RiskSense, Inc.
Python XML JSON Test

RiskSense, Inc.

June 19, 2015

Contents

```
{
  "days" : 365
}
```

(a) JSON data.

```
<object>
    <number name="days">
        365
    </number>
</object>
```

(b) XML data.

Figure 1: Example JSON to XML type and name mapping.

1 Introduction

This assignment entails creating a simple program that will allow users to give arbitrary JSON data and have it converted to XML according to the following standard.

In order to complete this assignment you must perform the following tasks.

- Fully implement the Python class `XMLJSONConverter` according to the given specification.
- Provide a simple command line interface for running your program.

Each of the aforementioned items is elaborated in the following sections.
In general, the more you treat this project as though it were going to be real production code the better.
You program MUST output valid XML for all valid JSON!

2 JSON to XML Specification

2.1 Overview

JSON does not strictly map to XML in a single way. There are many different ways one could create a mapping from a JSON document to a XML document. This section describes the mapping that your code MUST follow in order to be considered valid.

2.2 General Constraints

In JSON objects you are given name value pairs. This can be mapped to XML in several ways. For this program, the XML element name corresponds to the *type* of the element, **not the name of the element**. The name of the JSON element in a JSON object is given as an *attribute* on the XML element. Only JSON array and object values may be at the top level of a file. The specifics of these values are discussed in greater detail in later sections.

2.3 JSON Types

JSON supports the following types, all of which must be mapped into your XML output. **Only JSON objects and arrays are supported as *top level* values.**

```
-5
```

(a) A single number in JSON

```
<number>
    -5
</number>
```

(b) A single number in XML

Figure 2: Conversion of JSON number to XML

```
{
  "cars": 2,
  "trucks": 10
}
```

(a) Nested numbers in JSON

```
<object>
    <number name="cars">
        2
    </number>
    <number name="trucks">
        10
    </number>
</object>
```

(b) Nested numbers in XML

Figure 3: Conversion of JSON numbers to XML

2.3.1 Number

A JSON element with a number as a value should map to an XML element named `<number>`, with the number as the single and only value between the opening and closing tags.[1] An example of this is shown in figure 2 and 3.

2.3.2 String

A JSON element with a string as a value should map to an XML element named `<string>`, with the string as the single and only value between the opening and closing tags. An example of this is shown in figure 4 and 5.

2.3.3 Boolean

A JSON element with a Boolean as a value should map to an XML element named `<boolean>`, with the boolean as the single and only value between the opening and closing tags. The only valid boolean values are `true` and `false`.[2] An example of this is shown in figure 6 and 7.

[1] Whitespace is allowed, but discouraged. Newlines and tabs are allowed.
[2] These *are* case sensitive.

```
"Hello World"
```

(a) Single string in JSON

```
<string>
    Hello World
</string>
```

(b) Single string in XML

Figure 4: Conversion of a JSON String to XML

```
{
  "firstName": "John",
  "lastName": "Smith"
}
```

(a) Nested strings in JSON

```
<object>
    <string name="firstName">
        John
    </string>
    <string name="lastName">
        Smith
    </string>
</object>
```

(b) Nested strings in XML

Figure 5: Conversion of JSON Strings to XML

```
false
```

(a) A single boolean in JSON

```
<boolean>
    false
</boolean>
```

(b) A single boolean in XML

Figure 6: Conversion of a JSON Boolean to XML

```
{
  "isHuman": true,
  "isTall": false
}
```

(a) Nested booleans in JSON

```
<object>
    <boolean name="isHuman">
        true
    </boolean>
    <boolean name="isTall">
        false
    </boolean>
</object>
```

(b) Nested booleans in XML

Figure 7: Conversion of JSON Booleans to XML

```
[1, "test"]
```

(a) A single array in JSON

```
<array>
    <number>
        1
    </number>
    <string>
        test
    </string>
</array>
```

(b) A single array in XML

Figure 8: Conversion of JSON array to XML

2.3.4 Array

A JSON element with an array as a value should map to an XML element named `<array>`. For each element in the JSON array there should be a corresponding XML sub-element denoting the value and type. A JSON array may contain arbitrary sub values. *Elements of the array should not have a name attribute. Remember, JSON arrays are not required to be homogeneous!* An example of this is shown in figure 8 and 9.

