Historia pewnej strony WWW…

Wojciech Myszka
Drzewieckiego 18/11, 54–129 Wrocław
myszka@norka.eu.org

Pracę zgłosił: Jacek Kmiecik

Streszczenie

Serwery WWW to coraz popularniejsza forma prezentowania różnego rodzaju informacji. Czy człowiek przyzwyczajony do pisania różnego rodzaju tekstów, używający LaTeXa, może łatwo wykorzystać swoje przyzwyczajenia do zarządzania zawartością serwisu WWW? A jakie narzędzia ma do wyboru? Krótka historia pewnego serwisu WWW zarządzanego jak książka.

Wstęp

W podtytule powinno być „…albo czy LaTeX2HTML nadaje się do tworzenia serwisów WWW?"

Sytuacja wyjściowa była taka: Rektorzy kilku wrocławskich uczelni wyższych podpisali porozumienie o współpracy powołujące „międzyuczelniane studia magisterskie z zakresu najnowszych technologii". Powstał (w Wordzie) odpowiedni informator, a po jakimś czasie zadano mi pytanie czy nie można, dodatkowo, stworzyć stron WWW odpowiadających zawartością informatorowi.

Konwersja z programu MS Word

Ponieważ HTML znam średnio, a przekształcić trzeba było twór dosyć statyczny pomyślałem o przejściu Word→LaTeX→HTML, a nie o budowie portalu czy zastosowaniu systemu *Content Management*.

Pierwsza cześć zadania była koszmarna: jak zwykle broszura została sklejona z tekstów dostarczonych przez wiele osobistości, które pisały w Wordzie bez ładu i składu. Sklejenia dokonała kolejna osoba dbając już tylko o to, żeby strony się nie rozłaziły i cały tekst wyglądał mniej-więcej jednakowo. Żadna struktura logiczna nie powstała – było tylko formatowanie.

Próba użycia konwertera **rtf2LaTeX 2$_\varepsilon$** skończyła się niepowodzeniem – próbował on dostosowywać się do nieistniejącej struktury. Powodowało to, że niektóre wytłuszczenia awansowały do rangi tytulariów… Ale wydłubał ilustracje z tekstu.

Skończyło się na metodzie cut & paste z Worda do zwykłego edytora i pracowitym, ręcznym formatowaniu tekstu. Moje doświadczenia sugerują, że jest to jedna ze skuteczniejszych metod!

Konwersja do HTML

Następny problem – jakiego narzędzia użyć do automatycznej konwersji LaTeX→HTML? Sytuacja jest o tyle dobra, że istnieje ich kilka, poniżej lista ciekawych zestawień:

- *(La)TeX conversion to HTML* w *UK List of TeX Frequently Asked Questions*.
- W zestawieniu narzędzi na serwerze W3C „Word Processors Filters".
- W katalogu Google: Computers → Software → Typesetting → TeX → Converters.

Do najpopularniejszych należą:

- TTH
- TeX4ht
- LaTeX2HTML
- HEVEA

Jednym z ciekawszych[1] programów jest pyLaTeX.

W środowisku Windows jest nawet narzędzie, które z jednego panelu potrafi kilka z nich uruchomić na kliknięcie myszą – `http://www.mayer.dial.pipex.com/tex.htm`: TeX Converter.

Z niektórych z nich nawet kiedyś skorzystałem, czasami tylko testowałem i miałem różne doświadczenia.

Zamiana książki na strony WWW jest stosunkowo prosta – zapewnić należy jedynie formę stosunkowo wygodnej nawigacji. Zazwyczaj wystarcza „klasyczny" spis treści. Mi zależało jeszcze na wykorzystaniu automatycznie generowanego indeksu. Wzorów matematycznych nie było.

Z drugiej strony chodziło też o stworzenie tworu stosunkowo prostego ale, w jakimś zakresie odpowiadającego współczesnym trendom i w miarę nowoczesnego.

Trzecim celem byłą też chęć poduczenie się CSS (kaskadowych arkuszy styli) – żeby, przy okazji, być „bardziej na bieżąco" – co też narzucało pewne wymagania na użyte narzędzie.

