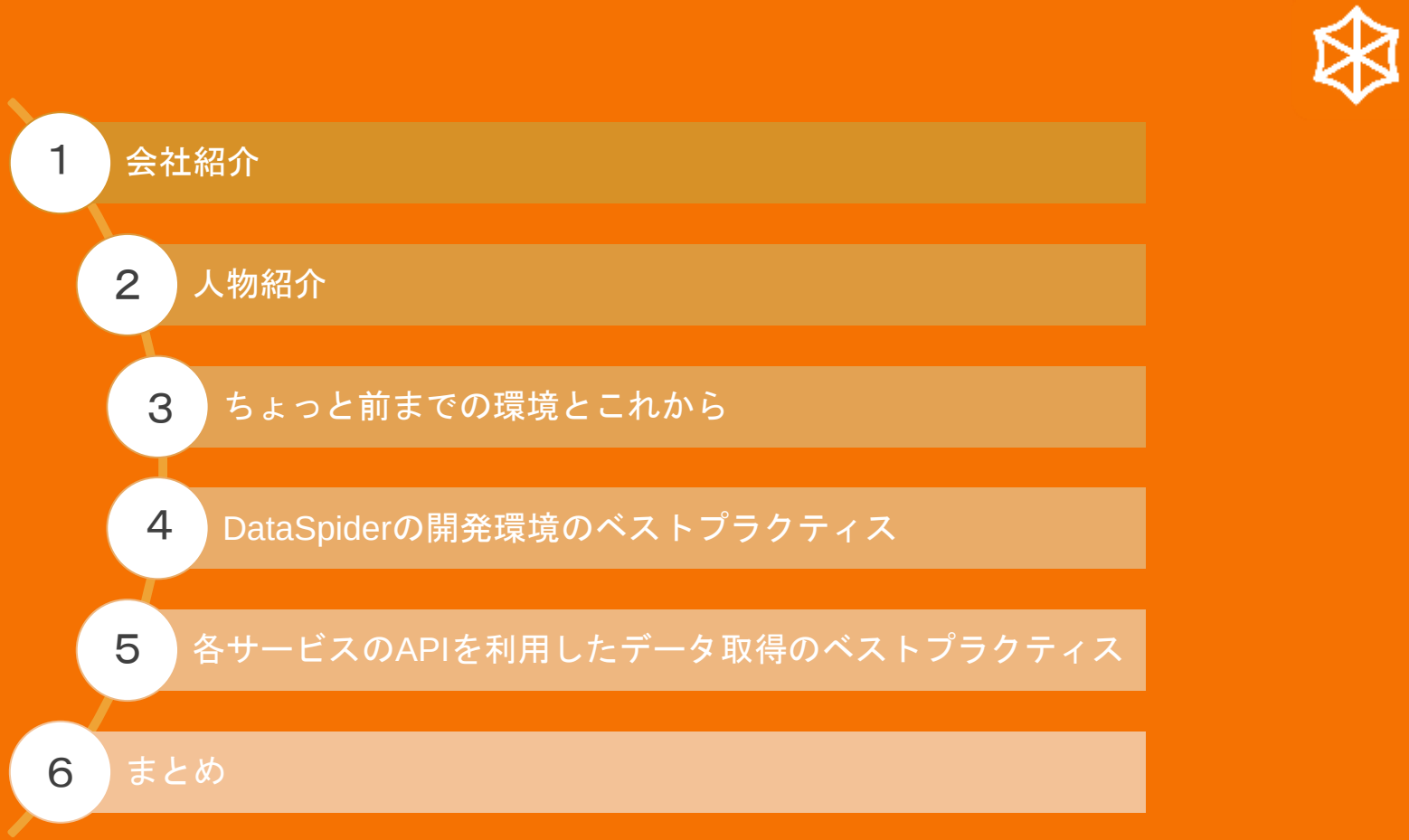
お客様の笑顔のために

データ分析基盤を支えるDataSpiderのノウハウを公開します

フリーランスBIエンジニア
りょうさん

ウイングアーク１ｓｔ株式会社
Customer Success部
宇根 昌和

AGENDA


1
会社紹介
2
人物紹介
3
ちょっと前までの環境とこれから
4
DataSpiderの開発環境のベストプラクティス
5
各サービスのAPIを利用したデータ取得のベストプラクティス
6
まとめ

会社紹介

1. 会社紹介

会社概要

商号	ウイングアーク1st株式会社（英文表記：WingArc1st Inc.）
所在地	〒105-0032 東京都港区六本木三丁目2番1号 六本木グランドタワー
創業	2004年3月
資本金	10億8,400万円（2022年2月現在）
代表者	代表取締役 社長執行役員CEO 田中 潤
決算期	2月
売上高	198億（2022年2月末）
従業員数	連結717人／単体623人（2022年2月末現在）

