帳票 API を利用した
カスタマイズ帳票作成

2018 年 4 月 25 日

日本医師会 ORCA 管理機構

内容

0. 改定履歴 1
1. 概要 2
2. 対象 2
3. 帳票 API を利用したカスタマイズ帳票作成の動作概要 2
4. push-exchanger 及び print001 プラグインの動作概要 3
5. push-exchanger のインストール及び設定 4
インストール 4
puth-exchanger の設定 4
設定事前準備 4
config.yml の編集 5
pusher との接続確認 5
6. print001 プラグインの設定 8
7. 外来カルテ 1 号用紙の出力 8
karte_no1m プラグインの設置 9
日レセプログラムオプションの設定 9
push-exchanger の起動 9
日レセによるカルテ発行操作 9
8. karte_no1m プラグインの解説 10
9. その他のサンプル帳票プラグイン 12
10. push-exchanger を利用しないカスタマイズ帳票の作成 13
11. まとめ 13
12. 参考文献 13

0. 改定履歴

初版・・・2017 年 3 月 30 日

第 2 版・・2017 年 11 月 24 日
　インストーラを差し替え

第 3 版・・2017 年 12 月 15 日
　インストーラを差し替え(32bit 対応)

第 4 版・・2018 年 4 月 25 日
　インストーラを差し替え
　・SSL 接続時のサーバ認証においてホスト名の検証を行わない不具合を修正しました。
　・サーバ印刷プラグイン(server_print.rb)による Windows 環境の印刷時にエラーが
　　発生する不具合を修正しました。

1. 概要

本文書ではORCAクラウド環境において帳票APIを利用した医療機関側でのカスタマイズ帳票作成について説明する。
前半は帳票 API を利用したカスタマイズ帳票作成の概要について説明する。後半は日レセPUSH通知駆動フレームワーク push-exchanger、帳票印刷プラグイン print001 を利用したカスタマイズ帳票の出力について解説を行う。

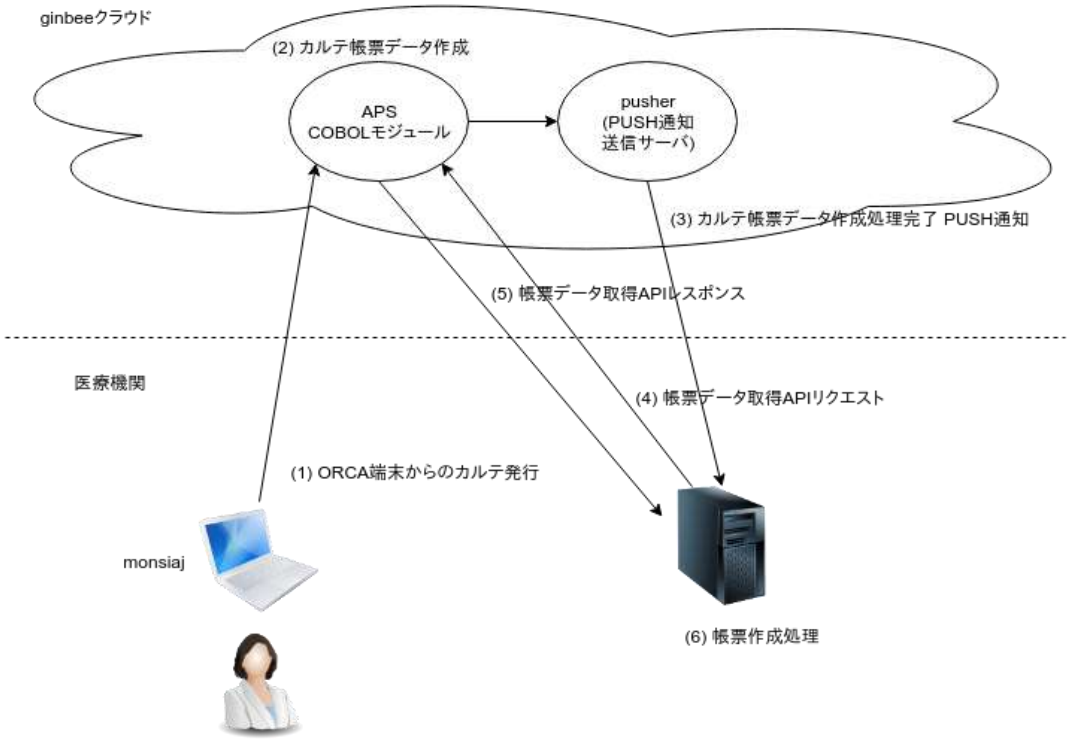
2. 対象

本文書の対象とする環境は以下とする。

- サーバ環境
  - ORCA クラウド
  - 日レセ 5.0
- クライアント環境
  - Windows OS 32bit/64bit 版
    ※バージョン 0.0.24 より 32bit にも対応

