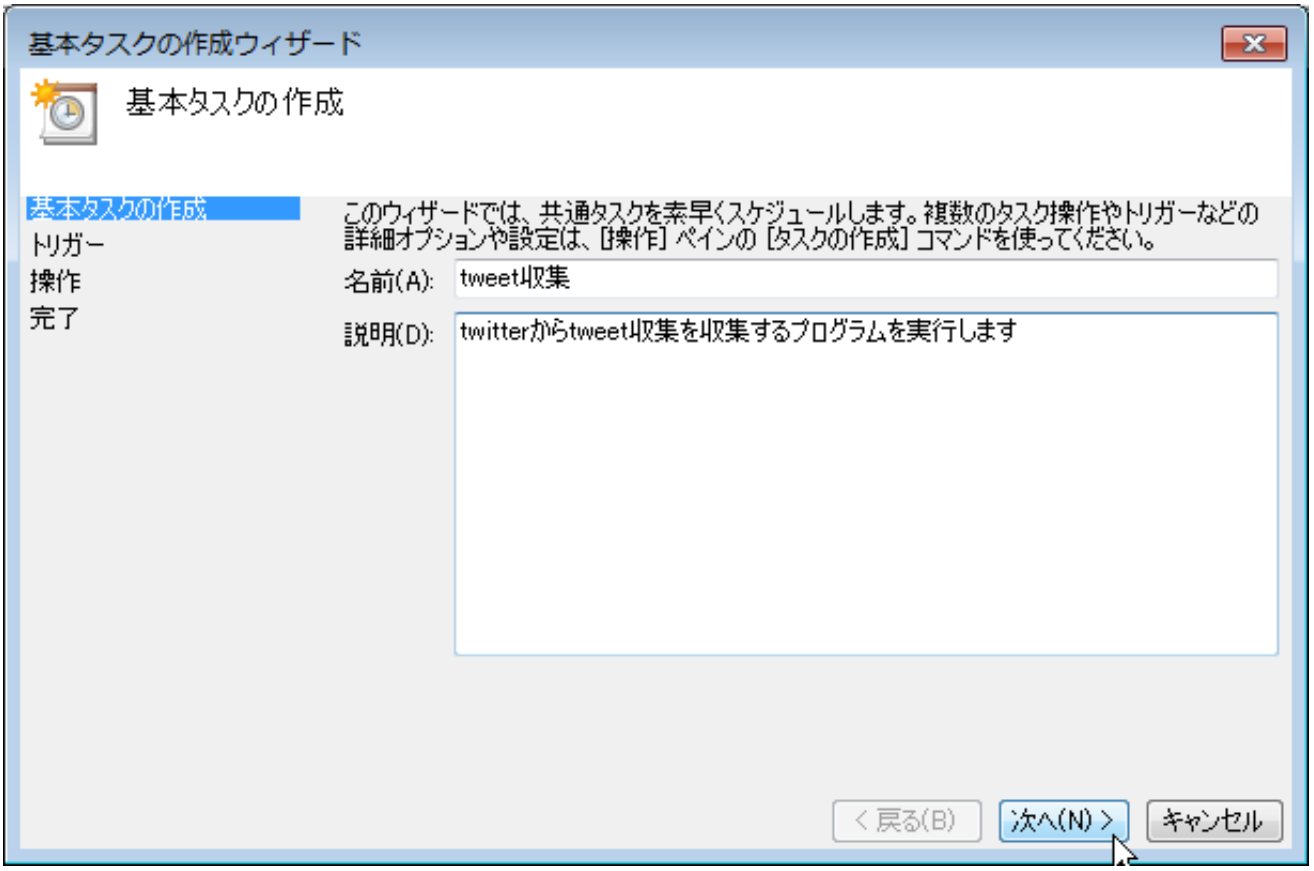
> [draw_chart.html]

11. 自動化する

データ収集や集計のスクリプトを毎回手動で実行するのは面倒ですし、休日人手が無い時や担当者が突然の急病になった時困ります。このような定型作業は自動化しましょう。Windowsの場合はタスクスケジューラ、LinuxやmacのCronというプログラムを利用します。ここではWindows向けにタスクスケジューラを用いて処理を自動化し、毎日5分おきに処理が実行されるよう設定する手順を説明します。
Windowsスタート→コントロールパネル→システムとセキュリティ→管理ツールと移動し、タスクスケジューラを左ダブルクリックで起動すると、次の画面が表示されます。