1. 会社紹介

ビジネスドキュメント事業

SVF
総合帳票基盤ソリューション

ノンストップ帳票運用

※オンプレミス・クラウド選択可

invoiceAgent
電子帳票プラットフォーム

※クラウド限定

電子取引/電子契約

文書管理

AI OCR

商取引の文書の電子化・効率化

※オンプレミス・クラウド選択可

データエンパワーメント事業

Dr. Sum
データ分析基盤

分析の思考を止めない高速集計

※オンプレミス・クラウド選択可

MotionBoard
BIダッシュボード

見える化からアクションへ

※オンプレミス・クラウド選択可

Dataring
データプレパレーションを統合した
データ分析基盤

dejiren
クラウドサービスを自由に繋ぐチャット型iPaaS（バーチャルアシスタント）

人物紹介

宇根 昌和

宇根 昌和

勉強中…

django

MathWorks

python

JavaScript

TensorFlow

JAVA

年	略歴
~2019	高専卒
~2021	大学院卒
2021~	ウイングアーク1st入社（2年目） プリセールス

DataSpider Servista

MOTIONBOARD

Dr. Sum

りょうさん @ryosanbimania

年代	企業	職種
2008-2016	ITベンダー	システムエンジニア・セールスエンジニア
2017-2021	ITメーカー	セールスエンジニア・カスタマーサクセス
2021-	フリーランス	BIエンジニア

セミナー → 商談支援 → 要件定義 → 設計 → 製造 → 保守

適用ツールの実績 10種類 28案件

ETLツール

DataSpider Servista

Asteria warp

talend

BIツール

MOTIONBOARD

Dr. Sum

Qlik View

WebFOCUS

Interstage

GLOVIA SUMMIT MI

Oh-Pa 1/3

サポート業種

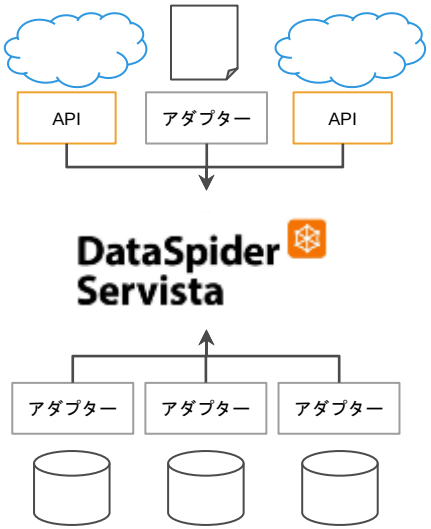
- 家具販売業
- オフィス家具販売業
- 海運事業
- 食品調味料製造業
- 輸入食品小売販売業
- 化粧品製造業
- アパレルメーカー
- 自動車部品卸業

etc.

ちょっと前までの
環境とこれから

3．ちょっと前までの環境とこれから

データ分析基盤の概要



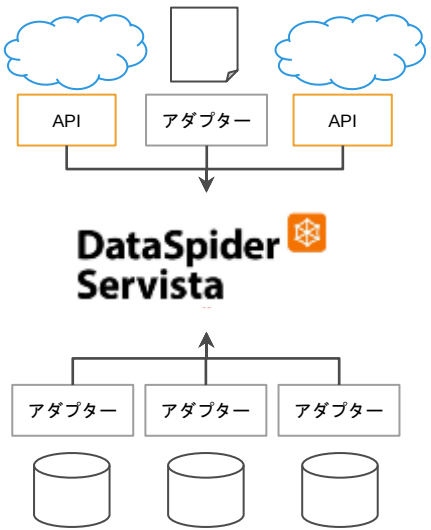
データ収集

データ統合

データ可視化

3．ちょっと前までの環境とこれから

データ分析基盤の課題

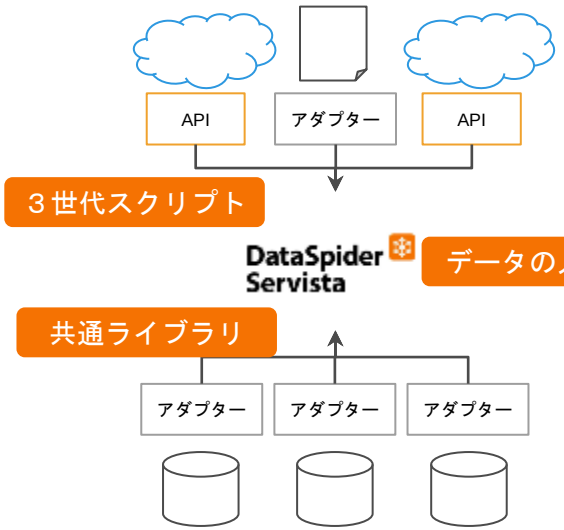


1. スクリプトが多く複雑で、引き継ぎが困難な状況になっていた
2. データ収集のノウハウがなかったため、開発コストがかかっていた
3. エラーが発生した時の調査や修正などの運用コストが多かった

