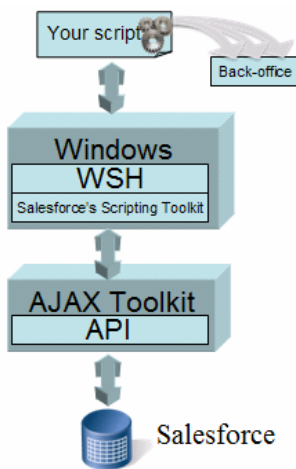


Scripting Toolkit



Scripting Toolkit は、AJAX Toolkit を利用してローカルの JavaScript から Apex API へのアクセスを行うためのツールキットです。

- Scripting Toolkit は API アクセス可能なすべての組織で利用できます
- Scripting Toolkit は AJAX Toolkit と Microsoft Windows Scripting Host (OS にネイティブに含まれます) に基づいています。
- Scripting Toolkit は最新の AJAX Toolkit v8.0 をサポートしています。
- アクセス権のあるあらゆる API、そして API オブジェクトに対し動作を実行できます。
- 非同期呼び出しに対応しており、コールバック関数を用いて結果を処理できます。AJAX Toolkit についてより深く知りたい方は [こちら](#) をご覧ください。
- Scripting Toolkit は全ての Windows マシンで利用可能です。
- あなたのマシン上で動作可能なあらゆる API に活用可能です。(COM オブジェクト, moniker...)
- Salesforce 上やバックオフィスにあるデータの作成・取得・更新・削除などを行うバッチスクリプトが作成可能です。
- Scripting Toolkit は Salesforce とあなたのデータベースの連携を容易にします。



Scripting Toolkit を使うには、JavaScript と AJAX Toolkit、および [Apex Web Services Developer's Guide](#) にある API に関する情報について知っておく必要があります。

既に Ajax Toolkit に慣れ親しんでいる方はすぐに Scripting Toolkit でバッチを作成できるようになるでしょう。

当ドキュメントは Scripting Toolkit を利用したバッチ作成の方法を説明するものです。

Scripting Toolkit を使ったバッチはスクリプトエンジンによって呼び出されるスタンドアロンの Javascript ファイル (*.js) です。バッチはコンソールモード(コマンドライン)で動作します。

Scripting Toolkit はスタンドアロンの Windows Scripting ファイル (*.wsf)です。このスクリプトエンジンそのものに手を加える必要はありません。Scripting Toolkit によって、Ajax の呼び出し・Salesforce データの処理・バックオフィスデータへのアクセスなどの機能を持った Javascript バッチを作ることができます。

Javascriptファイルの作成には、メモ帳はもちろんのこと、お好きな開発環境をご利用いただけます。

Scripting Toolkit の活用例

以下、Scripting Toolkit の活用場面の例です。

- ビジネスルールに則ったデータのロード・抽出・処理を行うバッチが必要なとき
- 何らかのバックオフィス(RDBMS, Web サービス, ファイル交換...)と Salesforce の連携が必要なとき
- より軽量のインターフェースでの簡単な開発、デプロイが必要な時”
- Javascript 以外の言語に精通していない場合
- 項目値の変換を伴った CSV ファイルのロードや抽出を行いたいとき
- データベースにデータを移動させたいとき
- データに基づいたダイナミックなパワーポイントプレゼンテーションを行いたいとき(OLE を活用したレポート)
- 質の悪いデータを判別・収集し、データクオリティをあげたいとき
- データをある Salesforce 組織から別の Salesforce 組織に移動させたいとき
- リストの値に大きな更新を加えたいとき(選択リスト値の更新の後)
- crontab に設定し、重要指標を週次でメール送信したいとき
- 一定の期間ごとにデータを取得して何らかの動作をトリガする場合(例: メールの送信)
- ドキュメントや添付ファイルを抽出し、ストレージスペースを拡げたいとき
- その他

Scripting Toolkit は以下の場面では使わないでください。

- ブラウザ上での動作を必要とするとき⇒Ajax Toolkit の利用をお勧めします。
- Salesforce 上のデータを更新することだけを目的とするとき⇒Apex language の利用をお勧めします。
- Unix を利用してバッチを動作させる必要があるとき⇒Java で開発してください。
- 初歩的なデータのインポートをしたいとき⇒Data Loader の利用をお勧めします。

