Zpravodaj Československého sdružení uživatelů TeXu

Michal Hoftich

Publikování z LaTeXu na web pomocí TeX4ht

Zpravodaj Československého sdružení uživatelů TeXu, Vol. 28 (2018), No. 1-4, 11–21

Persistent URL: http://dml.cz/dmlcz/150101

Publikování z LaTeXu na web pomocí TeX4ht

Michal Hoftich

Publikování dokumentů vytvářených pomocí LaTeXu jako webových stránek je poměrně komplikovaný úkol. Představíme si konverzní systém *TeX4ht*, který tento proces umožňuje, a sestavovací program `make4ht`, který ho řídí.

Klíčová slova: TeX4ht, HTML

Úvod

Tento článek navazuje na předchozí příspěvek autora ve *Zpravodaji* [1], který byl zaměřený na problematiku tvorby elektronických knih, ovšem byly v něm také představeny postupy využívané konverzním systémem *TeX4ht* pro konverzi z TeXu do výstupních formátů a nastíněna problematika samotné konverze z TeXu do formátů založených na značkovacím jazyce XML. Nyní se zaměříme na některé nové vlastnosti sestavovacího programu pro *TeX4ht* jménem `make4ht` a pokročilejší možnosti konfigurace podoby výstupních souborů.

Připomeňme si však některé základní informace. *TeX4ht* je konverzní systém který je založený na vkládání instrukcí výstupního formátu přímo při překladu dokumentu TeXem. Výsledkem kompilace je DVI soubor, který je následně zpracován příkazem `tex4ht`. Ten vytvoří výsledné XML soubory a speciální soubory s příponami *.idv* a *.lg*. První z nich obsahuje DVI kód, ze kterého mají být vytvořeny obrázky (v základním nastavení například matematická prostředí). Druhý je řídící soubor pro poslední příkaz, `t4ht`. Ten je využíván pro konverzi obrázků z *.idv* souboru, volání externích příkazů, nebo vytvoření hlavního CSS souboru.

V předchozím odstavci jsme použili výraz *TeX4ht* ve dvou různých významech. Jednak jako název celého konverzního systému, jednak jako jednu z částí tohoto systému, která zpracovává DVI soubor. Existuje ještě další součást systému s tímto názvem, balíček TeXových maker `tex4ht.sty`, který zajišťuje vkládání XML značek do výstupního dokumentu.

Sestavovací program `make4ht`

Jak do tohoto schématu zapadá `make4ht`? Protože se celý konverzní proces skládá z několika na sebe navazujících kroků, využívají se pro kompilaci systémem *TeX4ht* skripty, které tento proces ulehčují. Existuje jich celá řada, liší se v podporovaném výstupním formátu a použitém TeXovém enginu a formátu. Nejznámější z nich

je *htlatex*, který využívá engine pdfTEX s formátem LATEX a produkuje výstup ve formátu HTML. Skripty zajišťují načtení balíčku s makry `tex4ht.sty` bez nutnosti jeho použití v konvertovaném dokumentu.

Tyto skripty jsou ovšem neflexibilní, každé jejich spuštění provede trojnásobnou kompilaci dokumentu TEXem. To zajistí správnou strukturu hypertextových odkazů a tabulek, ty totiž vyžadují několikanásobnou kompilaci pro svojí správnou funkci. V následujících krocích je dokument zpracován `tex4ht` a `t4ht`. Pokud dokument obsahuje například seznam literatury nebo zkratek, které jsou vytvářeny externími programy, je třeba zavolat nejdříve `htlatex`, poté požadovaný program a poté opět `htlatex`. V případě větších dokumentů může být čas potřebný pro kompilaci tímto způsobem poměrně dlouhý.