引き継いだ人が笑顔になれない環境となっていた

3．ちょっと前までの環境とこれから

データ分析基盤に統一ルールをつくった



ログのDB書き込みによるデータ連携イベントの可視化

持てるツールをすべて活用した、運用コスト・開発コストを抑えた統一ルールを確立した

DataSpiderの開発環境のベストプラクティス

４．DataSpiderの開発環境のベストプラクティス

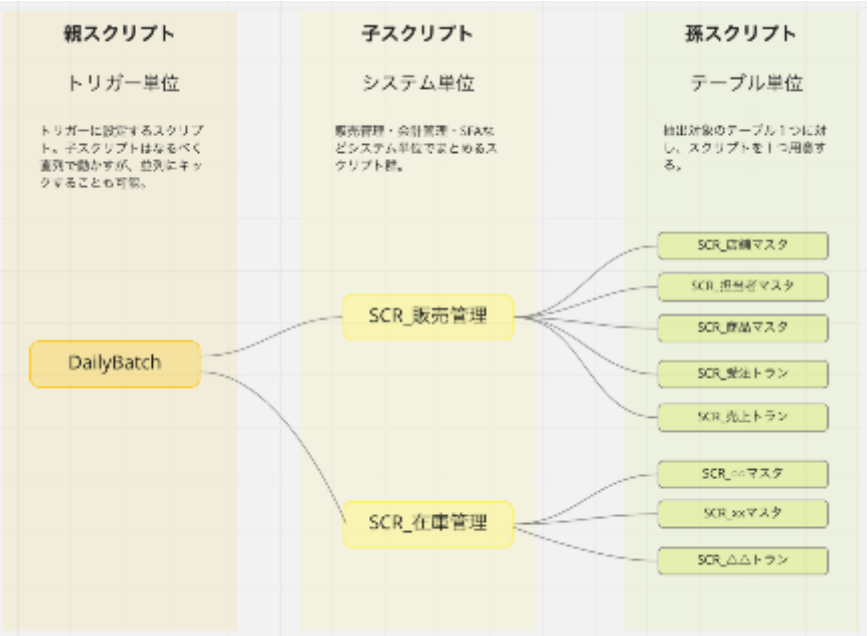
開発コスト・運用コストを抑えたベストプラクティス

- ３世代スクリプト
- データの入れ替えルール
- 共通ライブラリ
- ログのDB書き込みによるデータ連携イベントの可視化

本日はすべてを説明する時間がございませんので、詳細はこちらをご覧ください。
【DataSpider】開発コスト・運用コストを抑えたスクリプトの構築方法

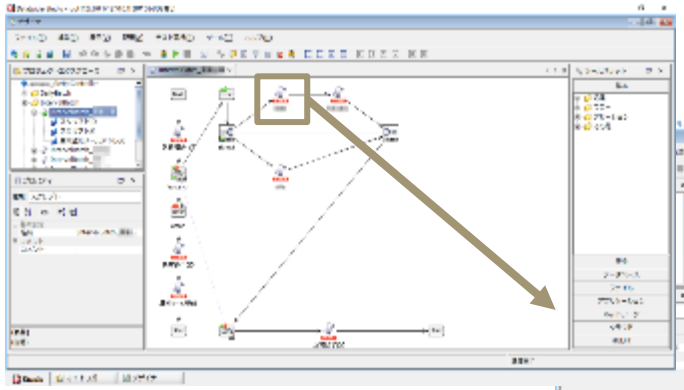
４．DataSpiderの開発環境のベストプラクティス

３世代スクリプト

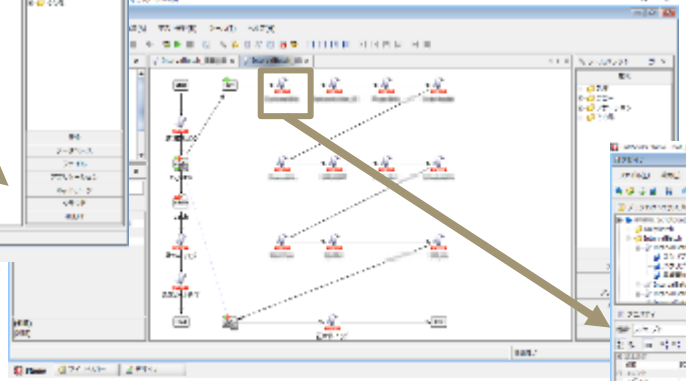


４．DataSpiderの開発環境のベストプラクティス

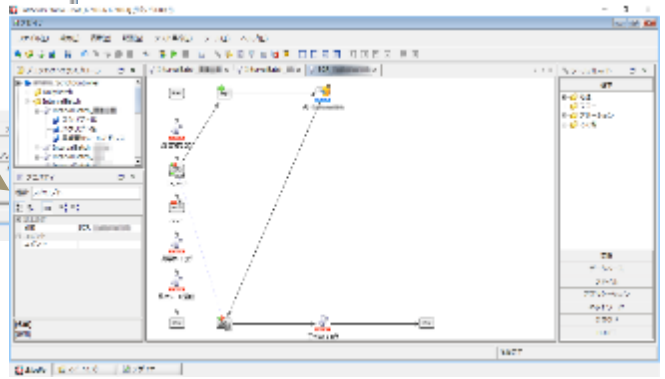
３世代スクリプト



親スクリプト



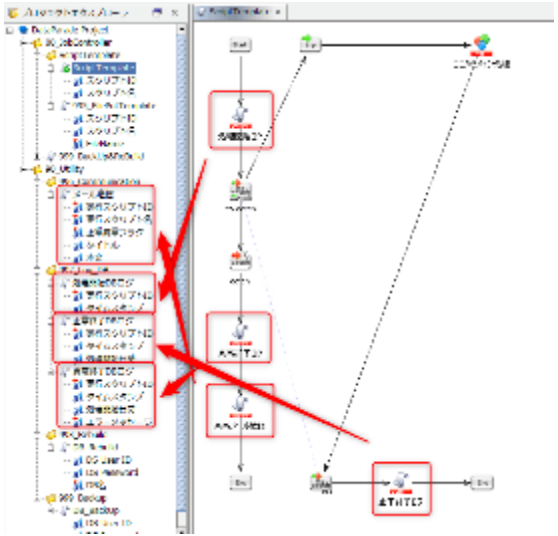
子スクリプト



孫スクリプト

４．DataSpiderの開発環境のベストプラクティス

共通ライブラリ



各ライブラリの詳細はこちら。

【DataSpider】使い勝手の良いライブラリたちを公開します

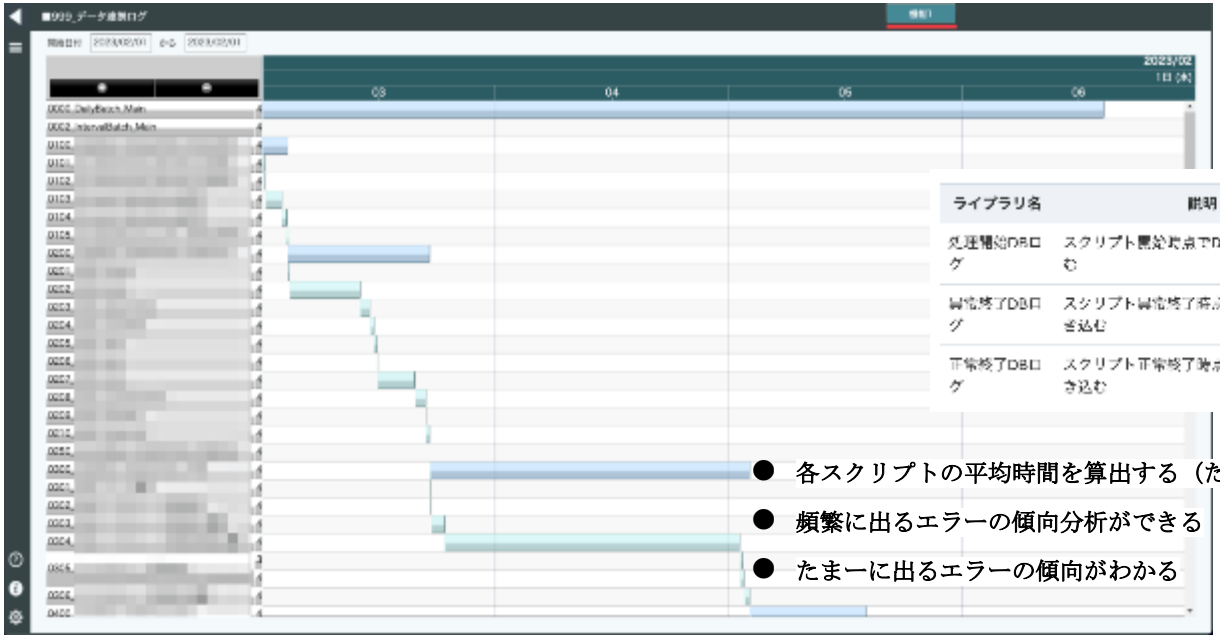
４．DataSpiderの開発環境のベストプラクティス

データの入れ替えルール

１日分のデータの入れ替え（or 挿入のみ）をすると、
エラー発生時のハンドリングがめんどくさい。
いっそのこと７日分のデータの入れ替えにしておくべき。
