# 1 Introduction

LuaXML is pure lua library for reading and serializing of the XML files. Current release is aimed mainly as support for the odsfile package.

# 2 Overview:

This module provides a non-validating XML stream parser in Lua.

# 3 Features:

- Tokenises well-formed XML (relatively robustly)
- Flexible handler based event api (see below)
- Parses all XML Infoset elements - ie.
  - Tags
  - Text
  - Comments
  - CDATA
  - XML Decl
  - Processing Instructions
  - DOCTYPE declarations
- Provides limited well-formedness checking (checks for basic syntax & balanced tags only)
- Flexible whitespace handling (selectable)
- Entity Handling (selectable)

# 4 Limitations:

- Non-validating
- No charset handling
- No namespace support
- Shallow well-formedness checking only (fails to detect most semantic errors)

# 5 API:

The parser provides a partially object-oriented API with functionality split into tokeniser and hanlder components.

The handler instance is passed to the tokeniser and receives callbacks for each XML element processed (if a suitable handler function is defined). The API is conceptually similar to the SAX API but implemented differently.

The following events are generated by the tokeniser

```
handler:start       - Start Tag
handler:end         - End Tag
handler:text        - Text
handler:decl        - XML Declaration
handler:pi          - Processing Instruction
handler:comment     - Comment
handler:dtd         - DOCTYPE definition
handler:cdata       - CDATA
```

The function prototype for all the callback functions is

```
callback(val,attrs,start,end)
```

where attrs is a table and val/attrs are overloaded for specific callbacks - ie.

| Callback | val | attrs (table) |
|---|---|---|
| start | name | `{ attributes (name=val).. }` |
| end | name | nil |
| text | <text> | nil |
| cdata | <text> | nil |
| decl | ”xml” | `{ attributes (name=val).. }` |
| pi | pi name | `{ attributes (if present).. _text = <PI Text> }` |
| comment | <text> | nil |
| dtd | root element | `{ _root = <Root Element>, _type = SYSTEM\|PUBLIC, _name = <name>, _uri = <uri>, _internal = <internal dtd> }` |

(start & end provide the character positions of the start/end of the element)

XML data is passed to the parser instance through the ‘parse’ method (Nore: must be passed a single string currently)

# 6 Options

Parser options are controlled through the 'self.options' table. Available options are -

- stripWS

  Strip non-significant whitespace (leading/trailing) and do not generate events for empty text elements

- expandEntities

  Expand entities (standard entities + single char numeric entities only currently - could be extended at runtime if suitable DTD parser added elements to table (see obj._ENTITIES). May also be possible to expand multibyre entities for UTF–8 only

- errorHandler

  Custom error handler function

NOTE: Boolean options must be set to 'nil' not '0'

# 7 Usage

Create a handler instance -

```
h = { start = function(t,a,s,e) .... end,
      end = function(t,a,s,e) .... end,
      text = function(t,a,s,e) .... end,
      cdata = text }
```

(or use predefined handler - see handler.lua)
Create parser instance -

```
p = xmlParser(h)
```

Set options -

```
p.options.xxxx = nil
```

Parse XML data -

```
xmlParser:parse("<?xml... ")
```

# 8 Handlers

# 9 Overview:

Standard XML event handler(s) for XML parser module (xml.lua)

# 10 Features:

```
printHandler        - Generate XML event trace
domHandler          - Generate DOM-like node tree
simpleTreeHandler   - Generate 'simple' node tree
simpleTeXhandler    - SAX like handler with support for CSS selectros
```

# 11 API:

Must be called as handler function from xmlParser and implement XML event callbacks (see xmlParser.lua for callback API definition)

## 11.1 printHandler:

printHandler prints event trace for debugging

## 11.2 domHandler:

```
domHandler generates a DOM-like node tree  structure with
a single ROOT node parent - each node is a table comprising
fields below.

node = { _name = <Element Name>,
         _type = ROOT|ELEMENT|TEXT|COMMENT|PI|DECL|DTD,
         _attr = { Node attributes - see callback API },
         _parent = <Parent Node>
         _children = { List of child nodes - ROOT/NODE only }
       }

The dom structure is capable of representing any valid XML document
```

## 11.3 simpleTreeHandler

simpleTreeHandler is a simplified handler which attempts to generate a more 'natural' table based structure which supports many common XML formats.

The XML tree structure is mapped directly into a recursive table structure with node names as keys and child elements as either a table of values or directly as a string value for text. Where there is only a single child element this is inserted as a named key - if there are multiple elements these are inserted as a vector (in some cases it may be preferable to always insert elements as a vector which can be specified on a per element basis in the options). Attributes are inserted as a child element with a key of '_attr'.

Only Tag/Text & CDATA elements are processed - all others are ignored.

This format has some limitations - primarily

- Mixed-Content behaves unpredictably - the relationship between text elements and embedded tags is lost and multiple levels of mixed content does not work
- If a leaf element has both a text element and attributes then the text must be accessed through a vector (to provide a container for the attribute)

In general however this format is relatively useful.

It is much easier to understand by running some test data through 'textxml.lua -simpletree' than to read this)

# 12 Options

```
simpleTreeHandler.options.noReduce = { <tag> = bool,.. }

    - Nodes not to reduce children vector even if only
      one child

domHandler.options.(comment|pi|dtd|decl)Node = bool

    - Include/exclude given node types
```

# 13 Usage

Pased as delegate in xmlParser constructor and called as callback by xmlParser:parse(xml) method.

# 14 History

This library is fork of LuaXML library originaly created by Paul Chakravarti. Its original version can be found at `http://manoelcampos.com/files/LuaXML--0.0.0-lua5.1.tgz`. Some files not needed for use with luatex were droped from the distribution. Documentation was converted from original comments in the source code.

# 15 License:

This code is freely distributable under the terms of the Lua license (`http://www.lua.org/copyright.html`)