PowershellのExcelっぽい使い方を切り開く

自己紹介

- 名前は「中村」平凡な名字です。
- SESで火消しやってます。
- 他、特に語るほどの肩書きは無い。
  (@ITでコラム書いてた)

アジェンダと言われるもの

- 前回の京都での登壇について
- 私がPowershellをいじりだした理由
- Powershellってどうなん？
- Powershellの導入方法

前回の京都での登壇について

あの夏、僕は京都で・・・

- いつもだったらLibreOfficeで登壇しているのだが、今年は何を思ったのか個人で登壇した。
- テーマはPowershellとExcelの連携。
- 登壇後、ホテルに泊まり翌日、観光で鉄道博物館に行った。

そして私は鉄道博物館で自分の過ちに気づく。

やってたPowershell、
オープンソース版じゃないやん！

しかもExcel、
全然オープンソースじゃないやん！

OSC(オープンソースカンファレンス)
でマイクロソフトのネタで登壇。

今更どうしようもないので、
しらをきり通すことに決めた。

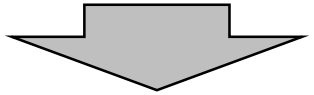
今回のテーマについて

- Excelと連携して行うデータ管理について。
  (LinuxやMacの人は、Excelの代わりにLibreOfficeでOK！)
- SESのクソ環境でも使えるサバイバル術として。
- 「簡単で分かりやすく」する気はこれっぽちも無い。
- 1000回コマンドを実行しないと、私の言いたいことは伝わらないと思う。

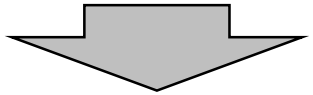
私がPowershellを使う理由

Powershellを弄りだしたいきさつ

1. プロジェクト(万年炎上SES)での
データ管理がめちゃくちゃだった。



2. 整理してぇ！



3. Poewrshellに行き着いた。

データ管理とは

- 台帳などでデータを適切に作成・更新する。
- データを再利用できる形で保管する。
- ソフトウェアの機能を使用してデータを分析に活用する。

→ 一般的にはExcelを使って管理されている

Excelが崩壊する臨界点

- 長文を扱を入力をしたとき
- データがツリー構造をとるようになったとき
- 簿記のような転記処理が必要になったとき

→ これを解消できる市販のソフトは今のところ無い

SESでの現実

- 印刷想定のフォーマットで値を直書きして管理。
  → データとして使えない「ネ申ｴｸｾﾙ」降臨。
- 適当にExcelにぶち込んで「完成」扱い。
  → 工夫という概念が無い。
- IT専門のはずだが一般的なOLさんよりITリテラシーが低い。
  → ソフトウェアの機能を活用するという概念が無い。

当然のごとくデータ管理崩壊！

何がまずいか

- 使いこなせないわりに何でもExcel
  → きっとExcelに片思いの恋をしてるんだね
- データの確認は全部目視
  → エンジニアとしての工夫の余地が無くなる
- なんでも一元管理
  → 無計画に情報を集約して用途が不明になる
- なんでもかんでも一発ノーミス想定
  → 必要以上に自ら難易度を上げてしまう

個人的に行き着いた結論

炎上して当然

これじゃヤバいのでやったこと

- 荒立てたくないので愛想笑いでスルー。
- Accessでプロジェクトで扱っているデータを整理。
- シェル芸に手を出した。 *1

※1
シェル芸 とは、主にUNIX系オペレーティングシステムにおいて「マウスも使わず、ソースコードも残さず、GUIツールを立ち上げる間もなく、あらゆる調査・計算・テキスト処理を CLI端末へのコマンド入力 **一撃で** 終わらせること」（USP友の会会長・上田隆一による定義）である。

プラン1　愛想笑い

- それぞれの立場相応に勤めを果たしていると信じてみる。
- ことを荒立てるより穏やかに言った方が結果がでるのでは。
- みんながやってるので付和雷同してみた。

そもそも笑ってられる状況じゃなかった。

- 実際は「愛想笑い」ではなく「苦笑い」だった。
- ヒアリング調査の結果、誰も何も把握していなかった。
- 情報を把握できないと発想が幼稚になることが分かった。

愛想笑いで問題解決したいなら、
悪魔に魂を売り払ってから死ぬ気で笑え。

プラン2　Accessによるデータ管理

- Excelより複雑なことができる
- 入力しやすいインターフェイスが作れる
- リレーショナルなデータベース

むしろ炎上。

- 複雑になりすぎて負債化した
- 入力はしやすいが共有させるにはハードルが高い
- リレーショナルが故にテーブルのデータが自由にいじれない

ExcelをDisっておいて
逆にExcelの利点を見直す機会になった

Accessの問題点

- とっさの処理でスピードが出ない。※1
- レポーティングの機能が致命的に弱い。
- 一人で使うには大げさチームで使うには貧弱。
- 共有されると変に使い込まれて変更できなくなる。

※1
使用する機能を最低限に絞って処理速度を上げる「三分間Access」というやり方を編み出したが、Excelと同様の問題が露呈したので使うのをやめた.

