コードレビュー Vol.2

CODE Review

USP 研究所技術研究員
written by 大内智明

**前回から引き続き、ディスク容量監視アプリをどのように「簡単に、短期間に」作成するのか説明していきます。
今回は「保存された結果をCGIで画面表示する」プログラムコードを紹介します。
ブラウザ上に表示するCGI部分を詳しく説明します。**

技術的な概要

一般的に使用される CGI（apache）とブラウザを使用してサーバー内の結果をパソコン上に表示します。

ユニケージ開発手法には、HTML を作成するコマンド群が充実しているため、それらを使用することで「簡単に、短期間に」CGI と HTML を作成することができます。

特に画面出力用 HTML を作成する際に、ユニケージ開発手法ではよく用いられる手法があります。

数個のコマンド（calsed と mojihame）を実行して、あらかじめ用意した加工データをテンプレート HTML にはめ込み、画面出力用 HTML を作成する方法です（画面 1 と画面 2）。

紹介するセクションは、以下の 3 つに分割できます。

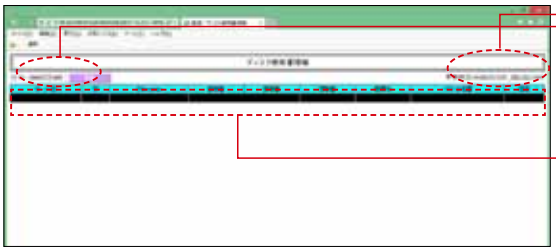
[1] 画面から POST 形式のデータを取得して、name 形式に変換する。

[2] サーバー内に保存された結果から画面出力用データを抽出する。

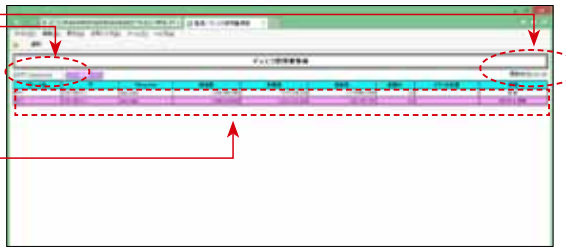
[3] 抽出したデータをテンプレート HTML にはめ込み、画面出力用 HTML を作成する。

【補足】usp Tukubai には、他にも HTML 各タグを読み取り専用にするコマンド、各種入力タグやプルダウンなどのタグに値をはめ込むコマンドが充実しています。