`make4ht` umožňuje tvorbu sestavovacích skriptů v jazyce *Lua*, ve kterých lze volat externí příkazy podle potřeby. S pomocí takzvaných *módů* lze ovlivnit pořadí kompilace pomocí přepínačů přímo z příkazové řádky. Například základní skript používaný `make4ht` podporuje *draft* mód, který spustí pouze jednu kompilaci dokumentu, místo obvyklých tří. Toho se dá využít pro zrychlení kompilace v případě, kdy se v dokumentu od poslední kompilace nezměnily křížové odkazy a pouze chceme získat rychlý náhled na jeho současnou podobu.

Původně byl `make4ht` součástí systému *TeX4ebook*, kde zajišťoval podporu pro sestavovací skripty, postupně se z něj však oddělil a v současné době plně nahradil staré skripty používané pro konverzi systémem *TeX4ht*. Kromě podpory pro sestavovací skripty také umožňuje snadný výběr TEXového engine použitého pro kompilaci, volbu výstupního formátu, podporu pro kódování Unicode ve výstupních souborech a další vlastnosti.

V současné době je podporován pouze LATEX, podpora pro PlainTEX je komplikovanější a ConTEXt podporován není. V následujícím textu se zaměříme pouze na LATEX.

Volby `make4ht` pro příkazovou řádku

`make4ht` podporuje řadu přepínačů a voleb, které ovlivňují průběh kompilace a zpracování výstupních souborů. `make4ht` může být spuštěn tímto způsobem:

```
make4ht [přepínače pro make4ht] soubor.tex "volby pro tex4ht.sty" \
"přepínače pro tex4ht" "přepínače pro t4ht" "přepínače pro TeX"
```

Tento komplikovaný způsob vychází z fungování `htlatex` a ostatních starých skriptů, které řešily nutnost předávání voleb pro všechny komponenty zapojené v kompilaci. Ve většině případů naštěstí není třeba této možnosti využívat, většinu vlastností, které poskytují `tex4ht` a `t4ht` lze vyžádat pomocí přepínačů pro `make4ht`.

Všechny přepínače, které `make4ht` podporuje, mají krátkou a dlouhou verzi. Krátké přepínače lze navíc spojovat dohromady. Následující dva příkazy jsou totožné:

```
make4ht --lua --utf8 --mode draft clanek.tex
make4ht -lum draft clanek.tex
```

Tento příkaz využije LuaLaTeX pro kompilaci, která proběhne pouze jednou díky využití *draft* módu, a výsledný dokument bude v textovém kódování UTF-8. Volby pro přepínače jsou od nich odděleny mezerou.

Kromě přepínačů `--lua`, `--utf8` a `--mode` existuje ještě řada dalších užitečných přepínačů:

`--config (-c)` konfigurační soubor pro *TeX4ht*. Umožňuje uživatelsky změnit značky vkládané do výstupních souborů.

`--build-file (-e)` sestavovací soubor.

`--output-dir (-d)` adresář, do kterého budou zkopírovány výstupní soubory.

`--shell-escape (-s)` použije volbu `--shell-escape` pro LaTeX, bude tedy možné spouštět z něj externí příkazy.

`--xetex (-x)` dokument bude kompilován pomocí XeLaTeXu.

`--format (-f)` volba výstupního formátu a rozšíření.

Existují ještě další přepínače, ale tyto jsou nejužitečnější pro běžné využití.

Výstupní formáty a rozšíření

TeX4ht podporuje širokou škálu formátů založených na XML, od XHTML přes ODT až po DocBook a TEI. Zájem uživatelů je však zaměřen prakticky pouze na varianty HTML a OpenDocument Format, který je podporován textovými procesory jako je LibreOffice nebo MS Word.

Přepínač `--format` tedy podporuje pouze následující formáty: html5, xhtml a odt (je třeba využít malá písmena). Přednastaveným formátem je HTML5. Ostatní formáty, jako je DocBook, lze vybrat pomocí volby pro `tex4ht.sty`:

```
make4ht filename.tex "docbook"
```

V přepínači `--format` lze také načíst rozšíření. Rozšíření umožňují ovlivnit kompilaci bez nutnosti použít sestavovací skript. Seznam využitých rozšíření lze zapsat za název formátu, zapínají se pomocí znaku plus, pomocí znaku mínus lze rozšíření zakázat[1]. Následující příkaz využije příkaz `HTML Tidy` pro opravení některých běžných chyb ve vygenerovaném HTML souboru:

```
make4ht -f html5+tidy simple-example.tex
```

Dostupná jsou následující rozšíření:

`latexmk_build` využije příkaz `latexmk` pro sestavení dokumentu. Ten se postará o volání externích příkazů, například pro tvorbu seznamu literatury.

