PowershellでExcelを制御しようとして何の成果も得られなかった話

自己紹介

-- お前は誰だ？オレの中のオレ --

中村です。

- 炎上火消しエンジニア。
- 今をときめくSESで働いてる。
- 肩書きは無い。
- コラム書いてました。

101回死んだエンジニア
http://el.jibun.atmarkit.co.jp/101sini/

私が何をやろうとしたか

-- 渇望と葛藤の狭間にて --

作りたかったもの

- PowershellでExcelを操作する汎用的なフレームワーク。
- Excelしか無い環境でも大活躍。
- コマンドラインだけでクールにExcelを制御。
- 面倒くさいドキュメントの作成の手間を大幅削減。

現実は厳しかった

- Excelが汎用過ぎてフレームワークに収まらない。
- 汎用性を追求するほど浮かんでくるマニアックなテクニック。
- 思いついたマニアックなテクニックで事足りてしまった。
- むしろスクリプトが不要になった。

結果。

皆さんには、
私の残念な経験を糧にして、
明日を力強く生き抜いて頂きたいです。

END

まずは知っておくべきこと

-- 明日を強く生き抜くために --

イケてないExcelの悪手を覚えておこう

- シートが何十個もあるExcelファイル
- マクロで記録した内容をそのまま使う天然難読化VBA
- 全ての機能を詰め込んだ万能Excelファイル
- クソフォーマット

シートが何十個もあるExcelファイル

作成されたであろう理由：

データの一元管理。「このファイルさえ見れば全てが分る」ようにすれば、業務を簡略化できるという意図。

実際：

シートが多すぎて逆に見にくい。目的のシートを見つけるまでが大変で、更新されなくなることが多い。

天然難読化VBA

作成されたであろう理由：

VBAの記録機能を活用すれば、高度な技術が無くても自動化や効率化ができるという意図。

実際：

技術も無しに自動化や効率化ができるほど世の中は甘くない。生成されたVBAのコードをリファクタリングしないので、やればやるほどブラックボックス化する。

万能Excelファイル

作成されたであろう理由：

全ての仕組みを集約すれば非常に効率の良いシステムが組めるはずだ。

実際：

整理をせずに集約するので余計にぐちゃぐちゃになる。中途半端な実績が出ると同時に、仕組みが複雑化して収拾がつかなくなる。

クソフォーマット

- あの有名なExcel方眼紙
- 何画面にもわたる巨大なマトリクス表
- 見出しが画面の2/3を占めるクソ表
- 芸術作品のような関係図

・・・・等々。

読めないものを書かないでくれ！

Excelでコーディングで活用する前に

- コーディングする前にExcelを活用できるようになる。
- コーディングは無茶をかなえる魔法ではない。
- Excelの機能で済むものはExcelの機能で済ます。

こういうバッドノウハウを
止めるだけで
充分に効率は上がります。

なぜPowershellか

-- 力こそパワー --

規模観に合ったやり方を選ぶ

同じファイル内での簡単な処理 (小規模な仕組み)
→ VBA

複数のファイルをまたいでの**転記** (中規模な仕組み)
→ Poewrshell

何でもできちゃう凄いやつ (大規模な仕組み)
→ プログラマーに金を払って作ろう

転記とは

てんき【転記】

《名・ス他》記載事項を他の帳簿などに書きうつすこと。
?「―漏れ」

簿記などで活用される手法。データを処理する上で最も重要。この手法をうまく使いこなせば、コンピュータが無くても複雑な計算処理をこなすことができる。

勝利の方程式

高度なテクニック ＋ IT技術

＝

PowershellでExcelを操作するメリット

- コードとExcelファイルを分離できる
- VBAでは困難な複数ファイルの処理がやりやすい
- 別のシステムとの連携がやりやすい
- モジュール化しやすい
- PSObjectでデータを扱える

PowershellでExcelを操作する

-- 能書きだけではなく、実際に操作する方法を説明します。--

Powershell でExcelを制御する仕組み

難しい言い方：

COMオブジェクトを呼び出して、メソッドやプロパティを通してExcelの処理を実行する。

砕けた言い方：

Excelを見えないように起動してVBAと似た感じで処理する。

注意点

- Excelの起動と停止
  → やり方を間違えるとプロセスが残る。

- コードの書き方の調べ方
  → 操作方法はPowershellではなくVBAで調べる

- PowershellでなくてもRubyやPythonでも同じことができる
  → ただしWeb上に出てる情報は起動するとこまで

Excelファイルを開く

1. Excelを見えない状態で起動

```
$Excel = New-Object -ComObject Excel.Application
$Excel.Visible = $false
```

2. Excelのファイルを開く

```
$File = Excel.Workbooks.Open(<開きたいファイル>)
```

3. いろいろな処理をする

「VBS Excel」のキーワードでググろう。

Excelを閉じる

1. ファイルを閉じる
$File.close($false)
$File = $null

2. Excelを終了する
$Excel.quit()
$Excel = $null
[GC]::Collect()

こうすると便利

- セルの値をListや名前付き範囲で管理する
  → コーディングの際に番地を指定せずに済む

- Excelのファイルは用途ごとに分ける
  → 保守性が飛躍的に向上

- 計算はPowershellで行う
  → 処理内容がブラックボックス化しにくくなる

注意点

- 雑に処理を停止するとExcelが立ち上がったままになる
- 使った変数を解放しないとメモリに無駄なデータが残る(らしい)
- ファイルの保存処理を雑にやるとダイアログが出てくる

何かあった時にGUIで操作できないと詰む。サーバでの大規模処理に向かない。

VBSと比較したPowershellの利点

- データのフィルタリングがやりやすい
- テキストファイルへの書き出しが楽
- プロンプトからコマンドでExcelを操作できる
- 優越感に浸れる

何がイケていなかったか

-- 理屈では説明できない何かに躓いた --

Excelをコーディングする限界点

- そもそもコーディングが前提のソフトではない。
- Excelの起動と停止やファイルを開く動作があるので、処理速度が致命的に遅くなる。
- 動作が複雑なので多くのパラメータが必要になってしまう。
- コーディングしても手作業で好き放題やられてしまう。

Excelの基礎スキル問題

・ 名前付き範囲やListの機能を理解する必要がある
→ その機能を使いこなしている人が少ない

・ フォーマットを制約する必要がある
→ Excelにフリーダム(笑)を求める人に受け入れられない

Excelの仕様問題

- SQLのように体系的に値を管理したい
  → アドレスでしか値にアクセスできないので死ねる。

- セルのデータを変数で扱いたい
  → Excelの仕様上無理。どうしてもコードが複雑になる。

レイアウト問題

- 書式を制御したい
  → 直接Excel触った方が早い。

- 印刷を制御したい
  → 計算が複雑になるので労力に合わない。

実際にやってみた所感

- Excelはプログラミングのコンポーネントじゃない
- Excelで汎用的なフレームワークは無理
- 心のこもったレイアウト(笑)を求めるなら手でやろう
- ゴチャゴチャやるよりフローを整理する方が得策
- コーディングするならExcel意外の仕組みを使うべき
- Excelへの過剰な期待はやめておこう

結論

-- 成果が得られなかった人の報告 --

Excel単体で事は足りた。

Powershell単体で
いろんなことができた。

なんだかだで
オレ、ハッピー。

この経験から学んだこと

- Powershellだけでもいろいろなことができる。
- Excelの機能を使いこなせるようになる方が先。
- コーディングするならExcelである必要は無い。
- PowershellでExcelを制御するなら転記に留めるのが良い。
- Excelに出力するよりCSVに出力した方が無難。

END