MASARYKOVA UNIVERZITA V BRNĚ
FAKULTA INFORMATIKY

Konverze technických textů z MathML do Braille

DIPLOMOVÁ PRÁCE

Josef Cacek

Brno, 2004

Prohlášení

Prohlašuji, že tato diplomová práce je mým původním autorským dílem, které jsem vypracoval samostatně. Všechny zdroje, prameny a literaturu, které jsem při vypracování používal nebo z nich čerpal, v práci řádně cituji s uvedením úplného odkazu na příslušný zdroj.

Vedoucí práce: RNDr. Petr Sojka

Poděkování

Děkuji vedoucímu své diplomové práce RNDr. Sojkovi za připomínky, odkazy a zdroje informací, které mi poskytl. Dále děkuji pracovníkům střediska Teiresiás za ochotu, se kterou mi přiblížili techniky převodu textů do Braille.

Shrnutí

Matematické zápisy pomocí značkovacího jazyka MathML jsou stále rozšířenější. Převážně jsou součástí XHTML dokumentů – webových stránek. Tyto dokumenty lze transformovat do dalších formátů, mimo jiné i do Braillova písma. Jeden z možných přístupů ukazuje aplikace M2B, která je výsledkem této práce.

Klíčová slova

Braille, BraMaNet, Java, MathML, M2B, XSL

Note légale

Masarykova Universita v Brně
Fakulta Informatiky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Datum: 8. července 2002

Student: Josef Cacek, 4. r. odborná informatika
Vedoucí DP: RNDr. Petr Sojka
Katedra: progr. syst. a komunikací
Specializace: Informační systémy
Garant spec.: prof. RNDr. Jaroslav Král, DrSc.

Název: Konverze technických textů z MathML do Braille

Zadání:

- Seznamte se s technikami tisku publikací v Braillově písmu z dostupné literatury (Neméth 1972), a verzemi Braillova písma; (`http://www.tsbvi.edu/math/math-resources.htm`, `http://www.braille.org/papers/unive/unive.html`, `http://www.w3.org/TR/1998/REC-MathML-19980407/chapter1.html`).
- Seznamte se s technikami konverze technických textů do Braille. Hlavní pozornost věnujte konverzi z XML, MathML (`http://www.w3.org/TR/MathML2/`) a dokončeným publikačním projektům technických textů střediska TEIRESIAS.
- Seznamte se s předchozími relevantními projekty MAVIS `http://www.NMSU.Edu/~mavis/`, TRIANGLE `http://dots.physics.orst.edu`, MEGAMATH, SWIFT `http://www.duxburysystems.com/`, bakalářskou prací pana Tesaře (2000) a diplomovou prací pana Suchého (2002).
- Seznamte se s XSL transformacemi a navrhněte nástroje pro převod MathML dokumentů do Braille.
- Zkušenosti získané v předchozích bodech zobecněte a na jejich základě naprogramujte transformace pro konverzi technických textů do Braille. Vytvořte webové rozhraní pro použití tohoto nástroje.

Základní literatura:

Požadovaná forma realizace: Vlastní textu DP alespoň 40 stran a konvertoru technických textů do Braille včetně průvodní dokumentace.

student, datum	vedoucí DP	vedoucí katedry	garant spec.

Obsah

Préface

Psaný text má v lidské komunikaci nezastupitelné místo. Spolu s mluvenou řečí je nejhodnotnější složkou komunikace. Oproti mluvenému slovu, jehož hlavní bariérou je existence mnoha jazyků a nářečí, přináší psaný text mnoho dalších problémů. Jedním z nich jsou různé formáty psaného textu. Vezměme v úvahu například jen různé druhy písem (latinka, azbuka, japonské písmo, různé verze čínského písma, atp.) a vedle problému překladu textů z jednoho jazyka do jiného se vynoří problém převodu zápisů.

S nástupem počítačů a textů přenášených v elektronické podobě se objevují další bariéry (text jako bitmapa, text jako posloupnost znaků, značkovaný text, kardinality abeced, různá kódování pro jeden jazyk), velkým přínosem však je, že je lze pomocí softwarových nástrojů překonávat. Tato diplomová práce by měla naznačit jeden z možných způsobů.

Nezanedbatelná část lidské populace (asi 0,5 %) žije s takovým zrakovým postižením, které omezuje začlenění do společnosti, ztěžuje přístup k informacím a vytváří pro postiženého speciální klima. Z těchto důvodů vznikají iniciativy, které si kladou za cíl vytvořit takové prostředí, které by nevidomým poskytlo stejné možnosti, jako mají vidoucí. Část této snahy je zaměřena na převod textů do a z Braillova písma.

Stále více informací z různých oborů je prezentováno na internetu pomocí webových technologií. Nutnost práce se specifickými tvary textů pro daný obor dala vzniknout novým standardům a obecně uznávaným doporučením pro publikování informací. Jedním z nich je i doporučení pro publikování technických a zvláště matematických textů MathML.

Tato diplomová práce ukazuje jeden z možných přístupů ke konverzi textů z formátu MathML do zápisu v Braille.

Kapitola 1

Formáty psaných textů

Písmo je jedním z největších lidských vynálezů. Písemné zprávy se zaznamenávaly na pevné materiály (kameny, kosti zvířat, hliněné tabulky, ...) a v současné době se využívá nových fyzikálních poznatků k uchování zpráv na paměťových médiích použitelných v počítačích.

Ve vývojovém procesu existují tyto základní kategorie písem:

1. obrázková písma (piktografická);
2. písma slovní (ideografická – hieroglyfy) – označují morfémy;
3. písma slabičná (např. japonské) – označují slabiky;
4. písma hlásková (latinka, azbuka) – označují jednotlivé hlásky;
5. speciální písma (Braille, Morseův kód, ...).

Jak už bylo naznačeno v úvodu diplomové práce, psaný text s sebou přináší řadu úskalí. Jedním z nich je existence mnoha různých forem zápisu. Základní kritéria, podle kterých můžeme dělit formáty psaného textu v počítačích, jsou:

- mohutnost použité abecedy – počet různých znaků, které mohou být v zápisu použity;
- značkování textu – značky v textu jsou posloupnosti symbolů, které části textu přiřazují nějakou vlastnost (např. doporučený typ fontu použitý při tisku), text bez značek se nazývá čistý text (plain text);
- použití víceznakých symbolů – bývá hlavně u abeced s menší kardinalitou; určité znaky pak jsou vyhrazeny jako prefixy pro změnu významu následujících znaků, čímž rozšíříme původní abecedu o další „víceznaké“ symboly.

1.1 Braillovo písmo a jeho formáty

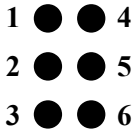
Písma pro nevidomé, mezi která patří i Braille, jsou zpravidla vnímána hmatem – takzvaná taktilní písma. Krátký pohled do historie jistě pomůže vytvořit si obrázek o problematice, která je s těmito písmy spojená.

Již od starověku se setkáváme se snahou zprostředkovat psané texty i pro nevidomé. Většinou je však tato snaha zaměřena na jednoho nevidomého nebo pouze na úzký okruh jedinců. Až v 17. století se objevuje návrh obecného písma, který vychází z jednoduchého kryptografického systému Francesca Terzy [smykal-94]. V 18. a 19. století jsou hledány nové cesty pro zprostředkování psaného textu nevidomým. Jsou zde patrné dvě hlavní linie – jedna zastává názor, že by slepci měli používat klasickou

latinku, která by byla vyvedena reliéfně v nějakém tuhém materiálu, a která by byla čitelná i vidoucím; druhý směr se přikláněl k názoru, že by měla vzniknout zvláštní reliéfní abeceda, která by byla co nejvíce přizpůsobena potřebám nevidomých.

Na počátku 19. století byla v pařížském ústavu pro nevidomé uspořádána studentská soutěž, jejímž cílem bylo vytvořit systém pro zápis not pro slepce. Nejúspěšnější prací byl návrh tehdy šestnáctiletého Louise Brailla, který svou formou (6 bodů uspořádaných do dvou sloupců) odpovídal anatomii bříška ukazováku a umožňoval snadné lineární čtení. Na konci století byl tento systém uznán jako standard a začaly ho přejímat i jiné jazyky než francouzština. Vývoj Braille pokračuje dodnes.

Protože počet znaků v Braillově abecedě je velice omezený (2^6=64 znaků), jednotlivé jazyky si vytvářely místní úpravy. Vznikaly také úpravy v rámci vědních oborů (matematika, fyzika, chemie, hudba, . . .), které jsou většinou založeny na systému prefixů. Například čísla se zapisují tak, že uvedeme znak, říkající že následuje číslo, a poté píšeme znaky abecedy (a – j).



Obrázek 1.1: Pro vyjádření znaku v Braille se používá číslování vytištěných bodů

1.1.1 Jazykové verze Braille

Louis Braille při svém návrhu slepeckého písma vycházel ze zkušeností s bodovým písmem Ch. Barbiera, ve kterém pro zápis jednoho znaku byly použity dvě svislé řady po šesti bodech. Tento zápis nepracoval s jednotlivými znaky francouzské abecedy, ale s jejich fonetickým vyjádřením. Braille do svislé řady umístil pouze 3 body tak, aby bylo možné celý znak pojmout bříškem ukazováku. Přestal také pracovat s fonetickým vyjádřením a použil znaky klasické abecedy (využívaje prefixů např. pro velká písmena), což později umožnilo snadné použití i v jiných jazycích.

Ačkoliv bylo Braillovo písmo popsáno již v roce 1827, jeho obecné přijetí na sebe nechalo ještě dlouho čekat. Na vině bylo hlavně jeho odmítání vidoucími učiteli, kteří tvrdili, že speciální písmo nevidomé ještě více izoluje. I přes tyto obtíže bylo začátkem 20. století ve Francii schváleno. Přednosti tohoto písma samozřejmě zaujaly nevidomé a jejich učitele i v jiných zemích a zajistily jeho rozšíření po světě.

Pro český jazyk se začal Braille využívat někdy mezi lety 1870 a 1890. Nezávisle na sobě ho přijali a začali upravovat tři učitelé – J. Malý (Praha), V. Novák (Praha) a J. Schwarz (Brno). Soustavy se od sebe lišily například v pojetí diakritiky. Na počátku 20. století je snaha tyto soustavy sjednotit. Definitivní forma, která platí dodnes, byla schválena roku 1922 tehdejším ministerstvem školství.

Zajímavý je vývoj Braille mezi asijskými jazyky. Ve druhé polovině 19. století se místní varianta Braille objevuje v Indii a v návaznosti na něj i ve středočínské jazykové oblasti (písmo se skládá z dvoj- a víceznaků). V polovině 20. století se začal rozšiřovat tzv. mandarinský Braille vytvořený na principu hlásek, který obsahuje 9000 znaků. To se může zdát jako krok zpět (viz Ch. Barbier), ale do čínské kultury, v níž běžné písmo nemá hláskovou podobu a vyjadřuje ucelené myšlenky, tento typ písma zapadá. V roce 1990 se v Číně konal kongres učitelů nevidomých, který rozhodl o sjednocení Braillova

písma pro každou ze tří jazykových oblastí Číny zvlášť. V Japonsku byl vytvořen hláskový systém Braille (tzv. Katakana), který se používá hlavně pro přepis cizích slov.

1.1.2 Braille podle oborů

Původní určení Braille bylo pro zápis not, ale vzhledem ke svým vlastnostem se uplatnil i v dalších odvětvích. Díky struktuře tohoto písma (systém prefixů) je možné ho rozšiřovat o další znaky a významy. Například základy matematiky byly zpracovány již v původním návrhu, ale s potřebou zapisovat vyšší matematiku logicky přichází i požadavek na rozšíření Braille, a tak v polovině 20. století vzniká Nemethovo kódování matematiky v Braille. Podobně je tomu i v ostatních vědních oborech; vznikají zápisy pro fyziku, chemii, geografii,. . .

Jazykové verze Braille tato kódování sice přebírají, ale upravují je pro dané jazykové prostředí. Tím se sice zvyšuje snadnost začlenění speciálního kódu do tohoto prostředí, ale zároveň tím vzniká bariéra pro použití materiálů v cizích zemích. V České republice byl v roce 1995 schválen předběžný návrh normy českých národních znakových sad Braillova slepeckého písma předložený PhDr. Rudolfem Volejníkem, Jindřichem Hegrem a RNDr. Wandou Gonzúrovou. Norma obsahuje doporučení pro zápis matematiky, fyziky a chemie.

V poslední době je snaha sjednotit zápisy alespoň na širší regionální úrovni. V roce 1997 byly organizací Braille Authority of North America schváleny standardy Braille Music Code a Braille Code for Chemical Notation.

Vývoj kódování Braille je neustálým procesem. Nově vznikající a rychle se vyvíjející odvětví (jako například informační technologie) si vynucují nové zápisy v Braille nebo změny ve stávajících. Jak ukáže následující kapitola, vývoj není jen věcí nových oborů, ale i oborů relativně ustálených, jako je matematika. Je těžké posoudit, zda současný vývoj bude směřovat ke sjednocení zápisů, nebo zda se diference mezi normami budou stále zvětšovat.

1.2 Normy pro zápis matematiky v Braille

Největším problémem při převodu z černotisku do Braille a naopak je existence mnoha verzí Braille, a to často i v rámci jednoho jazyka. Následující část textu je zaměřena na seznámení se základními formáty zápisu matematiky v Braille.

Původní verze, tak jak ji navrhl Louis Braille, zde není zmíněna. Diskutovány jsou zápisy, které umožňují pracovat s vyšší matematikou.

1.2.1 Nemethovo kódování

Nemethovo kódování je prvním rozšířeným zápisem pro matematiku v Braille. Vytvořil ho slepý Dr. Abraham Nemeth v roce 1946, když si při studiu potřeboval zapisovat matematické výrazy. Kód byl publikován a začal se používat. Schválila ho organizace Braille Authority of North America (BANA), která v něm provedla několik úprav. Poslední revize je z roku 1972.

Oproti anglickému literárnímu Braille jsou zde provedeny úpravy, které mohou znesnadnit překlad textu do černotisku. Například číslice v anglickém Braille je psána pomocí číselného prefixu následovaného písmenem (a – j), v Nemethově kódování je zápis písmene v Braille snížen o jednu řádku bodů.

Obrázek 1.2: Číslo 17 v anglickém Braille

Obrázek 1.3: Číslo 17 v Nemethově kódování

Nemethovo kódování je v současnosti asi nejrozšířenější formou zápisu matematiky v Braille. Z tohoto důvodu většina nástrojů pro převod černotisku do Braille pracuje právě s touto normou. Hlavní nevýhodou kódu je obtížný převod zpět do černotisku.

1.2.2 Kódování matematiky v českém Braille

Česká verze Braille pro zápis matematiky vychází z Nemethova kódování. Až donedávna se zápisy různých autorů lišily, proto byly navrženy úpravy, které zvyšují přesnost a jednoznačnost zápisů. Poslední schválené změny byly provedeny v roce 1995; jejich autory jsou PhDr. Rudolf Volejník, Jindřich Hegr a RNDr. Wanda Gonzúrová.

Online verzi příručky pro přepis textů do bodového písma lze najít na adrese `http://www.fi.muni.cz/~xgabrhel/projekt/`. V této příručce jsou kromě pravidel pro přepis matematiky uvedena i převodní pravidla pro fyziku a chemii. Znaky v Braille jsou zde reprezentovány bitmapovými obrázky, takže čtenář ihned vidí výsledný tvar. Bohužel u obrázků nejsou vyplněny atributy `alt`, a proto tato verze není použitelná pro čtení na textových terminálech nebo pomocí screenreaderu (software, který pomocí syntézy řeči čte, co je na obrazovce počítače).

Protože česká norma pro zápis matematiky nemůže obsáhnout celý obor, stále existují rozdíly v zápise (at' už v rámci lokalit či jednotlivých autorů). Oproti větším národům máme nevýhodu, že zde není tolik prostředků na vývoj nástrojů pro převod a také chybí dostatek institucí, které by se problémem zabývaly.

1.2.3 ASCII Mathematik Schrift (AMS)

`http://elvis.inf.tu-dresden.de/publichtml/asc2html/ams/h-000000.htm`

Na univerzitě v Karlsruhe byl navržen modelový systém pro zpřístupnění počítačových oborů nevidomým a slabozrakým studentům vysokých škol. Středisko pro nevidomé na této univerzitě a pracovní skupina „studia pro nevidomé a slabozraké“ technické univerzity v Drážďanech revidovali tento systém a vznikl nový kód pro zápis matematiky ASCII Mathematik Schrift. Na obou těchto univerzitách začaly vznikat studijní materiály ve formě „přátelské“ pro zrakově postižené. Přátelská forma znamená, že poskytované texty a diagramy jsou čitelné pro nevidomé, zrakově postižené i studenty bez zrakového postižení.

