Big Data – Exercises & Solutions

Spring 2019 – Week 4 – ETH Zurich

Introduction

This exercise will cover XML and JSON well-formedness.

We recommend you use an online editor to validate and check your solutions for well-formedness, such as this (https://www.xmlvalidation.com/) for XML and this (https://jsonlint.com/) for JSON. To edit XML files, something like this (https://www.webtoolkitonline.com/xml-formatter.html) might work better, which offers syntax highlighting and formatting.
If you prefer, you can instead install oXygen, an XML/JSON development IDE. Download links can be found on the course website (https://www.systems.ethz.ch/courses/spring2020/bigdataforeng/material), and the license key is available on the course Moodle.

1 XML

1.1 Well-formedness

For each of the following XML documents, say if it is well-formed or not; if not, then also correct the errors. An online editor may help with these tasks, but first try to solve them without software support.

Document 1

```
<Burger>
    <Bun>
        <Pickles/>
        <Cheese/>
        <Patty/>
    </Bun>
</Burger>
```

Solution

It is well-formed.

Document 2

```
<Burger>
    <Bun>
        <Pickles/>
        <Cheese/>
        <Patty/>
    </Bun>
</Burger>
<Cola>
    <Sugar/>
    <Water/>
</Cola>
<Fries kind="French"/>
```

Solution

This document is not well-formed because it contains more than one root element. We can fix this by wrapping all the elements in a single root element like so:

```
<Combo>
    <Burger>
        <Bun>
            <Pickles/>
            <Cheese/>
            <Patty/>
        </Bun>
    </Burger>
    <Cola>
        <Sugar/>
        <Water/>
    </Cola>
    <Fries kind="French"/>
</Combo>
```

Document 3

```
<Email>
    Dear Mr. John Doe,

    I hereby kindly request you to revise the attached draft
    of the merger contract and provide me with any feedback you may have.

    Best regards,
    Jane Doe
    Legal Department.
</Email>
P.S. Also, please stop sending me funny cat pictures; I laugh all day and cannot work.
```

Solution

This document is not well-formed because text is not allowed to follow the root element. One way of fixing it is to move the trailing text inside the root element:

```
<Email>
    Dear Mr. John Doe,

    I hereby kindly request you to revise the attached draft
    of the merger contract and provide me with any feedback you may have.

    Best regards,
    Jane Doe
    Legal Department.

    P.S. Also, please stop sending me funny cat pictures; I laugh all day and cannot work.
</Email>
```

Document 4

```
<Band name="Metallica"
      member="James Hetfield"
      member="Lars Ulrich"
      member="Kirk Hammett"
      member="Robert Trujillo"/>
```

Solution

This document is not well-formed because it contains an element with duplicate attribute names. We can fix it by making the attributes into child elements:

```
<Band name="Metallica">
    <member>James Hetfield</member>
    <member>Lars Ulrich</member>
    <member>Kirk Hammett</member>
    <member>Robert Trujillo</member>
</Band>
```

1.2 Create your own XML

Copy the text of the introduction above (including the title until the text 'by email') and paste it into an editor as plain text. Create a possible XML document keeping the same context and including formatting (title, sections, style, links, etc.).

Questions

1. Is your XML well-formed? If not, correct any mistakes. You can use an editor to help you with this task.
2. Is this data structured, unstructured, or semi-structured?

Solution

There is no unique way to create an XML encoding the text above and its structure. This is one possible solution:

```
<?xml version="1.0" encoding="UTF-8"?>
<document lang="en">
    <title font-size='15px' font-style='bold'>Introduction</title>
    <paragraph font-size='12px'>This exercise will cover XML and JSON well-formedness.</paragraph>
    <paragraph font-size='12px'>
        We recommend you use an online editor to check your solutions, such as <link url="https://www.webt
oolkitonline.com/xml-formatter.html">this</link> for XML and <link url="https://jsonlint.com/">this</link>
for JSON.
        If you prefer, you can instead install <link url="https://www.oxygenxml.com/xml_editor/software_ar
chive_editor.html">oXygen</link>, an XML/JSON development IDE. You should have received a license key for
it by email.
    </paragraph>
</document>
```

1. If all the syntax rules have been respected, the document will be well-formed.
2. This data is semi-structured: there is some structure, but not all of the content is structured as in a flat database.

1.3 XML Names

Which of the following are valid XML Names?

1. `<_bar/>`
2. `<Xmlelement/>`
3. `<Foo/>`
4. `<foo123/>`
5. `<foo_123/>`
6. `<foo-123/>`
7. `<foo#123/>`
8. `<foo.123/>`
9. `<-123/>`
10. `<123foo/>`
11. `<doctype/>`

Solution

1, 3, 4, 5, 6, 8, and 11 are valid names. Remember:

1. Element names must start with a letter or underscore.
2. Element names cannot start with the string `xml` (or XML, or Xml, etc).
3. Element names can contain letters, digits, hyphens, underscores, and periods.
4. Element names cannot contain spaces.
5. Element names are case-sensitive.