`tidy` vyčistí HTML soubor pomocí příkazu `tidy`.

`dvisvgm_hashes` efektivní generování obrázků pomocí příkazu `dvisvgm`. Dokáže využít více procesorových jader a vytváří pouze obrázky, které byly změněny

[1] Rozšíření lze zapnout i ze sestavovacího skriptu, takové rozšíření je poté možné vypnout z příkazové řádky.

nebo vytvořeny od poslední kompilace. Tím dochází ke znatelnému zrychlení kompilace.

common_filters a **common_domfilters** vyčistění dokumentu pomocí filtrů. Filtrům se budeme věnovat dále v textu.

mathjaxnode konverze matematického kódu ve formátu MathML do speciálního HTML za využití projektu *MathJax Node Page*[2]. Díky tomu lze zobrazit kvalitně matematiku i ve webových prohlížečích bez podpory MathML, bez nutnosti využít JavaScript.

staticsite vytvoří dokument použitelný pro generátory statických stránek, jako je například *Jekyll*[3]. Ty jsou využitelné například pro tvorbu blogu.

Rozšíření je možné dále konfigurovat. Tím se dostáváme ke konfiguračnímu souboru a sestavovacím skriptům.

Konfigurační soubor pro make4ht

make4ht podporuje sestavovací skripty v jazyce Lua. Jejich pomocí je možné volat externí příkazy v průběhu kompilace, předávat jim parametry, používat filtry na výstupní soubory, ovlivňovat konverzi obrázků nebo konfigurovat rozšíření.

Konfigurační soubor .make4ht je speciální sestavovací skript, který je načítaný automaticky a měl by obsahovat pouze obecné konfigurace. Oproti tomu normální sestavovací soubory mohou obsahovat konfigurace platné pro daný dokument. Konfigurační soubor může být umístěný v adresáři s dokumentem, nebo v jeho nadřazených adresářích. To je užitečné například pokud bychom tvořili blog, kde každý dokument je umístěný ve vlastním adresáři. V nadřazeném adresáři může být umístěný konfigurační soubor, který zajistí správné zpracování. Zde je drobný příklad:

```
filter_settings "staticsite" {
  site_root = "output"
}

Make:enable_extension("common_domfilters")
if mode=="publish" then
  Make:enable_extension("staticsite")
  Make:htlatex {}
end
```

Tento konfigurační soubor nastavuje volbu site_root pro rozšíření *staticsite* pomocí příkazu filter_settings. Pomocí tohoto příkazu lze nastavit volby pro filtry, ale také rozšíření. Název filtru nebo rozšíření je oddělený od příkazu mezerou, poté následuje další mezerou oddělené pole, ve kterém lze nastavit volby

[2] https://github.com/pkra/mathjax-node-page/

[3] https://jekyllrb.com/

daného filtru. Oproti běžným konvencím příkazů v jazyce Lua zde nevyužíváme uvozovky jako u běžných funkcí!

Dalším použitým příkazem je `Make:enable_extension`, který povolí rozšíření. V tomto případě je rozšíření `common_domfilters` použito pro každou kompilaci, ovšem rozšíření `staticsite` pouze v módu *publish*, čehož je dosaženo podmínkou `mode=="publish"`.

Příkaz `Make:htlatex {}` vynutí jednu kompilaci LaTeXem.

