de la HTML la PDF (cu Perl şi LaTeX)

Vlad Bazon, `http://sitsco.com`

Dec 2007

Obţinem varianta tipărită a unui document HTML, folosind modulul *perl* HTML::Latex şi limbajul de marcare/programare LaTeX; o anumită subrutină *perl* va asigura *download*-area documentului PDF obţinut.

Din nou, despre *documente*

document provine din latinescul **documentum** care se trage de la **docere** care înseamnă **a învăţa**; "document" este asociat adesea cu "a informa", dar "a informa" — şi-atât — este o *marginalizare* (obişnuită astăzi) pentru "a învăţa".

În principiu, Web-ul este o infinitate de documente "electronice" care sunt interconectate între ele şi sunt accesibile de oriunde (datorită unor scheme universale de adresare, de identificare şi de comunicare). Documentele de pe Web *nu* au în vedere "formatul A4" (nu vizează operaţia de tipărire pe hârtie), pentru motivul evident că de pe hârtie nu este posibilă interconectarea cu alte documente.

Dar operarea cu documente Web poate să aibă ca scop final chiar tipărirea pe hârtie (la edituri, instituţii, etc.); astfel că au devenit necesare instrumente de conversie de la reprezentarea specifică documentelor Web (bazată pe limbaje de marcare, în principal HTML şi XML), la "formatul A4".

Pentru reprezentarea pe hârtie au fost produse două categorii de instrumente: aşa numitele "procesoare de text" WYSIWYG, de exemplu aplicaţia Word din pachetul comercial Microsoft-Office, sau aplicaţia necomercială OpenOffice.org-Writer; respectiv, limbaje de marcare/programare (TeX, cu varianta LaTeX) precum şi formate de reprezentare corespunzătoare, "independente de dispozitiv" (deci şi pentru pagina de hârtie), precum PDF.

Instituţiile (funcţionarii acestora) folosesc de regulă produsele de tip *point-and-click*, în principal Word; în general, ele nu produc documente *interconectate*, ci doar documente tipărite (folosind Web-ul doar pentru *transmiterea* fişierelor .DOC către instituţiile subordonate sau către beneficiari - urmând ca fişierele descărcate să fie tipărite local).

În cadrul sistemului de învăţământ se studiază în mod extensiv, produsele *point-and-click* (şi anume, exact acelea din varianta *comercială*, de la Microsoft). Motivul ar fi *uşurinţa* cu care se poate învăţa să faci *click* pe diversele meniuri şi iconiţe, vizualizând *imediat* efectul (în principal, efectul de *formatare* asupra porţiunii selectate). Văzând această preferinţă exclusivă pentru *point-and-click*, poate fi îndreptăţită bănuiala că scopul învăţământului ar fi acela de a forma funcţionari. Adevărul este însă acesta:

- este mult mai uşor de folosit un editor de text *obişnuit*, decât un "editor Word"; în orice caz, programatorii folosesc editoare de text (nicidecum, Word) atât pentru scrierea programelor, cât şi pentru realizarea unor documente oricât de complexe;

- ca să foloseşti un editor de text *obişnuit* nu ai nevoie de o *licenţă* costisitoare, precum în cazul folosirii Microsoft-Office;
- editoarele de text *obişnuit* sunt independente de platformă (sau au versiuni pentru Windows, Linux, Mac, etc.);
- dacă ai învăţat să foloseşti un *limbaj de marcare* (de exemplu, HTML), atunci va fi uşor să înveţi şi *alte* limbaje de marcare (CSS, XML; LaTeX);
- cunoscând două-trei limbaje de marcare, poţi folosi un editor de text *obişnuit* pentru a realiza *orice* tip de document Web.

Desigur, însuşirea unui limbaj de marcare poate să nu fie la fel de uşoară precum învăţarea mecanismului *point-and-click*, mai ales dacă ai neglijat experienţa lucrului cu fişiere text şi cu linia de comandă şi te-ai lăsat păcălit de "valenţele" acestui mecanism. Marele avantaj al mecanismului *point-and-click* este faptul că vezi imediat efectul fiecărei operaţii de editare; păcăleala vine de la obişnuinţa pe care o capeţi de a opera pe bucăţele, scăpând din vedere aspectele globale de structură şi de formatare.