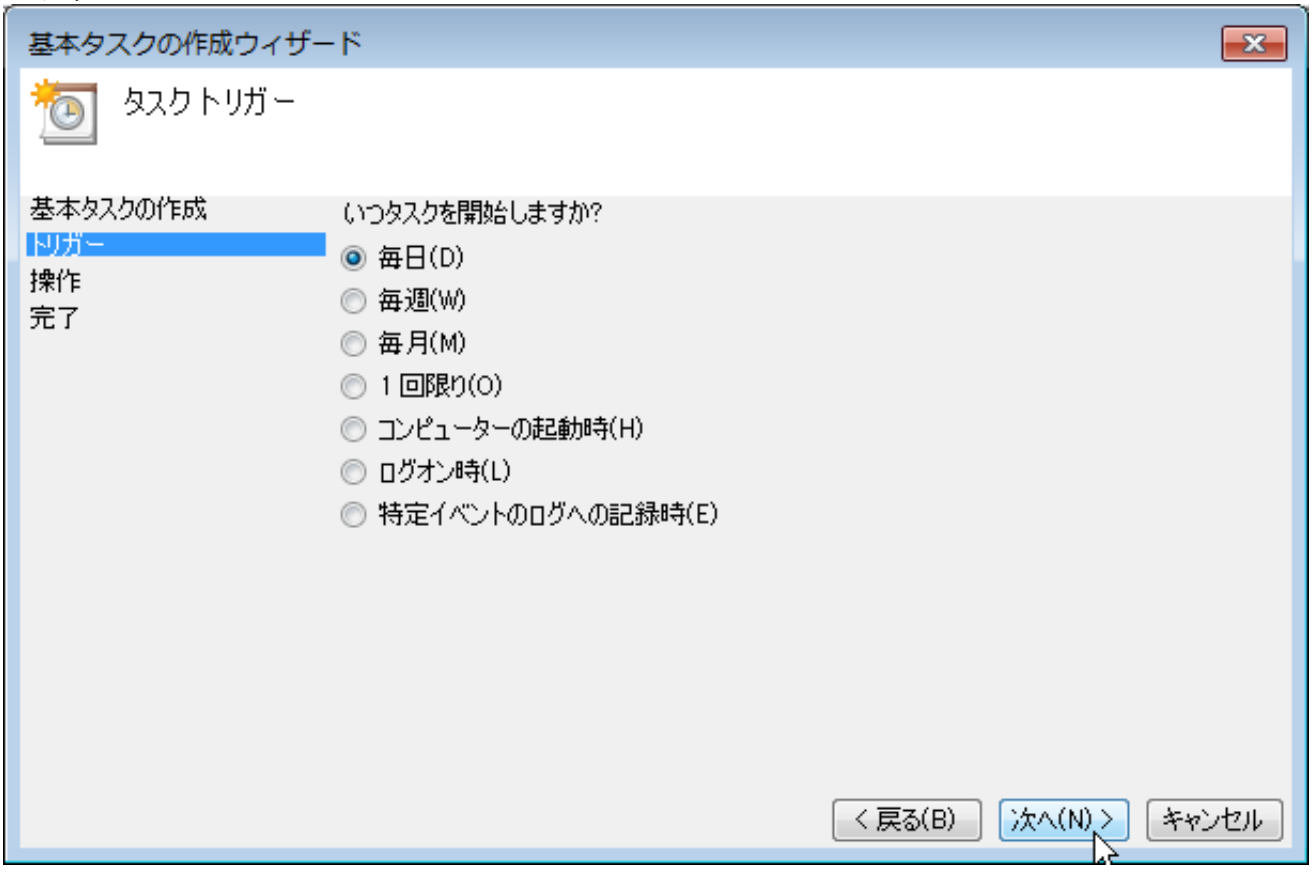


[Appendix.7 タスクスケジューラ起動]
この画面の右上部にある「基本タスクの作成」を左クリックします。



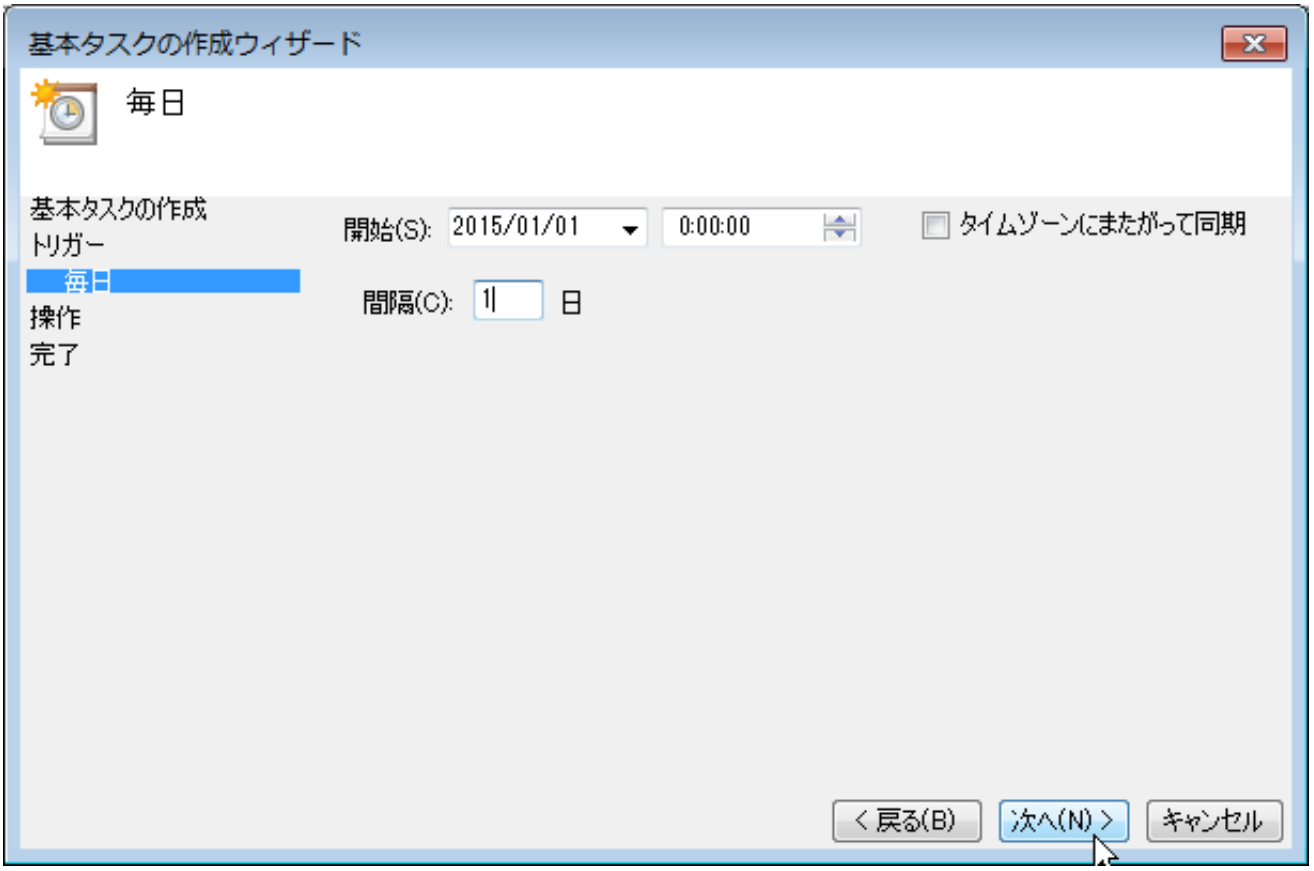
[Appendix.8 基本タスク作成]