Nyní lze spustit `make4ht` v módu *publish*:

```
make4ht -um publish simple-example.tex
```

Bude vytvořen adresář *publish*, pokud již neexistuje, do kterého budou zkopírovány HTML a CSS soubory ve formátu *datum publikace-originální název*. Statické generátory většinou očekávají názvy souborů v tomto formátu, čímž se zajistí chronologické řazení stránek.

Výsledný HTML soubor může mít následující podobu:

```
---
time: 1544811015
date: '2018-12-14 18:10:47'
title: 'sample'
styles:
- '2018-12-14-simple-example.css'
meta:
- charset: 'utf-8'
---
<p>Sample document</p>
```

Hlavička dokumentu uzavřená mezi dvojicí `---` obsahuje proměnné ve formátu YAML extrahované z HTML souboru, ze kterého zůstal pouze základní text. Statický generátor poté může vytvořit stránku na základě šablony a proměnných v hlavičce.

Toto byl pouze základní příklad, filtry a rozšíření mají mnohem širší možnosti nastavení, všechny volby jsou popsány v dokumentaci `make4ht` [2].

Sestavovací skripty

V sestavovacích skriptech můžeme využívat stejné postupy jako v konfiguračním souboru, ale více zaměřené na konkrétní kompilovaný dokument. Příklad sestavovacího skriptu byl představen v předešlém článku [1, s. 95], ukážeme si zde tedy pouze novou vlastnost, kterou jsou DOM filtry. Ty využívají možností knihovny LuaXML [3], která umožňuje zpracovávat XML soubory pomocí rozhraní *Document Object Model (*DOM*)*. Díky tomu lze snadno procházet elementy, upravovat je, vytvářet nebo odstraňovat.

Využití DOM filtrů si ukážeme na příkladu určeném pro LuaLaTeX:

```
\documentclass{article}
\begin{document}
Test {\itshape háčků}
\end{document}
```

Kvůli známé chybě při zpracování DVI souboru příkazem `tex4ht` bude vytvořen HTML soubor, kde každý znak s diakritikou bude umístěn v samostatném elementu:

```
<!--l. 4--><p class="noindent" >Test <span
class="rm-lmri-10">h</span><span
class="rm-lmri-10">á</span><span
class="rm-lmri-10">čk</span><span
class="rm-lmri-10">ů</span> </p>
```

Následující sestavovací soubor toto napraví pomocí vestavěného DOM filtru *joincharacters*. Navíc změníme atribut *class* všech elementů `<p>` na hodnotu *mypar*, abychom si ukázali práci s DOM rozhraním:

```
local domfilter = require("make4ht-domfilter")

local function domsample(dom)
  -- následující příkaz projde
  -- všechny elementy <p>
  for _, par in ipairs(dom:query_selector("p")) do
    -- nastavit atribut "class"
    par:set_attribute("class", "mypar")
  end
  return dom
end

local process = domfilter({"joincharacters", domsample})
Make:match("html$", process)
```

Skript využívá standardní funkci jazyka Lua `require` k načtení knihovny `make4ht-domfilter`. To vytvoří funkci `domfilter`, která má jako parametr seznam DOM filtrů, které se mají vykonat. Každé volání funkce `domfilter` vytvoří další funkci, která může být využita jako parametr pro funkci `Make:match`. Parametry v poli mohou být buď název existujícího DOM filtru, nebo funkce definovaná v sestavovacím souboru. Funkce `process` bude spuštěna na každém souboru jehož název končí na `html`.

Výsledný HTML soubor nyní neobsahuje nadbytečné elementy `<span>` a element `<p>` má hodnotu atribut *class* nastavenu na *mypar*:

```
<!-- l. 3 --><p class='mypar'>
Test <span class='rm-lmri-10'>háčků</span>
</p>
```

Konfigurace TeX4ht

Od `make4ht` nyní přejděme ke konfiguraci *TeX4ht* jako takového. Značky výstupního formátu vkládané do dokumentu jsou plně konfigurovatelné pomocí několika mechanismů. Nejjednodušší je využití voleb balíčku `tex4ht.sty`, větší možnosti nastavení nabízí konfigurační soubor a nejpokročilejší pak *4ht* soubory.