În cazul folosirii limbajelor de marcare, nu mai poţi vizualiza *imediat* efectul. Mai întâi trebuie să descrii în fişierul text respectiv, structura şi caracteristicile globale ale documentului (folosind marcajele şi comenzile specifice acelui limbaj de marcare); apoi, trebuie să apelezi la un compilator sau la un program care poate interpreta marcajele respective şi care fie produce ca rezultat un alt fişier text care va putea fi vizualizat cu o anumită aplicaţie existentă, fie chiar vizualizează documentul respectiv.

De exemplu, pentru a realiza un document HTML folosind un editor de text obişnuit, de obicei trebuie să completezi întâi secţiunea <head> precizând diverse caracteristici globale (de exemplu, *content*="text/html; charset=utf-8"; stilurile de bază folosite; bibliotecile sau funcţiile angajate; etc.); apoi desigur, se adaugă conţinutul propriu-zis, sub tag-ul <body>; vizualizarea se poate realiza prin încărcarea sursei respective într-un browser. De asemenea, pentru a realiza un document folosind LaTeX trebuie specificate de la bun început caracteristicile globale şi pachetele de stiluri folosite; fişierul text respectiv trebuie compilat (apelând *latex*, sau *pdflatex*, etc.) obţinând un fişier text care poate fi vizualizat cu anumite aplicaţii ("Acrobate Reader", etc.).

Prin urmare, pentru a folosi limbaje de marcare eşti obligat să deprinzi obiceiul (care şi este foarte firesc) de a prevedea din start structura documentului şi cerinţele principale de formatare pentru secţiuni, sau pentru diverse categorii de elemente de conţinut. Este adevărat însă că poţi şi să ajungi la *degradarea* limbajelor de marcare, lucrând ca şi în Word (doar că, în loc să faci click-uri, scrii un simbol de marcare); de asemenea, în principiu se poate opera şi în Word în maniera specifică limbajelor de marcare (optând de la început pentru un anumit *şablon* de document).

