Faire de l'Excel en Perl

Ou comment produire des rapports pour son manager sans dégainer Excel

Laurent Boivin

Premier contact

- Récupération de données du SCMS
- 2 sorties :
  - un fichier de données brutes,
  - un graphe

Premier contact

- Récupération de données du SCMS
- 2 sorties :
  - un fichier de données brutes,
  - un graphe

... Les Journées Perl 2004 ...

Premier contact

- Récupération de données du SCMS
- 2 sorties :
  - un fichier de données brutes,
  - un graphe

... Les Journées Perl 2004 ...

- Un 3e format de sortie :
  - un fichier Excel

De quoi va-t-il s'agir ?

- SpreadSheet::ParseExcel::Simple (Tim Boden)
- SpreadSheet::WriteExcel (John McNamara)
- Disponibles sur le CPAN
- Pur Perl donc... multi-OS

De quoi va-t-il s'agir ?

- SpreadSheet::ParseExcel::Simple (Tim Boden)
- SpreadSheet::WriteExcel (John McNamara)
- Disponibles sur le CPAN
- Pur Perl donc... multi-OS
- Et Win32::OLE ?
  - attaque directement les .DLL via le mécanisme d'OLE
  - nécessite que MS Excel soit installé sur le poste
  - multi-Windows

Lire mon premier classeur

```
use Spreadsheet::ParseExcel::Simple;

my $wb=Spreadsheet::ParseExcel::Simple
                                    ->read("foo.xls");
foreach my $s ( $wb->sheets )
{
   while ( $s->has_data )
   {
      my @data = $s->next_row;
      #...
   }
}
```

Spreadsheet::ParseExcel::Simple

- 3 méthodes sur le classeur

```
my $xls = Spreadsheet::ParseExcel::Simple->read();
my @sheets = $xls->sheets;
my $book = $xls->book;
```

- 3 méthodes sur les feuilles

```
if ( $sheet->has_data ) ...
my @data = $sheet->next_row;
my $obj = $sheet->sheet;
```

- C'est Simple

Mon vrai 1er classeur

```
use Spreadsheet::ParseExcel;
my $wb = Spreadsheet::ParseExcel::Workbook->Parse($f);
foreach my $sh (@{$wb->{Worksheet}}) {
  $sh->{MaxRow} ||= $sh->{MinRow};
  foreach my $r ($sh->{MinRow} .. $sh->{MaxRow}) {
    $sh->{MaxCol} ||= $sh->{MinCol};
    foreach my $c ($sh->{MinCol} .. $sh->{MaxCol}) {
      my $cell = $sh->{Cells}[$r][$c];
      if ($cell) {
        push @data, $cell->{Val});
      }
    }
  }
}
```

Effet du "Calcul automatique"

- S'il est activé :

  on récupère le résultat calculé des cellules

- S'il est désactivé :

  on récupère... cela dépend du type de données :

  - texte : le texte tel quel
  - nombre : "GENERAL" pour le format par défaut, la valeur sinon
  - formule : cela dépend si le dernier résultat calculé est un nombre ou un texte

Un peu plus sur la lecture

- Le fichier est **entièrement** lu lors du read(...);
- Le module SpreadSheet::ParseExcel
  - plus souple dans la façon de lire de récupérer les données
  - récupérer brutalement une feuille est plus long à écrire
  - même comportement par rapport au "calcul automatique"

Créer mon premier classeur

```
use Spreadsheet::WriteExcel;

my $wb = Spreadsheet::WriteExcel->new("foo.xls");
my $ws = $wb->add_worksheet('Ma feuille');

foreach my $r (0..3)
   {
   foreach my $c (0..3)
      {
      $ws->write($r, $c, "cellule $r $c");
      }
   }
$wb->close();
```

Créer mon 2e classeur

```
use Spreadsheet::WriteExcel;

my $wb = Spreadsheet::WriteExcel->new("foo.xls");
my $ws = $wb->add_worksheet('sheet1');

foreach my $r (0..3)
   {
   foreach my $c ('A'..'D')
      {
      $ws->write("$c$r", "cell $c$r");  # A1
      }
   }
$wb->close();
```

Formatter les cellules

```
my $wb = Spreadsheet::WriteExcel->new("foo.xls");

my $header = $wb->add_format(
                bold => 1,
                size => 16,
                );
my $fmt = $wb->add_format();
$fmt->set_property( font => "Arial" );
$fmt->set_color("blue");

my $ws = $wb->add_worksheet('page_formattee');
$ws->write(0, 0, "Exemples de formats", $header);
$ws->write(0, 1, "cellule bleue", $fmt);
```

Ajouter des formules

- C'est une chaîne qui commence par =
- `$wb->write($r,$c,"=somme(A1:A9)");`

Ajouter des formules

- C'est une chaîne qui commence par =
- `$wb->write($r,$c,"=somme(A1:A9)");`
  - À l'exécution : `Couldn't parse formula: =somme(A1:A9)`

Ajouter des formules

- C'est une chaîne qui commence par =
- `$wb->write($r,$c,"=somme(A1:A9)");`
  - À l'exécution : `Couldn't parse formula: =somme(A1:A9)`
- `$wb->write($r,$c,"=sum(A1:A9)");`

Ajouter des formules

- C'est une chaîne qui commence par =
- `$wb->write($r,$c,"=somme(A1:A9)");`
  - À l'exécution : `Couldn't parse formula: =somme(A1:A9)`
- `$wb->write($r,$c,"=sum(A1:A9)");`
  - À l'exécution : `Couldn't parse formula: =sum(A1:A9)`

- Pas glop pas glop !

