○「 WebRequest()の使い方（その１） 」

2014.12.31

・アメンボです、

今回は長丁場になりそうです、たった１つの関数の調査・解説なので造作ないと思ったのが大間違いでした。((その４）ぐらいまで掛ってしまいそうです)

・待望の、

ＷＥＢ閲覧関数「WebRequest()」が build670 以降代らサポートされました、メタクウォーツ社のサンプルコードは確かに動くのですが、正直言って判らないことだらけで気持ちが悪い。(スッキリしません、使い方がイマイチ理解できない！のです)

・ただ、

アメンボはＷＥＢ関係の技術には疎いのですが、この気持ち悪さを解消するのには、かなり時間が掛ることだけは少しづつ判ってきました。

・と言うわけで、

基礎の基礎から始めることにします。また、解析を進めながら方法自体を考えることにしましたので、途中で道に迷うかもしれませんことをご容赦。(試行錯誤の予定です)

＜本稿で使用した MQL4 コード＞

※使用コード添付；「WebRequest_00.mq4」(そのまま添付)

※本稿は「MT4；version 4.00 Build745」「MetaEditor；version 5.00 Buid996」にて確認済み。

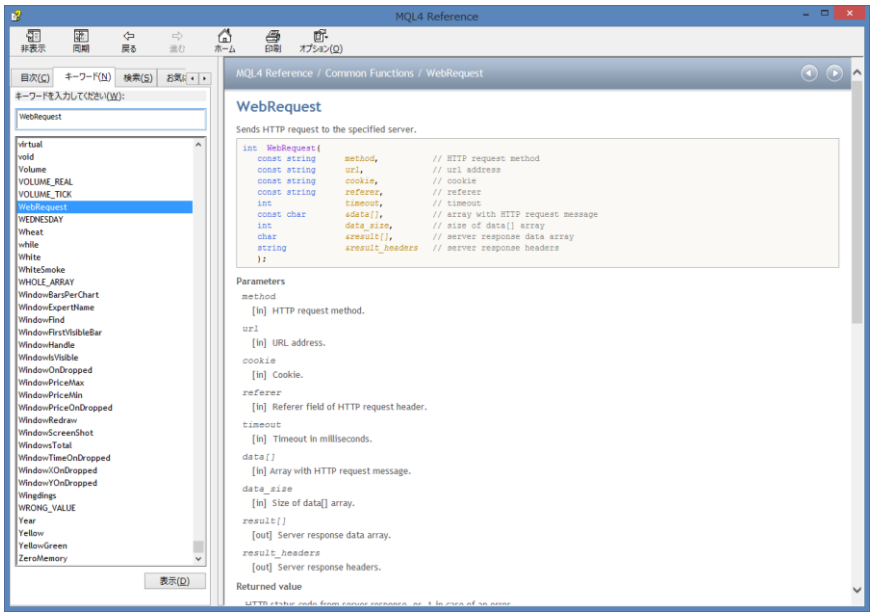
目次：

※本稿では「ツールと環境の準備のみ」で終わってしまいました、
WebRequest()関数の詳細な解析は次稿となります。(正直、こんなに手こずるとは！！)
メタクウォーツ社のサンプル・コードは直ぐ動くし、その結果のタグ・データも入手できるので、そのデータから必要な情報を入手する作業に入っても良いのですが、
WebRequest()関数自体が気になって、遥かな遠回りをすることにしました。

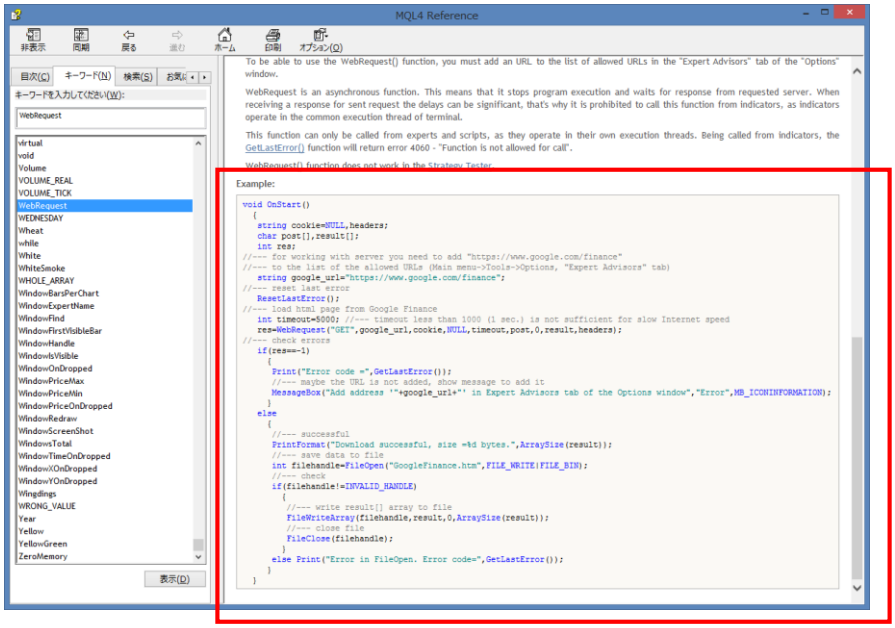
※本稿記載の「３ツール」は全て Web 上から入手した無料版です。(諸兄にて確認ください)

１．メタクウォーツ社のサンプル・コードを動かしてみた

・MetaEditor の画面で、［ヘルプ］－［MQL4 Reference］を選択することで表示される MQL4 Reference の［キーワード］タブで「WebRequest」で検索して得られる情報内に、サンプルコードがあります。



↓　スクロール



内容の吟味は後回しにして、このサンプルコード（スクリプト）を動かしてみました。

・コンパイルは Build670 以降の MT4 であれば通ります、
ただし動作させるには以下「(２)」に述べる URL リストへの追加が必要になります。

（1）コピーして作成したスクリプト；「WebRequest_00.mq4」

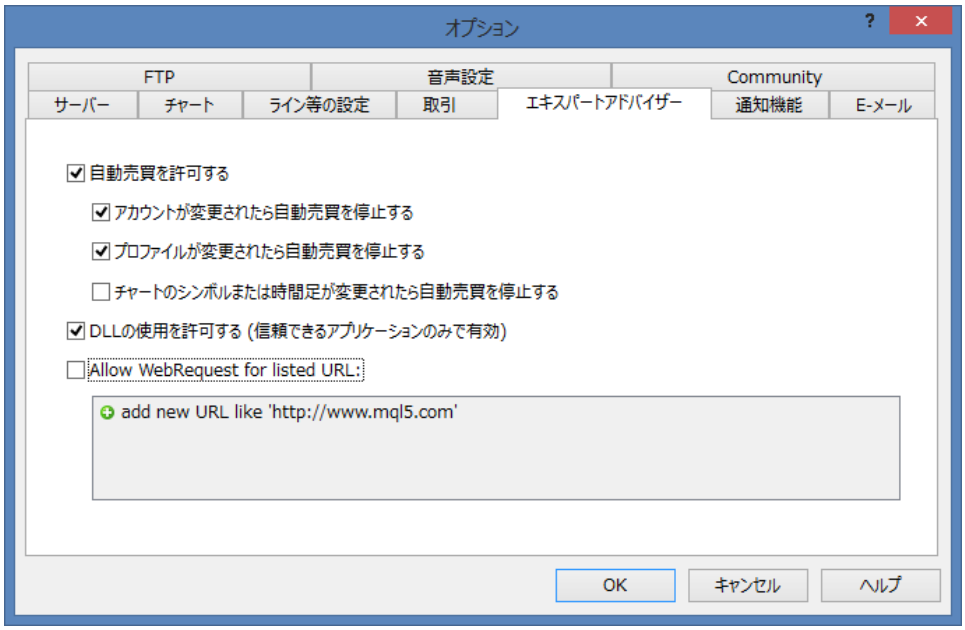
```
//+------------------------------------------------------------------+
//|                                                WebRequest_00.mq4 |
//|                                                           amenbo |
//|                                                 泉の森の弁財天池 |
//+------------------------------------------------------------------+
#property copyright "amenbo"
#property link      "泉の森の弁財天池"
#property version   "1.00"
#property strict
//+------------------------------------------------------------------+
//| Script program start function                                    |
//+------------------------------------------------------------------+
void OnStart()
  {
   string cookie=NULL,headers;
   char post[],result[];
   int res;
//--- for working with server you need to add "https://www.google.com/finance"
//--- to the list of the allowed URLs (Main menu->Tools->Options, "Expert Advisors" tab)
   string google_url="https://www.google.com/finance";
//--- reset last error
   ResetLastError();
//--- load html page from Google Finance
   int timeout=5000; //--- timeout less than 1000 (1 sec.) is not sufficient for slow Internet speed
   res=WebRequest("GET",google_url,cookie,NULL,timeout,post,0,result,headers);
//--- check errors
   if(res==-1)
     {
      Print("Error code =",GetLastError());
      //--- maybe the URL is not added, show message to add it
      MessageBox("Add address '"+google_url+"' in Expert Advisors tab of the Options window","Error",MB_ICONINFORMATION);
     }
   else
     {
      //--- successful
      PrintFormat("Download successful, size =%d bytes.",ArraySize(result));
      //--- save data to file
      int filehandle=FileOpen("GoogleFinance.htm",FILE_WRITE|FILE_BIN);
      //--- check
      if(filehandle!=INVALID_HANDLE)
        {
         //--- write result[] array to file
         FileWriteArray(filehandle,result,0,ArraySize(result));
         //--- close file
         FileClose(filehandle);
        }
      else Print("Error in FileOpen. Error code=",GetLastError());
     }
  }
```

（2）事前の設定；ＵＲＬリストへの追加手順

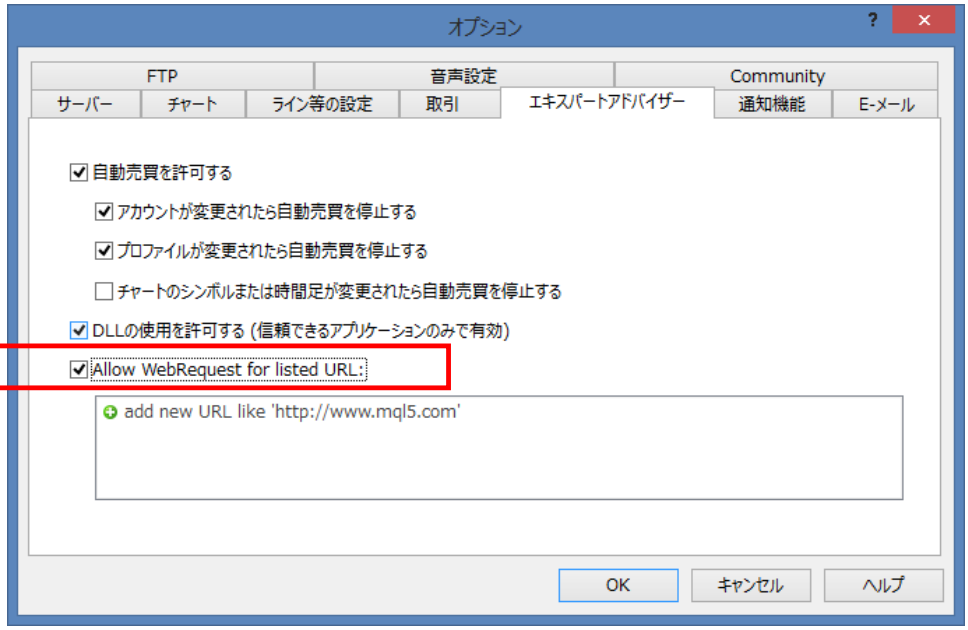
①［エキスパートアドバイザー］タブを表示する

サンプルコード中の記述（下記）に在るように、WebRequest()を動作させるためには［ツール］－［オプション］で表示される「オプション」設定の［エキスパートアドバイザー］タブを表示させ、アクセスするURLを事前に設定する必要があります。

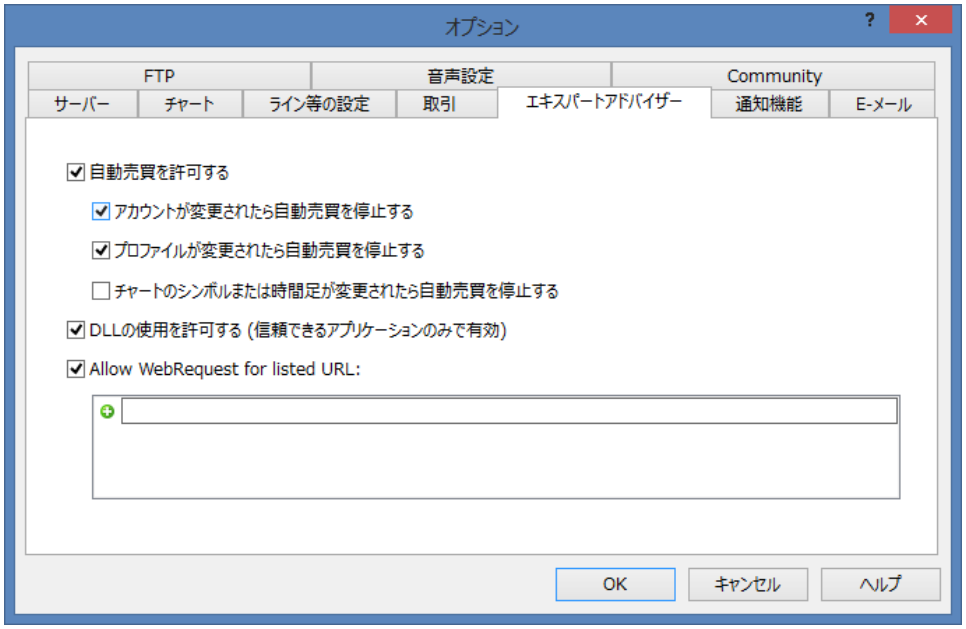
```
//--- for working with server you need to add "https://www.google.com/finance"
//--- to the list of the allowed URLs (Main menu->Tools->Options, "Expert Advisors" tab)
   string google_url="https://www.google.com/finance";
```



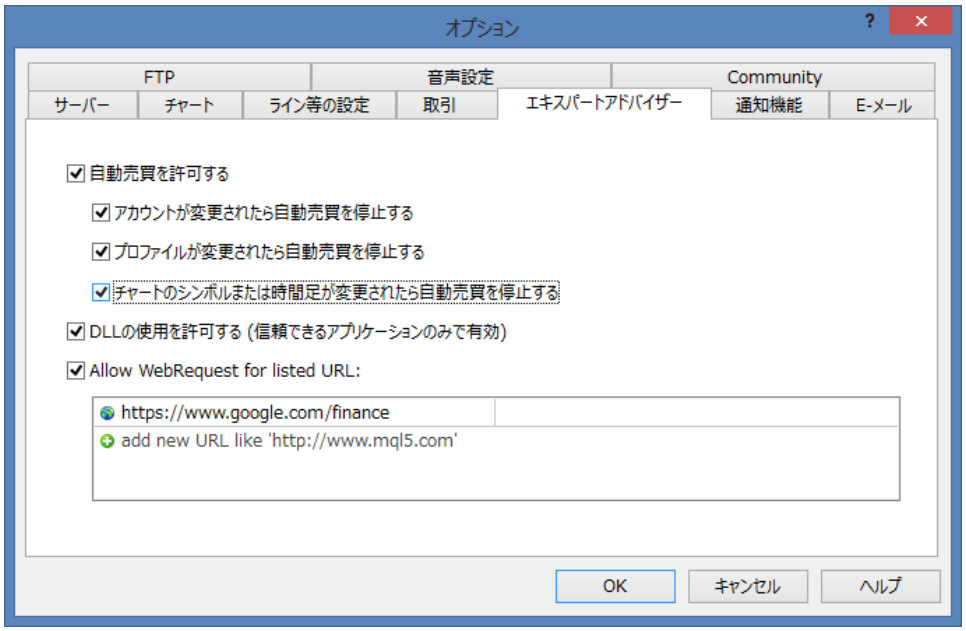
②［AllowWebRequest for listed URL:］にチェックを入れます



③「add new URL like'http://www.mql5.com'」部分をダブルクリックすると
リストへの入力が可能な状態になるので、このスクリプトでアクセスする
「 "https://www.google.com/finance" 」をリストに加えます。



