

## Profesjonalne raportowanie w Perlu

# Zróbmy to w Excelu

Stare powiedzenie, szczególnie popularne w gronie użytkowników Perla mówi, że wszystko można zrobić na wiele sposobów. Zależy to nie tylko od umiejętności i przyzwyczajen programisty, ale czasami też od potrzeb i przyzwyczajen odbiorcy – użytkownika programu.

WOJCIECH BOLIMOWSKI

Widać to doskonale na przykładzie raportów, gdzie w zależności od typu grupy odbiorców wymagane jest zastosowanie innego podejścia. O ile administratorzy w zupełności zadowolają się zwykłym plikiem tekstowym, o tyle dla odbiorcy biznesowego konieczne jest zastosowanie bardziej wyszukanego formatu. Perl przychodzi także i tutaj z pomocą, oferując gotowe moduły do formatowania dokumentów np. w XML, PDF, RTF czy choćby formacie Excela. Właśnie temu ostatniemu poświęcony będzie artykuł.

## Dlaczego Excel?

Właśnie, dlaczego Excel, skoro jest tyle innych, być może ideowo bardziej słusznych for-

matów? Otóż przemawia za tym kilka powodów. Po pierwsze jest to format przenośny, z którym bez większych problemów radzą sobie aplikacje linuksowe. Wymienię najważniejsze: Gnumeric, OpenOffice i StarOffice, KOffice. Jest więc w czym wybierać. Efekt przenośności możemy spotęgować, tworząc np. własną aplikację raportującą z interfejsem WWW, czerpiącą dane wprost z korporacyjnej bazy danych. Co więcej, sporządzając raport w tym formacie, umożliwiamy użytkownikom dokonywanie własnych przeliczeń, sortowanie i tym podobne operacje, właściwe dla arkuszy kalkulacyjnych.

## Koń roboczy – moduł Spreadsheet-WriteExcel

Główną siłą napędową naszych raportów będzie moduł Spreadsheet-WriteExcel autorstwa Johna McNamary. W chwili pisania tego artykułu, dostępna najnowsza wersja modułu nosiła oznaczenie 0.42. Za pomocą tego modułu można generować binarne pliki w formacie Excela kompatybilne z wersjami Excel 5, 95, 97, 2000 i 2002.

Jak zaznacza w dokumentacji sam autor,

moduł dostarcza interfejsów do maksymalnie dużej liczby właściwości Excela. Rzeczywiście mnogość metod modułu opisanych w dokumentacji z początku może trochę onieśmielać, ale nie należy się tym zrażać. Zastosowanie ich jest bardzo intuicyjne i nawet osoby niezaznajomione z podejściem obiektowym nie powinny mieć większych problemów i będą w stanie szybko pisać własne raporty.

Żeby to udowodnić, zaczniemy w sposób tradycyjny od napisania najprostszego raportu, który de facto niczego nie raportuje, ale pozwala zapoznać się ze sposobem korzystania z modułu (Listing 1). W pierwszym kroku deklarujemy wykorzystanie modułu. Następnie deklarujemy skoroszyt używając konstruktora new, do którego przekazujemy jako argument nazwę naszego pliku helloexcel.xls. Potem tworzymy arkusz i wreszcie na koniec, w komórce o adresie (0,0) wpisujemy napis powitalny.

Pierwszy program za nami, umiemy już utworzyć skoroszyt i arkusz oraz wyświetlać napisy. Czas, żeby przyjrzeć się bliżej możliwościom oferowanym przez moduł *Spreadsheet-WriteExcel*, a są one naprawdę niemałe.

Ogólnie rzecz biorąc, dostępne metody mo-

### Listing 2. Przykłady definiowania szablonów formatowania

```
01 my $title = $workbook->addformat (); # tworzenie szablonu jako
    obiektu o nazwie title
02 $title->set_font ('Arial CE'); # przyporządkowanie do szablonu
    czcionki...
03 $title->set_size (14); # i wielkości czcionki
04 $title->set_bold (); # ustawienie typu na pogrubiony
05 my $head = $workbook->addformat (); # tworzenie innego szablonu
06 $head->set_bold ();
07 $head->set_align ('center'); # tak możemy określić rozmieszczenie
08 $head->set_bottom (); # a tutaj ustawiamy wyświetlanie krawędzi
    komórek
09 $head->set_top ();
10 $head->set_left ();
11 $head->set_right ();
12 $head->set_text_wrap; # ta metoda powoduje, że tekst w komórce bę-
    dzie zawijany
13 my $numrow = $workbook->addformat ();
14 $numrow->set_num_format ('# ### ##0.00'); # tak możemy definiować
    wyświetlanie formatów liczbowych
15 $numrow->set_right ();
```