“７日”を任意の数値に変更できるようにすれば尚良い。

4．DataSpiderの開発環境のベストプラクティス

ログのDB書き込みによるデータ連携イベントの可視化



ライブラリ名	説明	参照記事URL
処理開始DBログ	スクリプト開始箇所でDBに時刻を書き込む	【DataSpider】処理開始DBにグライブラリ
異常終了DBログ	スクリプト異常終了箇所でDBに時刻を書き込む	【DataSpider】異常終了DBにグライブラリ
正常終了DBログ	スクリプト正常終了箇所でDBに時刻を書き込む	【DataSpider】正常終了DBにグライブラリ

- 各スクリプトの平均時間を算出する（たまに特定の曜日だけ処理が遅くなるときがあったりする）
- 頻繁に出るエラーの傾向分析ができる
- たまーに出るエラーの傾向がわかる

４．DataSpiderの開発環境のベストプラクティス

開発コスト・運用コストを抑えたベストプラクティス

- ３世代スクリプト
- データの入れ替えルール
- 共通ライブラリ
- ログのDB書き込みによるデータ連携イベントの可視化

笑顔で引き継ぎができ、運用にコストがかからない構築をする

次の章でお伝えしたいこと

企業のデータ分析は自社の基幹システムだけでなく、
導入しているSaaSからのデータ取得も重要視されてくる

各種サービスからのデータ取得はAPIが主流だが、
データ分析ができるデータに加工するにはPythonが必要

開発コストと運用コストを意識して、
DataSpiderとPythonの連携は最適解のひとつである

各サービスのAPIを利用したデータ取得のベストプラクティス

API とは？

API　外部プログラムやアプリがサービスとデータをやり取りするための仕組み

Google Analytics　twitter　BITPOINT　Coincheck　BitFlyer

```
import ccxt

coincheck = ccxt.coincheck([
        "apikey": apikey,
        "secret": secret
        ])
```