[1] Ale tylko z takiego powodu, że jest w stanie powstawania – być może da się nim sterować.

TTH Jeden z podstawowych problemów to licencja. Drugi – wersja darmowa tworzy jeden długi plik HTMLowy (bez podziału na fragmenty). Większe możliwości daje wersja komercyjna. Dostępne wersje pracujące zarówno w środowisku Windoows jak i Linux/Unix.

HEVEA [3] W pewnych sytuacjach może wymagać dodatkowej konfiguracji przeglądarki – niestety widać wyraźnie różnicę w prezentacji symboli pomiędzy IE a Mozillą (na korzyść tej ostatniej). Nie przypominam sobie aby oferowała wsparcie dla naszych literek. Dostępne wersje pracujące zarówno w środowisku Windoows jak i Linux/Unix.

TEX4ht [2] Z pewnych względów wydaje się najbardziej elastycznym narzędziem. Co więcej wykorzystuje natywne właściwości systemu LATEX (choć trzeba przyznać, że tworzone pliki DVI są tak przeładowane poleceniami `special`, że mój yap odmawiał posłuszeństwa. Z drugiej strony – **niemożliwy** do zainstalowania: w MiKTeXu, którego używam, istnieje, co prawda, odpowiedni pakiet, ale pozbawiony plików wykonywalnych. Te trzeba ściągać ze stron Autorów. A i tak nie działał... Dokumentacja zagmatwana... Wersja z TL7 (Linux) odmawiała współpracy z posiadanym ImageMagickiem. Szybko dałem sobie spokój.

Dostępne wersje pracujące zarówno w środowisku Windoows jak i Linux/Unix.

PyLATEX [4] Bardzo ciekawe rozwiązanie – konwerter napisany w języku Python. Jak go oglądałem – był w bardzo wstępnej fazie rozwoju. Jak dziś patrzę, nie rozwinął się zbytnio. Teoretycznie daje szanse na konwersję do postaci HTML lub XML. Autor jako jedną z głównych zalet wskazuje fakt zaprogramowania go w języku Python, który ma być łatwiejszy od Perla.

Dostępne wersje pracujące zarówno w środowisku Windoows jak i Linux/Unix. Wymaga instalacji języka Python.

LATEX2HTML [1, 2] Najstarszy: 10lat! Ale okres najbardziej bujnego rozwoju ma już poza sobą. Dający ogromne możliwości rozbudowy, ale napisany w Perlu, co dla niektórych (w tym i mnie) jest przeszkodą nie do przebycia. Dobrze obsługuje język polski, zawiera wsparcie dla CSSa. Dostępne wersje pracujące zarówno w środowisku Windoows jak i Linux/Unix (wymaga instalacji Perla).

Daje duże możliwości konfiguracji z poziomu parametrów. No, i używam go już od ładnych paru lat.

To ostatnie i permanentny brak czasu zdecydowały o wyborze LATEX2HTML.

Projekt układu stron

Dokument, który przetwarzałem, w sposób naturalny dzielił się na kilka części: Wstęp i uzasadnienie powstania projektu oraz oferta programowa uczelni i Instytutów PAN. Ten podział miał pozostać.

Podstawowe wymaganie „dodatkowe” to zapewnienie możliwości przenoszenia się pomiędzy poszczególnymi częściami i nawigacji w ramach jednej części (pliku) zrealizowana za pomocą odpowiedniego menu. I jeszcze indeks nazwisk.

Jak to tłumaczyć na strukturę dokumentu? Poszczególne części zostały potraktowane jako jednostki dokumentu poziomu section (klasa article), które, ewentualnie, dzieliły się na jednostki niższego rzędu.

Nawigacja obejmująca jednostki tekstu najwyższego poziomu powinna (może) zatem zostać zrealizowana za pomocą spisu treści (ale musiałby być generowany wraz z każdym plikiem); nawigacja poziomu niższego (`\subsection{}`) za pomocą bardziej szczegółowego spisu treści wewnątrz jednostki tekstu. I te wymagania nie były łatwe do automatycznego uzyskania.