↓

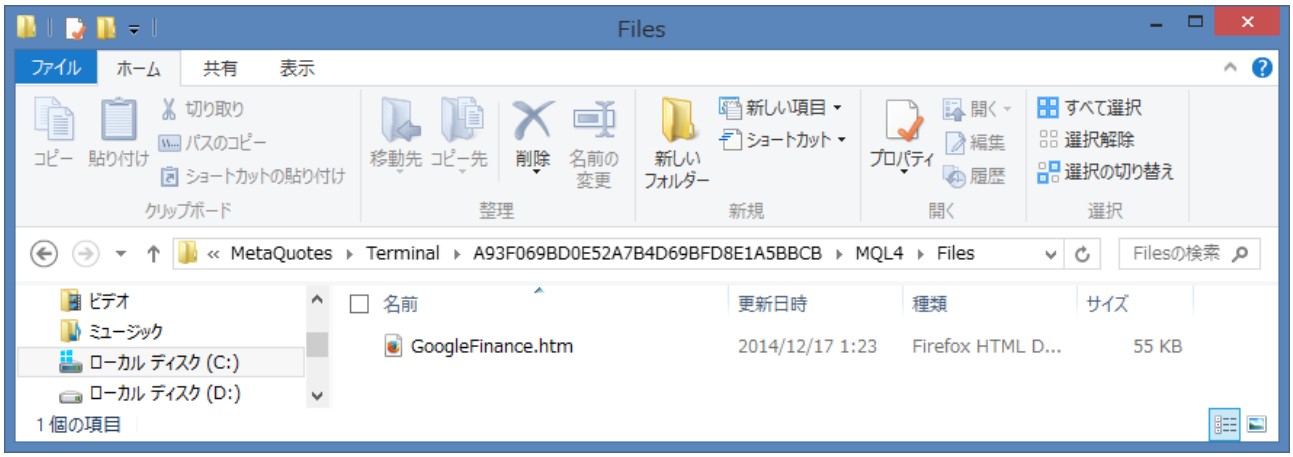


［ＯＫ］を選択して設定は完了です。

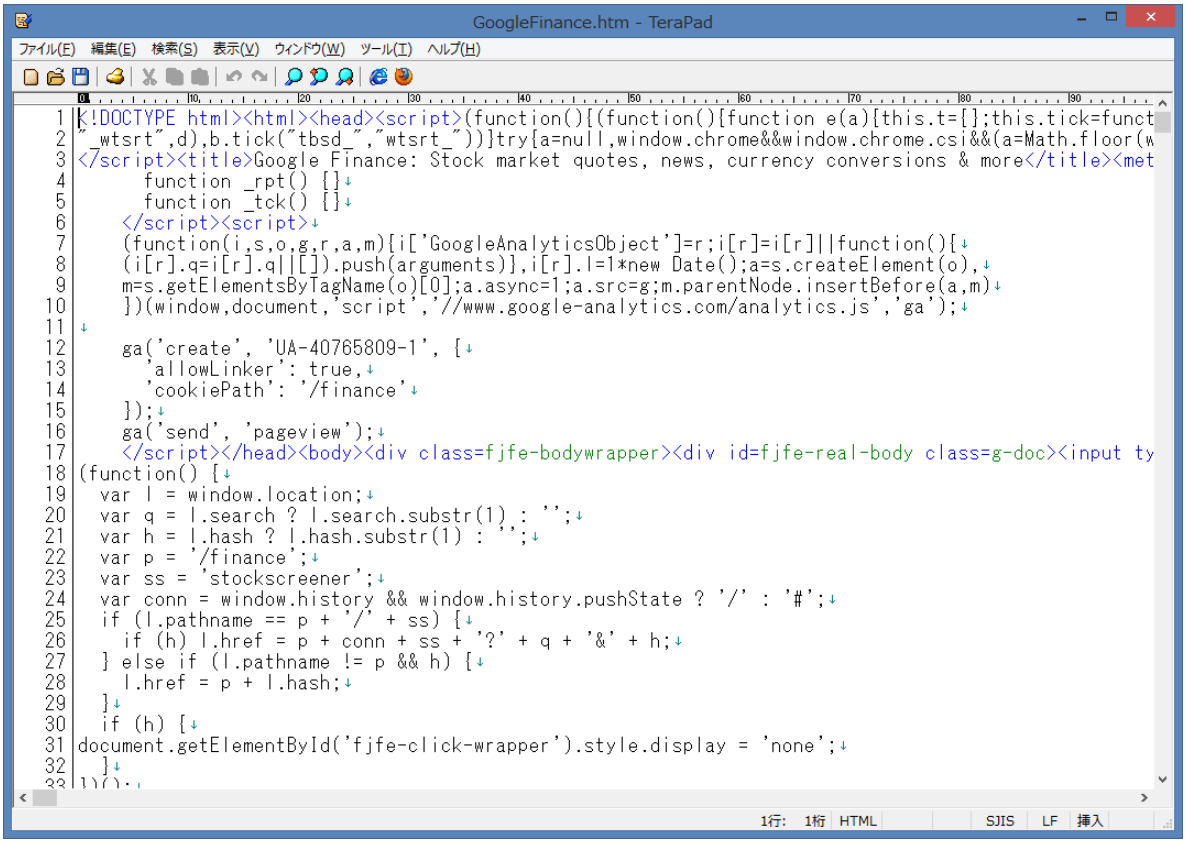
（3）「WebRequest_00.mq4」実行結果

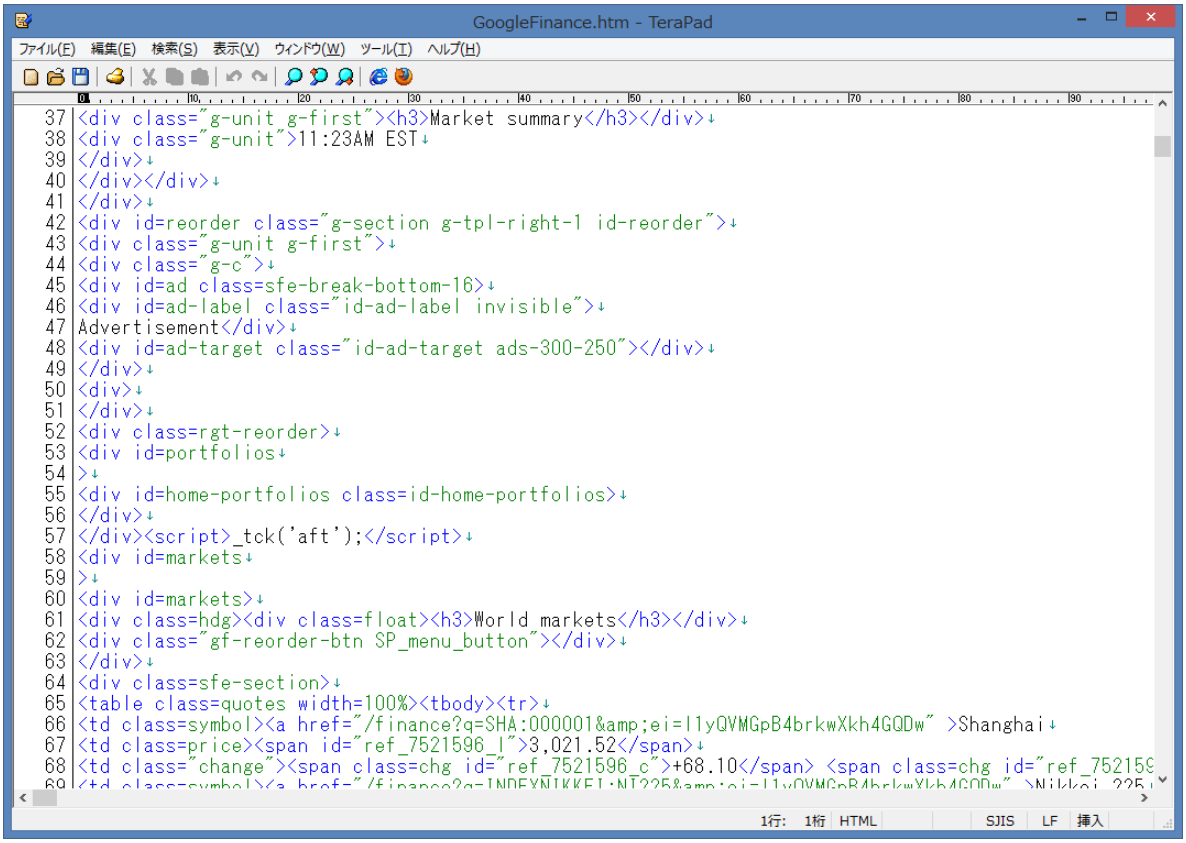
実行直後に[エキスパート]タブを観ると、

実行後、「MQL4¥Files」フォルダー中に「GoogleFinance.htm」と言うファイルが作られます。



①このファイルをテキスト・エディタで開いてみます



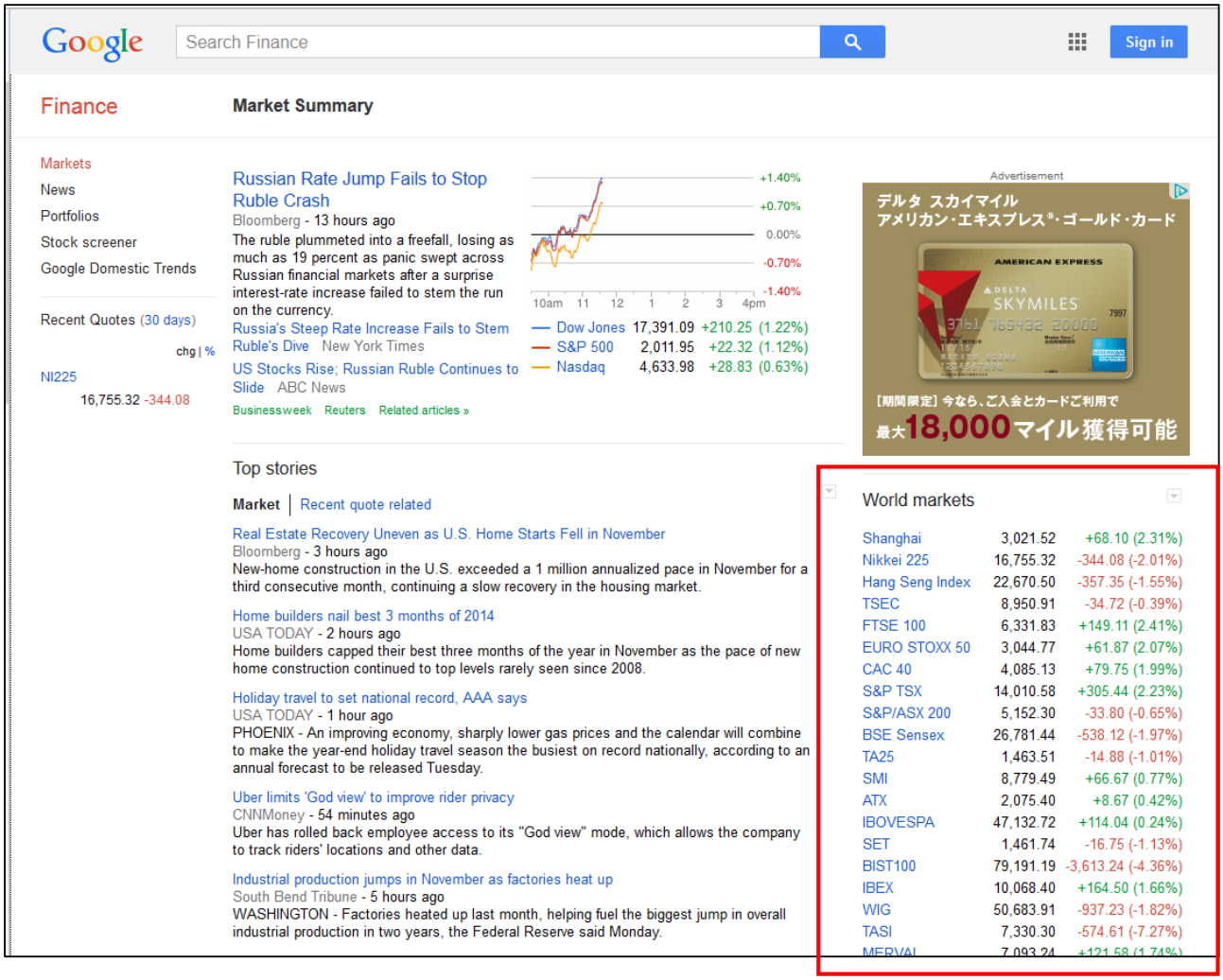


・HTML ファイル（タグで構成）であることが判ります。

②次に、このファイルをダブルクリックしてみます。

※なんか、すごく中途半端な表示です！

③比較のために、直接「https://www.google.com/finance」をアクセスしてみます



※赤枠でしめす「World markets」のデータが、「MQL4¥Files」フォルダー中の「GoogleFinance.htm」ファイル内に取り込まれていることが判ります。

（4）WebRequest()関数を解析していく手順を決めた

※上記の結果から、これから解析・調査していく手順を「現状では」下記の様に決めました。

ステップ１；　解析環境とツール類の整備　・・本稿

ステップ２；　WebRequest()関数の解析　・・次稿の予定

ステップ３；　ターゲットから得られるタグ・データの中から、必要な情報を入手する

ステップ４；　得られた情報をＥＡの判断条件に加える方法について考察する
（グローバル変数にセット、ファイルにセットし読み取る）

※本稿の記載内容は、「ステップ１」です。

2．HTTP プロトコルの基礎をおさらいする

・WebRequest()の使い方を理解するために最低限必要と思われる基礎知識をおさらいすることにします。（なにせ、アメンボはこの分野に関しては初心者ですので、おさらいが必要でした）

（1）サーチエンジンと使い方の確認

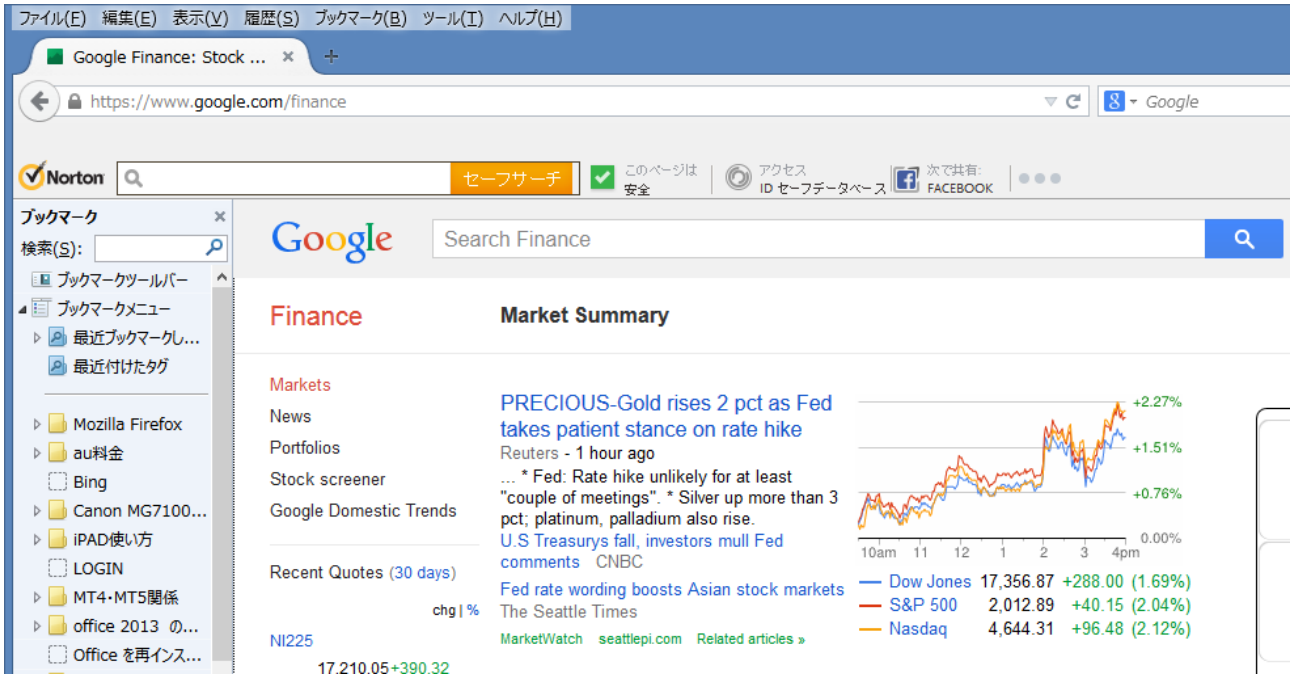
・普段何気なく使っていて「なんで？」と思いながらも、それ以上深くは追及しなかったものにサーチエンジンでの検索設定がありました。（アメンボの場合ですが！）

※そこで、少し解析を進めるために簡単な実験をしてみます。(本当に初歩の初歩から始めます)
検索対象はサンプルで使った「//www.google.com/finance」とします。

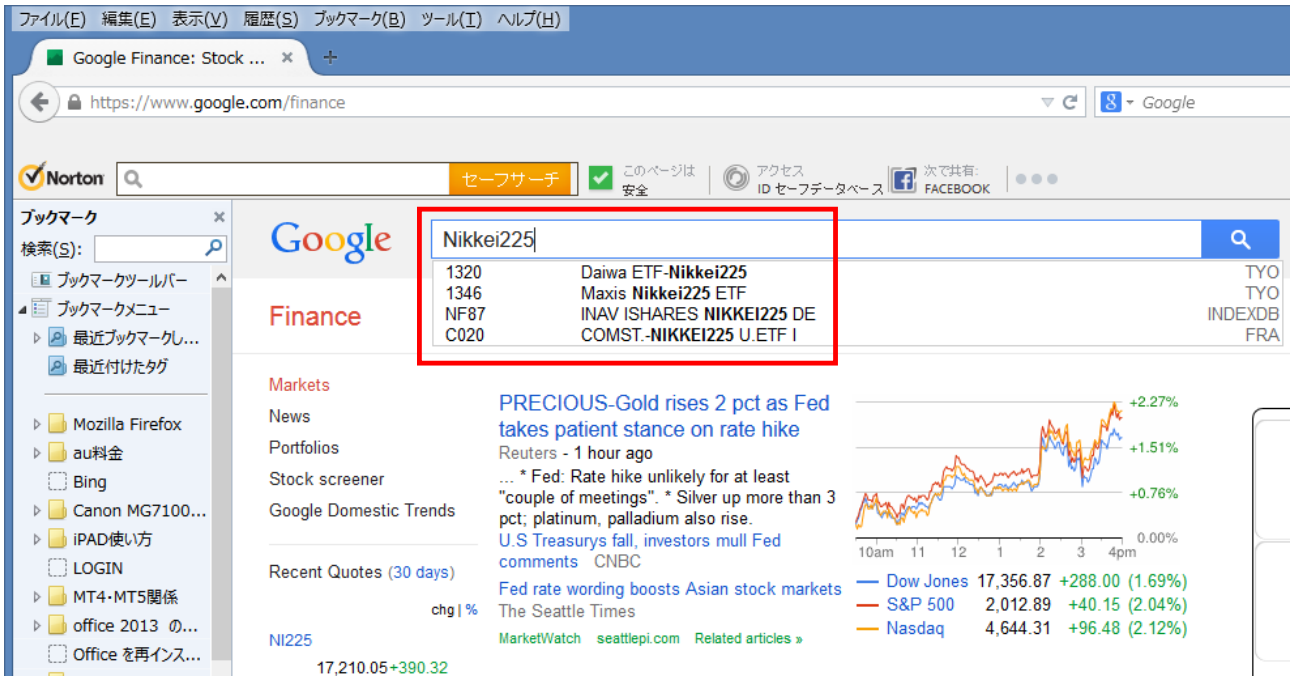
ルート１；　検索エンジンを間接的に使う

①「https://www.google.com/finance」を入力し、Return

↓

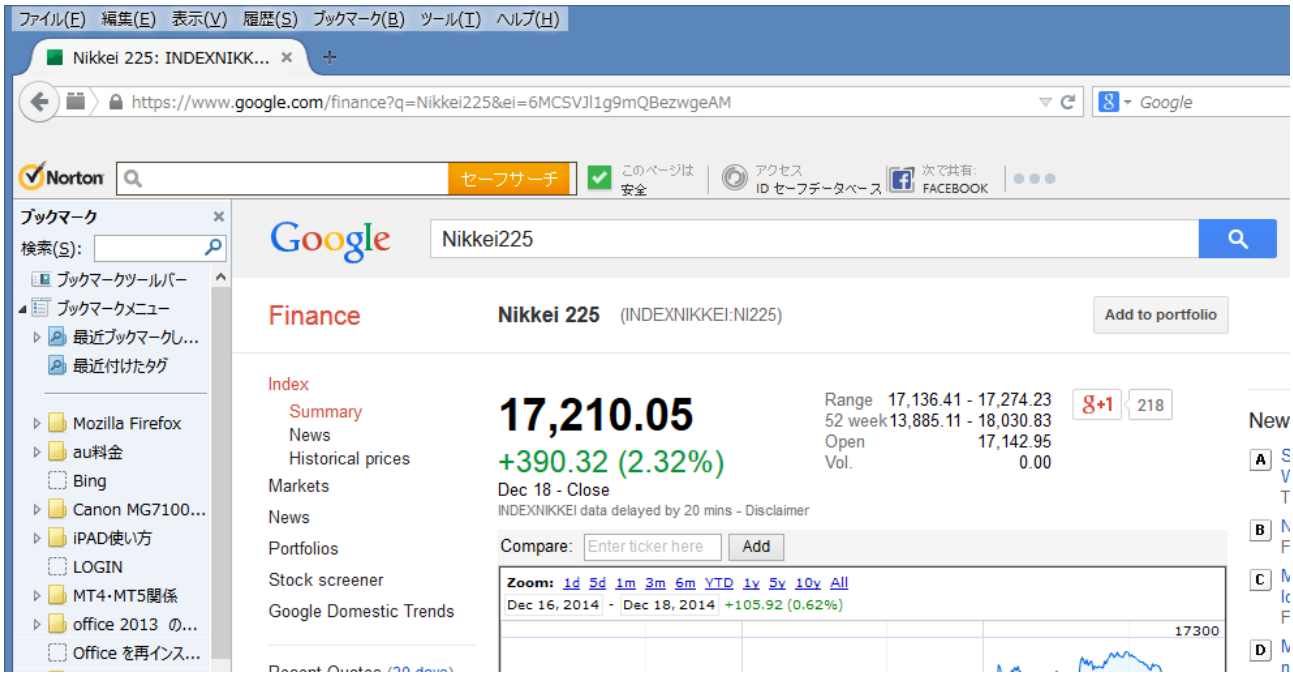


②Google の検索窓（検索エンジン）に「Nikkei225」を入力して、Return



↓

③日経インデックス 225 のページが表示されます

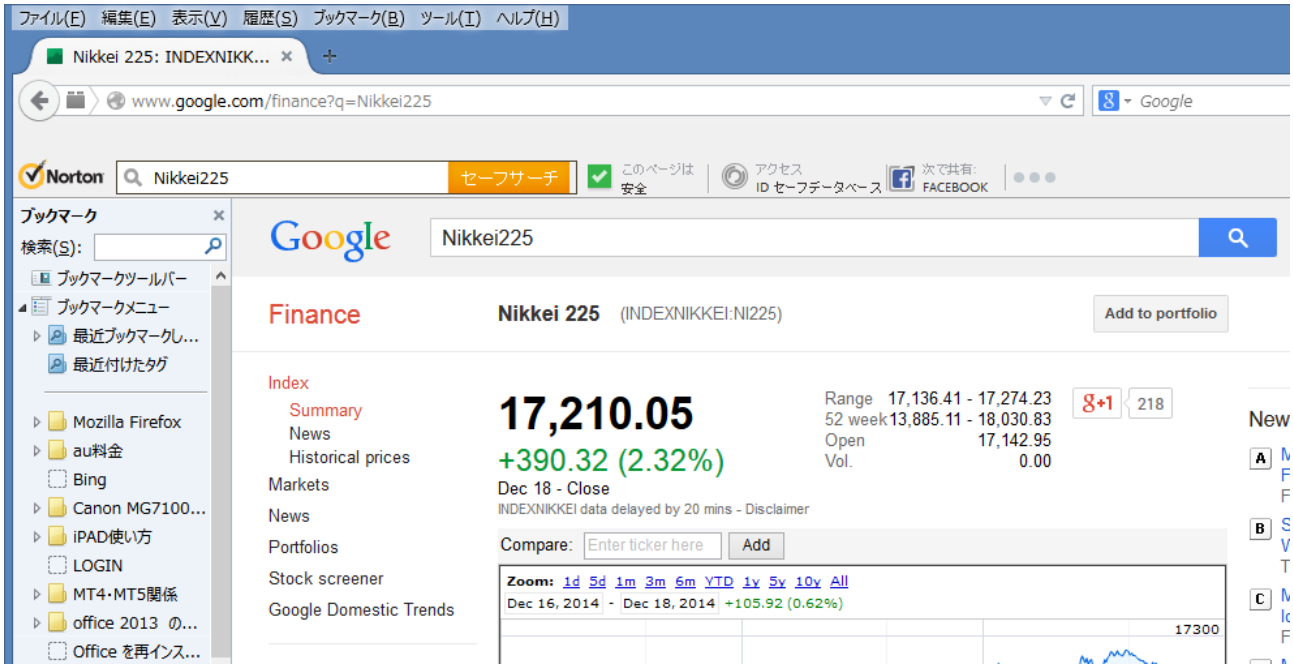


注目；「https://www.google.com/finance?q=Nikkei225&ei=6MCSVJl1g9mQBezwgeAM」
がアクセスされている！