Web API　主にHTTP・HTTPSでデータをやり取りするための仕組み

REST API　RESTと呼ばれる設計原則に基づいたWeb API

```
[HTTP Method] [Resource URI]
    ★Parameter1
    ★Parameter2
    ★Parameter3...
```

```
■ Qiita Web APIの例
GET /api/v2/oauth/team_authorize
    ★client_id
    ★scope
    ★state
```

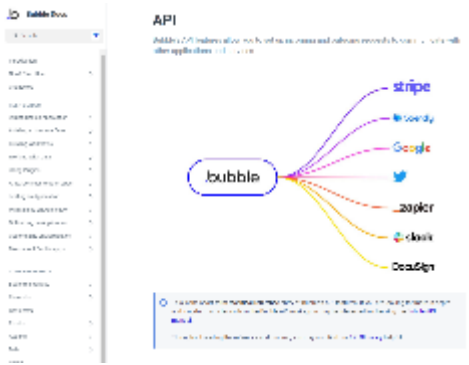
Web API 利用におけるムズカシさ

- Web API ごとのレスポンス形式の違い
- 段階的なデータ取得
- ページネーション
- データ項目の消失

Web API サンプルコンテンツ

- Qiita
  - エンジニアに関する知識を記録・共有できるサービス（Wikipediaより）
  - 記事情報を取得、コメントやいいねをつけるなど様々なAPIを用意
- Bubble
  - ノーコードでWebアプリの開発ができるサービス

https://qiita.com/api/v2/docs



https://manual.bubble.io/core-resources/api/the-bubble-api

レスポンス形式の違い

Qiita APIの例

ユーザーの所属チームを取得するAPI

出力結果：


```
[
  {
    "active": true,
    "id": "qiita-inc",
    "name": "Qiita Inc."
  }
]
```


記事のコメント一覧を取得するAPI

出力結果：


```
[
  {
    "body": "# Example",
    "created_at": "2000-01-01T00:00:00+00:00",
    "id": "3391f50c35f953abfc4f",
    "rendered_body": "<h1>Example</h1>",
    "updated_at": "2000-01-01T00:00:00+00:00",
    "user": {
      "description": "Hello, world.",
      "facebook_id": "qiita",
      "followees_count": 100,
      "followers_count": 200,
      "github_login_name": "qiitan",
      "id": "qiita",
      "items_count": 300,
      "linkedin_id": "qiita",
      "location": "Tokyo, Japan",
      "name": "Qiita キータ",
      "organization": "Qiita Inc.",
      "permanent_id": 1,
      "profile_image_url": "https://hogehoge",
      "team_only": false,
      "twitter_screen_name": "qiita",
      "website_url": "https://qiita.com"
    }
  }
]
```


ネストされていることも、、、

段階的なデータ取得

Qiita APIの例

例えば記事についているコメントを取得したい場合、2つのAPIをキックする必要有

ユーザーの記事の一覧を取得

https://qiita.com/api/v2/authenticated_user/items

```
[
  {
    <省略>
    "id": "c686397e4a0f4f11683d",
    "title": "Example title",
    "updated_at": "2000-01-01T00:00:00+00:00",
    "url": "https://qiita.com/Qiita/items/0f4f11683dc686397e4a"
    <省略>
  },
  {
    <省略>
    "id": "a0f4f11684fc686397e4",
    "title": "Example title",
    "updated_at": "2000-01-01T00:00:00+00:00",
    "url": "https://qiita.com/Qiita/items/a0f4f11684fc686397e4"
    <省略>
  }
]
```

記事IDをもとに記事のコメントを取得

https://qiita.com/api/v2/items/c686397e4a0f4f11683d/comments

```
{
    “body”: “とても良い記事です。助かりました！”,
    "created_at": "2000-01-01T00:00:00+00:00",
    “id”: "3391f50c35f953abfc4f",
    “rendered_body”: "<h1>とても良い記事です。助かりました！</h1>",
    "updated_at": "2000-01-01T00:00:00+00:00“
    <省略>
}
```

start　DataSpiderだけで実装するとしたら・・・　end

①POSTのアダプタを使ってID取得
②JSONの明細形式に変換（難しそう）
③IDの数だけループ
・IDからコメントを取得する
・JSONからコメントを抽出する。

ページネーション

Qiita APIの例

例えば記事の一覧を取得するAPIは一度に全記事を取ってきてくれない。
APIパラメータにページを指定して、繰り返しキックする必要がある。

GET /api/v2/authenticated_user/items

認証中のユーザーの記事の一覧を作成日時の降順で返します。

- page
  - ページ番号 (1から100まで)
  - Example: `1`
  - Type: string
  - Pattern: `/^[0-9]+$/`
- per_page
  - 1ページあたりに含まれる要素数 (1から100まで)
  - Example: `20`
  - Type: string
  - Pattern: `/^[0-9]+$/`