プラン３　シェル芸に手を出した

- 何やらすごいらしい
- 最低限の環境で実行できるらしい
- Unixの一般的なコマンドで処理を完結できるらしい

三秒で撃沈。

- **そもそも現場にWindowsしか無い。**
- フリーソフト禁止(Cygwinとか使えない)。
- テーブル系のデータで列を指定する手段が番号なのがきつい。
- 難しかった。

己の未熟さを噛みしめることになった。

そこで閃いた。

そうだ、Windowsには
Powershellがあるじゃないか！

なぜPowershellなのか

- Windows があればデフォルトで使える
  → 環境的な条件をクリア

- デフォルトでデータテーブル（っぽい機能）が使える
  → Unix系のコマンドよりも構造化したデータに強い

- CSVやXML、JSONなどにいろいろなフォーマットに対応
  → Accessとは勝手は違うが複雑な構造のデータが扱える

Powershellを使って便利だった点

- データの検証が捗る
- データの流し込みによるコマンドの作成に便利
- テーブルの加工がやりやすい
- グルーピングによる分析が便利
- テキストへの出力がフレキシブル

炎上プロジェクトで使うときの注意点

- Powershellでの作業は一人で完結させる
  → 訓練が必要な作業なので連携の難易度が高い。

- フォーマットをそろえるまではExcelでの地味な作業 *1
  → 興味のある方は次の枠の登壇でお話します。

- ミスしたらPowershellのせいにされる
  → だが気にするな。己の信じたやり方を貫こう。

余談

　　一応、SESのずさんな業務に対する対抗策として考えたPowershellを利用したデータ管理の技法に呼び方を付けている。

対SES強行機構計算術

力技

ちからわざ

おまけ

　実はシェル芸ができるようになろうと思ってPowershellを触るようになったが、全然別の方向に進化を遂げてしまった。

とりあえず呼び名を付けたのもシェル芸のオマージュ。

Powershellってどうなん？

Powershellを使って感じた利点

- あくまでシェルなのでワンライナーが書きやすい
- フィルタリングの機能が分かりやすい
- グルーピングの機能が強い
- 何をするコマンドかが一目瞭然

Excelとのコマンドレベルでの連携

- できるが言うほどフレキシブルでもない。
  (ワンライナーでは無理)
- Excelの関数で片づくことはExcelで完結した方が楽
- フォーマットとか書式と言われると死ぬ
- 仕組みが複雑化するのでVBAや関数との連携はやめた方がいい

CSVデータを介して連携するのが楽。

Powershellの特徴

コマンドが <動詞> - <名詞> という構造

- コマンドを見ただけで何をするものか想像がつく
- いちいちコマンドを覚えなくても入力補完でなんとかなる

コマンドを実行するとオブジェクトが返ってくる

- Unixのコマンドとコンセプトが違う
- C#のノウハウが流用できる
  (らしいが私はC#を知らないので分からない)

Powershellの考え方

- 出力がオブジェクト
- コマンドを打つとExcelのシートが一枚出てくる感覚
- パイプを通すと一行ずつ処理される

シェルとプログラミングのハイブリッドのような挙動をする。
Rubyでワンライナーを書く人なら感覚を掴みやすいと思う。

シェルとして使いこなすために

- パイプで繋いでいくスタイルでコーディングする
- 目の前のことを絞って確実にこなす
- 短期決戦型コーディング力が問われる

ワンライナーにこだわる必要は無いが、
ワンライナーで済ます方がこじれにくい。

短期決戦型コーディング力について

一般的なコーディング力:

可読性が高くメンテナンス性の高いコードを書く力。長期的に使用するコードを書くときに必要。

短期決戦型コーディング力：

可読性、メンテナンス性は一切無視。目の前のタスクに対して最短で答を出すためのコードを書く力。

Powershellの実践投入

今回のコンセプト

- 今回はExcelと連携したデータ処理にコンセプトを絞る
- ログの解析や処理の実行結果の分析にも活用可能
- テキスト出力への連携に繋げると活用の幅が広がる

ツールの役割分担

Excelでやる作業（主に前処理）

- データの入力
- テーブル形式への整形
- データ数に変更が発生しない演算

Powershellでやる作業（主に加工）

- データの分割やグルーピング
- データの検証
- テキストデータへの加工

Powershellでのデータ処理のフロー

1. CSVやテキストを変数に読み込み
2. データの加工、分析
3. 出力

使い方のフローをまとめておくと
必要な機能の切り分けがしやすくなる。

実践投入におけるポイント

- PSObjectでデータを処理
- SQLの感覚は一旦忘れる
- 基本的に書き捨て

なぜPSObjectを使うか

- コマンドを打つと出力されるのがコレだから
- Exportとインポートが簡単
- 意外と直感的に扱える (限定的)