Reprezentace matematických symbolů v počítači je většinou pomocí grafických symbolů, které nejsou čitelné pro nevidomé. Proto jsou v AMS zápisu reprezentovány skupinou znaků z ASCII tabulky, používané v počítačích jako základní sada. Vzniklé

texty jsou tedy systémově nezávislé a mohou být čteny a editovány obyčejným textovým editorem.

AMS si klade za cíl nejen vyjádřit syntaktickou stránku, ale i zlepšit sémantické vyjádření matematického výrazu. AMS také umožňuje používat při psaní obecně užívané zkratky. Těmito kroky se zvyšuje čitelnost zápisu.

Kód AMS má podobnou syntaxi jako programovací jazyky. Použitím znaků z ASCII tabulky je umožněn bezproblémový 1:1 převod do černotisku a naopak. Například integrál $\int_a^b f(x)\,dx$ má v AMS zápisu následující tvar:

```
Int[a;b] f(x) dx
```

V Německu je zápis pomocí AMS poměrně rozšířený, ale jednotnost zde opět chybí. Oproti jiným kódům je AMS nezávislý – nevyjadřuje grafickou formu, ale sémantiku výrazu. Nevýhodou zůstává fakt, že zápis pomocí ASCII znaků zvyšuje objem textu a pro zápis složitějšího vzorce je potřeba velkého množství závorek.

1.2.4 UBC a UEBC

http://www.iceb.org/ubc.html

V roce 1991 dva přední nevidomí vědci – Dr. Tim Cranmer a Dr. Abraham Nemeth – sepsali dopis adresovaný organizaci BANA, ve kterém poukazovali na obtíže nevidomých, kteří musí používat různé kódy Braille pro zápis literatury, matematiky, . . . Požadovali v něm, aby byl vyvinut jednotný systém zápisu Braille.

Požadavky sepsané v dopise byly pro organizaci BANA přesvědčivé, a proto vznikl projekt „Unified Braille Code Research Project“, jehož cílem bylo vyvinout šestibodový jednotný zápis, který by se co nejvíce podobal literárnímu kódu používanému v anglicky mluvících zemích.

Vznikající kódování dostalo označení Uniform Braille Code (UBC). Jeho úkolem bylo sjednotit zápisy Braille v různých oborech. Speciálně šlo o tyto (nechováme uvedeny anglické názvy kódů):

- The Literary Code – používaný v běžné literatuře. Tyto kódy jsou ve skutečnosti dva; jeden schválený organizací BANA v Severní Americe, druhý schválený organizací BAUK ve Velké Británii;
- The Braille Code for Textbook Formats and Techniques – používaný v učebnicích;
- The Nemeth Braille Code for Mathematics and Science Notation – pro zápis matematiky, fyziky a dalších technických textů;
- The Computer Braille Code – používaný v IT.

Každý z uvedených kódů se vyvíjel nezávisle na ostatních, což má za následek množství konfliktů v zápisech znaků a v pravidlech zápisu v daném kódu. Například znak „$“ nebo „%“ má v každém z nich jiný zápis. UBC zmíněné nedostatky odstraňuje a zlepšuje tak přehlednost zápisů v Braille.

V roce 1993 vývoj tohoto kódu zaštítila organizace International Council on English Braille (ICEB). O dva roky později byl vydán první návrh kódu a v roce 1999 byl změněn název na Unified English Braille Code (UEBC), aby se zdůraznil jeho vztah k anglicky mluvícím zemím.

Naší země se tato aktivita zatím netýká, ale budoucí vývoj může být přínosný i pro neanglicky mluvící země.

1.2.5 GS-kód

http://dots.physics.orst.edu/gs.html

V rámci projektu SAP vyvinuli profesoři John Gardner a Norberto Salinas duální 6/8 bodový kód pro kompaktní lineární zápis znakových informací (GS-kód). Kódování obsahuje tisíce znaků použitelných jak pro klasickou literaturu, tak i v publikacích pro matematiky, techniky atp.

Syntaxe kódování GS vychází z myšlenky UBC a je vystavěna na konstrukcích sázecího jazyka LaTeX, který je hojně využíván v univerzitním a vědeckém prostředí. Úroveň 2 (zkrácený kód) šestibodového GS6 kódu je podobná UBC kódu a tím samozřejmě i současnému anglickému literárnímu kódu. Nezkrácený kód je používán pro matematiku a od UBC se již výrazněji odlišuje.

Osmibodový kód GS8 je kompaktní verze GS6. Všechny symboly bez prefixů z GS6 se zobrazí stejně i v GS8, odlišnosti nastanou u víceznakých zápisů v GS6, ze kterých se v GS8 může stát jednoznaký zápis (využitím dodatečných 2 bodů). Jednoduchá transformační pravidla umožňují tomu kdo se naučí zápis GS6, bez problémů používat i GS8. Osmibodový kód je zejména vhodný při zobrazení informací z počítače na braillském řádku.

GS-kód se snaží co nejvíce věcí zachovat tak, jak jsou používány v anglické verzi Braille. Tím vzniká problém s použitím v jiných jazycích. Autoři uvádí, že řešením je použití kódu GS8, ale lze pochybovat o tom, že by se v tištěné literatuře přešlo na osmibodový systém zápisu.

Oproti AMS kódu je GS o něco více zaměřen na grafickou (prezentační) stránku matematického výrazu, což vyplývá z použití konstrukcí systému LaTeX. V zápisech, kde je nutné přesné vyjádření kontextu bychom proto doporučili spíše použití AMS.

1.3 Úvod do značkovacích jazyků

V mnoha textech je výhodné znát doplňující informace o struktuře daného textu – sémantický význam jednotlivých částí, formátování textu při tisku, informace pod čarou apod. Při zpracovávání textu člověkem nebo programem je výhodné užít ke strukturování značek vhodného značkovacího jazyka. Značky v textu jsou definované posloupnosti znaků, které se nesmí vyskytovat ve zpracovávaném textu jinak než právě v roli značky definující jednotlivé logické části textu. Souhrn pravidel pro doplňování značek do textu nazveme značkovacím jazykem. Samotný proces doplňování značek do textu někdy označujeme jako značkování textu [vochozka-00].

Na značkování může být nahlíženo z různých hledisek. Nejznámější rozdělení značek je na deklarativní a procedurální. Deklarativní popisuje logickou vlastnost části textu (např. být jménem) a procedurální popisuje akci, která se s danou částí textu provede (např. vytiskni tučným písmem). Příklad části textu označkované pomocí XML by mohl vypadat následovně:

```
<adresa>
  <jmeno>Josef Cacek</jmeno>
  <email><tiskniTucne>xcacek@fi.muni.cz</tiskniTucne></email>
</adresa>
```

V příkladu je použito procedurální (značka `tiskniTucne`) i deklarativní (`adresa`, `jmeno`, `email`) značení. Z příkladu je vidět, že značky mohou být také hierarchicky uspořádány. Při zvážení možností deklarativního a procedurálního značení dojdeme k závěru, že více informací nám podává značení deklarativní, které můžeme namapovat a softwarově převádět na procedurální.

Značkovací jazyky můžeme rozdělit na metajazyky a konkrétní implementace jazyků. Metajazyky neříkají, jaké značky budou kde použity, ale jakou má mít značkování pomocí těchto jazyků strukturu. Tvoří tedy nadmnožinu nad konkrétními implementacemi jazyků (takzvané rodiny jazyků). Příklady metajazyků jsou SGML a XML. Mezi konkrétní implementace patří HTML, MathML, značení v sázecím systému TEX a mnoho dalších.

Významnou roli v historii značkovacích jazyků hraje společnost World Wide Web Consortium (W3C). Byla založena v roce 1994 za účelem maximálního využití potenciálu webu vyvíjením veřejných protokolů, které mohou být dále rozšiřovány a které jsou schopné navzájem spolupracovat. Konzorcium W3C, které tvoří přibližně 450 členských organizací z celého světa, se zasloužilo o rozvoj webu a souvisejících technologií. Stránky konzorcia W3C jsou dostupné na adrese `http://www.w3.org/`.

1.3.1 SGML

Do šedesátých let minulého století můžeme datovat začátek vývoje značkovacích jazyků. Ve firmě IBM byl vyvinut jazyk GML (Generalized Markup Language), primárně určený pro značkování právních textů, který vytvořil základ pro jazyk SGML (Standard Generalized Markup Language). Ten byl v roce 1986 schválen organizací ISO jako mezinárodní standard (ISO 8879:1986).

SGML je rozsáhlý a přitom flexibilní, dává uživatelům volnost ve vyjadřování a v použití. Na úkor této volnosti je softwarové zpracování SGML – je velice složité vytvořit parser, který by pokrýval celý tento standard. Značkování publikací představuje hlavní oblast, pro kterou je SGML určen. Jeho použití podporuje stylový a transformační jazyk DSSSL.

Deklarativní jazyk DSSSL má velkou vyjadřovací schopnost, ale je netriviální a dosud existuje velmi málo procesorů zpracovávajících styly DSSSL (např. opensource projekt Jade). Omezení a nevýhody SGML a DSSSL řeší (za cenu větších omezení kladených na uživatele) odvozené jazyky, jako je XML.

1.3.2 XML

Extensible Markup Language (XML) je jednoduchý, flexibilní textový formát odvozený z SGML. Jeho vývoj začala v roce 1996 skupina „XML working group“ pod záštitou konzorcia W3C. Původně byl navržen pro publikační činnost, ale rozšířil se a uplatnil i jako formát pro výměnu dat mezi aplikacemi, na webu apod.

Zjednodušeně by se dal vznik XML z SGML přirovnat ke vzniku Javy z C++. Vývojová skupina odstranila nadbytečné, nepoužívané a matoucí části SGML. To, co zůstalo, byl čistý značkovací jazyk – XML. Cíle, kterými se vývoj XML řídí, jsou:

1. XML by mělo být jednoduše použitelné na internetu;
2. XML by mělo podporovat širokou škálu aplikací;
3. XML by mělo být kompatibilní s SGML;

4. mělo by být snadné psát programy zpracovávající XML dokumenty;

5. počet volitelných znaků (optional features) v XML by měl být minimální, v ideálním případě nulový;

6. návrh XML by měl být rychle připravitelný;

7. návrh XML by měl být formální a stručný;

8. XML dokumenty by měly být čitelné a dostatečně jasné;

9. XML dokumenty by měly být lehce vytvořitelné.

První verze doporučení XML (1.0) byla schválena v roce 1998 a v roce 2000 revidována. V době psaní této práce je očekáváno vydání verze 1.1.

1.3.3 (X)HTML

http://www.w3.org/MarkUp/

HyperText Markup Language (HTML) je základní jazyk pro publikování hypertextů na webu. Je založen na standardu SGML a je neustále vyvíjen a zdokonalován tak, aby reflektoval nové požadavky autorů webových stránek a tvůrců webových prohlížečů. Jazyk HTML je jedním ze základních produktů konzorcia W3C. Aktuální verze při psaní této práce je 4.01.

Jazyk HTML je sice podobný XML jazykům, ale nespadá do množiny pokryté XML, nemůže být tedy zpracováván stejnými nástroji. Autoři si tento problém uvědomili a přepracovali jazyk HTML do XML podoby. Nový jazyk má označení XHTML, v současné době je používán ve verzích 1.0 a 1.1.

1.3.4 MathML

http://www.w3.org/Math/

Mathematical Markup Language (MathML) je značkovací jazyk vytvořený konzorciem W3C k usnadnění použití matematických a vědeckých textů na webu. Byl vyvíjen s ohledem na možné použití v aplikacích jako jsou sázecí programy, programy řečové syntézy apod. MathML může uchovávat informaci jednak o grafické podobě matematického výrazu, a také o obsahu výrazu tam, kde klíčovou roli hraje sémantika (vědecký software, hlasová syntéza apod.).

MathML patří mezi XML jazyky, je tedy zpracovatelný běžnými nástroji pro práci s XML. Bližšímu pohledu na MathML se bude věnovat kapitola 3.

1.4 Převody XML dokumentů

Při zpracování XML dat nebo jejich přenosu mezi různými aplikacemi je často nutné změnit značkování v dokumentu, doplnit do dokumentu další prvky, nebo vybrat jen určité části v závislosti na značkování. K tomuto účelu vznikla řada transformačních jazyků a nástrojů pro práci s nimi.

1.4.1 CSS – kaskádové styly

Kaskádové styly (Cascading Style Sheets) jsou jazykem doplňujícím formátovací informace (barva textu, pozice na stránce, velikost a typ fontu, . . .) do HTML a XML dokumentů. Tvoří doplněk ke strukturování textu pomocí těchto značkovacích standardů. Kaskádové styly jsou vyvíjeny konzorciem W3C. První verze doporučení (CSS1) byla schválena v roce 1996. O dva roky později je schválena nová verze – CSS2. Umožňuje definovat pozici elementů na stránce a celkové rozvržení stránky, podporuje stahování chybějících fontů na klientský počítač a umožňuje formátování stránky pro tisk.

V současné době nemá žádný prohlížeč plnou podporu CSS2, ale některé (Opera 6, Mozilla) pokrývají většinu tohoto doporučení. Nová verze jazyka – CSS3 – se stále vyvíjí, měla by přinést podporu prezentačních médií (zvuková stylizace apod.). Další novinkou bude modulárnost CSS3.

CSS styl se skládá z jednotlivých pravidel, která mají tvar `Selector { seznam vlastností }`. Selector může být například název elementu v dokumentu. Seznam vlastností tvoří dvojice název vlastnosti a hodnota. Platnost pravidla není omezena na daný element, ale zabírá celý podstrom, na jehož kořen ukazuje Selector. Vlastnosti se pak mohou kaskádově rozšiřovat, každý podelement může být adresován dalším Selectorem a může přidávat nebo měnit vlastnosti. Následující příklad definuje barvu a zarovnání nadpisu (obsahu elementu <H1>...</H1>).

```
H1 {
  color: green;
  text-align: center;
}
```

Přínosem jazyka CSS je jeho jednoduchost a tedy i použitelnost mezi „neprogramátory“. Výhody použití formátování pomocí kaskádových stylů jsou především tyto:

- můžete změnit vzhled celého dokumentu pouhou změnou několika pravidel ve stylu;
- jeden styl může být použit pro celou řadu dokumentů;
- přesunutí formátovacích značek do stylu zvyšuje čitelnost dokumentu;
- dokumenty se díky redukovanému kódu stahují z internetu rychleji;
- pro různé klienty (například obrazovku, tiskárnu a hlasový výstup) mohou být použity různé styly.

Formátovací jazyk CSS se pro řešení této diplomové práce nehodí, protože neposkytuje možnosti širší práce s textem, není v něm možné přeuspořádávat elementy apod. Jazyk CSS se hodí spíše pro jednodušší práci s XML.

1.4.2 XSL transformace

XSL (The Extensible Stylesheet Language Family) je soubor doporučení z dílny konzorcia W3C pro transformaci a prezentaci XML dokumentů. Obsahuje doporučení pro

XML transformace (XSLT), navigaci v XML dokumentech (XPath) a slovník formátovacích objektů (XSL–FO) pro definici výstupního formátování dokumentu.

Tak jako je XML odvozeno z obecnějšího jazyka SGML, staví XSL svůj základ na standardu DSSSL, který poskytuje transformační i formátovací funkcionalitu. XSL přináší oproti DSSSL zjednodušení a snadnější implementaci nástrojů pro práci s tímto jazykem.

XSL na rozdíl od kaskádových stylů nabízí pokročilejší úroveň zpracování XML dokumentů. Výstupem XSL transformace může být úplně rozdílný dokument s jiným (případně žádným) značkováním. Navigace v XML, kterou pokrývá doporučení XPath, využívá i složitější konstrukce odkazů do XML stromu, jako jsou logické podmínky, aritmetické výrazy a navigace na základě znalosti jiné části dokumentu (předci, potomci, sousedé, . . .).