ここで設定内容の概要を記述（これは動作に影響する設定ではなく、後から人間が見て分かり易いようにするための注意書きです。書かなくても動作はしますが、設定は忘れがちであるため書いておいた方が良いでしょう）し、「次へ(N)」を左クリックします（以降「次へ進みます」と説明します）。

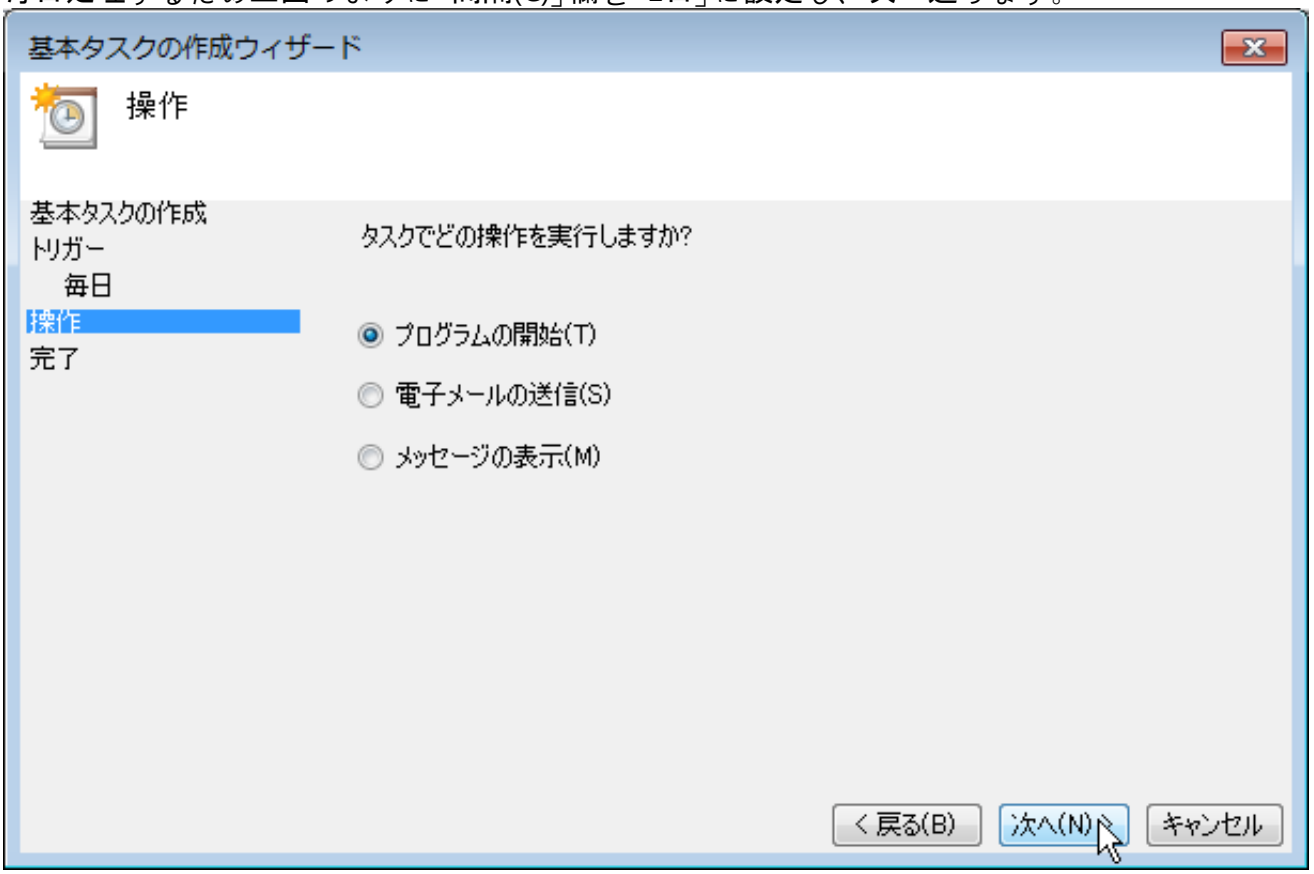


[Appendix.9 タスクトリガー設定]

