Dokumentace úlohy JSN: JSON2XML v PHP 5 do IPP 2016/2017
Jméno a příjmení: Marek Šipoš
Login: xsipos03

1 Analýza požadavků

Zadáním úlohy byla tvorba skriptu v jazyce PHP, který by umožnil načíst vstupní soubor[1] ve formátu JSON a výstupem by byl dokument ve formátu XML tak, aby byla struktura dokumentu XML validní, a který by poskytoval funkcionalitu definovanou zadáním (např. různé přepínače pro měnění chování skriptu).

1.1 Výběr nástrojů pro splnění požadavků

Jako u většiny podobných úloh, i zde bylo třeba si důkladně rozmyslet způsob načítání parametrů. Drtivou většinu formálních požadavků splňoval vhodně nakonfigurovaný nástroj `getopt`, který umožňuje přímo zadat, které parametry se mají načíst a jaké očekávají vstupy.

Před samotným zpracováním bylo nejdříve potřeba načíst dokument typu JSON do vnitřní reprezentace jazyka PHP. Protože formát JSON je velmi striktní co se chybovosti týče a tudíž netriviální na implementaci jeho načítání, jevilo se jako dobrá volba využití základního rozšíření jazyka PHP pro JSON a jeho funkce `json_decode`, která převede řetězec formátovaný jako JSON soubor a vytvoří z něj pole s jeho vnitřní strukturou.

Další otázka vyvstala při uvážení způsobu tvorby samotné struktury XML. V jazyce PHP je k tomuto účelu dostupná celá řada knihoven, z nichž několik velmi užitečných bylo povoleno i v zadání úlohy. Pro jednoduché sázení jednotlivých elementů a atributů výstupu bylo využito nástroje `SimpleXMLElement`, který poskytuje možnost tvorby struktury XML voláním metod jednotlivých objektů[2], čímž poměrně usnadňuje řešení formálních požadavků formátu XML a umožňuje soustředit se na úlohu samotnou. Nakonec se tento nástroj osvědčil jen částečně a vyskytly se různá omezení (viz část 2.4 na straně 2).

Protože nástroj `SimpleXMLElement` neumožňuje tvorbu dokumentu XML s vhodným odsazením jednotlivých elementů podle úrovně zanoření, bylo pro čistě vizuální stránku věci využito i rozšíření `DOM`, které umožňuje před exportem aktivovat přepínač `formatOutput`, který se o odsazení sám postará.

2 Implementace

Následující část popisuje samotný způsob implementace jednotlivých částí projektu, od parametrů až po různé přepínače definující detaily formátu výstupu.

2.1 Načítání parametrů

Zadání úlohy popisuje spoustu rozličných parametrů upravujících formát dokumentu XML a nebo definujících např. cestu ke vstupnímu/výstupnímu souboru. Některé parametry obsahují také zkrácený alias (ve tvaru -n) a parametry jsou také různorodé v tom, jak se mohou navzájem kombinovat. Navíc existují parametry se vstupní hodnotou i bez ní.

Protože nástroj `getopt` vyžaduje řetězec s krátkými parametry a pole s dlouhými parametry, byly tyto vstupy vytvořeny ještě před jeho zavoláním a zároveň byly stručně v komentářích popsány parametry samotné, což poskytovalo přehled o parametrech při práci. Všechny zpracované parametry se pak ukládaly do globálního[3] asociativního pole `$arg`. Pro zjednodušení práce bylo vytvořeno i pole parametrů, které vyžadují nějakou hodnotu. Všechny přijaté parametry funkcí `getopt` se zpracovaly cyklem `foreach`, kde se ověřovalo, jestli je parametru zadána hodnota tehdy a právě tehdy, když ji vyžaduje a zda se nějaký parametr nezadal vícekrát, zadání to totiž neumožňovalo. Samozřejmě bylo nutné i detekovat případné neexistující parametry. Poté se kontrolovaly nepovolené kombinace parametrů a další formální požadavky (např. formát vstupní hodnoty parametru `--start`).

2.2 Práce se soubory

Zajímavým je způsob práce se soubory v jazyce PHP. Ternárním operátorem se vždy nejprve zajistilo, že se bude v závislosti na parametrech `--input` a `--output` načítat/ukládat buď ze/do souboru, nebo přímo ze/do standardního vstupu/výstupu.