Následující příklad transformačního XSL stylu řeší sčítání v kontextovém značení MathML. Výstupem je textový soubor obsahující výsledek transformace.

```
<xsl:stylesheet version='2.0'
    xmlns:xsl='http://www.w3.org/1999/XSL/Transform'
    xmlns:mml='http://www.w3.org/1998/Math/MathML'>

  <xsl:output method='text' encoding='iso-8859-2'/>
  <xsl:strip-space elements='*'/>

  <xsl:template match='mml:apply[mml:plus]'>
    <xsl:for-each select="*[position() &gt; 1]">
      <xsl:apply-templates/>
      <xsl:if test="position() &lt; last()">
        <xsl:text>+</xsl:text>
      </xsl:if>
    </xsl:for-each>
  </xsl:template>

</xsl:stylesheet>
```

Elementy `xsl:output` a `xsl:strip-space` jsou konfigurační a říkají, že výstupem bude plain text a budou vynechány nadbytečné mezery. Základ stylu tvoří šablona – `xsl:template`. Atribut `match` pomocí XPath výrazu udává, pro které uzly je šablona určena (v našem případě se jedná o uzel `mml:apply`, který má přímého potomka `mml:plus`, vybíráme tedy aplikaci operace sčítání na operandy). Obsah elementu `xsl:for-each` se provede pro všechny uzly odpovídající výrazu v atributu `select` (vybíráme ty podelementy současného elementu – `mml:apply`, jejichž pozice je větší než jedna – první potomek je vždy operátor, tedy `mml:plus`).

Použitím `xsl:for-each` přestává být aktuálním uzlem `mml:apply`, aktuálními se stávají jeho potomci. Element `xsl:apply-templates` požaduje zpracování potomků aktuálního uzlu příslušnou šablonou. Protože v příkladu máme jen jednu šablonu, jsou uzly, které jí neodpovídají, zpracovány implicitní šablonou. Ta vytiskne textové uzly. Obsah elementu `xsl:if` se provádí v případě, že je splněna podmínka zapsaná v atributu `test`. V uvedeném příkladě je do textu vloženo znaménko '+', jestliže aktuální uzel není posledním potomkem elementu `mml:apply`. Jestliže tímto XSL stylem transformujeme následující XML dokument

```
<math xmlns="http://www.w3.org/1998/Math/MathML">
  <apply>
    <plus/>
    <ci>a</ci>
    <ci>b</ci>
    <ci>c</ci>
  </apply>
</math>
```

dostaneme jako výsledek očekávaný zápis

```
a+b+c
```

Technologie XSL tvoří základ této diplomové práce. V průběhu psaní se ukázalo, že je pro daný úkol nedostačující, a proto byly využity i další nástroje (Java, dom4j).

Kapitola 2

Současné nástroje pro převod textu do Braille

Ve světě existuje mnoho institucí zabývajících se převodem textů pro nevidomé. Velká část z nich vznikla v univerzitním prostředí nebo s univerzitami spolupracuje. Některé nadějné projekty ztroskotaly na přílišné obecnosti problému převodu textů, z toho vyplývající náročnosti řešení a nedostatku prostředků pro dokončení. Z tohoto důvodu jsou použitelné nástroje povětšinou komerční aplikace.

I nadějné aplikace, které jsou používány v jedné lokalitě, nemusí být (a většinou nejsou) díky nejednotnosti zápisů použitelné jinde. K vyřešení problému nepřispívá ani fakt, že další instituce vytváří nové normy pro zápis matematiky. I když tyto normy bývají přínosem, většinou se rozšíří pouze v lokalitě, kde daná instituce působí. Se vznikem normy vyvstává problém jak do tohoto kódování automaticky převádět publikace. Pak vznikají jednoúčelové nástroje nepoužitelné pro jiné druhy zápisu. V České Republice v dnešní době nejsou automatické nástroje, které by se daly bezproblémově využít pro převod matematických textů do Braille.

2.1 BraMaNet

http://handy.univ-lyon1.fr/projets/bramanet/

BraMaNet (Logiciel De Traduction Des Mathématiques En Braille) – projekt Frédérica Schwebela na Lyonské univerzitě – je nástroj pro převod z prezentační formy MathML do francouzského matematického Braille. Projekt pokrývá matematické zápisy až do úrovně univerzitní matematiky. Výstupem programu je textový soubor, který může být dále zpracováván systémy jako je Duxbury Translator nebo Braille Star (viz section 2.4 a section 2.5).

BraMaNet umožňuje pomocí entit určit, který znak na tiskárně reprezentuje znak v Braille. Hlavním nedostatkem je, že BraMaNet nepracuje s kontextovým značením MathML, které umožňuje přesnější vyjádření obsahu vzorce. Dalšími nedostatky jsou malá přizpůsobivost, práce pouze s francouzskou normou Braille pro zápis matematiky a částečná závislost na operačních systémech MS Windows.

Od března roku 2003 je BraMaNet vyvíjen pod licencí GNU GPL, díky čemuž je možné získat zdrojové soubory a upravit je například pro různé verze Braille.

2.2 The Science Access Project (SAP)

http://dots.physics.orst.edu/

Projekt SAP katedry fyziky Oregonské státní univerzity je zaměřen na vývoj metod pro zpřístupnění matematiky a technických informací lidem se zrakovým postižením. Pokrývá širokou oblast problémů od návrhu úprav Braille přes softwarové nástroje až po hardware pro nevidomé.

Softwarové nástroje WinTriangle a TRIANGLE (v prostředí MS-DOS) jsou aplikace pro zrakově postižené, kteří potřebují pracovat s matematikou. Používají GS-kód pro reprezentaci matematických výrazů a poskytují prostředky pro vyhodnocování a vykreslování výrazů. Grafový kalkulátor, který je součástí software, může poskytnout výsledky ve vizuálním, audio a hmatovém formátu. Tabulkový prohlížeč umožňuje uživatelům číst, procházet a editovat tabulky jednoduchým způsobem. Program ve verzi pro DOS je volně šiřitelný.

The Accessible Graphing Calculator je software pro platformy MS Windows. Má zabudovaný hlasový syntetizér, mimo běžné funkce kalkulátoru poskytuje možnost vykreslení grafu funkcí, a to i pomocí generování zvuku různých výšek.

The Tiger Tactile Graphics and Braille Embosser (TIGER) je komplexní systém (software i hardware) pro tisk z platforem Windows do Braille. Předností tohoto systému je kombinování textu a grafiky (bodové vyznačení hran, stupně tmavosti atp.).

Projekt SAP patří mezi úspěšné projekty, ale jak bylo řečeno v úvodu kapitoly, problémem je, že je podporován hlavně v jedné lokalitě – Severní Americe.

V současné době je v rámci projektu vyvíjen nástroj pro převod mezi MathML a formou Braille, kterou užívá software Triangle a WinTriangle. Bude určen pro platformu Windows a měl by být dokončen ve školním roce 2003/2004.

2.3 Mathematics Accessible to Visually Impaired Students (MAVIS)

`http://www.nmsu.edu/~mavis/`

MAVIS byl projekt univerzity v Novém Mexiku. Jeho cílem bylo vytvořit konvertor ze souborů pořízených programem Scientific Notebook, což je WYSIWYG editor LaTeXových dokumentů do Braille. Projekt byl po čtyřech letech vývoje v roce 2000 „zmrazen“ pro nedostatek finančních prostředků ve stadiu testování. Program konvertoval do Nemethova kódování [suchy-02].

Překlad z Braille do LaTeXu autoři projektu řešili použitím konstrukcí logického programování, přesněji DCG gramatik v Prologu. Braille je kontextový jazyk, ale MAVIS pracuje se zjednodušenou verzí Nemethova kódování, která patří mezi bezkontextové gramatiky (ale není to LALR(1) gramatika) [ICCHP-00]. Vývoj se dostal až do fáze α–testů, pracovalo se i na vývoji screen-readeru pro matematiku.

2.4 Duxbury systems

`http://www.duxburysystems.com/`

Duxbury systems je firma vyvíjející software usnadňující komunikaci s nevidomými – převod z Braille do černotisku a naopak. Jejich produkty vznikají pro různé platformy (Windows, Macintosh, DOS, UNIX). Vlajkovými produkty jsou Duxbury Braille Translator (DBT) a MegaDots. DBT podporuje více platforem, různé jazyky (a typy Braille). MegaDots je program pro MS-DOS, který je ze strany Duxbury stále podporován a s dodatečným modulem (MegaMath) zvládá i překlad matematického Braille v Nemethově kódování.

Duxbury jsou jednou z nejvýznamnějších firem dodávajících software pro převod mezi Braille a černotiskem. I když je jejich hlavní pole působnosti v Americe, snaží se prosadit i na ostatních kontinentech (podpora evropských a arabských jazyků, . . .). Firma poskytuje i několik freeware utilit pro zjednodušení komunikace v Braille, mimo jiné i sadu TrueType fontů pro vizuální zobrazení Braille.

2.5 Handy Tech – Braille Star

http://www.handytech.de/

Handy Tech Elektronik je německá firma vyvíjející technologie pro nevidomé. Hlavními produkty jsou výstupní zařízení, která provádí online převod informací zobrazených na monitoru počítače do zápisu v Braille (braillské řádky). Jejich technologie jsou použitelné v počítačích s operačními systémy MS Windows i Linux. Hlavními produkty firmy jsou:

- Modular Professional – modulární řešení pro pracovní stanice nevidomých; systém komponent poskytuje uživateli možnost zvolit si vstup–výstupní řešení, které mu vyhovuje;
- Braille Star 40 – braillský řádek vhodný k mobilnímu použití; obsahuje výkonnou logiku, která umožňuje práci se zařízením (čtení a úprava textů) i bez připojení k počítači;
- Braille Star 80 – braillský řádek s optimalizovanou ergonomií;
- Braillino – malý braillský záznamník (včetně 20-znakého braillského řádku), který je navržen pro použití s PDA zařízením Nokia Communicator.

Dalším zajímavým produktem je „Graphic Window Professional“, který pomocí 384 hmatových bodů zobrazuje grafické informace z obrazovky počítače. Použitelnost zvyšuje funkce transfokace (zoom).

2.6 Nástroje na Fakultě informatiky MU

http://www.fi.muni.cz/

Fakulta informatiky Masarykovy univerzity v Brně je jedním z českých center pro nevidomé. Středisko Teiresiás, které zde v roce 1999 vzniklo na pomoc nevidomým studentům fakulty, rozšířilo svou působnost na celou univerzitu. Zajišťuje studentům převod studijních materiálů do Braille a další služby související se studiem. Ze spolupráce FI a střediska Teiresiás vzešlo již několik projektů, které se snaží řešit převod mezi Braille a černotiskem.

2.6.1 DVI2Braille

http://nlp.fi.muni.cz/projekty/dvi2braille/

Bakalářská práce Jiřího Tesaře měla za cíl vytvořit ovladač, který dokument ve formátu DVI vytiskne jako dokument určený pro tisk na Braillově tiskárně. Převod probíhal ve čtyřech krocích:

1. převod z DVI na obyčejný text;
2. řešení řádkových zlomů;
3. označkování textu podle konvencí zápisu v Braille;
4. převod do znakové sady používané tiskárnou.

Největším problémem se ukázal být převod matematiky, neboť v DVI souborech se neudržují informace o sémantice, ale pouze o umístění znaků na stránce. Pak například nepoznáme, zda se jedná o zlomek nebo o podtržený text, pod nímž je další text.

Finální verze aplikace řeší převod čistého textu a jednoduchých tabulek do českého zápisu Braille.

2.6.2 TEX to Braille

http://nlp.fi.muni.cz/projekty/xsuchy/diplomka.pdf

Diplomová práce Miroslava Suchého řeší převod technických dokumentů do Braille. V práci jsou navrženy různé postupy převodu dokumentů. Součástí je také upravený sázecí systém Omega a skript, který by měl ulehčit převod z LATEXu do Braille.

Systém Omega je rozšířením TEXu, které vnitřně používá „široké znaky“ (vícebajtové sekvence pro znak). Je schopný zpracovávat texty v kódování podle standardu Unicode 3. Má propracovaný návrh vstupních a výstupních filtrů umožňujících uživateli pracovat s přepisovými schématy.

Nástroj se bohužel nepodařilo uvést do reálného provozu. Myšlenka převodu textů pomocí systému Omega je jistě užitečná, právě díky množství publikací napsaných v TEXu. Vzhledem k tomu, že je v TEXu vyjádřena pouze vizuální forma bez zachování sémantiky, nedá se převod plně automatizovat. Když se ale omezíme např. na makra z balíku LATEX, získáme lepší představu o sémantice a převodem můžeme dosáhnout dobrých výsledků.

2.6.3 Makra pro MS Word

Ondřej Nečas, který v době psaní této diplomové práce vykonává civilní službu ve středisku Teiresiás, naprogramoval sadu maker pro Microsoft Word usnadňující převod dokumentů z Wordu a WordPerfectu do Braille. V projektu zohlednil i práci s více jazyky (reálné využití je např. pro jazykové učebnice), speciální sadu maker vytvořil pro zápis matematiky.

Makra jsou ve středisku Teiresiás používána a zdokonalována. Jejich nasazení vede ke zjednodušení převodu a jeho menší časové náročnosti. Bylo by dobré, kdyby se další vývoj neomezoval pouze na Masarykovu univerzitu, ale kdyby byl tento nástroj vyvíjen širším kolektivem autorů z různých oborů. Doporučovali bychom také převést makra do některého volně šiřitelného kancelářského balíku (např. OpenOffice.org), aby i uživatelé jiných operačních systémů a nekomerčních řešení měli možnost využívat tento systém převodu.

Kapitola 3

MathML

Typickou vlastností matematiky je použití komplexního a vysoce vyvinutého systému dvoudimenzionálních symbolických zápisů. Matematika a její zápisy nemohou být nazírány jako jedna a ta samá věc. Matematické ideje existují nezávisle na zápisech, které je reprezentují. Nicméně relace mezi významem a zápisem jsou nepatrné a část matematické síly popsat a analyzovat je odvozena ze schopnosti reprezentovat ideje a pracovat s nimi pomocí symbolického zápisu [pierce-80].

Publikování matematiky na webu by mělo zachytit oba směry – zápis i obsah (tedy význam), a to takovým způsobem, že dokumenty mohou být snadno převedeny do různých matematických notací a také dále zpracovávány počítačovými programy. Značkovací jazyk MathML z rodiny XML se snaží tyto náročné požadavky úspěšně pokrýt.

3.1 Historie

Problém kódování matematiky pro počítačové zpracování a elektronickou komunikaci je mnohem starší než web. Obecnou praxí ve vědeckých kruzích byl zápis matematiky v některé formě kódování založené na znakové sadě ASCII. Některé značkovací jazyky pro matematiky, konkrétně například TEX, byly používány už v roce 1992, ještě před zlatou érou webu.

Již od svého vzniku demonstruje web efektivní metodu zpřístupnění informací širokým, různě zaměřeným skupinám lidí. Ačkoliv byl World Wide Web prvotně navržen a implementován vědci pro vědce, možnosti pro zápis matematických výrazů v HTML jsou velmi omezené. V současné době je většina dokumentů s matematickými zápisy na webu tvořena textem s obrázky matematických výrazů (v GIF nebo JPG formátu), které je obtížné číst a vytvářet, nebo jsou celé dokumenty ve formátu PDF.

Konzorcium W3C uznalo, že nedostatečná podpora vědecké komunikace může znamenat vážný problém. Dave Raggett zahrnul v roce 1994 návrh specifikace HTML Math v pracovní verzi specifikace HTML 3.0. V následujících letech se začala formovat skupina, která sdružovala mnoho zainteresovaných stran majících zájem na vytvoření matematického značkovacího jazyka pro web. V březnu roku 1997 byla tato skupina formálně přeorganizována v první „W3C Math Working Group“.

První verze jazyka MathML (1.0) byla uvedena v dubnu roku 1998. O rok později byly integrovány částečné změny ve specifikaci MathML 1.01. Další verze přišla v březnu 2001 – MathML 2.0 a v říjnu 2003 byly uvedeny další změny (MathML 2.0 Second Edition).

3.2 Technická dokumentace pomocí MathML

Specifikace MathML uvádí, že tento jazyk byl navržen tak, aby reflektoval následující cíle. MathML by měl:

- vhodně kódovat matematické materiály jak pro výuku, tak pro vědeckou komunikaci na všech úrovních;
- uchovávat matematický zápis i matematický význam;
- usnadnit převod (zápisu i významu) do a z jiných matematických formátů. Mezi výstupní formáty by měly patřit:
  - grafické displeje;
  - řečové syntetizéry;
  - vstupy pro systémy počítačové algebry;
  - další matematické sázecí jazyky, jako je například TEX;
  - textové displeje (např. emulátory VT100);
  - tištěná média, včetně Braille.