以下、ケースバイケースの分析が必要な場面の例です。

- 時間との連動性の高いプロセスを動かしたいとき⇒Apex と Scripting Toolkit 両方を利用することをお勧めします。
- サービスとして活用したいとき⇒スクリプトサービスを作るため、特別な開発環境、もしくは srvany.exe を利用することをお勧めします。

不適切な利用の一つの例はデータを毎秒 polling する、といったものです。このような処理は多量の API コールを必要とし、API コールの最大値に達する危険性があります。

Scripting Toolkit のサポートポリシー

Scripting Toolkit の最新リリース版のみがバグ修正と機能強化の対象となります。新しいバージョンがリリースされた時点で、過去のバージョンはその対象外となります。

Salesforce Labs に述べられている通り、当ツールキットには公式サポートがありません。Scripting Toolkit をダウンロードすることですべてのソースコードを取得できます。このツールキットに変更を加えたとしても再配布することはできません。当ツールキットは無料で活用していただけますが、あくまでご自身の責任で運用してください。

Scripting Toolkit のダウンロード

以下の URL からファイルとして保存してください。

scriptingtoolkit.wsf (28 Kb): Scripting Toolkitエンジン
<http://www.interclasse.com/toolkit/?dl=scriptingtoolkit.wsf>

myscript.js (1 Kb) :Scripting Toolkitのサンプルコード
<http://www.interclasse.com/toolkit/myscript.js>

connection.js (51 Kb) : Ajax ツールキット
<https://tapp0.salesforce.com/soap/ajax/9.0/connection.js>

これらのファイルはすべて同じフォルダ内に保管してください。

その他のリソース

ここまで説明した以外にも、Scripting Toolkit、またはその技術について学ぶために様々なリソースをご利用いただけます。

- Microsoft Script Center : <http://www.microsoft.com/technet/scriptcenter/default.mspx>
- An overview of WSH :
<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnanchor/html/scriptinga.asp>
- Sample WSH scripts : <http://www.microsoft.com/technet/scriptcenter/csc/scripts/default.mspx>

Scripting Toolkit と他の言語の比較

以下はツールキットの比較表です。最適なシステムを選んでご活用ください。

ツールキット	Scripting Toolkit	.Net	Office	Java	Ajax	PHP	Perl
言語	VbScript, Javascript	C#, VB	VBA	Java	Javascript	PHP	Perl
要コンパイル	N	Y	N	Y	N	N	N
S コントロールのスキル	95%	50%	50%	60%	100%	40%	30%
ブラウザ上で動作	N	N	N	N	Y	N	N
Web サーバ上で動作	N	Y/N	N	Y/N	N	Y	Y/N
バッチ処理 (コンソール)	Y	Y	N	Y	N	N	Y
必要条件	Windows 2k,XP, 2003 (native)	.Net Framework	MS Office	JDK 1.5	IE/Firefox	5.1.2 + Soap extension	Soap::Lite
デバッグ環境	Visual Studio or script debugger	VS	VBA	jdb	Firefox error console/Firebug	Dbg, nusphere...	Perl -d
IDE	ノートパッドでも何でも	Visual Studio	Integrated VBA editor	Eclipse	Salesforce !		
ソースコードの難読化	screnc	コンパイル	パスワード保護	コンパイル	なし	Nu-coder	?
実行時の制限	None	なし(IIS)	なし	None	Yes (ブラウザ)	Yes (Web サーバ)	なし

Scripting Toolkit のサンプルコード

以下のサンプルコードは myscript.js ファイルにコピー&ペーストすることでご活用いただけます。これらのスクリプトはデータをクエリし、スクリーン上に表示します。API の活用についてより深く知りたい方は Salesforce の API ドキュメントをご覧ください。

```
function Results(queryResult) {
```

```

if (queryResult.size > 0) {
    var output = "";
    var records = queryResult.getArray("records");
    for (var i=0;i<records.length;i++) {
        var account = records[i];
        output += account.Id + " " + account.Name + "¥r¥n";
    }
    WScript.echo(output);
}else WScript.echo("No records matched.");
Toolkit.Quit();
}
var callback = {onSuccess : Results,
                onFailure : function(e){WScript.echo(e)}};
var queryResult =
    sforce.connection.query("Select Id,Name from Account", callback);

