

PA193 Secure coding principles and practices

XML Parser

Matúš Nemeč
Pankaj Agarwal
Anupam Gupta

Scope

- Rules for the parser are chosen as close as possible to the XML specification
 - some exception had to be made
 - (you have to stop somewhere)
- Full scope specification provided with project
- Accepts and prints a document which satisfies given rules
- Reject any document that does not match our specification, stop at first error
 - provide an informative error code

Scope

- Some structures are left out from the project:
 - Processing Instructions, CDATA Sections, Document Type Declaration, Attribute lists and many others will cause the parser to throw an error
- Supporting the necessities
 - attributes in tags
 - content between tags
 - comments `<!-- ... -->`
- Partial support in some cases
 - document declaration `<?xml ... ?>`

Document declaration

- If present, only allowed at beginning of the file
- `<?xml version="1.1" standalone="yes" ?>`
- In XML specification names and values are precisely defined
- For simplicity only parse the values and check if at least one (any) is present (partial support)
 - Values do not play any role for our parser
 - All documents are standalone

Parsing a tag

- Find first < (beginning of this tag)
- Find next < (or EOF) and work with the text inbetween
- Find last > in the text, this should close the tag
- Anything after > and before < is text content
 - In content < and > is prohibited
 - content is parsed by higher level function
- Parse name of the tag, identify the type:
 - <begin> </end> <empty-element/>
- Parse attributes: name = "value" or name = 'value'
 - Inside value only < is prohibited
 - Attributes inside one tag must have unique name

```
<root att1 = "Here's" attr = 'Johnny'> content <something-else/>
```

The diagram shows the tag `<root att1 = "Here's" attr = 'Johnny'> content <something-else/>`. A red arrow spans from the opening `<` to the closing `>`. A green arrow spans from the opening `<` to the end of the tag. A yellow arrow spans from the opening `<` to the end of the tag. A blue double-headed arrow is positioned below the opening `<`.

Parsing a node

- Find an opening tag
- Repeat
 - Parse next tag
 - If there is missing closing tag – error
 - If next tag is <opening> – recursion, add result as child
 - Else if <empty/> – add as child, continue cycle
 - Else if </closing>
 - name must match and closing has no attributes – return
 - else error
 - Add text content (if any) to current node

Comments

- Allowed anywhere except inside other markup
 - `<root <!-- here you must not place a comment --> >`
- Treat as tag, opened by `<!--`
- closed by closest matching `-->`
- Inside a comment only a sequence of 2 "-" is prohibited (also ending with `--->` is error)
- Comments are deleted from buffer
 - content of a comment is ignored
- Comment may be before and after root element

Detected errors

- Mismatched tags:
 - `</closing-without-opening><a>`
 - `<a>`
 - `<a>`
 - `</closing-and-empty/>`
- Illegal characters in names
 - `!#$%&'()*+,-/;<=>?@[]^`{|}~\`"`
 - `0123456789.-` (at the beginning of a tag name)
- Duplicate names of attributes inside one tag

Detected errors

- "<" must not appear anywhere except when performing its markup-delineation roles
- Bad comments
- End of input
- Unexpected characters (context specific)
- Bad usage of quotes
- Behaviour for anything outside of our scope is not guaranteed
- For full list refer to specification

Thank you for your attention