Task documentation XQR: XML Query in PHP 5 for IPP 2015/2016
Name and surname: Andrej Barna
Login: xbarna01

1 Overview

This documentation explains techniques used to create a script, which performs SQL-like query on a XML document. The script itself consists of four parts - argument parsing, query processing, element filtering and selection of output elements.

2 Argument parsing

Arguments parsing is performed manually, without the use of argument parsing functions. This way illegal combinations of arguments are easily managed, but at the price of fairly increased function length.

3 Query processing

Query processing is the most complex part of the script. In the first step, `SELECT`, `LIMIT` and `FROM` clauses are parsed and the rest of the query is sent into a function, which processes `CONDITION` and `ORDER BY` clauses. Then `ORDER BY` clause is separated and parsed.

The remaining clause, `CONDITION`, is then translated into an evaluable PHP expression. First, condition is split into lexemes and `AND`, `OR`, `NOT` and = lexemes are translated into operators `&&`, `||`, `!` and ==, respectively. After that, the array containing lexemes is prepared for translation. During the translation, an array containing simplified syntax of the condition is generated, which is used after the translation to verify the validity of the condition.

Translation itself is performed by a cycle, that processes every lexeme of the condition. Lexemes for condition combining and condition negating, namely `(`, `)`, `&&`, `||` and `!`, are pushed into syntax validation array and then the cycle continues. If something other than any of these symbols is detected, then a basic condition in format `<ELEMENT-OR-ATTRIBUTE> <RELATION-OPERATOR> <LITERAL>` is expected and the script checks whether the checked lexeme is a valid `<ELEMENT-OR-ATTRIBUTE>` name. Then, depending on the relation operator, this basic condition is translated into a PHP expression. If the relation operator is `CONTAINS`, then the `strpos()` function is used. Otherwise, if the relation operator was either <, > or =, then the conditon is almost unchanged. In both variants, the value of the element is retrieved by the function `getElemOrAttrVal()`. If this condition is valid, then it is stored in the currently processed item of the condition array and next two items in the condition array – `<RELATION-OPERATOR>` and `<LITERAL>` are replaced by empty strings, since they were already processed and `c` is put into the condition syntax array.

After the translation is finished, the `checkConditionSyntax()` function is called, which checks using the syntax array, whether the condition construction is valid. The check itself is performed in three steps: in the first step, the `!` operators are removed from syntax array, while checking if their placement is valid. Second step is the removal of parentheses, where not only their placement is checked, but also their parity and order. After these two steps, only `c`, `&&` and `||` can be found in the syntax array. Third step checks, whether `c`, `&&` and `||` elements are correctly placed. Since neither two operators nor two conditions adjacent to each other are valid constructions, basic conditions and operators have to alternate, beginning and ending with conditions, because `AND` and `OR` operators are both binary.

4 Element filtering

The filtering of elements is performed in two phases, both using the `xpath()` method of the `SimpleXMLElement` class. In the first phase, the desired pool of elements from `FROM` clause is selected, where only the first element is chosen as pool. Then, in the second phase, are from the pool selected only those elements, whose names match the `SELECT` clause. The array of filtered elements is then passed for condition checking.

5 Selection of output elements

The last part of the script, which consists of three phases. First phase is condition checking, where are output elements selected from the array gained after element filtering. Elements are selected for output when they fulfill the condition given in the query. Condition is evaluated for each element using the `eval()` function, where the translated condition from query parsing is used. When the selection is done, the second phase starts. The array of output elements can be ordered by an element or an attribute, depending on the query. Then, in the third phase, the output elements are written into the output file. The number of elements, that are written into the output, can be limited by the `LIMIT` clause. After the outputting is done, the output file is closed and the script ends.

6 Extensions

Both possible extensions are implemented in the script – ordering of output elements and joining of conditions using logical operators. The implementation of the latter was straightforward, since PHP already has these logical operators built-in and both their priority and precedence are matching the ones that were supposed to be implemented by that extension. The former was implemented using user-defined sorting function `usort()`, for which a simple comparison function was made, which retrieves values of elements using already implemented `getElemOrAttrVal()` function and then compares them.