Dokumentace úlohy JSN: JSON2XML v Perl do IPP 2011/2012
Jméno a přijmení: Pavol Loffay
Login: xloffa00

1.1 Úvod

Tento dokument stručně popisuje převod z formátu JSON[1] do XML[2] v programovacím jazyce *Perl*, s využitím níže uvedených knihoven. Dále popisuje hlavní problémy, se kterými jsem se při implementaci setkal.

1.2 Návrh řešení problému

Hlavním implementačním problémem bylo správné načítání dat ve formátu JSON a identifikování datových struktur. Následná konverze do formátu XML byla jednoduchá. Po analýze problému jsem se rozhodl použít knihovny pro práci s daty ve formátu JSON a XML. Protože skript můžeme spouštět s různými parametry, použil jsem knihovnu `Getopt::Long`, která zpracování parametrů značně ulehčuje.

1.3 Zpracování vstupního souboru

Pro korektní načítání vstupního textu jsem použil knihovnu `JSON::XS`. Jednou z její výhod je, že automaticky kontroluje správnost formátu vstupních dat. Proto je nebylo potřebné explicitně ověřovat. Po načtení celého vstupu do proměnné, bylo zapotřebí identifikovat datové struktury JSON. Pokud byl načten objekt, tak v proměnné byl uložen jako hash. Pokud bylo načtené pole, bylo uloženo opět do pole. Ostatní prvky bylo možno rozlišit jednoduchým způsobem pomocí větvení programu.

1.3.1 Rozlišení řetězce od čísla

Skript má parametry, které umožňují transformaci hodnot dvojic typu `string` nebo `number` z atributů na textové elementy. Proto bylo zapotřebí tyto dva typy explicitně rozlišovat. K tomu jsem použil modul `Data::Dumper`, který v případe ak premňenná byla typu `string` tak k ní přidá uvozovky.

1.3.2 Identifikování `true`, `false`, `null`

Zvolením správnych parametrů skriptu je možné docílit, aby hodnoty literálů `true`, `false` a `null` byly transformovány na elementy `<true/> <false/> <null/>`. Proto bylo nutné rozlišovat tyto hodnoty, které *Perl* nerozlišuje. Hodnoty typu `Bool` bylo možné identifikovat pomocí funkce `JSON::XS::is_bool`, `null` jsem rozlišil funkcí `defined`.

1.4 Vytváření výstupu

Pro vytvoření výstupního XML souboru jsem implementoval modul `xml_writer`. Obsahuje metody pro vypisování **start, end** a **empty** XML značek. Tyto metody mi umožnili efektivně vytvářet výsledný soubor.

1.5 Kontrola správnosti XML značky

Pro kontrolu XML značky bylo nutné vytvořit netriviální regulární výraz, který dokázal rozpoznat nevalidní XML značku. Potom se dalším regulárním výrazem nahradili všechny nepovolené znaky za `-`. Poté byla opět provedena kontrola validnosti značky.

1.6 Převod JSON do XML

Samotná konverze formátu probíhá ve funkci `json2xml`, která se volá rekurzivně. Vstupu je načten do proměnné a následně procházen v této funkci. Dále následuje rozpoznávání typu proměnné, která může být hash, pole atd.. Pokud byla typu hash, nebo pole je funkce pro každou položku opět volaná rekurzivně. Vyžitím rekurze se daly jednoduchým způsobem identifikovat veškeré zanořené datové struktury formátu JSON.

[1] Popsán v RFC 4627
[2] Dostupné na http://www.w3.org/TR/REC-xml/

1.7 Rozšíření JPD

Implementace tohoto rozšíření spočívala v tom, že bylo zapotřebí zjistit minimální možný počet nul, které bylo zapotřebí přidat ke každému indexu tak, aby měly všechny indexy stejný počet cifer. Nejprve bylo nutné zjistit kolik cifer bude mít poslední index pole, následně se pak vypočítal počet cifer aktuálního indexu a provedla se korekce přidáním správného počtu nul.