3. 帳票 API を利用したカスタマイズ帳票作成の動作概要

帳票 API によるカスタマイズ帳票作成の処理の流れについて説明する。
以下のような流れになる。



1. 医療機関の日レセ端末より窓口帳票の印刷指示を行う
   - 事前に日レセ川で帳票 API 利用の設定が必要
2. ORCA クラウドの日レセ APS で帳票データを作成する
3. ORCA クラウドの PUSH 通知サーバ(pusher)より、帳票データ作成完了の PUSH 通知を送信する
4. 医療機関側で PUSH 通知を受信し、PUSH 通知に含まれる帳票 ID を元に帳票データ取得 API へリクエストを行い、帳票データを取得する
5. 医療機関側で帳票データを使用し、帳票の編集、印刷を行う

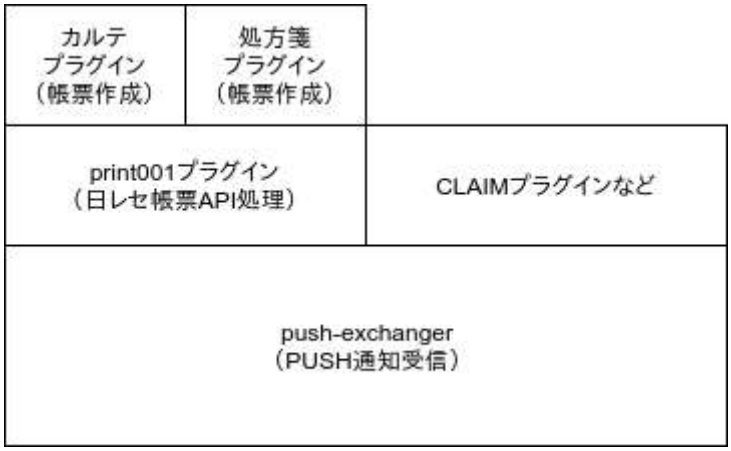
4. push-exchanger 及び print001 プラグインの動作概要

帳票 API によるカスタマイズ帳票作成の一例として日レセ PUSH 通知駆動フレームワーク push-exchanger とそのプラグインである print001 プラグインを紹介する。

push-exchanger は日レセ PUSH 通知クライアントであり、PUSH 通知を受信して PUSH 通知の種別毎に対応するプラグインを起動する。

print001 プラグインは帳票 API の PUSH 通知に対応するプラグインであり、起動すると帳票データ取得 API へリクエストを行い、帳票データを取得する。

帳票データを取得すると個々の帳票プラグインを呼び出し、帳票編集後、印刷処理を行う。



push-exchanger 及び print001 プラグインの詳細については push-exchanger の仕様書を参照。

5. push-exchanger のインストール及び設定

Windows 環境への push-exchanger のインストールと設定について説明する。

インストール

以下のリンクからインストーラをダウンロードし実行する。
https://ftp.orca.med.or.jp/pub/data/receipt/download/windows/ginbee/push-exchanger-installer-0.0.25.exe

完了すると c:¥program files¥push-exchanger に push-exchanger 及び Ruby 実行環境、サンプルプラグイン一式がインストールされている。

puth-exchanger の設定

設定は YAML 形式の設定ファイル c:¥program files¥push-exchanger¥config.yml を編集することで行う。

設定事前準備

ORCA クラウド環境の pusher に接続するため以下が必要になる。事前に ORCA クラウド環境管理画面にログインして取得しておく。

- 日レセ API キー
- 日レセ API 用 SSL クライアント証明書
  - CA 証明書(ca.crt)
  - クライアント証明書(*.crt)
  - クライアント証明書秘密鍵(*.pem)
  - 秘密鍵パスフレーズ

config.yml の編集

config.yml の以下の項目を編集する。

項目名	内容	説明
:ws_server	wss://pusher-proxy.orca.orcamo.jp/ws	pusher の URI
:api_key	日レセ API キー	
:api_server	ap-proxy.orca.orcamo.jp	API サーバ
:api_port	8080	API サーバポート
:use_ssl	true	
:ca_cert	"c:/program files/push-exchanger/cert/ca.crt"	CA 証明書
:cert	"c:/program files/push-exchanger/cert/tenant.crt"	証明書
:cert_key	"c:/program files/push-exchanger/cert/tenant.pem"	証明書秘密鍵
:passphrase	パスフレーズ	