  Je nutné zde dodat, že konverze do a z jiných systémů zápisu nebo médií může způsobit ztrátu informací během transformačního procesu;
- umožnit vkládání dalších informací pro konkrétní zobrazovací zařízení a aplikace;
- být rozšiřitelný;
- být čitelný pro člověka a snadno softwarově vytvořitelný a zpracovatelný.

Tyto body jsou postupně uváděny do praxe a cílem této diplomové práce je posunout hranice již zpracovaných oblastí o kousek dál.

Pro technickou dokumentaci je v současné době de facto standard sázecí systém Donalda Knutha TEX. Nastavuje vysokou latku v kvalitě vizuálního renderování a bylo vyvinuto velké úsilí, aby MathML mohlo poskytnout stejnou renderovací kvalitu. Vzhledem k množství dokumentů v TEXu je přirozeným požadavkem, aby existovaly nástroje pro import matematiky z TEXu do MathML. Tento požadavek se daří plnit, příkladem může být diplomová práce Heleny Vildové, jejíž součástí je filtr pro webový server Apache 2.0, umožňující převod matematiky v TEXu (LATEXu) na webu do HTML s MathML.

Atraktivitu MathML zvyšuje snadnost a jednotnost zpracování XML dokumentů. Dá se využít celá řada nástrojů k manipulaci s výrazy v MathML, od standardních nástrojů XML-processingu, přes šablony zajišťující převod do a z jiných formátů (např. TEXu), až po aplikace generující vizuální reprezentaci výrazu.

Vzhledem k úspěchům formátů z rodiny XML je pravděpodobné, že v budoucnosti převezme MathML po TEXu roli výchozího formátu pro zápis matematiky v technických textech. Ve spojení s publikačními formáty jako je DocBook, ve kterém je například tvořena i tato práce, pak může MathML vytvořit komplexní řešení pro psaní a zpracování technických dokumentací.

3.3 Použití MathML pro nevidomé

MathML není vhodný k přímému použití pro zápis matematiky pro nevidomé, protože je primárně určen k počítačovému zpracování. Zápisy v něm jsou relativně dlouhé, což je při tisku v Braille nevhodné, a rovněž složitost čtení je značná. Použití formátu MathML by bylo vhodné při zpracování matematických textů pro nevidomé na počítačích.

Existence mnoha kódování matematiky v Braille vytváří bariéry jednak pro konverzi zápisů do jiných formátů, jednak pro čtení v oblastech, kde je použito kódování odlišné. Protože většina takových textů v současné době vzniká na počítačích (nebo podobných programovatelných zařízeních), neměl by být problém používat jednotný formát pro ukládání matematických zápisů. Pro tento účel by mohl být vhodný i jazyk MathML. V jeho prospěch mluví například také skutečnost, že byl navrhován s ohledem na možnost tisku v Braille a čtení pomocí řečového syntetizéru.

Nasazení MathML pro ukládání matematiky pro nevidomé zatím brání malá podpora ze strany matematiků a nedostatek software pro zpracování. Také neexistence nástrojů, které by MathML zápis uměly generovat z nějaké formy jednoduchého zápisu, použitelného například ve škole při hodině matematiky, je určitě jedním z faktorů, proč zatím není možné nasadit MathML pro širší použití mezi nevidomými.

3.4 Možnosti značení

Jazyk MathML nabízí dva způsoby zápisu matematiky – prezentační a kontextový. Jak už název napovídá, hlavním úkolem prezentačního značení je vyjádřit grafickou reprezentaci vzorce. Naopak kontextové značení má vyjadřovat obsah (sémantiku) a grafické vyjádření (renderování) záleží na zvoleném prohlížeči.

3.4.1 Prezentační značení

Prezentační značení ve specifikaci MathML 2.0 obsahuje přibližně 30 elementů. Většina z nich popisuje rozvržení (layout) obsahu, například dolní a horní index, zlomek, tabulka. Další elementy mohou mít příznakový charakter (<mi/> – identifikátor, <mo/> – operátor, <mn/> – číslo, ...) nebo to mohou být prázdné elementy, které upravují zarovnání a řazení v matematickém výrazu.

Je důležité zmínit, že ve většině elementů záleží na pořadí potomků. Například první potomek elementu <mfrac/> je čitatel zlomku a druhý jmenovatel. Toto sice není (a ani nemůže být) vynucováno na úrovni XML pomocí DTD, ale je nutné s tímto počítat na úrovni aplikační.

Prezentační značení je vhodné použít v případě, že autor dokumentu požaduje konkrétní vizuální reprezentaci. Současné prohlížeče a editory většinou pracují s tímto značením. Ačkoliv toto značení obsahuje řádově méně elementů než kontextové, jeho automatické zpracování na úrovni matematiky je složitější, neboť důležité informace jsou uvedeny až na úrovni textových uzlů.

3.4.2 Kontextové značení

Kontextové značení obsahuje přibližně 120 elementů. Velká část z nich jsou prázdné elementy, které odpovídají různým operátorům, relacím a funkcím. Další skupinu tvoří elementy kódující matematické datové typy (<matrix/>, <set/>). Třetí skupina

obsahuje „aplikátory“, které vyjadřují, že daná operace je aplikována na výraz. Do této skupiny patří i element <apply/>, což je pravděpodobně nejvýznamnější kontextový element používající se k uplatnění operátoru nebo funkce na kolekci argumentů.

I v kontextovém značení platí, že záleží na pořadí potomků. V elementu <apply/> je první dětský element funkce, která bude aplikovaná na následující elementy v uvedeném pořadí. Zápis je tedy vždy prefixový, například výraz a + b bychom mohli zapsat takto:

```
<mrow>
  <apply>
    <plus/>
    <ci>a</ci>
    <ci>b</ci>
  </apply>
</mrow>
```

3.5 Nástroje pro práci s MathML

Pro rozšíření nových formátů je nutné, aby existovalo programové vybavení, které umožňuje s daným formátem pracovat. Software podporující MathML můžeme rozdělit do tří základních kategorií – prohlížeče, editory a pomocné nástroje (např. konvertory do jiných formátů). Ucelený seznam nástrojů je zveřejněn na stránkách konzorcia W3C (http://www.w3.org/Math/implementations.html).

3.5.1 Prohlížeče

MathML je primárně určen pro použití na webu, proto se nejprve zaměříme na možnosti jeho zobrazení ve webových prohlížečích. Nejrozšířenější prohlížeč v současné době – Internet Explorer od firmy Microsoft – zatím nemá přímou podporu MathML, ale existuje volně dostupný zásuvný modul (plugin) MathPlayer, který tuto funkcionalitu zajišťuje. Oproti Internet Exploreru jsou na tom lépe prohlížeče Mozilla a Netscape, které samy dokáží zobrazovat MathML. Poslední v řadě rozšířených prohlížečů je Opera, která v současné době podporu MathML nemá.

Za zmínku určitě stojí i webový prohlížeč Amaya, který je vyvíjen konzorciem W3C. Kromě schopnosti zobrazovat MathML nabízí i možnosti editace.

Mezi další software renderující MathML patří open-source program GtkMathView, který zobrazuje prezentační značení a prohlížeč technických dokumentů od IBM s názvem texexplorer.

3.5.2 Editory

Editačních nástrojů je celá řada a vzájemně se liší ve stupni implementace MathML, v uživatelské vstřícnosti, v ceně a licenci apod. Následující seznam obsahuje nástroje, které nás zaujaly:

- Amaya – výše zmíněný webový prohlížeč od W3C. Matematika může být vkládána ve WYSIWYG módu pomocí hlavního menu, nebo v módu zobrazení struktury dokumentu, což využijí hlavně pokročilejší uživatelé;
- Math Type a WebEQ – komerční software z dílny Design Science. Nabízí komplexní řešení pro různé platformy;

- SciWriter – vědecký WYSIWYG editor od firmy soft4science pro operační systémy Windows. Nativním formátem je XHTML a prezentační značení MathML. Tato firma vyvíjí také editační komponentu pro MathML v .Net framework;
- OpenOffice.org – volně šiřitelný kancelářský balík. Obsahuje editor rovnic Math, který umožňuje export do MathML 1.01. Bohužel neumí načítat dokumenty, které sám neexportoval;
- mathmled – volně šiřitelný plugin do webového prohlížeče Mozilla. Je naprogramovaný pomocí JavaScriptu a využívá kaskádové styly.

3.5.3 Další nástroje

Důležitou skupinu software tvoří nástroje pro konverze MathML do a z jiných formátů. Pozornost si zaslouží především nástroje pro konverzi dokumentů v TEXu:

- TEX4ht – pravděpodobně nejrozšířenější nástroj z této skupiny (distribuovaný pod licencí LPPL – LATEX Project Public License). Konvertuje TEXové dokumenty do různých formátů XML. Příkaz `mzlatex` konvertuje dokument do XHTML s prezentačním značením MathML;
- TtM (TEX to MathML translator) – konvertuje plainTEX a LATEX do HTML se zápisem matematiky v MathML. Verze pro Windows stojí 39$, pro linux je ke stažení zdarma;
- itex2mml – nástroj, který konvertuje webové stránky obsahující itex (zjednodušená forma LATEXu pro použití na internetu) do XHTML+MathML tvaru;
- EquationSheet.com – online konvertor z LATEXu, který bohužel zpracovává pouze výrazy kratší než 256 znaků;
- TEXMathFilter – modul pro webserver Apache 2.0, naimplementovaný ve formě výstupního filtru. Vzniká jako diplomová práce Heleny Vildové na Fakultě informatiky.

Je podporován i převod opačným směrem:

- tbook – DTD pro vědecké dokumenty. Matematika je vkládána ve formátu MathML. Součástí jsou i transformační styly pro převod do LATEXu, DocBooku, XHTML;
- xsltml (MathML to LATEX translator) – kolekce transformačních stylů pro konverzi prezentačního i kontextového značení do LATEXu.

Do kategorie konvertorů patří i projekt BraMaNet a aplikace M2B vytvořená v rámci této diplomové práce. Oba tyto programy pracují na principu XSL transformací a převádí MathML do zápisu matematiky v Braille. Posledním projektem, který by neměl zůstat nepovšimnut, je JEuclid, což je volně šiřitelná komponenta pro Apache Cocoon. Umožňuje převádět MathML do formátu GIF nebo do SVG (XML formát pro vektorovou grafiku).

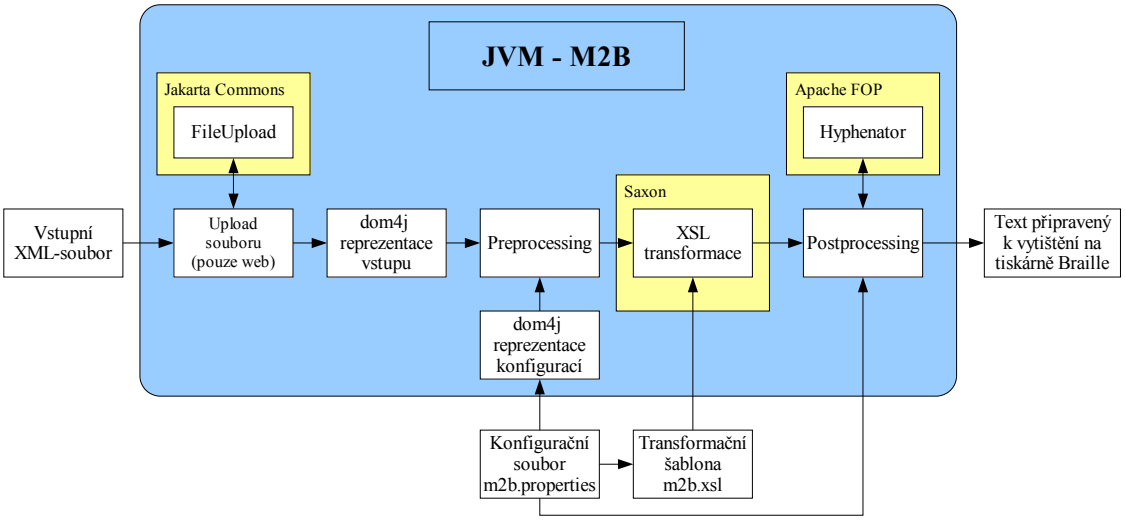
Kapitola 4

Převod z MathML do Braille pomocí aplikace M2B

M2B je aplikace, která transformuje MathML do Braille. Je použitelná pro různé jazyky i různé normy Braille. Díky zvoleným technologiím (Java & XSL transformace) je platformně nezávislá a může být snadno zařazena do komplexnější sady nástrojů.

Aplikace M2B je šířena pod licencí GNU GPL (General Public License) verze 2 a novější, která umožňuje další vývoj široké komunitě vývojářů open–source.

4.1 Návrh M2B



Obrázek 4.1: Schéma aplikace M2B

Na obrázku figure 4.1 představuje modrá (tmavá) oblast virtuální stroj Javy (JVM), ve kterém běží aplikace M2B. Žluté rámečky (světlejší) – Jakarta Commons, Saxon, Apache FOP – reprezentují použité externí knihovny.

Aplikace M2B dostane jako parametr XML soubor, který má být transformován do Braille. Soubor může být buď přímo v MathML formátu, pak je kořenovým elementem `<mml:math/>`, nebo může jít o jiný formát (např. XHTML) obsahující jako část XML–stromu MathML. Běží-li M2B jako webová aplikace, je nejprve nutné nahrát XML soubor na server. K tomu napomáhá třída `FileUpload` z knihovny Jakarta Commons.

Převod do Braille začíná vytvořením dom4j stromové reprezentace vstupního a konfiguračního souboru. Ještě před započetím XSL transformace se upraví textové uzly převáděného XML podle pravidel uvedených v konfiguraci (tzv. preprocessing). Následuje transformační část zajištěná XSLT procesorem Saxon. Výsledkem transformace je čistý text, u kterého zbývá vyřešit řádkové zlomy (postprocessing).

Algoritmus pro dělení slov využívá TEXových vzorů, které jsou obsaženy v knihovně FOP (Formatting Objects Processor – Apache XML Project). Třída Hyphenator, která je součástí uvedené knihovny, označí místa, na kterých je možné slovo dělit.

Návrh M2B byl do určité míry zjednodušen díky podobnosti projektu BraMaNet. Hlavním úkolem bylo vyřešit problémy, které brání nasazení BraMaNetu v jiných verzích Braille než francouzské. BraMaNet obsahuje pouze jednu vrstvu pro nastavení aplikace – výběr znaků, které se budou posílat na Braillovu tiskárnu. Vše ostatní je řešeno přímo uvnitř transformačního stylu.

V BraMaNetu není možné jednoduše ovlivnit překlad matematického výrazu (např. záměna znaku pro zlomkovou čáru, změna typu závorek, nastavení operátoru jako infixu apod.) v závislosti na použitém kódování Braille. Překlad operátorů je v Lyonském projektu řešen testem na ekvivalenci celého textového uzlu a nějaké předlohy. Znak navíc v definici operátoru, či operátor složený z více částí způsobí nerozpoznání, a tedy i chybné převedení do Braille. Například test převod operátoru „>“ (relace „větší než“) je řešen následovně:

```
<xsl:when test=".='&gt;'"> <!-- strict. superieur -->
  <xsl:text>&pt46;&pt345;</xsl:text>
</xsl:when>
```

Obdobně je zpracován i operátor „<“. Problém nastane, když někdo ve svém zápise použije operátor „<>“ nebo „>>“. Převod tohoto operátoru se nezdaří.

Aplikace M2B využívá konfigurační soubor `m2b.properties` ve formátu XML, který umožňuje převodní pravidla upravovat. Jeho struktura je navržena tak, aby se při změně překladu znaků a řetězců do Braille nemusela upravovat transformační šablona.

Komplexní práce s textovými uzly je při XSL transformaci skoro nemožná, proto jsme se rozhodli využít platformně nezávislý programovací jazyk Java k dílčím úpravám textu. V rámci preprocessingu řešíme převod čísel, velkých písmen a nahrazení MathML operátorů operátory z příslušné verze Braille pomocí tabulky náhrad. Tímto přístupem se eliminuje problém BraMaNetu, který způsoboval chybný převod matematických operátorů.

V konfiguračním souboru je použita ještě další vrstva s nastaveními usnadňujícími použití. Mimo zmíněné transformace matematického výrazu byla vzata v úvahu i jazyková závislost vstupního textu, tedy že jeden výraz může být v různých jazycích převeden různě. Proto nástroj M2B pracuje i s atributem `xml:lang` a podle něho určuje, která skupina nastavení bude použita.

4.2 Příprava textu

Příprava dokumentů před zpracováním XSLT šablonou řeší úkoly, které by při transformaci nebyly možné nebo které by transformaci znepřehlednily. Většina z těchto úkolů by se dala řešit i po transformaci na výsledném dokumentu, ale v tom případě už by byly ztraceny informace o kontextu.