ルート２；　　検索エンジンを直接に使ってみる

①「https://www.google.com/finance?q=Nikkei225」でアクセスする

↓



「ルート１」を辿った場合と同じページにたどり着きます。

まとめ；

実際に試してみた結果、検索エンジンを直接に使う場合、Google の場合は下記のいずれでもＯＫでした、「?q=」は検索キー指定に使う指示式です。（用語として正しいかは自信なし）

①「//www.google.com/フォルダ名?q=検索用語（キー)」の形式

②「//www.google.com/search?q=検索用語」の形式

「②」の場合、「search」は検索（サーチ）エンジン名を指定しているようですが、詳しいことまでは確認出来ていません。

基本形＝http://[ホスト名：ポート番号]/［パス］［クエリー（?q=「検索対象用語」）］
（ポート番号「80」の場合は省略可能）

実際に色々な検索サイト中の「検索（サーチ）エンジン」を直接に使う（アクセスする）場合で試してみた結果をまとめます。

検索サイト	検索キー指定式	
	実際に試すと表示されるデータ	これでも動く
ヤフー	/search;_ylt=A3xTs0NpSohUtj0AwbuJBtF7?p=	/search;?p=
グーグル	/?gws_rd=ssl#q=	/search?q=
楽天	/Web?qt=	/WebIS?qt=
グー	/web.jsp?IE=UTF-8&MT=	/?MT=では 2 バイト文字が NG

※「グー」の場合は「IE=UTF-8」を入れないと日本語（２バイト文字コード）が文字化けしてしまいました。

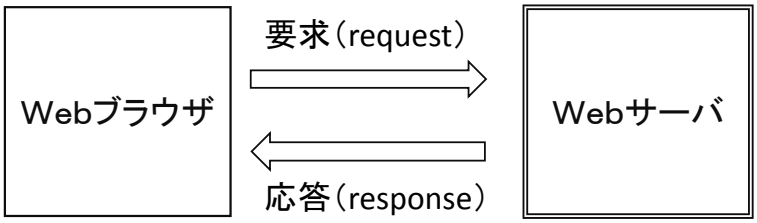
・ホスト名は、以下です。

検索サイト	//ホスト名
ヤフー	//www.yahoo.co.jp
グーグル	//www.google.co.jp
楽天	//websearch.rakuten.co.jp
グー	//search.goo.ne.jp

（2）HTTP プロトコル（特に GET と POST）

・アメンボは HTTP プロトコルに関して、超初心者レベルですので少し詳しい内容については、解析・実験を進める中で理解していくことにします。
実を言うと、理解していない部分が多すぎるので、進めながら（歩きながら）考える手法をとることにしたと言うのが真相です。（詳しい方は、本節を読み飛ばしてください）

・基本中の基本の再確認；



ご存知のように、Ｗｅｂブラウザで何処かのページを見るということは、上図のように
Ｗｅｂサーバーに要求(request)を発行し、応答(response)を受け取る処理でもあります。
（本稿では、接続確立は既にされているとの前提で解説します）
ただ、我々が見ているＷｅｂページは知らぬ間に複数回の「要求と応答」の繰り返しで
完成されている場合が殆どです。
HTTP プロトコルとは、この通信手順のことですので、まず要求（request）から
おさらいします。

要求（request）の形式；

サーバーとの接続確立後に発行する要求（request）の一般形

リクエスト行	；GET、POST など要求の種類
ヘッダ	；
空行	；必ず１行空ける
本文	；無いこともある（GET では無い、POST では在る）

本節では主に「リクエスト行」と「本文」について解説します。

[リクエスト行]； 書式は「 ［メソッド］ URI HTTP のバージョン 」(

例 GET /******/**/index.html HTTP/1.1

メソッドの種類（抜粋）

メソッド	内 容
GET	URI の内容を取得する
POST	URI に情報を送信する
HEAD	URI についてのヘッダ部分のみを取得する
PUT	URI の内容を作成、置換する
DELETE	URI の内容を削除する
OPTIONS	URI に対して利用できるメソッドの一覧を取得する
TRACE	クライアントからの要求をそのまま返す

本節では、HTML のフォームから「サーチエンジン」や「掲示板」にユーザー入力を
渡す場合、つまりデータを送信する場合での「GET、POST」の違いのみを確認して
おきます。

①GET メソッドの場合；

書式は「 GET URI + ?[クエリー（送信データ）] 」

例 GET /******/**.**？入力１＝設定値Ａ＆入力２＝設定値Ｂ HTTP/1.1

判りやすく書換えると、

GET
/******/**.**
？
入力１＝設定値Ａ
＆
入力２＝設定値Ｂ
HTTP/1.1

です。

・[本文]は内容がありません（空っぽ）

※「(１)」の例の様に、URI をＷＥＢブラウザで直接に入力する場合は、「HTTP/1.1」が無くても動きます。

②POST メソッドの場合；

書式； POST /******/**.** HTTP/1.1

・[本文]には「送信データ」が入ります。

GET では URI の後ろにくっ付いていた送信データが、POST の場合は[本文]に設定されます。

例； 入力１＝設定値Ａ＆入力２＝設定値Ｂ

判りやすく書換えると、

入力１＝設定値Ａ
＆
入力２＝設定値Ｂ

です。

この文字列を「クエリー」と呼びます。

応答（response）の形式；

ブラウザからの要求（request）に対して、サーバーが返す応答（response）の一般形

ステータス行	；応答の種類
ヘッダ	；本文内容の識別 ID、日付、種類、サイズなど
空行	；必ず１行空ける
本文	；応答内容の本体

本節では主に「ステータス行」と「ヘッダ」について解説します。

[ステータス行]； 書式は「 HTTP のバージョン ［ステータス・コード］ 解説文 」

例 HTTP/1.1 200 OK

ステータス・コードの種類概要（抜粋）

ステータス・コード	内　容
200 番台	要求が正常に処理された
400 番台	クライアント側に問題あり
500 番台	サーバー側に問題あり

ステータス・コード例（抜粋）

ステータス・コード	表示「解説文」	概要
200	OK	アクセス成功
403	Forbidden	URI に対するアクセスが拒否された
404	Not Found	URI に該当するものが無い
500	Internal Server Error	リクエストの実行中にエラーが発生した

［ヘッダ］;　書式は「 ［フィールド名］ ： 値 」

例　Content-Type: text/html

Content-Type は本文内容の種類（MIME 規格）を示します

MIME（Multipurpose Internet Mail Extension）規格； 通称「マイム」

US-ASCII のテキストしか使用できないインターネットの電子メールで

さまざまなフォーマット（書式）を扱えるようにする規格。

MIME の種類（抜粋）

値	内　容
text/plain	プレーンテキスト
text/html	HTML テキスト
application/xhtml+xml	XHTML テキスト
image/gif	GIF 画像
image/png	JPEG 画像
video/mpeg	MPEG 動画
application/octet-stream	任意のバイナリデータ
application/pdf	PDF 文書

※要求（request）と応答（response）の具体例は、「３.」以降の

「実際の動作確認」の中で解説することにします。

ポートＮｏについて；

プロトコルで使用するポートが異なります、下記に代表的なポートＮｏを記載。

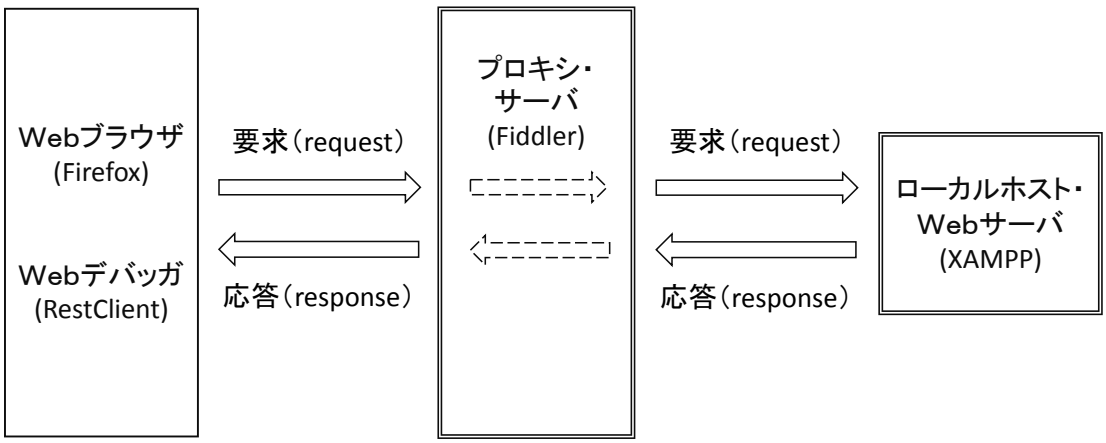
プロトコル	ポートＮｏ	用途
HTTP	80	WEB 通信
HTTPS	443	SSL 通信
SMTP	25	メール送信
POP	110	メール受信

３．解析に使用するツールの動作確認を行う

HTTP プロトコルには全く土地勘がないので、動作解析を開始するにあたり使うかもしれないツールは全て揃えておくことにしました。
（従って、結果として使用しないツールもあるかもしれません）
また下記以外にも試してみたものが何個かある（「telenet」など）のですが、チョット触ってみて、アメンボには手におえそうもないツールや操作が難しいものは除外しました。
アメンボは、ＷＥＢブラウザに「Firefox」をデバッカには「RestCliant」を使っていますが、これは好みの問題ですので読者は自由に選んでください。
ブラウザに「Internet Explorer」などを使う場合では、利用可能なデバッカは本稿に記述したもの（RestClient）とは別のものになります。
またホスト・サーバーは「XAMPP」、プロキシ・サーバーには「Fiddler」が特にお勧めですが、「Fiddler」を使えるようにするためにブラウザ側の設定が必要になる場合があります。
（ネット上に情報が溢れかえっているので、参照ください）

（１）全体像（解析システム構成）

WebRequest()が発行・受理する HTTP プロトコルをモニター、解析するためのツール全体構成は下記としました。
「ローカルサーバー」を使うのは、WebRequest()のアクセス先を単純なサーバー（プログラム）にして、解析を容易にするためです。（現状の意図）

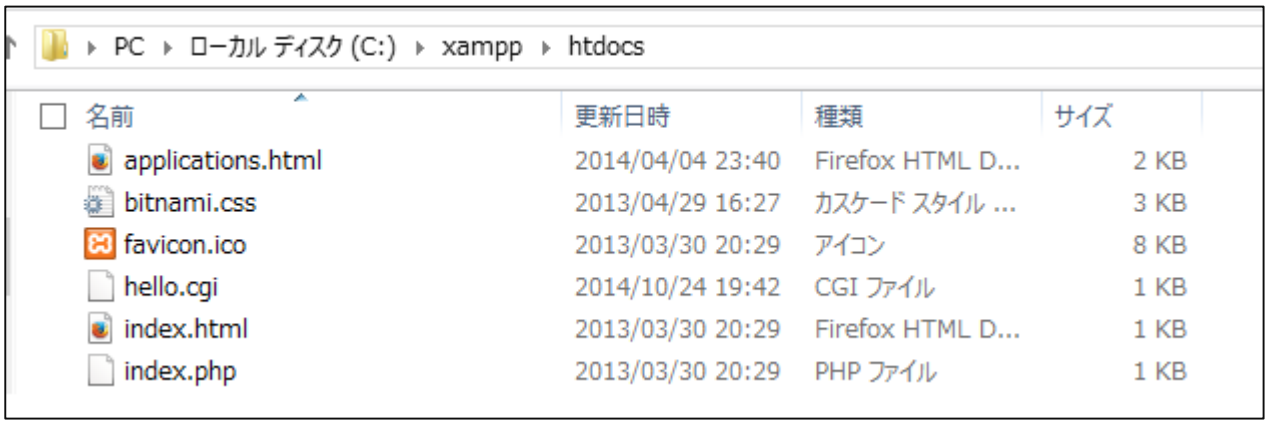


・Ｗｅｂブラウザ　；Firefox を使いました（単純に好みの問題）
・Ｗｅｂデバッカ　；RestClient を選択しました、理由は Firefox 用のアドオンだったからで、それ以外の理由はありません。
・プロキシサーバー　；Fiddler を使います、何といっても驚くほど高機能だからです。
（でもアメンボは今回初めて使います）
・ローカルサーバー　；XAMPP を使います、Apach の簡易版で定評があるからです。
（昔、興味があって Apach と tomcat をチョットだけ触ったので）
※Ｗｅｂブラウザとデバッカは、読者の好みで選択してください。
※各ツールの設定方法は本稿では解説しませんので、ネット上等の参考情報を参考にしてください。

（2）ローカルホスト・サーバー（XAMPP）

XAMPP が正常にインストールされると、アプリ一覧に下記の「XAMPP Control Panel」が表示されますので、[右クリック]－[スタート画面にピン留めする]で使いやすくしておきます。

また XAMPP ではインストールしたときから、「CGI」や「PHP」が使えます、CGI 記述用の perl や php がすぐ使える用に設定されています。（すごい！）
本稿では、アメンボは php を使っていきます。（perl でも良いのですが、php の方が簡単）
興味のある方は、動作確認に「indx.cgi」も加えてみてください。
「xampp¥htdocs」の中身の例；

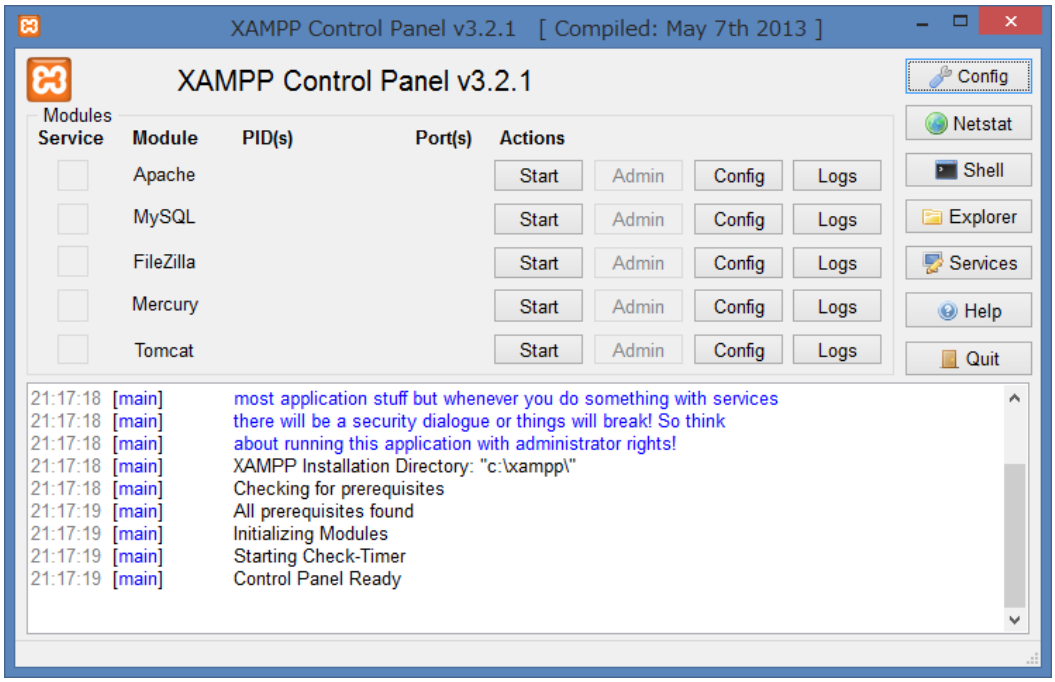


※「index.html」「index.php」「hello.cgi」ファイル等が初めから設定されています。

動作確認；

①「XAMPP Control Panel」のアイコンをダブルクリックする

当然と言えばそれまでですが、コントロール・パネルが表示されます

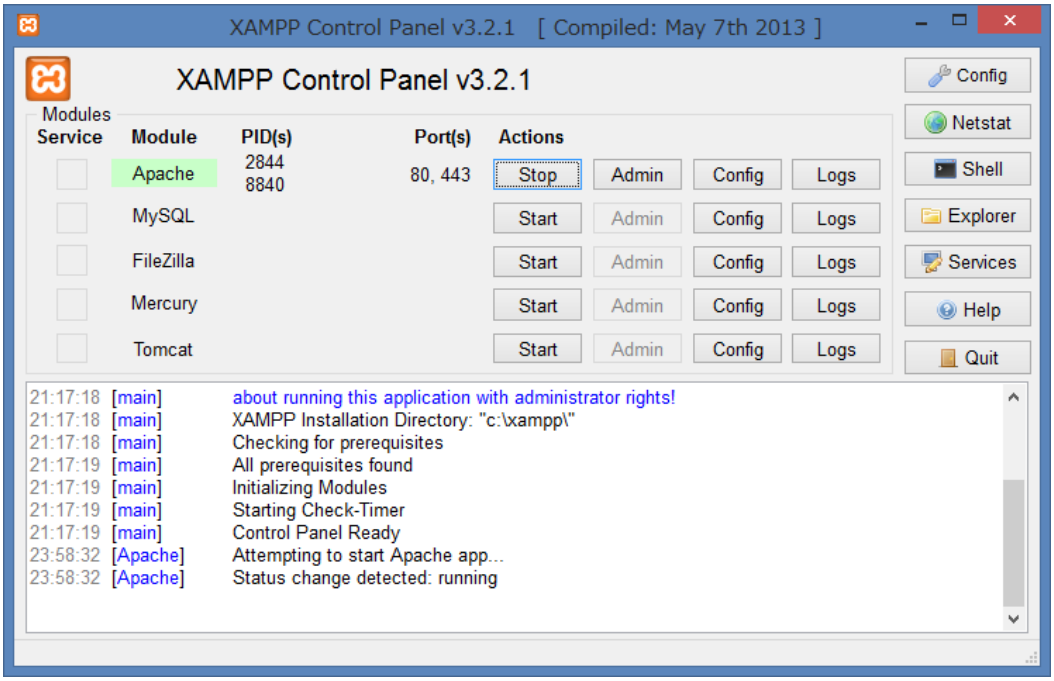


・バージョンや、インストール時の Module 選択により異なりますが、一般的には Apach（サーバー）、MySQL（データーベース）などがインストールされています。