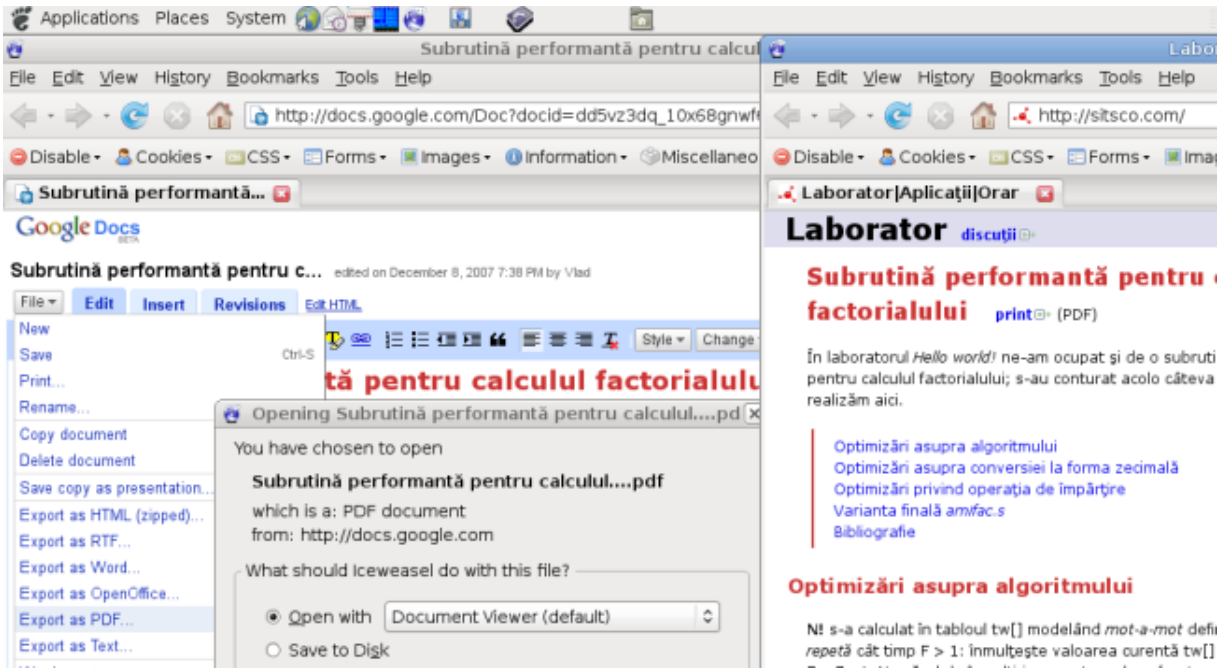
Limbajele de marcare (mai ales TeX) se bazează pe câteva aspecte care sunt binecunoscute de către oricine a răsfoit mai cu atenţie vreo două cărţi obişnuite şi vreo două "articole ştiinţifice": un document are un *titlu* şi un *autor*; are apoi, o mică "introducere" sau un *sumar* (sau "abstract", sau poate o "prefaţă"); are una sau mai multe secţiuni sau capitole, are un "cuprins", poate să aibă un glosar sau un index de termeni, are o "bibliografie"; fiecare secţiune sau subsecţiune are un titlu şi este formată din *paragrafe*, eventual *tabele*, *figuri*, sau imagini. Un limbaj de marcare instituie anumite secvenţe de caractere (denumite *tag*-uri, sau *simboluri de marcare*) cu ajutorul cărora se pot marca elementele de conţinut specificate mai sus; de obicei se prevăd şi diverse simboluri pentru eventuala formatare *ad-hoc* a unei secvenţe cât de mici, a documentului.

Documentul —fişier-text, nimic altceva— marcat astfel este "consumat" de obicei de un browser (Internet Explorer, Firefox, etc.) care ştie să interpreteze marcajele respective, redând documentul sub forma indicată prin marcajele fixate în text de către autor. Unele limbaje de marcare (de exemplu, LaTeX) sunt de fapt, chiar limbaje *de programare* (textul sursă trebuind să fie compilat) şi oferă mari facilităţi: constituirea automată a cuprinsului, generarea indecşilor, gestionarea referinţelor la pagini, tabele, la note de subsol sau la bibliografie, facilităţi de împărţire automată a unor porţiuni de text pe două sau mai multe coloane, şi altele; oferă de asemenea, marcaje corespunzătoare pentru scrierea de formule matematice şi pot produce în final (în funcţie desigur, de ce a prevăzut autorul) documente de înaltă calitate tipografică.

Transformarea documentelor cu Google Docs

Ştim desigur, că un fişier .DOC poate fi exportat în format HTML direct de sub Word, utilizând opţiunea "Save As..." din meniul "File". De obicei fişierul HTML rezultat arată destul de rău (sursa HTML obţinută trebuie "curăţată" de tag-urile inutile şi eventual trebuie efectuate multe "reparaţii" asupra unor tag-uri şi atribute care astăzi sunt declarate "deprecated"); de vină poate fi autorul .DOC-ului, fiindcă nu s-a preocupat câtuşi de puţin de structurarea coerentă a documentului şi nu a gândit unitar asupra formatării; dar chiar dacă s-ar proceda unitar, de exemplu formatând din capul locului toate paragrafele la fel - HTML-ul rezultat tot va conţine atributele (aceleaşi pentru toate paragrafele) în cadrul fiecărui tag de paragraf...

Pentru editarea documentelor şi pentru transformarea unui document dintr-un format în altul - şi tot pe bază de *point-and-click* - Google a pus la dispoziţie tuturor aplicaţia Docs. Descriem aici o modalitate de folosire — anume, ne propunem să obţinem formatul PDF pentru un articol existent pe `http://sitsco.com/`.



