Dokumentace úlohy JSN: JSON2XML v PHP do IPP 2013/2014
Jméno a příjmení: David Kašpar
Login: xkaspa34

1 Úvod

V této dokumentaci je řešena implementace úlohy JSN v předmětu IPP[1], na *Fakultě informačních technologií Vysokého učení technického v Brně*, pro akademický rok 2013/2014. Cílem této úlohy bylo vytvořit skript ve skriptovacím jazyku PHP[2] (verze 5.3.3), který převádí vstupní soubor v serializačním formátu JSON[3] na adekvátní výstupní soubor v serializačním formátu XML[4] (verze 1.0).

1.1 Zdroje informací

K počátečnímu obeznámení s formáty JSON a XML byly použity články na Wikipedii[5][6]. Tyto články však neobsahovaly všechny potřebné informace k úspěšnému zvládnutí projektu, proto bylo dále zapotřebí nastudovat všechny potřebné relevantní informace, a to ze standardů publikovaných mezinárodní organizací *World Wide Web Consortium* (W3C)[3][4].

Pro úvodní seznámení se skriptovacím jazykem PHP pak byly využity tutoriály a referenční příručky z webových stránek w3schools.com[7][8] a oficiálních webových stránek PHP[2].

V případě řešení některých sporných bodů a nejasností bylo využíváno privátní fórum předmětu[1].

1.2 Prostředí a testování

Jako referenční prostředí pro testování a následné hodnocení byl zadáním vybrán školní server Merlin[9]. Jedná se o linuxovou distribuci *CentOS*, čemuž byly dále přizpůsobeny další formální požadavky na výsledný skript, jako například vstupní i výstupní kódování v UTF-8, standardní Unixové konce řádků ‘\n‘, apod.

Společně se zadáním byla zveřejněna základní sada testů, která představovala možnosti použití skriptu, použití jeho parametrů, jednotlivých vstupů a k nim odpovídajících výstupů. Jelikož v dané sadě testů nebyly pokryty všechny možnosti předpokládaného chování skriptu, byla vytvořena tzv. *test suite* (známa taktéž jako *validation suite*). Jedná se o větší sadu testů, která se snaží pokrývat naprostou většinu aspektů a možností použití skriptu, za účelem validace vytvořeného návrhu a samotné verifikace chování skriptu. Tato *test suite* je online veřejně přístupná jako Git repositář[10], obsahuje přes 100 testů a splňuje požadavky na kompatibilitu s linuxovým prostředím serveru Merlin.

2 Rozbor zadání a návrhu řešení

Zadání vyžaduje, aby výsledný skript měl podobu tzv. textového filtru příkazové řádky. Jinými slovy, musí být schopen načítat vstupní data ze standardního vstupu a výsledek svého konání vypisovat na standardní výstup. To umožní přidávat skript do Unixových kolon (zřetězit ho) pomocí rour (angl. *pipelines*).

Skript musí být naprosto soběstačný s využitím pouze standardních a povolených knihoven. Je zakázáno spouštět jakékoliv další vlákna/procesy nebo příkazy operačního systému. Při jakékoliv chybě se všechny chybové hlášení musejí vypisovat na standardní chybový výstup a skript musí skončit s patřičným chybovým kódem, který je určen zadáním.

2.1 Převod formátu JSON na formát XML

JSON i XML jsou serializační formáty a díky tomu jsou i navzájem převoditelné. Obecný postup můžeme shrnout do těchto jednotlivých bodů:

1. Načtení vstupních dat a jejich validace.
2. Separování jednoho základního typu (angl. *basic types*) JSON formátu ze vstupních dat.
3. Aplikování filtrů na obsah momentálně zpracovávaného základního typu, a to podle aktuálního nastavení skriptu. (Toto nastavení se provádí použitím příslušných parametrů skriptu.)
4. Tisk obsahu zpracovaného základního typu ve formátu XML. Tento formát se odvíjí od aktuálního nastavení skriptu. (Nastavení lze opět modifikovat použitím příslušných parametrů skriptu.)
5. Přechod na bod číslo 2, pokud skript ještě nedošel na konec vstupních dat.

Z důvodu libovolného vnořování základních typů JSON formátu se přirozeně nabízí zpracovávat tyto typy pomocí postupného rekurzivního zanořování. Podle výše uvedeného postupu byl vytvořen i algoritmus pro výsledný skript. Hodí se také poznamenat, že body č. 1 a č. 2 lze případně sloučit do jednoho.