### Listing 1. Hello Excel

```
01 #! /usr/local/bin/perl -w
02 use Spreadsheet::Write-
    Excel; # Krok 1
03 my $workbook = Spreadshe-
    et::WriteExcel->new ('hello-
    excel.xls');
04 $worksheet = $workbook->
    add_worksheet ();
05 $worksheet->write (0,0,
    'Hello Excel! ');
06 $workbook->close ();
```

## Listing 3. Definiowanie rozmiarów pól

```
01 $worksheet->set_row (0, undef, $format1); # Zastosowanie formatu
    dla wiersza
02 $worksheet->set_column ('A: A', 40, $format2); # Zastosowanie in-
    nego formatu dla kolumny 1 i ustawienie szerokości na 40
03 $worksheet->write ('A1', 'Hello');          # Wyświetlenie napisu
    w komórce A1
```

dułu możemy podzielić na metody związane z własnościami skoroszytu, arkusza, komórki, układu strony i formatowania. Zanim zbudujemy pierwszy funkcjonalny raport, przyjrzymy się bliżej niektórym z nich.

Bardzo istotną z punktu widzenia webowej (CGI) aplikacji raportującej jest możliwość przekierowania standardowego wyjścia. Do tego celu wykorzystujemy specjalną nazwę pliku '-'. Dzięki temu zamiast zapisywać dane do pliku, wysyłamy je za pośrednictwem serwera WWW wprost do przeglądarki.

```
my $workbook = Spreadsheet::➤
WriteExcel->new ('-');
```

Nie musimy się także przejmować obsługą bardziej złożonych raportów, składających się z kilku arkuszy. Kolejne arkusze możemy dodawać używając metody *add\_worksheet*:

```
$report1 = $workbook->➤
add_worksheet ();      # Sheet1
$report2 = $workbook->➤
add_worksheet ➤
('Registered Customers'); ➤
# Registered Customers
```

To co najbardziej nadaje profesjonalizmu raportowi, to jego wygląd, dlatego szczególną uwagę powinniśmy poświęcić zagadnieniom formatowania. Dysponujemy w tym zakresie praktycznie pełnym wachlarzem możliwości Excela. Właściwości formatowania uzyskujemy poprzez grupowanie poszczególnych atrybutów w szablony. Do tworzenia poszczególnych szablonów formatowania służy metoda *add\_format*. Przykłady przedstawione na Listingu 2 pozwalają zapoznać się ze sposobem tworzenia własnych szablonów.

To tylko część możliwości. Oprócz podsta-

wowych metod przedstawionych w przykładach, możliwe jest m. in. ustawianie koloru czcionek i wypełnienia komórek, osadzanie bitmap, stosowanie formuł itd. Na szczęście do modułu dołączona jest obszerna dokumentacja z licznymi przykładami, gdzie można się zapoznać z interesującymi szczegółami. My natomiast będziemy kontynuować prace nad stworzeniem naszego raportu.

Przed przystąpieniem do wyświetlenia danych pozostaje nam jeszcze określenie wielkości komórek. Istnieją tutaj dwie główne metody pozwalające sprawować kontrolę nad wierszami i kolumnami: *set\_row* i *set\_column*. No i wreszcie przystępujemy do wyświetlania zawartości komórek. Mamy tutaj do dyspozycji metody: *write* (format ogólny), *write\_string* (do wyświetlania łańcuchów tekstowych) i *write\_number* (do wyświetlania liczb).

## Budujemy raport

Przedstawione do tej pory metody są wystarczające do sporządzenia pierwszego raportu. Jedyne co nam pozostaje, to utworzenie interfejsu użytkownika i pobranie danych ze źródła, w naszym przypadku będzie nim relacyjna baza danych Open Source PostgreSQL.