Protože se při práci se soubory může stát leccos, bylo nutné prověřit jednotlivé možnosti jazyka při práci s nimi a způsob ošetření chybových stavů. Nakonec se využilo řešení pomocí funkcí `set_error_handler` a `restore_error_handler`. Tyto funkce se volají před resp. po použití nějaké souborové funkce a zajišťují přesměrování veškerých upozornění nebo chyb do vlastní funkce, ve které se mohou tyto chybové stavy zpracovat. Připomíná to trochu zpracování výjimek, avšak procedurálně. Ve funkcích `input_chyba` a `output_chyba` byly chybové stavy poté zpracovány a tím se zároveň zajistilo, že „neprotečou“ ke koncovému uživateli.

[1] nebo přímo standardní vstup
[2] v tomto případě byly objekty přímo jednotlivé elementy dokumentu
[3] důvod využití globální proměnné je jednoduchý přístup k parametrům v pomocných funkcích

Samotné zpracování souborů se řešilo tak, že byly načteny a poté uloženy do a z řetězce. Na to posloužily funkce `file_get_contents` a `file_put_contents`, které zajistily i případné přepsání a vytvoření souboru, stejně jako správné zpracování relativních cest tak, jak to požadovalo zadání. Načtení a uložení celého souboru do/z řetězce je v PHP kupodivu efektivnější, než použití souborového deskriptoru a postupnou prací se souborem. Je to i mnohem jednodušší.

2.3 Načtení a zpracování struktury JSON

Soubor formátu JSON se nejprve načetl funkcí `json_decode`, která se stará o formální náležitosti a správnost vstupu a poskytuje zpracovaný soubor jako PHP strukturu (formálně se jedná o pole, které může obsahovat další pole[4]). Tato struktura se pak zpracovává rekurzivně pomocí funkcí `xml_zpracuj_pole`, `xml_zpracuj_objekt` a `xml_zpracuj_hodnotu`, které pracovaly a volaly se v závislosti na typu elementů, na které narazily. Důvod rozdělení do tří funkcí je přehlednost a zefektivnění organizace, každá funkce dělá právě tu svou činnost a může volat jiné `xml_zpracuj_...` funkce, kterým předá referenci na podřazenou XML instanci, kterou je třeba zpracovat. Tím se rekurzivně projde celá struktura.

2.4 Vytvoření výsledného XML dokumentu

Během procházení JSON struktury v PHP reprezentaci se volaly jednotlivé metody objektu `SimpleXMLElement`, jako např. metody pro vytvoření podřazeného elementu nebo přidání atributu. Tady se ukázala síla tohoto rozšíření, kdy při rekurzivním procházení JSON struktury bylo možné do vlastních pomocných funkcí předávat reference na podřazené elementy, které se pro ty funkce tvářily jako elementy kořenové. V `SimpleXMLElement` je každý element tvořeného XML dokumentu reprezentován právě instancí, na které lze vytvářet reference. A v tom tkví síla tohoto rozšíření.

Zjednodušení práce díky tomuto rozšíření je však draze zaplaceno tím, že neexistuje prakticky žádná možnost si výstup upravit k obrazu svému. Například výstupní XML dokument (exportovaný do řetězce) se nachází celý na jednom řádku, bez jakéhokoliv odřádkování nebo odsazování podle hloubky zanoření. Rozšíření také neumožňovalo vynechat XML hlavičku ani kořenový element[5], ačkoliv to zadání úlohy vyžadovalo[6], takže byla nutná konečná úprava. Skript byl „prohnán“ rozšířením `DOM` a tím se zajistilo správné odsazení, poté byl XML dokument exportován do řetězce metodou `saveXML` a pak byla použita funkce `explode` podle nového řádku (multiplatformní konstanty `PHP_EOF`) a odstraněny nežádoucí řádky. Dalším nedostatkem bylo automatické převádění nežádoucích znaků textů atributů na jejich HTML entity bez možnosti vypnutí tohoto chování. To se vyřešilo případným zavoláním funkce `htmlspecialchars_decode`, která tento proces zvrátila zpět.

3 Možná vylepšení

Největším problémem, který se během projektu vyskytl, byl zajisté nemožnost kontroly nad tím, jakým způsobem rozšíření `SimpleXMLElement` výsledný XML dokument tvoří. Vzhledem k tomu, kolik přepínačů zadání vyžadovalo, bylo tristní, že toto rozšíření neumožňovalo prakticky žádnou konfiguraci.

V budoucí verzi by bylo zajisté vhodné generování XML dokumentu dělat ručně (a tím získat plnou kontrolu), nebo využít jiného rozšíření či knihovny, které vyhovují zadání a nevyžadují dodatečnou úpravu.

[4] tato pole mohou být asociativní, podle toho, zda se jednalo v JSON o pole nebo objekt

[5] tím by totiž vznikl nevalidní XML dokument

[6] hlavička i kořenový element šly vynechat specifickým použitím parametrů skriptu