Dokumentace úlohy CSV: CSV2XML v jazyce Perl do IPP 2011/2012

Jméno a příjmení: Jan Bednařík
Login: xbedna45

1 Úvod

Tento dokument popisuje práci na implementaci skriptu, jenž převádí vstupní textový soubor z formátu CSV (podle standardu [1]) do formátu XML 1.0 (podle standardu [2]).

2 Návrh

Z důvodu přehlednosti kódu je vhodné rozdělit funkcionalitu celého skriptu do několika logických celků, jež jsou implementovány formou nezávislých funkcí. Jedná se o zpracování parametrů skriptu, načtení a zpracování jednoho řádku ze vstupního CSV souboru, zápis načtených dat do výstupního XML souboru a výběr a výpis případného chybového hlášení.

3 Zpracování parametrů

Zadání umožňuje využití modulu `Getopt::Long`. Po prostudování dokumentace a provedení několika experimentů jsem však zjistil, že dostupné funkce nelze použít tak, aby skript přesně odpovídal požadavkům zadání, tudíž parametry zpracovávám pomocí vlastních funkcí `GetOpts` a `VerifyParams`.

Informace o parametrech jsou uchovány v proměnné typu hash. Funkce `GetOpts` pro bezhodnotové parametry pouze definuje položku a pro hodnotové parametry uloží hodnotu do struktury hash.

Funkce `VerifyParams` následně kontroluje povolené kombinace parametrů a ověřuje validitu zadaných hodnot.

4 Čtení ze vstupního CSV souboru

Pro získání a zpracování dat ze vstupního CSV souboru využívám částečně vlastní funkce a částečně metody, jež implementuje třída modulu `Text::CSV 1.16`. Přestože třída `Text::CSV` nabízí dostatek potřených metod pro načtení i zpracování vstupního CSV souboru, v některých okrajových případech se chování metod poměrně zásadně rozchází se specifikacemi CSV souboru podle standardu [1], tudíž jsem musel využít i vlastní funkce. Dobrým příkladem je případ, kdy jsou řádky vstupního CSV souboru ukončeny znakem LF namísto dvojice CR LF, jak vyžaduje standard. Tehdy by se mělo jednat o nekorektní formát CSV souboru, avšak příslušná metoda `getline` řádek bezostyšně načte a nehlásí chybu. Nutno podotknout, že novější verze modulu `Text::CSV 1.21` se již chová korektně, avšak bohužel není nainstalována na školním serveru Merlin.

Pro získání jednoho vstupního řádku CSV souboru (ukončeného dvojicí bílých znaků CR LF) tedy implementuji vlastní funkci `GetCSVLine`. Kritickým prvkem funkce je zjištění, zda není poslední sloupec (myšleno z hlediska formátu CSV) načteného řádku rozdělen tak, že pokračuje na řádku následujícím. K tomuto případu může dojít tehdy, pokud se vyskytne dvojice bílých znaků CR LF uvnitř pole ohraničeného uvozovkami. Ošetření tohoto případu není zcela triviální, neboť se v rámci pole mohou vyskytovat také uvozovky, jejichž zvláštní význam je escapován dalšími předřazenými uvozovkami. Po neúspěšném koketování s poměrně složitým regulárním výrazem jsem se rozhodl problém řešit počítáním uvozovek na načteném řádku, přičemž je o připojení či nepřipojení následujícího řádku rozhodnuto podle sudého či lichého počtu výskytu uvozovek. Rozdělení načteného řádku na jednotlivá pole je následně realizováno metodou `parse` třídy `Text::CSV`.

Pakliže má první řádek vstupního CSV souboru představovat jména řádkových tagů výstupního XML souboru (tedy skript je spuštěn s parametrem `-h`), je nutné ověřit, že jednotlivé sloupce prvního řádku obsahují pouze znaky, jež připouští standard [2], a případně je nahradit. Pro tento úkon impelmentuji funkci `ValidateTags`.

Nastavení třídy `Text::CSV`:

```
binary => 1,
eol => "\r\n",
sep_char => načtení zadané hodnoty,
verbatim => 1
```

Popis parametrů:

- `binary`: Povolí výskyt některých znaků (jako CR a LF) v poli ohraničeném uvozovkami.
- `eol`: Nastavení znaků reprezentujících odřádkování.
- `sep_char`: Nastavení oddělovače polí.
- `verbatim`: Metoda `parse` nebude uvažovat zpeciální význam znaků CR a LF.

5 Zápis do výstupního XML souboru

Po každém načtení řádku ze vstupního CSV souboru a extrakci polí je nutné získaná data zapsat do výstupního XML souboru. K tomuto účelu využívám metody třídy modulu `XML::Writer`. Konkrétně se jedná o metodu `startTag` pro zápis otevíracího tagu, metodu `characters`, jež zapisuje obsah mezi otevírací a zavírací tag a stará se o převod problematických znaků (&, < a >), a metodu `endTag` pro zápis zavíracího tagu.

Nastavení třídy `XML::Writer`:

```
DATA_MODE => 1,
DATA_INDENT => 2,
UNSAFE => 1
```

Popis parametrů:

- `DATA_MODE`: Formátování výstupu - vkládá odřádkování za každý tag.
- `DATA_INDENT`: Formátování výstupu - hierarchické odsazování tagů.
- `UNSAFE`: Vypnutí kontroly korektnosti výstupního XML dokumentu (některé soubory nemusí mít kořenový tag).

6 Testování

Pro otestování skriptu jsem si připravil sadu automatizovaných testů, jež spouští skript s různými parametry, kontrolují návratový kód a porovnávají výstupní XML soubor s referenčními XML soubory za pomocí nástroje A7Soft JExamXML [3]. Referenční XML souboru jsem získal převodem vstupních CSV souborů za pomocí online nástroje Creativyst CSV to XML Converter v2.0c [4] a dodatečných ručních úprav.

Reference

[1] Y. Shafranovich: RFC 4180 - Common Format and MIME Type for Comma-Separated Values (CSV) Files [online]. 2005 [cit. 2012-03-31]. Dostupné na http://tools.ietf.org/html/rfc4180.

[2] Extensible Markup Language (XML) 1.0. W3C. World Wide Web Consortium [online]. 5. vydání. 26. 11. 2008 [cit. 2012-03-31]. Dostupné na http://www.w3.org/TR/xml/.

[3] A7Soft JExamXML is a java based command line XML diff tool for comparing and merging XML documents [online]. c2012 [cit. 2012-03-31]. Dostupné na: http://www.a7soft.com/jexamxml.html

[4] Creativyst Software [online]. 2011 [cit. 2012-03-31]. Dostupné na: http://www.creativyst.com/cgi-bin/Prod/15/eg/csv2xml.pl