Aplikace M2B očekává jako parametry XML soubory (většinou MathML nebo XHTML dokumenty), které mají být převedeny do Braille. Tyto soubory jsou po jednom zpracovávány v následujících krocích:

- načtení dokumentu do paměti – využívám stromově orientované rozhraní dom4j, které oproti standardnímu rozhraní DOM nabízí více funkcí a rychlou práci s XPath výrazy;
- úprava obsahu elementů mfenced – tento krok vyřeší separátory mezi jednotlivými podelementy;
- překlad čísel v textových uzlech a atributech do Braille;
- převod velkých písmen na malá s příslušným prefixem;
- zpracování tabulky náhrad, která je definována v konfiguračním souboru (obsahuje například náhrady operátorů).

Na takto připravený dokument použijeme transformační XSL styl.

Tabulka náhrad zpracovaná při preprocessingu tvoří základ převodu prezentačního značení MathML do Braille. Oproti tomu XSL transformace řeší většinu převodu kontextového značení.

4.3 Transformace

Hlavní část převodu je prováděna pomocí šablon v transformačním stylu `m2b.xsl`. Převod zajišťuje XSLT procesor Saxon ve verzi 7.3, který částečně implementuje pracovní návrhy (Working Drafts) doporučení XSL 2.0 a XPath 2.0.

4.3.1 Java část

Saxon je aplikace napsaná v Javě, je tedy možné použít její třídy v jiné Java-aplikaci bez nutnosti samotného spuštění Saxonu. Tento princip využívá i M2B. V konstruktoru aplikace je ze stylu `m2b.xsl` vytvořen objekt `sheet` implementující `interface javax.xml.transform.Templates` následujícím způsobem:

```
TransformerFactoryImpl saxonFactory =
                                    new TransformerFactoryImpl();
Source styleSource = new SAXSource(new InputSource(”m2b.xsl”));
Templates sheet = saxonFactory.newTemplates(styleSource);
```

Tento objekt je později použit k vytvoření transformeru, který převádí jednotlivé XML dokumenty do Braille:

```
Transformer instance = sheet.newTransformer();
Result result = new StreamResult(....);

for (int i = 0; i < files.length; i++) {
  Document doc = parse(new File(files[i]));
  instance.transform(new DocumentSource(doc), result);
}
```

Výsledný text je zpracován algoritmem, který řeší řádkové zlomy.

4.3.2 XSL část

Šablona `m2b.xsl` je základ aplikace M2B. V závislosti na jazykovém nastavení převádí vstupní XML strom do Braille. Při jejím programování se procházelo doporučení k MathML 2.0 a jednotlivé kapitoly do ní byly postupně zapracovávány. Pro snadnější opravy při změně v dalších verzích MathML, jsou v komentářích uvedeny odkazy na příslušné části doporučení.

Větší část XSL-šablony je věnována kontextovému značení MathML, ve kterém je zápis matematických výrazů přesnější. Pro většinu výrazů je kořenem element <`mml:apply/`> a jeho potomky jsou operátor a argumenty.

Bylo nutné vyřešit, jak se budou převádět jednotlivé elementy v závislosti na jazycích (`tan(x)` oproti `tg(x)`), na jakou pozici umístit operátor (infix: `1+1`, prefix: `sin(x)`, postfix: `n!`), jak uzávorkovat danou formuli (`x*y+z`, `x*(y+z)`, `sin(y+z)`, `sin((y+z))`). Tato nastavení jsou čtena z konfiguračního souboru `m2b.properties`.

Některé výrazy jsou zpracovány obecnou šablonou pro element <`apply/`> (například trigonometrické funkce) a jiné mají svá vlastní řešení (mocniny, integrály apod.). Obecná šablona zjišťuje kategorii, a tedy i způsob přepisu daného operátoru v konfiguračním souboru.

Prezentační značení je díky své povaze šablonou řešeno pouze částečně. Operátory jsou zde uváděny jako textové uzly a v různých jazycích mohou mít různé vyjádření (viz. například výše uvedený rozdíl tg a tan). Tento problém se v aplikaci M2B řeší při preprocessingu použitím tabulky náhrad. Obsah tabulky je opět čten ze souboru `m2b.properties`.

4.3.3 Struktura konfiguračního souboru

Konfigurační soubor slouží k nastavení pravidel převodu. Jednak se zde nastavuje tvar přepisu do Braille (znaky, operátory, ...), jednak sémantika některých výrazů (např. sinus je prefixový operátor, faktoriál je postfixový).

Jako nejvhodnější struktura pro konfiguraci se jevilo XML. Můžeme s ním pracovat stejnými nástroji jako s MathML a informace v něm můžeme stromově uspořádávat, což je využito při správě konfigurací pro různé jazyky.

Na začátku souboru `m2b.properties` jsou importovány entity reprezentující znaky Braillovy abecedy z dokumentu `printer.properties`. Definice těchto znaků je vydělena z konfiguračního souboru z důvodu snadné přenositelnosti na jinou tiskárnu Braille.

Potomky kořenového elementu (<`m2b/`>) jsou skupiny s nastaveními pro jednotlivé jazyky – elementy <`group/`>. První skupinou je vždy „univerzální skupina“ (<`group caj:lang="general"/`>), která obsahuje veškeré nastavení M2B, ostatní skupiny jsou tzv. „rozdílové“.

Rozdílová skupina se vztahuje k určitému jazyku, který je určen atributem `caj:lang`. Obsahuje základní sadu povinných elementů a rozdíly v přepisu daného jazyka oproti univerzální skupině. V některých případech je nutné při změně předefinovat celou související část, například při změně operátoru z infixového na postfixový je třeba znovu vyjmenovat v dané skupině celý infixový a postfixový seznam (<`postfix/`>, <`infix/`>).

V jazykové skupině jsou přepisy operátorů dále rozděleny podle toho, jak jsou (či mají být) zpracovávány šablonou `m2b.xsl`. Přepis operátorů, které jsou řešeny obecnou šablonou, je definován v elementech <`simple_ops/`>, <`enc_ops/`> – rozdíl je

v pravidlech pro uzávorkování těchto operátorů. Ostatní operátory většinou mají svou sadu pravidel uvedenou ve zvláštní skupině (např. <powers/>, <integrals/>).

Tabulka náhrad použitá v rámci preprocessingu je v jazykové skupině reprezentovaná obsahem elementu <replacements/>. Jednotlivé řádky tabulky tvoří obsah elementu <replace/>. Například následující převodní tabulka

Example:	ex:
Definition:	df:
Lemma:	lm:

bude mít v konfiguračním souboru zápis:

```
<replacements>
  <replace><from>Example:</from><to>ex:</to></replace>
  <replace><from>Definition:</from><to>df:</to></replace>
  <replace><from>Lemma:</from><to>lm:</to></replace>
</replacements>
```

Transformační styl se do souboru s konfigurací odkazuje přes globální proměnnou `m2b`.

```
<xsl:param name="op-doc" select="'m2b.properties'"/>
<xsl:variable name="m2b" select="document($op-doc)/caj:m2b"/>
```

Uvnitř jednotlivých šablon se v prvním kroku zjišťuje jazyk aktuálního uzlu elementu. Slouží k tomu funkce `getLang(Node)`, která testuje jazyky definované v konfiguračním souboru. Získání správných jazykových skupin a jejich použití k načtení konfigurace zajišťuje kód podobný tomuto:

```
<!-- zjištění kódu jazyka -->
<xsl:variable name="op-lang" select="caj:getLang(.)"/>

<!-- výběr použitelných skupin -->
<xsl:variable name="lgroup"
select="$m2b/group[@caj:lang=$op-lang or @caj:lang='general']"/>

<!-- použití jazykové skupiny k vypsání znaku integrálu -->
<xsl:value-of
  select="$lgroup[integrals/int][last()]/integrals/int"/>
```

4.4 Postprocessing

Zpracování vstupního XML souboru končí vyřešením řádkových zlomů. Tento krok obstarává třída `M2BHyphen`, která pracuje s třídami zajišťujícími dělení slov podle TEXových vzorů v knihovně FOP od Apache Foundation.

Současná verze aplikace M2B řeší řádkové zlomy v textu, ale neřeší zlomy u matematických výrazů. Tento problém by mohl odstranit stav, kdy by výstupem XSL

transformace nebyl pouze čistý text, ale kdyby v zápise zůstala jednoduchá forma značení. Toto značení by určovalo začátek a konec matematiky (které jsou nyní vyznačeny vloženými řádky s jednoznačným textem) a začátek a konec matematických operátorů. Zlom řádků by potom byl povolen například za infixovým operátorem s tím, že na novém řádku by byl tento operátor uveden znovu.

4.5 Webové rozhraní

Součástí projektu M2B je i malá webová aplikace, která provádí online převod XML dokumentů obsahujících MathML. Výkonná část je zajišťována servletem M2BWeb (`cz.muni.fi.xcacek.Web`). Tento servlet byl vyvíjen v servletovém kontejneru Jakarta Tomcat, ale funkcionalita by měla zůstat zachována i na jiných webových serverech (IBM WebSphere, BEA WebLogic, Jetty apod.).

Servlet je třída v programovacím jazyce Java (Java Servlet technology) používaná pro rozšíření možností aplikačních serverů založených na programovacím modelu výzva–odpověď. Ačkoliv mohou servlety zodpovídat různé typy dotazů, většinou jsou používány k rozšíření aplikací na webových serverech. Pro takovéto aplikace jsou nadefinovány HTTP-specifické servletové třídy.

Při programování servletu jsme narazili na problém s umístěním transformačních a konfiguračních souborů. Předpokládali jsme, že při umístění souborů do adresáře `m2bweb\WEB-INF\classes` bude Tomcat schopen tyto soubory najít bez udání absolutní cesty. Toto se bohužel nestalo, využili jsme tedy náhradní řešení, které využívá konfiguračního parametru předávaného servletu při inicializaci. Odpovídající část kódu potom vypadá následovně:

Zjištění cesty k webové aplikaci.

```
public void init(ServletConfig _config) {
  realPath = _config.getServletContext().getRealPath("/");
  realPath =
    realPath.substring(0, realPath.length() - 1)
                .replaceAll("\\\\", "\\/");
}
```

Nastavení cest k souborům.

```
options.xslFile = realPath + "/WEB-INF/classes/m2b.xsl";
options.propFile = realPath + "/WEB-INF/classes/m2b.properties";
```

Netriviální záležitostí při programování servletů je zajištění nahrání souborů z klientského počítače na server, kde běží webová aplikace (v případě aplikace M2B se nahrávají XML soubory obsahující MathML). Existuje několik knihoven, které implementují příslušné algoritmy. Pro potřeby této práce byla zvolena knihovna FileUpload, která je součástí balíku knihoven Jakarta Commons, vyvíjených pod vlajkou Apache Software Foundation.

Kapitola 5

Další vývoj

Rozvoj aplikace dalšími osobami a institucemi je umožněn díky použité licenci GNU GPL. Uživatel se znalostmi programování může díky GPL flexibilně reagovat na objevené chyby nebo se podílet na jejich hledání. Může také upravovat program tak, aby dokonale odpovídal jeho představám.

Aplikace M2B vytvořená v rámci diplomové práce je pouze kostrou, kterou je potřeba dále nastavit pomocí konfiguračních souborů. Nejdůležitějším počátečním krokem při nasazení aplikace M2B do provozu je doplnění tabulky náhrad pro daný jazyk tak, aby obsahovala operátory používané v prezentačním značení MathML a další náhrady, které se mohou v daném jazyce objevit.

Sestavení korektní tabulky náhrad vyžaduje pečlivé zvážení možných komplikací. Je například možné, že se po nahrazení části řetězce utvoří posloupnost, která je dále v tabulce náhrad, a místo jednoho nahrazení se provedou dvě, což znamená chybný výstup. Je dobré si také uvědomit, které znaky jsou použity pro tisk na konkrétní tiskárně Braille (nastavení v souboru `printer.properties`). Aby se při konfiguraci tabulky náhrad předešlo konfliktům s entitami reprezentujícími znaky v Braille (např. &b1;, &b12;, ...), může být použito systému jednoznačně rozpoznatelných řetězců zastupujících entity v tabulce náhrad. Samozřejmě musí být zaručeno, že tyto řetězce nebudou tabulkou náhrad změněny.

Například konfigurace

```
<replacements>
  <replace><from>[</from><to>&b6;&b236;</to></replace>
  <replace><from>]</from><to>&b6;&b356;</to></replace>
...
</replacements>
```

může být nahrazena zápisem

```
<replacements>
  <replace><from>[</from><to>@b6@@b236@</to></replace>
  <replace><from>]</from><to>@b6@@b356@</to></replace>
...
  <replace><from>@b236@</from><to>&b236;</to></replace>
  <replace><from>@b356@</from><to>&b356;</to></replace>
  <replace><from>@b6@</from><to>&b6;</to></replace>
</replacements>
```

V tomto příkladě počítáme s tím, že znak @ není v tabulce náhrad použit k jinému účelu.

V rámci dalšího vývoje aplikace M2B by bylo vhodné zaměřit se na zrychlení transformačních algoritmů. XSL transformace jsou řešeny pomocí Saxonu. Možná vylepšení tedy mohou být při převodu textových uzlů ve třídě `TextProcessor`, nebo v algoritmu dělení řádkových zlomů ve třídě `M2BHyphen`. Zpracování by se pravděpodobně zrychlilo a méně zatěžovalo paměť při použití rozhraní pro zpracování XML založeném na událostech (SAX), namísto nyní používaného stromově orientovaného rozhraní (dom4j).

Asi nejslabším místem je v současné verzi aplikace M2B řešení řádkových zlomů. Tato část představuje několik úskalí, která v současné verzi aplikace nejsou řešena:

1. pro všechny texty jsou použity české vzory dělení slov;
2. v matematických vzorcích není použito konvencí pro dělení matematiky v Braille (např. zopakování operátoru na novém řádku);
3. přesáhne-li zbytek slova po rozdělení délku řádku, nebo nelze-li slovo dělit na požadovanou délku podle vzorů, je děleno tak, aby co nejvíce zaplnilo řádek;
4. vzory pro dělení slov nejsou převedeny do tvaru pro tisk Braille (prefixy, nahrazení znaků apod.).

Řešením prvního a druhého bodu je uchování dodatečné informace o dokumentu při XSL transformaci. Tímto způsobem je nyní řešeno například rozpoznání matematiky. Značkování určující jazyk dané části dokumentu a doplňující informace do matematických výrazů by bylo vloženo při transformacích příslušných elementů a pracovala by s ním třída `M2BHyphen`. Výstupem transformace je v současné verzi plain text, pro zmíněné úpravy by bylo vhodné upravit výstup na XML a i při dělení řádků s ním pracovat pomocí nástrojů pro XML-processing.

Nepoužívá-li vstupní soubor s MathML jmenné prostory, XSL transformace neproběhne správně. V šablonách jsou totiž použity prefixy pro jmenný prostor MathML (mml) – například `<xsl:template match="mml:declare">`. Zamýšlené řešení bylo ignorovat jmenný prostor a použít konstrukci `*:JmenoElementu` (v uvedeném příkladě `*:declare`), ale Saxon s takovým XPath výrazem neumožňuje pracovat. Je tedy na uživateli, aby si kontroloval použití jmenných prostorů. V šabloně by bylo možné situaci, kdy není uveden jmenný prostor, řešit uvedením jména elementu s prefixem i bez prefixu – například

```
<xsl:template match="declare|mml:declare">
```

Aplikace M2B je zveřejněna na serveru SourceForge.net pod jménem Mml2B (`http://mml2b.sourceforge.net/`). SourceForge je největší server hostující open-source projekty na světě. Zpřístupnění aplikace včetně zdrojových kódů na internetu by mělo usnadnit její rozšíření jak mezi uživatele tak mezi vývojáře, kterým bude umožněno zapojit se do rozvoje projektu.

Kapitola 6

Závěr

Tato diplomová práce ukázala možné přístupy k převodu technických dokumentů do Braille. Jsou v ní také srovnány normy zápisu Braille a nástroje používané k převodu mezi černotiskem a Braille. Cílem bylo vytvořit ucelený seznam použitelných technologií a existujících řešení se zaměřením na technickou literaturu a využití značkovacích jazyků rodiny XML.