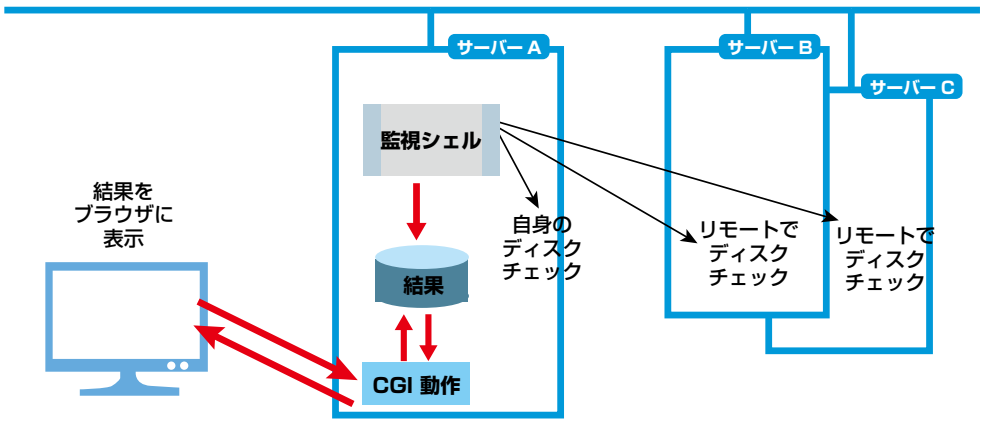
画面 1　テンプレート HTML



画面 2　画面出力用 HTML



概要図



リスト1　ディスク容量監視アプリ

```
 1 #!/bin/bash -xv
 2 #
 3 # KANSI.DISK_VOL.CGI
 4 # ディスク容量監視
 5 #
 6 # Written by J.Doi / Date: 2012/08/16
 7 #
 8 # ログの出力
 9 logfile=/home/usp/LOG/LOG.$(basename $0).$(date +%Y%m%d)_$(date +%H%M%S)_$$
10 exec 2> $logfile
11
12 # 変数の定義
13 export LANG=ja_JP.UTF-8
14 export PATH=/home/UTL:/home/TOOL:$PATH
15
16 home=/home/usp
17 logd=$home/LOG
18 semd=$home/SEMAPHORE
19 htmd=$home/KANSI/HTML
20 outd=$home/KANSI/OUTPUT
21 tmp=/tmp/$$
22 today=$(date +%Y%m%d)
23 todayhms=$(date +%Y%m%d%H%M%S)
24
25 # エラーチェックと終了処理の定義
26 ERROR_CHECK(){
27   [ $(plus ${PIPESTATUS[@]}) -eq 0 ] && return
28   touch $semd/$(basename $0).$HOSTNAME.ERROR.$today
29   exit 1
30 }
31
32 ############################################################
33 # 取り込み処理
34 ############################################################
35 # CGIデータ取込
36 dd bs=${CONTENT_LENGTH}                |
37 cgi-name -d_ -i_                       > ${tmp}-name
38
39 ##############################################################################
40 sdate=$(nameread DAY_SELECT $tmp-name)
41 [ -z ${sdate} ] && sdate=${today}
42
43 # 監視ファイル更新時刻取得
44 cat $outd/DISK_VOL.${sdate} |
45 lineup 4                    |
46 tail -n 1                  |
47 dayslash HH:MM 1            > $tmp-kousin
48
49 # 監視ファイルのデータ形成
50 # 1:サーバ名 2:ＩＰ   3:Filesystem   4:現在時刻      5:総容量
51 # 6:使用量   7:空容量 8:使用％       9:マウント位置 10:状況
52 # 11:色
53 cat $outd/DISK_VOL.${sdate} |
54 getlast 1 3                 |
55 comma 5 6 7                 |
56 msort key=9@1               > $tmp-check
57 ERROR_CHECK
```

実行結果をチェックする関数
エラーが発生すると終了

[1] 画面から POST 形式のデータを取得して、name 形式に変換する処理（36 ～ 37 行目）

cgi-name については画面 3 を参照

画面にセットされた日付データを取得している。
nameread は、name 形式のデータから値を抽出する。

日別の保存されている結果データを読み込んでいる。

lineup は指定フィールドのデータを抜き出し、ソート及びユニークする。

dayslash は日付時刻フォーマットを変換する。

```
$ echo 20140101090000 | dayslash
 HH:MM 1
09:00
```

[2] サーバ内に保存された結果から画面出力用データを抽出する。（40 ～ 58 行目）

getlast については画面 4 を参照

comma は 3 桁ごとにカンマを振る。

次ページへ続く→

```
58 itouch '_ _ _ _ _ _ _ _ _ _ FFFFFF' $tmp-check
59
60 ##########################################################
61 # HTML作成
62 ##########################################################
63 # 出力
64 cat << FIN > $tmp-calsed.list
65 ###KOUSIN_JIKOKU### $(cat $tmp-kousin)
66 ###DATE### ${sdate}
67 FIN
68
69 echo "Content-Type: text/html"
70 echo ""
71 cat ${htmd}/KANSI.DISK_VOL.HTML            |
72 calsed -f $tmp-calsed.list -               |
73 mojihame -l###REPORT_LOOP### - $tmp-check  |
74 cat
75
76 ##########################################################
77 # 終了
78 rm -f ${tmp}-*
79 exit 0
```

itouch は指定ファイルが存在しない、あるいは0バイトならば、指定した文字列でファイルの中身を初期化する。

[3] 抽出したデータをテンプレートHTMLにはめ込み、画面出力用HTMLを作成する。(64～74行目)。

calsed については画面5を参照

mojihame については画面6を参照

画面3　cgi-name はPOST形式のデータをname形式に変換する

```
$ dd bs=$CONTENT_LENGTH
place=tokyo&country=japan
$ dd bs=$CONTENT_LENGTH | cgi-name
place tokyo
country japan
```

name形式とは I：HTMLのタグ名 2：値 の2フィールドデータのこと

画面4　getlast は同一キーの最後の行を出力する

```
$ cat data
0000007 セロリ 20060201 117
0000007 セロリ 20060202 136
0000007 セロリ 20060203 221
0000017 練馬大根 20060201 31
0000017 練馬大根 20060202 127
0000017 練馬大根 20060203 514
$ getlast 1 2 data
0000007 セロリ 20060203 221
0000017 練馬大根 20060203 514
```