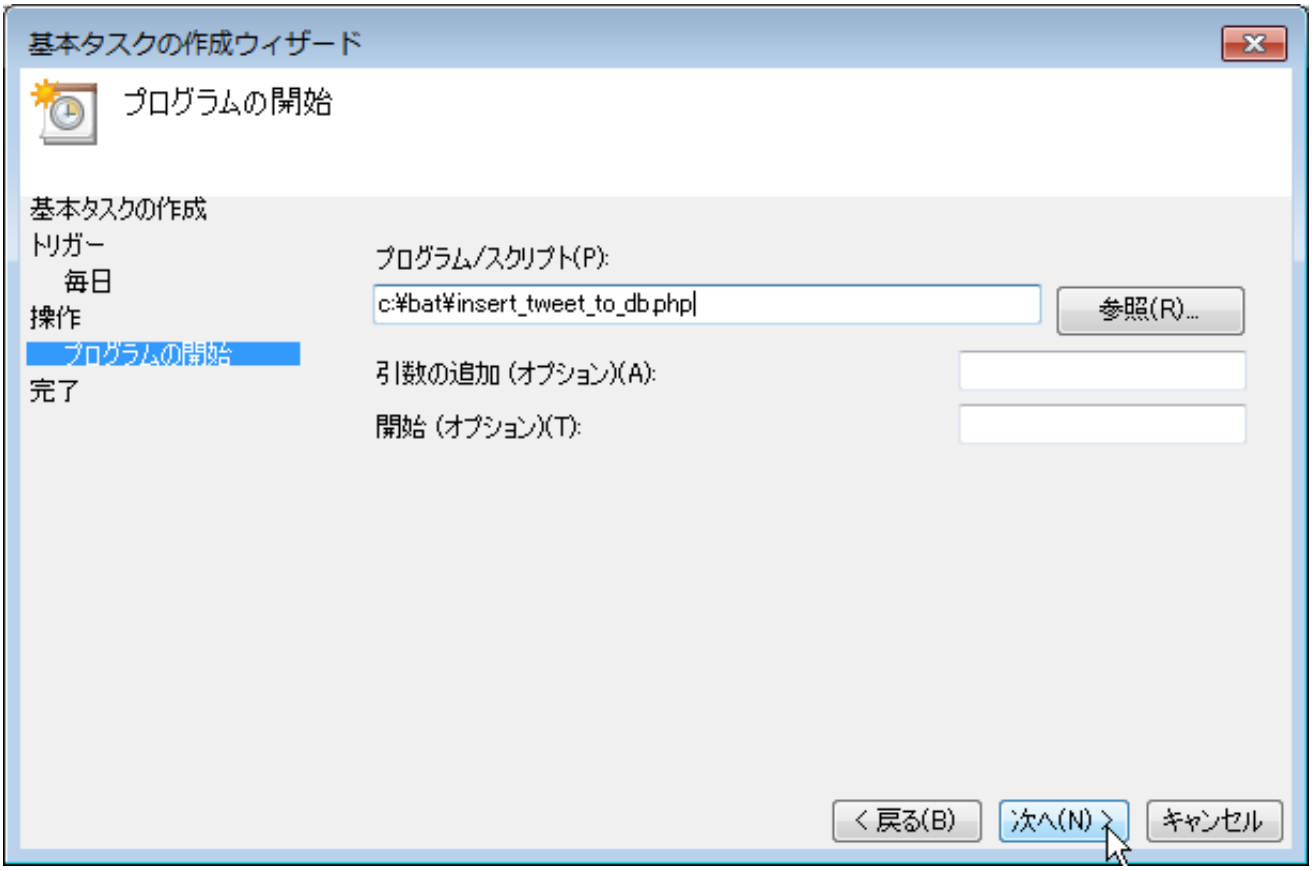
「基本タスクの作成ウィザード」画面が表示されますので、上図のように「毎日」をチェックして次へ進みます。