```

AJAX Toolkit と Scripting Toolkit の違い

AJAX Toolkit を使用するときは初期化処理が必要となります。Scripting Toolkit ではその必要性がありません。

非同期呼び出し(コールバック関数)を使用しているため、作業終了時にエンジンを止めるコードを必要とします。Toolkit.Quit();

テキストをコンソール画面(標準出力 stdout)に表示するためには、そのメッセージが文字列であるとき、WScript.echo(yourmessage)を利用してください。

スクリプトを実行する

Scripting Toolkit はコマンドラインで実行されます。

「スタート」メニューから「ファイル名を指定して実行」を選択し、「cmd」と入力してエンターキーを押してください。コンソールが表示されます。Scripting Toolkit を保管しているフォルダからカスタムスクリプト(myscript.js)を開いてください。以下のコマンドラインを入力します。

```

cscript //nologo scriptingtoolkit.wsf /?

```

cscript は Microsoft Windows Scripting Host の実行ファイルで、*.js, *.wsf or *.vbs などのファイルを実行することができます。上記の構文では cscript に scriptingtoolkit.wsf というファイルを動かす指示を行っています。//nologo は cscript にバナーの表示を禁止するものです。/? は

Scripting Toolkit にコマンドライン上での活用方法やオプションの表示を求めるものです。

利用方法 :

```
scriptingtoolkit.wsf [/user:value] [/pass:value] [/sid:value]
  [/server:value] [/proxyserver:value] [/proxyuser:value]
  [/proxypass:value] [/debug] [/sandbox]
```

オプション :

値	説明
user	API 利用の権限を持ったユーザー
pass	ユーザーのパスワード
sid	セッション ID(ユーザー・パスワードを指定しない場合)
server	ログインする組織のサーバ名。(例: na3.salesforce.com もしくは na3-api.salesforce.com)
proxyserver	プロキシサーバのホスト名、もしくは IP アドレスとポート番号 (例: proxy.internal.yourcompany.com:3128 or 192.168.0.7:3138)
proxyuser	プロキシサーバに認証を利用している場合のユーザ名
proxypass	プロキシサーバに認証を利用している場合のパスワード
debug	デバッグ情報を標準エラー出力にダンプする
sandbox	test.salesforce.com での利用

例 :

```
cscript //nologo scriptingtoolkit.wsf /user:myself@mycompany.com
  /pass:secret
```

```
cscript //nologo scriptingtoolkit.wsf /debug /server:emea.salesforce.com
  /sid:Xabu.ZcS.....9AsNCSeX5jsUoLXQ=
```

ユーザ名・パスワードが与えられない場合、スクリプトは IE のブラウザセッションから SID を利用することを試みます。

プロキシサーバを利用する場合 :

```
cscript //nologo scriptingtoolkit.wsf /proxyserver:192.168.0.7:3128
  /proxyuser:jla /proxypass:jla
```

Windows の設定を調べ、何らかの問題を検証したいとき、以下のコマンドラインを実行してください。

```
cscripct //nologo scriptingtoolkit.wsf //Job:diagnose
```



以下のコマンドラインを使用することでバッチのファイルへの出力を書き換えることができます。

```
cscripct //nologo scriptingtoolkit.wsf /user:xxxx@yyy.com /pass:toto  
>myfile.txt
```

/debug パラメータによる出力は標準出力(stdout)ではなく標準エラー出力(stderr)のままになります。

スクリプトをデバッグする

Visual Studio、もしくは Microsoft Script Debugger が利用できます。ブレイクポイントを付与し、オブジェクトの値を確認できるようになります。

スクリプトのエンコーディング

コードを他人に見られたくない場合は、[Microsoft Script Encoder](#)でエンコードが可能です。エンコード後はコードの書き換えが不可能になりますが、このエンコーディングは双方向関数なので、何らかのツールによってソースコードが取得できる可能性があります。

処理速度を向上させる

上級者専用

Scripting Toolkit はリモートサーバ上の connection.js Javascript ファイルをリクエストします。このファイルをダウンロードし、ツールキットと同じフォルダ内に入れると参照の対象をローカルフォルダに変更できます。これにより、バッチを起動するたびにダウンロードに伴う数秒の待ち時間と帯域幅が節約されます。

ツールキットオブジェクト

Scripting Toolkitを使用すると、自動でインスタンス化された ToolKit オブジェクトへアクセスが可能になります。

このオブジェクトはパブリックのプロパティとメソッドを持っています。

プロパティ:

値	説明
className	"ScriptingToolkit"
User	コマンドラインパラメータで指定されたユーザ /user:
Password	コマンドラインパラメータで指定されたパスワード /pass:
Server	コマンドラインパラメータで指定されたプロキシのサーバアドレス /server: ベーシック認証あるいは NTLM によるプロキシサーバの認証に対応しています
SID	コマンドラインパラメータで指定されたセッション ID /sid:
HasToExit	内部利用の際は常にfalse です。Quit()メソッドが呼び出されたときtrue となります。
ErrCode	エラーがない場合 0 を返します。メソッド呼び出しの際にエラーが発生した場合は 0 以外の値を返します。

メソッド::

値	説明
ClearError()	エラー情報をクリアし、エラーコードを0に設定します 入力:なし 出力:なし
SetError(errorcode, errortext)	エラーコードとエラーメッセージを設定します 入力: errorcode : integer, errortext : string 出力 : なし
ErrText()	エラー情報をダンプするために、現在のエラーコードからエラーメッセージを取得します。 入力 : なし 出力 : エラーコードによる標準のエラーメッセージとコンテキ

値	説明
	ストに含まれるカスタムのエラーメッセージ
VerifyLoginParameters()	(内部利用のメソッド) 入力：なし 出力：なし
Debug(text)	デバッグ情報の表示 入力：string 出力：なし コマンドライン上で /debug パラメータが設定された場合、このテキストは標準エラー出力(stderr)に出力されます
Login()	(内部利用のメソッド) 入力：なし 出力：なし
Quit()	スクリプトを停止します 入力：なし 出力：なし コールバック関数を利用した非同期でのリクエストが可能であるため、終了時には明示的にこのメソッドを呼び出す必要があります。 ジョブにこのメソッドが含まれない場合、スクリプトは終了しません。(Ctrl+C で終了してください)
SOQL2CSV(SOQL, Header, Sep, cr, strnull)	SOQL クエリー結果を CSV ファイルに変換します 入力：SOQL：結果セットを返す SOQL クエリ文 入力：Header：(オプション)：CSV ファイルの最初の行に指定するカラム名の配列。SOQL に含まれている項目名と一致する必要があります。配列の順番がは CSV ファイルの項目の順番になります。 入力：Sep：(オプション)：区切り値。デフォルトは “,” (カンマ) 入力：cr：(オプション)：改行文字。デフォルト値は “\r\n” 入力：strnull：(オプション)：null 項目値に対して割り当てる文字列。デフォルトは “” 出力：CSV 形式の文字列(エラー時は null)
CSV2Array(path, file)	CSV ファイルをメモリ上の配列にロードする 入力：path：ファイルが存在するフォルダへのパス

値	説明
	入力: file : CSV ファイル名 (パスを含まない) 出力 : 1次元目をレコード、2次元目をフィールドとする2次元配列。フィールド値にはカラム名を利用して参照が可能。
InsertFromArray(sObject,theArray,size)	メモリ上の配列からセールスフォースヘデータを挿入する 入力 : sObject : Salesforce オブジェクトの名前 入力 : theArray : CSV2Array() の結果とおなじ配列 入力 : size : (オプション) : Ajax 呼び出しごとの最大のレコード数 出力 : 成功時は true, エラー時は false 条件: 配列のインデックス名は Salesforce の項目名と一致している必要があります。
DeleteFromSOQL(SOQL,size)	SOQL クエリに合致するレコードを削除します 入力 : SOQL : Id を取得する SOQL クエリ 入力 : size : (オプション) : Ajax 呼び出しごとの最大のレコード数 出力 : 成功時は true, エラー時は false
sObject2OracleCreateTable(sObjectName)	SObject の構造から Oracle にテーブルを作成する SQL クエリを作成します 入力 : sObjectName : SObject 名 出力 : SQL 文 (エラー時は null)
StringToFile(str,filename)	与えられた文字列を中身としてファイルを作成します 入力 : str : ファイルの中身の文字列 入力 : filename : 作成 / 上書きするファイル名 出力 : なし
FileToString(filename,unicode)	ファイルの中身を読み込みます。性能を保つためにファイルサイズは 1MB よりも小さくある必要があります。 入力 : filename : 読み込むファイル名 入力 : unicode : ファイルが Unicode ファイルとして読み込まれる必要がある場合に true を指定 出力 : ファイルの中身
StartTimer()	操作時間の記録のためにタイマーを開始します 入力 : なし 出力 : なし

