Dokumentace úlohy JSN: JSON2XML v PHP5 do IPP 2016/2017

Jméno a příjmení: Michal Mrnuštík

Login: xmrnus01

Popis řešení

Řešení je rozděleno do 3 logických celků: zpracování argumentů, práce se soubory a samotný převod JSON objektu do XML.

Každý tento celek má své umístění v adresáři lib. V `./lib/arguments/` najdeme třídy odpovědné za práci s argumenty. V `./lib/io/` najdeme třídy odpovědné za práci se soubory. V `./lib/io/parser/` je třída, která provádí převod mezi formáty. Uvnitř těchto adresářů také můžeme najít složky exception, ve kterých se nachází třídy typu výjimka, které jsou vyhazovány při chybových stavech.

Samotný skript `jsn.php` provádí pouze `include` jednotlivých souborů a volá veřejné metody odpovídajících tříd.

Zpracování argumentů

Pro udržení informací o argumentech z příkazové řádky jsme se rozhodli pro vytvoření objektu `Arguments`, který bude zastřešovat všechny informace, které byly programu předány. Při návrhu tohotu objektu jsme vybrali návrhový vzor singleton (česky jedináček), použití tohoto vzoru je umožněno tím, že program nemůže mít za běhu více sad argumentů. Tento návrhový vzor následně umožňuje získat informace o argumentech programu z jakékoliv části kódu prostým zavoláním statické metody `Arguments::getInstance()`, která nám vždy vrátí stejnou instanci objektu.

Zpracování argumentů má na starost statická třída `ArgumentsParser`, která má jedinou veřejnou metodu `ArgumentsParser::parseArguments()`. Úkolem této metody je získat všechny vstupní argumenty a jejich hodnoty. Argumenty a jejich hodnoty jsou načteny pomocí funkce `getopt()`. Následně probíhá jejich kontrola a odpovídající nastavení objektu Arguments. Při jakékoliv chybě metoda vyhodí výjimku se zprávou odpovídající chybě a kódem odpovídajícím chybovému kódu programu.

Práce se soubory

Po načtení argumentů dochází ke kontrole souborů určených pro vstup/výstup skriptu.

Pro tuto kontrolu máme tři třídy. Třídu `File` a její potomky `JsonFile` a `XMLFIle`.

Třídá `File`

Konstruktor třídy `File` přijímá dva argumenty: cestu k souboru a způsob otevření. V rámci konstruktoru poté dochází ke kontrole existence souboru a přístupových práv. Soubor je otevřen pomocí `fopen`, následuje kontrola návratové hodnoty a poté dojde k uzavření pomocí `fclose` (není potřeba nechávat soubor otevřený). Třída File také, kromě getterů a setterů, implementuje metodu `getContents()`, která vrací, pokud to povolují práva, textový obsah souboru.

U potomků využíváme nestandardní vlastnost PHP oproti jiným objektově orientovaným jazykům (Java, C#,…) a tou je dědění konstruktorů. Není tím pádem potřeba konstruktor třídy `File` přetěžovat a volat rodičovský.

Třída `JsonFile`

`JsonFile` je pouze rozšířena o metodu `getJsonContent()`, která pouze nad obsahem souboru zavolá funkci `json_decode` a její výsledek vrátí. Pokud by došlo k chybě je vyhozena výjimka.

Třída `XMLFile`

`XMLFile` je rozšířena o metodu `getXMLWriter()`, která pouze vrací objekt `XMLWriter`, který pochází z rozšíření jazyka PHP a zaobaluje knihovnu `libxml`. Tomuto objektu je nastavena URI na základě zadaného výstupního souboru.

Převod JSON formátu do XML

Samotný převod má na starosti třída `JsonToXMLParser`. Dále je tu ještě pomocná třída `NameValidator`, která pomocí regulárních výrazů provádí kontrolu správnosti (a případná nahrazení vadných znaků) v rámci jmen XML elementů.

V konstruktoru tato třída přijíma objekt typu `XMLFile`, ze kterého si vezme, za pomocí metody `getXMLWriter()`, objekt `XMLWriter`. Tento objekt je potom používaný pro veškerý zápis

Dále obsahuje metodu `parse()`, která jako parametr přijíma JSON objekt (předem získaný z objetku `JsonFile` pomocí metody `getJsonContent()`). Tato metoda započne zápis do XML, přidává hlavičku (pokud není

v `Arguments` nastaveno jinak), kořenový prvek (opět `Arguments`) a následně zavolá metodu `parseValue()` nad přijatým JSON objektem.

`parseValue()`

Metoda `parseValue()` pomocí systémových funkcí (`is_object`, `is_array`, `is_int`,…) určí o jaký datový typ se jedná a volá odpovídající metody, které provedou zápis elementu (`parseObject`, `parseArray`, `parseNumber`, `parseString`, `parseLiteral`).

`parseObject()`

V `parseObject()` využíváme možnosti přetypování objektu v PHP na asociativní pole, kde klíč je jméno atributu a hodnota je jeho hodnotou. Objekt převedený na asociativní pole celý projdeme a pro každý atributu vytvoříme XML element a zapíšeme jeho hodnotu pomocí `parseValue()`. Jméno elementu je ještě kontrolováno na nepovolené znaky, které jsou nahrazeny pomocí metody `NameValidator::invalidateAndReplace()`.

`parseArray()`

V `parseArray()` přidáváme element zaobalující pole (definovaný v `Arguments`) a poté projdeme jednotlivé prvky pole, přidáme jim zaobalující elementy (definované v `Arguments`) a zapíšeme pomocí `parseValue()`.

`parseNumber()`

V `parseNumber()` dochází k zaokrouhlení čísel dolů a jejich následnému zapsání. Zapsání probíhá buď jako atribut value (`XMLWriter::write_attribute`) anebo jako text (`XMLWriter::text`) v závislosti na nastavení v `Arguments`. Předtím je ještě doplněn do zaobalujícího elementu typ proměné (pokud nastaveno v `Arguments`).

`parseString()`

V `parseString()` dochází k zápisu řetězců. Zápis se provádí buď jako atribut value nebo jako text (stejně jako v `parseNumber()`). Při zápisu jako text se buď použije metoda `xmlWriter::write_raw` anebo `xmlWriter::text` (v závislosti na tom jestli je v `Arguments` nastaven překlad znaků větších než 128). Předtím je ještě doplněn do zaobalujícího elementu typ proměné (pokud nastaveno v Arguments).

`parseLiteral()`

V `parseLiteral()` dochází k zápisu literálů (hodnoty `true`, `false`, `null`). Zápis se provádí opět jako atribut value nebo jako text. Předtím je ještě doplněn do zaobalujícího elementu typ proměné (pokud nastaveno v `Arguments`).