Am accesat *http://sitsco.com/* şi apoi (într-o a doua instanţă Firefox) *http://docs.google.com/*; am selectat cu mouse-ul textul articolului şi l-am copiat cu CTRL-C; în Google Docs am făcut click pe *New* şi am făcut "paste" (prin CTRL-V), iar ca urmare am obţinut articolul respectiv în fereastra de editare oferită; apoi, am deschis meniul *File* şi am făcut click pe opţiunea *Export as PDF...*, obţinând fereastra de dialog prin care se poate opta pentru deschiderea documentului PDF cu *Document Viewer*, sau pentru salvarea lui pe disk

Desigur, înainte de a exporta ca PDF, ar fi fost nevoie de unele retuşuri în fereastra de editare oferită — de exemplu trebuie eliminate secvenţele "^ TOP ^". Rezultatul PDF se obţine foarte uşor şi este de bună calitate; dar poate fi şi nemulţumitor, de vreme ce nu ai posibilitatea de a opta pentru un anumit format (de exemplu pagină cu două coloane, sau format de "articol ştiinţific"). Google Docs este adresat desigur utilizatorilor obişnuiţi; programatorul ar vrea mai degrabă un fişier text intermediar (cu marcajele corespunzătoare) pe care să-l poată eventual modifica, înainte de a genera PDF-ul.

Utilizarea modulului HTML::Latex

Avem nevoie de un program prin care tag-urile HTML să fie înlocuite corespunzător prin simboluri de marcare LaTeX; de exemplu, *<h2>Titlul secţiunii</h2>* să fie înlocuit prin *\section*{Titlul secţiunii}* (sau de exemplu, prin *\chapter{Titlul secţiunii}* - după cum dorim, sau în funcţie de genul de document LaTeX care se potriveşte). Documentul HTML iniţial poate fi reprezentat printr-un arbore (sau reţea) de obiecte de memorie, în care fiecare nod corespunde unui element HTML (înregistrând cel puţin tag-ul respectiv, lista atributelor din acel element, un pointer la "parent"-ul elementului şi unul la primul "child"); având acest arbore, rămâne ca el să fie traversat (recursiv) pentru a aplica fiecărui nod funcţia de înlocuire.

Pe CPAN am găsit un singur modul, **HTML::Latex**, destinat să creeze un fişier LaTeX dintr-unul HTML; se foloseşte HTML::TreeBuilder pentru a obţine arborele de memorie corespunzător documentului HTML (fiecare nod este un obiect HTML::Element) şi XML::Simple pentru a reconstrui arborele corespunzător noilor marcaje (şi desigur, acea funcţie care în asociere cu diverse alte subrutine, asigură corespondenţa dintre tag-uri HTML şi marcaje LaTeX). Modulul respectiv este cam vechi (din anul 2000) şi cu siguranţă că el s-ar putea rescrie acum mai bine (cel puţin, în privinţa consideării "caracterelor internaţionale"); folosim funcţionalitatea *minimă* oferită de el, intenţionând (şi preferând deocamdată) să operăm manual modificările dorite sau necesare asupra fişierului LaTeX rezultat.

```
#!/usr/bin/perl -w
# ./html-latex.pl htmllatex.html htmllatex.tex
use HTML::Latex;
my ($fi, $fo) = @ARGV; # preia de pe linia de comandă, numele celor două fişiere
my ($in, $out);
open $in, ':encoding(utf8)', $fo or die $!;
my $parser = new HTML::Latex();  # obiect capabil să analizeze HTML (din $in) şi
$parser->html2latex($in, $out);  # să "traducă" în LaTeX (în $out)
```

Pe acest site avem <meta http-equiv="Content-Type" content="text/html; *charset=utf-8*"> şi desigur, putem să includem această specificaţie şi în <head>-ul fişierului pe care-l transformăm; dar HTML::Latex *nu* ţine cont de specificaţiile din <head>-ul fişierului (deşi unele dintre acestea vizează chiar conţinutul propriu-zis şi deci nu pot fi considerate "colaterale"; este drept însă că există posibilitatea "legării" unor funcţii proprii, care să rezolve diverse aspecte particulare).