コード

[1]～[3]の3つに分割したセクションを順番に説明していきます。[1]画面からPOST形式のデータを取得して、name形式に変換するコードは、36～37行目に該当します。POST形式で取得したデータをユニケージ開発でよく使用するレイアウトのname形式（1:タグ名、2:値）に変換します。説明は画面3に掲載しました。

次に[2]サーバー内に保存された結果から画面出力用データを抽出するコードは、40～58行目に該当します。画面から渡された日付の結果から最新データを抽出して、画面にはめ込みできるように加工しています。最新データ抽出などで使用するコマンド「getlast」の説明は画面4に掲載しました。

最後に、[3]抽出したデータをテンプレートHTMLにはめ込み、画面出力用HTMLを作成するコードは64～74行目に該当します。パラメータの置換方法は、画面5と画面6に掲載しました。出力結果は、画面2となります。

画面5　calsed はsedの文字列置換機能の簡易版

```
$ cat $tmp-calsed.list
###KOUSIN_JIKOKU### 12:20
###DATE### 20131210
$ cat ${htmd}/KANSI.DISK_VOL.HTML
......
<td style="text-align:right;">更新時刻 ###KOUSIN_JIKOKU###</td>
......
<input type="text" size="10" maxlength="8" value="###DATE###" name="DAY_SELECT">
......
$ cat ${htmd}/KANSI.DISK_VOL.HTML | calsed -f $tmp-calsed.list -
......
<td style="text-align:right;">更新時刻 12:20</td>
```

置換前

```
.....
<input type="text" size="10" maxlength="8" value="20131210" name="DAY_SELECT">
.....
```

置換後

画面 6　mojihame は HTML の指定したパラメータ範囲で文字をはめ込み、リストを作る

```
$ cat $tmp-check
SV1 192.168.1.1 /dev/sda2 20131210122010 2,361,084,580 479,356,224 1,759,881,436 22 / 正常 FFFFFF
SV1 192.168.1.1 /dev/sda3 20131210122010 1,890,250,000 1,511,331,296 281,350,740 85 / 80%以上危険 FF9FFF
$ cat ${htmd}/KANSI.DISK_VOL.HTML
.....
<!-- ###REPORT_LOOP### -->
                        <tr bgcolor="%11">
                                <td align="left" >%1</td>
                                <td align="left" >%2</td>
                                <td align="left" >%3</td>
                                <td align="right">%5</td>
                                <td align="right">%6</td>
                                <td align="right">%7</td>
                                <td align="right">%8</td>
                                <td align="left">%9</td>
                                <td align="center">%10</td>
                        </tr>
<!-- ###REPORT_LOOP### -->
.....
$ cat ${htmd}/KANSI.DISK_VOL.HTML | mojihame -l###REPORT_LOOP### - $tmp-check
.....
                        <tr bgcolor="FFFFFF">
                                <td align="left" >SV1</td>
                                <td align="left" >192.168.1.1</td>
                                <td align="left" >/dev/sda2</td>
                                <td align="right">2,361,084,580</td>
                                <td align="right">479,356,224</td>
                                <td align="right">1,759,881,436</td>
                                <td align="right">22</td>
                                <td align="left">/</td>
                                <td align="center">正常</td>
                        </tr>
                        <tr bgcolor="FF9FFF">
                                <td align="left" >SV1</td>
                                <td align="left" >192.168.1.1</td>
                                <td align="left" >/dev/sda3</td>
                                <td align="right">1,890,250,000</td>
                                <td align="right">1,511,331,296</td>
                                <td align="right">281,350,740</td>
                                <td align="right">85</td>
                                <td align="left">/</td>
                                <td align="center">80%以上危険</td>
                        </tr>
.....
```

指定した範囲内（最初と最後の ###REPORT_LOOP### の間）のパラメータ（%1 ～ %11）に文字をはめ込む。

$tmp-check の 1 フィールド目の値を範囲内の %1 に、2 フィールド目以降も同様に範囲内の %2 以降にはめ込んでいく。

$tmp-check の 2 行目以降も 1、2……とフィールドごとに、はめ込んでいく。

$tmp-check の行数分、はめ込みを繰り返す

まとめ

保存された結果を画面に表示する部分のプログラムは 30 行ほどになります。開発期間も約 1 日です（ただし、HTML 作成時間は除く）。ユニケージ開発手法では、今回の例で示したように、機能や用途に適合したコマンドを使用することで開発を「簡単に、短時間に」行うことができます。

次回は、多数発生するトランザクションに対して処理を行う常駐型プログラムをご紹介します。