②Apach（サーバー）をスタートする

Apach の[Start]をクリックします、

[Apach]の背景色が緑になって、同時に[Start]⇒[Stop]と変わればローカルホスト・サーバーが動きだしています。

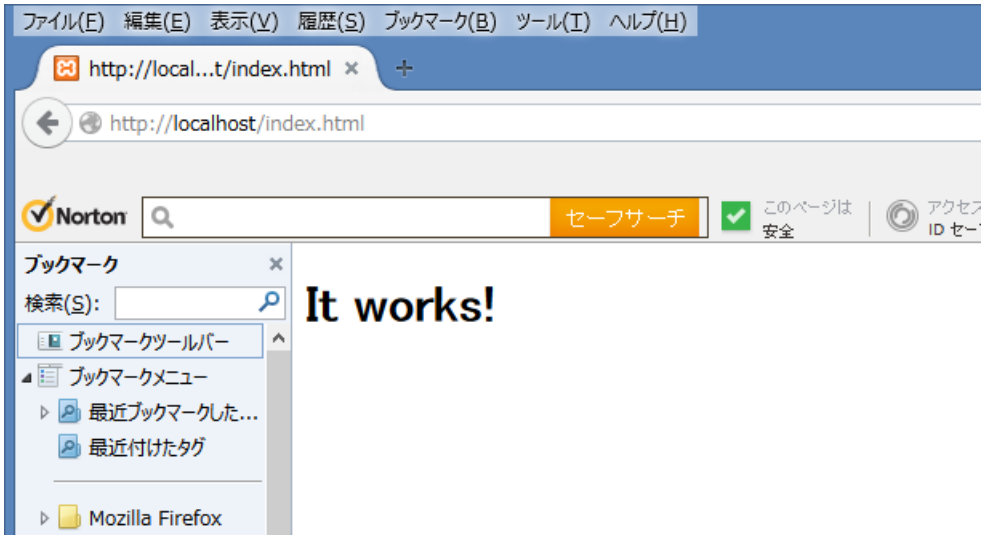


③「index.html」をアクセスしてみる

ブラウザのアドレス欄に「localhost/index.html」を打ち込みます

（xampp¥htdocs フォルダーが「localhost（ローカル・ホスト)」として認識されます）

↓ Enter



※「xampp¥htdocs¥index.html」ファイル内容（コード）を書き出すと下記です。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
    <head>
        <title></title>
    </head>
    <body>
        <h1>It works!</h1>
    </body>
</html>
```

④「index.php」をアクセスしてみる

ブラウザのアドレス欄に「localhost/index.php」を打ち込みます。

↓

今度は意外な表示になりました、これは XAMPP の管理画面で、
コントロール・パネルで[Admin]をクリックしたときの画面への入り口と同じなのですが、
[Admin]から入ったときは、更に右端の[日本語]を選択した状態になります。
上記の画面表示では、[日本語]をクリックする必要があります。
（何れにしても、php のリダイレクトを使ってジャンプしているようです）

↓　[日本語]をクリックします

一旦このように[日本語]を選択しておくと、以降は「localhost/index.php」にアクセスすると、
常にこの日本語表示の管理画面が表示されます。

アメンボは XAMPP の設定については初心者なので、ここら辺の事情はよくは理解して
いません、諸兄にて調べてみてください。

※PHP コードの確認

「xampp¥htdocs¥index.php」ファイル内容;

```
<?php
    if (!empty($_SERVER['HTTPS']) && ('on' == $_SERVER['HTTPS'])) {
        $uri = 'https://';
    } else {
        $uri = 'http://';
    }
    $uri .= $_SERVER['HTTP_HOST'];
    header('Location: '.$uri.'/xampp/');
    exit;
?>
Something is wrong with the XAMPP installation :-(
```

＜特記＞

WebRequest()で、直接に目的とするサイト（サーバー）をアクセスして、その内容を
モニターできれば良いじゃないか！、
なんで、わざわざ「ローカル・ホスト」サーバーを使うのか？、との質問が出ると思います。

アメンボは、WebRequest()関数の詳細なプロトコル解析を行うには、可能な限り単純な構造で
設定が自由に変更できる「サイト（サーバー)」が必要と考えました。

この判断が正しいか否かは、（その２）以降の投稿の中で確認していく予定です。

（３）WEB デバッカー（RestCliant）

WEB デバッガーはクライアント（ブラウザ）側から、サーバーにアクセスするメソッドを手動で色々と試してみることができます、が、アメンボが理解しているのは本当に初歩的な部分のみです。

本節での動作確認は、「WebRequest_00.mq4」の動作を簡易的にシミュレートすることで行います。つまり、「WebRequest_00.mq4」とデバッガーで同じ結果が得られるのかを試します。

RestClient をインストールすると、WEB ブラウザのアドオンとしてメニューから直ぐに使えるようになります。

WebRequest_00.mq4 の動作概要；　「１.」の結果を振り返ると

－１．サンプル・コード；　HTTP に係るポイント部のみ

```
    ・・・・・
  void OnStart()
    {
     string cookie=NULL,headers;
     char post[],result[];
     int res;
    ・・・・・
     string google_url="https://www.google.com/finance";
    ・・・・・
     res=WebRequest("GET",google_url,cookie,NULL,timeout,post,0,result,headers);
    ・・・・・
```

※ポイントをまとめれば以下の内容です、

メソッド＝GET

URL＝https://www.google.com/finance

クッキー＝無し、　など

－２．応答（レスポンス）内容（GoogleFinance.htm）；　テキスト・エディタで開いてみた

```
  <!DOCTYPE html><html><head><script>(function(){(function(){function
e(a){this.t={};this.tick=function(a,c,b){var d=void 0!=b?b:(new
Date).getTime();this.t[a]=[d,c];if(void
0==b)try{window.console.timeStamp("CSI/"+a)}catch(e){}};this.tick("start",null,a)
}var a;window.performance&&(a=window.performance.timing);var f=a?new
e(a.responseStart):new e;window.jstiming={Timer:e,load:f};if(a){var
c=a.navigationStart,d=a.responseStart;0<c&&d>=c&&(window.jstiming.srt=d-
c)}if(a){var b=window.jstiming.load;0<c&&d>=c&&(b.tick("_wtsrt",void
0,c),b.tick("wtsrt_",
"_wtsrt",d),b.tick("tbsd_","wtsrt_"))}try{a=null,window.chrome&&window.chrome.csi
&&(a=Math.floor(window.chrome.csi().pageT),b&&0<c&&(b.tick("_tbnd",void
0,window.chrome.csi().startE),b.tick("tbnd_","_tbnd",c))),null==a&&window.gtbExte
rnal&&(a=window.gtbExternal.pageT()),null==a&&window.external&&(a=window.external
.pageT,b&&0<c&&(b.tick("_tbnd",void
0,window.external.startE),b.tick("tbnd_","_tbnd",c))),a&&(window.jstiming.pt=a)}c
atch(g){}})();})();
</script><title>Google Finance: Stock market quotes, news, currency conversions &
more</title><meta name="Description" content="Get real-time stock quotes &
charts, financial news, currency conversions, or track your portfolio with Google
Finance."><meta http-equiv="X-UA-Compatible" content="IE=10"><link
rel="stylesheet" type="text/css" href="/finance/f/finance_us-
3578168175.css"><link rel="stylesheet" type="text/css"
href="/finance/_/ss/a/ver=-1kzhpkx9j35z4/am=!nhF7v0YMb2iwhB8q/bf=/r=0"><link
```