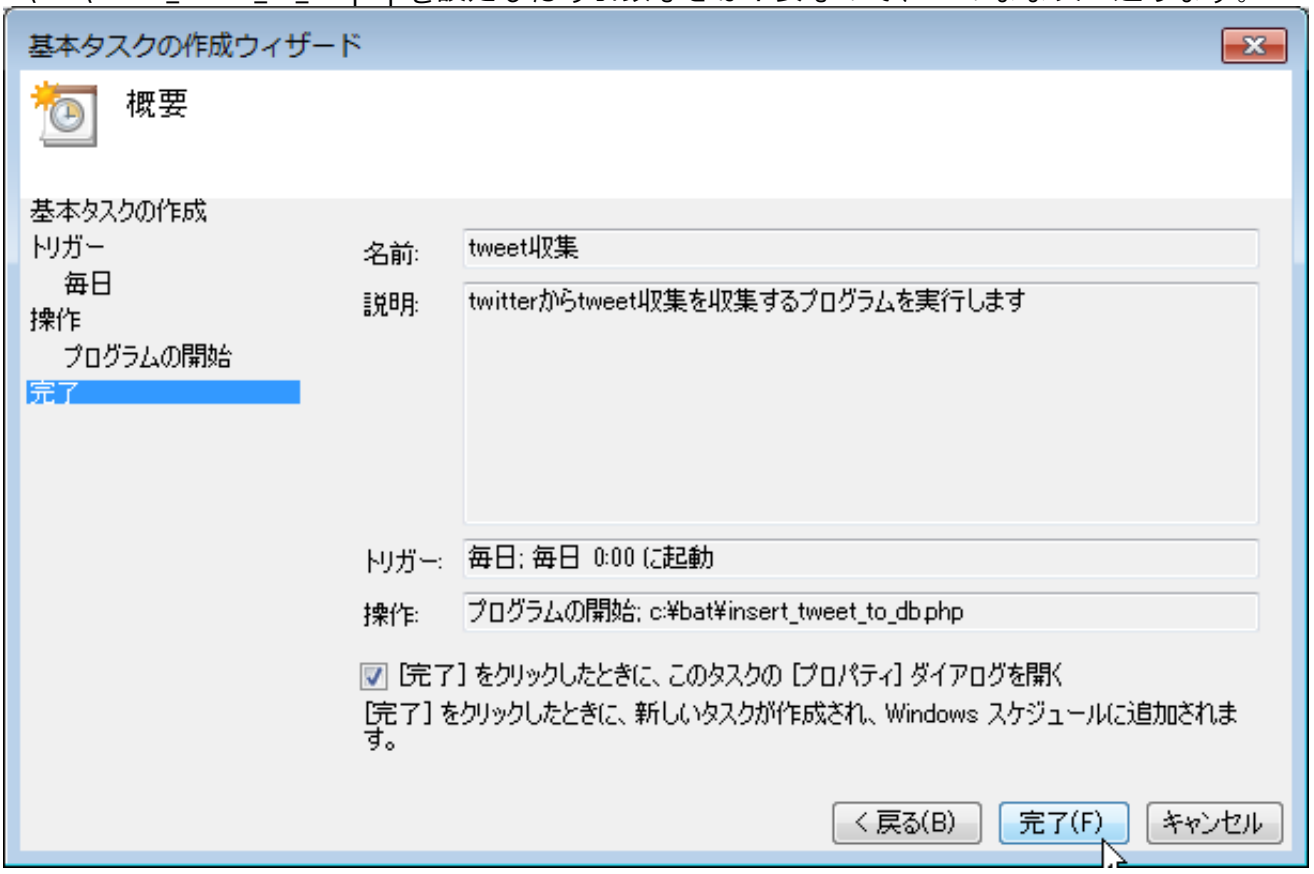
[Appendix.10 基本タスクの作成ウィザード]
毎日処理するため上図のように「間隔(C)」欄を「1日」に設定し、次へ進みます。