Ukázalo se, že největším problémem při převodu matematických dokumentů do Braille je existence mnoha norem tohoto písma. Problém je o to výraznější, že normy jsou průběžně upravovány tak, aby reflektovaly vyvíjející se požadavky na zápis matematiky. Stále vznikají nové kódy Braille, které mohou být pro nevidomé přínosem, ale které také mohou znamenat nepoužitelnost existujících transformačních řešení pro převod do nového zápisu.

Součástí práce je programová realizace převodníku z MathML do Braille (M2B) včetně webového rozhraní. Aplikace je napsána v platformně nezávislém jazyce Java a využívá XSL transformace, které zajišťují většinu převodu. Použité technologie umožňují využít aplikaci v různých operačních systémech a v různých jazykových prostředích. Oproti podobnému projektu BraMaNet, který je vyvíjen na Lyonské univerzitě a převádí prezentační značení MathML do francouzského Braille, pracuje M2B i s kontextovým značením a není vázán na konkrétní normu Braille.

Přínosem práce je vícevrstvá konfigurace převodních nástrojů a umožnění různého převodu matematiky u různých jazyků použitých v rámci jednoho XML dokumentu.

Reference

[ICCHP-00] *Computers Helping People with Special Needs – ICCHP 2000*, H. Guo, G. Gupta, A. Karshmer, C. Weaver, J. Mendez a S. Geiger, Austrian Computer Society (OCG), 2000, s. 319–326. 2.3

[docbook] *DocBook: The Definitive Guide*, Norman Walsh a Leonard Muellner, O'Reilly and Associates, Inc., ISBN 156592-580-7, 2002.

[gonzurova-97] *Příručka pro přepis textů do bodového písma (díly 1-4)*, Wanda Gonzúrová, Knihovna a tiskárna pro nevidomé K. E. Macana, 1997.

[i-amaya] `http://www.w3.org/Amaya/`, Amaya – W3C's Editor/Browser.

[i-ams] `http://elvis.inf.tu-dresden.de/publichtml/asc2html/ams/`, ASCII-Mathematikschrift (AMS).

[i-amsgs] `http://www.provvstudi.vi.it/erica/doc/matem1.htm`, Far matematica a scuola con computer e riga braille.

[i-bramanet] `http://handy.univ-lyon1.fr/projets/bramanet/`, BraMaNet, Logiciel De Traduction Des Mathématiques En Braille.

[i-brl] `http://brl.org/`, BRL: Braille Through Remote Learning.

[i-css] `http://academ.hvcc.edu/~kantopet/css/`, CSS: Cascading Style Sheets.

[i-gnu] `http://www.gnu.org/licenses/licenses.html`, GNU licenses.

[i-history] `http://www.nyise.org/blind/`, The History of Reading Codes for the Blind.

[i-iceb] `http://www.iceb.org/`, International Council on English Braille (ICEB).

[i-jakarta] `http://jakarta.apache.org/`, The Apache Jakarta Project, Apache Software Foundation.

[i-learnbr] `http://snow.utoronto.ca/prof_dev/tht/braille/`, Learn to Braille.

[i-mathzilla] `http://pear.math.pitt.edu/mathzilla/`, MathML+Mozilla – the future of Math on the Web.

[i-mavis] `http://www.nmsu.edu/~mavis/`, MAVIS – Mathematics Accessible to Visually Impaired Students.

[i-mozilla] http://www.mozilla.org/, Mozilla project.

[i-mplay] http://www.dessci.com/en/products/mathplayer/, MathPlayer <Display MathML in your browser>.

[i-sapgs] http://dots.physics.orst.edu/, The Science Access Project.

[i-saxon] http://saxon.sourceforge.net/, The XSLT Processor.

[i-tex4ht] http://www.cis.ohio-state.edu/~gurari/TeX4ht/, TEX4ht: LATEX and TEX for Hypertext.

[i-w3c] http://www.w3.org/, The World Wide Web Consortium.

[i-xml] http://www.w3.org/XML/, Extensible Markup Language (XML).

[i-xsl] http://www.w3.org/Style/XSL/, The Extensible Stylesheet Language Family (XSL).

[mml-01] *Mathematical Markup Language (MathML) Version 2.0 – W3C Recommendation*, Ron Ausbrooks, Stephen Buswell, Stéphane Dalmas, Stan Devitt, Angel Diaz, Roger Hunter, Bruce Smith, Neil Soiffer, Robert Sutor a Stephen Watt, The World Wide Web Consortium, 2001.

[pazdziora-97] *Algoritmy řádkového a stránkového zlomu v počítačové sazbě (diplomová práce)*, Jan Pazdziora, Fakulta informatiky MU, 1997.

[pierce-80] *An Introduction to Information Theory*, John Pierce, Dover Publications, ISBN 048624-061-4, 1980. 3

[smykal-94] *Pohled do historie slepeckého písma*, Josef Smýkal, Česká unie nevidomých a slabozrakých, 1994. 1.1

[suchy-02] *Konverze technických textů do Braille (diplomová práce)*, Miroslav Suchý, Fakulta informatiky MU, 2002. 2.3

[tesar-00] *DVIBRAILLE (bakalářská práce)*, Jiří Tesař, Fakulta informatiky MU, 2000.

[vochozka-00] *Značkovací jazyky a XML*, Josef Vochozka, Zpravodaj ÚVT MU, ISSN 1212-0901, 2000, roč.11, č.2, s. 7–11. 1.3

Příloha A

Popis přiloženého CD

Součástí diplomové práce je také CD obsahující aplikaci M2B, texty práce, zdrojové soubory a další software, který se týká této práce. Struktura přiloženého CD je následující:

```
[m2b]          # aplikace M2B
  [doc]          # Java API vygenerované pomocí javadoc
  [dtd]          # jednoduché DTD pro konfigurační soubor
  [lib]          # java archívy nutné k běhu aplikace
  [sample]       # referenční příklad - Pythagorova věta
  [src]          # zdrojové kódy aplikace
  [web]          # webová aplikace v archívu WAR
[Sofware]      # další software
  [Amaya]        # webový prohlížeč vyvíjený konzorciem W3C
  [BraMaNet]     # konvertor MathML -> Braille
  [itexToMML]    # konvertor itex -> MathML
  [JARs]         # knihovny použité aplikací M2B
  [JRE]          # Java Runtime Environment
  [mathmled]     # plugin do Mozilly - editor MathML
  [MathPlayer]   # MathML plugin do Internet Exploreru
  [Mozilla]      # webový prohlížeč s podporou MathML
  [TeX4ht]       # konvertor TeX -> (X)HTML, MathML
  [Tomcat]       # servletový kontejner (webový server)
  [TtM]          # konvertor TeX -> MathML
[SSGraph]      #  audiovizuální reprezentace 2D funkcí
[Text DP]      # text diplomové práce
  [src]          # zdrojový tvar DP
```

Příloha B

Dokumentace k aplikaci M2B

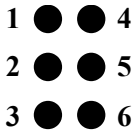
Tato dokumentace je rozdělena na část uživatelskou, která popisuje možnosti konfigurace a spuštění aplikace, a část programátorskou, ve které je ze zdrojových souborů v Javě vygenerována pomocí nástroje `javadoc` dokumentace pro Java API.

B.1 Uživatelská dokumentace

Aplikace M2B sloužící k převodu matematických textů zapsaných pomocí MathML do Braille je založena na XSL transformacích. Pro správnou funkčnost je potřeba aplikaci nakonfigurovat tak, aby reflektovala pravidla pro převod do Braille v daném jazyce.

B.1.1 Konfigurace M2B

Nastavení aplikace se provádí úpravou dvou textových souborů – `m2b.properties` a `printer.properties`. Změny těchto souborů mohou být prováděny jakýmkoliv textovým editorem podporujícím kódování UTF-8. Je lepší zvolit editor určený pro úpravy XML, který kontroluje, zda jsou XML dokumenty správně utvořeny (well-formed document).



Obrázek B.1: Číslování bodů u znaku v Braille

Soubor `printer.properties` definuje znaky, které jsou posílány na tiskárnu Braille. Znak je reprezentován entitou popisující použité body braillova znaku. Například znak 'a' reprezentuje první bod a zastupuje ho entita '&b1;', znak 's' je tvořen body 2,3,4 a příslušná entita má název '&b234;'. U tiskárny Braille musí být uvedeno, který znak z ASCII tabulky se na tuto tiskárnu má poslat, aby se vytiskl požadovaný znak v Braille. Mapování znaků Braille na ASCII znaky zajišťují zmíněné entity. Kdyby se například vytištění malého písmene 'a' mělo provést posláním znaku velkého 'A' na tiskárnu, vypadala by deklarace příslušné entity následovně:

```
<!ENTITY b1 "A">
```

Většinou není potřeba předefinovávat základní znaky (abeceda), ale například tisk prefixů už může být u různých tiskáren různý.

Nejdůležitější část konfigurace je úprava XML souboru `m2b.properties`, kde jsou definovány převodní pravidla z černotisku do Braille a pravidla pro zápis matematiky v Braille. Základní tvar tohoto souboru vypadá následovně:

```
<!DOCTYPE caj:m2b SYSTEM "./dtd/caj.dtd"
   [
   <!ENTITY % printer_properties PUBLIC
    "http://www.fi.muni.cz/xcacek/m2b/printer.properties"
    "./printer.properties">
   %printer_properties;
]>
<caj:m2b
     xmlns:mml='http://www.w3.org/1998/Math/MathML'
     xmlns:caj='http://www.fi.muni.cz/xcacek/m2b'>

  <group caj:lang="general">
    <!-- zde se vloží nastavení pro univerzální skupinu
    ...
    -->
  </group>

  <!-- další jazykové (rozdílové) skupiny
  <group caj:lang="cs">
  ...
  </group>
  <group caj:lang="de">
  ...
  </group>
  -->
</caj:m2b>
```

Pod kořenovým elementem <caj:m2b/> jsou elementy <group/> sdružující nastavení převodních pravidel pro jednotlivé jazyky. Kód jazyka (language code, ISO 639), pro který jsou nastavení určena, je uveden v atributu caj:lang. Jako první musí být uvedena univerzální skupina, ve které má atribut caj:lang hodnotu "general".

Univerzální skupina obsahuje základní nastavení. Všechny konfigurační elementy by měly být v této skupině uvedeny. Rozdílové skupiny obsahují rozdíly oproti univerzální skupině. U některých elementů stačí změnit pouze část, jiné musí být při změně uvedeny celé znovu (např. tabulka náhrad, seznam infixových operátorů, apod.). Následující seznam uvádí přehled použitých elementů a jejich význam.

```
number_prefix     - prefix udávající, že následující znaky jsou
                    zápisem čísla
number_postfix    - řetězec uváděný za číslem (jestliže takový
                    existuje)
hyphen            - znak/řetězec použitý pro rozdělení slova
no_translate_list - seznam MathML elementů, které nebudou
                    zpracovány v rámci preprocessingu
                    (například element degree udávající stupeň
                    derivace)
ignore_list       - seznam elementů ve vstupním XML souboru,
```

```
                    které budou transformační šablonou
                    ignorovány
d0, d1, ..., d9   - číslice (bez prefixu pro číslo)
replacements      - tabulka náhrad; řádky tabulky reprezentují
                    elementy <replace/> obsahující potomky
                    <from/> a <to/>
lowercase_prefix  - prefix pro malá písmena (např. po poslou-
                    pnosti velkých -- RAMdrive)
uppercase_prefix  - prefix pro posloupnost velkých písmen
uppercasechar_prefix - prefix pro jedno velké písmeno
lower_cases, upper_cases - seznam malých a velkých písmen
                    v daném jazyce (význam při určení, kam
                    vložit prefixy)
left_par_inner, right_par_inner - levá a pravá závorka pro
                    podvýrazy
left_par_inf, right_par_inf - závorky pro výraz s infixovým
                    operátorem (většinou zůstává prázdné, neboť
                    funkcionalita je zajištěna uzávorkováním
                    podvýrazu)
left_par_postf, right_par_postf
                  - závorky pro výraz s postfixovým operátorem
operand_sep_postf - oddělovač operandů pro výraz s postfixovým
                    operátorem
left_par_pref, right_par_pref
                  - závorky pro výraz s prefixovým operátorem
operand_sep_pref  - oddělovač operandů pro výraz s prefixovým
                    operátorem
left_par, right_par - standardní matematické závorky v daném
                    jazyce
mfenced_sep       - standardní oddělovač uzávorkovaných položek
                    (v elementu mfenced)
conjugate_simple  - operátor pro komplexně sdružené číslo
lambda_start, lambda_end
                  - začátek a konec lambda výrazu
lambda_sep        - oddělovač operandů lambda výrazu
doa_start, doa_end - začátek a konec obsahu elementu
                    domainofapplication (ve většině jazyků
                    začátek a konec dolního indexu)
low_index         - prefix pro dolní index
up_index          - prefix pro horní index
index_end         - označení konce indexu (horního nebo dolního)
log_start, log_end - začátek a konec logaritmu
log_base_start, log_base_end
                  - začátek a konec logaritmické báze
                    (většinou začátek a konec dolního indexu)
min, max          - funkce minima a maxima
lowlimit, uplimit - prefix pro meze limitních operátorů jako
                    jsou suma, produkt, ...
limit_end         - ukončení mezí limitních operátorů
```

limit_ops - seznam limitních operátorů v kontextovém značení MathML (uváděných bez prefixu mml); pro každý operátor je v elementech <start/> a <end/> uváděno ohraničení daného operátoru

interval - element sdružující nastavení intervalů obsahuje definici závorek a oddělovače

declare - potomci tohoto elementu definují, jak budou generovány deklarace (mml:declare)

choices - potomci tohoto elementu definují, jak budou generovány výběry hodnot při použití elementů <mml:piecewise/> a <mml:otherwise/>

choices - potomci tohoto elementu definují, jak budou generovány výběry hodnot při použití elementů <mml:piecewise/> a <mml:otherwise/>

sets - obsahuje nastavení pro zápis množin, seznamů a vektorů v Braille

powers - nastavení zápisu mocnin (a odmocnin) v Braille

quantifiers - nastavení převodu obecného a existenčního kvantifikátoru

integrals - nastavení pro integrální a diferenciální počet

tendsto - nastavení převodu elementu <mml:tendsto/> (šipky s určením směru)

matrix - potomci tohoto elementu definují pravidla pro zápis matic v Braille

selector - nastavení převodu elementu <mml:selector/> (výběru z matice, seznamu, vektoru)

numbers - skupina nastavení pro zápis čísel v Braille (závorky, zápis báze, polární a kartézský zápis komplexních čísel)

simple_ops - seznam jednoduchých operátorů v kontextovém značení MathML. Jednoduchý operátor znamená, že bude uzávorkován standardním způsobem se standardním oddělovačem operandů. Potomci tohoto elementu jsou MML operátory uváděné bez prefixu jmeného prostoru, jejich obsahem je textový uzel - řetězec jejich zápisu v Braille. Například

```
<factorial>&b235;</factorial>
```

pro zápis faktoriálu.
Pozice operátoru je standardně prefixová, ale může být změněna uvedením v elementu <infix/> nebo <postfix/>.

enc_ops - seznam ’závorkujících’ operátorů v kontextovém značení MathML. U těchto

```
                    operátorů je možné nakonfigurovat počáteční
                    a koncový řetězec výrazu a oddělovač
                    operandů. Například
                    <divide>
                      <start>&b23;</start>
                      <sep>&b12456;</sep>
                      <end>&b56;</end>
                    </divide>
                    pro zápis dělení.
presentation_enc_ops
                  - nastavení převodu některých elementů z
                    prezentačního značení MathML do Braille
                    (například podíl, horní a dolní indexy)
simple_list       - seznam operátorů, okolo kterých není nutné
                    psát v podvýrazech závorky (kontextové
                    značení MathML)
infix             - seznam infixových operátorů (kontextové
                    značení MathML)
postfix           - seznam postfixových operátorů (kontextové
                    značení MathML)
```

Pro pochopení detailů je vhodné podívat se přímo do souboru `m2b.properties`. Formou komentářů jsou vloženy poznámky k jednotlivým nastavením.

Při změnách v univerzální skupině lze doporučit neodstraňovat žádný z elementů. Namísto toho stačí odstranit obsah elementu, jedním z následujících způsobů.

```
<!-- prázdný element obsahuje pouze
  počáteční a koncovou značku -->
<jmeno_elementu></jmeno_elementu>

<!-- ekvivalentní zápis je -->
<jmeno_elementu/>
```

B.1.2 Instalace a spuštění aplikace

M2B je na přiloženém CD umístěna v adresáři `/m2b`. Program se neinstaluje, může být spuštěn přímo z přiloženého CD nebo ho lze nakopírovat na lokální disk.