Indeks został zrealizowany w sposób standardowy i klasyczny (choć były kłopoty z sortowaniem – te polskie litery).

Jednym ze sposobów realizacji zamierzenia była konwersja z użyciem dostępnego w LATEX2HTML stylu frames. Niestety, generuje on strony z ramkami, a tego chciałem uniknąć za wszelką cenę. (W ten sposób jest prezentowana dokumentacja systemu: `http://www-texdev.mpce.mq.edu.au/l2h/docs/manual/`.)

Możliwości LATEX2HTML

LATEX2HTML znakomicie nadaje się do konwersji „książki” do postaci HTML. Każda jednostka tekstu zostaje przekształcona w osobny plik, a program automatycznie tworzy spis treści i zapewnia łatwą nawigację pomiędzy kolejnymi jednostkami oraz „w górę” (do jednostki nadrzędnej).

Niestety program nie dorobił się porządnej wersji polskiej: brak jest standardowo używanych przez program klawiszy z polskimi odpowiednikami słów „Next”, „Previous”, „Up” (mam, co prawda, gdzieś jakiś zestaw takich klawiszy, ale żeby z nich korzystać, trzeba pracowicie przerobić plik `polski.perl` i później pamiętać przy kolejnym upgrade żeby go nie nadpisać...). Przyznać jednak trzeba, że Autorzy zapewnili podstawową obsługę stylów „polish” (Babel) i „polski”.

Niestety, w tym ostatnim, zaimplementowali notację ciachową w taki sposób, który uniemożliwia wprowadzenie jakiegokolwiek URLa (`http://!`); a i wprowadzanie polskich liter z jej wykorzystanie nie jest specjalnie łatwe. Co gorsza translacja ciachów zrealizowana jest w sposób istnie Perlowy – jedna linijka, no, długa dosyć, i już. Po kilku próbach zrozumienia i zmodyfikowania – zakomentowałem i działa (choć zupełnie nie wiem co straciłem). Ale teraz sekwencja `//` oznacza dwa ciachy i już!

Tekst źródłowy powinien być zakodowany w ISO-8859-2, inne tablice konwersji nie zostały przygotowane. Stosunkowo późno wprowadzono poprawki do kodu pozwalające na wybór kodowania dokumentu docelowego; dziś można wybrać kodowanie ISO-8859-2 albo unicode (UTF-8) w dwu wariantach: znaki kodowane binarnie albo jako encje numeryczne `&#nnn;`.

Program generuje kod HTML w wersjach 2.0, 3.0, 3.2, 4.0, 4.1. Obsługuje (w pewnym zakresie) CSS.

Fantastycznie sprawdza się w tekstach z wzorami matematycznymi – co prawda wszędzie gdzie nie jest w stanie czegoś skonwertować do HTMLa – konwertuje do obrazka lub serii obrazków. Niezły przykład tekstu matematycznego skonwertowanego z wykorzystaniem LaTeX2HTML znajduje się pod adresem `http://www.immt.pwr.wroc.pl/kniga`.

Parametry podawane w wywołaniu programu (lub wartość zmiennych środowiska lub zawartość pliku konfiguracyjnego `.latex2html-init`) pozwalają na sterowanie funkcjami konwertera w szerokim zakresie bez modyfikacji źródeł.

Korzystając z możliwości oferowane przez pliki stylów o nazwie `html.sty` i `hthtml.sty` można pewnymi zachowaniami programu sterować dodatkowo „z wnętrza” pliku źródłowego.

Funkcjonalność oferowana przez pakiet stylu realizowana jest przez odpowiedni zestaw procedur w języku Perl. Daje to (ale tylko tym, którzy są odpowiednio biegli) szansę rozszerzania możliwości konwertera. W szczególności plik `polski.perl` zawiera wszystkie(?) ustawienia istotne dla przetwarzania tekstów w języku polskim. Pozwala to, w pewnym zakresie, ingerować w zakres i sposób polonizacji. Na szczęście znaczną cześć polonizacji można wykonać definiując odpowiednie zmienne.