[Appendix.11 タスク操作選択]
今回設定するのはプログラムの処理であるため「プログラムの開始(T)」をチェックして次へ進みます。

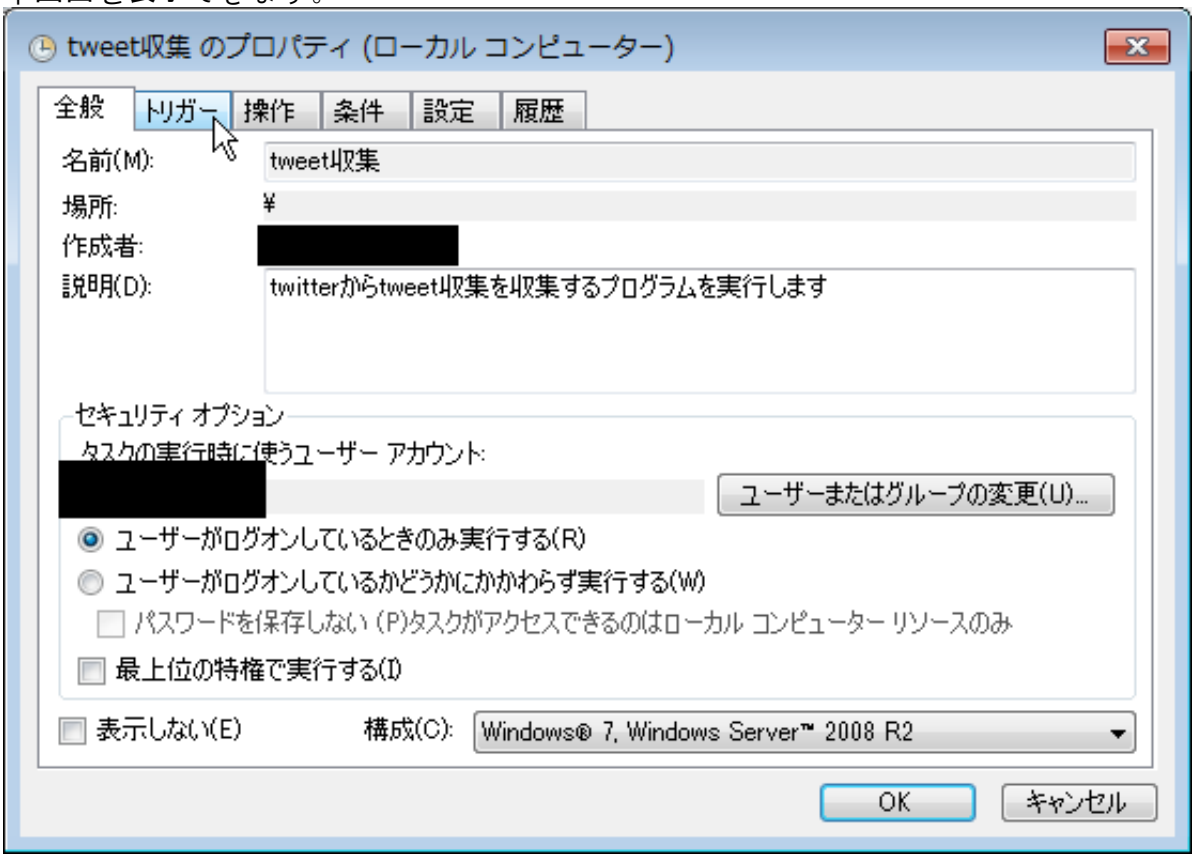


[Appendix.12 プログラム/スクリプト設定]
「参照」ボタンを左クリックし、自動実行するプログラムのファイルを指定します。今回は c:\bat\insert_tweet_to_db.phpを設定したら引数などは不要なので、このまま次へ進みます。



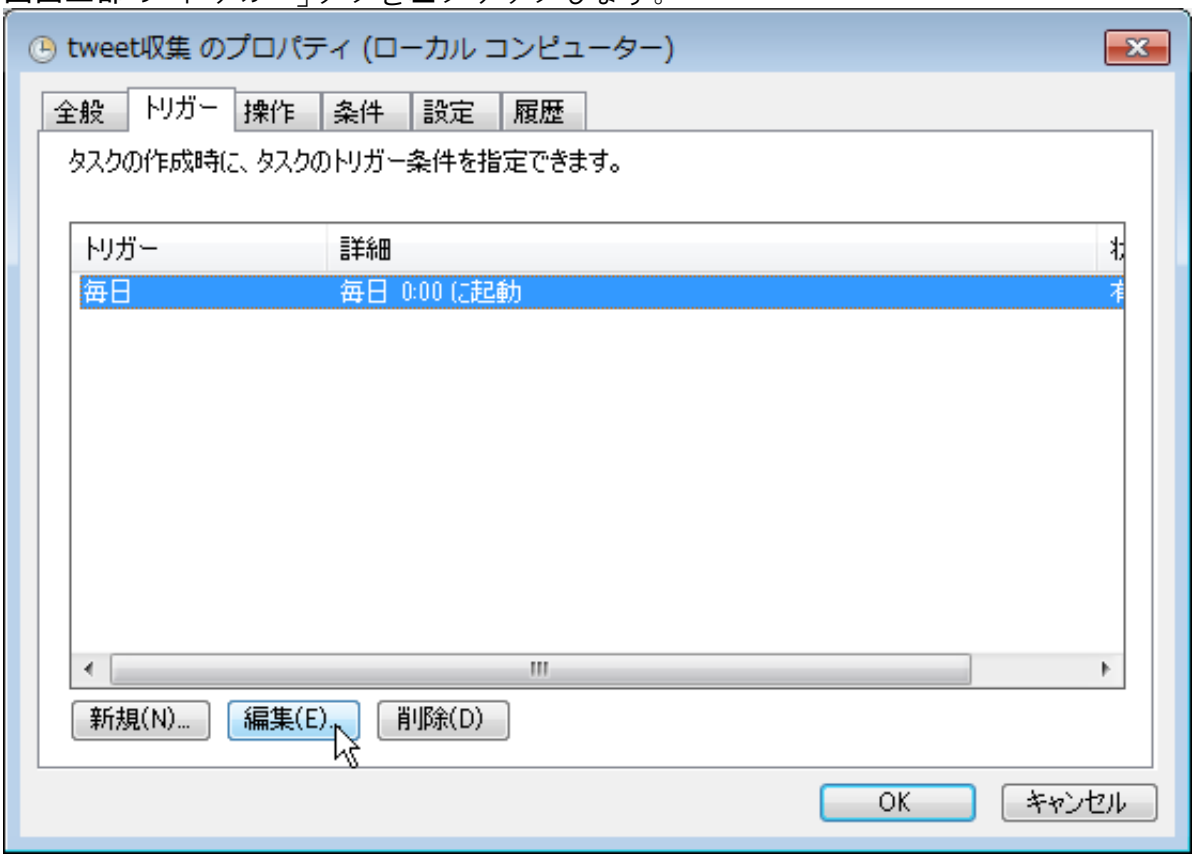
[Appendix.13 タスク設定完了]
上図のように（特に「トリガー」、「操作」欄が）設定されているか確認し、「[完了]をクリックしたときに、このタスクの[プロパティ]ダイアログを開く」にチェックを付けてから「完了(F)」ボタンを左クリックします。これで1日1回プログラムが自動実行されるようになりました。続いて、1日1回ではなく5分おきにプログラムを実行するように設定してみます。先ほどのチェックを付けたな

らば完了ボタンを押してすぐにプロパティ設定画面が表示されます。チェックし忘れた場合は、タスクスケジューラ起動時の画面で今回設定したタスクを右クリックすることによってプロパティ画面を表示できます。