Protože je M2B javová aplikace, je potřeba mít na počítači, kde bude spouštěna, nainstalován Java Runtime Environment ve verzi minimálně 1.4.0. Distribuce JRE od firmy Sun pro nejběžnější operační systémy ve verzi 1.4.6 jsou umístěny na přiloženém CD v adresáři `/Software/JRE`.

Javové třídy jsou zabaleny v souboru `/m2b/lib/m2b.jar`. Aplikace může být z adresáře `m2b` spuštěna příkazem

```
java -jar lib/m2b.jar [parametry]
```

nebo

```
java /
```

```
-classpath "lib/m2b.jar:lib/saxon7.jar:lib/dom4j.jar:lib/fop.jar" /
  cz.muni.fi.xcacek.m2b.M2BAppl [parametry]
```

Druhý způsob je ve tvaru pro Linux OS, pro MS Windows se jako oddělovač položek v parametru *classpath* používá středník.

Ke spuštění pod OS Windows může být použita dávka `m2b.bat`, ke stejnému účelu v Linuxu slouží soubor `m2b`. Aby dávky fungovaly správně, je nutné mít v systémové proměnné `PATH` nastavenu cestu ke spustitelným souborům Javy nebo doplnit celou cestu k programu java do dávky. Spuštění potom vypadá následovně:

```
m2b [parametry]
```

Program je ovládán parametry z příkazové řádky. Výstup programu jde na standardní výstup. Pro uložení výsledku do souboru můžete použít následující konstrukci přesměrování standardního výstupu.

```
m2b [parametry] >vystupni_soubor.txt
```

B.1.3 Parametry aplikace

Jak bylo zmíněno výše, spuštění M2B je možné následujícím způsobem:

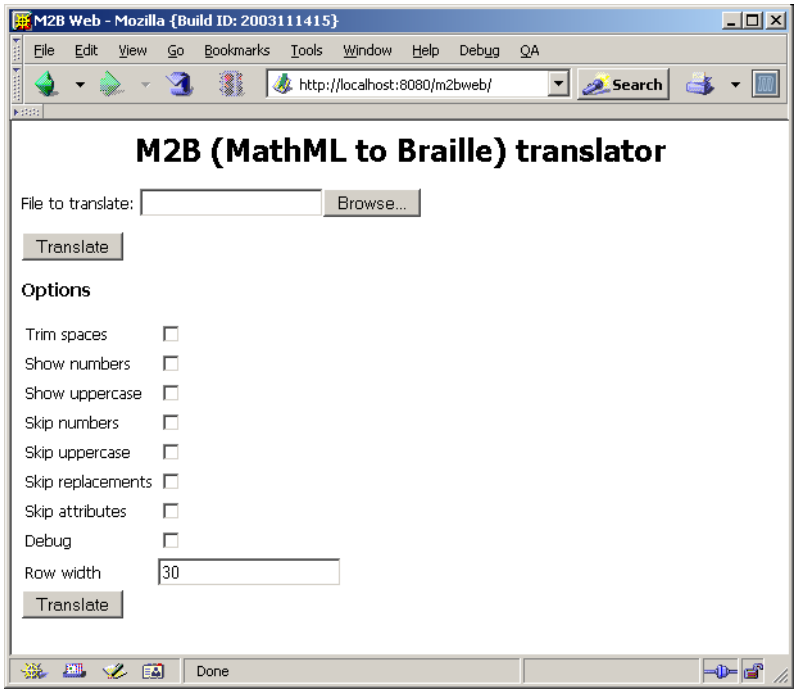
```
m2b [parametry]
```

přitom pod pojmem `[parametry]` jsou zahrnuty přepínače programu a také cesty k souborům, které mají být zpracovány. Detailnější zápis by tedy vypadal takto:

```
m2b [přepínače] soubor1 [soubor2 .. souborN]
```

Přepínače ovlivňují chování programu, mohou být následující:

```
--help,-h     vytiskne nápovědu (anglicky) a skončí
--version,-v  vytiskne verzi a skončí
--debug,-d    ladící mód, průběžně vypisuje informace
              o probíhajícím zpracování souborů
-w N          nastavuje šířku řádku pro Braillovu tiskárnu na
              hodnotu N (celé číslo)
-t            v textových uzlech vstupních XML dokumentů umaže
              počáteční a koncové mezery
-showNumbers  nenahrazuje čísla znaky Braille, pouze vloží prefix
              pro číslo (použití pro vizuální Braille)
-showUpper    nenahrazuje velká písmena, pouze vloží prefixy
              (použití pro vizuální Braille)
-skipNumbers  přeskočí zpracování čísel
-skipUpper    přeskočí zpracování velkých písmen
-skipReplace  přeskočí zpracování tabulky náhrad
-skipAttrs    přeskočí zpracování hodnot v atributech
-visual       ekvivalentní zápis pro '-showNumbers -showUpper'
```



Obrázek B.2: Webová aplikace M2B v prohlížeči Mozilla

B.1.4 Webová aplikace

Webová aplikace je distribuována pomocí systému „Web Application Archive“ (WAR). Na přiloženém CD je umístěna v souboru `/m2b/web/m2bweb.war`. Abyste mohli webovou aplikaci používat, musíte mít nainstalovaný servletový kontejner (Jakarta Tomcat, IBM WebSphere, BEA WebLogic, Jetty, apod.). Následující popis se týká webového serveru Tomcat, u ostatních konzultujte instalaci webových aplikací s dokumentací.

Instalaci serveru Jakarta Tomcat 5.0.16 pro MS Windows můžete nalézt i na CD v souboru `/Sofware/Tomcat/jakarta-tomcat-5.0.16.exe`.

Po nainstalování Tomcatu umístěte soubor `m2bweb.war` do adresáře `[tomcat_home]/webapps` a proveďte restart Tomcatu. Tomcat automaticky archiv rozbalí a webová aplikace je připravena k běhu. Většinou ji můžete začít používat zadáním podobné adresy jako

```
http://server:8080/m2bweb/
```

do webového prohlížeče.

Parametry předávané aplikaci M2B z příkazové řádky jsou ve webové aplikaci zastoupeny formulářem s volbami:

- File to translate – XML soubor, který má být převeden do Braille;
- Trim spaces – ořeže mezery v textových uzlech;
- Show numbers – nepřekládá čísla, pouze vkládá prefix pro číslo;
- Show uppercase – nepřekládá velká písmena, pouze vkládá prefixy;
- Skip numbers – přeskočí zpracování čísel;

- Skip uppercase – přeskočí zpracování velkých písmen;
- Skip replacements – přeskočí tabulku náhrad;
- Skip attributes – přeskočí zpracování textů v atributech;
- Debug – zapne ladící mód (průběžný výpis informací);
- Row width – udává šířku řádku na tiskárně Braille.

Převod se spustí stisknutím tlačítka `Translate`. Výstupním kódováním webové aplikace aplikace je UTF-8.

B.2 Referenční příklad

Na přiloženém CD je v adresáří `m2b/sample` uložen referenční příklad. Tvoří ho XML soubor `sample.xhtml` ve formátu XHTML obsahující MathML. Jedná se o česky a anglicky zapsanou Pythagorovu větu. Zdrojový kód vypadá následovně:

```
<!DOCTYPE html
  PUBLIC ”-//W3C//DTD XHTML 1.0 Strict//EN”
         ”http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd”>
<html xmlns=”http://www.w3.org/1999/xhtml”>
<head>
  <title>Pythagorova věta</title>
</head>

<body>
<h1>Pythagorova věta</h1>
<h2>Definice</h2>
<p>
Obsah čtverce sestrojeného nad PŘEPONOU (c) pravoúhlého
trojúhelníka je roven součtu obsahů čtverců sestrojených
nad jeho ODVĚSNAMI (a,b).
</p>
<h2>Rovnice</h2>
<math xmlns=’http://www.w3.org/1998/Math/MathML’>
<mrow>
  <apply>
    <eq/>
    <apply>
      <power/><ci>c</ci><cn>2</cn>
    </apply>
    <apply>
      <plus/>
      <apply>
        <power/><ci>a</ci><cn>2</cn>
      </apply>
      <apply>
        <power/><ci>b</ci><cn>2</cn>
```

```
      </apply>
    </apply>
  </apply>
</mrow>
</math>

<div xml:lang=”en”>
<h1>Pythagorean theorem</h1>
<h2>Definition</h2>
<p>
The sum of the areas of the squares on the LEGS
of a right triangle is equal to the area of the
square on the HYPOTENUSE.
</p>
<h2>Equation</h2>
<math xmlns=’http://www.w3.org/1998/Math/MathML’>
<mrow>
  <apply>
    <eq/>
    <apply>
      <power/><ci>c</ci><cn>2</cn>
    </apply>
    <apply>
      <plus/>
      <apply>
        <power/><ci>a</ci><cn>2</cn>
      </apply>
      <apply>
        <power/><ci>b</ci><cn>2</cn>
      </apply>
    </apply>
  </apply>
</mrow>
</math>
</div>
</body>
</html>
```

Tento soubor může být prohlížečem vygenerován například takto:

Pythagorova věta

Definice

Obsah čtverce sestrojeného nad PŘEPONOU (c) pravoúhlého trojúhelníka je roven součtu obsahů čtverců sestrojených nad jeho ODVĚSNAMI (a,b).

Rovnice

$$c^2 = a^2 + b^2$$

Pythagorean theorem

Definition

The sum of the areas of the squares on the LEGS of a right triangle is equal to the area of the square on the HYPOTENUSE.

Equation

$$c^2 = a^2 + b^2$$

Do konfiguračního souboru `m2b.properties` byla přidána rozdílová skupina pro angličinu následujícím kódem:

```
<group caj:lang="en">
   <powers>
      <power_string><multiple/>^</power_string>
   </powers>
   <uppercase_prefix>//</uppercase_prefix>
   <uppercasechar_prefix>/</uppercasechar_prefix>
</group>
```

Změnili jsme tedy znaménko pro mocninu a prefixy pro velké písmeno a posloupnost velkých písmen.

Spuštění jsme provedli v prostředí MS Windows následovně:

```
m2b.bat -w 28 -t sample\sample.xhtml >sample\out.txt
```

Výsledek převodu je uložen v souboru `out.txt`:

```
*pythagorova věta
*definice
*obsah čtverce sestrojeného
nad @přeponou (c) pravoé-
hlého trojéhelníka je roven
součtu obsahů čtverců se-
strojených nad jeho
@odvěsnami (a,b).
*rovnice
cí|bš”aí|bš +bí|bš
/pythagorean theorem
/definition
/the sum of the areas of the
 squares on the //legs of a
right triangle is equal to
the area of the square on
the //hypotenuse.
/equation
cˆ|bš”aˆ|bš +bˆ|bš
```

B.3 Java API

Následující stránky jsou vygenerovány programem javadoc ze zdrojových textů aplikace M2B.

cz.muni.fi.xcacek.m2b

Class LanguageSettings

```
java.lang.Object
  |
  +--cz.muni.fi.xcacek.m2b.LanguageSettings
```

---

public class **LanguageSettings**
extends java.lang.Object

Třída uchovávající nastavení překladu pro jednotlivé jazyky.

Author:
Josef Cacek

Field Summary

static java.lang.String[]	**basicTranslaions** seznam elementů jejichž obsah bude načten ze souboru m2b.properties

Constructor Summary

LanguageSettings(org.dom4j.Element langNode, M2BAppl.M2BOptions opts) načítá pro jazyk příslušný elementu langNode převodní pravidla (např. pro převod čísel, velkých písmen, apod.)

Method Summary

java.lang.String	**getName**() vrací kód jazyka, který tento objekt reprezentuje
java.lang.String	**getReplaceFrom**(int i) vrací i-tý řetězec, jenž má být nahrazen
int	**getReplacementsLength**() vrací počet prvků v tabulce náhrad pro příslušný jazyk
java.lang.String	**getReplaceTo**(int i) vrací náhradu pro i-tý řetězec
java.lang.String	**getTranslation**(java.lang.String s) vrací překlad elementu, podle nastavení v souboru m2b.properties
boolean	**hasReplacements**() test, zda v nastaveních pro tento jazyk je vyplněna tabulka náhrad
boolean	**hasTranslation**(java.lang.String s) test, zda má příslušný prvek (element) v daném jazyce překlad
boolean	**isIgnored**(org.dom4j.Node n) test zda má být pro tento jazyk daný uzel ignorován

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

basicTranslaions

```
public static java.lang.String[] basicTranslaions
```

seznam elementů jejichž obsah bude načten ze souboru m2b.properties

Constructor Detail

LanguageSettings

```
public LanguageSettings(org.dom4j.Element langNode,
                        M2BAppl.M2BOptions opts)
                 throws M2BException
```

načítá pro jazyk příslušný elementu langNode převodní pravidla (např. pro převod čísel, velkých písmen, apod.)

Parameters:
`langNode` - element `<group>` ze souboru m2b.properties
`opts` - nastavení aplikace M2B
Throws:
`M2BException`

Method Detail

getName

```
public java.lang.String getName()
```

vrací kód jazyka, který tento objekt reprezentuje

---

hasTranslation

```
public boolean hasTranslation(java.lang.String s)
```

test, zda má příslušný prvek (element) v daném jazyce překlad

Parameters:
`s` - jméno testovaného elementu

---

getTranslation

```
public java.lang.String getTranslation(java.lang.String s)
                                throws M2BException
```

vrací překlad elementu, podle nastavení v souboru m2b.properties

Parameters:
`s` - jméno elementu, pro který hledáme překlad
Returns:
překlad elementu
Throws:
`M2BException` - - jestliže konfigurace příslušného jazyka neobsahuje požadovaný element (překlad)

---

getReplacementsLength

```
public int getReplacementsLength()
                         throws M2BException
```

vrací počet prvků v tabulce náhrad pro příslušný jazyk

Throws:
`M2BException` - je-li počet elementů `from` různý od `to`

---

getReplaceFrom

```
public java.lang.String getReplaceFrom(int i)
                                throws M2BException
```

vrací i-tý řetězec, jenž má být nahrazen

Parameters:
`i` - pozice řetězce v tabulce náhrad
Throws:
`M2BException` - je-li i menší než nula nebo větší než maximální index

---

getReplaceTo

```
public java.lang.String getReplaceTo(int i)
                              throws M2BException
```

vrací náhradu pro i-tý řetězec

Parameters:
`i` - pozice v tabulce náhrad
Throws:
`M2BException` - je-li i menší než nula nebo větší než maximální index

---

isIgnored

```
public boolean isIgnored(org.dom4j.Node n)
```

test zda má být pro tento jazyk daný uzel ignorován

Parameters:
`n` - testovaný uzel

---

hasReplacements

```
public boolean hasReplacements()
```

test, zda v nastaveních pro tento jazyk je vyplněna tabulka náhrad

Returns:
true jestliže v nastaveních pro daný jazyk je uvedena tabulka náhrad

---

cz.muni.fi.xcacek.m2b

Class M2BAppl

```
java.lang.Object
  |
  +--cz.muni.fi.xcacek.m2b.M2BAppl
```

---

public class **M2BAppl**
extends java.lang.Object

Třída M2BAppl slouží jako vstupní bod k aplikaci M2B, zajišťuje nastavení parametrů aplikace a řídí převod jednotlivých XML souborů.

Author:
Josef Cacek

Nested Class Summary

class	**M2BAppl.M2BOptions** Třída M2BOptions slouží jako datový typ záznam k uchování nastavení aplikace M2B.

Field Summary

static int	**MIN_ROW_WIDTH** minimální délka řádku pro tiskárnu Braille
static java.lang.String	**PROPERTIES_FILE** standardní cesta k souboru s nastaveními (m2b.properties)
static java.lang.String	**VERSION** verze aplikace M2B

Constructor Summary

M2BAppl(java.io.PrintWriter out) tento konstruktor nastavuje PrintWriter (zapouzdřující výstupní proud)

Method Summary

static java.lang.String	**getLang**(org.dom4j.Node n) vrací jazyk použitý (pomocí `xml:lang` atributu) v daném uzlu
M2BAppl.M2BOptions	**getOptions**() vrací nastavení aplikace

static void	**main**(java.lang.String[] args) spuštění aplikace M2B
org.dom4j.Document	**parse**(java.io.File file) tato metoda načítá XML dokument do struktury dom4j
static void	**printHelp**() tisk nápovědy pro aplikaci M2B (při použití z příkazové řádky)
static void	**printVersion**() tisk verze aplikace M2B
void	**processArguments**(java.lang.String[] args) zpracování argumentů po spuštění aplikace
void	**processFiles**() převod jednotlivých XML souborů obsahujících MathML do Braille

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

VERSION

```
public static java.lang.String VERSION
```

verze aplikace M2B

PROPERTIES_FILE

```
public static java.lang.String PROPERTIES_FILE
```

standardní cesta k souboru s nastaveními (m2b.properties)

MIN_ROW_WIDTH

```
public static int MIN_ROW_WIDTH
```

minimální délka řádku pro tiskárnu Braille

Constructor Detail

M2BAppl

```
public M2BAppl(java.io.PrintWriter out)
        throws javax.xml.transform.TransformerConfigurationException
```

tento konstruktor nastavuje PrintWriter (zapouzdřující výstupní proud)

Parameters:

`out` - PrintWriter, do kterého budou zapisovány výsledky.