La formule du chef

- Ma 1ère ruse :

```
$wb->write($r,$c,'sum(A1:A9)');
```

et on ajoute le = dans le tableur

La formule du chef

- Ma 1ère ruse :

```
$wb->write($r,$c,'=sum(A1:A9)');
```

  et on ajoute le = dans le tableur

- Mais en fait, c'est tout bête :

```
$wb->write($r,$c,'=SUM(A1:A9)');
```

- Nom en anglais, en MAJUSCULE

Ajouter des graphes, des images

- `$ws->embed_chart('B2', 'graphe.bin');`

  `graphe.bin` doit avoir été extrait d'un fichier Excel
- `my $chart = $wb->add_chart_ext('chart01.bin', 'Chart1');`

  "This feature is new and would be best described as experimental."

- `$ws->insert_image('A1', 'perl.bmp');`
- `$ws->insert_image('A1', 'perl.png');`

Autres méthodes de saisie

- `$ws->write_string("A1", "toto");`
- `$ws->write_number("P1", 3.14159);`

- `$ws->write_col("A1", \@array);`
- `$ws->write_row("A1", \@array);`

- `$ws->write_formula ($r,$c,"=SUM(A1:A9)");`
- `$ws->repeat_formula($r,$c,"=SUM(A1:A9)");`

- `$ws->write_comment($r,$c,"blabla",...);`

Accès aux pages

- `$wb->add_worksheet("nom");`
- ```
  foreach $ws ($wb->sheets())
     {
     print $ws->get_name();
     }
  ```
- ```
  foreach $ws ($wb->sheets(0,1))
     {
     print $ws->get_name();
     }
  ```
- …

Les autres fonctionalités

- Largeur des colonnes, hauteur des lignes
- Format de la page (impression)
- Figer les lignes/colonnes
- Choix de l'encodage des caractères
- Départ de la date (1970 ou 1904)
- Groupage de lignes
- ...

Quelques exemples

- Comparaison de parcs de PC
- Constater les dégats

Comparaison de parc

- Problématique :
  - Croiser des données BO (6000 entrées) et Excel (12000 PC)
  - Tous les mois

Comparaison de parc

- Extraction des données BO vers Excel
- Filtre avec Excel pour réduire de 12000 PC à 3000
- Lecture des inventaires avec S::PE::S
- On mouline
- Création du rapport avec S::WE
- On l'envoie à sa chef

Création du rapport

```
my $workbook = Spreadsheet::WriteExcel->new($fname);
my $bold = $workbook->add_format(bold => 1);
my $ws = $workbook->add_worksheet('To be serviced');
my $row = 0;
if ( scalar @hp_headers ) {
    my $col = 0;
    foreach my $h ( @hp_headers ) {
        $ws->write($row, $col, $h, $bold); $col ++;
    }
    $row++;
}
foreach my $k ( sort @to_be_serviced ) {
    my $col = 0;
    foreach my $i ( @ {$hp{$k}} ) {
        $ws->write($row, $col, $i); $col++;
    }
    $row++;
}
```

Une autre page

```
# Worksheet Explanations
$ws = $workbook->add_worksheet('Explanations');
$row = 0;
while (<DATA>) {
    s/[\n\r]//g;
    if ( /^=B/) {
        s/^=B//;
        $ws->write($row,0,$_,$bold);
    } else {
        $ws->write($row,0,$_);
    }
    $row++;
}
```

Constater les dégats

1/3

```
# Write the headers
my $c = 0;
map { $ws->write(0, $c++, $_, $format->{header});}
  @$headers;

# Write the content
my $r = 1; # count the rows for the final total
foreach my $k ( sort keys %$data )  {
    my $z = get_zone($data->{$k}{'OPEN DATE'});

    # Fill the column with 'c' centered cells:
    map { $ws->write($r, $_, $data->{$k}->{$headers-
  >[$_]}, $format->{$z}{c}); } @{$f_of_h->{'c'}};

    # ...
```

Constater les dégats


```
# Fill the column with '-' neutral cells (no specific
  format)
    map { $ws->write($r, $_, $data->{$k}->{$headers-
  >[$_]}, $format->{$z}{n}); } @{$f_of_h->{'-'}};

    $r++;
    }

$ws->write($r, 3, "OPEN + CLOSED = TOTAL", $format-
  >{zO}{n});
$ws->write($r, 7, 'CONCATENATE(COUNT(J2:J' . $r . ')-
  SUM(J2:J' . $r .')," + ",SUM(J2:J' . $r . '),"  =
  ",COUNT(J2:J' . $r . '))', $format->{zO}{n});
$ws->write($r+2, 3, 'Dont WiFi :', $format->{zO}{n});
   $ws->write($r+2, 7, 'COUNTIF(I2:I' . $r . ',
  "WIFI*")', $format->{zO}{n});
```

Constater les dégats


```
 # Format the print out
 # Size the column (based on experiments as 'AutoFit'
can be specified)
 $ws->set_column(0, 0, 8);
 $ws->set_column(1, 1, 30);
 $ws->set_column(2, 2, 3);
 $ws->set_column(3, 3, 24);
 $ws->set_column(4, 4, 7);
 $ws->set_column(5, 5, 12);
 $ws->set_column(6, 6, 7);
 $ws->set_column(7, 7, 18);
 $ws->set_column(8, 8, 50);
 $ws->freeze_panes(1, 0);
 $ws->set_landscape();
 $ws->set_margins(0.5);
 $ws->print_area('A:I');
 $ws->repeat_rows(0);
```