配列やハッシュだけで済まそうとすると
コードが複雑になって逆に難しくなる。

CSVデータの出力と入力

<変数> = Import-CSV -Encording Defaut <ファイル名>
CSVファイルをPSObjectとして変数に読み込む。

<変数>｜Export-CSV -Encording Defaut <ファイル名>
PSObjectをCSVに出力。

あとは、Select-Object と Where-Objectの簡単な使い方が分かれば、データのフィルタリングが簡単にできるようになる。データの分割出力でのオペミスを激減して時間を大幅に削減できる。

PSObjectでの処理例

<PSObject>｜Select-Object <項目>
→ 項目を選択して出力する

<PSObject>｜？<項目> -eq <値>
→ <項目>が<値>と同じものを抽出。(記述を省略する手法を利用)

<PSObject>.count
→ データの数を数える

※ このくらいまでは直感的に操作できる

ちょっと難解なワンライナーの一例

%{ <変数1> } –PipelineVariablel <変数1の仮の名前> | %{ <変数2> } –PipelineVariablel < 変数2の仮の名前> | ?{ <変数1のキー> -eq <変数2のキー> } |

Select-Object @{n="<PSオブジェクトのプロパティ名>"e={変数1の仮の名前や変数2の仮の名前を使用した計算式}}....

SQLで言う"join"のような処理で得た出力に演算を加えて加工するワンライナーの構文

<変数> | Group-Objece <項目> | Select-Object @{n="<項目名>";e={$_.Name},@{n="<項目名>":e={<グルーピングしたPSObjectの個数やら合計を計算する式>}}...

項目名でグルーピングしたデータを処理して一覧表示するワンライナーの構文

※ ここくらいまでくるとキレる人続出

SQLは一旦忘れよう

- デフォルトでJoin構文を実現する仕組みが無い ※1
- 入れ子(ジャグ配列)の処理がSQLの概念で説明がつかない
- 処理によってはコードで対応した方が速い
- つーか、うだうだ言ってないでSQLの壁を越えろ。

※1 ワンライナーである程度対応可能

SQLとPowershellの処理フローの違い

SQL（フローがスマート）
テーブル → SQL文実行 → テーブル → 加工 → 結果

Powershell（フローがカオス）
テーブル → パイプ → 得体の知れない何か → パイプ → 結果

仕組みも概念も違うので割り切って使おう。
（これらは私の感じたイメージです）

基本的に書き捨て

- スクリプトを組むとメンテナンスが発生する
- スクリプトに組むまでも無く目的が達成しよう
- 変な仕組みを構築すると墓穴を掘りやすい

※ スクリプトを組むのは、書き捨てを繰り返してやり方が確立してから。

使用上の注意

- メンテナンス性無視
- うまくいくまで何度でもトライ
- ワンライナーの構文と動きを暗記
- スモールミッション・スモールクリアの繰り返し

※ コンセプトに対しての条件で、Poewrshell全般に対しての条件ではありません

実際にやってみた感想

- 無意識に思考回路がExcelベースでの効率の悪い処理で固まっているので、意図的に作り替える必要がある。
- CUIに慣れるまでは苦労したが慣れたら快適
- 口で説明しても無理。手を動かして感覚を掴もう。
- 現場で実際にPowershellを使うのはとても勇気が要る。

ツッコミに対する事前返答

他の言語使った方が効率よくね？

- 純粋にデータ処理だけなら、PythonのPandasを使う方がいい。
- Powershellに慣れないと使い物にならない。
- ちゃんとプログラムみたいに書きたいなら何かしらの言語を使用する。

※ 使う言語は何であれ、CUIでデータ処理の感覚を培えると強い。

Powershellって
処理速度遅くないっすか？

- 確かに遅い。
- 起動ももっさり。
- 大量のデータを扱わなければ問題は無い。

遅いとか重いとかガツガツ言われると、
心が苦しくなるので優しくしてください。

Dosコマンドで十分じゃね？

- そういうレベルのことしかやってないならそうだね。
- DosコマンドはCOBOLやるようなもんだ。
- Powershellは使い込まないとなかなか良さが分からない。