Do połączenia z bazami danych w Perlu używane są moduły DBI i DBD. Biblioteka DBI to uniwersalny, ujednolicony interfejs

## Listing 4. Przykład formularza raportu

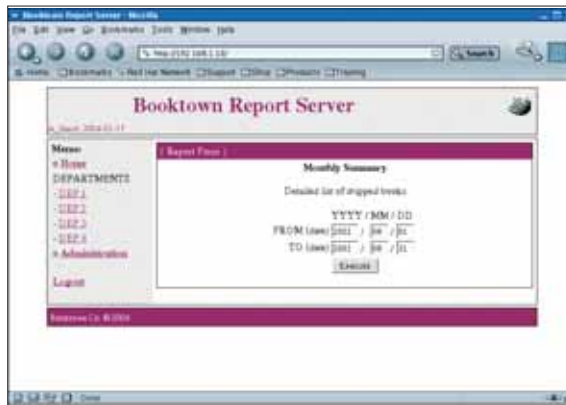
```
01 <b>Monthly Summary</b>
02 <P>Detailed list of
    shipped books</P>
03 <form method=POST
    action="booktown.pl">
04   <table border=0>
05     <tr valign=center>
06       <td align=center>
07         </td>
08       <td align=center>
09         YYYY /
10       </td>
11       <td align=center>
12         MM /
13       </td>
14       <td align=center>
15         DD
16       </td>
17     <tr valign=top >
18       <td align=right>
19         FROM (date)
20       </td>
21       <td align=left
    valign=center>
22     <input type=text
    name=year_from size=4
    maxlength=4> /
23     </td>
24     <td align=left>
25       <input type=text
    name=month_from size=2
    maxlength=2> /
26     </td>
27     <td align=left>
28       <input type=text
    name=day_from size=2
    maxlength=2>
29     </td>
30   </tr>
31   <tr valign=top >
32     <td align=right>
33       TO (date)
34     </td>
35     <td align=left
    valign=center>
36     <input type=text
    name=year_to size=4
    maxlength=4> /
37     </td>
38   <td align=left>
39     <input type=text
    name=month_to size=2
    maxlength=2> /
40     </td>
41     <td align=left>
42     <input type=text
    name=day_to size=2
    maxlength=2>
43     </td>
44   </tr>
45   <tr>
46     <td colspan=2
    align=left>
47     <input type=submit
    value="Execute">
48     </td>
49   </tr>
50   <td colspan=2>
51   </td>
52 </tr>
53 </table>
54 </form>
```

baz danych, dzięki któremu można uzyskać dostęp do danych składowanych w bazach różnych typów. DBD natomiast jest dedykowanym sterownikiem do konkretnej bazy danych, zapewniającym komunikację między interfejsem DBI i bazy danych. W przypadku PostgreSQL mamy do dyspozycji sterowniki *DBD::Pg* i *DBD::PgPP* (dla baz Oracle istnieje bardzo dobry sterownik *DBD::Oracle*).

Co do interfejsu użytkownika, to dobrym pomysłem wydaje się zastosowanie technologii intranetowej, która jest bardzo elastyczna i wygodna (nie wymaga instalowania dodatkowego oprogramowania na stacjach roboczych). Pozwoli to ponadto na sparametryzowanie wykonywanych raportów. W dalszej części artykułu przedstawiony zostanie sposób budowy kompletnego raportu. Załóżmy, że będzie to raport wspomagający pracę księgarń, a jego zadaniem będzie udzielanie informacji o sprzedaży książek w określonym przedziale czasu.

## Interfejs użytkownika

Biorąc pod uwagę powyższe założenia, możemy przystąpić do budowy interfejsu formularza HTML. Formularz pobierałby daty: po-



Rysunek 1. Formularz osadzony na stronie WWW.

czątkową i końcową, które byłyby przekazywane mechanizmem CGI do skryptu w Perlu, gdzie podstawiane byłyby następnie do zapytania SQL. Kod przykładowego formularza mógłby być na przykład taki, jak przedstawiono na Listingu 4.