1.4 Predefined entities

XML has only 5 predefined entities. Associate each escape code with the corresponding character.

1. `<` `>`
2. `&` `"`
3. `>` `'`
4. `&quot;` `&`
5. `&apos;` `<`

Solution

1. `<` `<`
2. `&` `&`
3. `>` `>`
4. `&quot;` `"`
5. `&apos;` `'`

2. JSON

2.1 JSON Values

JSON documents are composed of name–value pairs. List the 6 possible JSON value types.

Solution

1. Number (integer or decimal)
2. String (in double quotes)
3. Boolean (`true` or `false`)
4. Array (in square brackets)
5. Object (in curly braces)
6. null

2.2 JSON well-formedness

For each of the following JSON documents, state if you see any syntax mistakes. If yes, correct them.

Document 1

```
{
  "burger" : {
      "bun" : ["pickles", "cheese", "patty"],
      "extraIngredients" :
  }
}
```

Solution

The document has a syntax error—every key has to have a value, and `extraIngredients` lacks one. If we want to signal an absence of a value, we can use `null`:

```
{
  "burger" : {
      "bun" : ["pickles", "cheese", "patty"],
      "extraIngredients" : null
  }
}
```

Document 2

```
{
  "pizza" : {
      "topping" : "salami",
      "topping" : "cheese",
      "topping" : "oregano"
  }
}
```

Solution

This document has a syntax error—duplicate keys are not allowed. We can put the repeating name–value pairs into an array:

```
{
  "pizza" : {
      "topping" : ["salami", "cheese", "oregano"]
  }
}
```

Document 3

```
{
    name : "John Doe",
    age : 42,
    occupation : "Penguin Turner",
    motto : "Up is life!"
}
```

Solution

This document also has a syntax error—JSON keys must be double-quoted:

```
{
    "name" : "John Doe",
    "age" : 42,
    "occupation" : "Penguin Turner",
    "motto" : "Up is life!"
}
```

3. Conversions from a relational database

Imagine that we have a relational table which stores chat messages. Translate this table into XML and JSON.

conversation_id	people	sender	content	timestamp	is_read	attachment_id

42	charlie,ari,jesse	charlie	hey, here's the doc ><	1510410193	TRUE	NULL
42	charlie,ari,jesse	charlie	NULL	1510410244	TRUE	doc_6492
42	charlie,ari,jesse	ari	thanks!	1510432987	FALSE	NULL
17	rudy,sage	rudy	look at this cute cat!	1500897189	TRUE	img_91847
17	rudy,sage	NULL	aww	1506610190	TRUE	NULL

Solution

There are, of course, many possible solutions. Here's one of them:

JSON:

```
[
    {
        "conversation_id": 42,
        "people": ["charlie", "ari", "jesse"],
        "messages": [
            {
                "sender": "charlie",
                "content": "hey, here's the doc ><",
                "timestamp": 1510410193,
                "is_read": true
            },
            {
                "sender": "charlie",
                "timestamp": 1510410244,
                "is_read": true,
                "attachment_id": "doc_6492"
            },
            {
                "sender": "ari",
                "content": "thanks!",
                "timestamp": 1510432987,
                "is_read": false
            }
        ]
    },
    {
        "conversation_id": 17,
        "people": ["rudy", "sage"],
        "messages": [
            {
                "sender": "charlie",
                "content": "look at this cute cat!",
                "timestamp": 1500897189,
                "is_read": true,
                "attachment_id": "img_91847"
            },
            {
                "content": "aww",
                "timestamp": 1506610190,
                "is_read": true
            }
        ]
    }
]
```

XML:

```
<?xml version="1.0" encoding="UTF-8" ?>
<conversations>
    <conversation id="42">
        <people>charlie</people>
        <people>ari</people>
        <people>jesse</people>
        <message>
            <sender>charlie</sender>
            <content>hey, here's the doc &gt;&lt;</content>
            <timestamp>1510410193</timestamp>
            <is_read>true</is_read>
        </message>
        <message>
            <sender>charlie</sender>
            <timestamp>1510410244</timestamp>
            <is_read>true</is_read>
            <attachment_id>doc_6492</attachment_id>
        </message>
        <message>
            <sender>ari</sender>
            <content>thanks!</content>
            <timestamp>1510432987</timestamp>
            <is_read>false</is_read>
        </message>
    </conversation>
    <conversation id="17">
        <people>rudy</people>
        <people>sage</people>
        <message>
            <sender>charlie</sender>
            <content>look at this cute cat!</content>
            <timestamp>1500897189</timestamp>
            <is_read>true</is_read>
            <attachment_id>img_91847</attachment_id>
        </message>
        <message>
            <content>aww</content>
            <timestamp>1506610190</timestamp>
            <is_read>true</is_read>
        </message>
    </conversation>
</conversations>
```