Dokumentace úlohy XQR: XML Query v PHP 5 do IPP 2014/2015
Jméno a příjmení: Jakub Vitásek
Login: xvitas02

1 Assignment

The assignment was to write a script in PHP 5, which interprets a query similar to the SELECT SQL command on input data formatted as XML. The resulting XML output contains elements meeting the constraints of the input query. The program utilizes a number of subprograms, e.g. the built-in function `simplexml_load_file()` to read the input XML and `SimpleXMLElement::xpath` method to search within the XML.

2 Implementation

The program as a whole can be divided into a number of smaller subroutines, which are described below.

2.1 Getting the user arguments

Since PHP 5 is capable of parsing arguments via the built-in function `getopt()`, there is no need to use regular expressions to get the program options. The `get_options()` function parses the user arguments and assigns them to global variables to be used further in the program. The function also handles a number of error states.

2.2 Parsing the query

The parsing of the query is carried out in the function `parse()`, which uses the grammar defined in the assignment and regular expressions (*regex*) to get each part of the query into appropriate variables. First, I check the general format of the query using a *regex* and then go through the whole query taking out each element of each keyword. This is how the query is arranged in the `$parsed_query` array:

```
[SELECT] => element
[LIMIT] => number
[FROM] => Array
        (
                [0] => element
                [1] => .attribute
        )
[WHERE] => Array
        (
                [0] => element
                [1] => .attribute
                [2] => <RELATIONAL–OPERATOR>
                [3] => <LITERAL>
        )
[ORDER BY] => Array
        (
                [0] => element
                [1] => .attribute
                [2] => <ORDERING>
        )
```

2.3 Composing XPath

The most crucial part of the program lies within the function `searchXML()`. Based on the contents of the query array, the string that is finally inserted into xpath() is composed of three parts – `$select_xpath`, `$from_xpath` and `$where_xpath` – containing the respective XPath route. If the WHERE keyword is not present in the query, the `$where_xpath` variable is empty.

An example XPath route can be the one below, which gets composed upon calling the query `SELECT book FROM library`:

```
xpath(" //library[1]//book ");
```

Another example is an xpath with an attribute, which gets composed upon calling the query `SELECT book FROM ROOT WHERE .id CONTAINS "bk101"`:

```
xpath(" //book[contains(@id, 'bk101' ");
```

To further explain the special characters in the xpath route,

1. The `//` selects nodes from the current node that match the selection specified afterwards, no matter their position in the document.
2. The `[ ]` are a conditional operating on single node individually. It matches the node that matches the conditions inside the square brackets.
3. `contains` is a function operating on a string. If there is a node set passed as an argument, it is converted into a string by returning the string-value of the node in the set that is first in order.
4. The `@` selects attributes.

2.3 Formatting the results

The function `searchXML()` described above returns a `simpleXML` object, which has to be converted into an XML format before printing out. Using a `foreach` loop, which takes the object as an array, the respective parts of the object get concatenated to the result string $final_xml, which is returned and then printed to standard output or to an output file, depending on the user arguments. Also, during the `foreach` loop, there is a helper counter `$i` which increments each time passing an element. On each pass, the `$limit` variable (if set) is checked for equality with the `$i` variable. If the equality check returns true, the loop is broken and the function returns the resulting XML string. Furthermore, based on the user arguments, the XML header and the root element is concatenated to the resulting string by basic loop and helper counter methods.

3 Extensions

The only extension I implemented is the support of the "ORDER BY" clause (labeled ORD in the assignment). The description is in the section below.

3.1 ORD

All the ordering is carried out in the function `orderXML()`, which takes the initial `simpleXML` object, the resulting `simpleXML` object and the user-defined query. I first generate the xpath for the element/attribute I want to sort by, to later pass it into the built-in function `array_multisort()` with either SORT_ASC or SORT_DESC parameters, based on the user-defined query. I then return the return value of the sorting function.