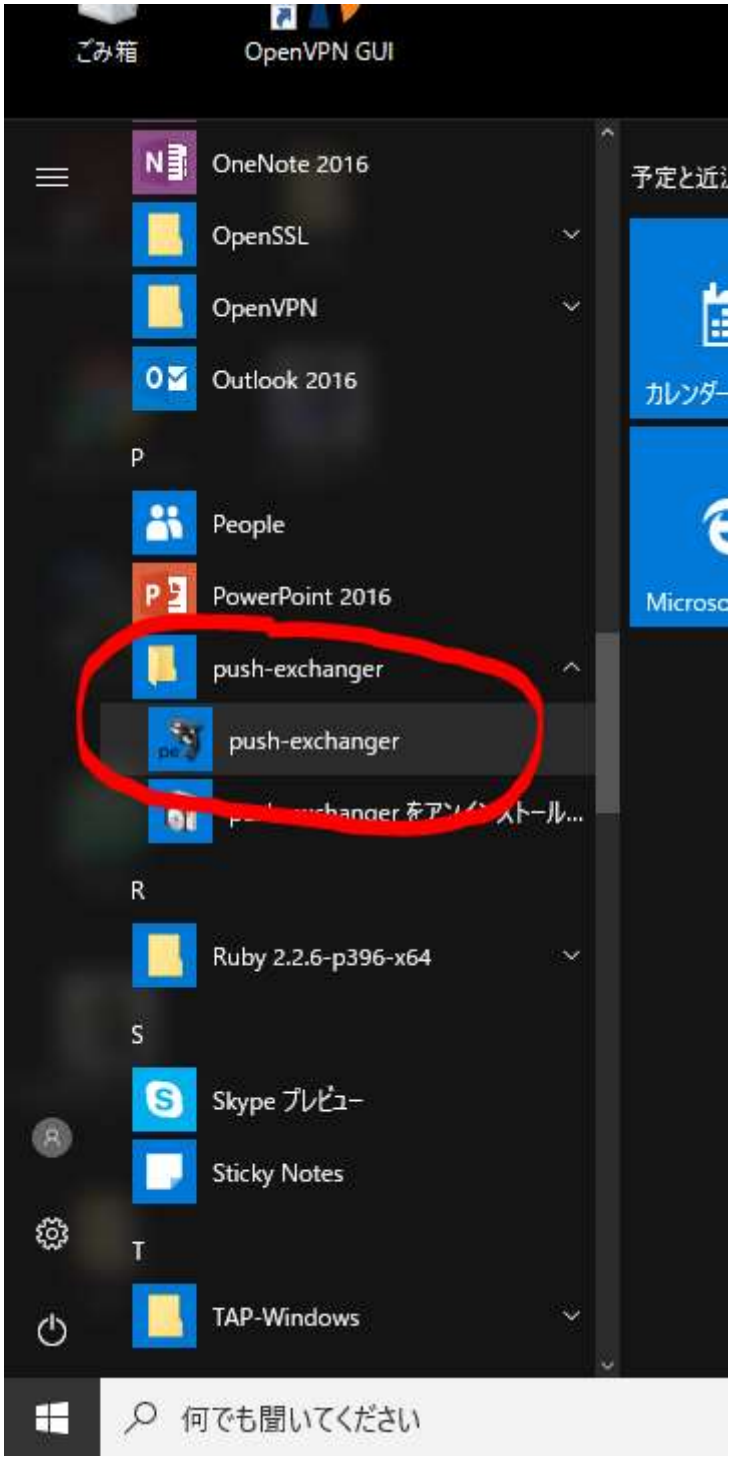
:ca_cert、:cert、:cert_key はダブルクォートで括った方がトラブルになりにくい。またパス区切り文字は「¥」ではなく「/」となっていることに注意が必要である。