De aceea am specificat ***utf8*** pentru deschiderea celor două fişiere — în absenţa acestor specificaţii, caracterele româneşti "dispar"; mai precis, un "caracter internaţional" este reprezentat prin 2-6 octeţi de memorie care sunt interpretaţi la redare ca fiind un *singur* caracter *numai* în prezenţa anumitor specificaţii de codare (cum ar fi pentru browser, "Encoding: UTF-8") — altfel, sunt redate eventual "caracterele" aferente celor 2-6 octeţi constituienţi şi nu caracterul reprezentat unitar de aceştia.

Elemente conjuncturale de limbaj LaTeX

Redăm porţiunile semnificative din fişierul *htmllatex.tex* obţinut prin executarea programului redat mai sus şi completat apoi manual (în *preambul*—partea dinaintea *\begin{document}*; am adăugat un "abstract"); secvenţele marcate prin '%' sunt "comentarii" (şi vor fi ignorate de compilator).

\documentclass[10pt,a4paper]{article}
\usepackage{fullpage, url, ucs, graphicx} *% pachete LaTeX*
\usepackage[utf8x]{inputenc}
\renewcommand{\abstractname}{ } *% elimină denumirea implicită "Abstract"*
\setlength{\parskip}{1ex} *% spaţiul dintre paragrafe*
\setlength{\parindent}{0ex}
\graphicspath{{/home/vb/lar/root/static/}} *% path-ul pentru imagini (.PNG)*

\begin{document}

```
\title{de la HTML la PDF (cu Perl şi LaTeX)}
\author{Vlad Bazon, \texttt{http://sitsco.com}}
\date{Dec 2007}
\maketitle  % comanda de includere a titlului

\begin{abstract}
Obţinem varianta tipărită a unui document HTML, folosind modulul ...
\begin{itemize}  % enumerare (similar cu <ul> şi <li> din HTML)
\item Din nou, despre \emph{documente}
\item Transformarea documentelor cu \emph{\url{http://docs.google.com/}}
...
\end{itemize}
\end{abstract}

\section*{Din nou, despre \emph{documente}}
\textbf{document} provine din latinescul ...

\^In principiu, Web-ul este o infinitate de documente ...
...

\section*{ Transformarea documentelor cu  \includegraphics[scale=0.6]{docslogo.png} }
Ştim desigur, că un fişier .DOC poate fi exportat \^{\i}n format HTML:.
...

\begin{center}\includegraphics[scale=0.7]{sitdocs.png}\end{center}

Am accesat \emph{\url{http://sitsco.com/}} şi apoi (\^{\i}ntr-o ...
...

\section*{ Utilizarea modulului HTML::Latex}
Avem nevoie de un program prin care tag-urile HTML să fie \^{\i}nlocuite...
...

\begin{verbatim}  % similar cu &lt;pre&gt; din HTML
#!/usr/bin/perl -w

# ./html-latex.pl htmllatex.html htmllatex.tex

use HTML::Latex;
...
\end{verbatim}

...

\end{document}
```

Preambulul conţine comenzi care afectează întregul document. Comanda *documentclass[options]*{*class*} precizează compilatorului ce tip de document alege să creeze autorul (în cazul de faţă - document din clasa "article", de tipărit pe hârtie cu formatul A4, cu dimensiunea de bază a fontului de 10 puncte). După executarea comenzilor din preambul (cu efectul setării corespunzătoare a diverşilor parametri globali), compilatorul va identifica textul documentului (conţinutul propriu-zis) pe baza marcajelor *begin*{*document*} şi *end*{*document*} şi va aplica asupra lui comenzile (şi marcajele de formatare) întâlnite[1].