Při kompilaci TEXového souboru pomocí `make4ht` nebo jiného kompilačního skriptu se ještě před načtením samotného dokumentu načte balíček `tex4ht.sty`. Volby balíčku jsou získány z argumentů kompilačního skriptu. Díky tomu není třeba vkládat balíček `tex4ht.sty` v samotném dokumentu.

Mechanismus načítání souborů je upraven tak, aby se každý načítaný soubor registroval do *TeX4ht*. Pro některé balíčky *TeX4ht* také obsahuje kód, který zabraňuje jejich načítání, nebo okamžitě předefinovává některá makra. To je nezbytné pro balíčky, které jsou s *TeX4ht* nekompatibilní, jako je například *Fontspec*.

Po vykonání *preamble* dokumentu se načítají konfigurační soubory pro balíčky detekované při jejím zpracování. Tyto soubory mají název `jméno souboru bez přípony + .4ht`. Jejich hlavní funkcí je vkládat konfigurovatelná makra, takzvané háčky, do příkazů poskytovaných balíčkem. Obecně je lepší makra nepředefinovávat, ale pouze záplatovat pomocí příkazů, které pro tento účel *TeX4ht* poskytuje.

Po zpracování konfiguračních souborů pro balíčky se načítají konfigurační soubory výstupního formátu. Ty definují obsah konfiguračních maker, háčků. Především se do nich vkládají značky výstupního formátu, ale mohou obsahovat libovolné příkazy. Je možná také podmíněná konfigurace na základě voleb balíčku `tex4ht.sty`. Každý konfigurační soubor výstupního formátu může testovat libovolnou volbu, z čehož vyplývá, že neexistuje jejich konečný seznam a pro každý formát existují jiné volby.

Volby balíčku `tex4ht.sty`

Protože se nedoporučuje vkládat balíček `tex4ht.sty` přímo do dokumentu, je možné předat mu volby jiným způsobem. Nejjednodušším je pomocí argumentu pro kompilační skript. Vždy je to argument následující po názvu dokumentu:

```
make4ht filename.tex "mathml,mathjax"
```

Volby použité v tomto případě jsou *mathml* a *mathjax*. Další možností předání voleb je předání pomocí příkazu `\Preamble` v soukromém konfiguračním souboru, který si ukážeme v další sekci.

Jak již bylo řečeno, neexistuje konečný seznam voleb, každý výstupní formát může podporovat jejich libovolné množství.

Můžeme si ovšem ukázat některé volby, které se týkají matematických výstupů v HTML. Normální konfigurace pro matematická prostředí produkuje směs for-

mátovaného textu a obrázků pro složitější výstupy, které nejdou snadno vytvořit pomocí HTML elementů. Někdy tato vlastnost nevypadá dobře. Jako alternativu lze použít obrázky pro veškerý matematický obsah. Toho lze dosáhnout pomocí voleb *pic-m* pro inline matematiku a *pic-název prostředí* pro matematická prostředí, například *pic-align*.

Normálně jsou obrázky vytvářeny ve formátu PNG. Vyšší kvality lze dosáhnout s využitím vektorového formátu SVG. Ten může být vynucen pomocí volby *svg*.

Dokumentace *TeX4ht* je poměrně spartánská. Obsáhlejší informace o konfiguraci příkazů lze nalézt v log souboru vytvořeném při kompilaci dokumentu s pomocí volby *info*.

Volby uvedené v prvním příkladu, *mathml* a *mathjax* poskytují obecně nejlepší výstup matematického obsahu. Značkovací jazyk MathML umožňuje zakódovat matematické informace, jeho podpora ve webových prohlížečích je však špatná. S využitím knihovny *MathJax*[4] je možné jej zobrazit ve všech moderních prohlížečích s podporou jazyka JavaScript.