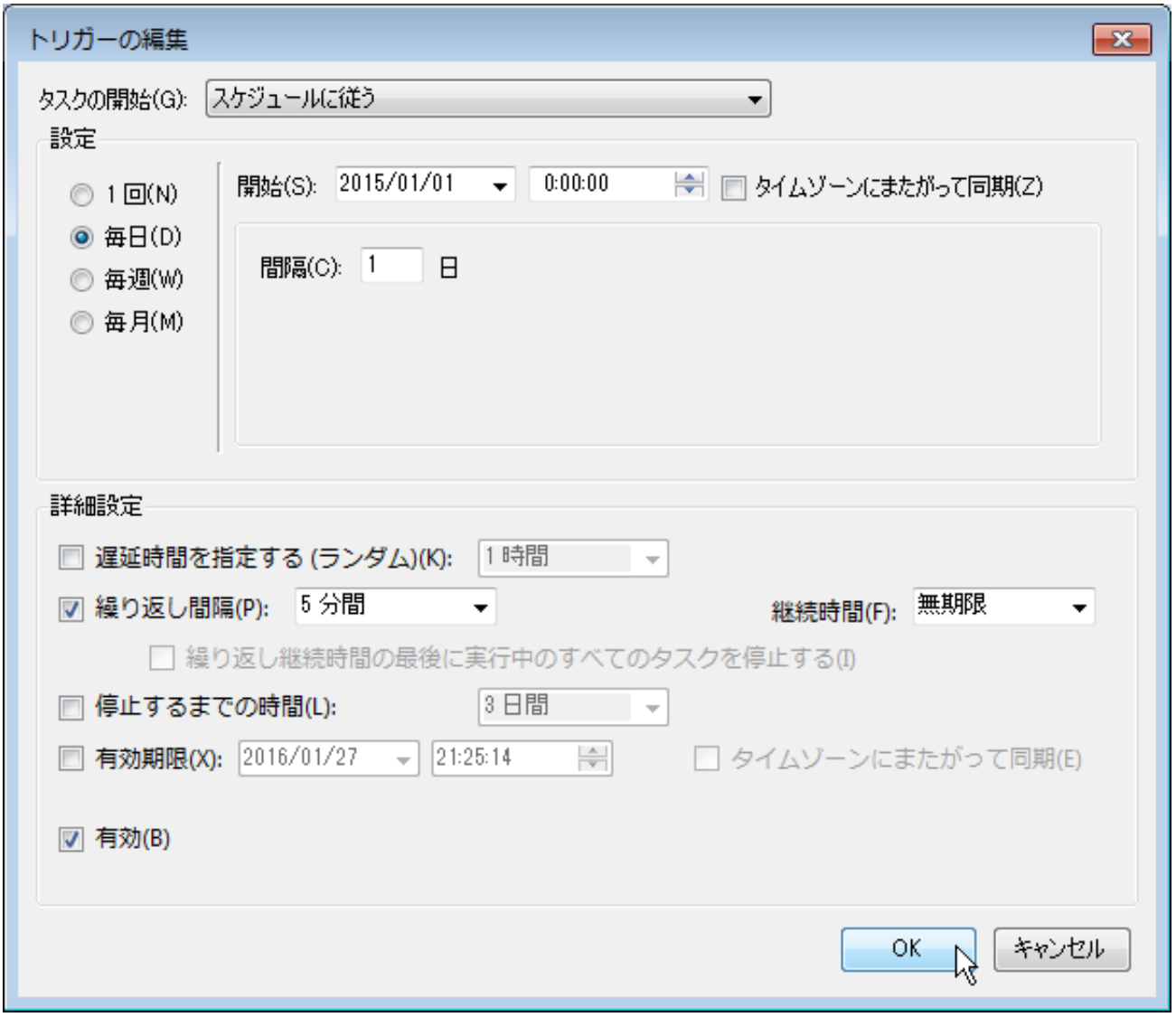
[Appendix.14 プロパティ設定]