LaTeX prevede mai multe clase de document, de exemplu: *article* pentru articole destinate revistelor ştiinţifice, pentru rapoarte de mică întindere, pentru documentaţii de programe, etc.; *report* pentru cărţi

[1] asemenea aserţiuni reducţioniste nu trebuie luate "mot-a-mot"...; de obicei lucrurile sunt mult mai complicate

mici sau rapoarte care conţin mai multe capitole, pentru teze de doctorat; *book* pentru cărţi; *slides* pentru prezentări (analog celor produse cu PowerPoint); etc.

Pentru fiecare tip de document sunt prevăzute mai multe nivele imbricate de secţionare a documentului; de exemplu, un *article* poate cuprinde \part{Numele-Părţii}, \section{Nume-Secţiune}, \subsection{...}, \subsubsection{...}, \paragraph{...}, \subparagraph{...}. Secţiunile (la fel, tabelele, figurile, notele de subsol) sunt numerotate automat, într-o manieră care ţine cont de ierarhia lor; titlurile de secţiuni pot fi adăugate automat într-un tabel de cuprins.

Imaginile pot fi incluse în document prevăzând în preambul includerea pachetului *graphicx* şi folosind apoi comanda *includegraphics[opţiuni]{cale-nume-fişier-imagine*}; anumite conversii sunt efectuate automat (de la GIF la PNG); *graphicx* oferă numeroase comenzi pentru integrarea imaginii, de exemplu scalare, "clip", rotire, "wrap", alipire de imagini, etc.

Diverse alte pachete LaTeX permit gestionarea aspectelor privitoare la tabele (*tabular*, *table*, etc.), figuri, redarea secvenţelor de program sau de "pseudocod" (*listings*, *algorithm*, etc.), împărţirea textului în coloane (*multicol*, etc.), folosirea culorilor, folosirea de formule matematice oricât de complexe (*math*, *displaymath*, *amsmath*, etc.), folosirea caracterelor internaţionale (*ucs*, *inputenc*), etc.

Pentru a obţine direct un document PDF corespunzător fişierului htmllatex.tex creat, putem apela din linia de comandă compilatorul **pdflatex** (vb@debian: $ pdflatex htmllatex.tex). Eventual, comanda trebuie executată de mai multe ori consecutiv, dacă există referiri (citări) la note de subsol, la bibliografie, la figuri, etc. Fişierul PDF obţinut astfel ocupă mai mult spaţiu decât fişierul DVI ("Device independent file format") care s-ar fi obţinut prin compilare cu **latex**, dar include toate fonturile necesare şi de asemenea, imaginile (DVI doar le referă, urmând ca fişierele cu imagini să fie "incluse" doar în momentul tipăririi documentului); în plus, *pdflatex* recunoaşte formatele de imagine JPG şi PNG, în timp ce *latex* recunoaşte numai formatul "Encapsulated PostScript" (EPS).

Fişierul PDF ("Portable Document Format") poate fi tipărit ca atare (în general, imprimanta va "şti" să producă documentul conform indicaţiilor şi comenzilor existente în fişier), sau poate fi vizualizat (şi tipărit) folosind aplicaţii precum Acrobat Reader, Xpdf, Evince, etc.

Subrutină perl pentru *download*

Articolele de pe acest site sunt fişiere HTML, dar nu sunt "documente HTML" în sensul standard; ele *nu* conţin partea "obligatorie" de <head>, fiind în principal doar o succesiune de -uri (aferente secţiunilor din articol) şi de paragrafe (desigur şi de alte elemente HTML, precum <h2> folosit pentru titlurile secţiunilor, <a> pentru diverse referinţe interne sau externe, <img> etc.). Am putut "evita" să includ câte o parte de <head> în fiecare articol, prin existenţa unei "pagini de bază" *index.html* al cărei <head> prevede toate cele necesare ("content-type", fişiere de stiluri, biblioteci *javascript* de inclus, etc.) şi de asemenea, prin implicarea mecanismelor *Ajax* prin care fişierul respectiv este transferat (la click pe link-ul corespunzător, din cadrul paginii de bază) şi este redat de browser în locul indicat în pagina de bază pentru aceasta. Cu alte cuvinte, articolele respective sunt *integrate la cerere* paginii de bază şi *nu* sunt accesibile (sau redate) în mod independent (dezavantajul major fiind acela că articolele nu sunt "văzute" de motoarele de căutare; iar soluţia acestei probleme pare a fi una singură...).