Dane z formularza: *year\_from*, *month\_from*, *day\_from* oraz *year\_to*, *month\_to*, *day\_to* przekazywane są metodą POST do skryptu wykonawczego *booktown.pl*. Wygląd formularza osadzonego na stronie WWW przedstawia Rysunek1. Jak widać, wykorzystując opisywany w artykule mechanizm można pokusić się o stworzenie szerszego systemu raportowania, obejmującego całościowe potrzeby firmy.

## Pobieranie danych do raportu

Jedyną co nam pozostało do zrobienia w tym momencie, to pozyskanie danych. Załóżmy, że mamy już przygotowane zapytanie SQL. W takim razie w skrypcie najpierw musimy obsłużyć odbiór danych przesłanych z formularza. Do tego celu doskonale nadaje się funkcja *param* z modułu CGI.pm (pierwsza część skryptu Listing 5). Otrzymane dane poddajemy obróbce funkcji *chomp*, aby obciąć ewentualne znaki końca linii, a następnie skleamy, przygotowując w ten sposób dane do podstawienia do zapytania SQL.

Połączenie z bazą odbywa się w sposób typowy dla DBI. Najpierw tworzymy uchwyt *\$dbh*, poprzez połączenie z bazą danych za pomocą sterownika *PgPP*. Następnie tworzymy uchwyt dyrektywy *\$sth* przygotowującej zapytanie SQL i wykonujemy je – *\$sth->execute()*. Wyniki zwrócone zapytaniem SQL pobieramy wsadowo, korzystając z metody *fetchall\_arrayref()*. Metoda ta pobiera cały zestaw wynikowy do struktury danych Perla, którą możemy potem dowolnie obrabiać. Następnie zamykamy uchwyt dyrektywy i uchwyt połączenia, kończąc pracę z bazą danych. Teraz możemy już śmiało przystąpić do generowania raportu.

## Listing 5. Pierwsza część skryptu – pobieranie danych

```

01 #!/usr/local/perl/bin/perl -w          21
02 use strict;                            22 chomp ($yf);
03 use DBI;                                23 chomp ($mf);
04 use CGI qw(:standard);                 24 chomp ($df);
05 use Spreadsheet::WriteExcel;          25 chomp ($yt);
06                                         26 chomp ($mt);
07 ### Parametry konfiguracyjne          27 chomp ($dt);
08 ### polaczenia z baza                  28
09 my $dbname="booktown";                 29 my $date_f="$yf" . "-" .
10 my $login="postgres";                   "$mf" . "-" . "$df";
11 my $pass="postgres";                    30 my $date_t="$yt" . "-" .
12                                         "$mt" . "-" . "$dt";
13 my                                       31
14 $ExcelFile='booktown_report.xls';      32 ### Zapytanie SQL ###
15 ### Odbior danych z                    33 my $sel="SELECT
16 formularza WWW ###                     34 s.ship_date,
17 my $yf=param("year_from");             35 b.title,
18 my $mf=param("month_from");             36 e.isbn,
19 my $df=param("day_from");                37 su.subject,
20 my $yt=param("year_to");                 38 st.cost
21 my $mt=param("month_to");                 39 FROM
22 my $dt=param("day_to");                  40 shipments s,
23                                           41 editions e,
24                                           42 books b,
25                                           43 stock st,
26                                           44 subjects su
27                                           45 WHERE
28                                           46 s.isbn=e.isbn
29                                           47 AND e.book_id=b.id
30                                           48 AND b.subject_id=su.id
31                                           49 AND e.isbn=st.isbn
32                                           50 AND s.ship_date between
33                                           ('$date_f') AND ('$date_t')
34                                           51 ORDER BY
35                                           52 su.subject ASC";
36                                           53
37                                           54 ### Polaczenie z baza i
38                                           pobranie danych ###
39                                           55 my $dbh=DBI-
40                                           >connect("dbi:PgPP:$dbname",
41                                           $login, $pass);
42                                           56 my $sth=$dbh->prepare($sel);
43                                           57 $sth->execute();
44                                           58 my $wyniki = $sth-
45                                           >fetchall_arrayref();
46                                           59 $sth->finish();
47                                           60 $dbh->disconnect;
```

## Generowanie pliku Excel

Ponieważ raport będzie generowany po stronie serwera WWW, pierwszą czynnością przed rozpoczęciem generowania właściwego pliku, powinno być wysłanie do przeglądarki informacji o typie MIME (Listing 6). Do tego celu służą poniższe informacje nagłówka, wypisywane zwykłym poleceniem *print*.

```
print "Content-type: >
application/vnd.ms-excel\n";
print "Content-Disposition: >
attachment; >
filename=$ExcelFile\n\n";
```

W dalszej kolejności korzystamy z poznanych już wcześniej metod modułu Spreadsheet-WriteExcel. Tworzymy uchwyt arkusza i skorygowano, określamy rozmiary kolumn, definiujemy formaty. Następnie wypisujemy tytuł i tworzymy nagłówki kolumn.