画面上部の「トリガー」タブを左クリックします。



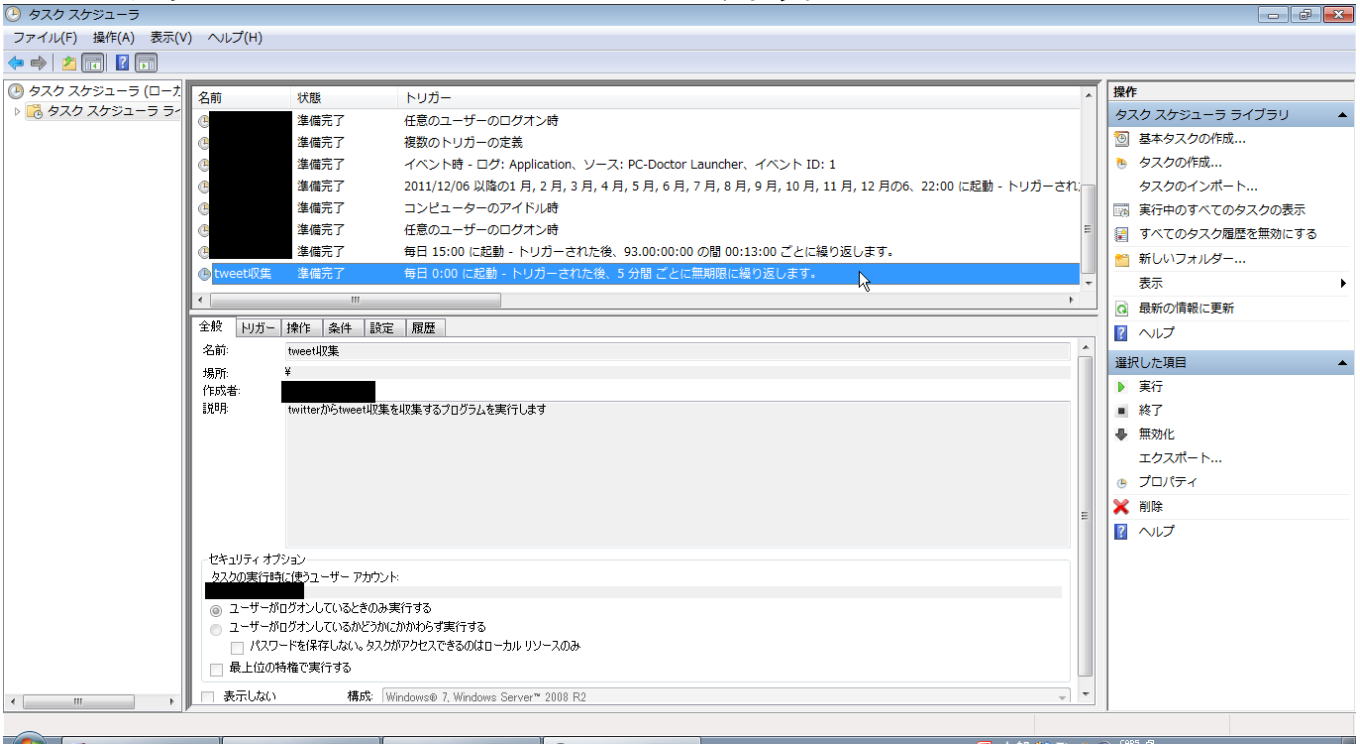
[Appendix.15 トリガー設定]

「編集(E)」ボタンを左クリックします。



[Appendix.16 トリガーの編集]

この画面で「繰り返し間隔(P)」を「5分間」に設定し、「継続時間(F)」を「無期限」、「有効(B)」にチェックを入れることで、設定後毎日5分ごとに自動実行されるようになります。「OK」ボタンを押すとこの画面からタスクスケジューラ起動時の画面に戻ります。



[Appendix.17 タスクスケジューラの設定確認]

この画面で設定内容が上図にように反映されていれば設定完了です。これにて毎日5分おきにツイートが収集されるようになりました。

同じようにして、日次で前日のtweet数を集計するタスクも設定しましょう。
まず、全データ日付の日次集計を行うseries.phpを拡張して前日の日次ツイート件数を集計するプログラムを作成し(※series_yesterday.php参照)、
次に、そのプログラムを1日1回実行するようにタスクスケジューラで設定します。
このように毎日自動で前日のツイート数合計を集計することで、最新の時系列データが折れ線グラフに反映されるようになります。
このタスクは毎日0時10分頃に1回だけ実行するよう設定すれば良いでしょう。

終わりに

簡単なものですが、本章の手順に従えば最低限の機能を持ったシステムが完成します。このような簡単なシステムですら決めるべき仕様や日々データの取得ミスやデータベースの肥大化に伴う対応などシステム運用について学ぶことが出来ます。最新のIT用語を散りばめた資料を読むだけではなく、一度簡単なものでもシステムを完成させる、システムを運用させるということの困難さを学ぶことで、分析ツールやBIツールの開発、また、施策実施時のエンジニアとのコミュニケーションをする際の貴重な経験になるでしょう。ぜひ実際に手を動かしてシステムを稼働させ続けて下さい。

参考書籍

本章ではPHPというプログラミング言語を用い、さらにデータベースを操作するSQLにも触れました。
PHPの入門書としては「岡本 雄樹：『イラストでよくわかるPHP はじめてのWebプログラミング入門』, インプレス（2013）」が分かり易いでしょう。また、初歩から一歩ずつ学ぶのと並行して「こういう処理をしたいけれど具体的にどうすれば…」と言う時に「鈴木 憲治 他：『PHP逆引きレシピ第2版』, 翔泳社（2013）」があると便利です。本格的に勉強したい場合は「小川 雄大 他：『パーフェクトPHP』, 技術評論社（2010）」を読むとWebプログラマとして活躍する一歩を踏み出せるでしょう。
SQLに関しては「アンク『SQLの絵本』, 翔泳社（2004）」がSQL未経験者にもハードルが低くてお薦めです。
もし余力がありセキュリティに興味がある場合は、本書の想定読者には難易度が高いと思いますが、「徳丸 浩 :『体系的に学ぶ 安全なWebアプリケーションの作り方』, ソフトバンククリエイティブ（2011）」という名著があり、これはプロのWebプログラマとしても読むべき重要な本です。
Webデータのデータ解析において、セキュリティは切っても切り離せない関係にあります。Webのデータは決して安全ではなく、SQLインジェクションと呼ばれる「データベースに(何の対策も取らずに)データを格納するだけでデータベースそのものが破壊される」ような攻撃を受けることもあります。また、攻撃への防御という面だけではなく、顧客情報を安全に管理するにはどうすればいいのかを知るという観点でも重要です。
本書のコードをそのまま利用するだけでも動くものは出来ますが、上記書籍を読み、将来的には自分用に発展させていけるようにしましょう。