300記事分のデータ取得する場合、こんなロジックが必要

https://qiita.com/api/v2/authenticated_user/items?page=1&per_page=100

↓

https://qiita.com/api/v2/authenticated_user/items?page=2&per_page=100

↓

https://qiita.com/api/v2/authenticated_user/items?page=3&per_page=100

↓

https://qiita.com/api/v2/authenticated_user/items?page=4&per_page=100

↓

記事数が0件なので終了

データ項目の消失

Bubble APIの例

Bubble のデータ例

floor_number	food	flag_indivisual_account
2	1610606550463x862956	true
2		true
2	1610606550463x862956	true

一般的なWeb APIのレスポンス

```
{
    "floor_number": 2,
    "food": "1610606550463x862956",
    "flag_indivisual_account": true
}
{
    "floor_number": 2,
    "food": "",
    "flag_indivisual_account": true
}
{
    "floor_number": 2,
    "food": "1610606550463x862956",
    "flag_indivisual_account": true
}
```

データ項目の消失

Bubble APIの例

Bubble のデータ例

floor_number	food	flag_indivisual_account
2	1610606550463x862956	true
2		true
2	1610606550463x862956	true

Bubble Web APIのレスポンス

```
{
    "floor_number": 2,
    "food": "1610606550463x862956841778950000",
    "flag_indivisual_account": true
}
{
    "floor_number": 2,
    "flag_indivisual_account": true
}
{
    "floor_number": 2,
    "food": "1610606550463x862956841778950000",
    "flag_indivisual_account": true
}
```

foodがない！

Web API 利用におけるムズカシさ

- Web API ごとのレスポンス形式の違い
- 段階的なデータ取得
- ページネーション
- データ項目の消失

DataSpider の REST アダプタを使っても難しいケースもあり
上記4点を考慮して構築するのは**難易度も運用コストも高くなってしまう**

DataSpider と Python のハイブリッド構築！！

餅は餅屋

DataSpider Servista

- GUIベースで簡単に構築が可能
- スクリプト間の連携のしやすさ
- エラーハンドリングのしやすさ

Python

- 非常に柔軟なデータ処理
- 直感的に扱いやすいデータの持ち方

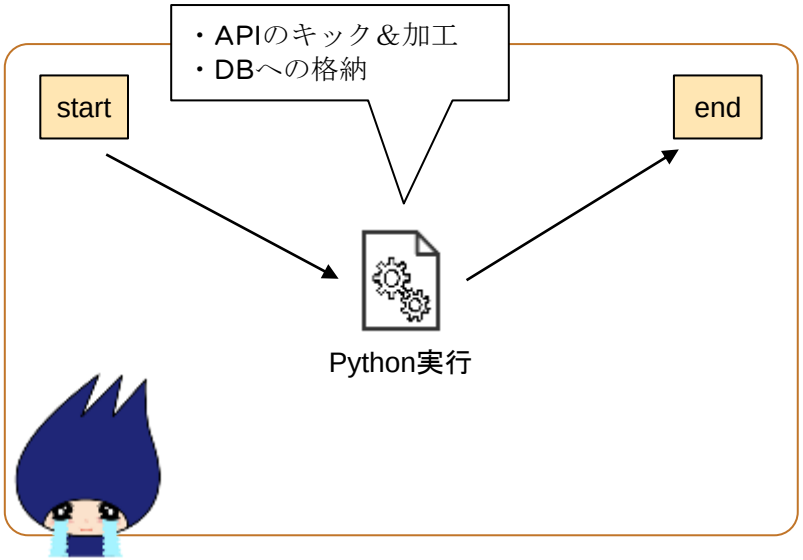
DataSpiderのだれでも簡単に構築できる点は強力！
Web APIの性質上加工しづらい部分をPythonが吸収！