```
rel="icon" type="image/vnd.microsoft.icon"
href="/finance/favicon.ico"><style>#gbar,#guser{font-size:13px;padding-
right:8px;padding-top:4px !important;}#gbar{padding-
left:8px;height:22px}#guser{padding-bottom:7px !important;text-
align:right}.gbh,.gbd{border-top:1px solid #c9d7f1;font-
size:1px}.gbh{height:0;position:absolute;top:24px;width:100%}@media
all{.gb1{height:22px;margin-right:.5em;vertical-
align:top}#gbar{float:left}}a.gb1,a.gb4{text-
decoration:underline !important}a.gb1,a.gb4{color:#00c !important}.gbi .gb4{color
:#dd8e27 !important}.gbf .gb4{color:#900 !important}</style><script></script><scr
ipt>
      function _rpt() {}
      function _tck() {}
    </script><script>
    (function(i,s,o,g,r,a,m){i['GoogleAnalyticsObject']=r;i[r]=i[r]||function(){
    (i[r].q=i[r].q||[]).push(arguments)},i[r].l=1*new
Date();a=s.createElement(o),
m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.insertBefore(a,m)
    })(window,document,'script','//www.google-analytics.com/analytics.js','ga');
    ga('create', 'UA-40765809-1', {
      'allowLinker': true,
      'cookiePath': '/finance'
    });
```

※解説が前後しますが、「**赤書き**」部は WebRequest()でアクセスした結果（応答）と、RestClient でアクセスした結果（応答）での異なる部分です。

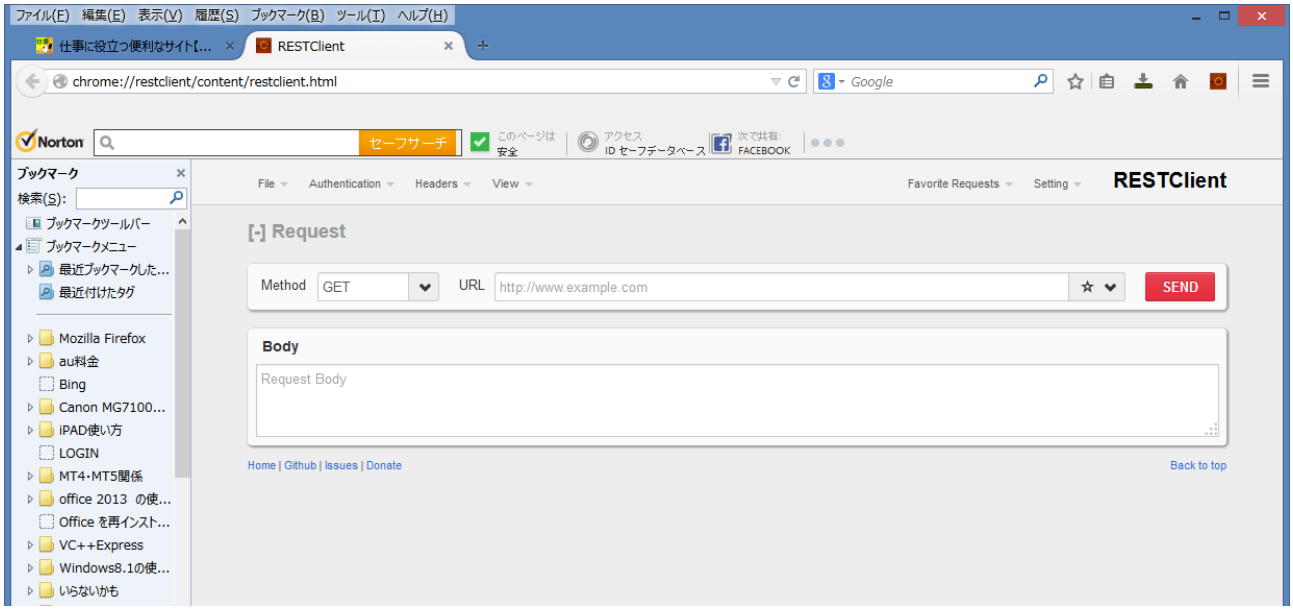
RestClient の動作確認；

－１．RestClient による要求（リクエスト）のシミュレーション；

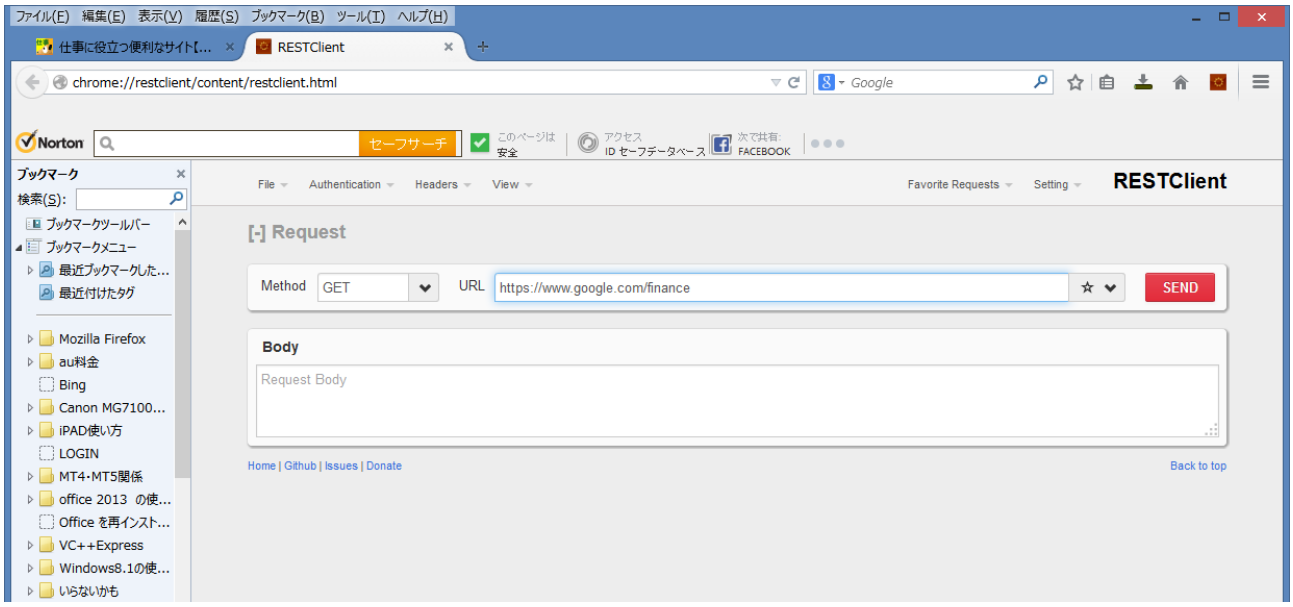
①RestClient を立ち上げます

WEB ブラウザで、[ツール] － [RESTClient] を選択か、ブラウザ右上のアイコンをクリックする

↓

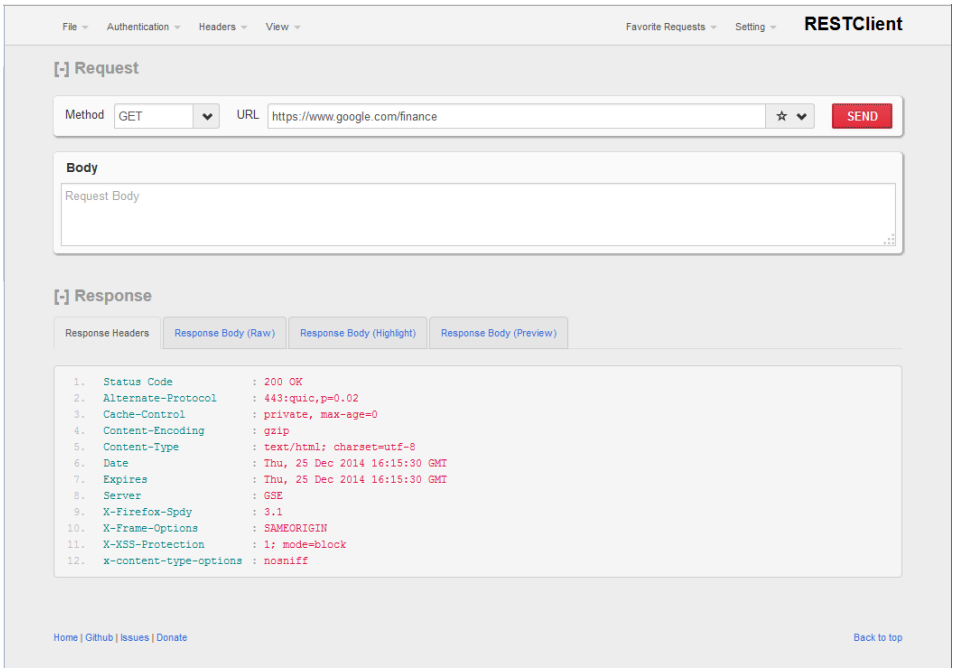


②Method（メソッド）＝GET、URL＝https://www.google.com/finance　と設定します
（クッキーの指定法はまだ理解していないので設定せず）



③［SEND］をクリック　⇒　応答（レスポンス）が返されます

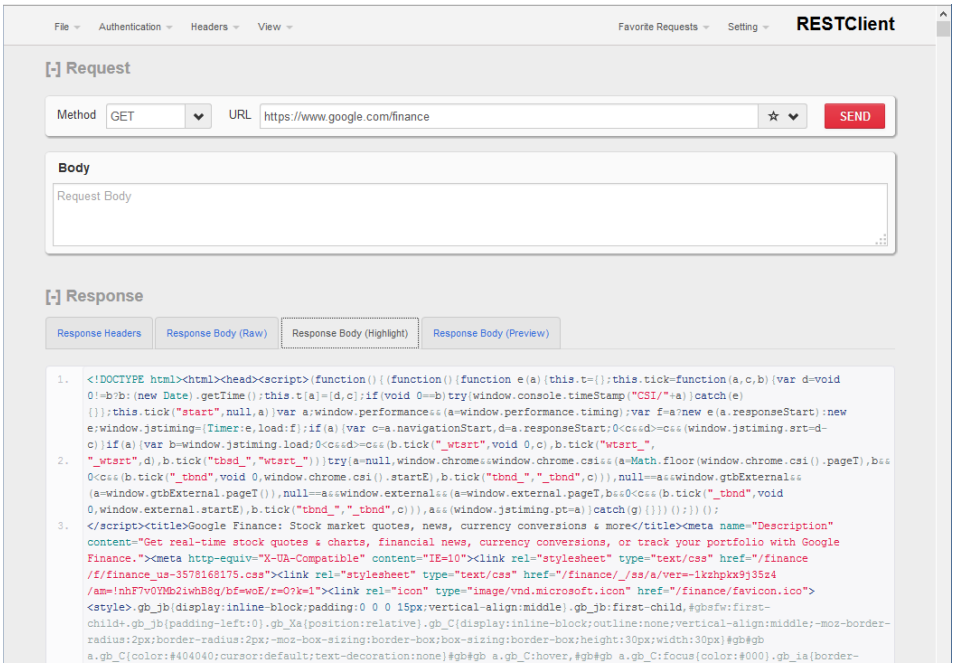
＜ヘッダー部＞表示



ヘッダの内容；

1. Status Code: 200 OK
2. Alternate-Protocol: 443:quic,p=0.02
3. Cache-Control: private, max-age=0
4. Content-Encoding: gzip
5. Content-Type: text/html; charset=utf-8
6. Date: Thu, 25 Dec 2014 16:15:30 GMT
7. Expires: Thu, 25 Dec 2014 16:15:30 GMT
8. Server: GSE
9. X-Firefox-Spdy: 3.1
10. X-Frame-Options: SAMEORIGIN
11. X-XSS-Protection: 1; mode=block
12. x-content-type-options: nosniff

＜ボディー部＞　例



－２．応答（レスポンス）の内容；

・ <!DOCTYPE html><html><head><script>(function(){(function(){function e(a){this.t={};this.tick=function(a,c,b){var d=void 0!=b?b:(new Date).getTime();this.t[a]=[d,c];if(void 0==b)try{window.console.timeStamp("CSI/"+a)}catch(e){}};this.tick("start",null,a)}var a;window.performance&&(a=window.performance.timing);var f=a?new e(a.responseStart):new e;window.jstiming={Timer:e,load:f};if(a){var c=a.navigationStart,d=a.responseStart;0<c&&d>=c&&(window.jstiming.srt=d-c)}if(a){var b=window.jstiming.load;0<c&&d>=c&&(b.tick("_wtsrt",void 0,c),b.tick("wtsrt_",