設定編集後、所定のディレクトリに CA 証明書、クライアント証明書を配置する。

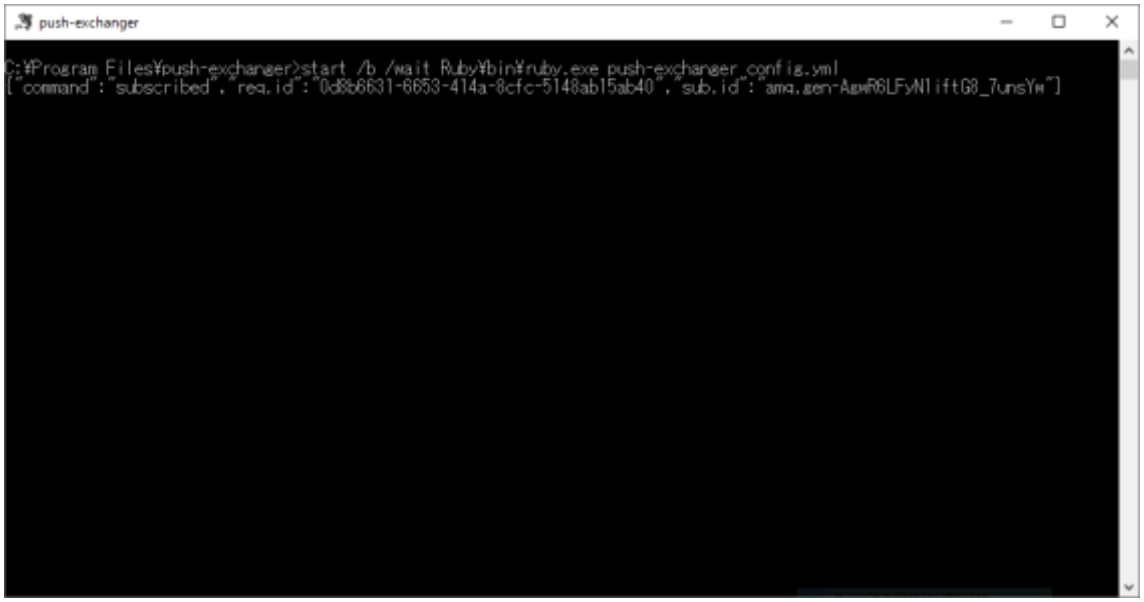
pusher との接続確認

push-exchanger を起動し、pusher との接続を確認する。

スタートメニューから push-exchanger フォルダを選択し push-exchanger をクリックして起動する。



以下のようなコマンドプロンプトが表示される。



push-exchanger はコマンドプロンプト中で動作し、pusher から受信したメッセージやエラーメッセージをコマンドプロンプト中に表示する。上記のような subscribed となっている JSON 文字列が表示されると接続に成功している。

上記のように Ruby のエラーメッセージが表示される場合は接続に失敗している。

C:¥program files¥push-exchanger¥log¥push-exchanger.log.txt に作成されるログを確認した上で、以下の確認を行う。

- pusher-proxy.orca.orcamo.jp に ping が到達可能か
  - VPN が正しく設定されているか
- 設定ファイルの確認
  - 不正な YAML 形式となっていないか

- API キーが正しいか
- CA 証明書、クライアント証明書の記載が正しいか
- CA 証明書、クライアント証明書を正しく設置しているか
- 正しいパスフレーズを設定しているか

これらを確認した上で接続できない場合はサポートセンターに連絡する。

6. print001 プラグインの設定

print001 プラグインの設定は
c:¥program files¥push-exchanger¥plugin¥print001¥print001.yml を編集することで行う。
ここでは以下のプリンタ及び印刷の設定のみを行う。

項目名	説明
:printer_settings:	プリンタ設定ハッシュ
:default:	デフォルトのプリンタ指定
:print_command:	印刷コマンド

:printer_settings:にて帳票 ID と出力するプリンタを指定する。帳票 ID が:default:の場合はデフォルトプリンタとなる。ここではデフォルトプリンタの設定のみとする。

プリンタ名には Windows に登録されているプリンタを設定する。
プリンタ名に空白が含まれる場合は以下のようにダブルクォートで括る。

```
:printer_settings:
  :default:  “RICOH IPSiO NX85S RPDL”
```

