Dokumentace úlohy JSN: JSN2XML v PHP do IPP 2016/2017
Jméno a příjmení: Adrián Tomašov
Login: xtomas32

# 1 Introduction

The task was to create a script which converts XML format to JSON according to RFC and with possible adjustments of the final result. It was necessary to keep in mind that these languages have different standards and do not accept the same characters. Therefore it was necessary to replace malicious characters from data structure, otherwise it would break the consistence of XML.

# 2 Implementation

Because of my little knowledge of PHP language I used numerous minor functions to make it easier to implement the solution of this problem without using system libraries to encode XML files.

## 2.1 Argument Parser

I created my own class Parser which helped me to receive and verify arguments, because standard function `getopt()` did not meet the requirements for this project. In the main function I created an instance of a class Parser and filled it with known arguments according to required specifics. Then I verified and loaded line arguments and at the end I had to check all dependencies.

## 2.2 JSON decode

With using `json_decode()` I decoded input json file string read from input file. The result from this step was a tree structure consisted of following: associative arrays, object(stdClass) and encoded data. After this the program was ready to encode data to XML format.

## 2.3 Conversion

Especially in this part I used many minor functions to help me manage challenges connected with converting to XML format with adjustable output format. I recursively went through the data tree structure read from `json_decode()` and with using minor functions I checked and encoded the data.

The name of the recursive function that I wrote was `encode_xml()`. Main function arguments were (data structure, parser data and few other subsidiary arguments). I iterated over data structure which was in `key` $\Rightarrow$ `value` format. If value was array or object(stdClass) I called this function on current value.

If value was plain data I called the function `ncode_data()` with key and values as parameters which returned the string with encoded data according to input arguments.

## 2.4 Malicious characters

There are several characters that cause inconsistency of XML formatted file. Those characters are: $<, >, \&, ", '$

I needed to create a function that substituted these malicious characters with string that was set with argument `-h` or substituted with an alternative form that would work for XML. However, this did not automatically mean that the word was correct, therefore I also had to create another function for verifying key and value to see if these were acceptable for XML standard.

# 3 Extensions

## 3.1 JPD

This extension was solved with using function `str_pad()`, which filled `size="001"` attribute required size. Number of elements in array together with `--start` switch made maximum possible number in attribute `size`. Then I used function `strlen()` to obtain size of each value.

## 4 Conclusion

As required in the beginning I was supposed to create a program to convert JSON file format to XML file format with possible adjustments of the final results according to user needs. When creating this program I had to write several functions to remove the inconsistency between both languages (malicious characters in XML). Program also contains several minor functions that help user to adjust final XML format.