2.3.5 Object

A JSON element with an object as a value should map to an XML element named `<object>`. Objects can contain arbitrary sub-values.An example of this is shown in figure 10. Objects are the only JSON types that generate *name* attributes on XML tags.

```
{
  "fibs": [0,1,1,2,3,"fibs", true]
}
```

(a) Nested array in JSON

```
<object>
    <array name="fibs">
        <number>
            0
        </number>
        <number>
            1
        </number>
        <number>
            1
        </number>
        <number>
            2
        </number>
        <number>
            3
        </number>
        <string>
            fibs
        </string>
        <boolean>
            true
        </boolean>
    </array>
</object>
```

(b) Nested array in XML

Figure 9: Conversion of JSON arrays to XML

```
{
  "profile": {
    "firstName": "John",
    "lastName": "Smith",
    "age": 20,
    "friends": ["Joe", {
      "firstName": "Sue",
      "lastName": "Jones"
      }]
  }
}
```

(a) Objects in JSON

```
<object>
    <object name="profile">
        <string name="firstName">
            John
        </string>
        <string name="lastName">
            Smith
        </string>
        <number name="age">
            20
        </number>
        <array name="friends">
            <string>
                Joe
            </string>
            <object>
                <string name="firstName">
                    Sue
                </string>
                <string name="lastName">
                    Jones
                </string>
            </object>
        </array>
    </object>
</object>
```

(b) Objects in XML

Figure 10: Conversion of JSON objects to XML

```
null
```

(a) Null in JSON

```
<null/>
```

(b) Null in XML

Figure 11: Conversion of JSON null to XML

```
{
  "computer_name": null
}
```

(a) Nested null in JSON

```
<object>
    <null name="computer_name"/>
</object>
```

(b) Nested null in XML

Figure 12: Conversion of JSON null to XML

2.3.6 null

A JSON element with an null as a value should map to an XML element named `<null/>`. Unlike other XML representations for JSON values, a null value does not have a closing tag, rather it is self-closing by adding a forward slash before the end of the element tag. An example of this is shown in figure 11 and 12.

2.3.7 Examples

Besides the examples shown here, there examples files in the `examples/` folder in the archive with which you were provided. These were generated with our own reference implementation.

If anything in those files seems to not match up with this document, please contact us and let us know.

3 Python Code

3.1 Overview

There are very few constraints on what you may and may not do in this project. The few that do exist are listed below. Otherwise feel free to do whatever you think is best to accomplish the task.

We will be providing you a few basic Python files from which to build the project. Their use and purpose is described below.

Your program may be made of as many files as you like, but all the source files you write should be in the /src directory. You may place library code anywhere.[3]

3.2 Python Version

You code must be compliant with Python 3.4.2. Code that is valid Python 2 but not Python 3.4.2 will not be considered correct.

[3] See section 3.3 for more details about libraries.

```
# python3 your_program.py json_file xml_file
```

Figure 13: Example CLI invocation

3.3 Open Source Libraries

You may use *any* open source libraries to help you accomplish this project without any restrictions, however *You must either include these libraries in your submitted code, or have them automatically downloaded through your build tool.*[4]

3.4 XMLJSONConverter

The core of the program you will implement revolves around the class `XMLJSONConverter`. This class defines a single method `convertJSONtoXML` which takes two `FilePath String` values, an input JSON and an output XML. Your code must convert the given JSON to XML based on the specification in section 2.

3.5 CLI

You program must have a *very* simple command line interface. The first argument is taken to be the path to the input JSON file and the second argument is taken to be the path to the XML output file. Any other number of arguments should print an usage message. The XML file may or may not exist, but the path must be valid. Example invocation can be seen in Figure 13.

4 Building

4.1 Python Version

You code must run on a Linux system running Python 3.4.2. Although it should not matter, I will be testing your code on a Debian 8 system running the `python3` package.

4.2 README

Make sure to include a `README` file in your project that explains how to build and run your project. Please name this file `README`, please make sure that it is in plaint text format, and please place it in the root (top-level) of your project turn in.

This file must explain *exactly* how to build and run your project. It must also list all libraries that you used[5], including a URL so that we can look them up.

You may include a short writeup in this file describing your general design motivations. Please mention the location and purpose of any test or auxiliary files you have included in your submission.

5 Submitting Your Project

Please submit your project as a `.zip`, `.tar.gz`, `.tar.bz`, or `.tar.xz` to `david.strawn@risksense.com`. Please do not include any compiled code, unless it is an library.

[4] See section 4 about building your project.
[5] Excluding the standard library

6 Questions

If you have any questions or need clarification about anything please feel free to email me at, david.strawn@risksense.com.