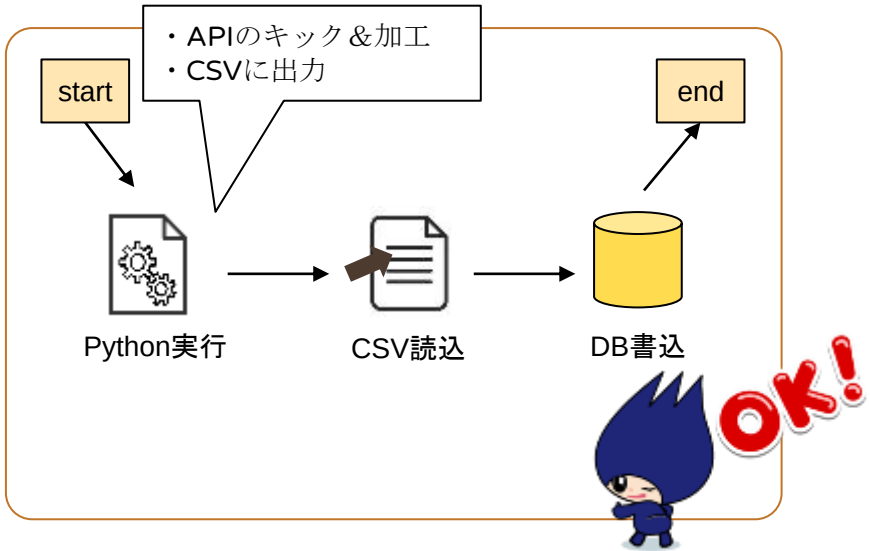
Pythonとのハイブリッド構築において大切なこと

Pythonが担う役割は極力小さく！

DataSpiderとのI/F 部分を汎用的に！

Pythonが担う役割は極力小さく！





スクリプトの使い回しが可能
実装も最小限でメンテナンスリスクも少ない

Pythonとのハイブリッド構築において大切なこと

Pythonが担う役割は極力小さく！

DataSpiderとのI/F 部分を汎用的に！

DataSpiderとのI/F部分を汎用的に！

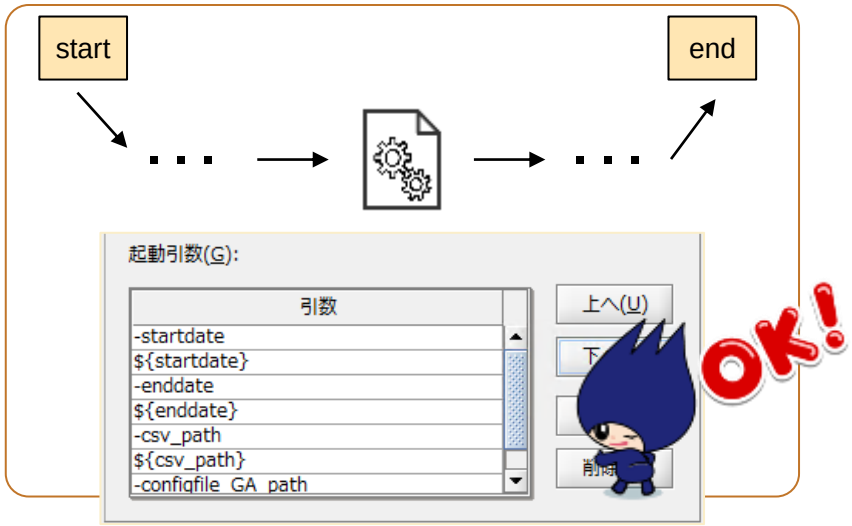
データ取得期間を指定する場合

Pythonスクリプト

```
from_date = today(-7) # 7日前の日付を取得
to_date = today(-1) # 昨日の日付を取得
request_api(from_date, to_date) # 指定期間のデータを取得
```

いわゆるハードコーディング

DataSpider

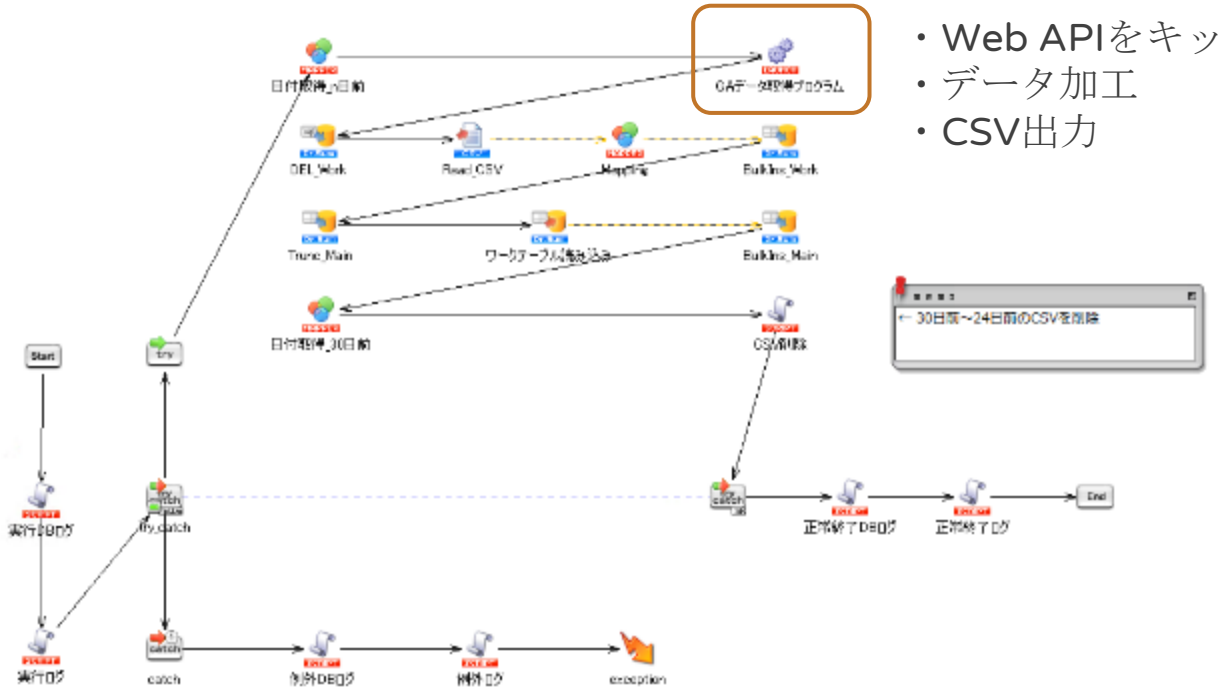


実装スクリプト

スクリプトで行っていること

1. サービスからWeb APIをキックしてデータを取得・加工
2. DBへ格納（指定期間のデータを洗い替え）
3. その他のエラーハンドリング・ログ書き込み

API取得&DB格納するスクリプト



- Web APIをキック
- データ加工
- CSV出力

DataSpiderの役割・・・CSVの読み込み、DBの洗い替え、エラーハンドリング

Python のI/F部分

起動コマンド(C):

/E:/Projects/API/GoogleAnalytics/get_GA_data_main.exe　参照(B)...