Pozwala na definiowanie i używanie nowych poleceń oraz środowisk, ale w przypadku tych ostatnich – standardowo realizację zleca LaTeXowi i generuje obrazek, chyba, że stworzymy odpowiednią procedurę (w języku Perl) dokonującą odpowiedniego przetwarzania.

Wsparcie dla CSS w LaTeX2HTML

Podczas pierwszego przebiegu LaTeX2HTML tworzy plik CSS o nazwie *jobname*`.css` zawierający podstawowe definicje. Odwołania do pliku styli zawarte są w każdym tworzonym pliku HTML.

Pozostaje sprawą użytkownika co się w tym pliku znajdzie. W kolejnych przebiegach plik CSS – o ile już istnieje – nie jest modyfikowany – co zapobiega niszczeniu zmian wprowadzanych przez użytkownika.

W przypadku gdy zdecydujemy aby był tworzony HTML w wersji większej lub równej 4.0 – do dyspozycji mamy dodatkowe możliwości.

Użycie w tekście źródłowym konstrukcji:

```
\begin{flushright}
zawartość
\end{flushright}
```

Powoduje wygenerowanie w pliku HTML:

```
<DIV ALIGN="RIGHT"> zawartość</DIV>
```

a w pliku CSS:

```
DIV.flushright      {   }
```

ale, jak widać, przydatność takiego rozwiązania jest żadna.

Natomiast konstrukcja typu (aby dodatkowa para nawiasów kwadratowych mogła być użyta należy skorzystać z pakietu `html.sty`!):

```
\begin[]{flushright}
zawartość
\end{flushright}
```

generuje w pliku HTML:

```
<DIV ALIGN="RIGHT" ID="flushright27"
CLASS="flushright">zawartość</DIV>
```

a w pliku CSS:

```
DIV.flushright      {   }
#flushright27       {   }
```

Jest już znacznie lepiej...

Jako opcjonalny parametr środowiska można podać albo **selektor klasy** (czyli to, co pojawi się w pliku HTML w cudzysłowach po `CLASS=`), albo **specyfikację stylu**, albo jedno i drugie:

```
\begin[ala|color=red]{flushright}
zawartość
\end{flushright}
```

co da odpowiednie efekty w plikach HTML i CSS:

```
<DIV ALIGN="RIGHT" ID="flushright29"
CLASS="ala">zawartość </DIV>
```

```
DIV.ala     {   }
#flushright29       { color : red }
```

Co już zaczyna mieć sens, zwłaszcza że posiadamy dodatkowe polecenia pozwalające definiować styl klasy:

```
\htmlsetstyle[DIV]{ala}
              {text-decoration=underline}
```

Co jeszcze warto wiedzieć? Zmiany kroju i wielkości czcionki oraz wszystkie środowiska wewnętrzne (*inline*) – tłumaczone są na elementy `SPAN`.

Środowiska i wszystko to co jest składane w trybie blokowym – tłumaczone jest na elementy `DIV`. Tablice (środowiska `table` i `array` tłumaczone jest na elementy `TABLE`; środowiska typu `verbatim` na `PRE`.

Wszystko działa bardzo ładnie za wyjątkiem list – LaTeX2HTML nie robi z nimi nic, tłumacząc je tylko na odpowiednie polecenia HTML. Niestety zanurzenie listy w jakimkolwiek innym otoczeniu – psuje całą konstrukcję.

Dodatkowo program elementom nawigacyjnym również nadaje odpowiednie klasy i identyfikatory. I tak:

- Panel Nawigacyjny opisany jest jako:

```
<DIV CLASS="navigation">
 <!--Navigation Panel-->
</DIV>
```

- dodatkowo Spis Treści oznaczony jest jako:

```
<UL CLASS="TofC">
 <!--Table of Contents-->
</UL>
```

- oraz „małe spisy treści” jako:

```
<UL CLASS="ChildLinks">
 <!--Table of Child-Links-->
</UL>
```

Ale już tytuł spisu treści (czy tytuł „małego spisu treści”) nie ma nadanych żadnych specjalnych „atrybutów” co utrudnia sterowanie ich wyglądem czy położeniem.