Method Detail

parse

```
public org.dom4j.Document parse(java.io.File file)
                         throws org.dom4j.DocumentException,
                                java.net.MalformedURLException
```

tato metoda načítá XML dokument do struktury dom4j

Parameters:

`file` - XML soubor

Returns:

Document

Throws:

`org.dom4j.DocumentException`

java.net.MalformedURLException

---

processArguments

```
public void processArguments(java.lang.String[] args)
                     throws M2BException
```

zpracování argumentů po spuštění aplikace

Throws:
`M2BException`

---

processFiles

```
public void processFiles()
                  throws java.lang.Exception
```

převod jednotlivých XML souborů obsahujících MathML do Braille

Throws:
`java.lang.Exception`

---

main

```
public static void main(java.lang.String[] args)
```

spušťení aplikace M2B

Parameters:
`args` - pole argumentů z příkazové řádky

---

getLang

```
public static java.lang.String getLang(org.dom4j.Node n)
```

vrací jazyk použitý (pomocí `xml:lang` atributu) v daném uzlu

Parameters:
`n` - uzel u kterého chceme znát jazyk
Returns:
String obsah atributu xml:lang u daného uzlu nebo jeho nejbližšího předka obsahujícího tento atribut, není-li takový nalezen je vrácen řetězec 'general'

---

printVersion

```
public static void printVersion()
```

tisk verze aplikace M2B

---

printHelp

```
public static void printHelp()
```

tisk nápovědy pro aplikaci M2B (při použití z příkazové řádky)

---

getOptions

```
public M2BAppl.M2BOptions getOptions()
```

vrací nastavení aplikace

---

cz.muni.fi.xcacek.m2b

Class M2BException

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--cz.muni.fi.xcacek.m2b.M2BException
```

All Implemented Interfaces:
java.io.Serializable

---

public class **M2BException**
extends java.lang.Exception

Třída reprezentující výjimky v aplikaci M2B.

Author:
Josef Cacek
See Also:
Serialized Form

Constructor Summary

M2BException() požadováno ukončení aplikace a vytištění zásobníku, není specifikována konkrétní zpráva
M2BException(java.lang.String s) jako parametr jde text výjimky; požadováno ukončení aplikace a vytištění zásobníku
M2BException(java.lang.String s, boolean stop) jako parametr jde text výjimky a informace, zda se má aplikace ukončit; požadováno vytištění zásobníku
M2BException(java.lang.String s, boolean stop, boolean print) jako parametr jde text výjimky a informace, zda se má aplikace ukončit a zda má být vytištěn zásobník

Method Summary

boolean	**isPrintStack**() určuje, zda má být při této výjimce vytištěn zásobník
boolean	**isStop**() určuje, zda má být při této výjimce aplikace ukončena

Methods inherited from class java.lang.Throwable
fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

M2BException

```
public M2BException()
```

požadováno ukončení aplikace a vytištění zásobníku, není specifikována konkrétní zpráva

---

M2BException

```
public M2BException(java.lang.String s)
```

jako parametr jde text výjimky; požadováno ukončení aplikace a vytištění zásobníku

---

M2BException

```
public M2BException(java.lang.String s,
                    boolean stop)
```

jako parametr jde text výjimky a informace, zda se má aplikace ukončit; požadováno vytištění zásobníku

---

M2BException

```
public M2BException(java.lang.String s,
                    boolean stop,
                    boolean print)
```

jako parametr jde text výjimky a informace, zda se má aplikace ukončit a zda má být vytištěn zásobník

Method Detail

isStop

```
public boolean isStop()
```

určuje, zda má být při této výjimce aplikace ukončena

---

isPrintStack

```
public boolean isPrintStack()
```

určuje, zda má být při této výjimce vytištěn zásobník

---

cz.muni.fi.xcacek.m2b

Class M2BHyphen

```
java.lang.Object
  |
  +--cz.muni.fi.xcacek.m2b.M2BHyphen
```

---

public class **M2BHyphen**
extends java.lang.Object

Třída M2BHyphen zajišťuje v aplikaci M2B zarovnání výstupního textu na určenou délku řádku.

Author:
Josef Cacek

Field Summary

static java.lang.String	**WHITE_CHARS** řetězec "bílých znaků" (mezera, tabulátor, ...)

Constructor Summary

M2BHyphen(M2BAppl.M2BOptions options, int left, int right) konstruktor, na vstup dostává požadovanou délku řádku, limity pro dělení slov a nastavení aplikace

Method Summary

void	**hyph**(java.io.File fi) zpracování dočasného souboru s neděleným textem
static boolean	**isWhite**(char curChar) test zda je daný parametr "bílým znakem"

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

WHITE_CHARS

```
public static java.lang.String WHITE_CHARS
```

řetězec "bílých znaků" (mezera, tabulátor, ...)

Constructor Detail

M2BHyphen

```
public M2BHyphen(M2BAppl.M2BOptions options,
                 int left,
                 int right)
          throws M2BException
```

konstruktor, na vstup dostává požadovanou délku řádku, limity pro dělení slov a nastavení aplikace

Parameters:

`left` - minimální levý počet znaků pro rozdělení slova
`right` - minimální pravý počet znaků pro rozdělení slova
`options` - nastavení aplikace M2B

Throws:

`M2BException`

Method Detail

hyph

```
public void hyph(java.io.File fi)
          throws java.lang.Exception
```

zpracování dočasného souboru s neděleným textem

`java.lang.Exception`

---

isWhite

```
public static boolean isWhite(char curChar)
```

test zda je daný parametr "bílým znakem"

Parameters:

`curChar` - testovaný znak

---

cz.muni.fi.xcacek.m2b

Class M2BAppl.M2BOptions

```
java.lang.Object
  |
  +--cz.muni.fi.xcacek.m2b.M2BAppl.M2BOptions
```

Enclosing class:

M2BAppl

---

public class **M2BAppl.M2BOptions**
extends java.lang.Object

Třída M2BOptions slouží jako datový typ záznam k uchování nastavení aplikace M2B.

Field Summary

boolean	**bDebug** tiskni průběžně debug informace
boolean	**bHelp** vytiskni help
boolean	**bSkipAttrs** přeskoč zpracování atributů

boolean	**bSkipNumbers** přeskoč zpracování čísel
boolean	**bSkipReplace** přeskoč zpracování tabulky náhrad
boolean	**bSkipUpper** přeskoč zpracování velkých písmen
boolean	**bTrimSpaces** ořež mezery
boolean	**bVBNumbers** nenahrazuj číslice písmeny (ale přidej prefix pro čísla)
boolean	**bVBUpper** nenahrazuj velká písmena malými (ale přidej prefix pro velká písmena)
boolean	**bVersion** vytiskni verzi
java.io.PrintWriter	**out** writer zapouzdřující výstupní proud
java.lang.String	**propFile** cesta k souboru s nastaveními (m2b.properties)
int	**rowWidth** požadovaná délka řádku pro tiskárnu Braille
java.lang.String[]	**sFiles** pole souborů, které mají být zpracovány
java.lang.String	**xslFile** cesta k transformačnímu souboru (m2b.xsl)

Constructor Summary

M2BAppl.M2BOptions()

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

bHelp

public boolean **bHelp**

vytiskni help

---

bVersion

public boolean **bVersion**

vytiskni verzi

---

bDebug

public boolean **bDebug**

tiskni průběžně debug informace

---

bTrimSpaces

public boolean **bTrimSpaces**

ořež mezery

---

bVBNumbers

```
public boolean bVBNumbers
```

nenahrazuj číslice písmeny (ale přidej prefix pro čísla)

---

bVBUpper

```
public boolean bVBUpper
```

nenahrazuj velká písmena malými (ale přidej prefix pro velká písmena)

---

bSkipNumbers

```
public boolean bSkipNumbers
```

přeskoč zpracování čísel

---

bSkipUpper

```
public boolean bSkipUpper
```

přeskoč zpracování velkých písmen

---

bSkipReplace

```
public boolean bSkipReplace
```

přeskoč zpracování tabulky náhrad

---

bSkipAttrs

```
public boolean bSkipAttrs
```

přeskoč zpracování atributů

---

propFile

```
public java.lang.String propFile
```

cesta k souboru s nastaveními (m2b.properties)

---

xslFile

```
public java.lang.String xslFile
```

cesta k transformačnímu souboru (m2b.xsl)

---

sFiles

```
public java.lang.String[] sFiles
```

pole souborů, které mají být zpracovány

---

out

```
public java.io.PrintWriter out
```

writer zapouzdřující výstupní proud

---

rowWidth

```
public int rowWidth
```

požadovaná délka řádku pro tiskárnu Braille

Constructor Detail

M2BAppl.M2BOptions

```
public M2BAppl.M2BOptions()
```

---

cz.muni.fi.xcacek.m2b

Class TextProcessor

```
java.lang.Object
  |
  +--cz.muni.fi.xcacek.m2b.TextProcessor
```

---

public class **TextProcessor**
extends java.lang.Object

Třída zajišťující dodatečnou funkcionalitu k XSL transformacím v aplikaci M2B. Má na starosti převod velkých písmen, čísel a zpracovává tabulku náhrad určenou elementem `replacements`.

Author:
Josef Cacek

Field Summary

static java.lang.String	**REGEX_DIRTY_CHARS** znaky u nichž požadujeme potlačení speciálního významu v regulárních výrazech

Constructor Summary

TextProcessor(org.dom4j.Document prop, M2BAppl.M2BOptions opts) vytváří sadu objektů LanguageSettings ze souboru m2b.properties

Method Summary

java.lang.String	**correctRegexString**(java.lang.String s) vkládá zpětné lomítko před znaky, které by mohly způsobit nesprávnou interpretaci v metodách jako je String.replaceAll() (field REGEX_DIRTY_CHARS)
static LanguageSettings	**getGeneralLS**() vrací nastavení pro obecnou skupinu - general
java.lang.String	**getText**() vrací právě zpracovávaný text
void	**process**() řízení překladu do Braille
void	**processMFenced**() převod elementu `mfenced` je zajišťován touto metodou, protože XSL transformace neposkytují dostatečné možnosti pro práci s textem
void	**setNode**(org.dom4j.Node n) nastavení uzlu ke zpracování

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

REGEX_DIRTY_CHARS

```
public static java.lang.String REGEX_DIRTY_CHARS
```

znaky u nichž požadujeme potlačení speciálního významu v regulárních výrazech

Constructor Detail

TextProcessor

```
public TextProcessor(org.dom4j.Document prop,
                     M2BAppl.M2BOptions opts)
              throws M2BException
```

vytváří sadu objektů LanguageSettings ze souboru m2b.properties

Parameters:

`prop` - dom4j reprezentace dokumentu s nastaveními (m2b.properties)

`opts` - nastavení aplikace M2B

Throws:

`M2BException`

Method Detail

setNode

```
public void setNode(org.dom4j.Node n)
```

nastavení uzlu ke zpracování

---

getText

```
public java.lang.String getText()
```

vrací právě zpracovávaný text

---

process

```
public void process()
             throws M2BException
```

řízení překladu do Braille

Throws:

`M2BException`

---

correctRegexString

```
public java.lang.String correctRegexString(java.lang.String s)
```

vkládá zpětné lomítko před znaky, které by mohly způsobit nesprávnou interpretaci v metodách jako je String.replaceAll() (field REGEX_DIRTY_CHARS)

Parameters:

`s` - řetězec ve kterém chceme nahradit výskyty znaků z REGEX_DIRTY_CHARS

Returns:

String

---

processMFenced

```
public void processMFenced()
                    throws M2BException
```

převod elementu `mfenced` je zajišťován touto metodou, protože XSL transformace neposkytují dostatečné možnosti pro práci s textem

Throws:

`M2BException`

---

getGeneralLS

```
public static LanguageSettings getGeneralLS()
```

vrací nastavení pro obecnou skupinu - general

---

cz.muni.fi.xcacek.m2b

Class Web

```
java.lang.Object
  |
  +--javax.servlet.GenericServlet
        |
        +--javax.servlet.http.HttpServlet
              |
              +--cz.muni.fi.xcacek.m2b.Web
```

All Implemented Interfaces:

java.io.Serializable, javax.servlet.Servlet, javax.servlet.ServletConfig

---

public class **Web**
extends javax.servlet.http.HttpServlet

Webové rozhraní aplikace M2B. V některých případech nahrazuje třídu M2BAppl, jindy s ní spolupracuje.

Author:

Josef Cacek

See Also:

Serialized Form

Constructor Summary

Web()

Method Summary

void	**doGet**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) obsluhuje požadavky, které přišly pomocí metody GET (redirect na úvodní stránku - platné jsou pouze požadavky podané pomocí metody POST)
void	**doPost**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) obsluhuje požadavky, které přišly pomocí metody POST; obsahuje výkonnou část servletu
void	**init**(javax.servlet.ServletConfig _config) při inicializaci servletu zjistí cestu k webové aplikaci (později bude použita ke správné lokalizaci souborů m2b.xsl a m2b.properties)

Methods inherited from class javax.servlet.http.HttpServlet

service

Methods inherited from class javax.servlet.GenericServlet

destroy, getInitParameter, getInitParameterNames, getServletConfig, getServletContext, getServletInfo, getServletName, init, log, log

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Web

```
public Web()
```

Method Detail

doGet

```
public void doGet(javax.servlet.http.HttpServletRequest request,
```

```
                javax.servlet.http.HttpServletResponse response)
         throws java.io.IOException
```

obsluhuje požadavky, které přišly pomocí metody GET (redirect na úvodní stránku - platné jsou pouze požadavky podané pomocí metody POST)

Overrides:

`doGet` in class `javax.servlet.http.HttpServlet`

Throws:

`java.io.IOException`

---

doPost

```
public void doPost(javax.servlet.http.HttpServletRequest request,
                   javax.servlet.http.HttpServletResponse response)
            throws java.io.IOException
```

obsluhuje požadavky, které přišly pomocí metody POST; obsahuje výkonnou část servletu

Overrides:

`doPost` in class `javax.servlet.http.HttpServlet`

Throws:

`java.io.IOException`

---

init

```
public void init(javax.servlet.ServletConfig _config)
```

při inicializaci servletu zjistí cestu k webové aplikaci (později bude použita ke správné lokalizaci souborů m2b.xsl a m2b.properties)

Specified by:

`init` in interface `javax.servlet.Servlet`

Overrides:

`init` in class `javax.servlet.GenericServlet`

Příloha C

GNU Free Documentation License

GNU Free Documentation License
Version 1.2, November 2002

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document ”free” in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of ”copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The ”Document”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as ”you”. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A ”Modified Version” of the Document means any work containing the

Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A ”Secondary Section” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The ”Invariant Sections” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The ”Cover Texts” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A ”Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not ”Transparent” is called ”Opaque”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, \LaTeX{} input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The ”Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, ”Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

A section ”Entitled XYZ” means a named subunit of the Document whose

title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as ”Acknowledgements”, ”Dedications”, ”Endorsements”, or ”History”.) To ”Preserve the Title” of such a section when you modify the Document means that it remains a section ”Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document’s license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an

Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
D. Preserve all the copyright notices of the Document.
E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document’s license notice.
H. Include an unaltered copy of this License.
I. Preserve the section Entitled ”History”, Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled ”History” in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the ”History” section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
K. For any section Entitled ”Acknowledgements” or ”Dedications”, Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
M. Delete any section Entitled ”Endorsements”. Such a section

   may not be included in the Modified Version.
N. Do not retitle any existing section to be Entitled ”Endorsements” or to conflict in title with any Invariant Section.
O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.

You may add a section Entitled ”Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled ”History” in the various original documents, forming one section Entitled ”History”; likewise combine any sections Entitled ”Acknowledgements”, and any sections Entitled ”Dedications”. You must delete all sections Entitled ”Endorsements”.

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents

released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an ”aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled ”Acknowledgements”, ”Dedications”, or ”History”, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License ”or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.