:print_command:に実際に印刷するコマンドを記載する。%FILE%、%PRINTER%はコマンド実行時にそれぞれ PDF ファイル名、プリンタ名に置き換えられる。コマンドやプリンタ名に空白が含む場合は全体をシングルクォートで括った上でそれぞれダブルクォートで括る必要がある。

```
:print_command:  '"C:¥Program Files (x86)¥Adobe¥Acrobat Reader DC¥Reader¥AcroRd32.exe" /N /T "%FILE%" "%PRINTER%"'
```

上記の例ではアクロバットリーダを起動して印刷を行う設定となっている。

その他の設定については push-exchanger の仕様書を参照。

7. 外来カルテ 1 号用紙の出力

サンプルとして含まれている外来カルテ 1 号用紙(karte_no1m プラグイン)のカスタマイズ

帳票を出力する。

karte_no1m プラグインの設置

karte_no1m プラグインを print001.rb から動作可能にするために C:¥Program Files¥push-exchanger¥plugin¥print001¥plugin¥sample にある karte_no1m ディレクトリを C:¥Program Files¥push-exchanger¥plugin¥print001¥plugin¥にコピーする。

日レセプログラムオプションの設定

外来カルテ 1 号用紙の帳票 API を有効にするため、日レセでプログラムオプションの設定を行う。

1. システム管理より「1910 プログラムオプション情報」を選択し、プログラムオプション設定画面を表示する
2. プログラム名一覧から ORCHC01 を選択する
3. オプション内容の入力欄に以下の 2 行を追加する

```
API_DATA=1
API_CUSTOM_ID=karte_no1m
```

push-exchanger の起動

スタートメニューから push-exchanger を起動し、pusher と接続する。

日レセによるカルテ発行操作

日レセでカルテ発行操作を行う。

1. 患者登録画面を開く
2. 患者 ID を入力してカルテ発行を行う患者を開く
3. 生年月日の右のコンボボックスから「1 カルテ発行あり」を選択する
4. 登録ボタン押下

プログラムオプションの設定が正しくされていれば push-exchanger のコマンドプロトに以下のような帳票 API の PUSH 通知が表示されます。表示されない場合はプログラムオプションの設定を確認する。

```
C:¥Program Files¥push-exchanger>start /b /wait Ruby¥bin¥ruby.exe push-exchanger config.yml
{"command":"subscribed","req.id":"d977c8cf-1eac-4c9f-977e-3778eaae4f7a","sub.id":"amq.gen-btIEDmYr_DenTgJKjOvEKw"}
{"command":"event","sub.id":"amq.gen-btIEDmYr_DenTgJKjOvEKw","data":{"body":[{"Custom_ID":"karte_no1m","Data_ID":"ONLINE#fe28c157-a429-4e87-bc36-e64f4dabad24#20170328102248#0001#1001","Form_ID":"karte_no1","Form_Name":"カルテ１号紙
"},{"Custom_ID":"","Data_ID":"","Form_ID":"","Form_Name":""},{"Custom_ID":"","Data_ID":"","Form_ID":"","Form_Name":""},{"Custom_ID":"","Data_ID":"","Form_ID":"","Form_Name":""},{"Custom_ID":"","Data_ID":"","Form_ID":"","Form_Name":""},{"Custom_ID":"","Data_ID":"","Form_ID":"","Form_Name":""},{"Custom_ID":"","Data_ID":"","Form_ID":"","Form_Name":""},{"Custom_ID":"","Data_ID":"","Form_ID":"","Form_Name":""},{"Custom_ID":"","Data_ID":"","Form_ID":"","Form_Name":""}],"event":"print001","time":"2017-03-28T10:22:49+0900","user":"ormaster"}}
```

print001 プラグインの設定が正しければアドビリーダーが起動されカスタマイズカルテ 1 が印刷される。
印刷されない場合は、push-exchanger のログを確認し、print001.yml の設定を見直す。

正常動作していれば帳票 API のレスポンスデータが
c:¥program files¥push-exchanger¥tmp¥print001-data¥に作成されている。