3 Způsob řešení

Jelikož se autor projektu se skriptovacím jazykem PHP ještě nikdy nesetkal, bylo po rozboru zadání rozhodnuto řešit projekt pomocí čistě *procedurálního paradigmatu*, a to z 2 následujících důvodů:

1. *Objektově orientované paradigma* má strmější křivku náročnosti učení, jelikož je potřeba si nastudovat většinu zvláštností pro daný jazyk, stejně jako jmenné prostory, třídy, apod.
2. Projekt není nijak rozsáhlý a při dodržování určitých konvencí správného programování jej lze snadno vyřešit i za použití procedurálního paradigmatu, se kterým má autor již značné zkušenosti.

3.1 Vybrané části

Zde jsou uvedeny některé části výsledného skriptu s popisem jejich chování nebo zvláštností:

- o analýzu a převod JSON formátu na reprezentaci pomocí interních datových typů jazyka PHP se stará standardní funkce PHP zvaná `json_decode()`, která vrací objekt reprezentující daný vstup ve formátu JSON
- celý vstupní soubor je načítán do datového typu `string` pomocí funkce `file_get_contents()`, protože výše zmíněná funkce `json_decode()` pro svoji práci tento `string` vyžaduje

  (Určitě se však nejedná o nejvhodnější řešení. Pokud by uživatel zadával na vstup data o velikosti větší než pár desítek megabajtů, začal by se skript chovat jako tzv. *bottleneck* – čekal by na načtení všech dat ze vstupu, a až poté by provedl analýzu a tisk výsledku na výstup. Pro řešení této situace by bylo zapotřebí vytvořit vlastní analyzátor vracející právě jeden zanalyzovaný základní typ. Jeho obsah by se okamžitě zpracovával a tisknul na výstup, což by *bottleneck* eliminovalo.)
- kompletní algoritmus převodu dekódovaných JSON dat na odpovídající XML výstup provádí funkce `json2xml()`, která v případě vnoření základních typů JSON volá rekurzivně sama sebe
- pro tisk obsahu XML elementů se využívá sada funkcí začínající názvem `xml_print_`, které mimo jiné provádí i modifikaci výstupního formátu podle aktuálního nastavení skriptu
- jelikož standardní PHP funkce `getopt()` neumožnovala potřebnou analýzu parametrů skriptu (požadovanou zadáním), byla nahrazena funkcí `process_params()` – ta umožňuje důkladnější analýzu a ošetření případných chybových stavů, stejně tak jako nastavení chování skriptu

4 Rozšíření

V rámci projektu bylo naimplementováno jednoduché rozšíření **`JPD`**, které doplňuje zleva nuly všem použitým čítačům tak, aby měly v dané části stejnou minimální šířku. Toto rozšíření se zapíná přepínačem `--padding`. Nad rámec zadání projektu byl navíc naimplementován přepínač `--offset-size=N`, kde `N` určuje počet mezer, které se mají použít pro odsazení výsledného XML formátu. Implicitní jsou 4 mezery.

Reference

[1] Principy programovacích jazyků a OOP, 2014. [Online]
https://www.fit.vutbr.cz/study/courses/index.php?id=9384

[2] PHP: Hypertext Preprocessor, 2014. [Online]
http://php.net/

[3] JSON-LD 1.0, 2014. [Online]
http://www.w3.org/TR/json-ld/

[4] Extensible Markup Language (XML) 1.0 (Fifth Edition), 2014. [Online]
http://www.w3.org/TR/REC-xml/

[5] JSON – Wikipedia, 2014. [Online]
http://en.wikipedia.org/wiki/JSON

[6] XML – Wikipedia, 2014. [Online]
http://en.wikipedia.org/wiki/XML

[7] PHP 5 Tutorial, 2014. [Online]
http://www.w3schools.com/php/default.asp

[8] PHP 5 Reference, 2014. [Online]
http://www.w3schools.com/php/php_ref_array.asp

[9] Server of Faculty of Information technology, Brno University of Technology, 2014. [Online]
http://merlin.fit.vutbr.cz/

[10] BUT FIT - IPP 2014 - JSN testsuite (PHP version), 2014. [Online]
https://bitbucket.org/deekej/but-fit-ipp-2014-jsn-testsuite-php-version/