



# XML damage control

Silvan Jegen

me@sillymon.ch

24. September 2016





- 1 XML
  - XML in theory
  - XML in practice
- 2 Dealing with XML
  - Programming interfaces
  - Benchmark
- 3 Conclusion





## Extensible Markup Language (XML)





## Extensible Markup Language (XML)

### XML aspects

- Well-formedness
- Validation
- Namespaces
- Entities





## Related specifications

- XSLT
- XPath
- XQuery
- XML Encryption
- ...





## Variants

- RDF XML
- XMPP
- EPUB
- XHTML
- ...
- 200+ more





# XML in practice





## Enterprise usage

- SOAP
- Configuration
- Data storage/exchange
- Java ecosystem...







Annotate parts of text with additional information





Annotate parts of text with additional information

## Text Markup

```
<document>Some text <tag>some other text that  
should be tagged</tag> even <tag2>more</tag2>  
text...</document>
```





## XML

```
<document>Some text <tag>some other text that  
should be tagged</tag> even <tag2>more</tag2>  
text...</document>
```



# XML vs. JSON



## XML

```
<document>Some text <tag>some other text that  
should be tagged</tag> even <tag2>more</tag2>  
text...</document>
```

## JSON

```
["Some text ", {"t": "tag", "s": "some other text  
that should be tagged"}, "even", {"t": "tag2",  
"s": "more"}, "text..."]
```





# Dealing with XML





- Stream-oriented (SAX, Stax)
- Tree traversal (DOM)
- XML Data binding
- Transformation languages (XSLT, XQuery)





- Stream-oriented (SAX, Stax)
- Tree traversal (DOM)
- XML Data binding
- Transformation languages (XSLT, XQuery)
- Other?



# Benchmark



- ezxml
- Golang encoding/xml
- mxml ('Mini-XML', not 'Minimal XML')
- Python 2 ElementTree
- sxmlc
- yxml

All code available at:

[git://git.sillymon.ch/slcon3.git](https://git.sillymon.ch/slcon3.git)





# Benchmark setup



Linux machine, i7 CPU, 8GB RAM

- 627MB of XML in 10'000 files from PubMed Central
- Printing the article titles
- 20 runs (after cache warming)
- Single-threaded (except for Go)





## ezxml

- Program: 21 lines
- Library: 623 lines
- URL: <http://ezxml.sourceforge.net/>
- Type: DOM (Level 3; XPath)

```
ezxml_t title = ezxml_get(ezdoc, "front", 0,\  
    "article-meta", 0, "title-group", 0,\  
    "article-title", -1);
```





## Go encoding/xml

- Program: 30 lines
- Library: 6235 lines (stdlib)
- URL: <https://golang.org/pkg/encoding/xml/>
- Type: XML Data binding

```
type article struct {  
    Title string `xml:"front>article-meta>title-group\  
        article-title"`  
}
```





## mxml

- Program: 38 lines
- Library: 9633 lines
- URL: <http://www.minixml.org/>
- Type: DOM (Level 3; Xpath)?

```
node = mxmlFindElement(root, root, "title-group",  
    NULL, NULL, MXML_DESCEND);
```





## Python 2 ElementTree

- Program: 12 lines
- Library: 1107 lines (stdlib)
- URL:  
<https://docs.python.org/2/library/xml.etree.element-tree.html>
- Type: DOM (Level 3; Xpath)?

```
tg = r.findall("./front/article-meta/title-group")  
at = tg[0].find("article-title")
```





## sxmlc

- Program: 59 lines
- Library: 2690 lines
- URL: <http://sxmlc.sourceforge.net/>
- Type: DOM

```
const char *path[] = {"front", "article-meta", \
                    "title-group", "article-title", NULL};
for (int i = 0; path[i]; i++) {
    next = find_child_node(next, path[i]);
    if (!next) {
        fprintf(stderr, "Could not find '%s'
                    tag.\n", path[i]);
        return;
    }
}
```





## yxml

- Program: 103 lines
- Library: 1039 lines
- URL: <https://dev.yorhel.nl/yxml>
- Type: Stream-oriented

```
case YXML_ELEMSTART:
  if (!strcmp(state->elem, "title-group")) {
    intitlegroup = 1;
  } else if (!strcmp(state->elem, "article-title")
    && intitlegroup) {
    printf("%s: ", state->elem);
    inarticletitle = 1;
  }
  break;
```





	mean	$\sigma$	min	max	size
Python ElementTree	155.5	8.869	145.7	172.3	N/A
Go encoding/xml	48.34	4.982	35.33	52.92	2M
mxml	23.96	1.841	22.32	27.64	N/A
sxmlc	15.51	0.259	15.12	16.01	41K
ezxml	6.460	0.058	6.366	6.592	31K
yxml	4.123	0.220	3.885	4.520	18K





# Conclusion



- Complex specifications, (comparatively) hard to parse and verbose
- Use the ezxml or yxml libraries





- Complex specifications, (comparatively) hard to parse and verbose
- Use the ezxml or yxml libraries
- Ok:

```
<document>Some text <tag>some other text that  
should be tagged</tag> even <tag2>more</tag2>  
text...</document>
```





- Complex specifications, (comparatively) hard to parse and verbose
- Use the ezxml or yxml libraries

- Ok:

```
<document>Some text <tag>some other text that  
should be tagged</tag> even <tag2>more</tag2>  
text...</document>
```

- No:

```
<thing><key>Key</key><value>Val</value></thing>
```



Thanks for your attention



Questions?