Samotná volba *mathjax* bez *mathml* úplně vypne zpracování matematiky při kompilaci, veškerý matematický obsah zůstane v HTML dokumentu jako TEXová makra. *MathJax* poté dokument zpracuje a vykreslí matematiku správným způsobem. Nevýhodou tohoto způsobu je, že *MathJax* nepodporuje všechny balíčky a uživatelské příkazy.

Soukromý konfigurační soubor

Pomocí soukromého konfiguračního souboru je možné vkládat vlastní obsah do konfiguračních háčků. Tento soubor má zvláštní strukturu:

```
...Definice v~preamble...
\Preamble{volby pro tex4ht.sty}
... Normální konfigurace ...
\begin{document}
... Konfigurace pro hlavičku \HTML{} souboru
\EndPreamble
```

Příkazy `\Preamble`, `\begin{document}` a `\EndPreamble` musí být v konfiguračním souboru obsaženy. Cesta k souboru může být předána *make4ht* pomocí parametru `-c`:

```
make4ht -c myconfig.cfg filename.tex
```

Pokud konfigurační soubor není uložený v aktuálním pracovním adresáři, je nutno použít plnou cestu k souboru.

Existuje několik konfiguračních příkazů. Nejdůležitějším je `\Configure`, který nastavuje běžné konfigurace, dalšími jsou například `\ConfigureEnv` pro nastavení prostředí, nebo `\ConfigureList` pro seznamy.

[4] `https://www.mathjax.org/`

Příkaz `\HCode` se používá pro vkládání značek výstupního formátu. Součástí jeho argumentu může být příkaz `\Hnewline` pro vložení zalomení řádku. Instrukce `\Css` vloží kód pro kaskádové styly do CSS souboru.

Následující ukázka použije element `<em>` pro příkaz `\textit`:

```
\Configure{textit}{\HCode{<em>}\NoFonts}{\EndNoFonts\HCode{</em>}}
```

Příkaz `\Configure` podporuje variabilní počet argumentů, záleží na definici háčků, kolik argumentů je potřeba. Prvním argumentem je vždy název konfigurace, další argumenty poté vkládají kód do háčků. V typickém případě vyžaduje konfigurace dva háčky – jeden umístí kód před začátek příkazu, druhý za jeho konec. Tak je tomu v případě konfigurace pro `textit`. Název konfigurace se může shodovat s názvem konfigurovaného příkazu, jako v tomto případě, ale vždy záleží na konfiguračním *4ht* souboru pro daný balíček, jaký název použije.

Příkaz `\NoFonts` zakáže vkládání formátovacích elementů při zpracování DVI souboru. `tex4ht` vkládá automaticky tyto elementy při změně písma. Díky tomu je možné vytvořit základní formátování i pro nepodporované příkazy bez konfigurací, ovšem pro konfigurované příkazy je toto zbytečné.

Komplikovanou problematikou jsou odstavce. *TeX4ht* je někdy vkládá na nevhodná místa. Týká se to především konfigurací prostředí, která mohou obsahovat několik odstavců a celý svůj obsah umísťují do jednoho elementu. Může se stát, že počáteční značka odstavce je umístěna před začátkem tohoto elementu, správně by ovšem měla následovat až po něm. Pro tento případ existují příkazy `\IgnorePar`, který zabrání vložení značky pro následující odstavec, a `\EndP`, který vloží zavírací značku pro předešlý odstavec. Existuje více příkazů pro práci s odstavci, ale tyto jsou nejdůležitější.

V následujícím příkladu použijeme hypotetické prostředí `rightaligned`, které by mělo být umístěno v elementu `<article>`.

```
\ConfigureEnv{rightaligned}
{\HCode{<section class="right">}}
{\HCode{</section>}}{}{}
```

Příkaz `\ConfigureEnv` očekává pět parametrů, prvním je název konfigurovaného prostředí, druhým je kód vložený na začátku prostředí a třetím kód vložený na jeho konci. Zbylé dva argumenty se používají pouze pokud je konfigurované prostředí založeno na seznamu, ve většině případů mohou zůstat prázdné. HTML kód vytvořený touto konfigurací může vypadat následovně:

```
<p class="indent" >   <section class="right">
...
</p><p class="indent></section>
```

Tento kód je nevalidní, protože ukončovací značka pro element `<p>` je umístěna na špatné úrovni zanoření. Nevalidní kód může způsobit přerušení běhu DOM filtrů a jiných nástrojů, které zpracovávají výsledné XML soubory. Této situaci je třeba předcházet.