値	説明
StopTimer()	操作時間の記録のためにタイマーを停止します 入力：なし 出力：なし
TimeDiff()	ベンチマーク時間を取得します 入力：なし 出力：ミリ秒単位の時間
SendMail(from,to,subject,body)	CDO コンポーネントを利用してメール送信します。設定情報を見るには Job:diagnose を利用してください 入力：from：メールの送信者 入力：to：メール受信者 入力：subject：メールの件名 入力：body：メールの本文。テキストでも HTML でも可(自動検出されます) 出力：成功時は true、エラー時は false
GetFile(SOQL,filename)	Salesforce 上のリモートドキュメントを取得して、ローカルに保存します 入力：SOQL:1レコードのみを返す Body 項目へのクエリ文。 例： "Select Body from Document where Id='015300000000CJuAAM'" 入力：filename：作成あるいは上書きするファイル名 出力：成功時は true、エラー時は null

Scripting Toolkit のサンプルスクリプト

以下のサンプルを `myscript.js` に保存し、コマンドラインを使用して実行してください。

Hello World

```
WScript.echo("Hello World !");
Toolkit.Quit();
```

最初の行はコンソールにテキストをプリントするメソッドです。二行目は Scripting Toolkit に終了を知らせています。

このサンプルでは salesforce のデータにはアクセスがありませんが、内部的にアプリケーションにログインという動作が行われています。

同期的なデータ取得

```
try{
  var queryResult =
  sforce.connection.query("Select Id,Name from Account");
  if (queryResult.size > 0) {
    var output = "";
    var records = queryResult.getArray("records");
    for (var i=0;i<records.length;i++) {
      var account = records[i];
      output += account.Id + " " + account.Name + "¥r¥n";
    }
    WScript.echo(output);
  }else WScript.echo("No records matched.");
}catch(e){
  WScript.echo(e);
}
Toolkit.Quit();
```

非同期的にデータを取得するには、前の章のサンプルを参照ください。

スクリプトをベンチマークする

```
Toolkit.StartTimer();
var queryResult =
    sforce.connection.query("Select Id,name,Industry from Account");
Toolkit.StopTimer();
WScript.echo(Toolkit.TimeDiff()+" milliseconds");
Toolkit.Quit();
```

コードを実行するのにどれくらいの時間がかかったかを調べるためにタイマー機能が提供されています。

CSV ファイルにデータを抽出する

```
var strTest =
    Toolkit.SOQL2CSV("Select type, Id, name, Industry from Account",
        ["Name", "Id", "Type", "Industry"], ";");
WScript.echo(strTest);
if (strTest) Toolkit.StringToFile(strTest, "test.csv");
Toolkit.Quit();
```

以下の 2 つのステップがあります :

- 結果セットを CSV フォーマットの文字列として取得する
- 得られた文字列をファイルに保存する。

文字列が null の場合はエラーです。Toolkit.ErrText() を参照してください。

ファイルを読み込む

```
WScript.echo(Toolkit.FileToString("test.csv", false));
Toolkit.Quit();
```

ファイルは Ascii、Unicode のいずれかの形式となります。(unicode では二番目のパラメータは true となります)

利用可能な SObject を取得する

```
var describeGlobalResult=sforceClient.describeGlobal();
var types=describeGlobalResult.types.toString().split(",");
for (i=0;i<types.length;i++) {
    WScript.echo(types[i]);
}
```

```
}  
Toolkit.Quit();
```

SQL クエリで利用可能な SObject のリストを検索できます。

SObject の詳細情報を取得する

```
var describeResult=sforceClient.describeSObject("Attachment");  
var props=  
  ["activateable","createable","deletable","custom","keyPrefix",  
   "label","labelPlural","layoutable","mergeable","name",  
   "queryable","replicateable","retrieveable","searchable",  
   "undeletable","updateable","urlDetail","urlEdit","urlNew"];  
for (x=0;x<props.length;x++){  
  WScript.echo(props[x]+"="+describeResult[props[x]]);  
}  
for (recordNumber in describeResult.fields){  
  WScript.echo("Field " + recordNumber + ":" +  
    describeResult.fields[recordNumber].name + " (" +  
    describeResult.fields[recordNumber].label+ ")");  
  for(columns in describeResult.fields[recordNumber]){  
    //if(typeof(describeResult.fields[recordNumber][columns])!='function')  
    //WScript.echo(columns+"="+describeResult.fields[recordNumber][columns  
    ]);  
  }  
}  
Toolkit.Quit();
```