また印刷に利用した PDF データが
c:¥program files¥push-exchanger¥tmp¥print001¥に保存されている。

8. karte_no1m プラグインの解説

karte_no1m プラグインは print001 プラグインの帳票プラグインという形で実装されている。帳票プラグインの仕様は以下である。

- プラグイン名は帳票 ID(Form_ID)またはカスタム ID(Custom_ID)と一致すること
- クラス名はプラグイン名と一致させる(karte_no1m であれば Karte_no1m となる)
- 帳票作成処理を行う export メソッドを定義すること
  - export メソッドはファイル名、帳票データ(JSON)、オプションの 3 つの引数を受

け取る
  - export メソッド内で帳票データを利用して引数のファイル名の PDF を作成する

帳票 API を利用した帳票データの取得、印刷処理は print001 プラグインで行うようになっている。

karte_no1m は monpe Ruby ライブラリを利用して monpe のテンプレート帳票ファイルを使用して PDF を作成する。

monpe Ruby ライブラリについては monpe Ruby ライブラリの仕様書を参照。

karte_no1m ディレクトリを参照すると以下のようになっている。

- HCM01.red
  - カルテ 1 号用紙頭書きの monpe 帳票ファイル
- HCM011.red
  - カルテ 1 号用紙続紙の monpe 帳票ファイル
- karte_no1m.rb
  - プラグイン本体の Ruby スクリプト

karte_no1m.rb の抜粋を以下に示す。Karte_no1m クラスに export メソッドが実装されていることがわかる。export メソッド内で帳票データを参照し、monpe Ruby ライブラリを用いてデータ埋め込みを行っている。

```
# coding : utf-8

require 'monpe'
require 'pp'
require_relative '../lib/orca-string'
require_relative '../lib/orca-wareki'

class Karte_no1m

  FIRST_PAGE_DISEASE_ROWS = 8
  MIDDLE_PAGE_DISEASE_ROWS = 22

  # call from print001.rb
```

```
  def export(fname,report_data,opt)
    reports = []
    report_data['Forms'].each do |form|
      form_data = form['data']
      if form_data['Order_Class'] == '2'
        reports << make_later_page(form_data)
        next
      end
      form_data['Disease_Information']   ||= []
      form_data['Disease_Information'].delete_if do |item|
        item['Name'].to_s.empty?
      end
      reports << make_1st_page(form_data)
      start_row = FIRST_PAGE_DISEASE_ROWS
      while form_data['Disease_Information'].size   > start_row
        reports << make_later_page(form_data,start_row)
        start_row += MIDDLE_PAGE_DISEASE_ROWS
      end
    end
    Monpe.export(reports,fname)
  end

<省略>
```

9. その他のサンプル帳票プラグイン

push-exchanger には karte_no1m 以外にもサンプル帳票プラグインが同梱されている。
c:¥program files¥push-exchanger¥plugin¥print001¥plugin¥sample に以下がある。

- karte_no1
- meisaisho
- okusuri_joho
- okusuri_techo
- seikyusho
- shohosen
- yoyakuhyo
- yoyakukanjalist

- yoyakulist

これらはいずれも帳票テンプレートに monpe ではなく ThinReports を利用しているが、ほぼ同様の処理となっている。
各プラグインの確認は帳票 API のレスポンスの仕様書を参考にすると理解しやすい。

10. push-exchanger を利用しないカスタマイズ帳票の作成

帳票 API を利用したカスタマイズ帳票の作成については現状においては push-exchanger を利用した実装例しかないが、他のプログラミング言語で独自に実装することが可能である。また帳票テンプレートとして monpe と ThinReports を挙げたがその他のライブラリやソフトを利用しても問題ない。

11. まとめ

- 帳票 API を利用して ORCA クラウド環境でもカスタマイズ帳票の作成が可能である
- push-exchanger と print001 プラグインを利用して医療機関の Windows 環境でカスタマイズ帳票の作成が可能である
  - ➢ ただし必ずしも push-exchanger を利用する必要はなく独自プログラムでも同じ仕組みが利用できる

12. 参考文献

- 日医標準レセプトソフト PUSH 通知仕様書
- 日レセ PUSH 通知駆動フレームワーク push-exchanger 仕様書
- monpe Ruby ライブラリ仕様書
- 帳票データ取得 API について
- 帳票データを作成するための日レセの設定について
- push-exchanger の設定について
- 画像データ取得 API
- オブジェクト指向スクリプト言語 Ruby https://www.ruby-lang.org/ja/
- ThinReports http://www.thinreports.org/