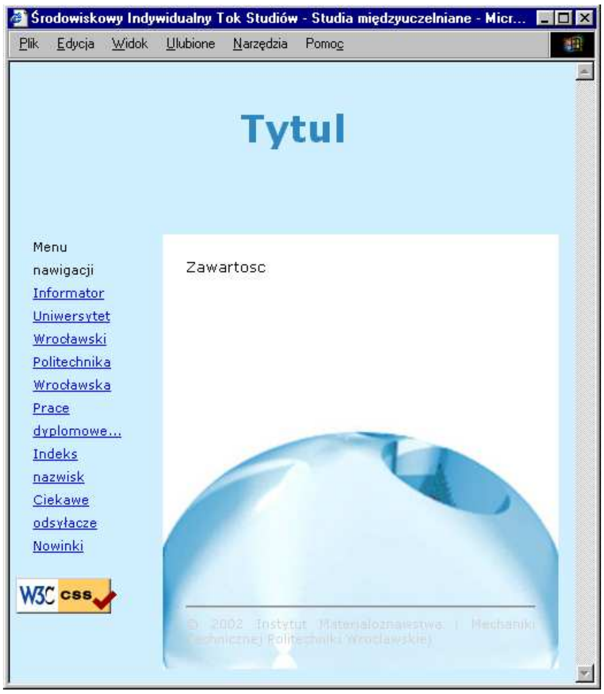
Tak więc możliwości dostarczane „out of the box” przez LaTeX2HTML okazały się niewystarczające do realizacji mojego zamysłu.

Projekt strony

Założenia jakie przyjąłem były następujące:

1. Nie będzie ramek.
2. Ma być zachowana podstawowa funkcjonalność w przeglądarce tekstowej.
3. Podział strony na pola nie będzie realizowany za pomocą tabel.
4. Wygląd strony oglądanej współczesnym Internet Explorerem jak i współczesną Mozillą będzie zbliżony.

Sam układ miał być następujący: u góry tytuł, po lewej stronie menu nawigacji i prawie cała reszta – okno z zawartością; na dole tego okna data modyfikacji i copyrighty, jak na poniższej ilustracji:



Marzenie było takie, żeby menu nawigacji było wyświetlane na wskazanej pozycji względem **okna**, a nie **zawartości**, ale udało się to tylko uzyskać w Mozilli a nie udało się w IE, więc dałem spokój.

Realizacja

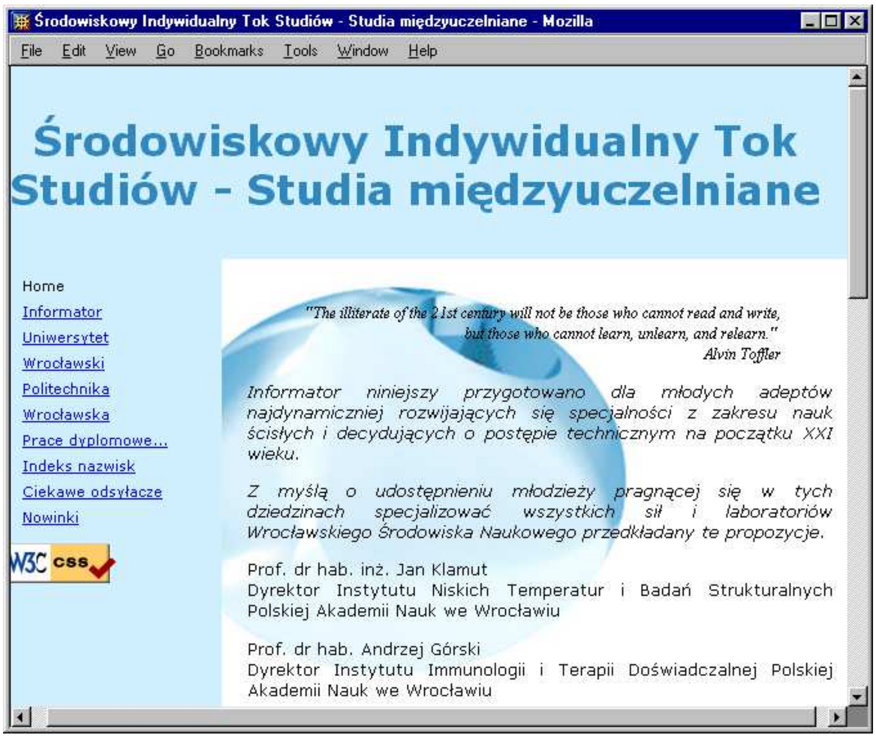
Realizacja „nie wyszła” w tym sensie, że nie udało się zaprojektować takiej **struktury** dokumentu, realizowanej **wyłącznie** za pomocą standardowych poleceń LaTeX, która po automatycznej translacji LaTeX → HTML dawałaby wymarzony efekt…