・
"_wtsrt",d),b.tick("tbsd_","wtsrt_"))}try{a=null,window.chrome&&window.chrome.csi&&(a=Math.floor(window.chrome.csi().pageT),b&&0<c&&(b.tick("_tbnd",void 0,window.chrome.csi().startE),b.tick("tbnd_","_tbnd",c))),null==a&&window.gtbExternal&&(a=window.gtbExternal.pageT()),null==a&&window.external&&(a=window.external.pageT,b&&0<c&&(b.tick("_tbnd",void 0,window.external.startE),b.tick("tbnd_","_tbnd",c))),a&&(window.jstiming.pt=a)}catch(g){}})();})();
・ </script><title>Google Finance: Stock market quotes, news, currency conversions & more</title><meta name="Description" content="Get real-time stock quotes & charts, financial news, currency conversions, or track your portfolio with Google Finance."><meta http-equiv="X-UA-Compatible" content="IE=10"><link rel="stylesheet" type="text/css" href="/finance/f/finance_us-3578168175.css"><link rel="stylesheet" type="text/css" href="/finance/_/ss/a/ver=-1kzhpkx9j35z4/am=!nhF7v0YMb2iwhB8q/bf=woE/r=0?k=1"><link rel="icon" type="image/vnd.microsoft.icon" href="/finance/favicon.ico"><style>.gb_jb{display:inline-block;padding:0 0 0 15px;vertical-align:middle}.gb_jb:first-child,#gbsfw:first-child+.gb_jb{padding-left:0}.gb_Xa{position:relative}.gb_C{display:inline-block;outline:none;vertical-align:middle;-moz-border-radius:2px;border-radius:2px;-moz-box-sizing:border-box;box-sizing:border-box;height:30px;width:30px}#gb#gb

※解説が前後しますが、「**赤書き**」部は WebRequest()でアクセスした結果（応答）と、RestClient でアクセスした結果（応答）での異なる部分です。

結果まとめ；

・現状では「WebRequest()」と「ReatClient」の実行結果比較では、応答（レスポンス）内容の後半部が異なる。
・現状では相違の原因は不明だが、「ReatClient」はメソッドや URL を組み合わせてデバッグ出来そうなので、解析ツールとしてリザーブしておくことにしました。

（4）プロキシ・サーバー（Fiddler Web Debugger）

Fiddler は HTTP プロトコル解析に特化したツールです。（でも、ただ者ではありません）

＜特記＞

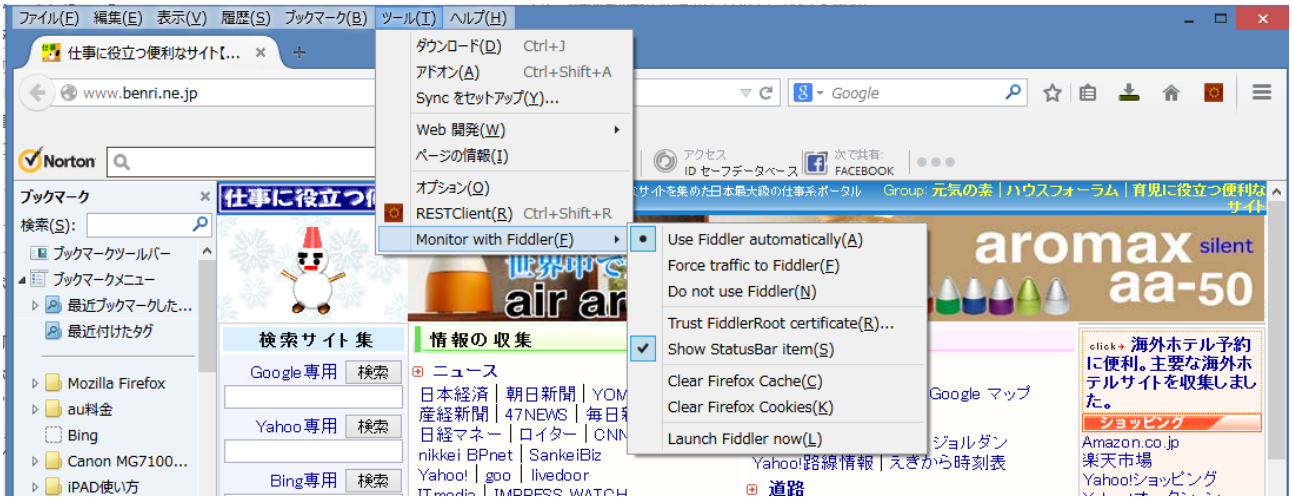
・Fiddler（バイオリン）をインストールしたところ、同時に「FiddlerHook2.4.8.3」が Firefox のアドオンとしてインストールされました。
（他の WEB ブラウザを使っている場合にどうなるかは、試していません）

・Fiddler は非常に強力なツールなので「取扱注意！」かもしれません、理由は使ってみると判ります。

－１．WEB ブラウザの一般動作をモニターする；

①ブラウザ側の設定を行う

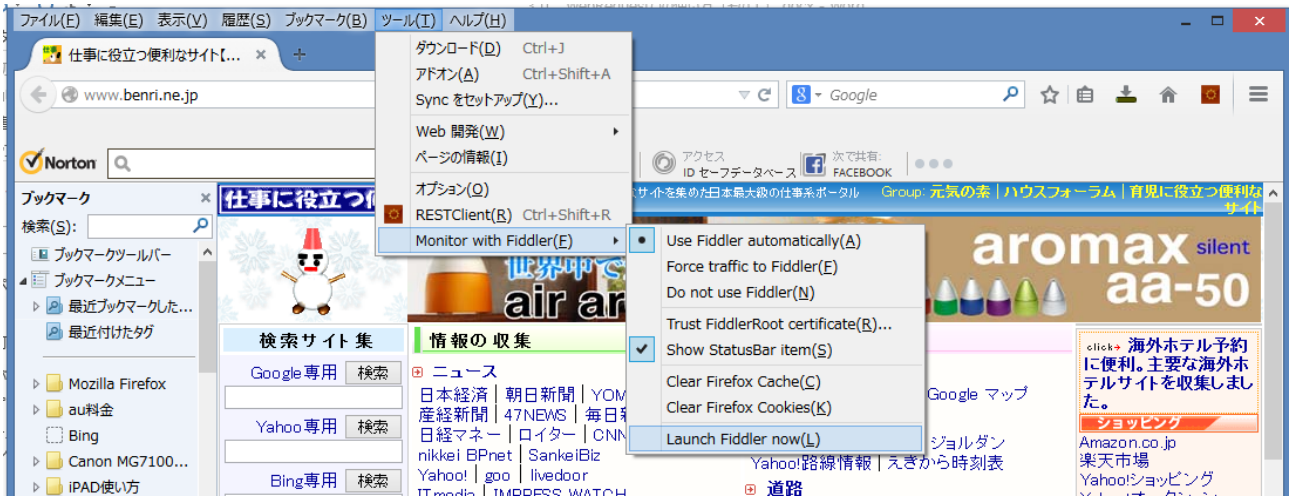
［ツール］－［Monitor with Fiddler］で表示されるメニューの
「Use Fiddler automatically」を選択（使わないときは Do not use Fiddler を選択）



②Fiddler を立ち上げます

WEB ブラウザから一緒に立ち上げる場合；

［ツール］－［Monitor with Fiddler］で表示されるメニューの
「Launch Fiddler now」を選択する

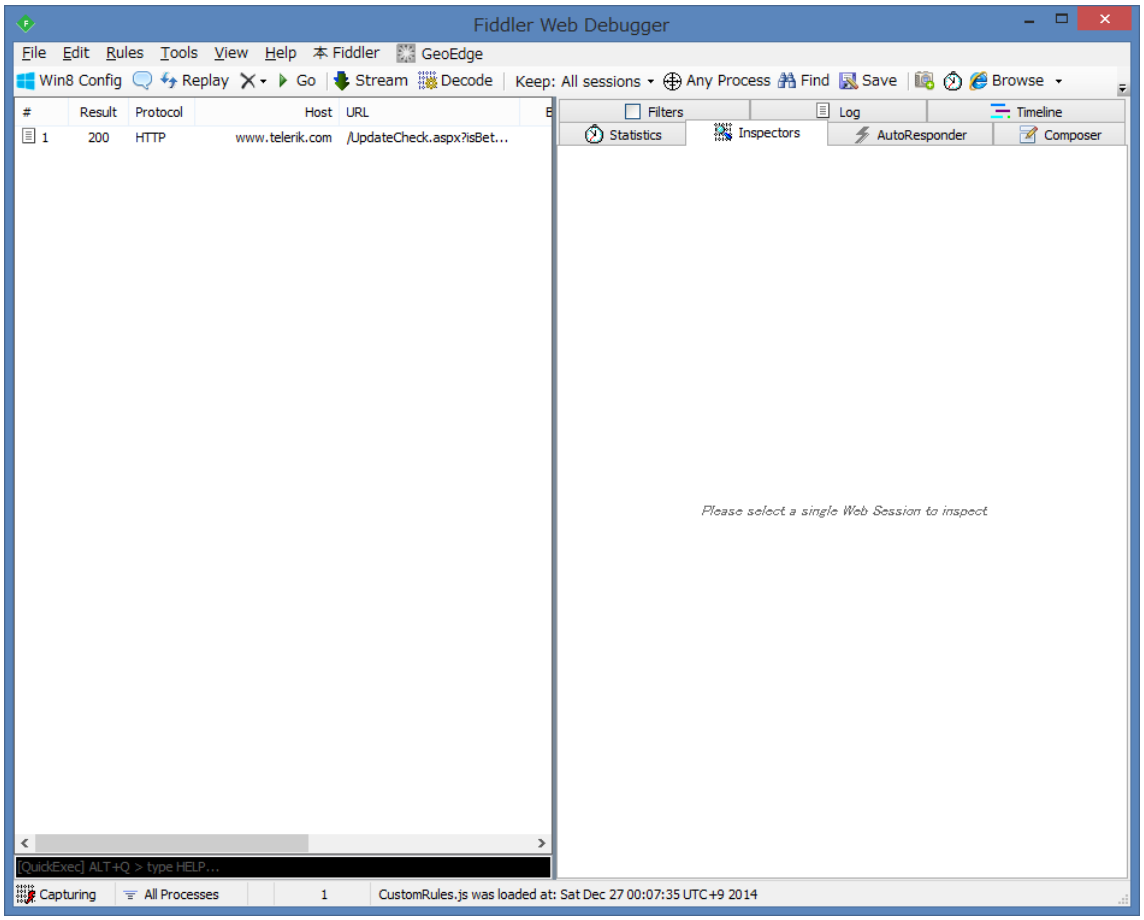


Fiddler のみを立ち上げる場合；

下記のアイコン（Fiddler）を直接にダブル・クリックします

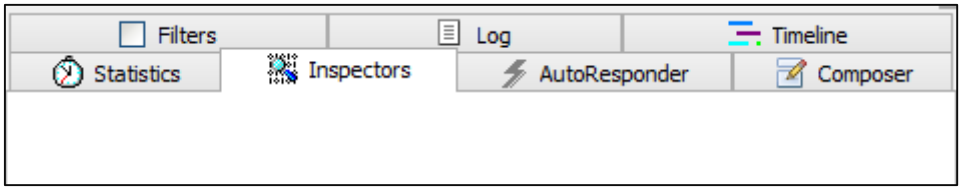
↓

Fiddler のモニター画面が表示されます。



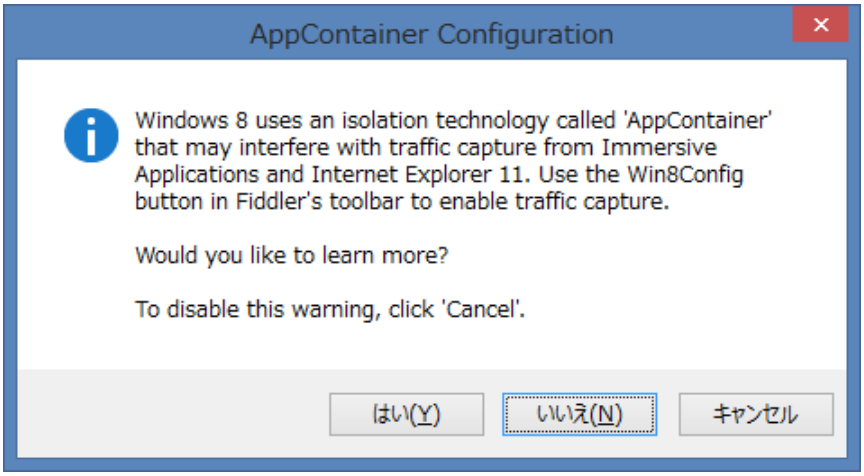
・左側がプロセスＮｏを示します

・右側で、選択したプロセスの中身を観ることができます。



＜補足＞

Fiddler を立ち上げようとするとき、下記の表示が出る場合があります。