寝言ぼやいてないで
自分の技術をアップデートしよう。

Powershellの使用に関して

- コマンドが長い
- パイプの動きがつかみづらい
- .countの使い方が気持ち悪い

...etc

うだうだ言わずに使って慣れろ。

SES勤務ですが、このやり方でプロジェクトを成功させたいです！

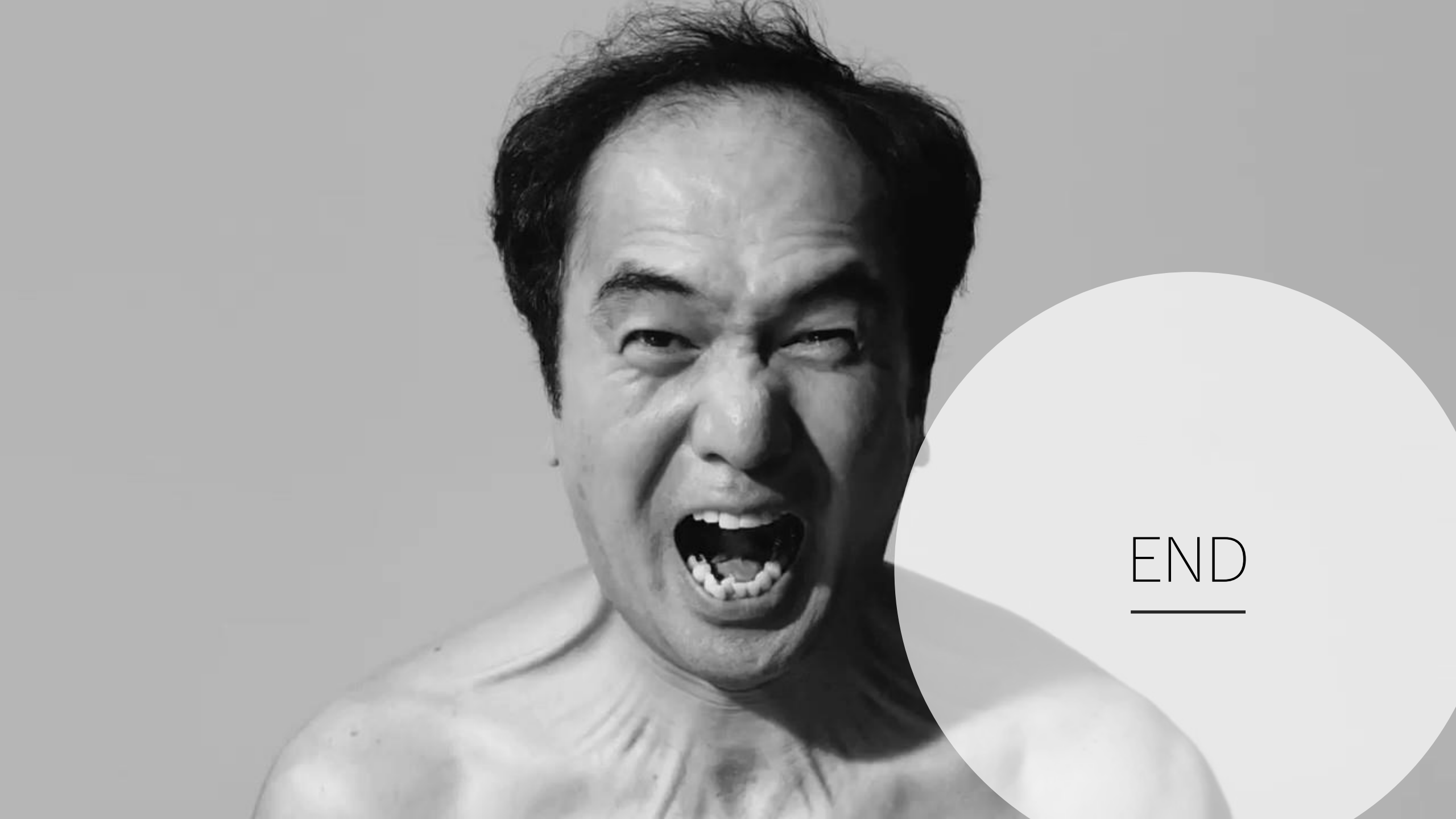
- 無理なのでやめとけ。

  他人の怠慢をすべて精算できるような神業ではない。

- 失敗が約束されたプロジェクトは何をやっても失敗する。
- むしろ、転職のためのスキルアップに活用して欲しい。

総括

- これから画期的なGUIのソフトがリリースされることは期待できない。
- やりたいことはCLIのツールで実現していこう。
- 理屈を考えるのもいいが実行に移してみよう。
- SESの怠惰さに流されるとエンジニアとして終わる。



END