Aşa stând lucrurile, pare dificil şi să copiezi (altfel decât ca text neformatat) şi să printezi vreunul dintre articole (mai ales dacă e vorba să faci aceasta de sub Windows şi cu Internet Explorer; altfel, am arătat mai sus că se poate foarte uşor cu Google Docs). În loc de a renunţa la Ajax (soluţie nedorită, dar... de loc rea) - sau poate şi mai bine... de a abandona totul - ne-am pus această chestiunea practică: a asocia unora dintre articolele de aici o "variantă PDF". Am arătat mai sus pe larg, demersurile şi instrumentele necesare pentru aceasta (plus diverse conexiuni *specifice...*) şi ar rămâne acum de arătat cum anume producem "download/save" pentru PDF-ul respectiv.

Ca interfaţă, am adăugat link-ul "print" (alături de titlul articolului) care (la *click*) accesează în mod explicit serverul pentru a executa funcţia necesară (transmiţându-i ca parametru numele fişierului PDF care trebuie returnat). Funcţia respectivă este încorporată în modulul Controller::Qnote:

```
package lar::Controller::Qnote;
use strict;
use warnings;
use base 'Catalyst::Controller';  # vezi http://catalyst.perl.org (şi [5])

# alte module incluse, alte subrutine...

sub down_exe : Local {
  my ($self, $c, $fin) = @_;       # $fin preia numele fişierului de downloadat
  my $fi = $c->config->{home}.'/root/'.$fin;
  my $size = (stat($fi))[7];       # dimensiunea fişierului
  open(INX, $fi) || die "nu se poate accesa $fi, $!";
  local $/;
  my ($data) = <INX>;
  close(INX);
  $c->res->content_length($size);  # transmite întâi header-ele necesare
  $c->res->headers->header('Content-Type' => 'application/pdf');
  $c->res->headers->header('Content-Disposition' => "attachement;filename=$fin");
  $c->res->body($data);            # transmite fişierul respectiv
}
```

Până să poată fi considerat ca "acceptabil" pentru a fi oferit spre *download*, PDF-ul respectiv necesită de obicei, unele retuşuri; de exemplu pentru cazul articolului de faţă, a apărut o problemă chiar nebanală: textul din secţiunea anterioară care descrie structura de principiu a unui fişier LaTeX, conţine *comenzi* LaTeX şi trebuie încorporat şi el în fişierul "htmllatex.tex" — se pune problema de a *evita* ca aceste comenzi să fie şi executate de către *pdflatex*; iniţial, am inclus textul respectiv sub "environment"-ul *verbatim*, care are o comportare asemănătoare cu <pre> din HTML (mai precis, programul *html-latex.pl* care folosea HTML::Latex a transformat <pre> în *verbatim*, dar fără să "escapeze" *corect* comenzile Latex din interiorul elementului <pre>). Numai că în cadrul *verbatim* poate apărea orice secvenţă de caractere (şi va fi redată ca atare), cu excepţia oricărei *comenzi* LaTeX (adică nu este permis să apară de exemplu "\begin{document}"); soluţia este desigur "escaparea" caracterului special "\", prin "\textbackslash{}" (deci comenzile din cadrul *verbatim* trebuie scrise sub forma: \textbackslash{}begin{document}).

Bibliografie

[1] `http://en.wikibooks.org/wiki/LaTeX` guide to the LaTeX
(From Wikibooks, the open-content textbooks collection)

[2] `http://www.latex-project.org/`
LaTeX – A document preparation system (LaTeX project site)

[3] `http://ctan.org/`
CTAN (Comprehensive TeX Archive Network)

[4] `http://cpan.org/`
CPAN (Comprehensive Perl Archive Network)

[5] Vlad Bazon - *Încorporarea unui editor WYSIWYG într-o pagină Web*
Gazeta de Informatică, mai 2007