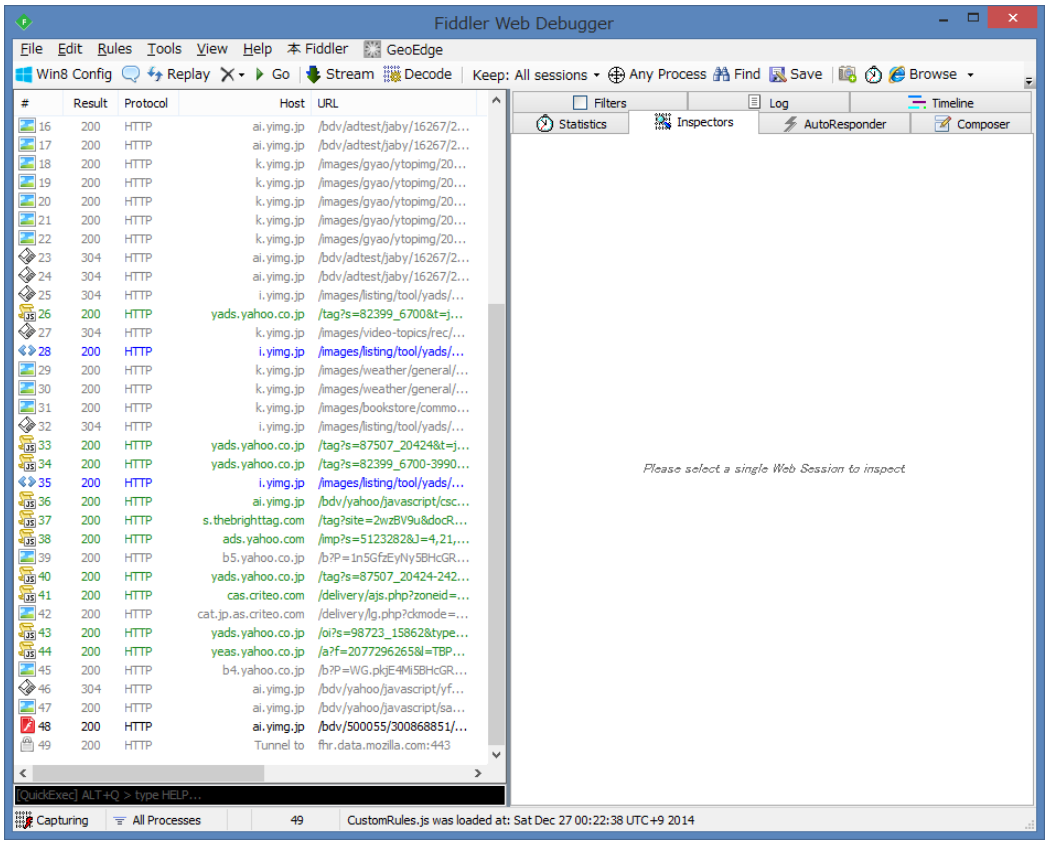


OS に Wndows 8 以降、WEB ブラウザに「Internet Explorer 11」以降を使うときのみ、前処理（設定）が必要となるようですがアメンボは特に調べていません。アメンボは FireFox を WEB ブラウザに使っているので［いいえ］を選択して先に進み、Fiddler を立ち上げました。

③試しに、WEB ブラウザで「URL= http://www.yahoo.co.jp/」をアクセスしてみます

WEB ブラウザ；

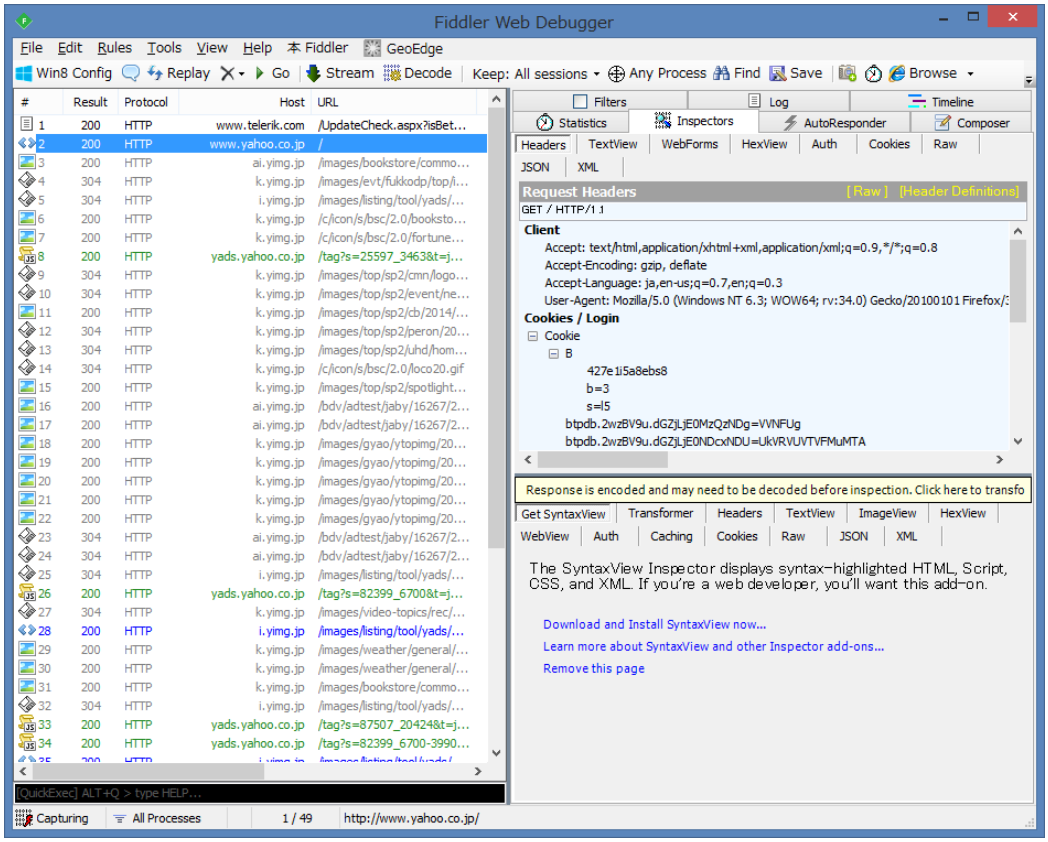
Fiddler；



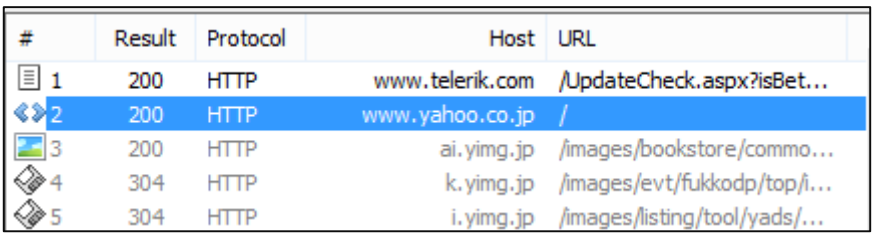
※非常に多くのプロセスが表示されています。

④プロセスから「www.yahoo.co.jp」をアクセスした部分を選択します

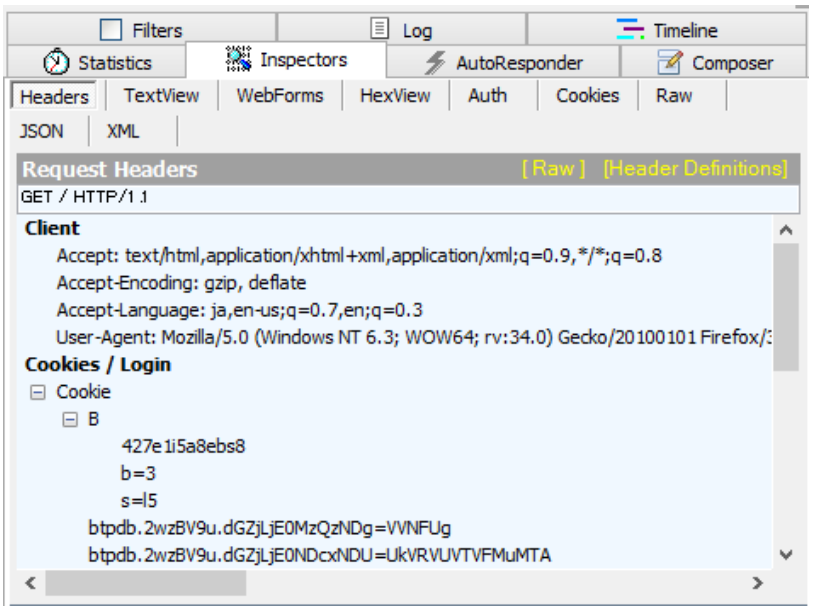
右側の「上半分」にアクセスしたときの要求（request）のプロトコル内容、
「下半分」に応答（response）のプロトコル内容が表示されます。



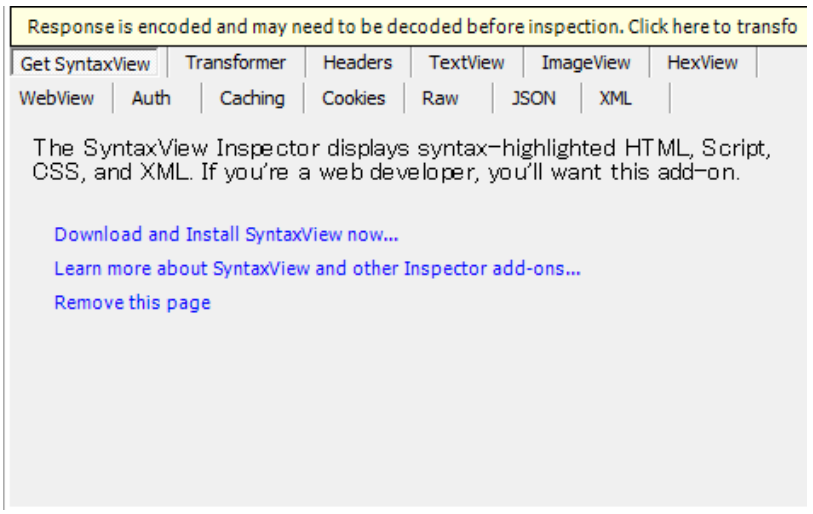
・左側のプロセス



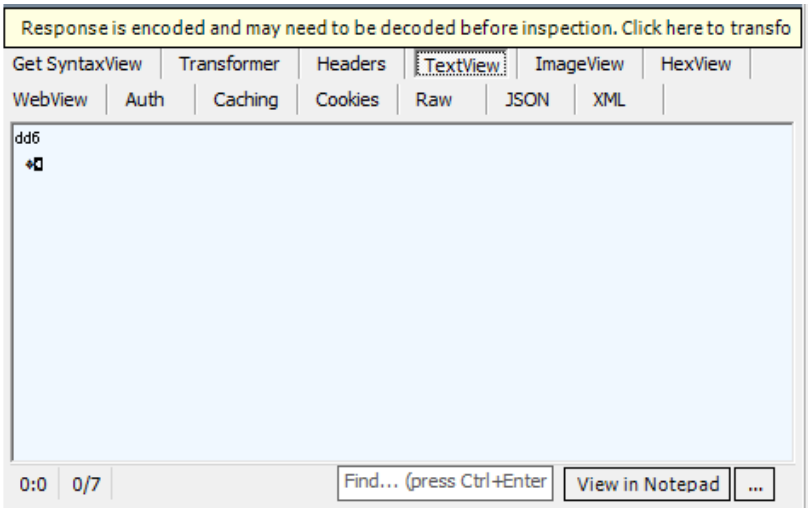
・右側の上半分；　要求（request）部分



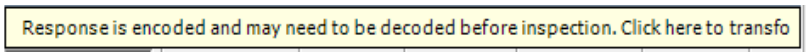
・右側の上半分；　応答（response）部分



［TextView］タブを開いてみた

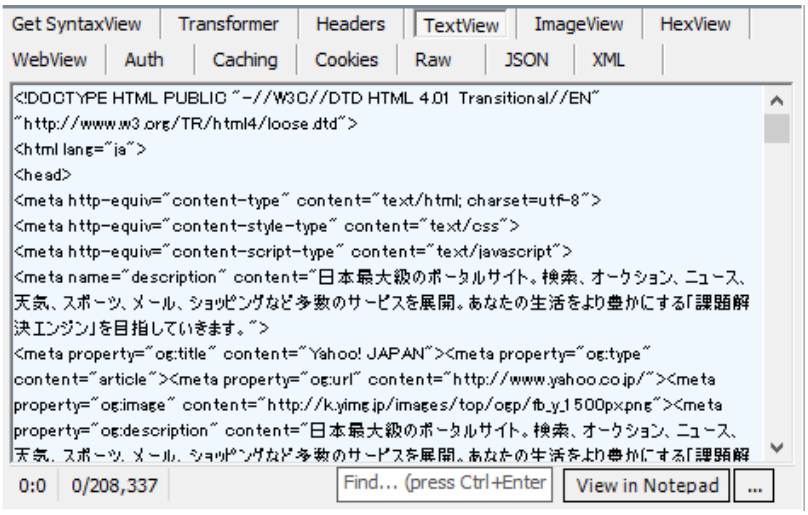


下記の表示が気になったのでクリックした



↓

エンコードされた応答（response）内容が表示されました



※一般的な動作の確認は、これくらいで切り上げます。

機能が多すぎて解説しきれません、興味のある諸兄は色々調べてみてください。

－２．WebRequest()動作をモニターしてみる；

WebRequest()で発行されるHTTPプロセスをモニターしようとして、トライしたのですが、３～４日間ほどは失敗し続けました。
理由がわかり対応も取れたので、動作確認の手順を解説する前に「原因とその対応」のポイントを述べます。

○MT4側での「プロキシー・サーバーの経由」設定が必要；

まったく使ったことが無い機能たっだので、当初は気が付かなかったのですが、
Fiddlerはプロキシー・サーバーなので、MT4側で設定する必要がありました。
でも、設定するとFiddlerが立上っていないとMT4は回線不通になります。
（ついやってしまう、回線不通）

○HTTPSプロトコルのモニターをFiddlerで可能にする処理が必要；

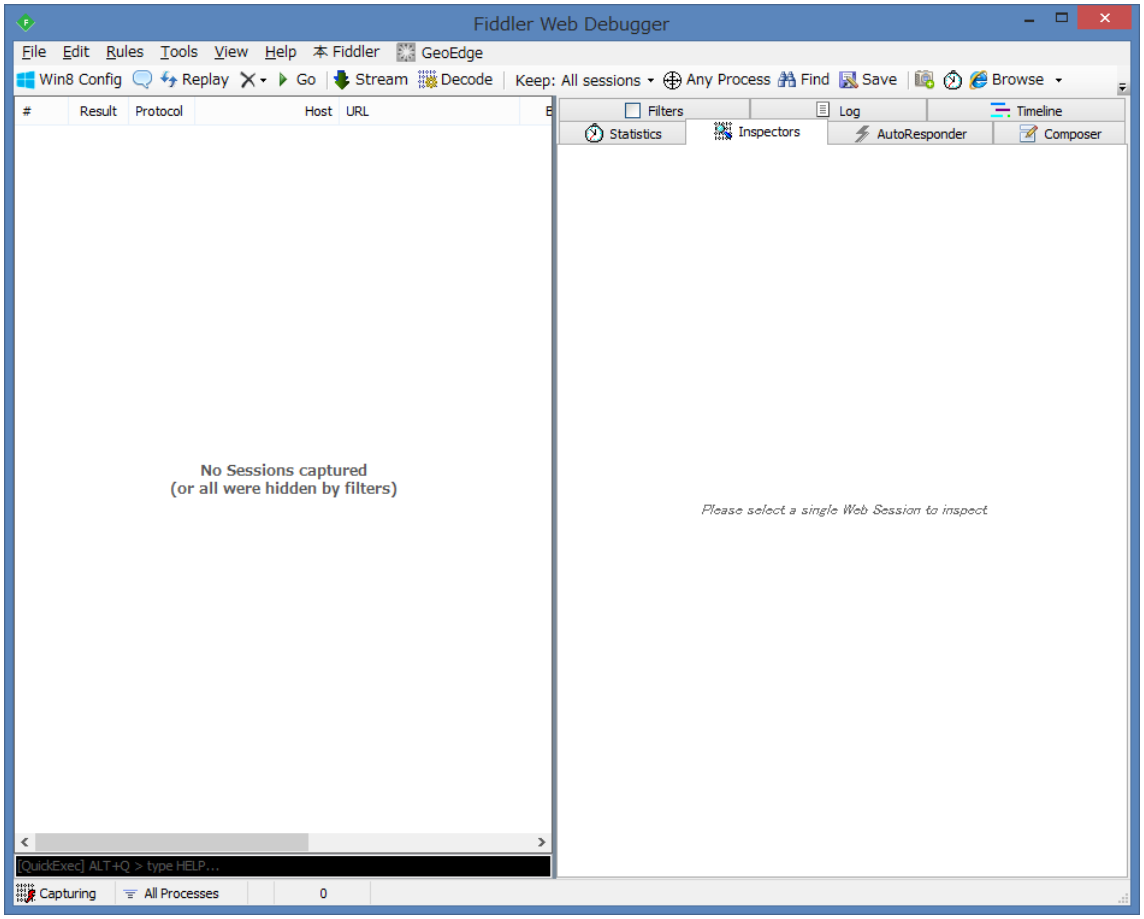
メタクウォーツ社のスクリプト・サンプルをコピーした「WebRequest_00.mq4」
コード中の「URL＝https://www.google.com/finance」を観ると判るように
そもそもgoogleのサイトをアクセスするプロトコルは「HTTPS」となります。
（手動でgoogleのサイトをアクセスしてみると直ぐに確認できます）
HTTPS（Hypertext Transfer Protocol Secure）は、HTTPによる通信を安全に（セキュアに）行うためのプロトコルおよびURIスキームでした！
つまり、安全な要求（request）と確認できない限りモニターはできないので、
Fiddlerにアクセス権限を疑似的にでも設定する必要があります。

動作確認方法；
MT4でスクリプト「WebRequest_00.mq4」を動作させて、Fiddlerでモニターします。
ただ、試してみて判明したのですが「HTTPS」で通信するサーバー（サイト）をモニターする場合、[Trusted CA list」発行済か否かでFiddlerの立ち上げ手順が異なります。

Ⅰ．Fiddlerで「HTTPS」サーバーを初めてモニターする場合（Trusted CA list発行前）

本稿の例では、[https://www.google.com/finance]をモニターします、
※Trusted CA listとは「trusted certificate authority（信頼できる認証局）」が発行した認証を意味するようですが、アメンボは未だ良くは理解していません。

①まずFiddlerを立ち上げておきます
本節ではMT4のセッションのみモニターしたいので、Fiddlerのアイコンから立ち上げます。（WEBブラウザは立ち上げないこと！）

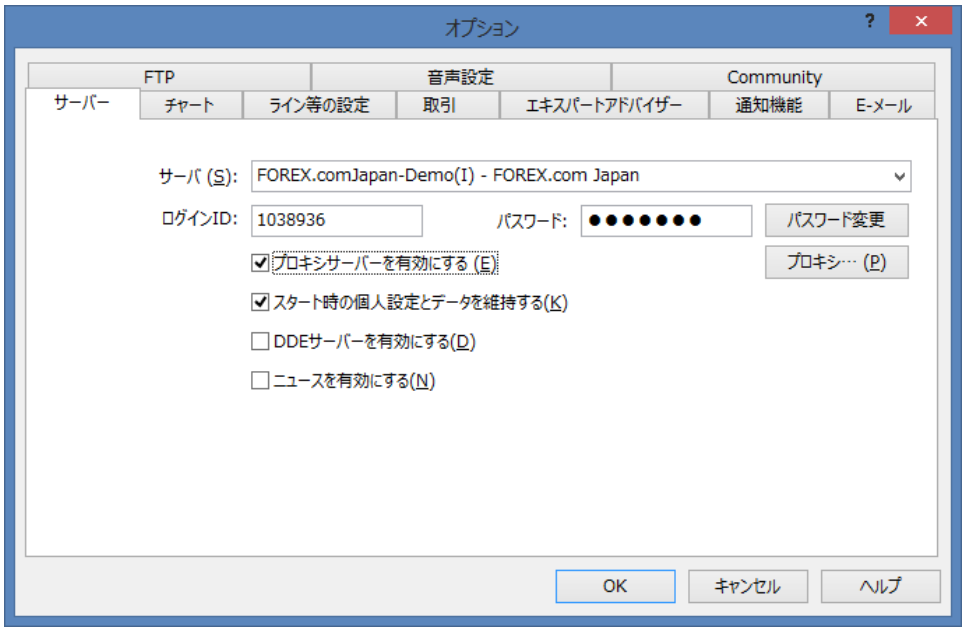


②MT4 を立ち上げ、プロキシー・サーバーの設定を行う

［ツール］－［オプション］で「オプション」ウインドウを表示し、

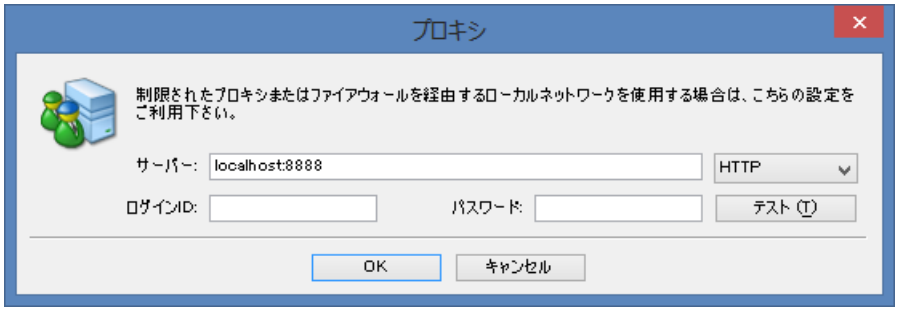
［サーバー］タブを開き、

「プロキシーサーバーを有効にする」にチェックを入れる



↓

③［プロキシ…］を選択し、下記の様に設定します



設定は以下のようにします、

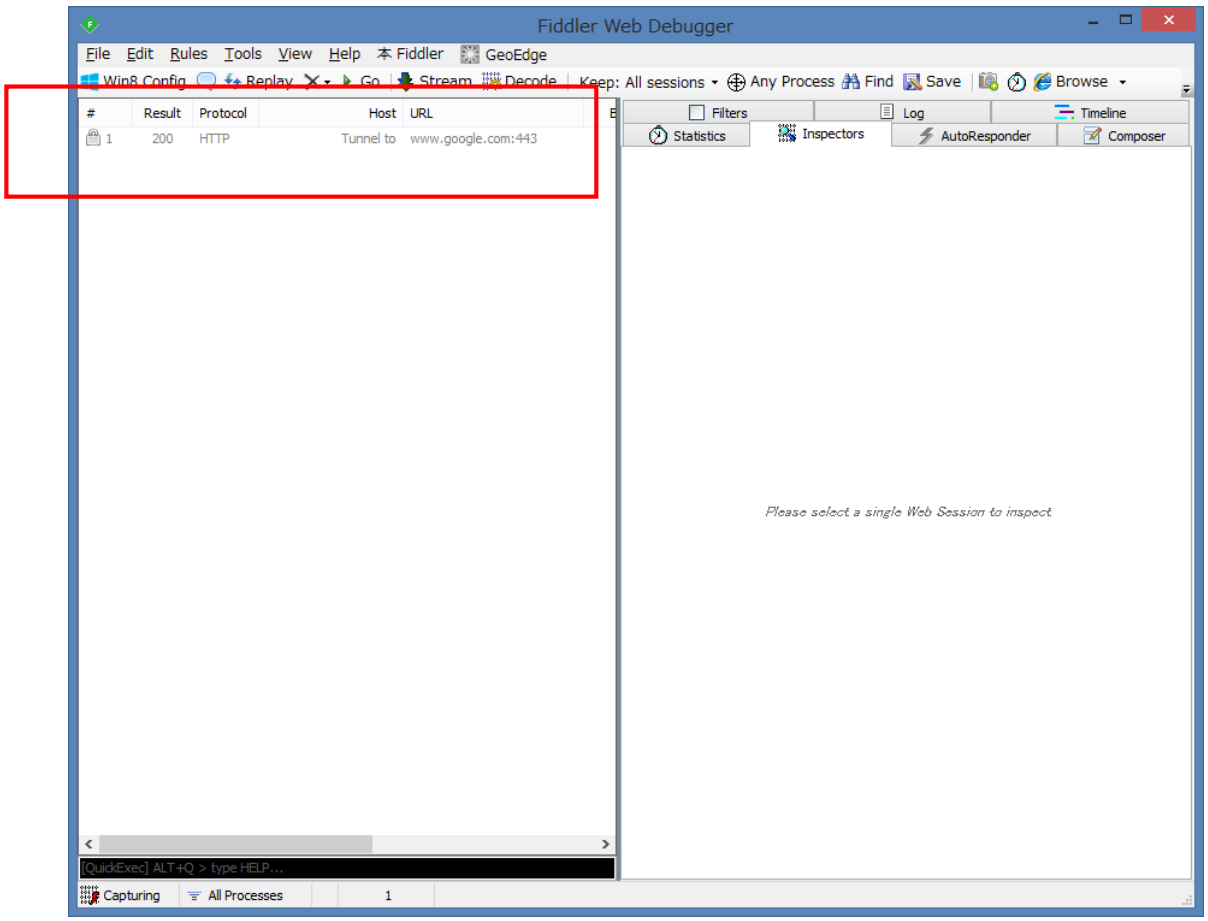
［サーバー］；　「localhost:8888」（または「127.0.0.1：8888」）　「HTTP」

※「127.0.0.1」はローカルホストのアドレス、

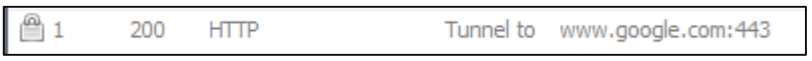
「8888」はアクセスするプロキシー（Fiddler）のポートＮｏです

↓［ＯＫ］を２回

④スクリプト「WebRequest_00.mq4」を動作させます



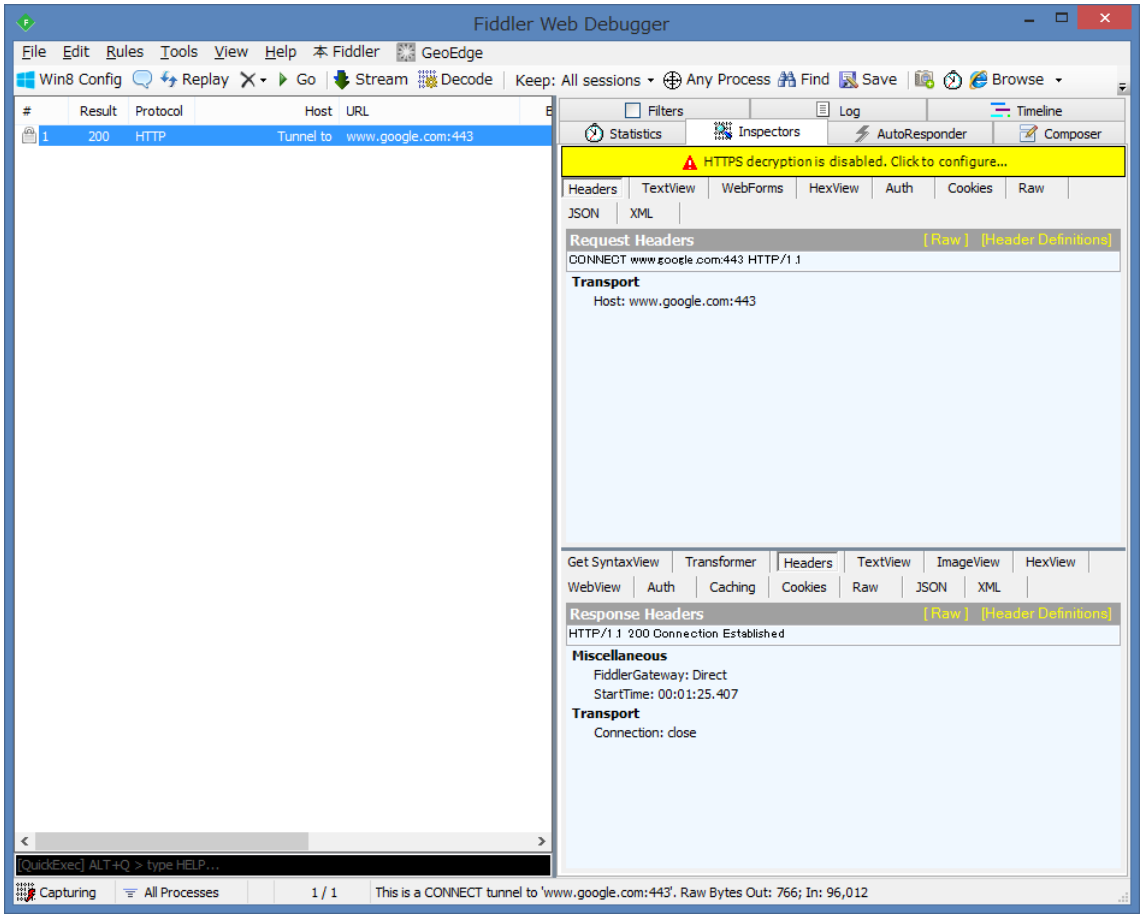