Správná konfigurace je poněkud komplikovanější:

```
\ConfigureEnv{rightaligned}
{\ifvmode\IgnorePar\fi\EndP\HCode{<section class="right">}\par}
{\ifvmode\IgnorePar\fi\EndP\HCode{</section>}}{}{}
```

V tomto případě řídíme vkládání značek pro odstavce sami a výsledek je v pořádku:

```
<section class="right">
<!--l. 9--><p class="indent" >
...
</p></section>
```

V konfiguracích můžeme také vyžádat konverzi části dokumentu na obrázek. Toho se dá docílit pomocí příkazů `\Picture*`, respektive `\Picture+`. Rozdíl mezi nimi je ten, že první zpracovává svůj obsah jako vertikální box. V každém případě se obsah mezi těmito příkazy a ukončovacím příkazem `\EndPicture` převede na obrázek.

Těchto příkazů lze využít například pro konverzi složitější matematiky nebo diagramů, ale čistě technicky podporují veškerý obsah, který dokáže zpracovat použitý konvertor z DVI do obrazových souborů, typicky *Dvipng* nebo *Dvisvgm*.

Následující příklad vytvoří obrázek pro text obsažený v prostředí *topicture*:

```
\documentclass{article}
\newenvironment{topicture}{\bfseries}{}
\begin{document}
\begin{topicture}
Obsah tohoto prostředí by měl být zobrazen jako obrázek
\end{topicture}
\end{document}
```

Konfigurace pro prostředí *topicture* využije příkazu `\Picture*`:

```
\ConfigureEnv{topicture}{\Picture*{}}{\EndPicture}{}{}
```

Závěr

Možnosti konfigurace *TeX4ht* jsou rozsáhlé, v předešlém textu jsme se dotkli pouze základů, které by však měly vystačit pro řešení nejběžnějších otázek, které uživatelé tohoto systému řeší. Vynechali jsme také ukázky, jakým způsobem lze přidat konfiguraci pro nový LaTeXový balíček. Tato témata by vydala na rozsáhlé samostatné články.

Díky sestavovacímu systému `make4ht` je použití celého systému snazší a efektivnější než tomu bylo v minulosti.

V současné době probíhá za finanční podpory poskytnuté *Československým sdružením uživatelů TeXu* tvorba nové dokumentace pro *TeX4ht*, kde budou

témata tohoto článku probrána do větší hloubky, spolu s konkrétními příklady užití.

Odkazy

[1] HOFTICH, Michal. Elektronické knihy a systém TeX4ebook. *Zpravodaj Československého sdružení uživatelů TEXu*. 2016, roč. 26, č. 1–4, s. 94–105. Dostupné také z: `http://bulletin.cstug.cz/doi.php/10.5300/2016-1-4/94`.

[2] HOFTICH, Michal. *The Make4ht package: A build system for tex4ht* [online]. 2018. Verze 0.2c [cit. 2018-12-14]. Dostupné z: `http://www.ctan.org/pkg/make4ht`.

[3] HOFTICH, Michal. *The Luaxml package: Lua library for reading and serialising XML files* [online]. 2018. 0.1g [cit. 2018-12-14]. Dostupné z: `http://www.ctan.org/pkg/luaxml`.

Summary: LaTeX to Web Publishing using TeX4ht

The article gives overview of the current state of development of TeX4ht, LaTeX to XML convertor. It introduces `make4ht`, a build system for TeX4ht as well as basic ways how to configure TeX4ht.

Keywords: TeX4ht, html

Michal Hoftich