Skończyło się na umieszczeniu w treści dokumentu odpowiednich makr generujących źródłowy HTML, które pozwoliły, w pewnym zakresie, marzenia zrealizować.

Jeżeli teraz trzeba dokonywać jakichkolwiek modyfikacji stron – wprowadzam je w w „normalny”, dobrze opanowany, sposób: modyfikując LaTeXowy kod; później dokonując konwersji i kopiując pliki na serwer WWW.

Menu nawigacji zrealizowane jest jako lista z dodanymi hipertekstowymi odsyłaczami plus polecenia `\label` w odpowiednich miejscach. Polecenia `\label` i `\htmlref` załatwiają również całą resztę nawigacji wewnątrz dokumentu.

Do translacji użyłem polecenia następującego:




```
latex2html -address 0 -info 0 \
 -html_version "4.0,latin2,unicode" \
 -nonavigation -notransparent \
 -numbered_footnotes -link 0 -split 5 \
 informator.tex
```

Znaczenie parametrów jest następujące:

`-address 0` wyłącza generowania informacji o dacie i nazwie użytkownika dokonującego konwersji;

`-info 0` wyłącza standardową informację o programie użytym do konwersji;

`-html_version "4.0,latin2,unicode"` wersja HTML oraz sposób kodowania znaków: znaki w dokumencie w standardzie ISO-8859-2, standard kodowania na wyjściu – UNICODE; dodanie na końcu, po przecinku, `latin2` spowoduje kodowanie w ISO-8859-2;

`-nonavigation` wyłączenie standardowej nawigacji generowanej przez LaTeX2HTML;

`-notransparent` wyłączało generowanie obrazków z przezroczystym tłem (przezroczyste tło powodowało nieprzyjemne efekty w załączonych ilustracjach z półtonami);

`-numbered_footnotes` notki generowane z numerami;

`-link 0` poziom generowanego spisu treści (czyli bez spisu treści);

`-split 5` poziom, poniżej którego jednostki tekstu nie są umieszczane w osobnych plikach (0 – wszystko w jednym pliku, 1 – part,..., 5 – subsubsection,...)

Stronę można obejrzeć pod adresem `http://biomat.immt.pwr.wroc.pl`.

Literatura

[1] Nikos Drakos, Ross Moore. The LaTeX2html Translator, 1999. `http://www-texdev.mpce.mq.edu.au/l2h/docs/manual/`.

[2] Michel Goossens, Sebastian Rahtz. *The LaTeX Web companion: integrating TeX, HTML, and XML*. Tools and Techniques for Computer Typesetting. Addison-Wesley Longman, Harlow, Essex CM20 2JE, England, 1999. With Eitan M. Gurari and Ross Moore and Robert S. Sutor.

[3] Luc Maranget. HeVeA: LaTeX to HTML translator. World-Wide Web document., 1998.

[4] Kevin Smith. pyLaTeX Developer's Guide, 2001. `http://pylatex.sourceforge.net/pylatex/index.html`.