要求（request）部の拡大；



「Tunnel to ・・・」とは、HTTPS の様に通常はモニターできないセッション等のことらしい！！

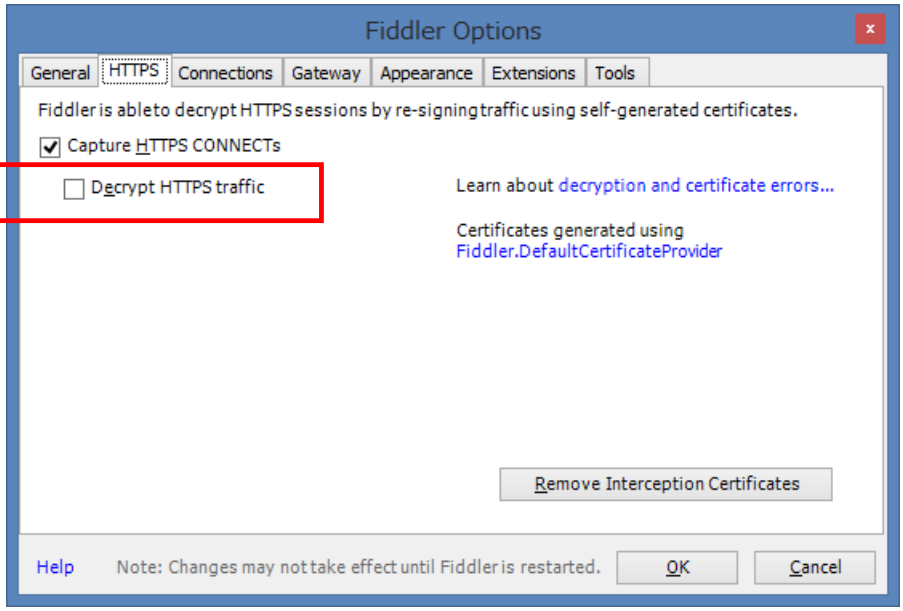
「443」とは、HTTPS が使用するポートＮｏです。

⑤「Tunnel to www.google.com:443 ・・・」を選択した



応答（response）の注意マーク部を拡大

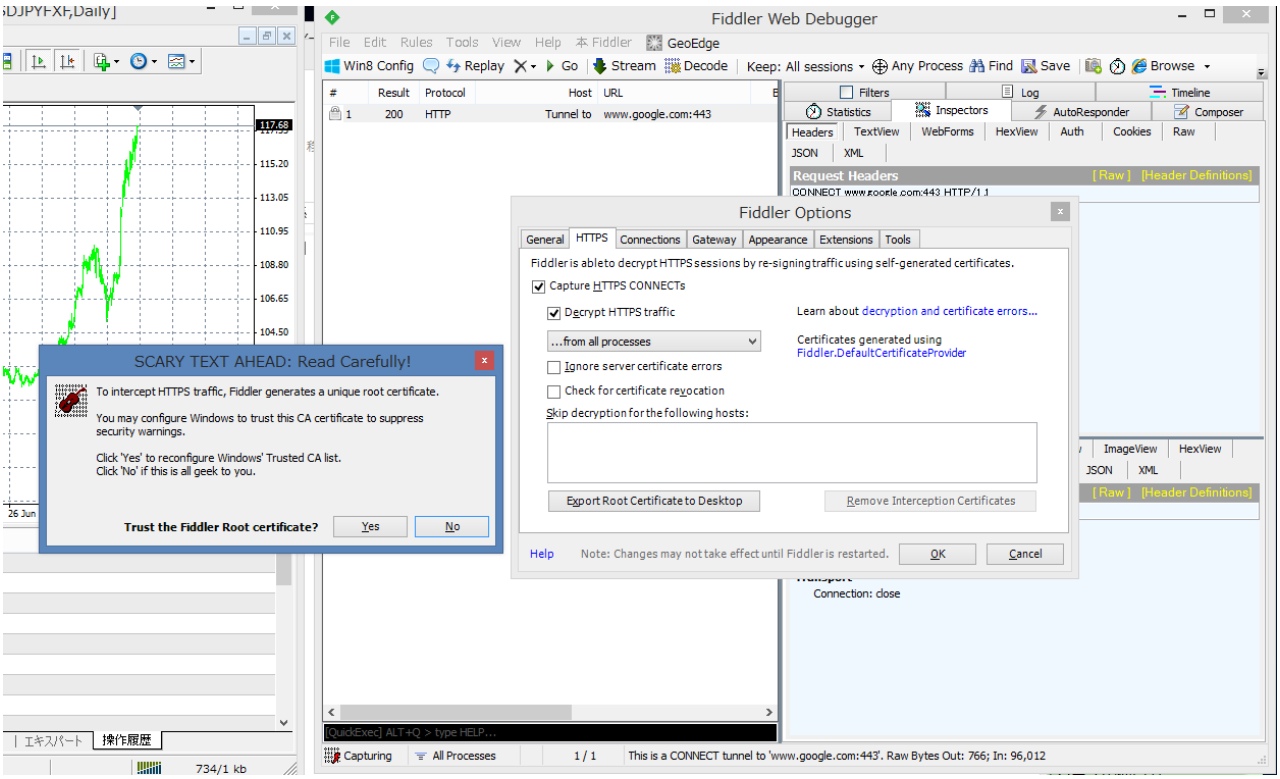
⑥**これをクリックしてみた**ところ、下記を表示した



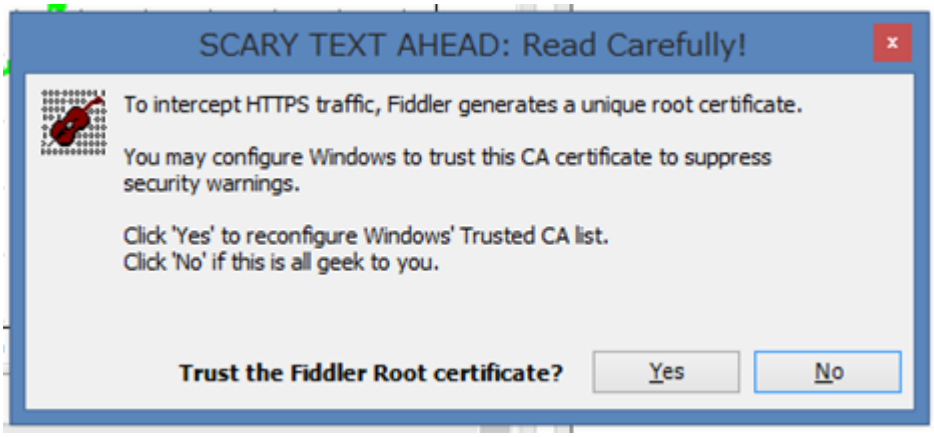
ここで、「Decrypt HTTPS trafic」にチェックを入れて、［ＯＫ］を選択、

↓

なんか、またまた「警告」が表示されました！！



警告表示を拡大；



HTTP トラフィックをモニターできるとのことなので、［YES］を選択？しかし、「Windows' Trusted CA list」って何だろう。

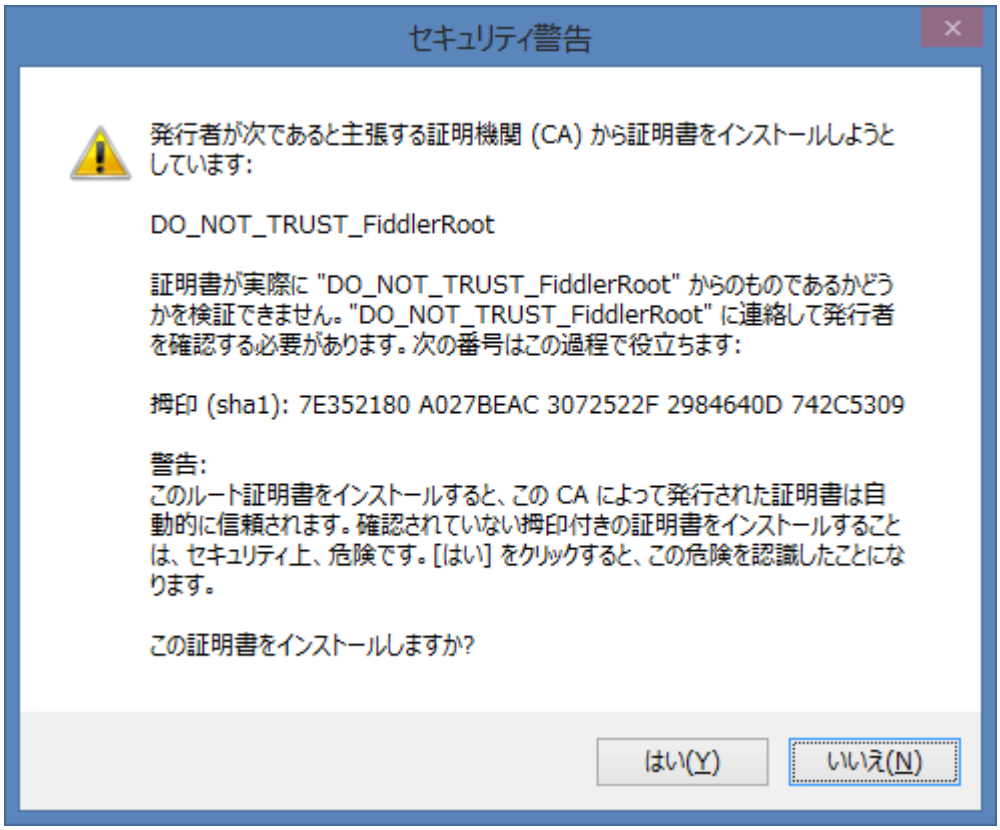
（CA とは認証局のことらしいことまでしか調べていません）

trusted certificate authority (CA)

↓

さらに「セキュリティー警告」が表示された

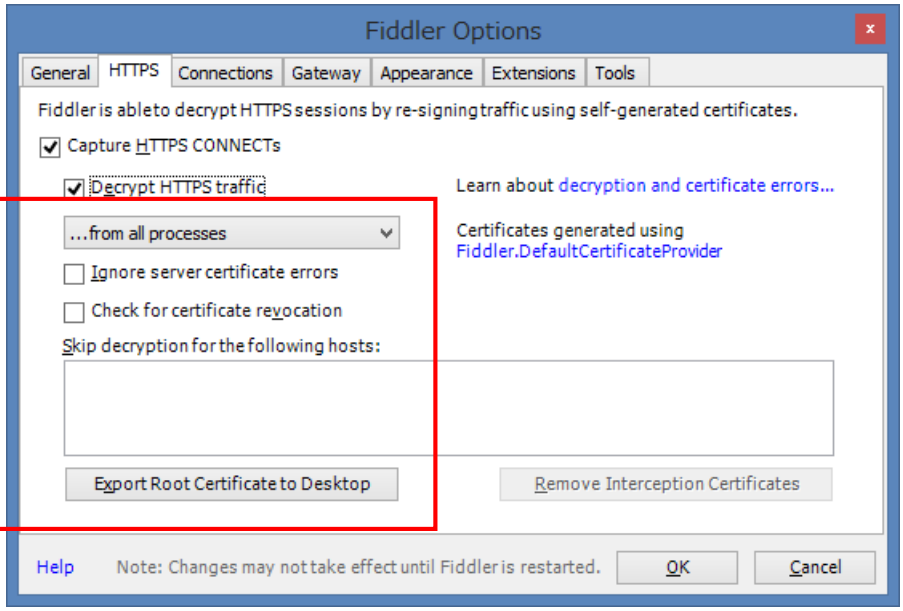
（なんか、大変なことになってきた！）



［はい］ ↓

⑦先にすすむ

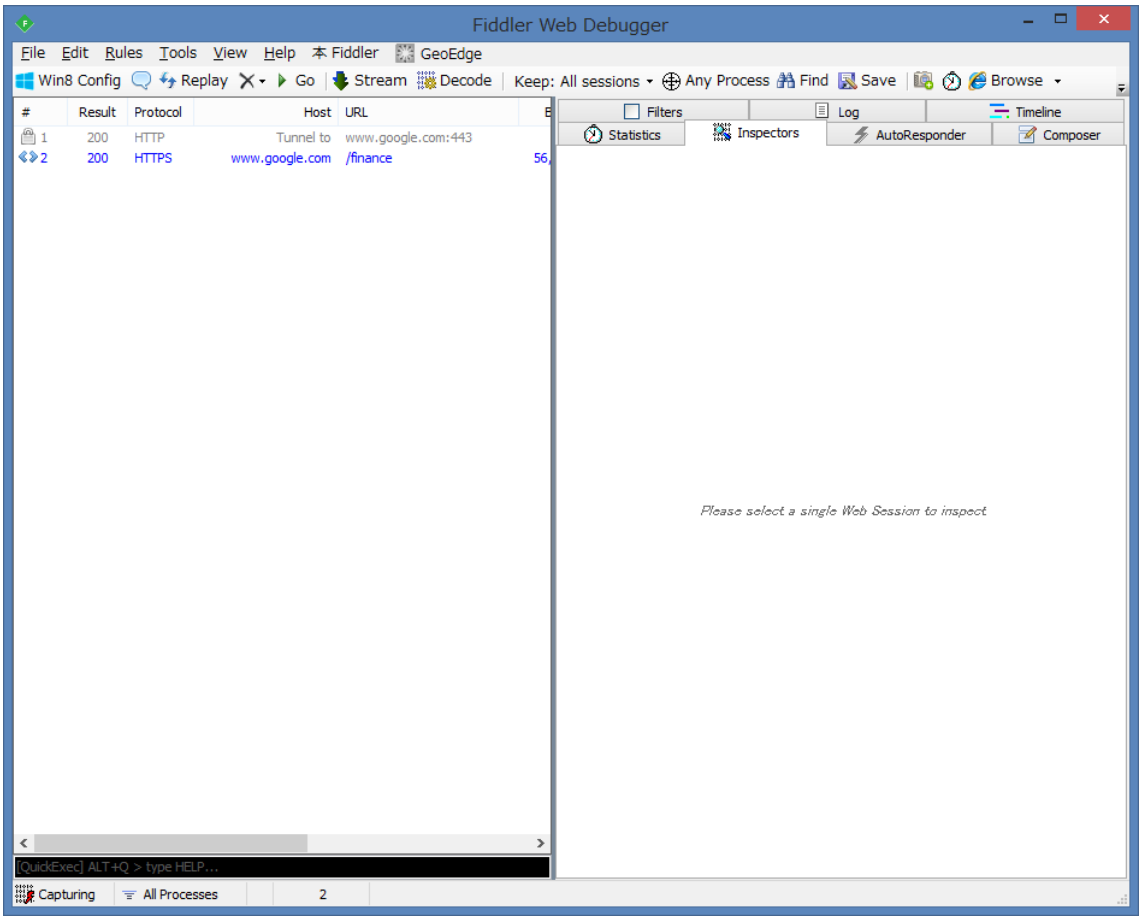
「DO_NOT_TRUST_FiddlerRoot」と言う、恐ろしげな発行者からの証明書らしいのですが、ここまで来たので思い切って［はい］を選択

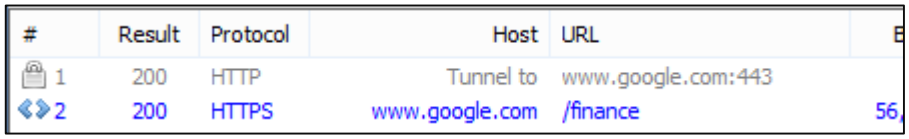


赤枠内の表示が追加された！　［ＯＫ］を選択

↓

⑧MT4 上で、再度「WebRequest_00」を実行



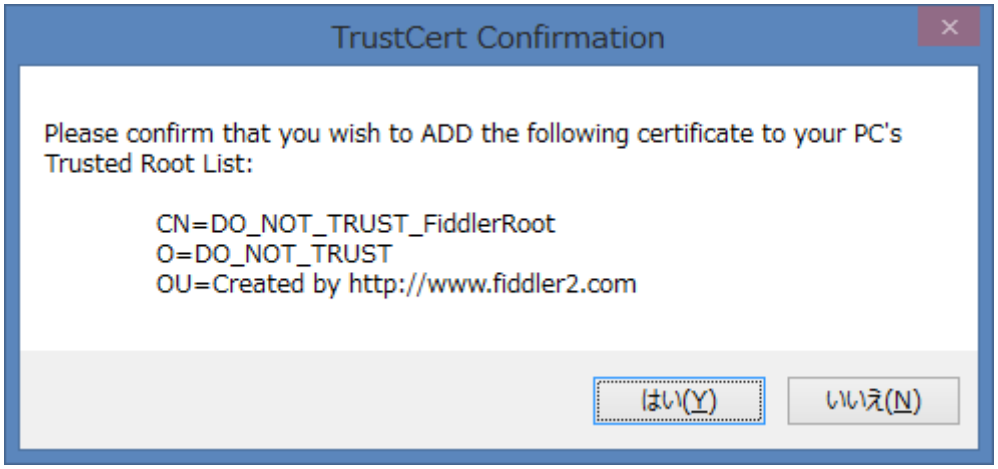


今度は、セッションをキャプチャーできたようです！・・鍵が開いている

前回は「Tunnel to www.google.com:443 ・・・」だったものが、

「HTTPS www.google.com/finance」と表示されるようになりました。

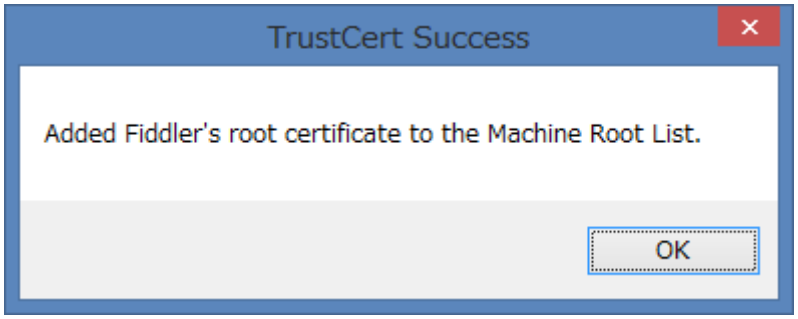
と、思ったら「確認画面」が出てきた！



今さら、後戻りはできないので［はい］を選択

でも、Trused Root List を殆ど理解していない！です。

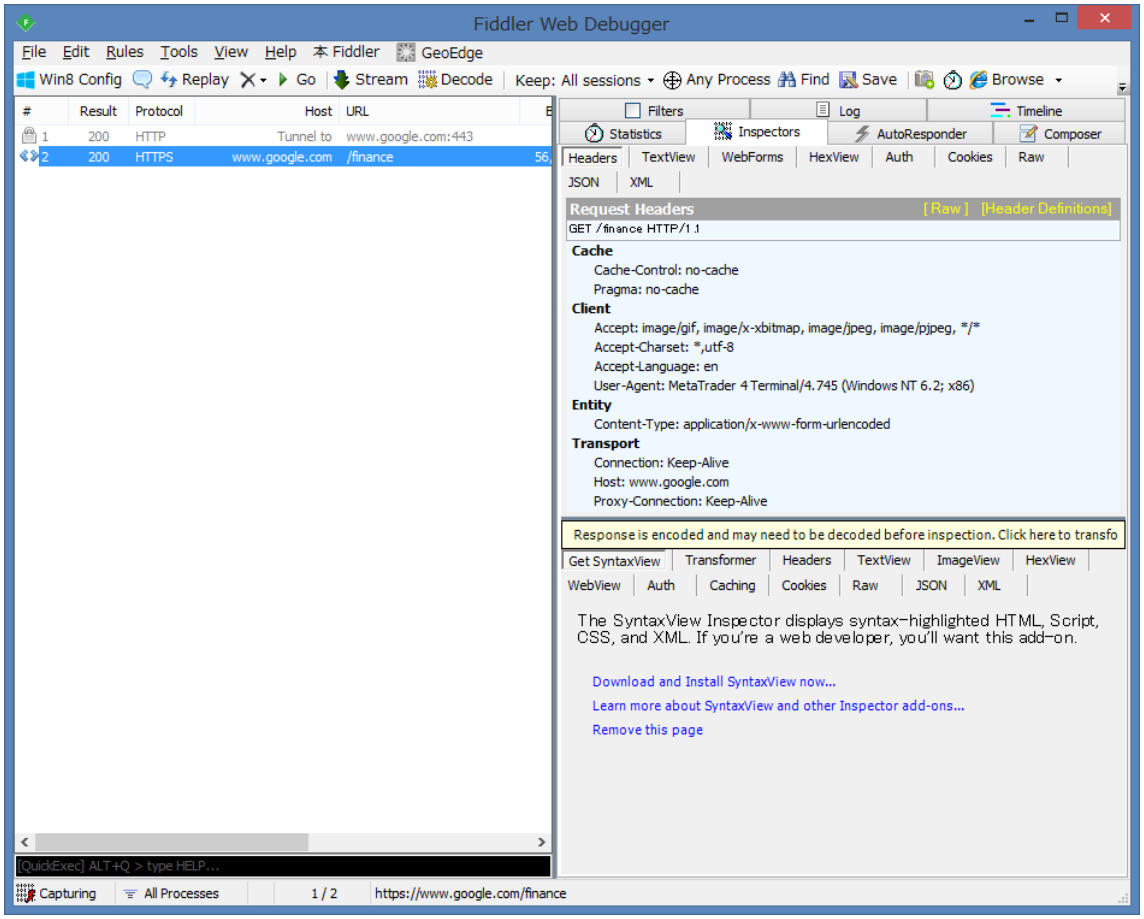
↓



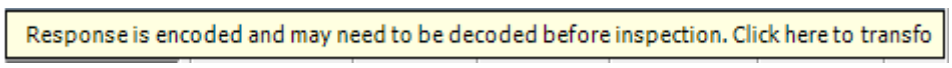
［ＯＫ］としか言えないので、選択

↓

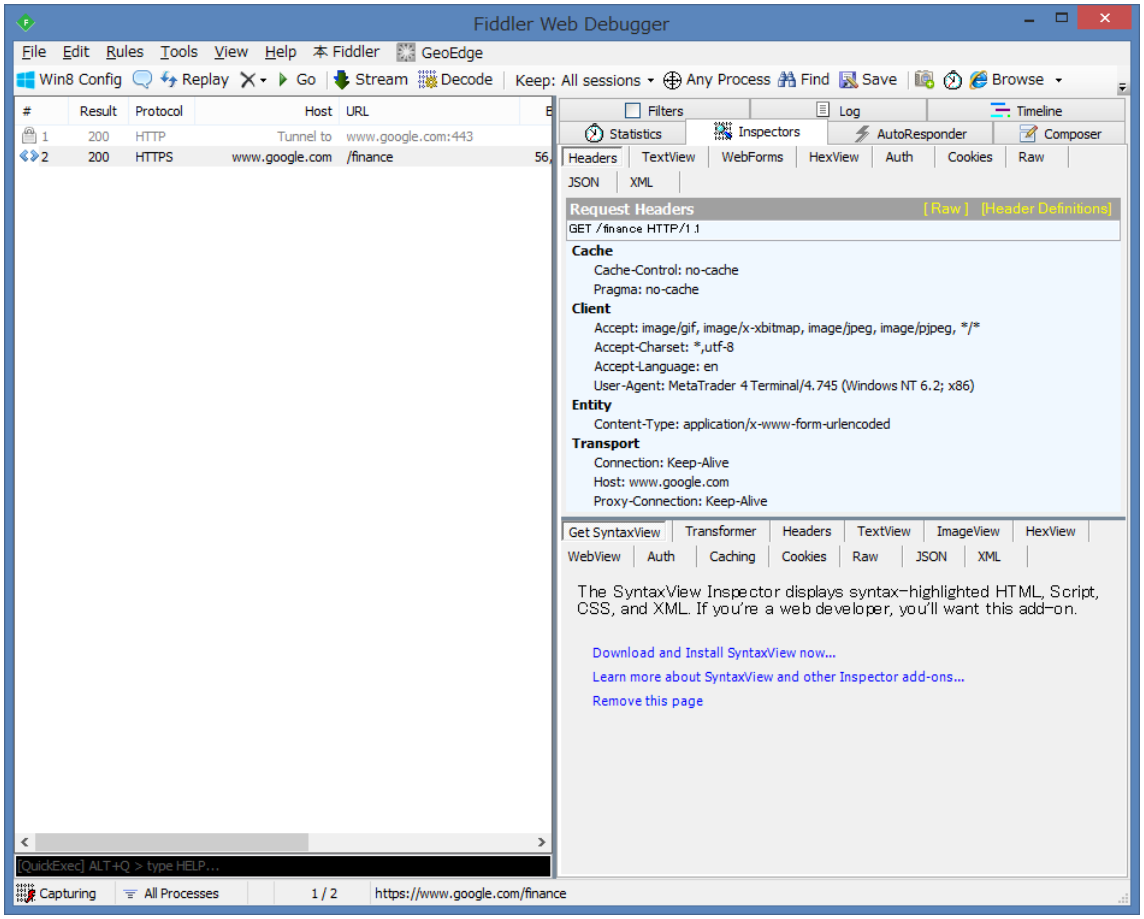
⑨「HTTPS www.google.com/finance」を選択した



下記の部分をクリック



↓



※ようやく、設定・確認は終了して「詳細」の解析ができる状態になったようです。
とにかく、セッション「https://www.google.com/finance」をキャプチャーし、
モニターができるようになったようです。

＜考察＞

※メタクゥオーツのサンプル・コードが「HTTPS」プロトコルを使用している google を
アクセスする例だったので、これを試したため、ヤヤコシイ認証が必要になって
しまいました。

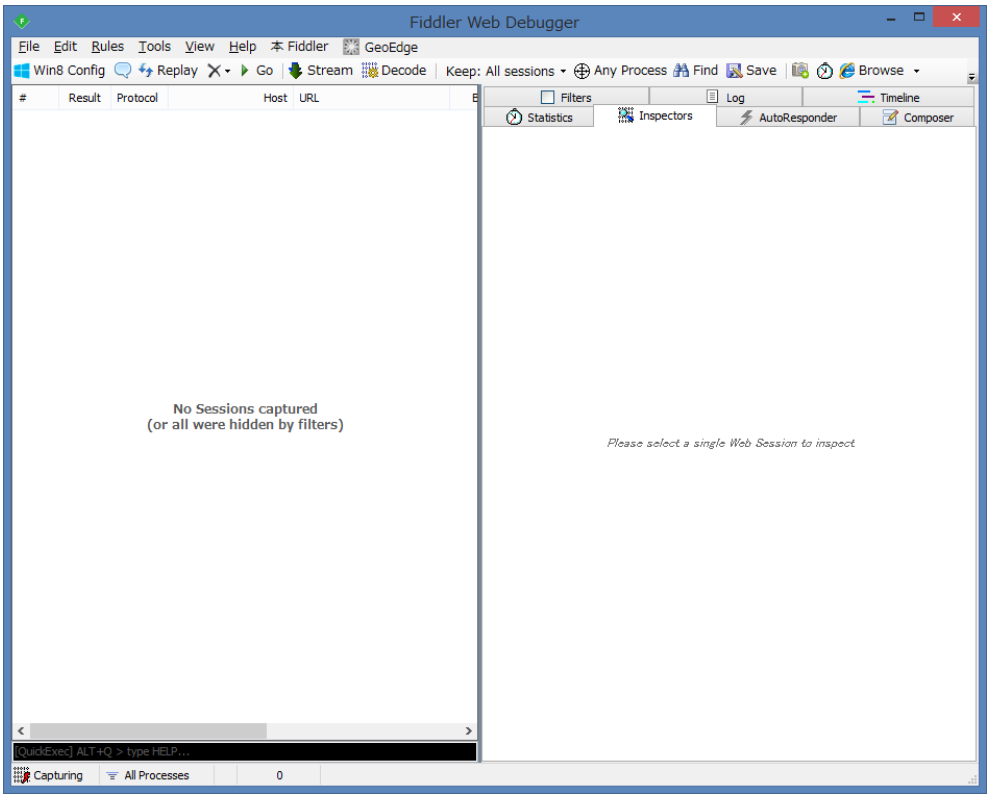
通常の「HTTP」プロトコルのサイト（サーバー）へのアクセスで試していれば、
こんな面倒な設定は不要だったのに！、と多少反省しながら進めた解析でした。

ただ、利用したいサイトが「HTTPS」の場合でも、モニターする手立てがあるとは、
「**Fiddler は凄い！**」と驚いています。

Ⅱ．[Trusted CA list 発行済み以降]

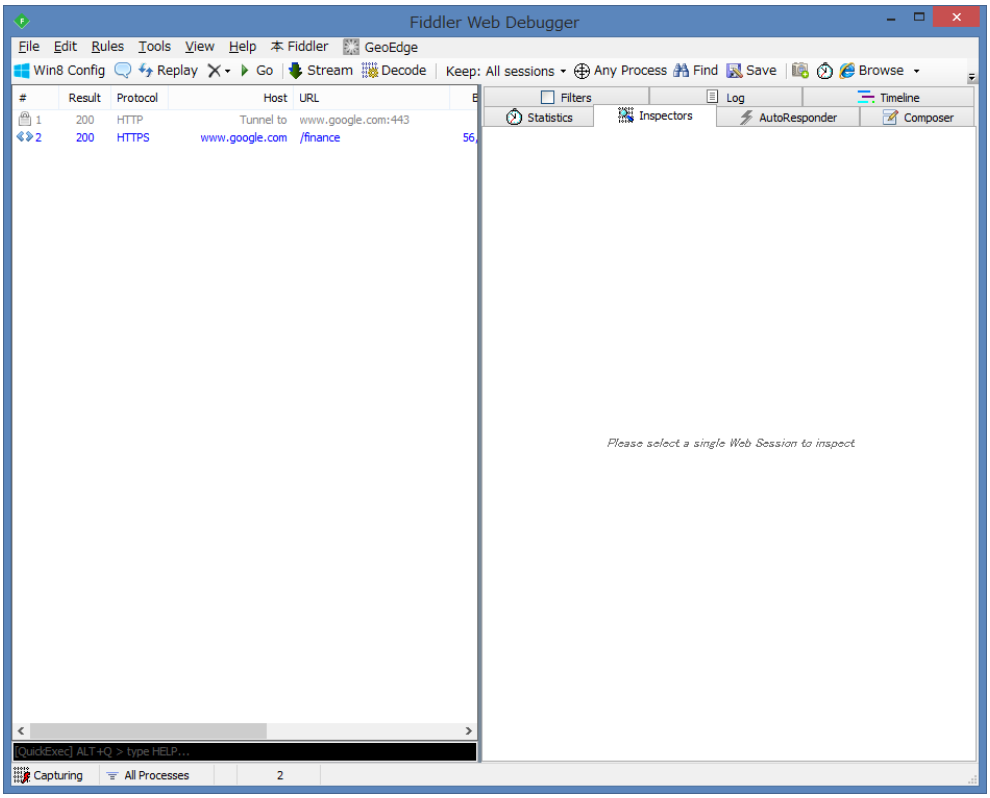
※今回の認証は「https://www.google.com/finance」にのみ有効のようですが、
一旦 CA が発行されたサイトに対しては、Fiddler は直ぐにモニターが可能です。

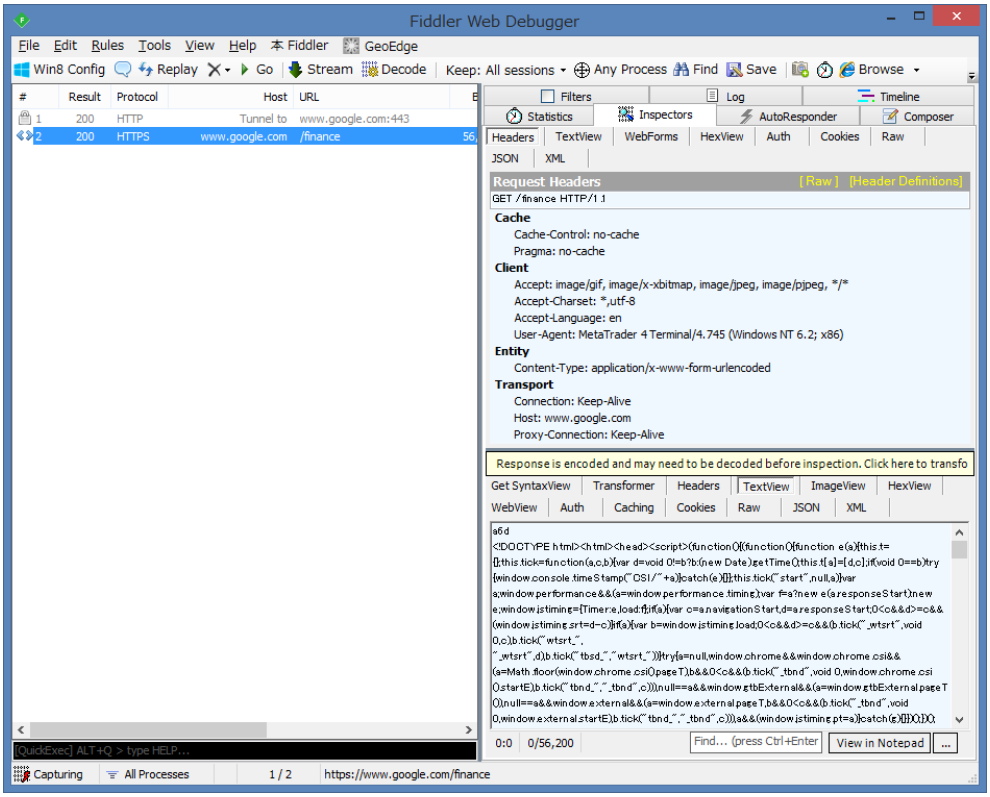
①Fiddler を立ち上げる



②MT4 を立ち上げ、「WebRequest_00」を実行します

③Fiddler を観てみます

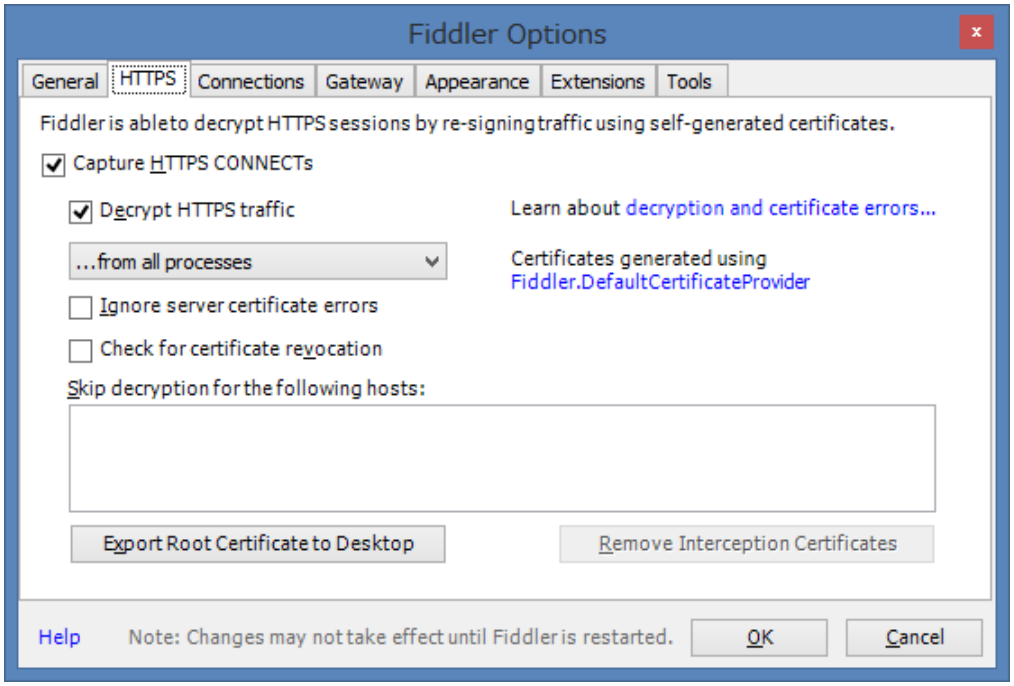




※あとは、下記をクリックすれば内容を解析できる状態になります。

※［Trusted CA list］発行後は、簡単なステップでモニターを開始できました。

④参考；　［Tools］－［Fidler Options...］で設定を確認してみた

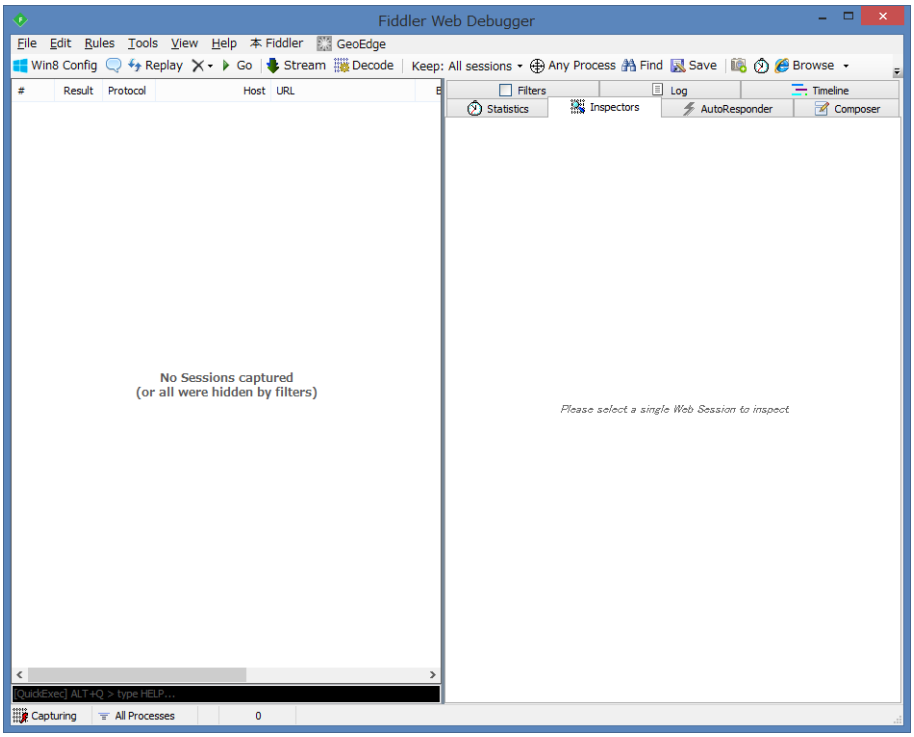


＜補足＞

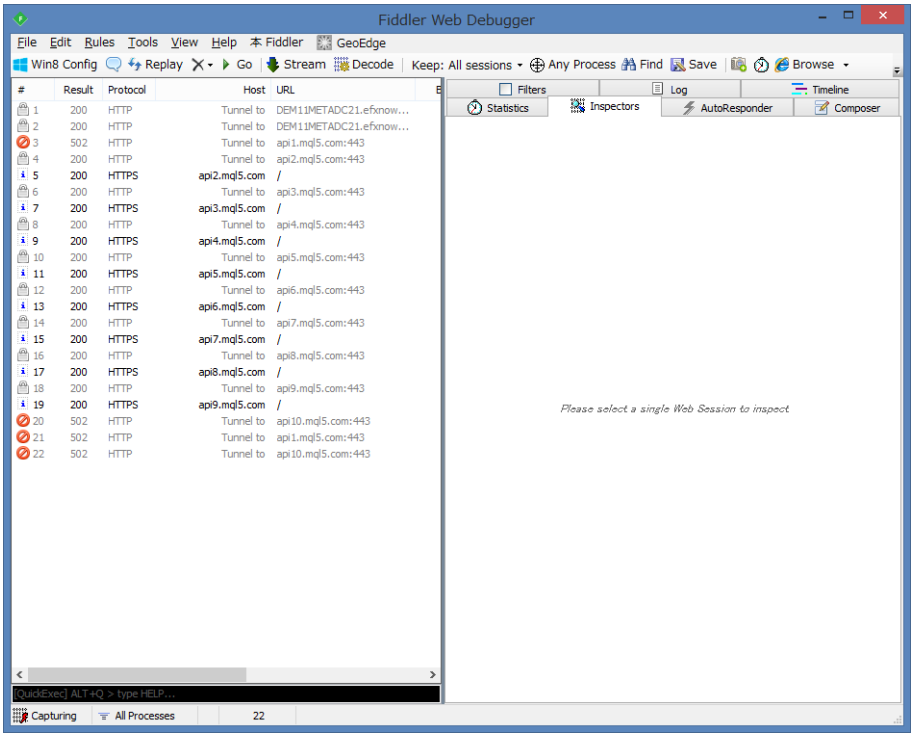
Fiddler が立上り済みの状態で、MT4 を立ち上げ、どのようなセッションが発生するかを観察してみた。（おまけ）

①WEB ブラウザは終了しておきます（影響を消すため）

②Fiddler の［Edit］－［Remove］－［All Sessions］で事前に全てのセッションをクリア



③MT4（プロキシーサーバーは設定済み）を立ち上げる



！！？？**内容はさっぱり判りません**（解析もしてない）。でも色々とやっている！！

以　上