Wreszcie przystępujemy do wprowadzenia samych danych. Najpierw wyodrębniamy poszczególne wiersze ze złożonej struktury danych *@\$wyniki*, będącej referencją do tablicy zmiennej *\$item*, a z niej z kolei w podobny sposób wydobywamy wartości poszczególnych pól. Wartości pól wypisujemy w pętli w standardowy sposób, korzystając z metody *write* (). Na zakończenie zamykamy uchwyt arkusza. Rezultat działania skryptu jest przedstawiony na Rysunku 2.

## Podsumowanie

Wykorzystując opisaną w tekście metodę sporządzania raportów, można zbudować kompletne środowisko raportowe, będące w stanie zaspokoić dowolne potrzeby raportowe firmy. Jej największą zaletą,

Rysunek 2. Gotowy raport w postaci pliku Excel.

oprócz prostoty i dużej szybkości tworzenia raportów, jest możliwość sięgania do różnych baz danych i niezależność od platformy systemowej.

## Listing 6. Druga część skryptu – generowanie pliku w formacie Excel

```
01 ### Generowanie raportu ###
02 print "Content-type:
application/vnd.ms-excel\n";
03 print "Content-Disposition:
attachment;
filename=$ExcelFile\n\n";
04
05 my $workbook =
Spreadsheet::WriteExcel-
>new("-");
06 my $rap = $workbook-
>addworksheet("Summary");
07
08 $rap ->set_column(0, 0, 6);
09 $rap ->set_column(1, 1, 26);
10 $rap ->set_column(2, 2, 28);
11 $rap ->set_column(3, 3, 12);
12 $rap ->set_column(4, 4, 18);
13 $rap ->set_column(5, 5, 8);
14
15 my $title = $workbook-
>addformat();
16 $title->set_font("Arial CE");
17 $title->set_size(14);
18 $title->set_bold();
19
20 my $header = $workbook-
>addformat();
21 $header->set_bold();
22 $header->set_align('center');
23 $header->set_bottom();
24 $header->set_top();
25 $header->set_left();
26 $header->set_right();
27 $header->set_text_wrap;
28
29 my $num = $workbook-
>addformat();
30 $num->set_num_format('# ###
##0.00');
31
32 my $toright = $workbook-
>addformat();
33 $toright->set_align('right');
34
35 my $toleft = $workbook-
>addformat();
36 $toleft->set_align('left');
37
38 my $center = $workbook-
>addformat();
39 $center->set_align('center');
40
41 my $cap = $workbook-
>addformat();
42 $cap->set_font("Arial CE");
43 $cap->set_size(8);
44 $rap ->write(0, 0, "Booktown
Detailed Report: $date_f -
$date_t", $title);
45 $rap ->write(3, 0, "No",
$header);
46 $rap ->write(3, 1, "Ship
date", $header);
47 $rap ->write(3, 2, "Book
title", $header);
48 $rap ->write(3, 3, "ISBN No",
$header);
49 $rap ->write(3, 4,
"Category", $header);
50 $rap ->write(3, 5, "Price",
$header);
51
52 my $line=4;
53 my $no=1;
54 foreach my $item (@$wyniki) {
55 my
($shipdate,$booktitle,$isbn,$
category,$price) = @$item;
56 $rap ->write($line, 0, $no,
$toright);
57 $rap ->write($line, 1,
$shipdate, $center);
58 $rap ->write($line, 2,
$booktitle, $center);
59 $rap ->write($line, 3,
$isbn, $toright);
60 $rap ->write($line, 4,
$category, $center);
61 $rap ->write($line, 5,
$price, $num);
62
63 $line++;
64 $no++;
65 }
66
67 $workbook->close();
```