起動引数(G):

-configfile_GA_path
${configfile_GA_path}
-startdate
${startdate}
-enddate
${enddate}
-csv_path
${csv_path}

上へ(U)　下へ(D)　追加(A)　削除(R)

認証キーのファイルパス

データを取得する期間

CSVを出力するフォルダパス

【Appendix】Pythonの実装例

```
parser = argparse.ArgumentParser(description='This script gets GA data. ')
parser.add_argument('-configfile_GA_path', help = "Set path including config file name e.g. C:/Users/hogehoge/config.ini ", required = Tr
parser.add_argument('-startdate', help = "Set startdate to insert into database as hyphen dilimiter yyyy-mm-dd e.g. 2021-01-01 ", require
parser.add_argument('-enddate', help = "Set enddate to insert into database as hyphen dilimiter yyyy-mm-dd e.g. 2021-01-07 ", required =
parser.add_argument('-csv_path', help = "Set path to save csv path e.g. C:/Users/hogehoge", required = True)
parser.add_argument('-csv_prefix_name', help = "Set prefix csv name as [prefix name]yyyy-mm-dd~yyyy-mm-dd[suffix name]. ", default = "",
parser.add_argument('-csv_suffix_name', help = "Set suffix csv name as [prefix name]yyyy-mm-dd~yyyy-mm-dd[suffix name]. ", default = "",
args = parser.parse_args()
```

```
headers = {
    'Authorization': 'Bearer ' + token
}
for i in range(9):
  params = {
      'page': str(i + 1),
      'per_page': '100',
  }
  jsondata = con_qiita.get_QiitaData(headers, params)

  for item in jsondata:
    try:
      pagejson = con_qiita.get_pagestatus(headers, item)
    except Exception as e:
      exit(e)

    data_store(pagejson)

  if jsondata[0]['user']['items_count']<100*(i+1):
    break
```

記事一覧の取得

記事の数分繰り返し

次ページがなければ終了

まとめ

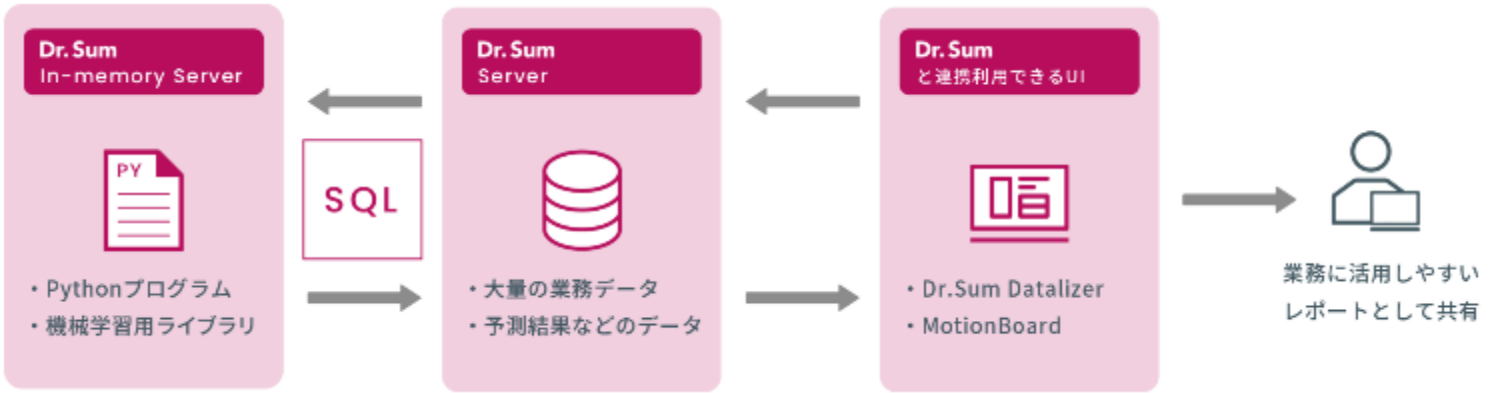
０６．まとめ

企業のデータ分析は自社の基幹システムだけでなく、
導入しているSaaSからのデータ取得も重要視されてくる

各種サービスからのデータ取得はAPIが主流だが、
データ分析ができるデータに加工するにはPythonが必要

開発コストと運用コストを意識して、
DataSpiderとPythonの連携は最適解のひとつである

将来的に・・・



Dr.SumのPython連携機能を使った機械学習
データの蓄積・可視化の次のステップへ

おわり