Task documentation XQR: XML Query in PHP 5 for IPP 2015/2016
Name and surname: Maroš Kopec
Login: xkopec44

Check arguments

The first thing script does, is arguments validation check. For this I use function `getopt` which allows me to set required value for some arguments. It also recognizes long arguments, such as `--input` and short, for example `-n`.

I stumbled on problem with short arguments where I was not able to successfully check validity these short arguments. The fact that I have only one short argument helped me because I did not check for short arguments in `getopt`. I used regular expression `/^-\w*n\w*/` in function `preg_match` in `foreach` loop. If search is successful, it means there is at least one short argument, for example `-xny`. Then if there is match also with regular expression `/-n$/` it means that only one and correct short argument was passed.

In this section I do not check if given files exists or if they are readable.

Get input

If all arguments are valid the script continues with getting input either from file or from standard input. To get content I use `file_get_contents` which returns file as one string. But before that I check for file existence and its readability.

XML input

While I have input in string I can use `SimpleXML` extension for representation XML structure. The problem arises when input XML file is not valid XML. That means it do not have correct syntax or semantic, for example it is missing root element. In this case the SimpleXML `_construct` throws exception which I try to catch.

Query input

Query input is defined by long argument. It is either in query file (`--qf=<file>`) or directly as part of arguments `--query='<query>'`. When used second option I already have query stored in array by `getopt` function. In first case I use the above-mentioned method.

Handling query

After I get query I have to check for syntax and semantic errors. To successfully do that I again use function `preg_match` with regular expressions. It allows me to store query as multidimensional array with query logic representation. One interesting part is this `/(?<=FROM\s)\s*\K\w*[.]?\w+/` regular expression. It matches element or element with attribute or attribute only. Then the matched substring is split by `preg_split` function.

However according to the specification it is allowed to have empty terminal after clause FROM. For this specific case I simply assign value `false` to representative key in query array when above-mentioned regular expression could not have found any match. Later at executing query script returned empty string.

Executing query

With checked syntax and semantics in the query script executes this query over input XML. To iterate over each element in XML, which is now represented as `SimpleXMLElement`, and find desired output I used function `xpath` wich is part of the `SimpleXML` extension.

To successfully find needed element by `xpath` one must learn syntax which is used with this function. With help from colleagues and from manual pages I was able to extract specific elements. I used this tool to create multiple filters on different logic levels.

- First I picked out all elements specified by FROM clause which represents abstract root element in which specific elements or attributes will be searched.

- After that I applied filter SELECT which chooses only wanted elements.
- Then most difficult filter WHERE is on turn. Here I specific conditions under which elements, attributes or values are selected. Here I encountered first problem with `xpath` as a solution for this assignment. The script should evaluate input as not valid and return error code 4 if there are another children, means elements, in element specified by clause WHERE. It is because `xpath` do not consider this obstacle. Surprisingly I solved this problem using only `xpath`. By adding additional search for children in element specified by WHERE. For me it was not easy because I struggled at syntax of this tool.
- Lastly I filtered how much results are needed which is specified by LIMIT clause in query. For this I used simple `array_slice` function. I was able to do that because after filter WHERE was applied the elements were added as array parts.

Producing output

After all `xpath` filters the final XML is represented in array containing `SimpleXMLElement` objects. To be able to use function `file_put_contents` for writing to file I had to convert this array to string. For that I used `foreach` loop and method `asXML`.