このスクリプトは sObject 内の利用可能なフィールドとその属性をリスト化します。

SObject に基づいてオラクルの”create table” SQL 文を生成する

```
SQL=Toolkit.sObject2OracleCreateTable("Account");  
if(SQL){  
  WScript.echo(SQL);  
}else{  
  WScript.echo(Toolkit.ErrText());  
}  
Toolkit.Quit();
```

制限値以上の数のレコードを処理する

```
for(x=0;x<queryResult.size;x++){
    //your code here to use queryResult.records[x]);
    //...
    if(queryResult.done=='false')
        queryResult=sforceClient.queryMore(queryResult.queryLocator);
}
```

これはバッチサイズに縛られた結果よりも多くのレコードを処理する方法です。

ドキュメントや添付ファイルを取得する

```
if(Toolkit.GetFile("Select Body from Document "+
    "where Id='0153000000000CJuAAM'", "test.doc")){
    WScript.echo("success!")
}else{
    WScript.echo(Toolkit.ErrText());
}
Toolkit.Quit();
```

このスクリプトはリモートのファイルをローカルに書き出します。

BodyフィールドはSOQL文中にかならず必要です。クエリごとに一つのファイルのみが取得されます。これはIDによってフィルタリングが可能です。

内部スケジューラー

```
var frequency=10*1000;
function myScheduler(){
    WScript.echo(Date()+" scheduled.");
    //Run your scheduled code here
    window.setTimeout( "myScheduler()", frequency);
}
window.setTimeout( "myScheduler()", frequency );
```

スクリプトを数分おきや数時間おきに、もしくは毎分走らせるために特別なスケジューラーを使う必要性はありません。

頻度はミリ秒で指定します。

- 10 秒毎 : 10*1000
- 毎時 : 60*60*1000

これはブラウザでの S コントロールの場合と同様に使えます。

Email を送信する

```
Toolkit.SendMail("Jean-Luc Antoine <antoinejl@xxartabus.com>"
, "Jean-Luc Antoine <xxantoinexx@xxsalesforce.com>"
, "Just a test"
, "This is a text body¥nwhich has two lines.");
Toolkit.SendMail("Jean-Luc Antoine <antoinejl@xxartabus.com>"
, "Jean-Luc Antoine <xxantoinexx@xxsalesforce.com>"
, "My second email"
, "<html><body><h2>Cool !</h2>Very nice HTML</body></html>");
```

メールの送信はわずか一行のコード(引数は from, to, subject, body)で実行されます。Body はテキスト、または html での記述となります(自動検出されます)。技術的なパラメータは CDO の設定で定義されていますが、これは診断モード(//Job:diagnose)を利用することでダンプすることができます。詳しくは当ページの上部を参照ください。

SOQL に基づいてデータを削除する

```
if(Toolkit.DeleteFromSOQL(
"Select Id from Account where Name like 'ZZZyourprefix%'")){
WScript.echo("Succesfully deleted");
}else WScript.echo(Toolkit.ErrText());
Toolkit.Quit();
```

条件フィルタを指定してレコードを簡単に削除できます。

Salesforce に CSV ファイルを挿入する

```
var CSVResult=Toolkit.CSV2Array("c:¥¥temp¥¥", "account.csv");
if(Toolkit.InsertFromArray("Account", CSVResult, 200)){
WScript.echo("Succesfully inserted");
}else WScript.echo(Toolkit.ErrText());
Toolkit.Quit();
```

この例では以下の内容のファイルを利用しています:

```
NAME, INDUSTRY
My new account 1, Manufacturing
My new account 2, Services
My new account 3, Manufacturing
My new account 4, Services
```



この Scripting Toolkit は Jean-Luc Antoine (antoinejl @ artabus.com) によって開発されました。

原文: <http://www.interclasse.com/toolkit/>

日本語訳: 株式会社セールスフォース・ドットコム