Task documentation XTD: XML2DDL in PHP 5 for IPP 2012/2013
Name and surname: Miroslav Cibulka
Login: xcibul10

PROBLEM ANALYSIS

Main objective in this project is to create translator from eXtensible Markup Language (XML) to Data Definition Language (DDL). There are several problems to be solved: Argument parser, Lexical analyzer, XML class, DDL class, error reporting.

STRUCTURE

There are several modules in this project that I have made. Every class has it's own module, because of 'magic' method *__autoload* that *include*s it, when it is needed.

Module *interfaces.php* contains interfaces of each class used in this project. It is used to prototype classes and public functions in them.

SOLUTION & IMPLEMENTATION

ERROR REPORTING

I used Exception system to report error or warning for this project because Exceptions are easily catchable. This is useful mostly for testing. Every exception message is printed on standard error output.

ARGUMENT PARSER

The main purpose of this module is to read arguments of program and return associative array of options and values. It is achieved via regular expression that analyzes every parameter individually.

If argument is matched, it will be added to output array. If argument is wrongly formulated then regular expression doesn't match it and error will be thrown.

LEXICAL ANALYZER

It is implemented as class *LEXIC*, contains functions and variables needed to read whole file and fill new *XML* object created from class *XML*.

First of all, input file is read to variable and then analyzed by regular expression which returns two-dimensional array of matched strings. This array is used as content of *XML* class. But it has to be transposed first.

Regular expression I used should match whole file if file is valid. It ignores white spaces in text value of node but not in attributes. So if there are white spaces in attribute value, type of attribute will be *NVARCHAR* not *BIT*.

There is also simple validity check but it is not part of assignment.

XML CLASS

Represents XML document divided to binary tree (represented as array[1]) which can be converted to DDL object. Class contains variables like *$nodes*, which is array of XML nodes.

Node (represented by *node* class) has pointer at parent and children. If there is no parent (node is root), value of pointer is NULL. One node has precisely one parent. If it is leaf then array is empty. *value* is variable that contains type of value (*Integer*, *Boolean*, *String*, *Float*, etc.) and value itself.

Parameters are represented by array of instance of *class param*. Every parameter has it's value and type. Type is found out with function *typeOf($str)* via regular expression. If there are no parameters array is empty.

Conversion to DDL class is achieved with pre-order traversal through XML nodes and creating new DDL nodes with following algorithm:

1. Create new DDL object with node named tables
2. For each node in XML nodes, except root node, create new DDL node representing table
3. For each node children (if has any) create new DDL node, which represents reference to table

DDL CLASS

Is derived from XML class because structure is the same but there are added/override some functions.

There is 'magic' method *__toString()* that returns string representation of DDL object in format, which is chosen by parameters. Commonly it returns data in DDL, but if '-g' parameter is used, method returns XML representation of relations in DDL object.

Relations between tables are provided by function *getRelations()*, which multiple times iterates through array of nodes and sets all relations (even using relation transitivity) among them.

array[1]: To represent binary tree I have chosen simple array because it is faster then recursive (binary tree) traversals and linear (array) traversal goes through the same way as pre-order.